NJSZT
MŰSZAKI ÉS TERMÉSZETTUDOMÁNYI EGYESÜLETEK SZÖVETSÉGE
NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG

PROGRAMOZÁSI RENDSZEREK '75

SZEGED

1975. augusztus 25-27

MTESZ

NEUMANN JÁNOS SZÁMITÓGÉPTUDOMÁNYI TÁRSASÁG
valamint az

MTA
MATEMATIKAI ÉS FIZIKAI OSZTÁLYA
által szervezett

PROGRAMOZÁSI RENDSZEREK'75

konferencia előadásai

1975 augusztus 25-27
SZEGED

TARTALOMJEGYZÉK

Oldal:

Oldal:

VI. Alkalmazási programok, programrendszerek

A CHANGE NYELV SZEMLÉLETE ÉS IMPLEMENTÁLÁSA A CDC 3300 SZÁMITÓGÉPEN

Dr. Legendi Tamás

MTA Matematikai logikai és Automataelméleti Tanszéki Kutató Csoport

A szerző által tervezett CHANGE nyelv szemlélete alapvetően dinamikus:

a; a fő utasitáscsoportok nagyfoku hozzáférést biztositanak a CHANGE multiprocesszor definiált részeihez

b; ujszerü számitógépes modellek fogalmazhatók meg, mivel változó mennyiségeknek processzorok és utasitások is megfeleltethetőek /nemcsak adatok/.

Az implementálás egy kb. 130 utasitást tartalmazó subsetig jutott el. Ez tartalmazza a főbb utasitáscsoportokat a kiterjesztő utasitások kivételével /az előforditó programok szintaxis-vezéreltek, igy csak az értelmező rendszer rutinjai hiányoznak/

1. A CHANGE nyelv szemlélete

A CHANGE nyelv kialakitásához a hagyományos nyelvek kritikai szemlélete vezetett el. A közepes illetve magasszintü nyelveknél ugyanis nagyrészt elveszett az a processzororientált szemlélet, hogy a programozó közvetlenül egy számitógépet vezérelt a programjával, amelynek teljes hozzáférése van a számitógép minden "látható" /a programozó számára egységként kezelhető/ eleméhez. Igy például a regiszterek, a programot tartalmazó memóriaterületek általában nem érhetők el, a vezérlés irányitása korlátozott mértékü, az utasitás- és adattipus készlet kötött, a forrásnyelvi szintü nyomkövetés szegényes.

Az /1/ dolgozat javaslatot tesz egy virtuális processzormodellre, és egy, ennek a programozására szolgáló közepes szintü nyelvre. Ez a modell a CHANGE multiprocesszor, a-

mely párhuzamos müködésü, szinkronizált processzorokból áll. Minden egyes processzor utasitásszámlálóval, utasitásszámláló-módositóval /ez inkrementálja lépésenként az utasitásszámlálót/ és kiterjeszthető-szükithető utasitáskészlettel rendelkezik. A processzorok alá- és mellérendeltségi viszonyban lehetnek egymással.

A multi-processzor programozására szolgáló CHANGE nyelv főbb utasitáscsoportjai lehetővé teszik a végrehajtási mód vezérlését, az adattipus és utasitáskészlet bővitését ill. szükitését, futás közben a végrehajtható programba /fordítás nélkül/ bekerülő utasitások generálását és széleskörü forrásnyelvi szintü nyomkövetést is biztositanak.

A végrehajtási mód programozása mellé- ill. alárendelt processzorok inditásával, processzorok felfüggesztésével és megszüntetésével, utasitásszámlálók és -módositók értékének beállitásával, stb. történhet.

Az /5/ összefoglaló cikk terminológiája szerint a nyelv önkiterjesztő; uj utasitások és adattipusok szintaktikája és szemantikája definiálható, utasitások és adattipusok törölhetőek. Különböző processzorok utasitáskészlete különböző lehet, azonos szintaktikáju utasitásokhoz különböző processzorokban más-más szemantika tartozhat.

A programgenerálást /"önmódositást"/ biztositó utasitáscsoport lehetővé teszi tetszőleges program futás alatti generálását.

A nyomkövető utasitások a sokféle forrásnyelvi szintü nyomkövetés mellett utasitásközi müveletek programozására, igy tulajdonképpen végrehajtási mód vezérlésére is alkalmazhatóak.

A /2/ dolgozat ismerteti a nyelv tervezett főbb alkalmazá-

si területeit, részletesen tárgyalja az egyes utasitáscsoportok történeti előzményeit és kialakitásuk elveit, az utasitáscsoportok összefüggőségét /a nyelv egységességének bizonyitására/ példaprogramokkal is alátámasztja.

A nyelv főbb alkalmazási területei:
a; célorientált nyelvek processzorainak előállitása /kiterjesztéssel/.

A fő szempont az, hogy a kiterjesztéssel történő processzorirás kevésbé munkaigényes és rugalmasabb a forditásos és/vagy interpretativ feldolgozásnál.

Ezenkivül erre a célra a nyelv ereje különösen akkor használható ki, ha a kiterjesztett utasitások szemantikáját célszerü párhuzamos algoritmussal fogalmazni meg. Végül szemléleti különbség a hagyományos feldolgozással szemben /Pl. NC nyelveknél/, hogy a célorientált nyelv adatstruktúrára való forditása és az adatstruktúrán történő müveletvégzések helyett CHANGE-ben a természetesebb ut választható: a kivánt müveletek a program végrehajtásával és a programstrukturán /utasitásokon/ végrehajtott müveletekkel valósithatók meg /6/.

b; diszkrét rendszerek szimulálása
Párhuzamos folyamatok leirása természetesen történhet /1/. Aktiv komponenseknek processzorok is megfeleltethetők,
pl. ég-e a lámpa - változó
kapcsoló - utasitás /amely a fenti változó értékét beállitja/
kéz - processzor /amely a "kapcsoló" utasitásra rálépve szimulálja a kapcsoló lenyomását/
A nyelv minőségi továbblépést nyujtó tulajdonsága éppen a fenti primitiv példa által is mutatott lehetőség, amely a párhuzamos algoritmusiráson /pl. /2/ mátrix szorzás,

szinusz-koszinusz rutinok/ tulmutat:
változó mennyiségeknek processzorok és utasitások is megfeleltethetőek.

Például hagyományos szabványfeladat a szimbólumtáblából való visszakeresés. Ez CHANGE-ben programozható ugy is /2/, hogy a szimbólumtáblából generálunk egy programot /statikus tábla esetén természetesen csak egyszer, de változás esetén a dinamikus generálás is lehetséges/. A visszakeresés ennek a programnak a végrehajtásával fog történni ugy, hogy egy felismerendő szimbólum /karaktersorozat/ beérkezésekor annak minden karaktere indit egy-egy processzort és a szimbólumtábla minden egyes szimbóluma is indit egy-egy processzort. /Ezek a szimbólumtábla alapján generált programot hajtják végre/. A program müködésének eredményeként /a szimbólumtábla ill. a beérkezett szimbólum méreteitől függetlenül/ 4 végrehajtási lépés mulva csak a beérkezett karaktersorozattal megegyező szimbólumnak megfelelő processzor marad müködő állapotban, /és ekkor sorszámot, szimbólumtáblabeli választ ad vissza, vagy előre meghatározott akciót hajt végre/.

2. A CHANGE NYELV IMPLEMENTÁLÁSA A CDC 3300 számitógépen.

Az /1/-ben példaként részletesen definiált közepes szintü CHANGE nyelv leirásakor már elve az implementációs szándék szabta meg a nyelvi szintet: sem magasszintü gépi kód, sem magasszintü programozási nyelv implementálása nem látszott elérhetőnek az elvek alátámasztására.

A nyelv "tul önálló", azaz főbb utasitáscsoportjai tul speciálisak ahhoz, hogy valamilyen hagyományos nyelv /pl. preprocesszálással történő/ kiterjesztésével implementáljuk.

További nehézség, hogy a programgenerálás és a programozható végrehajtási mód eleve szükségessé teszi, hogy a leforditott programban az eredeti forrásnyelvi utasitásoknak megfelelő egységek hozzáférhetőek legyenek. Igy az inkrementális forditási technika és az interpretálás állt rendelkezésre. Az egyszerűbb megoldást, a belső adatstrukturára /CHANGE belső assembly nyelvre/ való előforditást és ennek a belső assembly nyelvnek az interpretálását választottuk.

A belső adatstruktúrát ugy alakitottuk ki, hogy a nyelv jellemző utasitásainak végrehajtását gyorsitsuk és az ismétlődő paraméterek illetve utasitások csak egyszer kerüljenek tárolásra /3/, /4/.

Két előforditó program készült:
Az egyik egy egyszerüsitett /lényegében assembly szintü/ bemenő formátumu

FORTRAN IV bázisu forditó /Ehhez MP/O makrokkal egy előfeldolgozó passz készül, amely CHANGE-ről erre a - külső - assembly szintre fordit/. A másik előforditóprogram egy CDL nyelvü forditó, amely kisebb megkötésekkel /pl. a tipusdeklaráció statikus, CHANGE-ről belső assembly nyelvre fordit.

Mindkét forditó szintaxisvezérelt, a felismerendő CHANGE utasitások szintaktikáját /a kiterjesztő utasitásokban szereplő szintaktikai leirásnak megfelelő formátumu/ táblázatból kapják.

A végrehajtó program FORTRAN IV nyelvü, három részre tagolódik:
MULTI-0: A multiprocesszor iniciálását végzi
MULTI-1: Az egyes végrehajtási lépéseket végzi el
MULTI-2: Az egyes végrehajtási lépéseket lezárja.

A MULTI-1 programrész a müködő processzorok láncáról pro-

cesszor sorszámokat ad át a hagyományos végrehajtó programnak /EXEC1/, amely egy processzor egy aktuális /utasításszámlálójának tartalma által előirt/ utasitását hajtja végre a cimszámitó rutin és a /nyilt valamint zárt rutinokból álló/ könyvtár segitségével. A MULTI-2 programrészre azért van szükség, mert a végrehajtási módot vezérlő utasitások a processzorstruktura módositását /értelem és definicioszerüen/ csak a következő végrehajtási lépésre irják elő. Igy a végrehajtási lépés közben meg kell jegyezni /fel kell listázni/ ezeket az utasitásokat és tényleges végrehajtásuk csak a lépés végén történhet meg.

Jelenleg egy subset áll rendelkezésre, amely I/O, értékadó, vezérlésátadó, utasitásgeneráló, végrehajtási módot előiró, kisszámu nyomkövető és speciális utasitásból áll.

IRODALOM:

Legendi Tamás:

/1/ A CHANGE nyelv. multiprocesszor.
SZTAKI Tanulmányok 1973/7.

/2/ Legendi Tamás:
A CHANGE nyelv szemlélete, implementálása és alkalmazása. Kézirat /1974/

/3/ Meggyesi K.:
A CHANGE nyelv implementálása a MINSZ-32 számítógépen
SZTAKI Tanulmányok 1973/12

/4/ Karvaly G.:
A CHANGE nyelv implementálása a CDC-3300 számítógépen.
Diplomamunka, JATE 1975.
CHANGE 1.0 Felhasználói kézikönyv
CHANGE 1.0 Karbantartói kézikönyv

/5/ A Survey of extensible programming languages
N. Solntseff, A. Yezersky
Mc. Master University, Hamilton, Ontario. Computer Science Technical Report 71/7

/6/ Legendi T.:
The use of the CHANGE language for NC and CAD languages PROLAMAT'73 IFIP konferencia

Az OS/i-RTS/i, a TPA/i kisszámitógép uj operációs rendszere

Szabó András
MTA Központi Fizikai Kutató Intézet

BEVEZETÉS

A KFKI TPA/i kisszámitógépe elsősorban müszaki-tudományos számitások mérés-adatgyüjtési, valamint a real-time és folyamatellenőrzés területén kerül felhasználásra.

A most bemutatásra kerülő operációs rendszerek /Operating System, OS/i és Real-Time System, RTS/i/ is ezeken a területeken nyujtanak uj lehetőségeket.

AZ OS/i OPERÁCIÓS RENDSZER

1. Perifériafüggetlenség

Több perifériát tartalmazó gépi konfiguráció esetében felmerül egy olyan software igénye, amely az aktuális feladatok szempontjából legalkalmasabb perifériákat kezeli, és illeszkedik változó, bővülő periféria konfigurációhoz. Az OS/i operációs rendszer és rendszerprogramjai ilyen teljeskörü perifériafüggetlenséget biztositanak. A perifériafüggetlenség megvalósításának alapelve, hogy sem az egyes perifériákkal kapcsolatos input/output tevékenységet, sem a háttértárak szervezését nem közvetlenül az egyes rendszerprogramok végzik, hanem egy-egy rendszerszinten elhelyezett, minden rendszerprogramból hivható software csomag. A rendszer több ilyen csomagot tartalmaz és a csomagok cserélhetők. A perifériafüggetlenség megvalósitása a következő részekből tevődik össze.

1.1 Aktiv perifériák generálása

Az OS/i táblázatokat tartalmaz, amelyek nyilvántartják az aktuálisan használható perifériák logikai neveit, hardware tulajdonságait, valamint az adott periféria kezelését, illetve szervezését végrehajtó csomag címét.

Ezeket a táblázatokat a BUILD rendszerprogrammal lehet egyszerü konzol utasitásokkal generálni. Az aktiv perifériák száma 15 lehet. A BUILD segitségével használaton kivüli perifériákat törölni lehet, uj perifériákat lehet aktiválni és ezekhez neveket rendelni. Háttértárok esetén a kezelés vagy szervezés tipusát lehet beállitani vagy megváltoztatni. Igy például irásra, olvasásra engedélyezni vagy letiltani csak egy file létrehozását engedélyezni vagy több file nyilvántartásához szükséges adminisztráció végrehajtását beállitani és a fileszervezés módját kijelölni vagy a háttértáron használható területet specifikálni.

1.2 Perifériák kijelölése egy rendszerprogram végrehajtásához

Az OS/i rendszerprogramok beinditásuk után a Command Decoder elnevezésü segédprogramon keresztül információkat kérnek be arról, hogy melyik perifériáról/fileból kérjék a bemenő- és hová küldjék a kimenő adatokat. Interaktiv üzemmódban a felhasználó a konzol segitségével specifikálja a legalkalmasabb prifériákat, BATCH üzemmódban az információkat egy, a programok fölé rendelt BATCH monitor adja át.

1.3 A programok periféria kezelése

A rendszerprogramok az információk alapján behivják a megfelelő perifériakezelő rutint /handlert/ és adatátvitelt kérnek. A handler a programtól függetlenül hajtja végre a perifériaspecifikus müveletet és egy minden rendszerprogram által kezelhető standard formátumu input buffert hoz létre. Az input adatok feldolgozása ezután perifériafüggetlenül, programspecifikusan folytatódik. Szükség esetén a program létrehoz egy standard formátumu output buffert és adatkivitelt kér a a specifikált output perifériát kezelő handlertől.

1.4 A programok háttértár szervezése

A fileszervezés hasonló módon történik. A rendszerszinten elhelyezett háttérszervező programcsomag a következő szolgáltatásokat nyujtja:

a/ megkeres egy periférián egy már létrehozott file-t /OPEN INPUT/
b/ uj output file-t hoz létre /OPEN OUTPUT/
c/ egy kész output filet lezár /CLOSE OUTPUT/
d/ file-t töröl /DELETE FILE/

A szervező programcsomag figyelembe veszi a periféria tipusát /karakterorientált, file szervezésü/, illetve a periféria file-szerkezetét. Igy lehetőség van arra, hogy a rendszerprogramok a perifériákat egyformán /nem periféria specifikusan/ kezeljék, de a perifériákon periféria specifikus müveletek és a különböző file szerkezetek kezelése történjék.

2. CHAIN

Az OS/i láncolási /CHAIN/ opciója egyrészt lehetőséget nyujt olyan programok futtatására, amelyeknek méretei meghaladják a memóriakapacitást, másrészt csökkenti a program végrehajtáshoz szükséges beavatkozások számát. A láncolási opció beépithető közvetlenül az egyes programokba egy összetartozó programcsomag létrehozása céljából, vagy felhasználható egyébként függetlenül is használható programok konzol utasitással történő összekapcsolására.

2.1 A láncolás megvalósitása

A láncolás egy rendszerszintü szolgáltatás, amelyet a programból lehet kérni. A kérés hatására az operációs rendszer betölti a megjelölt programot, amelyhez a láncolás történik és elinditja. A következő program számára szükséges információkat az előző program az operációs rendszeren belül meghatározott helye, standard /minden rendszerprogram által érthető/ formátumban tárolja. Ha a következő program láncolással /közvetlen operátori beavatkozás nélkül/ indul el, ugy a munkáját vezérlő információkat, igy pl. a használandó perifériák és file-ok megjelölését, itt találja meg. Ha ellenben a program a felhasználó közvetlen kérésére indul el, a Command Decoder segédprogram

tölti ki ezt a területet a felhasználó információi alapján. Igy lehetővé válik az OS/i rendszerprogramjainak használata felhasználói vezérléssel vagy külső beavatkozás nélkül. A láncolási opció használatára néhány példa:

2.2 Rendszerprogramok egymáshoz láncolása

A gépi kódban a diszken rendelkezésre álló programok egymáshoz láncolása vagy feltétlenül /fix programcsomagok/, vagy felhasználó által adott opció hatására történik. Az első esetre példa az OS/i FORTRAN IV. Compiler program, amely négy, egyenként 8 K szó memóriaigényt nem meghaladó részből áll. Az opció segitségével egyrészt a compiler futtatása már egy 8 K szó memóriával rendelkező gépen is lehetséges, másrészt csak az első rész behivása történik felhasználói utasitással és a következők láncolással indulnak el. Az opcionális láncolásra példa a MINIBOL /mini COBOL/ rendszer, amely a Compiler és az Executive programokból áll. A két programot lehet külön-külön is használni forrásnyelvü MINIBOL program leforditására, illetve egy már leforditott program futtatására, de lehet a Compiler inditásakor láncolást is specifikálni, amely hatására a leforditott program további felhasználói beavatkozás nélkül kerül végrehajtásra.

2.3 Magas szintü nyelven irt programok láncolása

A CHAIN opció kihasználásával összetettebb láncolások is lehetségesek. A magas szintü nyelvek executiv, illetve run-time rendszerprogramjainak önmagukhoz láncolása segitségével lehetséges olyan magas szintü nyelven irt programok futtatása, amelyek helyigénye meghaladja a rendelkezésre álló memóriakapacitást. A hosszu programokat fel lehet osztani szegmensekre és ezeket egymás után futtatni. Az információ átadás a háttértárokon létrehozott file-ok segitségével lehetséges. A láncolás a különböző magas szintü nyelvekben különböző módon van megvalósitva. A FORTRAN II. rendszerben a láncolást a programból egy könyvtári rutin hivásával lehet kérni. A MINIBOL rendszerben a programrészek számát és egymásutánját az első program futtatása előtt lehet specifikálni. Igy lehetséges az egyes részek külön-külön kipróbálása, vagy egymás utáni futtatása a programok megvál-

toztatása nélkül. A BASIC rendszerben a rendszerprogramok zárt láncolását lehet kérni, igy egy program a BASIC editor segitségével történt megirása után a futtatási utasitás hatására előbb a BASIC Compiler és a Loader, majd a Run-Time-System kerül sorra, végül a program lefutása után a vezérlés automatikusan visszatér a BASIC Editor programhoz.

3. BATCH feldolgozás

Az OS/i operációs rendszerben rendelkezésre álló BATCH monitor segitségével lehetséges bonyolult programcsomagokat operátori felügyelet nélkül futtatni, valamint ismértlődő munkák végrehajtását a felhasználói beavatkozások csökkentésével gyorsitani.

3.1 Egy BATCH csomag futtatásának előkészitése

Egy BATCH csomag futtatásához egy, a BATCH monitor inputjául szolgáló file-t kell összeállitani, amely tartalmazza a programok futtatásához szükséges standard OS/i utasitások sorozatát, valamint az egyes programok végrehajtásához szükséges input/output specifikációkat és programvezérlő opciókat.

Egy BATCH csomag tartalmazhat OS/i rendszerprogramokat és/vagy a felhasználó által készitett leforditott és a diszken tárolt programokat. A programok előkészitést nem igényelnek.

3.2 A BATCH monitor müködésének elve

A BATCH monitor elinditása után a felhasználó specifikálja, hogy melyik BATCH csomag kerül végrehajtásra, illetve melyik file-ban van a végrehajtandó BATCH csomag specifikálva. A BATCH monitor ennek a filenak alapján egymás után behivja és elinditja az ott felsorolt programokat.

A programok a müködésükhöz szükséges információkat /input, output, opciók/ - amelyeket interaktiv üzemmódban a Command Decoder felhasználásával a konzolon keresztül kérnek be - nem a konzolról, hanem a feldolgozást meghatározott file-ból kapják a BATCH monitor segitségével.

Ha a programcsomag lefutása közben az operátor jelen van, ugy olyan programokat is tartalmazhat a BATCH csomag, amely kézi beavatkozást /papir szalag betöltést/ igényel. Ez esetben a BATCH konzolon jelez, és várakozik a teendő elvégzését nyugtázó jelre. A BATCH csomag végrehajtását specifikáló file-ba beépithetők további, az operátornak szóló utasitások, illetve az ezek végrehajtását nyugtázó jelre való várakozás.

A BATCH figyeli a programok futását, esetleg további információ kérést szolgál ki, valamint észleli a program befejezését és inditja a következő programot.

A BATCH a programcsomag futásáról naplót készit, amely tartalmazza a végrehajtott programok nevét, müködésük specifikációját, és a hibajelzéseket.

3.3 Spooling

A program végrehajtás gyorsitása érdekében, vagy ha a programcsomag végrehajtásakor nincs jelen output perifériákat kiszolgáló operátor, használható a BATCH spooling opciója. Ekkor az egyébként lassu karakterorientált output periférián /papirszalag lyukasztó, sornyomtató/ kiadandó adatok egy kijelölt gyorsabb manuális beavatkozást nem igénylő háttértárra kerülnek /pl. mágneslemezre/, ahonnan alkalmas időpontban kiadhatók. Az opció végrehajtása ugy történik, hogy a programok a karakter-orientált periféria kezdő rutin bekérésekor a rendszertől nem ezt, hanem a megfelelő helyettesitő háttértár kezelő programot kapja meg.

4. Az OS/i programkönyvtára

Az OS/i operációs rendszer programjai elsősorban az assembler programok elkészitését, a magas szintü nyelvü programok irását és futtatását, valamint az egyszerü rendszerkezelést segitik.

4.1 Assembler nyelvü programok fejlesztése

Az OS/i rendszerben rendelkezésre állnak programok forrásnyelvü programok irására és javitására /Symbolic Editor, Text Corrector/,összehasonlitására /Source Compare/, forditására /Slang1, Slang3, Slang4/ és a forditás listázására /Cross-Reference Program/. A leforditott abszolut vagy relokálható bináris program betöltésére az Absolut Loader és a Linking Loader, a programok helyfoglalásának ellenőrzésére a Bitmap és belövésére az Octal Debugging Technique programok szolgálnak.

4.2 Magas szintü nyelvek

Programcsomagok állnak rendelkezésre FORTRAN II. nyelven irt programok kezelésére /Compiler, Assembler, Linking Loader, Library, Library készitő/, FORTRAN IV. programokhoz /Compiler, Assembler, Loader, Library, Library készitő, Run-Time-System/, BASIC nyelvü programok elkészitéséhez /BASIC Editor, Compiler, Loader, Run-Time System beépitett függvénykönyvtárral/, adatfeldolgozási feladatok megoldásához /mini Cobol Compiler és Executive/, valamint a müszaki-tudományos számitások párbeszédes üzemmódu végrehajtására szolgáló FOKAL interpreter.

4.3 Utility programok

A felhasználó munkáját több segédprogram gyorsitja, mint például másolási /Peripher Interchange Program, File Oriented Transfer Program/, rendszer nyilvántartási /Resorces/ és rendszer initializálói /Bootstrap/ programok.

AZ RTS/i REAL TIME RENDSZER

5. Az OS/i real-time rendszere /RTS/i/

Az RTS/i egy olyan software eszköz, amely képes külső eseményekre reagálni, munkáját real-time események vezérelhetik. Képes egymástól független job-okat /taskokat/ egymás mellett, prioritásuk szerint futtatni, és az input/output perifériákat egyidejüleg kiszolgálni.

Az RTS/i rendszert mérés adatgyüjtési feladat elvégzésére, ipari folyamatok ellenőrzésére, szabályozására lehet felhasználni.

5.1 Az RTS/i felépitése

Az RTS/i egy memória-rezidens moduláris rendszer, amely egyrészt maximum 63, egymástól független, fix prioritásu job-ot /taskot/, másrészt egy monitort tartalmaz, amely a taskok futtatását, egymás közti kommunikációját és a program megszakitás kezelését végzi. A rendszer maximális müködési sebességeérdekében a taskok assembler nyelven kerültek megirásra.

5.2 Az RTS/i monitor

Az RTS/i monitor egy 1K szónál kisebb helyfoglalásu memória-rezidens program, amely a rendszer real-time funkcióit vezérli. Nyilvántartja a rendszerhez tartozó taskokat és ezeknek az állapotát. A futtatható taskok közül mindig a legmagasabb prioritásut müködteti. A monitor bonyolitja le a taskok közötti jel- és adatcserét. Felfüggeszti egy task futtatását, ha ez egy jelre vagy adatokra várakozik /EVENT WAIT, MESSAGE WAIT/, elfogadja egy esemény bekövetkezésének jelzését /POST EVENT/, prioritás szerint átadja egy másik tasknak küldött adatokat /SEND MESSAGE, RECEIVE MESSAGE/ és ujraindit taskokat, amelyek erre várakoztak.

5.3 Órakezelő task

Az órakezelő tasktól az egyes taskok kérhetik saját maguk vagy más taskok leállitását vagy ujrainditását meghatározott időpontban, egy adott idő eltelte után vagy periodikusan megadott ciklusidővel. Ezenkivül kérhetik eseményjelek /EVENT FLAG/ kiadását szintén meghatározott időpontban, egy adott idő elteltével vagy periodikusan.

5.4 Az RTS/i interaktiv vezérlése

Egy konkrét feladatot végrehajtó rendszer belövéséhez és vezérléséhez a Monitor Consol Task ad lehetőséget. A task segitségével informálódhat a felhasználó a rendszer taskjainak állapotáról, és ezeket u-

tasitással leállithatja vagy elindithatja. Hatékony segitséget nyujtanak egy real-time rendszer belövéséhez azok az utasitások, amelyekkel események bekövetkezését lehet szimuláltatni és a taskokat órajelekre lehet futtatni.

5.5 Periféria kezelő taskok

A perifériák kezelése az RTS/i-ben interruptosan történik. Minden perifériának saját taskja van, és a rendszer további periféria taskokkal bővithető. A perifériákat tartósan vagy átmenetileg hozzá lehet rendelni egyes feldolgozó taskokhoz /ASSIGN/, ilyenkor az adott perifériát csak ez a task tudja használni. Az input perifériákat kezelő taskok az elfogadott adatokat a SEND MESSAGE monitor szo gáltatással küldik a feldolgozó taskoknak, vagy egy EVENT FLAG beál litásával jelzik az adat beérkezését. Az output perifériákat kezelő taskok MESSAGE WAIT állapotban várnak output adatokra és csak akkor lépnek müködésbe, ha valamelyik task kiadandó adatot küld nekik, ezenkivül EVENT FLAG hatására meghatározott jelzéseket, információkat küldenek ki. Egy rendszerben több konzol /és hozzátartozó konzo task/ is lehet, ezeket a rendszer egymástól függetlenül is tudja ke zelni. Továbbá rendelkezésre állnak taskok a gyorsolvasó, lyukasztó sornyomtató és a mágneslemez kezelésére.

5.6 Hardware hibavédelem

Az RTS/i rendszerben külön taskok figyelnek egyes meghibásodásokra. A Power fail task a hálózat kimaradása esetén lép müködésbe, leálli ja a vezérlési funkciókat, és jelzést ad ki, majd a hálózat visszatérésekor ujra inicializálja a taskokat és a perifériákat. Taskok figyelik a perifériák müködését is, és ezek meghibásodása esetén le tiltják az input funkciókat, illetve átirányitják az output funkció kat más output perifériákra. A diszket kezelő task kérésre ellenőrz az irás vagy olvasási müveletet és kiküszöböli a tranziens hibákat.

5.7 Az OS/i rendszer müködtetése az RTS/i alatt

Az RTS/i és az OS/i rendszerek képesek együtt is müködni, és igy eg rendszerben összpontosul a real-time feladatok elvégzésének lehető-

sége és a háttérfeldolgozási vagy programfejlesztési problémák megoldása. Az OS/i rendszer müködését az RTS/i rendszerben szintén egy task vezérli. Segitségével az OS/i az RTS/i legalacsonyabb prioritásu taskjaként, tehát háttérben müködik. Az OS/i task segitségével lehetséges az előtérben real-time módon összegyüjtött adatok feldolgozása egy OS/i rendszerprogrammal, magas szintü nyelvvel vagy egy BATCH programcsomaggal. Másik lehetőség az RTS/i-től független programfejlesztési munka elvégzése az RTS/i kihasználatlan gépidejében.

ÖSSZEFOGLALÁS

Megitélésünk szerint az OS/i és az RTS/i operációs rendszerek jelentős lépést jelentenek a kisszámitógépek alkalmazási szinvonalának emelésében, a real-time mérés-, adatgyüjtési és adatfeldolgozási feladatok fejlett, rugalmas megvalósithatósága következtében.

A COS/i ADATFELDOLGOZÁSI PROGRAMRENDSZER

Kóta Gábor

KFKI-MSzKI

1. BEVEZETÉS

A hagyományos könyvelőgépek és a nagyobb számítógépes adatfeldolgozó rendszerek közötti hézag betöltésére az elmult években több irányból történtek erőfeszitések. A könyvelőgépeket továbbfejlesztették, mágneslemezzel és távadatátviteli lehetőségekkel szerelték fel, kialakitották a terminálos rendszereket, de ezek mellett a feladat-orientált minikomputerek is erőteljesen behatoltak erre a területre. A kisszámitógépekkel kialakitott rendszerek a tároló eszközök alkalmazásában és a nyujtott programozási lehetőségekben erősen eltérhetnek egymástól. Áruk az 5-75 ezer $ tartományba esik, és nyugaton egyrészt a 20-500 alkalmazottat foglalkoztató cégek adatfeldolgozási és irányitási feladatait, másrészt a nagy intézmények egyes részlegeiben folyó decentralizált adatfeldolgozást kivánják ezekkel megoldani. [1], [2]

1972-ig összesen 400 millió $ értékben adtak el adatfeldolgozó kisszámitógépes rendszereket, ez a szám 1975 végéig előreláthatóan eléri az 1,4 milliárd $-t [3].

Ezek a kisszámitógépes rendszerek három, nagyjából azonos volumenü kategóriába sorolhatók, ezek a kötegelt adatfeldolgozás, a távadatfeldolgozás és az interaktiv feldolgozás.

A KFKI TPA-i kisszámitógépére kialakitott COS/i adatfeldolgozási programrendszer az interaktiv feldolgozásra irányul. Az interaktiv feldolgozás ugy definiálható, hogy egy vagy több felhasználó a gép mellett interaktiv módon utasitásokat és adatokat visz be, és a feldolgozáshoz a helyi adatbázist használja fel 3 .

2. A COS/i FŐ JELLEMZŐI

A COS/i operációs rendszer a feldolgozás jellegétől és a felhasznált hardware-konfigurációtól függően két nagy részre bontható:

- egyterminálos alaprendszer
- többterminálos adatkezelő programcsomag.

2.1. Az alaprendszer

A COS/i alaprendszer a felhasználó számára egy magasszintü, COBOL-szerü nyelv /MIDIBOL/ használatát és az alapvető file-kezelési feladatok elvégzését teszi lehetővé. A MIDIBOL-programok a Monitorba szervesen beépülő szerkesztő program felhasználásával irhatók és javithatók, a rendszerrel lefordithatók és futtathatók. A fő rendszer- és utility-programok:

- Monitor/Editor
- Rendszergeneráló
- MIDIBOL forditóprogram és futtató rendszer
- Adathordozó - konvertáló
- File-létrehozó
- SORT/MERGE program
- File-karbantartó /UPDATE/
- Riportkészitő

Az alaprendszer az alábbi hardware-konfigurációt tételezi fel:

- 8K /12 bites szó/ operativ memória
- Konzol-irógép v. display
- Mágneslemez-egység
- Sornyomtató.

A rendszer hatékonyságát javitja:

- további 4K memória
- lyukszalagolvasó
- lyukszalaglyukasztó
- mágnesszalagos egységek használata.

2.2. A többterminálos adatkezelő programcsomag

A többterminálos adatkezelő programcsomag 1-6 terminálról való egyidejü adatbevitelt és lekérdezést tesz lehetővé, miközben háttérprogramként a COS/i rendszer tetszőleges utility vagy felhasználói programja futhat. A programcsomagnak az alábbi üzemmódjai vannak:

- Adatfile bevitele
- Adatfile felfrissitése
- Adatfile-hoz való hozzáfüzés
- Lekérdezés.

Külön meg kell még emliteni a többterminálos MIDIBOL-forditóprogramot, amely lehetővé teszi a 6 terminál MIDIBOL-programban való kezelését.

A szükséges hardware-konfiguráció:

- 12-20K operativ memória
- Konzol irógép
- Diszk
- 1-6 Video-terminál

3. A COS/i ALAPRENDSZER MÜKÖDÉSI ELVEI

3.1. File-ok

A COS/i rendszer négyféle file-tipust kezel:

- Rendszerfile-ok: a rendszer és utility programok
- Forrásfile-ok : a MIDIBOL-nyelven célszerüen az editorral irt programok
- Compiler-bináris file-ok : a forditóprogram által leforditott forrásprogram bináris kódja /interpretativ végrehajtás/
- Adatfile-ok

Ezek közül az első három tipus felhasználhatóságban és tárolási strukturában eltérő, közös bennük azonban az, hogy valamennyien az u.n. rendszerperiférián /a diszk egy kijelölt részén/ kerülnek tárolásra és közös nyilvántartásuk /directory-juk/ van. A rendszerfile-ok fixen beépitettek, a forrásnyelvü és bináris file-ok Monitor-parancsokkal a diszken elhelyezkedő szerkesztési, illetve bináris munkaterületről a rendszerperifériára felvihetők vagy onnan kitörölhetők. Az

adatfile-ok nem vihetők a rendszerperifériára, hanem ugyancsak a diszken vagy a mágnesszalagon kijelölt u.n. logikai egységeken helyezhetők el.

Említettük a diszk-terület kétféle felhasználását, a területek kijelölése a rendszergeneráló programmal végezhető el. 48K szó területet ajánlatos a diszk alsó területén a rendszerprogramok, forrásnyelvü és bináris programok számára fenntartani, ez az u.n. rendszerperiféria. A diszk többi részét 4K szónyi szegmensekben egy-egy adatfile tárolására alkalmas logikai egységre lehet szétbontani /max. 15 db/. Egy mágnesszalagos egység ugyancsak egy logikai egység lehet. A logikai egységek és a többi periféria közötti átviteleket az adathordozó-konverziós rendszerprogram végzi el.

Az adattárolás fizikai egységei:

1 szó = 2 karakter /6 bites ASCII/
1 blokk = 256 szó /csak fizikai egységet jelent/
1 szegmens = 16 blokk

3.2. A MIDIBOL nyelv

A MIDIBOL-nyelv adatfeldolgozásra orientált magasszintü nyelv. A COBOL-tól a szükitésen kivül tömörebb jelölésrendszerében is eltér.

Minden MIDIBOL-program Adatleiró Főrészből és Eljárási Főrészből áll. A következőkben az adatok szerkezetét és kezelését, majd az Eljárási Főrész utasitásait foglaljuk össze.

.2.1. Az adatok leirása, szerkezete és kezelése

Az Adatleiró Fórész a programban felhasznált valamennyi adatterület tipusát és terjedelmét irja le. A leirás rekordokra és ezen belül elemi adatokra /mezőkre/ vonatkozik.

A rekordterületeket akkor kell névvel ellátni, ha arra egy átviteli utasitás hivatkozik. Ekkor maximális mérete 510 karakter lehet. A rekord egy vagy több mezőből áll, amelyekre az Eljárási Fórész utasitásai hivatkozhatnak. Egy rekord tetszőleges számban ujradefiniálható.

Kétféle tipusu mező előirása lehetséges: alfanumerikus vagy decimális. A mezőkhöz kezdeti érték rendelhető, ennek megadása nélkül azok üres vagy zérus karakterekkel indulnak. Előirható futás közbeni konzolról való egyszeri értékmegadás is.

Lehetőség van egydimenziós tömbök definiálására, tömbön belüli mezők vagy egy mezőn belüli karakterek indexelt felhasználására.

Az adatkezelő utasitások olyan értékadó utasitások, amelyeknél az egyenlőségjel baloldalán egy alfanumerikus vagy decimális változó, jobboldalán egy alfanumerikus változó vagy egy decimális kifejezés szerepel. Ha a két oldalon különböző tipusu változók vannak, az adatmozgatás egyuttal konverzióval jár együtt. Eltérő hosszuságu változók esetén a futtató rendszer az alfanumerikus tipusuakat balra igazitja és a jobboldalon szóköz karakterekkel tölti fel, a decimális tipusuakat jobbra igazitja és

a baloldalon zérusokkal tölti fel. Kiíráshoz lehetőség van szerkesztett decimális mezők definiálására is, amelyben megadható előjel kiirása, a zérus karakterek elnyomása vagy csillag karakterekkel történő helyettesitése, vesszőknek és tizedespontoknak az elhelyezése.

3.2.2. Perifériás utasitások

A perifériás utasitások vonatkozhatnak mezők terminálon való megjelenitésére, illetve megadására /DISPLAY, ACCEPT/, vagy pedig rekordok átvitelére.

A rekordok átvitelének két módja lehetséges. A soros hozzáférésü XMIT utasitás kiadása tetszőleges perifériára lehetséges és ekkor a rekordhosszuságok esetleges eltérésének kijavitása is automatikusan megtörténik. A közvetlen hozzáférésü READ és WRITE utasitás a diszken már létrehozott file-ok esetében alkalmazható. Ilyenkor a file-nak azonos méretü és szerkezetü rekordokból kell állnia, az átvitelre kerülő rekord megadása a rekord sorszámát megadó decimális kifejezéssel történik. Mindkét tipusu átvitel az u.n. csatornaszámok kijelölésével, majd a csatornaszámra való hivatkozással történik. Egyidejüleg 8 csatorna lehet megnyitott állapotban. Adatfile-ok kijelölése név szerint, a logikai egység opcionális megadásával lehetséges. Ha nem adunk meg logikai egységet, vagy a megadott egységen nem található a megadott nevü file, a rendszer futás közben MOUNT filenév üzenettel érdeklődik a logikai egység iránt, amely igy konzolról megadható.

Speciális perifériás utasitás a TRAP, amely egy sornyomtató-rutin magasszintü programozásával lehetővé teszi a spooling-ot, a program futásával párhuzamosan adatterületek kinyomtatása hajtható végre.

3.2.3. Programvezérlő utasitások

A vezérlésre vonatkozó utasitások a programnyelvekből jól ismert utasitások, igy ezeknek csak a felsorolására szoritkozunk.

- GO TO : Lehet feltétlen, feltételes, vagy számitott.
- CALL : Egymásbaskatulyázható szubrutinhivó utasitás.
- RETURN : Szubrutinból való visszatérés.
- ON ERROR : Ha a következő utasitásban nemfatális hiba fordul elő, a vezérlés a Monitor helyett a megadott cimkére történik.
- STOP : A program befejezése, visszatérés a Monitorba.
- IF : Egyenlő, nem-egyenlő, kisebb, kisebb-egyenlő, nagyobb, nagyobb-egyenlő relációkkal megadható, a fenti vezérlő vagy pedig a nyomkövető utasitások valamelyike.

3.2.4. Nyomkövető utasitások

A nyomkövetés feltételesen be- vagy kikapcsolható. Nyomkövetésnél a végrehajtott utasitás sorszáma,

értékadás esetén pedig az adott érték is kinyomtatásra kerül a sornyomtatón.

3.2.5. Programok láncolása

Egy MIDIBOL-program maximálisan kb. 400 sorból állhat. Lehetőség van hosszabb program futtatására is, ha azt több, maximálisan 8, programra bontjuk szét. Ezek után bármelyik program a CHAIN utasitással behozhat egy tetszőleges ujabb programot, és annak az elejére adhatja át a vezérlést. A behozott programban elhelyezett CHAIN-nel ujabb, akár korábban már lefutott program indítható.

3.3. A szerkesztő program /editor/

A szerkesztő program a Monitor állandó része. A szerkesztési parancsok a Monitorban bármikor kiadhatók. Ezek a diszken lévő 4Kszó hosszuságu munkaterületre dolgoznak.

A forrásnyelvi programok irását az automatikus sorszámozás gyorsitja meg. Ilyenkor a szerkesztő program a megadott sorszámtól kezdve megadott növekménnyel előre kiadja azokat a sorszámokat, amelyekre a szerkesztési parancsok a későbbiekben hivatkozhatnak. Az elkészült program további felhasználás céljából forrásfile-ként a rendszerperifériára vihető, vagy lyukszalagra is kiadható. Az editor a forrásprogramot konzolról, lyukszalagról vagy a rendszerperiféria egy forrásfile-járól fogadhatja.

3.4. A fordítóprogram

A fordítóprogram a megadott forrásfile/ok/-t vagy ennek hiányában a szerkesztési munkaterületen elhelyezkedő forrásprogramot alakitja olyan bináris kóddá, amelyet a futtató rendszer értelmezni tud, és ezt a diszk bináris munkaterületén helyezi el. A bináris munkaterület tartalma Monitor-paranccsal bináris fileként a rendszerperifériára vihető.

3.5. A BATCH üzemmód

A Monitor valamennyi parancsa, a szerkesztéseket is beleértve, editor-parancsokkal előre egy forrásfileban elhelyezhető, a futtatás ezután erre a file-ra hivatkozó BATCH-paranccsal lehetséges. Az operátornak szóló üzenetek továbbitására szolgál a PLEASE-parancs, aminek kiadása után a Monitor egy válaszkarakter leütéséig várakozik.

3.6. File-kezelő programok

A file-kezelő programok az adatfile-okon végzett alapvető müveleteket végzik el: a létrehozást, rendezést, összefésülést, naprakész állapotra hozást. Ezeknek a programoknak közös jellemzőjük, hogy azonos hosszuságu és szerkezetü rekordokból álló file-okkal tudnak dolgozni. Futtatásukhoz az editorral előzőleg egy vezérlő-programot kell irni, amely az adatrekordok leirását, az input és output file-okat, a szükséges paramétereket /rendezési kulcsok/ és az előirt opciókat tartalmazza. A programok által használt file-oknak valamelyik logikai egységen kell elhelyezkedniük.

3.6.1. File létrehozása

A file egy rekordját u.n. input sorokban kell megadni. Az input sor egy kulcsszóból és több mezőből állhat, amelyeket előírható módon szóköz vagy vessző választhat el egymástól. A kulcsszóval rendelkező sorok bevitele tetszőleges sorrendben történhet, a kulcsszó nélküli sorokat a vezérlő-programban megadott sorrendben kell megadni. A beirandó mezők alfanumerikusak vagy decimálisak lehetnek, rövidebb adat megadásakor a program a konvenciónak megfelelő igazitásokat elvégzi. Mezők kihagyása megengedett, erre az esetre a mező átlépése, egy konstans bevitele vagy hibajelzés irható elő. Minden decimális mezőre határértékellenőrzés, kontrol-összeg képzése, ellenőrző karakter vizsgálata, bármelyik mezőre jelenlét-jelző beállitása specifikálható.

Az adatbevitel történhet teletype-ről, lyukszalagolvasóról vagy a rendszerperiféria egy file-járól. Különös jelentősége van a teletype-ről való beirásnak, ezen belül is a PROMPT üzemmódnak, amelyben a program az esetleges hibákat azonnal kijelzi, az input sor megadása megismételhető, ezután kerül sor a következő input sorra. Ebben az üzemmódban a vezérlő-programban leirt kulcsszavakat a program kiiratja a teletype-on, a felhasználónak csak a megfelelő input sort kell begépelnie.

Egyidejüleg négy output-formátum irható elő, igy egy adatbevitellel akár négy file is létrehozható. Az output rekord mezőire annak átvitel utáni törlése vagy változatlanul hagyása irható elő. Lehetőség van továbbá számláló mezők kialakitására.

.6.2. Rendező-összefésülő program

A rendező program vezérlő-programjában nyolcféle kulcs szerinti növekvő vagy csökkenő értékü rendezés irható elő. A rendelkezésre álló tárolóterületnek megfelelően a munkaterületekként szolgáló logikai egységek számának megadásával a rendezési idő optimalizálható, minimálisan 3 munkaterület szükséges.

.6.3. Karbantartás /UPDATE/

A file-ok karbantartására háromféle parancs szolgál: rekord közbeszurása, törlése vagy megváltoztatása. Ezek a parancsok teletype-ról, szalagolvasóról vagy a rendszerperiféria egy file-járól adhatók ki. A program két menetes. Az első menetben ellenőrzi a vezérlő programot és a felfrissitőparancsokat. Ha ezek hibátlanok, sor kerülhet a második menetre is, amelyben a file felfrissitése megtörténik.

Ha eredményül rendezett file-ot akarunk kapni, az első fázis után a felfrissitő parancsokat a rendező program behivásával rendezni kell.

. TÖBBTERMINÁLOS ADATBEVITELI PROGRAMCSOMAG

A terminálok müködtetése az alábbi fő fázisokban történik:

- Formátum-vezérlő file létrehozása
- Kezdeti dialógus lefolytatása
- Müveletek

4.1. A Formátum-vezérlő file

A Formátum-vezérlő file a képernyőn megjelenitésre kerülő szöveg-sorozatokat és a bevitelre kerülő adatmezőket definiálja. Az egyes adatmezőkre az alábbi opciók irhatók elő:

- az elfogadott karakterek tipusa /tetszőleges, betü vagy szóköz, számjegy vagy pont vagy minusz jel/
- ellenőrző számjegy alkalmazása
- jobbraigazitás a képernyőn
- határértékek ellenőrzése
- összegképzés egy lekérdezhető regiszterben
- autoduplikálás: a mezőt csak egyszer kell bevinni, ezután automatikusan megjelenik mindegyik rekordban
- tul sok karakterre való ellenőrzés
- a mező kiirásakor való törlése
- nem-látható mezők megadása.

A Formátum-vezérlő file-t a Formátum Compiler forditja bináris formára.

4.2. A kezdeti dialógus

A kezdeti dialógus során a rendszer jelszó megadását kérheti az operátortól, ezenkivül a kivánt üzemmód és a Formátum-vezérlő file megadására szolgál.

4.3. Adatbevitel

A formátum megjelenitése után a kurzor az első beirandó mező elejére áll. A rekord beirása és a bevitel helyességének megerősitése után a rekord a Formátum-vezérlő file-ban előirt file-ba kerül.

4.4. Felfrissités

Felfrissitéshez az ernyőn egy keresési kulcsot kell megadni, amely tetszőleges adatmező tetszőleges karaktereiből állhat. A rendszer kikeresi és megjeleniti a keresett rekordokat, ezután a kivánt változtatások a rekordon belül végrehajthatók.

4.5. Hozzáfüzés

Az adatbevitelhez hasonló, azzal a különbséggel, hogy egy létező file végéhez füz hozzá ujabb rekordokat.

4.6. Lekérdezés

A keresési kulcs megadása után a kérdezett rekord megjelenik az ernyőn, de nem változtatható meg. Lineáris vagy bináris keresési mód irható elő.

IRODALOM

[1] Small Business Systems
Computer Weekly, 12. Dec. 1974.

[2] All About Small Accounting Computers.
Feature Report, Datapro Research Corp., 1973.

[3] The Mini Computer: A Place in Both Large and Small Businesses.
Data Management, 1973. September.

[4] COS/i felhasználói kézikönyv /előkészületben/.

[5] Többterminálos adatbeviteli programcsomag.
Kézikönyv. /előkészületben/

OS-10 REAL TIME DISZKES MONITOROK FILE-KEZELŐ RENDSZERREL

Trencséni István
INFELOR Rendszertechnikai Vállalat

1. BEVEZETÉS

Hazánkban az utóbbi években egyre inkább előtérbe került a távadatfeldolgozás. Ez a tény, valamint az, hogy az ESzR gépcsalád legkisebb elemét az Rlo-et gyártjuk, meghatározza feladatunkat: felhasználni az Rlo-et távadatfeldolgozás céljaira is.
Az OS-lo rendszer keretein belül ki kell dolgozni az erre megfelelő eszközöket. Az OS-lo négy komponensből áll. Ezek a következők:

a, vezérlőprogramok /monitorok/
b, fordítóprogramok
c, egyéb programok /utility-k, szerkesztők, könyvtárkezelők.../
d, rendszergeneráló programok.

Ezen komponensek felhasználásával a felhasználók egy un. bővített géphez jutnak.

A fentiekből következik, hogy ezen komponensek /a,b,c,d/ alkalmas kiegészitésével kell a problémát megoldani. Továbbiakban fontossága miatt a monitorral foglalkozunk, ugyanis ezen komponens legbelső szintjének kell lefednie az alapgépet, valamint ezen komponens legkülső szintjére épül az operációs rendszer többi eleme. A monitor két részből tevődik össze:

a., Megszakitáskezelő programok
b., Supervisor.

2. A MEGSZAKITÁSKEZELŐ PROGRAMOK FUNKCIÓI

- fizikai perifériák kezelése /helyi és kihelyezett perifériák/,
- órakezelés,
- kapcsolat az operátorral /pult mint periféria/,
- tápfeszültség védelem /lásd supervisor/.

A fenti funkciókból látszik, hogy a kihelyezett perifériák kezeléséhez implementálni kell az aszinkron és szinkron handlereket és az órakezelés segitségével lehetőséget kell teremteni az időzitésekre.

3. A SUPERVISOR FUNKCIÓI

3.1 A programok és a hardware felügyelete

A programok felügyelete a betöltés, futtatás stb. kérdéseit oldja meg.
A gép hardware-ének felügyelete hardware és software eszközök közös halmazával valósitható meg. Ezen közös halmaz elemei a következők:

- tápfesz. védelem /hardware/ + feszültség kiesés és visszatérés kezelő program /Software/
- eltérülések /hardware/ + M:TRAP supervisor szekció /software/.

3.2 I/O rendszer

Az I/O rendszer a kérelmek sorbaállitásával és lebonyolitásával foglalkozik. Kapcsolatot teremt a felhasználó logikai szintje /operációs cimkék/ és a tényleges fizikai periféria között. Adatátviteli vonalak esetén az I/O rendszer nem foglalkozik sorkezeléssel, hiszen ezt a funkciót egy másik supervisor szekció /M:DTM/ látja el.

3.3 Szolgáltatások a felhasználó számára

A supervisor egy olyan szekcióhalmazt ad a felhasználóknak, amely segitségével tehermentesiti a programozót és a megadott halmaz ügyes használatával a problémáit könnyen megoldhatja.

3.4 Erőforráskezelés

Az erőforrás egy olyan eszköz, melynek birtoklásáért versenyhelyzet alakulhat ki /pl. közös táblák, stb./
Erőforráskezelés alatt azt értjük, hogy a Supervisor lehetőséget nyújt egy erőforrás lefoglalására /request/, egy erőforrás felszabaditására /release/ és egy erőforrás szabad voltának tesztelésére /test and request/. Ezek a szekciók szintén supervisor szekciók segitségével valósulnak meg /M:RQST, M:RLSE, M:TAR/.

3.5 Eseménykezelés

Egy eseményt egy byte ábrázol. Ez ekvivalens egy flaggel, amely a folyamatban lévő /futó/ programot blokkolja az eseményre való "várakozással" /Flag=a byte 0. bitje, a blokkolást kiválthatja maga a program, M:IO, M:DLAY, M:DTM/.
Az esemény deblokkolódik, amikor egy másik program aktiválja az eseményt: a flaget módositja és a program "ready to run" állapotba kerül.
Az eseményt dinamikusan definiálja egy várakozás /M:WLST vagy M:WAIT/ egy byte-on, amely ezt az eseményt ábrázolja. Az aktiválás szintén egy supervisor szekció valósitja meg /M:ACTV/.

3.6 Időzités

Gyakran előfordulhat, hogy egy program egy késleltetésre várakozásba kezd, vagy time-out-ot ad /vonali algoritmusok/. Ezen problémákat az M:DLAY és az M:SDLY supervisor szekció, valamint a real-time órát kezelő megszakitáskezelő program oldja meg.

3.7 Alrendszerkezelés

Az alrendszerkezelés megvalósitja, hogy egy alrendszerként definiált program /a foregroundban helyezkedik el/ kapcsolatot tartson az operátorral, tehát az operátor által vezérelhető legyen /megvalósitása a "%x" operátori parancs és az M:CSOL supervisor szekció segitségével történik/.
Példa: hardware terminálok szimulálása.

3.8 Filekezelés

A diszkes filekezelés a diszk DA /adat/ zónájában elhelyezett felhasználói file-ok kezelését biztositja.
A felhasználó számára megkönnyiti a diszk elérését, azaz elősegiti egyszerüen és gyorsan kezelhető file-ok létrehozását és használatát.

A file-ok a backgroundból és foregroundból egyaránt elérhetőek.
A supervisorban implementált részen kivül /rezidens/ van egy kiegészitő rész /nem rezidens/.
Ezen két rész együttesen oldja meg a filekezelés problémáit.

A file-ok lehetnek blokkositottak vagy blokkositatlanok /file szervezés/, illetve direkt és szekvenciális elérésüek.
A filekezelés rezidens részét az M:FILE, M:ASGN, M:OPEN, M:IO, M:CLOS supervisor szekciók valósitják meg, melyek lehetőséget nyújtanak dinamikus filekezelésre /CSV M:.../ illetve statikus /%.../ filekezelésre.

4. A fejlesztés iránya

Látható, hogy az emlitett monitor szolgáltatások lehetővé teszik az adatátvitel további eszközeinek a kialakitását.

A monitor fejlesztési iránya egy multitask monitor felé halad, melyben az adatátviteli vezérlők megvalósitása önálló programokkal történhet. Továbbá látszik, hogy filekezelést ki kell egészíteni az indexelt szekvenciális elérési móddal.

R-10 BÁZISU TÖBB DISPLAY-ES REAL-TIME INFORMÁCIÓS RENDSZER

Homitzky Lajos Szabó József

Számitástechnikai Koordinációs Intézet

A RENDSZER KIDOLGOZÁSÁNAK CÉLJA

A Számitástechnikai Koordinációs Intézetben 1974. szeptemberében, konkrét felhasználói igény alapján, egy VIDEOTON R-10 kisszámitógépre alapozott információs rendszer kidolgozása kezdődött meg.
Az igények felmérése alapján tisztázódott, hogy néhány meghatározott rekord szerkezetü file létrehozására, valamint a rekordokon belül néhány mező szerinti lekérdezést és módositást lehetővé tevő programokra van a felhasználónak szüksége.

Részben a felhasználói igények várható módosulásai okozta problémákat megelőzendő, részben feltételezve, hogy hasonló felhasználói igények is kielégithetők egy általánositott, a konkrét igényeket tulhaladó programrendszer kidolgozásával, a feladat általános megoldása mellett döntöttünk. Ezt indokolta az is, hogy a hazánkban installált számitógépek többsége batch-üzemü feldolgozást végez, akár tudományos kutatások segitésére, akár különböző ipari, vagy kereskedelmi folyamatot kisérő információ feldolgozására használják.
Bizonyos területeken minőségi ugrást jelentene /nyilvántartási rendszerek/, máshol viszont eleve követelmény lehet /helyfoglaló rendszerek, vezetői információs rendszerek/ a számitógép real-time üzemü alkalmazása.

Az általános megoldás mellett szólt az is, hogy igy a hazai gyártmányu R-10 számitógép alkalmazási programkészletét, és ezzel a felhasználási területek körét bővithetjük, összhangban az Intézet

célkitüzéseivel.

HARDWARE KOMPONENSEK

A felhasználói programrendszer kidolgozásánál a felhasználónál üzembehelyezett R-10 konfiguráció lehetőségeiből indultunk ki, igy ez tekinthető alapkiépitésnek:

Központi egység	32 Kbyte operativ memóriával	
800 Kbyte-os fix diszk		2 db.
VT 340 display		4 db.
VT 343 nyomtató		
MOM Lyukszalag olvasó		
MOM lyukszalag lyukasztó		
VT 2010l operátori irógép		
TAM 200 modem	telefon vonalanként	2 db.

Természetesen kevesebb display-el is üzemeltethető a rendszer, és ha a display-ek lokális üzemben kerülnek felhasználásra, ugy a modemekre nincs szükség.

A display-ek vagy párhuzamos multiplexeren, vagy soros aszinkron illesztőegységen keresztül csatlakozhatnak a számitógéphez.

A felhasználói program megirásánál ügyeltünk arra, hogy mindkét lehetőséget biztositsuk, sőt a két csatlakoztatási megoldás együttes alkalmazása is megoldott. Az üzembehelyezett rendszernél három display a multiplexer csatornán, egy pedig TAM 200-as modemek és telefonvonal felhasználásával 15 kilométer távolságból aszinkron csatolóegységen keresztül csatlakozik a rendszerhez.

Érdemes még talán annyit megjegyezni, hogy az installált rendszerben a display-ek ciril opcióval, a sornyomtató latin és ciril jelkészlettel rendelkezik, igy a felhasználói program alkalmas orosz nyelvü szöveg fogadására, ill. kiirására is.

Ez alól csak a konzol irógép kivétel /csak latin betüs klaviaturával rendelkezik/, de ez nem okoz problémát, mivel itt csak a rendszer beinditásánál, vagy hibajelzésnél jelenik meg néhány karakternyi információ.

ALAP-SOFTWARE

A rendszer a VIDEOTON által a fenti konfiguráció kiszolgálására generált RTDM monitor felügyelete alatt fut. Speciális ez a monitor abban a tekintetben, hogy a soros aszinkron és a multiplexeren keresztüli display kezelést biztositó handlert is tartalmazza. Megemlitjük, hogy a két handler interface nem teljesen azonos, az eltéréseket felhasználói program szinten kellett kompenzálni. A felmerült problémák gyors megoldását elősegitette a gyártó cég szakembereivel folytatott hatékony és közvetlen együttmüködés.

A felhasználói program kidolgozásakor felhasználtuk a géppel szállitott FMS-M minidiszkes file kezelő rendszert, igy nem kellett külön file kezelő modulokat készitenl. Ez a file kezelő rendszer a könyvtár adatzónáján létesitett file-ok kezelésére hivatott; a file megnyitásával, lezárásával, rekordok file-ba való kivitelével és beolvasásával kapcsolatos feladatokat látja el. A gyakorlat azt mutatta, hogy jól használható eszköz adatkezelési feladatoknál, de feltétlenül szükséges némi időt forditani a rendszer megismerésére és gyakorlati kipróbálására. Kis módositást kellett például végrehajtani a file kezelő rendszerben, mert a file név definiálásnál a ciril betüket nem tekintette alfabetikus karakternek.
A fenti hardware és software eszközökre alapozva dolgoztuk ki a felhasználókat time-sharing üzemmódban kiszolgáló információs rendszert, mely az alábbi lehetőségeket biztositja.

A RENDSZER SZOLGÁLTATÁSAI

A kifejlesztett programcsomag lehetővé teszi tetszőleges számu

és rekordszerkezetü file

- generálását
- lekérdezését
- módositását
- karbantartását

A file-ok karbantartására, törlésére, az időlegesen nem használt file-ok kimentésére, visszavitelére, illetve a biztonsági okokból készitendő másolati szalagok előállitására szolgáló modulokat tartalmazó file-karbantartó program teszi teljessé a timesharing szolgáltatást nyujtó programot.

A file-ok számára, az egyes file-ok terjedelmére vonatkozó korlátozást csak a rendelkezésre álló háttér-tároló kapacitása ad. A program által megszabott, az ésszerüség által diktált és könnyen betartható korlátozás, hogy a rekordok maximális hossza 1024 karakter lehet. A rekordok tetszőleges méretü, de maximum 20 mezőre bonthatók.

File generálás történhet az adatok meghatározott szintaktika szerinti lyukszalagra vitele után batch üzemü generálással, illetve "üres" file generálása után on-line üzemben, valamelyik displayről tölthető fel a file. Batch üzemben felvitt file javitása is történhet on-line üzemben.

Az információs rendszerben tárolt adatok védelme többszintü és a kivánt biztonsági követelményeknek megfelelően alakitható ki. Igy például a felhasználó által beirt "kulcsszó" és a felhasználó nevének tesztelése után jogosult a kezelő személy

- a file nevek
- a kulcsszó által meghatározott file-strukturák és adatok lekérdezésére
- a kulcsszó által meghatározott file-ok módositására.

A felhasználónak módjában áll minden egyes lekérdezésnél megvál-

toztatni a kulcsszavát ezzel is csökkentve a jogosulatlan hozzáférés lehetőségét.
Lehetőség van a file-ok display-ekhez történő hozzárendelésére is, ha a felhasználói igények ezt szükségessé teszik.
A rendszer üzemeltetésének biztonságát és az üzemeltetési fegyelmet szolgálja a rendszerben lévő automatikus naplózás. Minden hozzáférésről a diszken lévő napló file-ba bejegyzés történik, rögzitve a display számát, a felhasználó nevét, a file nevét és a hozzáférés módját /lekérdezés, módositás/.

Müszak végén a rendszer lekapcsolásakor a sornyomtatón a hozzáférések sorrendjében kerülnek kilistázásra az egyes hozzáférésekre vonatkozó bejegyzések.

A lekérdezni vagy módositani kivánt rekordok azonositása, a definiált mezők bármelyike szerint történhet egyenlőség, kisebb, nagyobb illetve intervallum relációkkal. A különböző vagy azonos mezőkre megadott fenti relációk logikai ÉS/VAGY kapcsolatokkal láncolhatók és igy a kiválasztott rekordok halmaza a kivánalmaknak megfelelően szükithető, illetve bővithető. A parancsok által meghatározott rekordokból kiválaszthatók és táblázatba rendezhetők a látni, illetve módositani kivánt mezők. A kivánalmaknak megfelelően a lekérdezésekről, illetve módositásokról a diszk munka file-jaiban "hard-copy" készülhet a felhasználó igénye szerint, mely a lekérdezés/módositás után megjelenik a sornyomtatón.

A felhasználó munkáját befejezve logikailag lezárja a rendszert az általa használt display-en és ujabb felhasználó csupán az előzőekben vázolt jogosultság ellenőrző eljárás után férhet az információs rendszerhez.

A felhasználó és a rendszer közötti dialógus három nyelven folyhat /magyar, orosz, angol/ és az egész rendszer alkalmas latin és ciril betüs információk kezelésére.

A kommunikációs nyelvek egyszerü módon változtathatók, és minimális beavatkozással bővithető az alkalmazható nyelvek köre. Minden

felhasználó tetszés szerint választhat a megadott három nyelv közül, és az egyes display-eken egymástól függetlenül különböző nyelveken folyhat az információs rendszerrel a dialógus.

A rendszer párbeszédes üzemben történő vezérlésével az ember-gép kapcsolat olyan formáját kivántuk kialakitani, ami minimális kiképzési időt igényel. A gyors működést azzal segitettük elő, hogy a lehetséges válaszok az egyes kérdéseknél megjelennek a képernyőn, de válaszként a felhasználónak mindig csak a felsorolt lehetőségek közül kiválasztott szó első betüjét kell a klaviaturán leütni. Például a

LEKERDEZES:NEV/REKORD/ADAT/VEGE?

LEKERDEZES:

kiirás a display-en sorrendben a

- file nevek kiirása
- egy file rekord szerkezetének kiirása
- adat lekérdezése
- a lekérdezés befejezése

alternativák közötti választásra szólit fel. Az első karakter az N, R, A vagy V betük egyikének leütése a kiválasztott funkció végrehajtását eredményezi.

A rendszer működésének meggyorsitását szolgálja az a lehetőség is, hogy a felhasználó, amikor a file vagy mező azonositására van szükség, akkor ezt nemcsak az azonositó név, hanem a sorszám bebillentyüzésével is elérheti.

A rendszer szolgáltatásainak vázlatos ismertetésével reméljük sikerült érzékeltetni a kidolgozott információs rendszer flexibilitását és széleskörü felhasználási lehetőségét.

A FELHASZNÁLÓI PROGRAMRENDSZER STRUKTURÁJA

Nem célunk a programrendszer részletes ismertetése, ezért a felépités

rövid áttekintése mellett csak néhány jellemző problémára és azok megoldására térünk ki.

A programrendszer az R-10 assembler nyelvén került megirásra, igy az ismertetés egyes részeinél az utalások esetleg csak az R-10 assembler-ét ismerők részére lesz élvezhető.
Törekszünk azonban az egyes megoldások géptől és programnyelvtől független megfogalmazására.

A program teljesen a modularitás elvei alapján épül fel, az egyes modulokat külön-külön tesztelve épitettük be a programba. Egy-egy modul terjedelme 100-200 utasitás körül van.

A rendszer tervezésénél és kidolgozásánál a top-down elvnek megfelelően jártunk el, a munkát a legfelső szint, a főbb funkciók megtervezésénél és az ezek vezérlését realizáló modulok kidolgozásánál kezdtük el és haladtunk az egyes részfunkciók teljes kialakitása felé. Meg kell azonban emliteni, hogy a munka első szakaszában igyekeztünk néhány alapvető funkciót végző modult definiálni és kidolgozni, amelyekre minden szinten szükség lehet.

Alapvető problémát jelentett és ugy véljük jelent minden hasonló rendszer kiépitésénél, hogy a rendelkezésre álló tárkapacitás jelentős részét "elviszik" az egyes display-eknek, azaz felhasználóknak fenntartott I/O pufferterületek. Esetünkben például csak a diszken lévő file-ok pufferterülete 1024 byte, azaz 1 Kbyte felhasználónként! És akkor még hol van a "hard-copy" részére fenntartott munka file-ok puffere, a display-ek I/O puffere, a naplózás pufferterülete!

A megoldás csak az lehetett, hogy alaposan mérlegelni kellett, melyek azok a pufferterületek, amelyeknek állandó helyet kell biztositani az operativ memóriában, a többinek pedig egy közös helyet fenntartani.

Az egyes felhasználók pufferjeinek tartalmát a diszken tárolja a rendszer és mindig csak az éppen szükségesek kerülnek az operativ tárba.

A sorozatos diszk átvitelek ugyan némileg lassitják a rendszer működését, de ez a felhasználó számára nem jelent érzékelhető késedelmet.

Az előzőekben emlitett problémákat a "szervező" modul oldja meg, mely gondoskodik a diszken tárolt /display-enként különböző/ információknak az operativ tár állandó területére történő betöltéséről, továbbá a display-enként különböző, de az operativ tárban tárolt mezők címének aktualizálásáról.

A szervező modul az R-10 pultján kijelölt display-ket aktivizálja csak, lehetővé téve a rendszer további védelmét illetéktelen hozzáféréssel szemben, és egyben módot nyujt a rendszer optimális kihasználására a konkrét felhasználók számának figyelembevételével.

A speciális szervező-modul eredményeként a programrendszer moduljai - kevés és jól definiálható szabály betartásán tulmenően - olyan felépitésüek, mintha csupán egyetlen display-re dolgozna a rendszer.

A modulokat két csoportra osztjuk:

a./ egyszeres lefutásu modulok
b./ többszörös lefutásu modulok

Többszörös lefutásu moduloknak nevezzük azokat, melyek display, sornyomtató, vagy lyukszalag perifériát használnak. Ezek a modulok input/output átvitel beinditása után megszakitásra kerülnek és az I/O átvitel ideje alatt más display-eket is kiszolgálhatnak. Ezért az ilyen moduloknál a megszakitási pontok közt az aktuális paraméterek mentéséről gondoskodni kell. A több display-t kiszolgáló egyetlen perifériák /erőforrások/ - sornyomtató, lyukszalag állomás, operátori konzol - dinamikus lefoglalásával és szabaddá tételével a rendszer nagy hatékonysággal tudja a felhasználók igényeit kielégiteni.

A display-k kommunikációs / I/O / vezérlése olyan megoldásu, hogy

egyaránt alkalmas a lokális üzemü multiplexeres, vagy a távadatfeldolgozásnál használatos soros-aszinkron információátvitel kiszolgálására.

A RENDSZER ALKALMAZHATÓSÁGA

A rendszer információs bázisa a sajátos helyi igények szerint állítható össze és hatékonyságát erősen befolyásolja a felhasználó által definiált rekord-struktura célszerüsége.

A rendszer több szakmai bemutatón kivül az 1975. évi Budapesti Tavaszi Vásáron is kiállitásra került, ahol - raktározás, személyi nyilvántartás és kimenő levelezés - minta file-ok kerültek bemutatásra.

A fenti példákon tulmenően létrehozható egy olyan vezetői tájékoztatást célzó adatbázis is, mely egy gyár termékeinek nomenklaturáját, azok gyártási ütemtervét, a beszerzés és kiszállitás koordinációs tervét és valamennyi tevékenységre vonatkozó határidőket gyüjti, rendszerezi és szolgáltatja.

A rendszer segitségével lényegesen leegyszerüsithető a raktárról szállitó kereskedelmi vállalatok un. diszponibilis készlet nyilvántartása, valamint nagyobb szállodák, utazási irodák helyfoglaló rendszere is.

A kisszámitógépes információs rendszer egészségügyi területen - például kórházi ágynyilvántartásnál, betegirányitó szolgálatnál stb. - is hatékonyan alkalmazható.

További lehetőségeket jelent az a tény, hogy a programok tovább fejlesztésével a rendszer adaptálható az R-12-re, igy az alkalmazások köre kiszélesithető olyan adathalmazokat kezelő területekre is, ahol egy R-10 kisszámitógép háttértárolói kevésnek bizonyulnak.

FOLYAMATIRÁNYITÓ SOFTWARE RENDSZERTERVEZÉSI PROBLÉMÁI

Kondorosi Károly, Langer László, Megyeri József,
Risztics Péter
Budapesti Müszaki Egyetem Folyamatszabályozási Tanszék
Mosoni Imre
Videoton Fejlesztési Intézet

BEVEZETÉS

Előadásunk a "Testvériség" szovjet-magyar gázvezeték számitógépes irányitórendszerének ismertetésével, és a folyamatirányitó software rendszer tervezési problémáival foglalkozik. A problémák nagy része általános érvényü, igy az előadás felhivja a felhasználók figyelmét a technológia, az irányitó rendszer, a software rendszer és a megvalósitandó funkciók kölcsönhatására, az összhang megteremtésének szükségszerüségére.

A TECHNOLÓGIA

A "Testvériség" gázvezeték, mint irányitandó technológia az alábbi sajátosságokkal rendelkezik:
Folytonos, állandó felügyeletet igénylő, lineárisan nagy kiterjedésü,(több száz km), az emberi felügyeletet nem igénylő állomásokon vannak az ellenőrző pontok. Vegyes dinamikus tulajdonságu, a vezetéket jellemző paraméterek viszonylag lassan, mig az egyes berendezések (pl. kompresszor állomások) paraméterei gyorsan változnak. Az energia szállitás és a fogyasztók kiszolgálása miatt jó, központi áttekinthetőséget igényel.

A GÁZVEZETÉK IRÁNYITÁSI FOLYAMATA

Az irányitó rendszer a gázvezetéki technológiának megfelelő hierarchikus felépitésü. A legalsó szinten a vezetékre ill. csomópontjaira telepitett állomási irányitó rendszer helyezkedik el. Az ezekben összegyüjtött információk a vezetékszakasz diszpécserközpontjába kerülnek, ahonnét a távadatátvi-

teli rendszer közvetitésével jut az információ a központi, üzemviteli központba. Itt helyezkedik el a telemechanika vezérlő egység a speciális kezelő berendezésekkel(sématábla, irányitó pult) és a VT 1010 (R 10) számitógép, amely nyiltláncu, felügyelő irányitást valósit meg. A döntés és a beavatkozás a kezelő személyzet feladata marad, bár a számitógép szolgáltatásait igénybe veszi.

A központi számitógép technológiai felügyelő feladatköre:
-Analóg folyamatváltozók rendszeres mérése és feldolgozása (dimenzionálás, határérték-tullépések figyelése, változási sebesség figyelése,stb.).
-Kétállapotu jelzések (zavar- és vészjelzések) kezelése.
-Számitott változók meghatározása és feldolgozása (korrekciós áramlásmérés, hozamintegrálás, anyagmérleg készités).
-A folyamat berendezéseinek vezérlése(tolózárak müködtetése).
-Kijelzések kiadása a kezelő számára (nyomtatott üzenetek: üzemi napló, eseménynapló; analóg regisztrátumok; mérési és számitási értékek számkijelzőkön; lát- és hangjelzők müködtetése).
-A kezelő által kért rendszermódositások végrehajtása.

Az irányitási rendszer kezdeti kiépitésben alkalmas max. 30 db állomás adatainak (210 db jelzési szó, 280 db mérési szó) fogadására, tárolására, feldolgozására. A telemechanikai rendszer 200 baud-os jelátviteli sebessége következtében ezen kiépitésnél a várható adatbeviteli ciklusidő értéke 120 sec körüli érték.
A rendszer végleges kiépitésben alkalmas 80 db állomás adatainak (1000 db távmérés, 2500 db kétállapotu jelzés) fogadására, tárolására, feldolgozására.

AZ IRÁNYITÓ SOFTWARE RENDSZER

A rendszer funkcióit az irányitó program rendszer biztositja, amely két fő részből tevődik össze: a VT 1010 alapsoftware elemek és a felhasználói programrendszer.
Az irányitási feladat megoldása érdekében a következő lényegesebb alapsoftware elemeket használjuk:
Handlerek, matematikai könyvtári modulok, folyamatirányitó real-time monitor (PCM), stb.

A PCM monitor a különféle folyamatirányitási alkalmazásoknál igényelt általános programfunkciókat valósitja meg és megkönnyiti az egyedi alkalmazásoktól függő felhasználói programrendszer kialakitását.

A Process Control Monitor (PCM) real-time tipusu mérésadatgyüjtési és folyamatszabályozási feladatok, u.n. taskok futtatására alkalmas.
A PCM legfőbb szolgáltatásai:

- 128-ra bőviti a hardware által eddig 32-re korlátozott megszakitási szintek számát
- Közvetlen tároló segitségével az egyidejüleg futtatható feladatok összméretét 64 K byte-ról 2-300 K byte-ra növeli
- Különböző feladatok között kommunikációt biztositó eszközöket biztosit
- Szemafor rendszerrel rendelkezik a programok szinkronizálására
- Időzités kezeléssel biztositja, hogy a programok a külső környezettel szinkronizálhassák magukat
- Vezérlőkonzolon keresztül megadott operátori parancsokkal PCM tábláinak kiolvasása, módositása ill. a PCM alatt futó programok futásának módositása lehetséges
- A programok közös funkcióinak ellátására ujrabelépő szekciókönyvtár áll rendelkezésre

A PCM által kezelt minimális hardware konfiguráció: R 10 központi egység + 16 K memória konzolirógéppel, 400 K fixfejes

mágneslemez, az 1-2-3 IT szinteket kezelő kártya, valamint a real-time óra.

A PCM magas szinten multiprogramozott rendszer, amely 32 valódi és további 96 virtuális IT szinten futó program prioritás vezérelte futtatását engedi meg. A feladatokat több szempontból csoportosithatjuk: prioritási szintjük, hozzáférésük, időzitésük szerint.

Prioritás szerint lehetnek

- 2-31 prioritásu közvetlen feladatok (valódi hardware IT szint aktiválásával indithatók)
- 33-127 prioritásu u.n. elhalasztott feladatok virtuális IT szinten futnak (ezt az 1 szint multiplexálása teszi).
- 0 szinten futó program un. Back ground program nem real-time jellegü feldolgozások számára.

Hozzáférésük szerint lehetnek

- Tárrezidens feladatok a rendszer inicializálásától a következő ujrainiciálásig az operativ tárban vannak (a MONITOR FOREGROUND, BACKGROUND zónájában helyezkednek el.)
- Diszkrezidens feladatok a diszk EP zónájában foglanak helyet. Csak futásuk időtartamára kerülnek a tár MIDDLEGROUND zónájába, és ha egy magasabb prioritásu feladat miatt a tárat el kell hagyniuk, un. swapping tevékenység folyik le és a diszk SW zónájában kerülnek elhelyezésre.

Időzitésük szerint lehetnek: időzitést nem használók; egyszeres futásra időzitettek; ciklikus futásra időzitettek.

A feladatok prioritásuk szerint kerülnek futásra. A PCM monitor két ütemező szerint futtatja a feladatokat: hardware és software ütemező szerint.

A hardware ütemező az R 10 megszakitási rendszere.

A software ütemező, scheduler a PCM 1. IT szinten futó rendszer programja. Táblái segitségével inditja a feladatokat.

A két ütemező egymással kölcsönös kapcsolatban van. A hardware ütemező inditja magát a software ütemezőt is, az 1. IT szint aktiválásával. Az utóbbi viszont közvetlen feladatokat

is indithat, aktiválhat más IT szinteket is.
A feladatok 5 különböző állapotban lehetnek: végrehajtás alatt álló, végrehajtásra választható , várakozó, felfüggesztett, nyugalomban lévő.
Végrehajtás alatt álló feladat az R 10 felépitéséből következően csak egy van, ez rendelkezik egy adott pillanatban a gép erőforrásaival.
Választható minden olyan feladat, amelyet az operátor vagy valamelyik más feladat azonnali végrehajtásra kijelölt (választhatóvá tett).
Várakozó feladat eseményre, késleltetésre vagy erőforrás szabaddá válására várakozik, s annak megtörténte után választhatóvá válik.
Felfüggesztett feladatok saját magukat zárják ki a választható feladatok sorából. Ujra érvényesitésüket más feladat vagy az operátor kéri.
Nyugalomban van egy feladat, ha nem tartozik egyetlen előző csoportba sem.

A prioritási szintek 0-127-ig terjednek, ezen belül a prioritás 1-31-ig növekszik (hardware prioritás), 32-127-ig csökken (software prioritás) és a 0 a legalacsonyabb szint.

Program iniciálás és swapping
A tárrezidens programok egyszer kerülnek be a tárba a monitor iniciálásakor.A diszkrezidens programok a monitor iniciálásakor csak paramétereikkel kerülnek be a rezidensen kezelt rendszertáblákba. Egy feladat számára akkor keres csak helyet a monitor, ha az mint legnagyobb prioritásu választható feladat diszken várakozik. Hely biztositása érdekében a monitor folyamatban lévő vagy várakozó feladatokat is eltávolithat a tárból.

Input-output müveletek
A PCM az OS-10 operációs címkéit ha sználja. Lehetővé teszi

az operátor részére, hogy egy fizikai perifériához más-más operációs címkét rendeljen, ill. egy operációs címkéhez (logikai periféria) más-más fizikai perifériát rendeljen. A transzfer kérések egybeesése esetén a PCM minden fizikai perifériához várakozási sort épit fel, a kéréseket időrendi (FIFO) sorba állitja.

Supervizor hivások

A PCM monitor tartalmazza az OS-10 rendszer többi monitorjai által nyujtott supervizor szolgáltatásokat. Ezen felül rendelkezik néhány uj szekcióval, amelyek a real-time szolgáltatásokat növelik.
Lehetővé teszi, hogy a felhasználó is definiáljon közösen használt ujrabelépő szekciókat, ezeket a rendszer iniciálásakor helyezzük el a monitorban és paraméteres- indexelt CSV utasitással hivjuk.

Operátori beavatkozás

A PCM lehetőséget ad az operátornak, hogy a konzolirógépen keresztül módositásokat hajtson végre a rendszer tábláiban ill. lekérdezze azokat. Az operátori kéréseket értelmező task overlay-ezett és az EP könyvtárban foglal helyet.

Rendszer konfiguráció

A PCM az OS-10-ben meglévő GSYS 10 program segitségével generálható könyvtáraiból. Ezáltal tetszőleges supervizor felépitést tár és diszk felosztást, periféria kezelést valósithatunk meg.

Összefoglalva az R 10 Process Control Monitor multiprogramozott, foreground, middle-ground, back-ground szerkezetü monitor, amely fix prioritásu feladatok ütemezéssel, perifériakezeléssel, operátorkommunikációval, real-time feladatok megoldását teszi lehetővé.

Tetszőleges, OS-10.-ben meglévő nyelven irt feladatok futtatására alkalmas. Overlay-ezési és swappingelő monitorelemekkel a futtatandó feladatok összmérete jóval meghaladhatja az R 10 maximális kiépitésének megfelelő 64 K byte-ot. Rendszergenerálással tetszőleges R 10 konfiguráció kiszolgálására alkalmas.

A FELHASZNÁLÓI PROGRAMRENDSZER

A felhasználói programrendszer teljes egészében az adott alkalmazás igényeinek megfelelően, az adott feladathoz és a berendezésekhez igazodóan terveztük meg.

Elkészitése az alábbi kiindulási adatok alapján történt:

- Feladatspecifikáció
- Számitógépkonfiguráció
- A számitógéphez csatlakozó berendezések müködésének részletes leirása
- A rendelkezésre álló folyamatirányitó monitor és a számitógépgyártó cég által szolgáltatott egyéb programok (un. alapsoftware)

Az adatstruktura valamint a programrendszert alkotó taskok és rutinok felépitését és kapcsolódását a feladatból adódó sajátosságok, a hardware adottságok, valamint a VT 1010 alapsoftware által biztositott szolgáltatások határozzák meg.

A feladat sajátosságából adódóan döntő az adatgyüjtés jellege és a feldolgozások.

A bemeneti adatok állomásonként csoportositottak, azonos állomásról származó adatok csak az állomás valamennyi adatának helyes beérkezésekor fogadhatók el és dolgozhatók fel.

Az adatgyüjtés alapvető üzemmódja szabadonfutó ciklus, amelyben az utolsó állomás letapogatása után azonnal elkezdődik az ujabb ciklus.

Operátori kérésre az alapciklusba beékelve egyes adatokat sürítetten kell beolvasni.

Bármilyen feldolgozást csak érvényes, formátumát tekintve helyes adaton lehet elvégezni.
A feldolgozások időzités szempontjából: az adat-frissülés ütemében végzendő; rögzitett ciklusidejü; operátori beavatkozással inditott feldolgozások lehetnek.
Fizikai értékre való átszámitás és határértékvizsgálat minden adatra történik.
A többi számitásra (hozam,különbség, trend, stb.) fajtánként 20-60 adat kerül.

A hardware adottságoknál a telemechanikai rendszer - az R 10 adottságain kivül - sajátosságai a leglényegesebbek. A telemechanika hir-format egysége és az állomások közötti adatforgalmat lebonyolitó hircsatorna sebességéből és az információ formátumából adódóan a real-time numerikus bemeneti vonalakon történő két adatbeadás között kb. 100 msec telik el. Ez az idő a feldolgozó programok futtatására használható fel. Az adatgyüjtés ciklusidejét a hircsatorna sebessége korlátozza.

A fenti szempontok és sajátosságok figyelembevételével az adatstrukturát és a programrendszert a következőképpen épitettük fel:

A mérési adatokat fizikai értékre átszámitott formában ferritrezidensen tároljuk.Az adatok állomáscimük és alcimük alapján kereshetők meg pointer-táblán keresztül. Az adatgyüjtés állomásonkénti szervezése miatt egy állomásnyi nyers adat tárolására szolgáló pufferterületet tartunk fenn.
A feldolgozások közül a minden mérési értékre való határértékfigyelés és fizikai értékre való átszámitás paramétereit az adattárhoz hasonló szervezésben tároljuk.
A többi feldolgozás az operandusokat definiáló cimtáblákat és az eredményeket tároló számitási táblázat használja.

A programrendszert két irányból épitve alakitottuk ki. Egyrészt a rendszer feladatainak (vezérlés, adatgyüjtés, naplózás, stb.) megfelelő bontásban u.n. master-taskokat készitettünk, amelyek az adatmozgatást és feldolgozást végző, valamint a kimeneteket generáló taskok és közös rutinok megfelelő futtatását szervezik.
Másrészt az adattáblák és kimenő információk formátumának ismeretében az egyszerübb feldolgozási lépésekre közös rutinokat, az összetettebb, de több feladat során alkalmazott lépésekre taskokat specifikáltunk. Ezek futtatását vezérlik a master-taskok.
A rendszer müködését vezérlő és a feldolgozásokat paraméterekkel ellátó és azok változtatását végző operátori parancsok párbeszédes jellegüek.

RENDSZERTERVEZÉSI TAPASZTALATOK

A számitógépes irányitó rendszer tervezésének kulcskérdése a megfelelő feladatspecifikéció.

A feladatspecifikáció tartalmazza a paraméterek értékvariációinak számát. Ha a feladatspecifikációban egy-egy paraméterre tul sok értékvariációt engedünk meg, jelentős hasznos memória területet veszithetünk el.

A feladatspecifikációban általában előirják az adatok ábrázolási és feldolgozási pontosságát. A megkivánt pontosságnak összhangban kell lennie az alkalmazott mérőmüszerek pontosságával, hiszen nincs értelme a számitógépen belüli feldolgozás eredményeire nagyobb pontosságot előirni, mint a kiinduló adatok pontossága. A tulzott pontossági igény felesleges memória felhasználást és a feldolgozási idők indokolatlan megnövekedését eredményezi.

Jellegzetes hiba pl. a feladatspecifikációban a tulzott mennyiségü adatkivitel előirása. Tul gyakori naplók nyomta-

tása, felesleges lyukszalagos adatkivitel, stb. Aszámitógép megjelenése az irányitó rendszerben felvetette a közvetlenül mért folyamatváltozókon alapuló számitások elvégzését, igy pl. gázszállitásnál a gépi adatok alapján történő számlázás gondolatát. A megoldás látszólag egyszerü, hiszent a hozamszámitásokat és a pillanatértékek integrálását a számitógép el tudja végezni, az adatokat megfelelő ideig tárolni tudja és a kivánt formátumban meg is jeleniti. Problémát okoz azonban a hitelesség és az illetéktelen adatmódositással szembeni védelem biztositása. Hagyományos fogyasztásmérők (érzékelő és kijelző együttesen) hitelesithetők, a megfelelően elhelyezett ólomzárakkal az illetéktelen beavatkozás megállapitható. Számitógépes elszámolásképzés esetén a "fogyasztásmérő" a hagyományos értelemben nem körülhatárolható, Ólomzárakkal nem védhető, stb. Ezekben a kérdésekben tudomásunk szerint világviszonylatban sincs egyértelmü állásfoglalás.
Véleményünk szerint néhány kisérletet kell végezni, amelyek tapasztalatai alapján az Országos Mérésügyi Hivatal kialakithatja ezzel kapcsolatos állásfoglalását.

A feladatspecifikáció gyakran tulzott rugalmasságot követel meg a számitógéptől. Jellegzetes igény például a háttér munka igény. Ezek a követelmények szükségtelenül bonyolitják a programrendszert és hátrányosan befolyásolják a rendszer megbizhatóságát. Kimondottan veszélyesnek és megengedhetetlennek tartjuk a közvetlen irányitást végző számitógépen olyan háttérmunkát futtatni, amely pl. valamilyen helytelen kezelői beavatkozás folytán - a normál folyamatirányitási funkciók ellátását veszélyezteti.

A programrendszer elkészitése közben felvetődő problémák egy része kellő tapasztalattal és gondos mérlegeléssel a tervezés megelőző fázisaiban kiküszöbölhető, de mindenképpen cél-

szerü ha a megvalósitásban együttmüködő vállalatok és intézetek képviselőiből álló csoport dönt a felvetődő müszaki kérdésekben.

A TPA-i KISSZÁMITÓGÉP MAGASSZINTÜ REAL-TIME PROGRAMRENDSZERE

Buday László
MTA Központi Fizikai Kutató Intézet

A real-time rendszer jellegzetessége, hogy a "külvilág eseményei" által vezérelt, azokra gyorsan reagáló rendszer. A "valódi" /másodpercben mért/ idő vezérlő hatása is beleérthető ebbe, hiszen az idő mulását is események /a real-time clock interrupt-jai/ jelzik a számitógépnek. Az ilyen jellegü rendszer megvalósitásának hagyományos módja az, hogy a teljes feladatot viszonylag kis méretü és rövid futási idejü programokkal elvégezhető, de azért értelmes, kerek egészet alkotó részfeladatra /task-ra/ bontjuk. A real-time executive feladata ezen task-ok megfelelő sorrend szerinti futtatása. Egyuttal az is látható, hogy a kisszámitógépek szükebb memóriakapacitása és szerényebb utasitáskészlete nem komoly korlátozó tényező, hiszen ma már rendelkezésre állnak nagy sebességü, nagy kapacitásu háttértárolók, komolyabb aritmetikai igény pedig általában nem szokott felmerülni. Rendszereink további jellegzetessége a megnövekedett periféria igény, elsősorban az un. real-time perifériák vonatkozásában. Ezek sokfélesége, univerzálisan nehezen, vagy egyáltalán nem kezelhető voltuk elég sok problémát jelent különösen akkor, ha kezelésüket egy magasszintü nyelv keretei között akarjuk megoldani. Ilyenkor általában elkerülhetetlen egyedi periféria-driverek irása és a rendszerbe való beillesztése.

AZ INDAL NYELV

Az INDAL /INdustrial Data Acquisition Language/nyelv az előzőkben ismertetett elveknek megfelelően felépitett magasszintü nyelv. A nyelv négy alapvető szegmense a következő:

I. Feladatspecifikáló szegmens
Itt deklarálhatjuk a különleges kezelést igénylő perifériákat

/multiplex és interrupt device-ok/, valamint a közös adatmezőt.

II. PHASE szegmens

A szegmens ACTIVITY szekciója tartalmazza a "menetrendező" utasitásokat:

```
#100 DO SNAP #10 EVERY 5 SEC PRIORITY 3
#101 DO SNAP #11 AT 14:15 PRIORITY 9
#102 DO SNAP #12 DELAY 10 MIN PRIORITY 5
```

Ezek az "utasitások" értelemszerüen bizonyos task-ok /SNAP/ futtatásra való előjegyzését irják elő időben ciklikusan, egy bizonyos időpontban, illetve egy eseményhez /TIMER START/ képest valamennyi késleltetéssel.

A szegmens ACTION szekciója deklarálja az INTERRUPT SNAP-et, és itt irhatunk elő bizonyos iniciáló előjegyzéseket.

```
.ACTION
DO SNAP #20 PRIORITY INTERRUPT
TIMER(START,#100)
DO SNAP #30 PRIORITY 1
```

Az itt deklarált un. INTERRUPT SNAP kerül előjegyzésre az INTERRUPT DEVICE-ok bármelyikének jelentkezésekor. A TIMER utasitással indithatjuk /TIMER START/, illetve állithatjuk le /TIMER STOP/ az "idő mulását" a megfelelő ACTIVITY utasitásban. A DO utasitások nem szubrutinszerü hivások, hanem futtatási előjegyzések a várakozólista megfelelő prioritási szintjére.

III. SNAP szegmens

Ez a szegmens felel meg a bevezetőben emlitett task-nak. A deklarációs részben lokális adatmezőt és FORMAT utasitásokat specifikálhatunk, mig a PROCESS szekcióban foglalnak helyet a BASIC-szerü végrehajtható utasitások. Itt emlitjük meg, hogy a real-time perifériáknak szóló input/output utasitások /GET/SEND/ azonos szerkezetüek a hagyományos perifériákra /TTY, PTP stb/ vonatkozó utasitásokkal, az adatlista elemeinek értelmezését a már emlitett speciális periféria-driver-ek végzik.

A végrehajtható utasitások között szerepelhetnek a PHASE szegmens ACTION szekciójában megismert TIMER és DO utasitások is.

Végül lehetőség van arra, hogy az INDAL nyelvü utasitások mellett TPA assembler utasitásokat használjunk, sőt az assembler blokkok használhatják az INDAL program változóit és viszont.

IV. SZUBRUTIN szegmens

A SNAP-hez hasonló szerkezetü szegmens és a különböző SNAP-ekben előforduló azonos feladatok elvégzésére használhatjuk.

AZ INDAL INTERPRETER

Az INDAL COMPILER a forrásprogramot un. "vektorkódok" sorozatára forditja le. Ez a forma valahol a compiler és az interpreter szint között van, a run-time rutinoknak ui. nem valami elvont azonositója, hanem a belépési cime kerül a leforditott programba. Az INDAL interpreter a run-time rutinokon kivül tartalmazza a futtatásvezérlő-menetrendező executive-ot és a standard device-ok /teletype, clock, diszk stb/ driver-eit.

Az INDAL program összes szegmensei a közös adatmező kivételével diszk rezidensek, és csak a végrehajtásuk idejére kerülnek az operativ memóriába.

A várakozólista 11 prioritási szintre van tagolva. Az első hét szint az un. foreground szint. Ez azt jelenti, hogy ha egyszer egy SNAP végrehajtása foreground prioritáson megkezdődött, akkor azt másik előjegyzés nem függesztheti fel /még magasabb prioritásu sem/. Ezzel szemben az un. background prioritásu SNAP-ek futása megszakitható, ha közben foreground előjegyzés történt.

Az INDAL interpreter a prioritásuk sorrendjében futtatja a SNAP-eket. Ha valamelyik szinten több előjegyzés van, akkor a korábban előjegyzett SNAP kerül futtatásra.

Az INDAL interpreter részévé válnak az előbb emlitett speciális real-time periféria-driver-ek, sőt a felhasználó által megirt speciális függvények is. Ezeket megfelelő kódolási szabályok betartásával a TPA assembler nyelvén kell megirni. Ezeknek a rendszerbővitő elemeknek a beépitését egy erre a célra készült rendszergeneráló program végzi. A beépitett elemek adatait többféle nyilvántartásba is bejegyzi, ilyen módon ezekről a bővitésekről a compiler is tudomást szerezhet.

AZ INDAL RENDSZER KONFIGURÁCIÓIGÉNYE

TPA-i számitógép 16 K szó operativ memóriával
Bővitett aritmetikai egység
Négyszintü interrupt opció
Teletype
256 K szó kapacitásu diszk
Real-time clock
Szalagolvasó

Bővitési lehetőségek:
Operativ memória 32 K szóig
Diszk 1024 K szóig
További teletype-ok
Szalaglyukasztó
CAMAC
•
•
•

AZ INDAL RENDSZER FELHASZNÁLÁSA

Az INDAL rendszer segitségével valósitottuk meg a Dunamenti Hőerőmü mérési adatgyüjtő rendszerét.

AZ R10 SZÁMITÓGÉP MULTIPROGRAMOZÁSÁNAK NÉHÁNY KÉRDÉSE

Kotsis Erzsébet - Lugosi Károly

VIDEOTON Számitástechnikai Gyára
Fejlesztési Intézet

A VIDEOTON Számitástechnikai Gyára Fejlesztési Intézetében az R10 számitógép operációs rendszerét az un. OS-10-t dolgozzuk ki.

A real-time operációs rendszer egyik alapelemeként fejlesztettük ki a PCM monitort, amely az R10 multiprogramozhatóságának kihasználását hivatott növelni.

Tapasztalataink szerint nagy real-time jellegü feladatok megoldásánál a rendszertervezők és programozók munkáját nagymértékben megkönnyiti ha az elvégzendő munkát kis önállónak tekinthető részfeladatokra bonthatják, és ezeket kis önállónak tekinthető programmal oldják meg.

Nagy kiépitésü, perifériákkal bőven ellátott R10 gépeknél az eredetileg rendelkezésre álló 32 megszakitási szint nagy részét handlerek, megszakitás kezelő programok foglalják el. A fennmaradó néhány szint már nem teszi lehetővé a megfelelő számu részprogramra bontást.

A felhasználói programok összmérete nagyobb feladatok esetén gyakran meghaladja a R10 maximális 32 K-tárméretét. Az OS-10 rendszerben általánosan alkalmazott overlay technika nem jelent mindig megoldást részben az overlay gyökerek helyfoglalása miatt, részben mert a futásidő tetemes részét a diszkmüveletek ideje foglalja el s ez a válaszidőt rontja.

Ismertetett szempontjaink figyelembevételével a PCM monitort az OS-10-ben meglévő real-time monitorokhoz képest két lényeges uj tulajdonsággal terveztük:

- 32-128 program "egyidejü" futtathatóságának biztositása
- a futtatandó programok összmérete 32-300 K lehet fixfejes mágneslemez bevonásával.

Dolgozatunkban e két funkció megvalósitását tárgyaljuk részletesen.

Virtuális megszakitási szintek

128 program futtatása a meglévő 32 hardware megszakitási szint mellé 96 uj szint bevezetését igényli, ezek az uj software-sen szimulált szintek a gépi megszakitási szintekhez hasonlóan prioritással rendelkeznek.

Az uj szintek felett már nem hardware megszakitás kezelő, hanem egy rendszerprogram az ÜTEMEZŐ felügyel. Az ujonnan bevezetett prioritási szinteken futó programok az 1. hardware megszakitási szinten futnak. Ugyancsak ezen a szinten fut az ÜTEMEZŐ is.

Az ÜTEMEZŐ

A virtuális szintek bevezetésével két uj programosztály jött létre.

- a hardware IT szintekhez kötött programok, ezeket közvetlen programoknak nevezzük, mert a hardware IT érvényrejutása esetén azonnal lefutnak.

- virtuális szinten futó programok, ezeket elhasztott programoknak nevezzük, mert futásuk egy másik program, az ÜTEMEZŐ lefutása után történhet meg.

A PCM monitor alatt tehát két ütemező fut.

- a hardware megszakitás kezelő rendszer
- ÜTEMEZŐ rendszerprogram.

Vizsgáljuk meg a két ütemező müködését és segédeszközeit.

Megszakitás kezelő

- akkor müködik, ha egy IT kérés érkezik illetve egy futó IT szint müködése befejeződött
- az aktuális szint számát gyorsregiszterben tárolja
- az egyes IT szintek állapotát szintenként 3 biten tárolja. Ezek

  D IT kérés jelzése
  V érvényre juttathatóság jelzése
  A külső IT kérés teljesithetőségének jelzése.

- segédtábla

  CPT vagy kontextuspointer tábla, amely az egyes szintekhez tartozó kontextusok abszolut cimét tartalmazza.

ÜTEMEZŐ

- müködik minden 100 ms-ban
  vagy akkor ha egy virtuális prioritásu program
  befejezi futását
  várakozik
  felfüggesztődik illetve
  ha egy program kéri, hogy az ÜTEMEZŐ fusson le.

- az aktuálisan futó program számát a memória egy byte-ján tárolja

- az egyes szintek állapotát egy a memóriában elhelyezett táblában tárolja minden szinthez egy byte-ot használva fel. Ennek az ugynevezett STATUS táblának egy byte-ja a következő felépitésü

A W D O S R M C

ahol a bitek jelentése:

A	a program aktiv, kerüljön végrehajtásra
W	a program eseményre vár
D	a program az operativ tárban van
O	a feladat belépési pontján áll
S	felfüggesztett program végrehajtása az A bit állapotától függetlenül tiltott
R	tárrezidens program
M	elmozditható program
C	ciklikus program.

Az ütemező a fenti bitek közül csak az A, D, O, S biteket kezeli. A fennmaradó bitek más rendszerprogramok számára szükségesek.

- segédtáblák

az ÜTEMEZŐ használja a fenti STATUS táblán kivül a CPT táblát is, amelyet az uj szintek bevezetésének megfelelően meghosszabbit.

Az ÜTEMEZŐ algoritmusa

A felesleges futás elkerülése céljából a PCM tárolja a mindenkori aktiv feladatok számának összegét, az ÜTEMEZŐ minden futása alkalmával eltárolja ezt az összeget. Következő lefutásánál összehasonlitja az aktuális és a megelőző futás idején volt összeget, ha ezek megegyeznek, azt a programot futtatja tovább, amely utoljára futott. Ha az összeg 0 a 0. szint fut. Abban az esetben, ha uj feladatot kell kiválasztania, megvizsgálja először a STATUS tábla közvetlen feladatokra vonatkozó részét. Aktiválja a legnagyobb prioritásu aktivat közülük. Majd áttér az elhalasztott feladatokra és aktiválja közülük a legmagasabb prioritásu tárban lévőt. Ha a legpriorabb program nincs a tárban, aktiválja a TÁRKEZELŐ programot.

Dinamikus tárkezelés

Az elhalasztott programok mérete és száma megkivánja, hogy azok az operativ tárban csak aktiv müködésük idejére foglaljanak helyet. Az ilyen programok diszkrezidensek. Ha az ÜTEMEZŐ észleli, hogy egy diszkrezidens program aktiv, felszólitja a második megszakitási szinten futó TÁRKEZELŐ rendszerprogramot, hogy töltse a tárba a kért programot.

Az ÜTEMEZŐ és a TÁRKEZELŐ szinkronizálását a STATUS tábla és egy, a kért program számát tartalmazó memóriarekesz biztositja.

Az operativ tárban a monitor fenntart egy területet, amely csak és kizárólag a diszkrezidens programok futtatására szolgál. A terület nagysága 1-24 K lehet. Gyakorlatilag a legnagyobb diszkrezidens feladat mérete az alsó határ, és a felhasználó által egyéb célokra el nem foglalt szabad területet teljes egészében magában foglalja. Ezt a területet 8 egyenlő részre osztjuk un. dichotomikus módszerrel.

Fenti módszer lényege, hogy egy zónát felezéssel addig oszt, amig a kapott blokk fele már kisebb az igényelt méretnél.

A dichotomikus felosztást "first fit" módszerrel együtt alkalmazva a memória allokálás következő algoritmusát valósitjuk meg:

A diszkrezidens programokat 4 osztályba soroljuk aszerint, hogy hány felezést kell végrehajtani ahhoz, hogy egy szükséges méretü memóriablokk előálljon.

Egy ilyen blokkba két program tölthető a blokk kezdetétől előre és a végétől visszafelé, attól függően, hogy már korábban milyen részt töltöttünk be.

A TÁRKEZELŐ tehát megkisérli a kért programot betölteni minden, az osztályának megfelelő blokk alsó vagy felső végébe, és végül a program oda lesz betöltve, ahol az első elegendő nagyságu szabad helyet megtaláljuk.

Példával szemléltetve:

A memóriába egymás után a következő programokat kivánjuk betölteni

A 3.5 partició hosszu
B 1.5 partició hosszu
C 0.5 partició hosszu

Az algoritmust kövessük egy olyan tártérképen, amely minden partició számára két rekeszt tartalmaz. A baloldali a blokk elején, a jobboldali a blokk végén lévő program azonositóját tartalmazza.

0	0	A	A	A	A	A	A
0	0	A	A	A	A	A	A
0	0	A	A	A	A	A	A
0	0	A	0	A	0	A	C
0	0	0	0	B	B	B	B
0	0	0	0	B	0	B	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Ha egy program számára nincs hely az operativ tárban, a TÁRKEZELŐ un. swapping tevékenységbe kezd, hogy az uj számára helyet biztositson.

A PCM monitor a programok számára két területet tart fenn a mágneslemezen.

- prototipusok számára
  ahol a programok eredeti formájukban helyezkednek el. Az első aktiválás alkalmával innen másolja ki az operativ tárba a TÁRKEZELŐ a programokat.

- futó programok számára
  ahová a tárból ideiglenesen kikerülő programok kerülnek aktuális tárképüknek megfelelően.

A swapping nem jelent mást, mint az aktuális program hosszának megfelelő terület felszabaditását ugy, hogy a tárban lévő programok közül diszkre mentjük a kevésbé fontosakat.

Az egyes programok kivonhatják magukat a swapping müködése alól ha - supervisor hivás segitségével - befagyasztják magukat a tárba, azaz pillanatnyi helyzetüket rögzitik.

Fixfejes diszket használva az átvitel sebessége kb. 12 μs/szó. Azaz 1 memóriaszó eltávolitása alatt kb 4 utasitás hajtódik végre. Látható, hogy a swapping algoritmusának kulcspontja az eltávolitandó program kiválasztása, hiszen egyetlen program diszkre vitele alatt /kb 15 ms/ 10 ms átlagos elérési idő és kb 800 szó esetén kb 5000 utasitás hajtódik végre, rossz választás/ok/ esetén a gép holtideje megnő.

A swapping algoritmusa röviden a következő:

- megállapitani a kért program osztályát
- prioritási sorba állitani a kért osztályba tartozó blokkokat
- a legkisebb prioritásu blokk legkevésbé fontos programját diszkre menteni.

Fenti eljárás folyik mindaddig, mig elegendő nagyságu szabad terület elő nem áll. A swapping eljárás minden átvitel után ujra értékeli a blokkkok tartalmát, hiszen az átvitel ideje alatt a programok STATUS-a megváltozik.

Az értékelés, amely mindig csak a blokkok kiigényelt részére vonatkozik, a következő szempontok szerint rangsorol:

- van-e befagyasztott program a területen
- hány aktiv program van, és milyen a prioritásuk
- hány karaktert kell elmozditani a terület kiüritéséhez.

A TÁRKEZELŐ tehát arra törekszik, hogy olyan blokkhoz nyuljon, amelynek üritésével a legrövidebb idő alatt végez, miközben másodlagos szempontként figyelembe veszi, hogy lehetőleg az átvitel alatt is fusson program.

Időviszonyok

Vizsgáljuk meg az elhalasztott programok válaszidejét és az ÜTEMEZŐ és TÁRKEZELŐ müködéséből származó időveszteséget /overhead/. Az időbecsléshez közelitő képletet állithatunk fel:

$$T_i = (ST+ÜE)\cdot\left(\frac{T_i}{BTU} + u\right) + \sum_k (SK\cdot k) +$$

ÜTEMEZŐ Overhead

Kiválasztás ideje

$$+ \sum_k (AL+EL+AS\cdot H_k) + \sum_j (SW+EL+AS\cdot H_j)$$

Allokálás ideje

Swapping ideje

ahol

T_i az i prioritásu feladat válaszideje ms-ban

ST program indítás ideje 50 /us szint

ÜE ÜTEMEZŐ érvényrejutás ideje 180 /us

u a T_i idő alatt bekövetkező várakozások a program befejeződések száma

BTU időalapegység 100 ms

SK STATUS tábla egy elemének vizsgálata 20 /us

k változó értékkészlete a i-nél nagyobb prioritásu programok prioritásszámaiból áll

AL terület allokálás ideje 300 /us

EL minidiszkes elérési idő 10 /us

AS átviteli sebesség 6 /us/byte

Hk k feladat hossza byte-okban

SW a memóriából eltávolitandó program kiválasztásának ideje 1 ms

j változó értékkészlete a tárban lévő és onnan kiviendő programok prioritásaiból áll.

Vizsgáljuk meg a T_i időt a következő esetekben:

A legkedvezőbb eset, az aktiválás pillanatában nincs nagyobb prioritásu program

B átlagos
3 - 4 program már aktiv
mindegyik két várakozást tartalmaz és méretük 0,5 K

C legrosszabb eset
csak diszkrezidens feladatok esetén
16 feladat távolitandó el a tárból összméretük 20 K

A válaszidő
ha a kért program

	a tárban	a diszken	található
A	500 µs	16 ms	
B	2 ms	80 ms	
C	-	500 ms	

A fenti értékek figyelembevételével, valamint emlékeztetve arra, hogy az ütemező minden 100 ms-ban lefut az átlagos adminisztrációs idő a CPU idő 1 %-a és ez az érték rossz esetben 3-4 %-ra nöhet fel.

ADATÁTVITELI VEZÉRLŐ KIEGÉSZÍTÉS R10 MONITOROKHOZ

Rajki Péter
IKTELOR Rendszertechnikai Vállalat

0. BEVEZETÉS

Az adatátviteli vezérlő kiegészítés /AVK/ lehetővé teszi, hogy

- egyrészt a felhasználó az R10 adatátviteli csatolóin keresztül egyidejűleg tetszőleges számú adatátviteli forgalmazást végezhessen;
- másrészt a rendszerprogramozó az R10 batch és real-time jellegü monitorok supervisor moduljai és handlerei alkalmazásával, illetve az adatátviteli vezérlő kiegészítés által különböző vonalak kiszolgálására nyújtott supervisor modulok és handlerek felhasználásával a felhasználó igényeinek megfelelő és minimális tárigényü monitort generáljon.

Az AVK létrehozásakor fordítási direktivák segitségével kell meghatározni a kiszolgált vonalak tipusát, számát /vonalcsoportok kialakitása/, a terminálok speciális jellemzőit, illetve a szolgáltatások szintjét /felügyelet nélküli forgalmazás, eszközkiválasztás, üzenetfej, stb./
Az AVK képes szinkron és aszinkron vonalakon a vonali kapcsolatnak, illetve a kiszolgált terminál, ellenállomás tipusának megfelelő vonali algoritmus szerinti forgalmazásra. A blokkositott vonali forgalmazást a felhasználó logikai szinten rekordonként, illetve blokkonként makróhivások felhasználásával vezérli, ellenőrzi.

Az AVK a következő vonali forgalmazások kiszolgálására képes:

- szinkron kapcsolat
  - IBM BSC: IBM 277o
    IBM 278o
    AP 50
    VTS 5o1oo
  - Univac DCT 2000
  - Siemens TRANSDATA 340
- aszinkron kapcsolat
  - távgépíró /5 és 7+1 bites/
  - display.

Az AVK moduláris felépitése lehetővé teszi, hogy az AVK jelen verzióiban nem specifikált vonali algoritmusokat adatátviteli ismeretekkel rendelkező rendszerprogramozó az AVK alapmoduljai felhasználásával könnyen megvalósitson és az AVK-ba integráljon.

A szükséges vonali kapcsolatokat biztositó AVK létrehozásához a következő komponenseket biztositjuk a rendszerprogramozó számára:

- BIO adatátviteli csatoló kezelők /handlerek/;
- adatátviteli rendszer-, vonalcsoport-, és vonali táblák makróértelmezője;
- adatátviteli sorkezelő;
- vonalkezelő program.

1. AZ AVK ÁLTALÁNOS LEIRÁSA

Távadatfeldolgozási feladatok megoldásánál szükség van a rendszer által kiszolgált különböző tipusú és sebességű adatátviteli vonalak az ellenállomás tulajdonságait is figyelembe vevő vonali algoritmusainak megvalósitására, a forgalmazott adatok logikai szintü kezelésére.

A vonali algoritmus - többek között - előírásokat ad meg:
- az adatátviteli út létrehozásáról /kapcsolat felépitése/;
- az ellenállomással való logikai kapcsolat megvalósitásáról /vonal megnyitása/;
- az üzenetek forgalmazásáról:
  - alkalmazott kódkészlet,
  - vezérlő karakterek, ill. vezérlő szekvenciák,
  - hibaellenőrzés képzése, figyelése,
  - nyugtázások,
  - hibajavitó eljárások,
  - kapcsolat fenntartásának eszközei,
- az ellenállomással való logikai kapcsolat megszüntetéséről /vonal lezárása/;
- az adatátviteli út felszabadításáról /kapcsolat bontása/.

Az AVI lehetővé teszi, hogy a felhasználónak a távadatfeldolgozás adatátviteli problémáival ne kelljen foglalkozni, mivel olyan szolgáltatásokat nyújt, amelyek segitségével a felhasználó a helyi perifériák logikai kezeléséhez hasonló módon vezérelheti a távoli állomásokkal való adatforgalmazást. A felhasználó az adatátviteli forgalmazással kapcsolatos igényeit makróhivások formájában adja meg.
A makróértelmező a hivások formai és tartalmi helyességének vizsgálata után
- egyeseket saját maga hajt végre,
- másokat a vonalvezérlőnek ad át végrehajtásra.
A vonali tevékenységekkel kapcsolatos igényeket vonalanként képzett várakozási sorokba helyezi a makróértelmező.

Az adatátviteli sorkezelő biztositja azt, hogy a felhasználói program az általa kért vonali forgalmazás időviszonyaitól függetlenül futhasson.
A vonalvezérlő biztositja az egyes vonalanként eltérő vonali algoritmusoknak megfelelő adatforgalmazást.

. vonalvezérlőt hívja
- a makróértelmező az azonnali tevékenységek vonali algoritmusfüggő részeinek /vonal megnyitása, vonal lezárása, vonali forgalmazás beindítása, várakozásban lévő vonali forgalmazás újrainditása stb./ végrehajtása céljából,
- az adatátviteli vonal megszakításkezelő rutinja /handlere/ a vonali algoritmus realizálása céljából.

A vonalvezérlő egy felhasználói igény /pl. blokk nyugtázott - sikeres - átvitele/ kielégítése után visszajelzést küld erről a felhasználónak. Az adatátviteli szerkezet lekérdezésével információt szerez arról, hogy van-e újabb felhasználói igény a vonal használatára. Ha igen, akkor megkapja az igény leirását és hozzálát annak vonali algoritmusfüggő végrehajtásához.

A vonali algoritmus paramétereit
- felhasznált adatátviteli kód,
  speciális karakterek, azok értelmezése,
- hibaellenőrzés tipusa,
- hibajavitás módja,
- szinkronizáló és kitöltő karakterek használata,
- időtényezők, stb.

a vonalcsoport- és a <u>vonalleiró táblák</u> tartalmazzák. A vonalvezérlő adja át a szükséges paramétereket az adatátviteli csatoló inditó rutinjának. Az átvitel tényleges végrehajtását az adatátviteli csatolóhoz tartozó mikroprogram vezérli az átadott paraméterek alapján, és tevékenységéről megszakitásokkal értesiti a vonalvezérlőt.

Az IVK minden olyan OS-IO operációs rendszerben felhasználható, mely az előirt supervisor modulokat tartalmazza.

...z AVK-t olyan adatátviteli vonalak kezelésére célszerű alkalmazni, melyeken megvalósítandó vonali algoritmusok bonyolultsága a felhasználó adatátviteli ismereteit túlnövi...

...z AVK-t batch jellegű monitorra /MRB, MRBL, MBM, MBL/ építve a nulláis szinten futó programok vezérelhetik a távadatfeldolgozást.
...z AVK-t real-time monitorra /RTM, RTBM/ építve
- mind a nulláis szinten futó háttérfeldolgozás /pl. batch feldolgozás/ programjai,
- mind a "software vezérelt" megszakítási szinteken célszerűen alrendszerként megvalósított, egymástól függetlenül működő, vagy egymással kapcsolatban lévő programok vezérelhetik a távadatfeldolgozást.

...z AVK lehetőséget ad arra is, hogy ugyanazt a vonalat - természetesen időben nem átfedve - több, különböző szinten futó program kezelje.

2. AZ AVK FELÉPÍTÉSE, VONALI ALGORITMUS FUNKCIÓI ÉS FELHASZNÁLÁSA

2.1 Felépítése

...z AVK három fő program szegmensből és egy adatszegmensből áll. A külvilággal való kapcsolata supervisor szekciók hívásából áll, illetve az AVK-t is két típusú supervisor modul hívásán keresztül lehet elérni: az egyik a felhasználó szám...ra /ilyenkor a hívott rész a felhasználó szintjén fut/, a másik hívás a vonali csatoló kiszolgáló programja számára /ilyenkor a hívott rész a csatoló megszakítási szintjét fut/.

2.1.1 A három program szegmens

- Makróértelmező: **mindig a felhasználó hívja, igy a felhasználó szintjén fut, visszatéréskor hibakódot szolgáltat.**

- Vonalvezérlő: egyrészt a csatoló kiszolgáló programja hívja az egyes megszakítások vonali algoritmusfüggő lekezelésére, visszatérésekor a csatoló kiszolgáló programja tovább elemzi a csatoló működésekor előfordult egyéb megszakítási okokat, és azokat egyenként a vonalkezelőnek adja át lekezelésre. A csatoló szintje a csatolóhoz tartozó összes megszakítás lekezeléséig aktív marad. Másrészt a makróértelmező hívja a vonal megnyitásakor, elindításakor, ilyenkor a felhasználó szintjén fut.

- Adatátviteli sorkezelő: a makróértelmező a felhasználói igények sorbaállítására használja, ilyenkor a felhasználó szintjén fut. A vonalkezelő a vonalanként aktuális igények megismerésére, a már lekezelt igények törlésére használja, ilyenkor a csatoló szintjén fut.

2.1.2 Adatszegmens

Az adatszegmens a három program szegmens konstansait, címeit, a rendszer-, vonalcsoport- és vonaltáblákat tartalmazza. A vonalankénti táblák létrehozása lehetővé teszi azt, hogy a programszegmensek egyszerre több vonal kezelését "újra belépheto" módon megoldják. Az egyes funkciókat megvalósító programok csak egy példányban vannak az operatív tárban és a hivó program közös adatszegmensének munkablokkján keresztül mindig a vonalra vonatkozó táblát használják. Tehát az AVK program szegmenseinek nagysága csak a megvalósitandó funkciók bonyolultságától, adatszegmensének nagysága egy konstans helyfoglalás mellett megközelitőleg egyenes arányban a vonalcsoportok és a vonalak számától függ.

2.2 Vonalalgoritmus funkciói

Az AVK lehetővé teszi, hogy a felhasználó a vonal ismerete nélkül - csak az adatátviteli monitor előirásait figyelembe véve - forgalmazást végezhessen a vonalon.

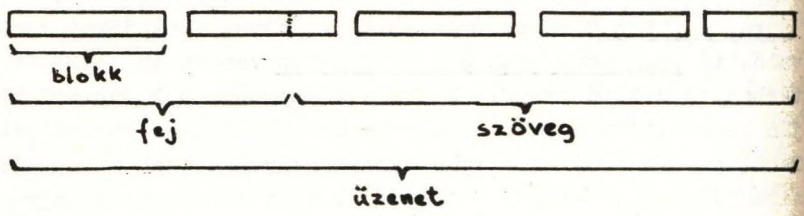
2.2.1 Adatátviteli kód

A felhasználó a forgalmazni kívánt adatokat kódfüggő esetben EBCDIC kódban /az RIO belső kódja/ helyezi az adási pufferekbe, illetve kapja a vételi pufferekben a vonalon használt adatátviteli kódtól függetlenül.

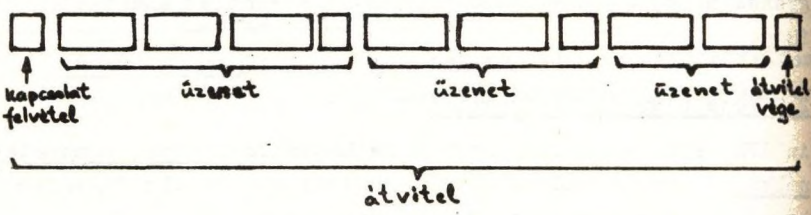
2.2.2 Átvitel felépitése

A felhasználó a vonalon egy kapcsolat fennállása alatt egy vagy több üzenetet forgalmazhat.
Az üzenetek egy vagy több blokkból állhatnak. Minden üzenet állhat fejből és szövegből, pont-pont kapcsolat esetén a szöveg első eleme előtt kimeneti eszköz kiválasztás hajtható végre, többpont kapcsolat esetén a cimzési /lekérdezés és kiválasztás/ szekvenciók adása és fogadása az üzenet első blokkjának forgalmazása előtt történik. /1. ábra/

1. ábra
Átvitel felépitése
Üzenet:



Átvitel:



2.3 Speciális szolgáltatások

A felhasználó adáskor az egy blokkon belüli fejet és szöveget külön fej- és szövegpufferből adhatja, míg vételkor az adatátviteli monitor megadja a vételi pufferben lévő fej hosszát, a szöveg első szignifikáns elemének címét és a szöveg hosszát.

A címzési, lekérdezési, illetve kiválasztási szekvenciák adását, illetve fogadását a felhasználó kezdeményezi, megfelelő kódalakjuk létrehozásáról, vonali algoritmusfüggő végrehajtásukról, az AVK gondoskodik.
Az AVK lehetővé teszi adásirányforditáskérelem adását, illetve vételét. A felhasználó az adásirányforditás-kérelem adásának sikeréről, illetve vételének bekövetkezéséről értesül, és a kapcsolat bontása és újbóli felépitése nélkül képes a megváltozott irányban forgalmazni.
Az AVK lehetővé teszi pont-pont kapcsolatban másodlagos adásjoggal felruházott állomás adáskezdeményezése esetén fellép/het/ő versengés feloldását. Erről a felhasználó értesül, és a kapcsolat bontása és újbóli felépitése nélkül képes vételi forgalmazásra.

Az adási, illetve vételi forgalmazás befejezése /átvitel vége, adásirányforditás kérelem teljesülése/ esetén, ha a befejezés nem járt együtt a vonali kapcsolat fizikai bontásával, a felhasználó ellenkező irányú forgalmazási igényeinek végrehajtását kezdi el az adatátviteli monitor, illetve kész azok fogadására.

2.3 Felhasználása

Az AVK lehetőséget ad a felhasználó/k/nak arra, hogy a rendszerben megvalósitott - eltérő vonali algoritmust és adatátviteli kódot használó, különböző sebességü, tehát eltérő karakterisztikájú vonalakat azonos módon kezel-

hessen/ek/.
Az adatátviteli monitor módot ad arra, hogy egy felhasználó egy vagy több vonalat lefoglaljon, igy ehhez a vonal/ak/hoz más felhasználó addig nem fér hozzá, mig a vonal/ak/at foglaló felhasználó a vonal/ak/at felszabaditja.
A felhasználói program futása a vonal/ak/on folyó adatátvitel időviszonyaitól függetlenül történhet. Ugyanakkor lehetőség van az adatátvitel pillanatnyi állapotának megismerésére, a szinkronizáló pontok létrehozására.

Az adatátviteli monitor az egyes vonalakon forgalmazott blokkok számáról és a forgalmazás alatt előfordult hibák számáról, stb. un. vonali statisztikát készit a felhasználó számára.

3. ADATÁTVITELI MAKRÓHIVÁSOK

A felhasználó a vonalon folyó adatátvitellel kapcsolatos igényeit makróhivások formájában adja meg. Az igény leirását jelen verzióban a regiszterekben, illetve néhány makróhivás esetén egy regiszterrel kijelölt vezérlő blokkban, valamint a felhasználóhoz - mint a vonal pillanatnyi tulajdonosához - tartozó vonalazonositó táblában adja meg. Makróassembler felhasználásával az igények leirása kulcsszavas makróformátumban adható meg.
A felhasználó a vonalat logikai szinten kezeli. A vonal használatának igényét vonalmegnyitó /OPEN/ makróhivással fejezi ki, ami után a vonalat csak ő használhatja addig, mig vonallezáró /CLOSE/ makróhivást ki nem ad.

A vonal megnyitása és lezárása között a felhasználó egy vonalazonositó táblával azonositja magát, ennek címét vagy egy regiszterben, vagy a vezérlő blokk adott helyén minden makróhiváskor megadja.

A vonal megnyitása és lezárása között kiadható makróhívások kétfélék:

- azonnali végrehajtású, vagyis nem tartozik hozzájuk vételi blokk;
- [illegible] végrehajtású, vagyis a felhasználó a vételi blokk elérhetővé válása után kéréssel értesül a makróhívás végrehajtásának befejeződéséről.

A vonalfüggő végrehajtású makróhívások lehetővé teszik a felhasználó számára, hogy

- a vonali forgalmazás befejeződése után /vagy előtt/ bevárja a vonali forgalmazás befejeződését saját mintájának felfüggesztésével /így bevárhatja még a vonal felszabadul/ /WCHKS, WCHKR/;
- adási, illetve vételi indikátor kezelésével a vonali forgalmazás azonnali megszakitását, abortálását kérje /SIS, SIR, RIS, RIR/;
- a vonalon blokkonként forgalmazott adatokat rekordonként adásra összegyüjtse, illetve vételkor rekordokra felbontsa /PUT, GET/.

A vonalfüggő végrehajtású makróhívások lehetővé teszik a felhasználó számára az adatok adatátviteli egységenkénti /blokkonkénti/ forgalmazását:

- adását /SEND típusú makróhívások/;
- vételét /RECEIVE típusú makróhívások/.

4. AVK-T TARTALMAZÓ MONITOR GENERÁLÁSÁNAK FELTÉTELEI

4.1 Szerkesztőprogram

A szerkesztőprogramnak /OS-10-ben: LINK, LINKD/ ismerni kell az AVK saját supervisor moduljának nevét:

M:DTM makróértelmező
M:LINA vonali algoritmus modul.

4.2 Szükséges supervisor modulok

AVK minden olyan OS-10 monitorba beleépitheto, mely az alábbi supervisor modulokat is tartalmazza:

M:IO	I/O művelet inditása,	
M:IIO	I/O művelet lezárása,	
M:WAIT	eseményre várakozás,	
M:DLAY	késleltetés inditása,	
M:SDLY	késleltetés törlése,	
M:ACTV	eseményre várakozó program aktiválása,	
M:RQST	erőforrás lefoglalása,	} ha nincs erőforráskezelés, akkor maskolva futhatnak a megfelelő részek
M:RLSE	erőforrás felszabaditása,	
M:EBAS	EBCDIC-ASCII kódkonverzió,	
M:ASEB	ASCII-EBCDIC kódkonverzió.	

4.3 Adatátviteli csatoló megszakitás kezelő programok /handlerek/

Amennyiben valamely tipusú adatátviteli csatoló /CLS, CLA, COS/ több vonalat szolgál ki, többvonalas handler szükséges.

- Szinkron csatoló /CLS/esetében egy handler két félduplex vonal kezelését végzi. Kettőnél több vonal használatakor a további handlerek csak CDS-ből, LDS-ből és egy ugróutasitásból álló LDB-ből, valamint contextus-ból állnak.
- Aszinkron csatoló /CLA/ esetében egy handler 16 félduplex, illetve duplex vonal kezelését végzi. Ennél több vonal esetében a további handlerek egy ugróutasitásból álló LDB-t igényelnek.
- Communication Oriented Scanner /COS/esetében mind a szinkron, mind az aszinkron vonalak kezelését egy közös handler végzi, mely a maximális vonalszámot kiszolgálja.

4.4 Rendszerleiró forditási direktivák

A kezelt adatátviteli ren szer leirását forditási direktivákkal lehet meghatározni az M:DTM, M:LIM közös LDS-sel ren elkezd supervisor modulok forditása előtt.

A fordítási direktivák lehetővé teszik:
- egyrészt, hogy feltételes fordítással csak olyan funkciók kerüljenek az NVK-ba, amelyek az adatátviteli rendszer kiszolgálásához szükségesek:

- csatoló típusa,
- vonali algoritmus típusa,
- sorkezelés használata,
- fej, címzési szekvencia, hívási indikátor használata,
- kapcsolt - bérelt hálózat,
- pont-pont, illetve többpont összeköttetés,
- erőforráskezelés használata,
- használt adatátviteli kódok, stb;

- másrészt, hogy a rendszer-, vonalcsoport és vonaltáblák a megvalósítandó adatátviteli kapcsolatok előírásainak megfelelően legyenek kitöltve:
  vonalszám csatolótípusonként,
- vonalcsoportok létrehozása és leírása,
- csatolt terminál, ellenállomás definiálása.

INTELLIGENS TERMINÁLOK AZ R10-en

Földvári Iván
INFELOR RENDSZERTECHNIKAI VÁLLALAT

0. BEVEZETÉS

A programozható terminálokat intelligens terminálóknak nevezzük. Ha számitógépalapú terminálokat veszünk figyelembe, akkor ez a feltétel adott. Az intelligenciát tovább növelhetjük az egyes gépek adottságainak megfelelően.

Az R10-es gép multiprogramozási rendszere kinálja a többcélú, nagy intelligenciájú felhasználásokat. Az R10-es gép a speciális adatátviteli - real-time jellegü - software-rel, ez a tipusú felhasználás mutathatja meg az egyik legnagyobb erejét ennek a kiszámitógépnek.

A feladatok megvalósitásakor is tapasztalhattuk, milyen előny az, ha egy bizonyos tipusú felhasználást a gép felépitése, utasitás rendszere, eddigi software bázisa "szivesen fogad". Igy - meghatározásunkat kiterjesztve - az intelligens terminál R10-en való megvalósitásakor képes több vonalon tipustól és vonali protokolltól függetlenül adatátviteli forgalom lebonyolitására, miközben más real-time és számitóközponti feladatokat is elvégez, illetve elvégezhet.

1. A terminálok kialakitása

1.1 A szükséges eszközök

Ahhoz, hogy az R10-en intelligens terminálokat alakithassunk ki, hardware és software alap feltételeknek kellett teljesülni.

A hardware oldalról elmondhatjuk, hogy az R10 gép rendelkezett azokkal az adottságokkal, magas szinten, amelyek lehetővé tették az adatátviteli vonalak kényelmes használatát.

A software oldali alapok úgyszintén biztositva voltak, egyrészt a Real-time monitorok mint vezérlő programok, másrészt a vonali csatolók kezelésére szolgáló handlerek tekintetében. Két francia készitésü terminálszimulátor is segitséget nyújtott a fejlesztéseknél. Az egyik az IBM 2780, a másik a UNIVAC DCT 2000 terminál önálló rendszere volt.

Ahhoz, hogy nagy intelligenciájú, többoldalú adatátviteli rendszerek megvalósulhassanak a software további fejlesztésére volt szükség.

Ezeket az alapokat teremtette meg az INFELOR PRF-en fejlesztett real-time diszkes RTDM és az adatátviteli DTM monitor.

1.2 Terminálok

A kifejlesztésre került terminálokat az elkészités módja szerint két részre lehet osztani:

- az egycélú vagy öntöltős rendszerbe integrált terminálszimulátor
- adott real-time környezetben müködő, terminál szolgáltatásokat nyújtó vonali vezérlőprogram.

Az első tipushoz tartozik a franciák emlitett két rendszere és az INFELOR-ban tervezett AP-50 is.

A rendszerek fő jellemzője, hogy a számitógépet teljes egészében lekötik az adatforgalmazás idejére, önálló rendszerként kerülnek a tárba és un. remote batch terminál-

ként müködnek.

Az AP-50 jelentős előnye, hogy viszonylagos periféria függetlensége az adott I/O eszközre elkészitett formátumkezelő modullal biztositható. Igy az AP-50 olyan remote batch terminálnak tekinthető, amely egyrészt az R10-hez kapcsolt bármely perifériával képes helyi konverziós és adatátviteli feladatok megoldására.

Az AP-50 további előnye, hogy a kapcsolat tipusa szerint:

- pont - pont és
- többpontos kapcsolatban üzemelhet, s lehetőséget biztosit a felhasználónak a felügyelet nélküli adatforgalmazásra.

A második tipusú terminálok már un. foreground feladatoknak tekinthetők egy adott real-time rendszerben. Gyakorlatilag megszakitási szintre kötött, operátor által parancsokkal vezérelhető program, amely képes adatátviteli kapcsolat teremtésére a megfelelő vonal, ill. vonalak és háttértárolók között.

Elvi különbség az egycélú terminálokkal szemben, hogy ezek a vezérlő programok real-time jelleggel müködnek a rendszerben, ahol a többi feladat - akár real-time, akár batch - szimultán müködik.

A most tárgyalásra kerülő intelligens terminál a háttértár szempontjából két tipusú lehet:

- mágnesszalagos háttértárral,
- mágneslemezes háttértárral dolgozó.

A háttértár kezelését két program végzi egy adatfelvivő és

egy adatmegjelenitő program. Ezek a programok batch üzemmódban futtathatók és az adatátviteli vezérlő számára előkészitik, illetve a vezérlőtől kapott információkat megjelenitik adott fizikai I/O eszközön. Ezen programok - a remot batch követelményeknek megfelelően - célszerüen kártyaolvasó bemenetet és sornyomtató kimenetet képzelnek el, de adott esetben a programok periféria függő részei . tetszés szerint cserélhetők.

További lényeges különbség ezen vezérlőknél, hogy az átvitelre kerülő adattömeg egy közbülső háttértárolóra vive várja a végleges feldolgozást.

Az ilyen tipusú felhasználást a következőkkel indokolhatjuk:

1. A perifériális környezet jobb kihasználása. A vonali adatforgalmazás alatt az egyébként is igen lassú perifériák nincsenek még jobban - a vonali sebességnek megfelelően - lelassitva.

2. Szimultán feldolgozást tesz lehetővé. Pl. A kártyaolvasót és sornyomtatót leginkább igénybevevő batch feldolgozással.

3. Megteremti a lehetőségét egy remote- job -entry-nek, ahol futtatható jobok formájába kerülhetnek háttértárra az adatok jobok , majd az eredmények.

4. Alapját képezheti hálózati pontokba elhelyezett RIO vezérlő rendszerének.

2. Jelentősebb software elemek

A jelenleg legfejlettebb intelligens terminálok elő-

nyei a következőkben összegezhetők:

- több, eltérő vonali protokollal rendelkező adatátviteli vonal egyidejü kezelése,
- real-time jelleg,
- háttértáros feldolgozás,
- alrendszerként történő vezérlés.

A több, jelenleg mx. 4 eltérő vonali protokollal rendelkező adatátviteli vonal egyidejü kezelését megalapozta az adatátviteli monitor DTM vonalkezelő supervisora és a vonali csatoló handlere a rendszer oldaláról, a vezérlőprogram részéről pedig a program ezen szempontok szerint való fejlesztése.

A real-time jelleget a real-time monitor RTDM kibővitett szolgáltatásai tették lehetővé, úgymint az eseménykezelés, várakozási listakezelés, erőforráskezelés.

A háttértáras feldolgozást, annak dinamikussá tételét a real-time monitorba épitett file-kezelő rendszer és egy a vonali forgalmazást kiszolgáló speciális supervisor szekció adta. Ezen szekció segitségével automatizálhatta a vezérlőprogram a dinamikus diszk felosztást és az adott vonalakhoz tartozó file-ok kezelését.

Az alrendszerként történő vezérlés annyit jelent, hogy a rendszer kapcsolatot biztosit az operátori kommunikáción keresztül az adott megszakitási szinteken müködő programokkal. Ezt úgy teszi, hogy az ezen a szinten lévő program az adott karaktersorozatot, célszerüen parancsot megkapja és a program szintje a parancs megérkezésekor aktivizálódik. Ezeket a szolgáltatásokat ismét a real-time

monitorba RTDM épitett különböző programrészek biztositották.

3. A terminálok működésének leirása

Ha áttekintünk a fent felsorolt terminálokon a működés a fejlettség szerint rendre igy következik:

öntöltős rendszerek: IBM 2780
DCT 2000
AP-50

önálló real-time feladatok:
intelligens MT
intelligens DC

Az öntöltős rendszer üzembe helyezése:

a. a rendszer tárba töltése
b. a rendszer számára operátori parancsok kiadása
c. a perifériák működőképes állapotba hozása
d. a forgalmazás megindítása

Az önálló real-time feladatokként működő vonali vezérlők üzembe helyezése:

a. A már működő, dolgozó rendszer FOREGROUND zónájába töltjük operátori paranccsal a vezérlő programot és működőképes állapotba hozzuk.
b. Egy, az alrendszernek szóló operátori paranccsal elinditjuk a vonali forgalmazást.
c. Mindezen müveletek alatt a többi feladat zavartalanul működik a gépen.
d. Az adatok háttértárra kerülését, illetve azok háttértárról való megjelenitését előzőleg, illetve később

egy batch-ban lefuttatott job-bal végezhetjük.

3. A fejlesztés iránya:

A jelenlegi munkák is előrevetitik az aszinkron vonalak kezelésének igényét. Ez koncentrátori és interaktiv üzemmódban kell, hogy müködjön igy a többvonalas vezérlő modularitását felhasználva a lassú vonalak kezelését is belátható módon biztositani lehet.

További fejlesztésnek tekinthető, ha a terminál intelligenciája a fejlett remate-job-entry felé közeledne, ahol a job-ban lévő egyes részek módositásával más terminálok, nagy gépek felé ismét jobokat lehet majd generálni a futó job igénye szerint.

S/4R-10 AZ ICL SYSTEM 4 INTELLIGENS TERMINÁLJA

Ságody István és Gellért János
Országos Tervhivatal Számitástechnikai Központja

A FEJLESZTÉS ELŐZMÉNYEI

A számitástudomány jelentőségének felismeréséből kiindulva, a hatvanas évek végén a kormány programot fogadott el a számitástechnikai eszközök gyártására és az alkalmazások elterjesztésére. A program előrehaladásával, valamint az eszközök fejlődésével lehetővé vált a célkitüzések további bővitése. Az uj feladatok között egyik legfontosabb az államigazgatás korszerüsitése. Az erre vonatkozó határozat, uj technikai lehetőségként, már figyelembe vette a távadatfeldolgozást is. Igy alakult ki a bázisközpontok koncepciója.

Az Országos Tervhivatal Számitástechnikai Központja - mint az államigazgatás egyik bázisközpontja - a fenti határozat alapján, a Számitástechnikai Központi Fejlesztési Program keretében kezdte meg a System 4-70 tipusu központi gépéhez kapcsolódó terminálrendszer kifejlesztését. Ez lehetővé teszi, hogy a kijelölt, felsőbb igazgatási szervek a hazai gyártásu, R-10 kisgép mellől vegyék igénybe az OTSZK szolgáltatásait. Ennek a fejlesztésnek az eredménye az S/4R-10 terminál, amelyet a következőkben ismertetünk.

AZ S/4R-10 TERMINÁL FOGALMA, SZOLGÁLTATÁSAI

Az S/4R-10 terminál az ICL System 4 központi gép intelligens terminálja. Az intelligens jelző arra utal, hogy a terminál programozható, hardware konfigurációjának és vezérlő programjának változtatásával, bővitésével különböző funkcionális követelmények kielégitésére képes. Lényeges a központi gép kiemelése, mert a terminál vezérlő programjának kommunikációs

elemei speciálisan az ICL System 4 kommunikációs rendszeréhez illeszkednek, a terminál szolgáltatásai pedig kifejezetten a System 4 MULTIJOB operációs rendszerét tételezik fel.

Az 1. számu ábrán látható kommunikációs kapcsolattal az R-10 a System 4 terminál jaként müködtethető. Emellett az R-10-nek is lehetnek saját terminál jai /irógépek, képernyős megjelenitők/, illetőleg egyidejüleg más számitógéppel is lehet terminál-kapcsolatban. A sokféle lehetőség megkivánja, hogy egyértelmüvé tegyük a következőkben használandó terminálfogalmat.

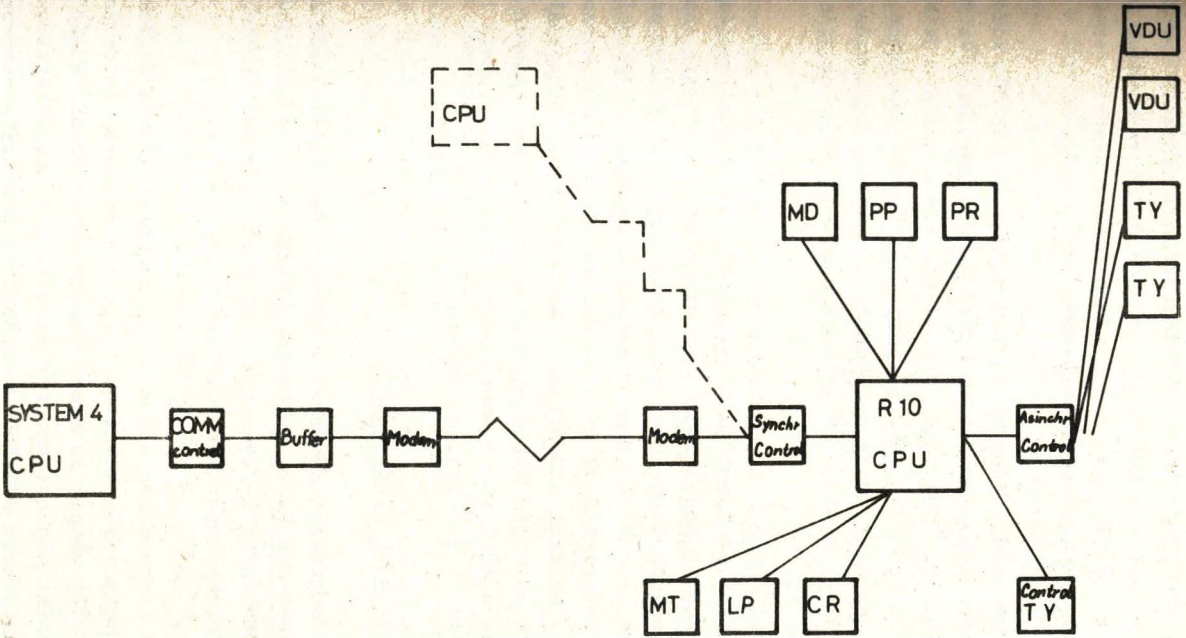
Az S/4R-10 terminál logikai fogalom, amely a MULTIJOB operációs rendszer üzemeltetése alatt az ICL System 4 központi géphez kapcsolódó, következő komplex terminálrendszert jelenti:

a./ távoli adatvégállomás,

amely saját adatperiférial /olvasó, lyukasztó, nyomtató/ és a központi gép között kapcsolóberendezésként müködik; a kommunikációs kapcsolattal folyamatos adatátvitel kiszolgálására alkalmas

b./ az adatvégállomáshoz rendelt párbeszédes terminálok, amelyek mindegyike a System 4 önálló, párbeszédes terminál jaként használható; nagyobb mennyiségü adat átvitelét a közös adatvégállomásra kérhetik

c./ a párbeszédes terminálok közül egy kitüntetett, az adatvégállomás vezérlő konzoljának kinevezett párbeszédes terminál, amely egyéb funkciói mellett, az adatvégállomás müködésével kapcsolatos parancsok kiadására, illetőleg üzenetek, hibajelzések vételére szolgál.



1.ábra

Az S/4R—10 TERMINAL- KONFIGURÁCIÓ

A System 4 MULTIJOB operációs rendszerének mindazon szolgáltatásai, amelyeket a különböző, standard ICL terminálok felhasználóinak nyujt, együttesen elérhetők az S/4R-10 terminál mellől. Röviden ezen szolgáltatások hátteréről:

- A MULTIJOB prioritásos multiprogramozással müködő, időosztásos operációs rendszer. A konfiguráció particiói /streamjei/ közül egy vagy több időosztásos kiszolgálásra jelölhető ki; a többiben batch-feldolgozás folyik, az egyes particiókon belül is multiprogramozva. A multiprogramozás szempontjából minden időosztásos partició egyetlen szintet képvisel.

- A kommunikációs supervisor biztositja, hogy a párbeszédes terminálokat a felhasználói programokból interaktiv input/output berendezésként lehessen használni. A MULTIJOB ezen kivül két speciális, kommunikációs alrendszerrel rendelkezik az adatterminálok és a párbeszédes terminálok kiszolgálására. Mindkét alrendszer a kommunikációs felhasználói programok szintjén müködik, rezidens részükkel azonban beleépülnek a MULTIJOB kommunikációs rendszerébe, amely ilyen módon nyitott és szabadon bővithető.

Az adatterminálok kiszolgálása

A MULTIJOB operációs rendszer a távoli adatvégállomások számára a Remote Job Entry funkciókat biztositja. A végállomás kiválasztott input perifériája a központi gép Job Inputhoz rendelt, forrás-input berendezését helyettesiti.

A MULTIJOB rendszerben a Job Input a munkaleirások paramétereinek feldolgozásán kivül biztositja a rendszer információbázisához való védett és szabályozott hozzáférést is. Az információbázist mágneslemezeken tárolt, a rendszerkatalógusban nyilvántartott file-ok alkotják. Az operációs rendszer

programoktól független manipulációs lehetőséget is biztosit ezen file-okhoz.

Igy a forrás-input berendezésről beolvasott file-ok bekerülhetnek a rendszer nyilvántartásába, és később a System 4-esen futó programokból, vagy a létrehozó terminálról ujra elérhetők. Az igy létrehozott file-ok tartalmuk szerint lehetnek:

- munkaleirások /Job Control paraméterek/,
- forrásnyelvi modulok a forditóprogramok számára,
- próbaadatok, kisebb adatfile-ok.

Az inputhoz hasonlóan, a végállomás kiválasztott output perifériája a központi gép output berendezését helyettesiti. Az operációs rendszer minden adatvégállomáshoz fenntart egy várakozó sort a kérdéses terminálon kinyomtatandó file-ok számára. Ezek tartalmuk szerint lehetnek:

- a kérdéses terminálról kezdeményezett programok futtatásával létrehozott eredményfile-ok,
- az adatvégállomás, illetőleg az onnan kezdeményezett programok aktivitásával kapcsolatos események jegyzéke, üzenetek /ilyenek az RDT Journal, Job Journal beleértve a Post Mortem Dump lehetőségét, a PRINT Queue tartalma, stb./,
- általában az információbázisban tárolt és nyilvántartott bármely más file, amelyet a terminál felhasználója hozott létre, vagy amelyhez a hozzáférés számára engedélyezett.

Egyébként bármely file kiiratása kérhető a központi gép sornyomtatójára is.

A párbeszédes terminálok kiszolgálása

A MULTIJOB operációs rendszer egy speciális parancsnyelv se-

gitségével lehetővé teszi, hogy interaktiv módon vegyük igénybe a rendszer szolgáltatásait. A parancsnyelv főbb lehetőségei:

- a párbeszédes terminálok mellől kezdeményezhető interaktiv programok futtatása, amelyek választ, döntést váró üzeneteiket a kezdeményező terminálra küldik;

  az információbázishoz való közvetlen hozzáférés, azaz uj file-ok létrehozása, meglévők lekérdezése, módositása, törlése;

- uj programok /forrásnyelvi modulok/ forditásának, szerkesztésének kezdeményezése, interaktiv tesztelése;

- már kipróbált programok futtatásának kezdeményezése akár háttérfeladatként /batch/, akár időosztásos kiszolgálással; futó programok felfüggesztése, vagy erőszakos befejezése;

- a párbeszédes terminálok asztali számológépként is használhatók aritmetikai és függvénykifejezések, formulák kiértékelésére;

- egyszerübb programok irására, futtatására a párbeszédes terminálok mellett rendelkezésre áll az interaktiv BASIC-nyelv.

A parancsnyelv lehetőségei közé tartozik az adatvégállomás vezérlő konzoljának kezelése is. A párbeszédes terminálok közül erre a kitüntetett terminálra érkeznek az adatvégállomás müködésével kapcsolatos üzenetek, illetőleg innen lehet kérni egy, a központi géptől érkező file nyomtatásának felfüggesztését, vagy folytatását. Egy file kinyomtatását lehet véglegesen törölni, vagy későbbre halasztani. Minden, folyamatban lévő nyomtatás ujra kezdhető a file elejétől, vagy meghatározott lapjától.

A FIZIKAI MEGVALÓSITÁS

Az S/4R-10 terminál tényleges megvalósitásának alapja a hazai gyártású, VT 1010 /R-10/ kisgép. Konfigurációja a funkcionális követelményektől függ. Minimális kiépitésben elegendő egy-egy input és output periféria, valamint a konzol-irógép. Szélesebb körü szolgáltatáshoz és "intelligensebb" viselkedéshez memóriabővités, háttértárak és saját párbeszédes terminálok szükségesek. A konfiguráció kiépitettségétől függően a vezérlőprogram különböző változatai állithatók elő.

A szolgáltatások teljes körét biztositó kiépitésben az R-10 terminált a központi géppel két adatátviteli vonal köti össze. Ebben a kapcsolatban az R-10 a System 4 számára transzparens, front-end processzor, amely kettős funkciót lát el:

- többszörös pufferhasználattal, illetőleg háttértárak közbeiktatásával biztositja a lassu perifériák és az átviteli vonal teljesitményének maximális kihasználását,
- a párbeszédes terminálok üzeneteit egy vonalra koncentrálja.

Ezzel a megoldással lehetett elkerülni a System 4 kommunikációs rendszerének módositását. Az R-10-re készitett kommunikációs software igy a System 4 termináljait utánozza a megfelelő vonali algoritmusok átvételével. Ezek a következők:

- az ICL 7020 RDT vonali algoritmusa az adatvégállomás kezeléséhez,
- az ICL 7180 VDU a 7182/2 QLSA-val /Line Sharing Adaptor/ vonali algoritmusa a párbeszédes terminálok kezeléséhez.

Mindkét vonali algoritmus többpontos kapcsolatra készült, cimezhető terminálokat tételez fel.

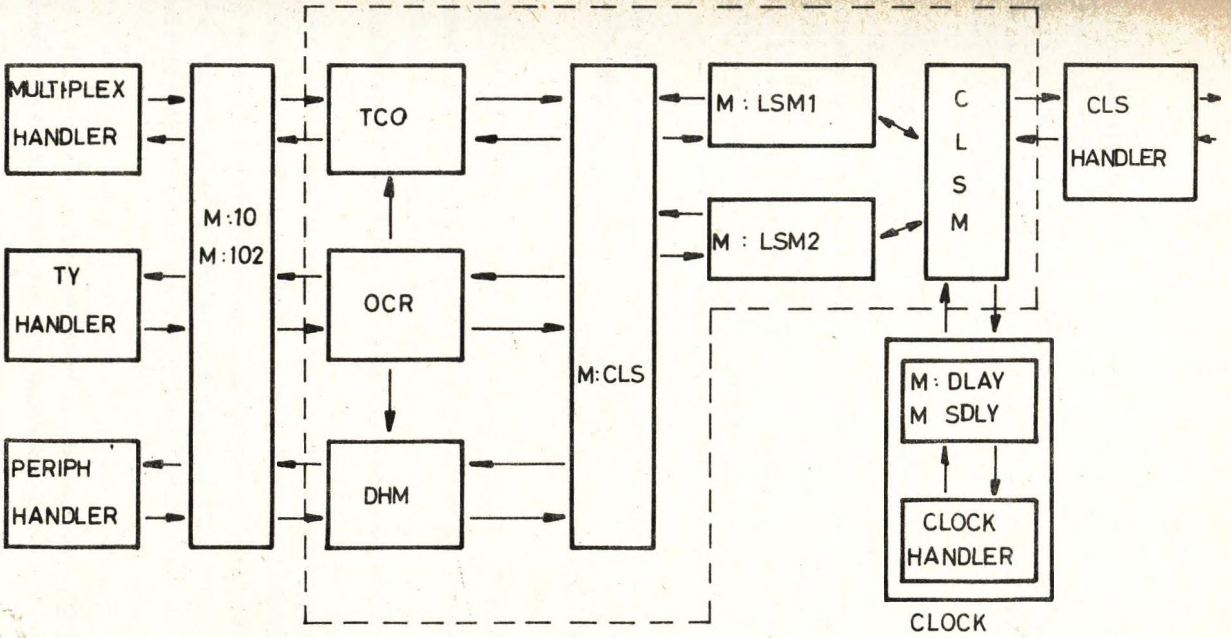
Az RDT vonalán két-két input és output periféria, valamint a vezérlő konzolirógép küldő vagy fogadó állapotának kiválasztására van lehetőség. A terminál status-információt küld a cimzés, kiválasztás /polling/ lépéseiben, valamint minden adatblokk vételének nyugtázásánál. A kezdeményezés, a küldő vagy fogadó állapot /master-slave/ beállitása, a periféria kiválasztása a központi gép joga. Ezt a kiválasztást azonban a status-információkkal befolyásolhatja a terminál is.

A párbeszédes terminálok /display-k/ kezelésénél a cimzés, kiválasztás lehet specifikus, egy meghatározott berendezésre vonatkozó, vagy általános, a lekérdezés sorrendjében az első aktiv terminálra.
Különválik a status-információk, valamint az adatblokkok átviteléhez szükséges cimzés és kiválasztás.

Az R-10 terminál és a System 4 központi gép adatátviteli kapcsolatához tehát két, félduplex módban müködő telefonvonal szükséges; az egyik vonal az adatvégállomás, a másik a párbeszédes terminálok kiszolgálására. Az R-10 mikroprogramozott, szinkron-adatátviteli vezérlőegysége /CLS/ is két félduplex vonal kiszolgálására képes egy adott szinten.

Az adatátvitel jellemzői mindkét vonalra azonosak:

adatátviteli kód:	ISO /7 bit + paritásbit/
átviteli mód:	szinkron
átviteli teljesitmény:	1200/2400 bit/s
blokkhossz:	változó,
	adatterminálnál max. 80 karakter,
	display esetén max. 2008 "
blokk-ellenőrzés /BCC/:	hosszanti blokkparitás
A vonali interface:	CCITT V24



2.ábra

Az S/4R—10 TERMINÁL VEZÉRLŐPROGRAMJA

AZ S/4R-1O TERMINÁL VEZÉRLŐPROGRAMJA /TCP/

Az S/4R-1O terminál vezérlőprogramján /Terminal Control Package/ összefoglalóan azt a programozási rendszert értjük, amely a kisgép monitorának bővitésével biztositja a kivánt terminálfunkciókat. A változó követelményekhez alkalmazkodva moduláris felépitésü, a következőkben ismertetendő elemekkel.

A System 4 központi gép és az R-1O terminál közötti adatátviteli vonalak kiszolgálását az R-1O Real Time monitorába beépülő, uj Supervisor-modulok látják el a szinkron adatátviteli handler /CLS/ utján. A CLS-handler több vonal párhuzamos kiszolgálására alkalmas. A két System 4-es vonali algoritmusnak megfelelő vonalkezelő modul az M:LSM1 és M:LSM2, a közös kommunikációs interface az M:CLS modul.

Az S/4R-1O terminál funkcióinak ellátásához szükséges további programok közvetlen /real-time/ feladatként, a szabad megszakitási szintekhez kötve müködnek. Ezek:

DHM adatkezelő modul a terminál adatátviteli feladataihoz,

TCO a párbeszédes terminálok üzenetváltozásainak kezeléséhez,

OCR operátor-interface az S/4R-1O terminál vezérlő konzoljának müködtetéséhez.

A konfiguráció kiépitettségétől függően az S/4R-1O terminál szolgáltatásainak szükebb vagy bővebb korét, különböző területeit érhetjük el. Ezekhez különböző TCP-változatok tartoznak, amelyek az előbbiekben felsorolt modulokból állithatók össze. Az alapváltozatok a következők:

TCP1 változat: az adatvégállomás funkcióit biztositja, beleértve azonban egy vezérlő konzolként müködő párbeszédes terminált is /fizikailag az R-1O

konzolirógépe/.

Ez a változat a következő moduláris egységekből állitható össze: M:CLS, M:LSM1, DHM, OCR-TCO. Ebben a változatban az egyetlen párbeszédes terminál, a vezérlő konzol kezeléséhez a TCO-nak egyszerüsitett változata áll rendelkezésre, célszerüségi okokból az OCR-rel egybeépitve.

TCP2 változat: a párbeszédes terminálok koncentrált kiszolgálását végzi.

Ebben a változatban a párbeszédes terminálok között nincs kitüntetett. Ezekről a terminálokról nem kérhetünk nyomtatást az adatvégállomásra, mert az a System 4 számára nem létezik. Ezen változat moduláris egységei: M:CLS, M:LSM2, TCO, OCR.

TCP3 változat: az S/4R-10 terminál szolgáltatásainak teljes körét biztositja; alapjában a TCP1 és TCP2 változatok egyesitése.

Moduláris egységei: M:CLS, M:LSM, DHM, TCO, OCR. Ebben a változatban az M:LSM modult a két vonalkezelő modul egybeépitéséből kapjuk. Rétegezett felépitése lehetővé teszi, hogy a hasonló funkcióból következő közös részek ne ismétlődjenek.

Az S/4R-10 terminál megvalósitása az R-10 kisgépre épül, amelynek teljes konfigurációját a helyi feldolgozások köthetik le, amikor nincs a központi géppel adatátviteli kapcsolatban. A helyi feldolgozást azonban a kommunikáció sem zárja ki; minthogy a TCP kommunikációs komponensei, az S/4R-10 terminál funkcióit biztositó programok közvetlen /real-time/ feladatként magasabb megszakitási szinteken müködnek. Az R-10-es processzor az adatátviteli feladatokkal egyidőben, háttérfeladatként, helyi batch-feldolgozást is végezhet a

0-ás szinten.

A KOMMUNIKÁCIÓS INTERFACE /M:CLS/

Az M:CLS Supervisor-modul a kommunikációs átvitelek kezeléséhez ugyanazt a logikai szintet biztositja, mint az R-10 monitoraiban a helyi átvitelekhez rendelkezésre álló, M:IO modul. Megvan a formai hasonlóság is; a kommunikációs átvitelekhez is esetenként egy vezérlőblokkot kell összeállitani. Ez tartalmazza az átvitelhez hozzárendelt puffer cimét, az átviendő, illetőleg ténylegesen volt byte-ok számát, az átviteli parancs és a vonal kódját, stb.

Az adatátviteli vezérlőblokkban is egy un. eseménybyte jelzi, hogy az átvitel még folyamatban van, illetőleg sikeresen vagy meghatározott tipusu hibával befejeződött.
Az R-10 kommunikációs felhasználói programjaiból adható adatátviteli parancsok, a vonali algoritmusnak megfelelően, a következő főbb funkciók ellátására szolgálnak:

a kivánt adatátviteli vonal megnyitása és bezárása, amelyek hatására a terminál "élővé válik", válaszol a cimzésre, illetőleg megszakitja a kapcsolatot a központi géppel;

küldő és fogadó adatátviteli állapot kezdeményezése;

adatblokkok küldése, illetőleg fogadása; a küldő állapot bezárása;
a párbeszédes terminálok aktivitásának jelzése;
a párbeszédes terminálok üzeneteinek fogadása és küldése.

A VONALKEZELŐ MODULOK /M:LSM/

A vonalkezelő modulok alapfeladata, hogy a vonali algoritmusok keretein belül

a System 4 számára biztositsák az R-10 transzparens jellegét, az R-10 kommunikációs programjai számára pedig elvégezzék a vonalkezelés részfeladatait, mint a vezérlő üzenetek kezelése, az átviteli hibák elháritása, stb.

Elvben két különböző vonalkezelő modul van a felhasználói interface és az átvitelek fizikai kezelését végző CLS-handler közé beiktatva, az adatvégállomás és a párbeszédes terminálok vonali algoritmusának megfelelően. Ezek hasonlósága miatt a két vonal kezelésének közös részfunkcióit a rétegezett software megfelelő szintjein közös rutinok látják el. A két vonal kezelésére alkalmas TCP3 változatban a két modul igy egybeépül.

A vonalkezelő modulok automatikusan elvégzik a sikeres vétel nyugtázását, hiba esetén pedig a vonali algoritmus biztositotta ciklikus eljárással megkisérlik az átviteli hiba elháritását. Az egyes átvitelekhez előirt késleltetés értékével beállitják a real-time órát, hogy vonalhiba esetén is visszakapják a vezérlést.

A TCP KOMMUNIKÁCIÓS PROGRAMJAI

A vezérlőprogramnak a kommunikációs interface kisgép-oldalán három alapfeladatot kell ellátnia:
az adatátvitel és a párbeszédes terminálok kiszolgálása, valamint a Terminál operátorával való közlekedés.
Ezeket már a felhasználói programok szintjén, három közvetlen feladatként müködő modul végzi.

Az adatkezelő modul /DHM/ a TCP alapváltozatában a központi gép és az R-10 adatperifériái közötti, blokkonkénti átvitelek logikai szintü kezelését végzi. Ez az adatkezelés alapváltozata abban az értelemben, hogy a DHM a vett és küldött adatblokkok lényeges átalakitása nélkül, csupán közvetitő

szerepet tölt be. A file formátumát, az adatblokkokat a vételnél is változatlanul hagyja, csak a sornyomtató vezérlő karaktereit cseréli fel a helyi berendezéseknek megfelelően.

Minthogy egyidejüleg csak egyirányu adatátvitel folyhat, egy meghatározott tipusu adatperifériával, a DHM szegmensei, az irányitó alapszegmens kivételével, egymást átlapolják.

Az operátor-interface /OCR/ biztositja az S/4R-10 vezérlő konzoljánál a terminál operátora és a TCP közötti közlekedést. Az OCR kezeli a terminál müködésével kapcsolatos parancsokat és üzeneteket.
Minthogy az adatvégállomás vezérlőkonzolja célszerüen az R-10 konzolirógépe, ezek a parancsok és üzenetek az R-10 standard monitorparancsai, illetőleg üzenetei bővitésének tekinthetők.

A TCP-parancsok a következőkre terjednek ki:

a kijelölt adatátviteli vonal megnyitása és lezárása, adatátviteli kezdeményezése az input periféria kijelölésével, kapcsolat kezdeményezése, illetőleg üzenet küldése valamely párbeszédes terminálról.

A TCP üzenetei részben a parancsok értelmezésével kapcsolatosak, részben az elhárithatatlannak bizonyult átviteli hibákról értesitenek.

A párbeszédes terminálok kezelése /TCO/

A TCO a System 4 és az R-10 terminálok, valamint az adatvégállomás vezérlő konzolja és a Terminálok közötti üzenetek közvetitésével az R-10 saját párbeszédes terminál jainak kiszolgálását végzi.
Dinamikus pufferkezeléssel biztositja a változó hosszuságu üzenetek /néhány karaktertől a teljes képernyő tartalmáig/

átmeneti tárolását. Saját táblázataiban nyilvántartja az egyes terminálok állapotát, aktivitását, kezdeményezését, stb. Ennek alapján állitja össze a megfelelő adatátviteli parancsokat a kommunikációs interface-hez. Vizsgálja a központi géptől jövő üzeneteket, hogy eldöntse választ várnak-e valamint a terminálóktól jövő üzeneteket, hogy szétválassza őket rendeltetési helyük szerint /a központi gép, vagy a vezérlő konzol/.

A TCO megvalósitásának legfőbb problémája az R-10 képernyős berendezéseinek kezeléséhez rendelkezésre álló software.

A TCP FEJLESZTÉSE

Az OTSZK-ban folyó fejlesztés során az S/4R-10 terminál vezérlőprogramjának TCP1 változata 1974-ben készült el. Jelenleg a TCP3 változat megvalósitása folyik. 1975 végéig már két R-10-es terminál kapcsolódik az OTSZK System 4-70 központi gépéhez.

A TCP3 változattal az S/4R-10 terminál funkciói, a System 4 MULTIJOB jelenlegi szolgáltatásait illetően kimerültek, de a fejlesztési lehetőségek természetesen nem. A programozható R-10 terminál programozási rendszerének bővitésével, valamint MULTIJOB operációs rendszer fejlesztésével lehetővé válik a szolgáltatások finomitása, a terminál "intelligenciájának" növelése. A fejlesztés a két rendszer közötti munkamegosztásra törekszik. Az ehhez vezető fontosabb lépések: a filekezelés, elő-, utófeldolgozás, programkapcsolás.

Háttértárolókkal bővitett terminálkonfigurációval a TCP-ben megvalósitott file-kezelés eggyel magasabb logikai szintet biztosit az adatátvitel kezeléséhez. Ez az adatkezelő modul müködését is eltakarja.

R-10-es felhasználói programokból olyan output file hozható létre, amelynek rendeltetési helye a System 4 információbá-

zisa. A TCP ezen file-okat az R-10 háttérmemóriájában tárolja, majd a létrehozó programtól függetlenül gondoskodik átvitelükről. A file-t csak akkor törli, ha a teljes file átvitele sikeresen befejeződött.
Ennek forditottjaként az R-10 párbeszédes termináljairól lehet kérni a System 4 információbázisában tárolt file-ok áthozatalát az R-10 háttérmemóriájába, további helyi feldolgozásokhoz.
Az előbbi értelemben vett file-kezelés megvalósitásán kivül szükség van a vonalkezelés olyan módositására, amely lehetővé teszi a kódfüggetlen átviteleket is /ennek hiánya System 4 hardware korlátozás/. Ez már alapot teremt a két gép közötti munkamegosztásra.

Egyszerübb módja az elő- és utófeldolgozás. Az adatvégállomáson a System 4-en futtatandó programokhoz összeállitott input file-oknál adatellenőrzés, konvertálás, az ellenőrzött rekordokból kompresszált formátumu file kialakitása. A System 4-ről érkező kompresszált eredmény file-ok dekompreszszálása, a nyomtatási kép kialakitása, szerkesztése.

Erre épülhet az automatikus programkapcsolás. Megvalósithatók olyan programrendszerek, amelyek egyes programjai meghatározott sorrendben erőforrásigényüktől függően a terminálon, illetve a központi gépen futnak.

ESZR SZÁMITÓGÉPEK - ELSŐSORBAN R 20 - AS - TELJESITŐKÉPESSÉGÉNEK NÖVELÉSE

Dr. Bindics Ferenc
Déldunántuli Áramszolgáltató V.

A nagyrészt adatfeldolgozási jellegü munkákat végző ESZR berendezések teljesitőképességének felső határát a lassú input-output készülékek áthocsátó képessége határozza meg. Ezeknek a készülékeknek /papirperifériák/ teljesitménye messze elmarad a számitógépek egyéb részeinek teljesitményétől.

A kártyaolvasó, nyomtató perifériák és a rendszer más részei közötti teljesitményben ellentmondás van, igy igen jelentős várakozási idők képződnek és a központi egység jó esetben is csak 20-25 %-ra van terhelve. A rendszer kihasználása ilyen adottságok mellett távolról sem kielégitő. Az operációs rendszer készitői a multiprogramozási lehetőség biztositásával igyekeztek ezt az ellentmondást részben kiküszöbölni, azonban a multiprogramozási lehetőségnek igen sok hátránya van.

A multiprogramozás rendszere ideálisnak látszik abban az esetben, ha az egyidőben a tárban levő programokat úgy tudjuk összeválogatni, hogy központi egység és periféria igényük ellentétes legyen, ebben az esetben látszólag maximális kihasználást biztositottunk.

A multiprogramozás technikája sajátos problémákat hordoz magában. Egyik ilyen a tárterület felosztása.
A tár nagysága adott, azonban a felhasználókat nem, vagy nagyon nehezen korlátozhatjuk.

Az u.n. fejlett programnyelvek /PL/I tárigénye egyenes arányban növekszik szolgáltatásaikkal. A sok felhasználót kiszolgáló gépparknál igen kis valószinüséggel tudjuk összeegyeztetni úgy a programok tárigényét, továbbá a multiprogramozás service jellegü előfeltételeit, hogy egyáltalán sor kerülhessen multiprogramozás alkalmazására. Az adatfeldolgozási feladatok általában perifériaigényesek és ma még kevés az a számitóközpont, ahol bőven állnak rendelkezésre perifériák, pedig az is alapfeltétele a multiüzemmódnak.

A másik nagy probléma a perifériaigényes, illetve központi egység igényes programok multi-üzemmódban való futtatása. Itt még komolyabb kellemetlenségek lépnek fel. Az üzemeltető /gépterem/ nem is tudja, hogy a felhasználó programja milyen jellegü. Ezen úgy lehetne segiteni, hogy a futási lapokon rögzittetjük, periféria vagy központi egység igényes a program. Itt azonban ujabb probléma van. Nem tudjuk megmondani, hogy hol a határ az igények szempontjából, tehát mi tekinthető központi egység vagy periféria igényesnek. Továbbá bonyolitja a kérdést, hogy egy és ugyanazon produktumot nyujtó program igényjellege attól is változik, hogy milyen programnyelven irtkák, nem beszélve arról a már furcsának tünő megállapitásról, hogy az igényjelleg a programozó személyiségétől is függ.

Az ESZR /elsősorban R 2o-as/ gépek többségének tárkapacitása 64 ill. 128 KByte, abból következően a rendszert ilyen nagyságú gép oldaláról kell vizsgálnunk.

Lehet e ilyen tárkapacitás mellett, nagy tárigényü programnyelvet /PL/I, stb./ használva egyáltalán a multiprogramozás lehetőségéről beszélni. Elméleti megfontolások és a gyakorlati tapasztalat is azt mutatja, hogy multiprogramozású mint rendszeres és állandóan használt üzemmód szóba sem jöhet. A tapasztalati számok azt mutatják, hogy ilyen adott-

ságokkal rendelkező számitóközpontok többprogramos üzem - módja a havi 500 órás gépidőből max. 10-20 óra között mozig, ami nem éri el az 5%-ot sem. Az elméleti lehetőség ennél természetesen nagyobb lenne, /nem sokkal/ azonban ebben az esetben el kellene tekinteni az egyszerüen elsajátitható programnyelvek használatától és a meglevő programokat át kellene irni a géphez közelebb álló nyelvre.

Lehetne természetesen a tárakat böviteni, ez azonban tetemes anyagi ráforditást jelentene és megalapozottnak látszó vélemények szerint igy sem kerülnénk alapvetően jobb helyzetbe.

A feladat megoldására azonban szükség van és amennyiben ehhez a jelenlegi rendszer /DOS/ adottságai nem megfelelőek, uj rendszert kell teremteni, amely az adottságaink mellett számunkra optimális.

Az optimális értelmezéséről Starr a következőket irja:
..."egyensúlyt kell találni a rendszer egymással szembenálló tényezői /vagy céljai/ között. A döntésnek általában a létesitendő és uzemeltetendő rendszer általános hasznosságát kell maximalizálnia."

Jelen esetben akkor beszélhetnénk a multiprogramozás optimális voltáról, ha időben tudná egyeztetni a különböző jellegu feladatokat, már pedig erre a rendszer képtelen.
Éppen a fentiek miatt ki kell mondjuk, hogy a supervisort esetünkben at kell értékelni.'

Több éves nehéz körülmények között végzett /irodalom hiánya, forrásnyelvi programok hiánya, stb./ kutatómunka után megállapitható volt , hogy a DOS felugyelo programjainak /supervisor/ egy része távolról sem gazdaságos helykihasználással készült. Ennek nyilván az az oka, hogy a rendszert igen nagy-

letszámú kollektiva készitette a feladatokat szegmentálva, igy a részek közötti kapcsolat megteremtése bonyolult és helyigényes lett.

A tények arra utalnak, hogy a programozás technikájában rejlö egyik ellentmondással állunk szemben,

vagy: kevés programozó, kis helyigényü programok, gyorsan futó program, hosszú elkészülési idö,

vagy: sok programozó, nagyobb helyigényü program, lassúbb programfutás, rövidebb elkészülési idő

ahol a második megoldás került kivitelezésre.

Amennyiben ezt az ellentmondást jobban szervezett supervisorral fel tudjuk oldani, úgy hogy az eredeti rendszer használatát nem korlátozza, helyet nyerhetünk az operativ tár olyan részén, amelyhez a felhasználó egyáltalán nem tud hozzáférni. E területen supervisor állapotban el tudnánk végezni a lassú perifériák adatmozgatását felhasználói program nélkül olyan helyzetben, amikor a központi egység egyébként várakozik. Igy nem vennénk el egyetlen byte-nyi területet sem a felhasználótól és igen tetemes mértékben meg lehet növelni az géppark átbocsátóképességét. Minden jól szervezve egyesitené magában a multi üzemmód legfontosabb előnyeit, annak hátrányai nélkül. Megszünne vagy nagymértékben csökkenne az adatfeldolgozásra jellemzo I/O probléma és pszeudó perifériák alkalmazásával tulajdonképpen növelnénk a periférikészletet is. /pszeudó perifériának nevezem a helyettesitést, vagy fizikailag egyáltalán nem létező perifériát/

Igy tulajdonképpen a DOS kiegészitésével uj feldolgozási módot vezetnénk be az eddig ismerteken kivül, továbbá biztositanánk az eddig a DOS által nyujtott lehetőségeket is. /batch, multiprogramozás, időosztás/

A probléma megfogalmazása

Olyan input-output rendszerre van szükség, amely a lassú I/O meggyorsitásával rendszertechnikai uton lehetővé teszi a gépek átbocsátóképességének növelését. A megoldás ne vegyen el a felhasználó számára hasznos tárterületből egyetlen byte-nyit sem, egyszerü kezelhetőséget biztositson és ne változtasson a programozás eddigi rendszerén.

A probléma megoldása /csak vázlatosan/

Az ESZR/DOS - al kompatibilisen kidolgozásra került egy több supervisor aktivizálására alkalmas rendszer. Az egyik supervisor képes a "Rapid" /fantázia név/ funkció ellátására is a következőképpen:

- a feladatprogramot képes függetleniteni a papirperiférik lassúságától
- lehetővé teszi pszeudó periférik alkalmazását
- a kártya - szalag, szalag - nyomtató müveleteket hasznos tárterület lefoglalása nélkül, továbbá programozás nélkül hajtja végre. A müveleteket a központi egység várakozói idővel végzi, tehát a mellette futó felhasználói programot nem zavarja.

Rapid koncepció

Az eljárás 7. féle feladatnak tegyen párhuzamosan eleget anélkül, hogy ez a felhasználót akár időben, akár tárterületben zavarná.

Supervisor állapotban és területen

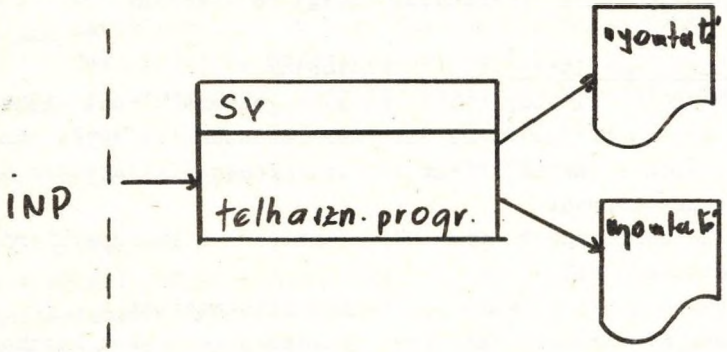
1./ A felhasználói programtól vegye el a papir outputot
2./ A felhasználói programtól vegye el a papir inputot
3./ Kezeljen egy mágnesperiféria - papirperiféria adattransfert
4./ Kezeljen egy papirperiféria - mágnesperiféria adattransfert
5./ Adjon lehetőséget pszeudó periférik használatára
6./ Biztositson a mágnesperifériákon ésszerü információtömöritést

7./ Tegye lehetővé az R 2o-as gépek ékezetes betü nélküli nyomtatóhengerein az ékezetes betük nyomtatását.

Fejtsük ki egy kicsit bővebben az egyes feladatokat.

1./ A felhasználói programtól vegye el a papiroutputot

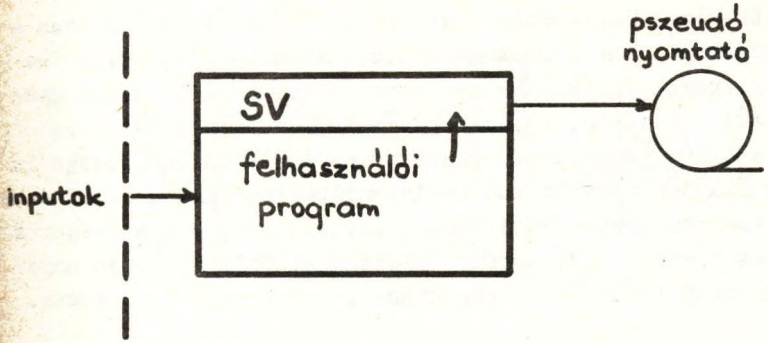
a./ A feladat megoldása Rapid nélkül



Látható, hogy a lassú nyomtató/k/ határozzák meg a feladat végrehajtási idejét. /Itt szeretnék kitérni az ESZR rendszer nyomtatási teljesitményére. A nyomtatók elméleti /fizikai/ teljesitőképessége 6oo-9oo sor/perc. A LIOCS müködése a nyomtató kezelésénél távolról sem ideális, ugyanis 1 sor kinyomtatásához 2 utasitást használ, egy nyomtató utasitást és egy sorhuzó utasitást. A LIOCS-nak ilyen formán való müködése a nyomtatók eredeti /elméleti, fizikai/ teljesitményét 25-3o %-kal csökkenti. Erről a felhasználó nincs értesülve.

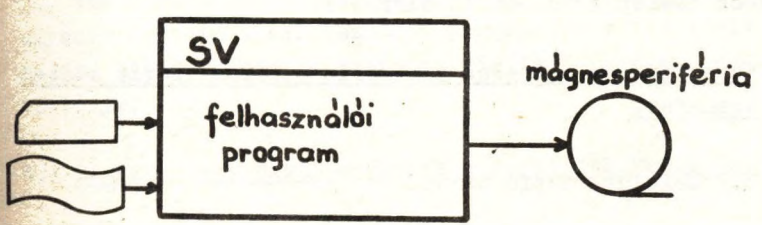
A Rapid eljárásban egyetlen utasitás végzi el a feladatot és bizonyithatóan megemelkedett a nyomtatók teljesitménye, illetve megszünt az indokolatlan teljesitménycsökkenés.

b./ Megoldás Rapid funkcióval



Ennél a megoldásnál nincs szükség soronként és sorhuzásonként csatornaprogram /EXCP/ inditására és lekezelésére, ellenkezőleg a Rapid a supervisor területen összegyüjti a nyomtatandó sorokat /a sorokban levő üres helyektől függően 6-25 sort/ és tömöritve egy csatornautasitással mágneses perifériára irja.
Ilyen esetben meg van a lehetőség 2-3 nyomtató használatára függetlenül attól, hogy a gépteremben nincs 1-nél több nyomtató /Lásd: paszeudó perifériák és tömörités/. A nyomtató és a szalag sebességkülonbségéből adódó időnyereség minimálisan 50 %.

2./ <u>A felhasználói programtól vegye el a papirinputot</u>
a./ Megoldás Rapid nélkül



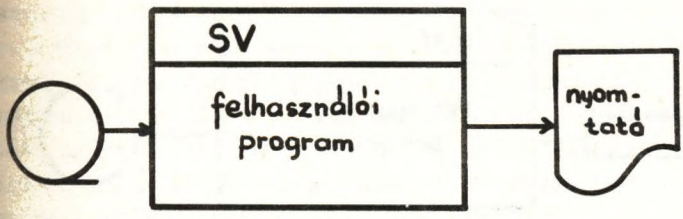
A szük keresztmetszetet ilyen megoldásnál minden esetben a lyukkártyaolvasó lassúsága okozza. Elméleti sebessége 500 kártya/perc. Ennek kihasználására azonban nincs meg a gyakorlati lehetőség, ugyanis programnyelvek használata esetén a LIOCS csak olvasó utasitás és dupla puffer esetén is csak 250-300 kártyát tud beolvasni percenként. A LIOCS minden esetben várakozó állásban tartja a központi egységet a teljes kártya beolvasásáig, ezután kezdődik a kártya anyagának feldolgozása és csak utána az uj kártya beolvasása.

b./ Megoldás Rapiddal

A korábban supervisoron keresztül központi egység várakozási időben blokkolva szalagra vitt /4.funkció/ és tömöritett kártyaanyagot a supervisor blokkoltan olvassa 10-50 kártya anyagát egy olvasó utasitással, majd a sajat területén deblokkolja, eredeti formájára szethúzza és igy adja át a felhasznaló programjának. Első pillanatra látszik a nagymértékü időcsökkenés, ami a gyakorlatban kozepes kártyafile-ok esetén a 70%-ot is elérheti.

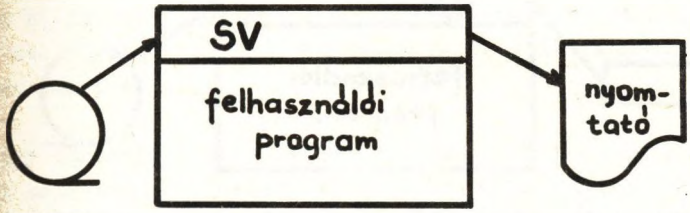
3./ Kezeljen egy mágneses periféria papirperiféria adattranszfert

a./ Megoldás Rapid nélkul



Ennél a megoldásnál a nyomtató határozza meg az egész rendszer kihasználási fokat, Mivel a nyomtató az egyik leglassúbb periféria, látható a nagy időigény.

b./ Megoldás Rapiddal



A korábban mágneses perifériába rögzitett nyomtató file-t vagy file-okat /1 funkció/ a Rapid a kozponti egység várakozó állapotaiba supervisor területen keresztül kinyomtatja, emellett a felhasználó programja zavartalanul futhat, ugyanis a Rapid a felhasználói program várakozó ideit használja. Látható, hogy a transfer "időfelhasználás nélkul" történik.

4./ <u>Kezeljen egy papirperiféria - mágnesperiféria transfert</u>

a./ Megoldás Rapid nélkül

Ilyen megoldásnál ismét az az állapot, hogy a lassú kártyaolvasó lefogja a központi egységet.
A szükséges időt ismét lassú periféria határozza meg.
A gép kihasználása igen rossz.

b./ Megoldás Rapiddal

Ez a Rapid funkció a felhasználó programban várakozó állapotaiban végzi el a feladatát /blokkolja, tömöriti a kártyát és mágnesperifériára irja/ mikozben a felhasználó programja zavartalanul futhat.
Hasznos időfelhasználásra nem kerül sor.

5./ Adjon lehetőséget pszeudóperiférиák használatára

a./ Megoldás Rapid nélkül

Megoldás Rapid nélkül nincs, ugyanis olyan periféria, amely nincs a gépteremben, tehát nem került be a logikai készüléktáblába, nem használható. Ez igen hátrányos a következő esetekben:

- hiányzó nyomtató

/Pl. hatékonyan és gyorsan megoldható feladat 2-3 nyomta-

tóval, pl. számlázás, számlaösszesítés, statisztika/, azonban csak 1 vagy 2 nyomtató áll rendelkezésre. Erre az esetre részleges megoldást kinál az operációs rendszer, de csak perifériafüggetlen programozásnál, így is nagyon sok perifériát kell igénybe venni. /minden nyomtató helyett egy másikat, pl. szalag/. Természetesen így is külön kell gondoskodni a nyomtatásról /külön menetben, külön programmal/ és a mágnes periféria kihasználása is igen rossz /1 sor = 1 blokk/.

- elromlott nyomtató/k/
  Különlegesen felkészitett programok nélkül a nyomtató/k/ miatt leáll az egész rendszer.

b./ Megoldás Rapiddal

A Rapid eljárás fizikai szinten kezeli a rendszert, ami azzal az előnnyel jár, hogy amit a logikai rendszer hibás megoldásnak tekint és nem is hajlandó folytatni a munkát, /két különböző file egyszerre akar egy perifériát használni, stb./ ezt az eljárás bizonyos feltételek fennállása esetén akceptálja.

Ez úgy történik, hogy a PIOCS /fizikai I/O rendszer/ fel lett készitve nem létező csatornákon és nem létező alcsatornák /azaz nem létező készülékek/ kezelésére. Ilyen pszeudóperiféria használatakor az operátor több pszeudóperiférához hozzárendelhet egy tényleges perifériát és a felhasználói program függetlenül attól, hogy milyen nyelven iródott és azonos időben kivám több perifériát használni, egy létező perifériához van kényszeritve.

Nézzük meg hogyan müködik a Rapid az előző többnyomtatós példánál. Induljunk ki abból a hipotézisből, hogy két nyomtatóra van szükség, de csak egy van és az sem müködik. Az operátor /programozó/ pontosan úgy kell hogy használja a rendszert, mintha a nyomtató/k/ müködésképesek lennének, mindössze annyit kellotennie, hogy a nyomtató fileokhoz pszeudóperifériát jelöl ki mindkettőhöz, vagy mindháromhoz

egyet, amely mágneses periféria is inditja a programot. A Rapid gondoskodik róla, hogy 2-3 nyomtató file nyomtatandó sorai megfelelő blokkolással kerüljenek szalagra, mégpedig egy blokkon belül olyan sorrendben, ahogy a tárgyprogramból a SVC 0 megszakitások érkeznek.
Igy természetesen meg nem állapitható sorrendbe kerülnek a különböző nyomtató fileok sorai a mágneses perifériára. Egy blokkon belül is elhelyezkedhet különböző nyomtatóra szánt sor, de ezen kivül bármilyen kombináció előfordulhat.

A több nyomtatóra szánt file fizikailag 1 filenek van kezelve, tehát csak a file végén van végjel. A munka befejezése után tetszőleges időpontban a Rapid 3-as funkciójával a szalagon levő információ kinyomtatható az eredetinek szánt formában, természetesen program nélkül és úgy, hogy közben felhasználói program futhat. A nyomtató file-ok különválasztását a Rapid 3. funkciója elvégzi és a kivánalomnak megfelelő fileot nyomtatja.

A Rapid a funkciójának úgy tud eleget tenni, hogy a szalagra irás előtt az SVC 0 megszakitásakor megvizsgálja melyik pszeudóperifériát akarja a rendszer használni és a pszeudóperiféria számát /Pl. X'01E'/ megjegyzi a felirandó blokk minden rekordjánál.

A módszerrel kiküszöbölhető a gyakran fennálló probléma is, hogy egy feldolgozást a példányszám miatt /pl. banki inkasszók/ többször kelljen megismételni, ugyanis egyszeri programfutás után tetszőleges azonos példány nyomtatható.

Ismét hangsúlyozom, hogy mindezt a supervisor végzi holtidőben, tehát nem veszi igénybe különa központi egységet sem és a felhasználó tárterületét sem.

6./ Biztositson a mágneses periférián ésszerü tömörítést

a./ Megoldás Rapid nélkül

Az ESZR operációs rendszerek esetenként lehetővé teszik, hogy periféria függetlennek deklarált file-ok esetén a nyomtatóra szánt információkat mágneses perifériára tegyük. Ezt azonban ezek a rendszerek igen rossz helykihasználással hajtják végre. Pl. a mágnesszalag esetében a nyomtatandó sor, tehát maximum 156 karakter kerül egy blokkba és minden blokk közötti blokköz /fizikailag a fékezéshez és gyorsitáshoz szükségesek/ sokkal nagyobb helyet foglalnak, mint az értékes információ.

Megvizsgálva egy átlagos nyomtatandó sor tartalmát, megállapitható, hogy az adatfeldolgozásnál az olvashatóság és formázás miatt hozzávetőleg a sorba irható karakterek 2o-4o%-a üres hely /space/. Belátható, hogy az üres helyek teljesen feleslegesen iródnak, tárolódnak, stb. Ezzel ismét csak a szalag kihasználtsága romlik.
Ezekre a problémákra Rapid nélkül nincs megoldás.

b./ Megoldás Rapiddal
A blokkolatlan felirásból adódó gazdaságtalan helykihasználást a Rapid 5. funkciója már megoldotta. Az üres helyekkel kapcsolatban egy soron belül elegendő lenne a felirás előtt megjegyezni, hogy két értékes információ között hány db space fordul elő és csak ezt felirni. A nyomtatáskor a felirt információból megállapitható, hogy hány db üres karakter helyezendő el a sorban, ennek végrehajtása után a sor az eredeti formajában nyomtatható. A Rapid 6. funkciója éppen ezt teszi.

7./ A rendszer tegye lehetővé az R 2o-as gépek ékezetes betük nélküli nyomtatóin az ékezetes betük nyomtatását.

a./ Megoldás Rapid nélkül

Az ékezetes betük nyomtatására az R 2o-as gépeknél nincs lehetőség, a nyomtatóhengeren nincs az ékezetes betü kiképezve. Ez a gyártó szempontjából érthető, hiszen el kellett helyezni a latin abc betüi mellett a cirill betüket is, továbbá nem gyárthatnak minden felhasználónak különböző iróhengereket. Ez azonban igen jelentős kellemetlenséget okoz, magyar személynevek, helységnevek, stb. nyomtatásánál. Erre igen sok példát lehetne hozni, de most ezektől tekintsünk el.

b./ Megoldás Rapiddal

A megoldás feltétele az volt, hogy a rendszer végezze a feladatot programozói beavatkozás nélkül. A megoldást a következő gondolat hozta meg. Tulajdonképpen az adott ékezetes betük /A,O,E,U/ felbonthatók egy-egy ékezet nélküli betüre /A,O,E,U/ és egy-egy ékezetre /','',',''/, tehát ha kinyomtatjuk az ékezet nélküli betüket és sorhuzás nélkül utána az ékezeteket, megkaptuk az ékezetes betüt.

A folyamat: A felhasználó programjában, amikor nyomtatás szükséges supervisor osztályú megszakitás jön létre, kiadásra kerül az SVC 0. Ekkor a Rapid elveszi a vezérlést a supervisortól, továbbá átveszi a regiszterből a nyomtatandó sor tárbeli cimét. Ezután a tárbeli cimen megvizsgálja /egyetlen utasitással/, hogy szerepel-e a sorban különleges jel. Az ékezetes betük belső kódja mindig valamilyen különleges jelnek felel meg. Ha nincs /nem szükséges ékezetes betü/, visszaadja a vezérlést a supervisornak és szabályos megszakitás lefolyás jön létre. Amennyiben talál különleges jelnek megfelelő bitkombinációt, két lépést hajt vére:

- a különleges jelek helyett a nyomtatandó sorban beállitja az ékezetes betük valamelyikének alapbetüjét /A,O,U,E/ és a saját supervisor területén létrehoz egy sort, ami csupa üres helyből áll, kivéve az A,O,U,E betük soron belüli cimeit, ahova elhelyezi a /','',/ jelek egyikét.

- ezután a megszakitást ténylegesen lekezelő LIOCS-nek egy cim helyett /a ténylegesen nyomtatandó sor cime/ címlán-cólást alkalmazva két cimet ad át a vezérléssel együtt a tényleges sor és a pszeudó sor cimét.
Ezután a LIOCS szabályosan végrehajtja a csatornaprogramot két sor kinyomtatására sorhúzás nélkül. A probléma megoldódott és a nyomtató teljesitménye sem csökkent észrevehetően, ugyanis ha nincs szükség átalakitásra, összesen néhány utasitás kerül végrehajtásra.

Elérhető megtakaritások Rapid rendszer alkalmazásával /R 2o esetén/

- minden kártyafelvitelnél legalább 5o%
- minden nyomtatás esetén legalább 5o%
- a mágnesszalagok kihasználtsága kártyánál 2-2,5-szeres
- a mágnesszalagok kihasználtsága nyomtatónál 1,5-2,5 "

Ezzel minden számitóközpontban /hangsúlyozva a Rapid rendszer más előnyeit is/ megtakaritható egy kártyaszala, szalagnyomtató MDS rendszerü off-line berendezés.

Egyéb pozitivumok

- külső okok miatt megszakadt feldolgozásokat nem kell elölről kezdeni, hanem a megszakadás helyétől folytatható
- több példányos tablók /melyeknél a kivánt példányszám nem nyomtatható egy menetben/ programjainak lefutására csak egyszer van szükség, a többszöri nyomtatást Rapid funkció végzi és ez alatt a teljes tárterületen a felhasználó dolgozhat.
- lehetőség van pszeudó perifériák alkalmazására. Igen gyakran előfordul, hogy egy adott feladatprogram gazdaságos megoldása 2 vagy több nyomtatót igényelne, ez azonban nem áll rendelkezésre. A Rapid lehetőséget nyujt, hogy 2-3 nyomtatót helyettesitsünk egy szalag-egységgel.

A rendszer készitői bizonyitani tudják, hogy a Rapid funkció célszerü alkalmazásával 1-1 adatfeldolgozó K 2o-as gép átbocsátóképessége legalább 2o%-kal megnövekszik.

AZ IBM OS/MFT IMPLEMENTÁLÁSA MEGHATÁROZOTT ÜZEMELTETÉSI KÖVETELMÉNYEKNEK MEGFELELŐEN

Garainé Erhardt Anikó

Központi Statisztikai Hivatal
Számitástechnikai Igazgatóság

Az IBM nagy számitógépeken - eltérően más cégek gépeitől - a használni kivánt operációs rendszert a felhasználó generálja. Ez azt jelenti, hogy a rendszer tulajdonságait bizonyos határok között a felhasználó szabadon választhatja meg, sőt, üzemeltetési tapasztalatok birtokában módjában áll a felhasználónak uj, hatékonyabb operációs rendszert generálni.
Az itt következő ismertetésben a rendszergenerálás általános, a paraméterezést meghatározó problémáiról és azok konkrét megoldásának módjáról lesz szó.

1. A rendelkezésre álló környezet

A Központi Statisztikai Hivatal 1974-ben egy IBM 370/155-ös nagy számitógépet szerzett be.
A számitógép hardware konfigurációja jelenleg a következő:

- 512 K központi memória
- 8 cserélhető mágneslemezes egység /egyenként 30 millió byte kapacitással/
- 8 mágnesszalagos egység két vezérlőegységgel /800/1600 BPI, 320 KB átviteli sebesség/
- 2 sornyomtató /132 pozició, 1100 sor/perc/
- 1 kártyaolvasó /1200 cpm/
- 1 konzolirógép /85 cps/

Ehhez a géphez OS/MFT /Multiprogramming with a Fixed Number of Tasks/ 21.7 verziójú rendszert generáltunk, kiegészitve a HASP /Houston Automatic Spooling Priority/ 3.1 verziójával.

2. Az operációs rendszer generálásának főbb szempontjai

A generálás alapvető szempontjait az határozta meg, hogy a gépen statisztikai adatfeldolgozást kell végezni.

A statisztikai feldolgozások néhány sajátossága.

A statisztikai adatfeldolgozásra jellemző, hogy nagyon nagy adattömegekkel dolgozik. Ez azt jelenti, hogy az adatok tárolásának, biztonságos megőrzésének, elérésének módja a meghatározója annak, hogy egy-egy feldolgozás mennyire hatékony, gazdaságos.
Másik jellemző tulajdonság az, hogy a feldolgozások nagy része egyedi, tehát egyfajta tábla kiirására készitett program egyszer fut.
Természetesen vannak ismétlődő, havonta futó anyagok, de ezek nagy része is változik egyik évről a másikra.
Az egyedi anyagok között sokszor fordul elő hasonlóság, sokszor kis változtatás elég egy programon ahhoz, hogy uj, másfajta táblát lehessen vele elkésziteni.
Ebből a tulajdonságból következik az az üzemeltetési adottság, hogy a program módositások és próbák aránya a gépen futó összes feldolgozáshoz képest viszonylag nagy, és ez az arány egyes időszakok kisebb eltéréseitől eltekintve állandó.
A statisztikai feldolgozások legnagyobb része nyomtatásigényes, tehát a több óráig tartó kiirásokat a job-októl függetlenül kell tudni vezérelni.

Nagyon sok rendezés fordul elő a különböző fekvésben kiirt táblák készitése között, ezért a rendszer tervezésekor számitásba kell venni a job-ok várható lemezes és szalagos munkaterület-igényét.

További szempontok

Az operációs rendszer generálásánál a fent emlitettekkel összhangban a következő üzemeltetési szempontokat vettük figyelembe:

- A generált operációs rendszer a lehető legkevésbé legyen merev, hogy szükség esetén uj generálás nélkül lehessen rendszerbeli változtatásokat eszközölni.
- Legnagyobb prioritása a program módositásoknak, forditásoknak és rövid futási idejü, raktárból előkészitendő adathordozót nem igénylő job-oknak legyen.
- A gép maximális kihasználásának érdekében a teljes átbocsájtóképesség növelésére törekszünk akkor is, ha ez egyes job-ok egyetlen futását tekintve esetleg hátráltatja azt.
- Az alkalmazott operációs rendszerben /OS/MFT/ meg kell határozni az egymás mellett futtatható job-ok maximális számát /ahány partició van, annyi job futtatható egyszerre/.Ezt mi tapasztalatainkra és nagy gyakorlattal rendelkező rendszerprogramozók tanácsaira támaszkodva négynek választottuk.
- Az adathordozó állomány hatékony használata érdekében központositott adatállomány kezelést kivántunk bevezetni olymódon, hogy az adatállományok nyilvántartása a rendszerkatalógus alapján történjék.

- A programpróbákat megkönnyitendő közös program-könyvtárakat hoztunk létre. A rendszert ugy kellett megterveznünk, hogy ezek a könyvtárak állandóan online legyenek, és hogy bárki olvashassa a könyvtárat, de egyszerre legfeljebb egy irás vagy módositás történhessék.

. A jelenlegi konkrét megoldás

Partició, job-osztály, prioritás

Ezekről a fogalmakról az IBM irodalomban részletes leirás található, itt csak a későbbiek megértéséhez elengedhetetlenül szükséges rövid ismertetés következik.

Az IBM OS/MFT-ben a job-ok végrehajtásának ütemezése /scheduling/ a particiókra és a job-osztályokra épül. Az operációs rendszer állandóan a memóriában tartózkodó része a Supervisor. A fennmaradt központi tárolóterület áll a felhasználói programok rendelkezésére. Ezt a területet kell rendszergeneráláskor particiókra osztani. A particiók méretét bármikor, ha nem fut a gépen job, megváltoztathatja a gépkezelő, számukat azonban nem növelheti.
Természetesen a legcélszerübb olyan partició-felosztást generálni, amelyet ritkán kell megváltoztatni.
Az egyes particiókhoz job-osztályokat kell generáláskor rendelni. Minden particióhoz lehet több job-osztályt, és minden job-osztályt lehet több particióhoz rendelni. A particiókhoz tartozó job-osztályokat is, hasonlóan a memória-mérethez, meg tudja változtatni a gépkezelő, de a cél az, hogy minél ritkábban legyen erre szükség.

Részben emiatt, részben a gépkezelés megkönnyitése érdekében vezettük be a HASP-ot. Az emlitetteken kivül még számos más előnnyel is jár a HASP használata, amit mi sem bizonyit jobban, mint hogy az IBM uj, virtuális gépre készült operációs rendszere, a VS a HASP számos elemét átvette.
Egy job osztályát a felhasználó adja meg a job vezérlő kártyákon, illetőleg, ha nem adja meg, a job A osztályu lesz.

A beolvasott job a neki megfelelő osztályu várakozó sor végére kerül.
Ha van olyan partició, amelyik éppen üres, és a hozzá rendelt osztályoknak megfelelő várakozó sorok valamelyikében van végrehajtásra váró job, akkor ez kerül sorra.
A particióhoz tartozó job-osztályok hozzárendelési sorrendje igen fontos: a másodiknak hozzárendelt osztályba tartozó job csak akkor kerül végrehajtásra, ha az elsőnek a particióhoz rendelt job-osztály várakozó sora kiürült.
A job-oknak prioritás is adható, ez az azonos osztályu job-oknak a várakozó sorban elfoglalt helyét, sorrendjét befolyásolja.

Azt, hogy egy feldolgozási rendszerben mi az optimális partició és job-osztály felosztás, csak hosszu tapasztalat és nagymennyiségü futási adat kiértékelése után lehet - ha lehet egyáltalán - meghatározni.
Egy viszonylag hatékony rendszer azonban néhány hónapos üzemeltetési adat birtokában és sok-sok megfontolás, egyéb szempont figyelembe vétele, megvitatása és eldöntése után kialakitható.

A szükséges technikai változtatások keresztülvitele után az egyik legnagyobb problémát az operációs rendszer generálásakor a job-osztályok kialakitása jelentette.

A job-osztályok kialakitása.
Alapvető szempont az volt, hogy minél több várakozó sort alakitsunk ki, tehát minél jobban "szét kell szórni" a job-okat. Igy biztositható ugyanis a gép leghatékonyabb kihasználása, a legnagyobb átbocsájtóképesség.
Másik szempont, hogy minél kevesebb erőforrást vesz igénybe egy job, annál nagyobb legyen a prioritása.
Harmadik szempont, hogy a cikk elején ismertetett konfiguráció szük keresztmetszete a mágneslemez, ezért a job-osztályok kialakitásának egyik meghatározó tényezője a lemezhasználat.

A partíciók mérete a következőképpen alakult ki: a Supervisor és a HASP betöltése után 360 K központi tároló maradt a felhasználói programok számára. Ezt a területet kellett négy particióra osztani.

A programkönyvtárak kialakitása és használatának módja, valamint az a tény, hogy a programpróbák nagy prioritással kell, hogy fussanak, meghatározta az első partició funkcióját és méretét. A forditóprogramokhoz igazodva az első particiót 110 K byte körüli méretünek kellett választanunk.
A maradék 250 K-t többféleképpen oszthattuk el. Feltételeztük azt, hogy állandóan mind a négy partícióban fog futni job, ezért nem választhattunk egyetlen particiót sem kisebbnek, mint amekkorában az iniciátor

elfér. Ez a mi rendszerünknél 56 K. Ha volna ennél kisebb partició a generált rendszerben, ugy abban csak akkor indulhatna egy job, ha egy másik, az iniciátor méretének megfelelő partició is üres lenne. Tekintettel arra, hogy a programpróbáknál az első particióban 110 K memória áll a felhasználó rendelkezésére, a felhasználó kényelmének érdekében generáltunk még egy 110 K-s particiót.

A jelenlegi gépre generált első operációs rendszerben tehát a következő particióméretek voltak:

110 K, 56 K, 94 K /ez a maradék/, 110 K.

Néhány hónappal az üzembeállitás után szükségessé vált a rendszer módositása. Igy kerültek ujra szóba a particióméretek. A felhasználók nehezményezték, hogy a futtatásra használható particiók /illetőleg a felhasználók számára ezek mint job-osztályok jelentek meg; erről a későbbiekben lesz szó/ mind különböző méretüek.

Megnéztük tehát annak a lehetőségét, hogy a particiók közül három 100 K-s és egy 60 K-s legyen. A fordítóprogramok is elférnek 100 K-ban, természetesen a memóriacsökkenés hatékonyságuk rovására megy.

A felhasználói programokra vonatkozólag a következőképpen szereztünk adatokat:

az SMF /System Management Facilities/ rutin által gyüjtött üzemi adatokból megfelelő programok segitségével kiiratható a gépen futott programok memóriaigénye. Minden job-nál a legnagyobb memóriát vettük figyelembe, és összevetettük ezeket az adatokat a job-ok futási idejével. Két hét anyagát átvizsgálva a következő eredményre jutottunk: A munkáknak kb. egynegyede program próba; több mint 60 %-a 70 K-nál kevesebb memóriát igényel, kb. 11 %-a 70 és 110 K között van,

a fennmaradó néhány százalék particióösszevonást igénylő job.
Ennek eredményeképp a partícióméretek igy alakultak:
110 K, 70 K, 70 K, 110 K.

A mágneslemez használat a következőképpen befolyásolta a job-osztályok kialakitását:
Maga az operációs rendszer a programkönyvtárakkal együtt három lemezcsomagot foglal el, ezeken a lemezeken egy kevés munkaterülettől eltekintve nincs szabad hely.
Három lemezcsomagot állandóan online használunk /rezidens lemezek/, ezekre a felhasználók azokat a kis adatállományaikat teszik, amelyekre gyakran van szükségük, és itt viszonylag sok munkaterület áll rendelkezésre temporális állományok részére.
A maradék két lemezegységre tetethetik fel a felhasználók az egyéb lemezes állományaikat tartalmazó lemezcsomagokat.

Az első particióban biztosan nem futhat lemezt igénylő job, tehát a hozzá rendelt job-osztályok egyik jellemzője ez lesz.
A másik három particióban alkalmasan választott job-osztály sorrenddel elérhető, hogy ritkán forduljon elő lemezcserére való várakozás, ha az alapértelmezés szerinti job-osztályok legfeljebb egy lemezt használhatnak.
Ha egy job két lemezt igényel, particióösszevonást kell végezni.
A job-osztályok elnevezésénél megtartottuk a már korábban használt neveket: az A osztály az alapértelmezés, ezért az átlagos igényü job-ok osztálya, a

B a programpróbáké, a C a sok erőforrást igénylőké, a D a kevés erőforrást igénylő job-ok osztálya.

Igy a következő job-osztályokat alakitottuk ki:

H - program módositás, forditás céljára, 110 K, 2 mágnesszalag

B - program módositás, forditás, próbafutás céljára, 110 K, 2 mágnesszalag

D - csak rezidens lemezt használó job, 70 K, 3 mágnesszalag

G - csak rezidens lemezt használó job, 110 K, 3 mágnesszalag

A - egy cserélhető lemezt igénylő job, 70 K, 3 mágnesszalag

E - egy cserélhető lemezt igénylő job, 110 K, 3 mágnesszalag

Ezek a rendszer alapértelmezése szerinti job-osztályok Particióösszevonással kialakithatók a következők:

M - csak rezidens lemezt használó job, 180 K, 6 mágnesszalag

C - két cserélhető lemezt igénylő job, 180 K, 6 mágnesszalag

N - két cserélhető lemezt igénylő job, 250 K, 6 mágnesszalag

Az egyes particiókhoz tartozó job-osztályok a következők /a lemezt használó osztályok alá vannak huzva/:

P1	HBDG		
P2	DA		P2+P3+P4 N
P3	AD	P3+P4 MC	
P4	GEDA		

További generálási paraméterek.
A lemezhasználat meghatározásánál ügyelnünk kellett arra is, hogy a cserélhető lemezt nem igénylő job-ok munkaterületeiket se tegyék cserélhető lemezre.
Ezt a lemezegységek hardware cimeihez generált unitnevek megválasztásával értük el.
Az általánosan használt SYSDA és WORK unitnevek rendszerünkben csak az első hat lemezegységre vonatkoznak. A két cserélhető lemez egységét STORAGE unitnévvel lehet elérni. Megkülönböztettük ezenkivül a három rezidens lemez egységét a többitől: az OUTPUT unitnév segitségével.
A nagymennyiségü nyomtatott outputnak a job-tól független kiirására vezettük be az offline nyomtatást. A H és B osztályokban legfeljebb 5000, a többi job-osztályban legfeljebb 30 ezer sort nyomtathat közvetlenül egy job, ennél nagyobb outputot mágnesszalagra kell vinni. Ha egy job tullépte a számára előirt korlátot, a HASP üzenetet küld erről a gépkezelőnek.

A központositott állománykezelés azon alapul, hogy mágneslemezen és szalagon csak katalogizált, szabályos nevü és megőrzési idejü állományok tárolhatók.
Ennek megfelelően választottuk meg a rendszerkatalógus méretét és helyét.

Nem volt szó azokról a generálási paraméterekről, amelyek nem közvetlenül érintik az üzemeltetési rendet, de meghatározásuk komoly megfontolást igényel /pl. a fordítóprogramok paraméterei, a Supervisor rezidens részének meghatározása, stb/.

Az itt ismertetett operációs rendszert meghatározott üzemeltetési feltételeknek megfelelően kellett kialakitanunk.

A konkrét üzemeltetési rendet ugyanis a generált operációs rendszer határozza meg. Az üzemeltetési ren kialakitása tehát az operációs rendszer paramétereinek meghatározásával egyidőben kell, hogy történjék. A generálással egyidőben történhet a különböző funkciókat betöltő dolgozók oktatása /gépkezelők, állománykezelők, stb./; valamint a rendszer ismertetése a felhasználóknak, és az uj operációs rendszerrel egyidőben kell az uj üzemeltetési rendnek indulnia.

STRUKTURA GENERÁTOR ÉS COPY FUNKCIÓ A DOS PL/L-BEN /ELOPLI/

Szávai László

Mezőgazdasági és Élelmezésügyi Minisztérium
Országos Földügyi és Térképészeti Hivatal
Gépi Adatfeldolgozó Központ

A programrendszer bemutatása

Hazánkban az elmult években elterjedt a PL/I programozási nyelv használata.

A DOS PL/I-nek a jó tulajdonságai mellett néhány hiányossága is van. A programozónak fárasztó munkával kell a strukturákat leirni. A deklarációs rész egy-egy programban meghaladhatja a program egyharmad részét is, ilyenkor a struktura leirása sok időbe telik. A programokban előforduló azonos strukturáknak a leirása is feleslegesen sok időt vesz el a programozóktól.

Az assembly programozási nyelven megirt forrásprogramokat tárolni lehet a source könyvtárba és az egész programot /CSECT-et/ bemásolhatjuk egy másik assembly programba.

Ezt a COPY funkciót a COBOL magasszintü programozási nyelv is ismeri. Sajnos a DOS PL/I nem tudja a source könyvtárat kezelni és ezért ha egy másik programból programrészt fel tudunk használni, akkor a programozó kénytelen ujból leirni kódlapra. Ez szintén sok idővel és hibalehetőséggel jár.

A programozás megkönnyitésére iródott ez a segédprogram, amely a DOS rendszerben alkalmazható IBM/360-as és R 20-as számitógépen.

Az ELOPLI elősegiti és megkiméli a programozót a strukturák fárasztó leirásától és lehetővé teszi, hogy a már megirt és

és private source library-ba elhelyezett programból részeket átvegyen, valamint PL/I programot aktivizáljon.
Az ELOPLI program kisterjedelmü assembler szinten irt program. Futási ideje minimális. Pótolja a DOS PL/I-nek azt a hiányosságát, hogy nem tudja a source könyvtárat kezelni.
Mivel a DOS SYSTEM SOURCE LIBRARY-ba nem célszerü felhasználói programot elhelyezni, a program csak a private source library-t tudja kezelni.
A munkát mindig az ASSGN SYSSLB,X'130' paranccsal kell kezdeni, ami a magánkönyvtárat hozzárendeli a rendszerhez.

A rendszer megszervezésénél a következő fő szempontokat kell figyelembe venni:
1/ adattipusok meghatározása
2/ a struktura leirása minél egyszerübb legyen
3/ a PRIVATE SOURCE LIBRARY elérése és a megfelelő program vagy programrész kimásolása
4/ az egész folyamat beillesztése a DOS rendszerbe.

A program a PL/I-nek azt a tulajdonságát használja ki, hogy az első oszlopra nem lehet irni /kivétel a * PROCESS STMT kártya/. A programot az ELOPLI számára ugy kell összeállitani, hogy a PL/I programba, a megfelelő helyre a generáló kártyákat berakjuk és az igy összerakott kártyákat adjuk az ELOPLI programnak input adatként. Az ELOPLI program outputja egy lemezterület, ahová felkerül a PL/I program, a generál struktura és a bemásolandó programrész. Ez a lemezterület a PL/I forditóprogram input területe lesz.

A rendszer két fő részből áll:
1/ Struktura generáló programrész
2/ Copy funkciót ellátó programrész.

Az ELOPLI program listát készit a generáló kártyákról és a generált strukturáról. Besorszámozza a kártyákat az azonositás

megkönnyitésére. A struktura generátor 99. szintig tud elemeket generálni. Egy strukturában maximálisan 99 változó elem lehet.

G STRUKTURA GENERÁLÁS
C COPY FUNKCIÓ
/A G és C betü a kártya első oszlopára kerül./

A szintelemek a következő tipusok lehetnek:

tipus	jelölés
PIC	S Z 9 . V
CHAR	blank
FIXED DEC	Fdd,d

Pl.: S99 ⁄⁄⁄⁄⁄ F12,5 megfelel a PIC'S999' CHAR(5) és a FIXED(12,5) -nek.

megszakitójelek:

"vagy" jel	|	mező elválasztó
" és " jel	&	mező vége
	|&	megszakitás jelölése

szintszám:
A szintszámot ugy növelhetjük, hogy a mező elválasztójelet kétszer irjuk le egymás után. A csökkentés _ /aláhuzással/ történik.

Példa a generálásra:

```
9|ZZ||F15,5|F12,5|_⁄⁄&
G,
    2 C   PIC'SS9B9/9',
99.9|ZZZ.V9|⁄⁄⁄⁄⁄|F1|&
GSTRNEV,N
```

A generált struktura

```
DCL 1 STRNEV,
      2 NO1      PIC'99.9',
      2 NO2      PIC'ZZZ.V9',
      2 NO3      CHAR(5),
      2 NO4      FIXED(1),
      2 C        PIC'SS9B9/9',
      2 NO5      PIC'9',
      2 NO6      PIC'ZZ',
      2 NO7,
      3 NO8      FIXED(15,5),
      3 NO9      FIXED(12,5),
      2 NO10     CHAR(2);
```

Ha egy struktura generálását megszakitjuk és folytatni akarjuk, akkor egy olyan kártyát kell berakni, amelynek az első két pozicióján "G," áll és utána a kártya üres.

Copy funkció

```
C[OPY] P.NEV        [0040,[0090]]
                     tól   ig
```

/A szögletes zárójel az elhagyhatóságot jelenti./

Az első oszlopon a "C" betü kötelező, utána a modull neve áll. Ha a tól-ig hiányzik, akkor az egész modull, ha az ig hiányzik, akkor az adott sorszámtól az egész modull kerül bemásolásra a programba.
Az ELOPLI a régi sorszámot a 77-80 oszlopról előreviszi a 73-76 oszlopra és ujra besorszámozza a programot.

Az ELOPLI használata és JOB STREAM-ba illesztése

Az ELOPLI program a rendszer könyvtárkezelő service programjaival közösen egy aktiv programrendszert képez.
Az ELOPLI segitségével kártyáról az 1. példa alapján tudunk programot forditani, linkelni és futtatni.
A program elhelyezését a private source library-ba a 2..példa mutatja. A példából kitünik, hogy a könyvtárba elhelyezett modull már tartalmazza a generált strukturát és a bemásolt progrészt.
A modull neve két részből tevődik össze:

sublib.bookname

Az ELOPLI-nál a sublib kötelezően a "P", jelentése; a PL/I modull, a bookname a modull könyvtári nevét jelenti.

```
// JOB PROGRAM1
// UPSI Ø1
// EXEC ELOPLI
    .
    .
  PL/I
  STRUKTURA GENERÁLÓ   kártyák
  COPY
    .
    .
/*
// OPTION LINK
 ASSGN SYSIPT,X'130'     SYSIPT a 130-as lemezen
// EXEC PL/I
 CLOSE SYSIPT,X'00C'     SYSIPT a kártyaolvasón
// EXEC LNKEDT
// EXEC
    .
  adatok
/*
/&
```

1. példa: kártyáról forditás,futás

```
// JOB PROGRAM2
// UPSI Ø1
// EXEC ELOPLI
 CATALS P.bookname
 BKEND P.bookname
    .
    .
    PL/I
    STRUKTURA GENERÁLÓ    kártya
    COPY
    .
    .
 BKEND
/*
 ASSGN SYSIPT,X'130'
// EXEC MAINT
 CLOSE SYSIPT,X'00C'
/&
```

2. példa: PL/I program könyvtárba való elhelyezése a struktura generálás és copy funkció után.

Katalogizáláskor a modull a CATALS P.bookname kártyán megadott book névvel kerül a könyvtárba. A programot két BKEND kártya közé kell tenni, ahol az első BKEND kártyán a P.bookname-nek is szerepelnie kell.
Az ELOPLI programmal nemcsak PL/I programot, hanem egy egész JOB-ot is aktivizálhatunk a PRIVATE SOURCE LIBRARY-ból. A JOB-ot a MAINT segédprogrammal vihetjük be a könyvtárba. Amikor az ELOPLI-vel aktivizáljuk a JOB-ot a ** -ból /* a * -ból / lesz.

```
// JOB JOBFELV
// EXEC MAINT
 CATALS P.FEL
 BKEND P.FEL
// OPTION LINK
```

```
// EXEC PL/I
   .
   .
   PL/I program
   .
/*                                  /a /* kártya helyett/
// EXEC LNKEDT
// EXEC
/&                                  /a /& kártya helyett/
 CLOSE SYSIN,X'00C'
 BKÉND
/*
/&
```

3. példa: egy JOB felvitele a private source library-ba.

```
// JOB LEHOZ
// EXEC ELOPLI
 COPY P.FEL
/*
 ASSGN SYSIN,X'130'
```

4. példa: JOB aktivizálása a private soruce library-ból.

Az ELOPLI program megvizsgálja a rendelkezésre álló generáló kártyákat, ha azokban hiba van, és a JOB-ban // UPSI Ø1 kártya van, akkor hibaüzenetet ir ki a listára. Ha nincs UPSI-kártya, akkor nem ad listát a program.
A JOB concel-odik, ha a következő hibák valamelyike bekötkezik:
a/ struktura generáláskor van strukturanév, de hiányzik a kártyáról a vessző,
b/ nincs szintnév,
c/ pontosvesszőt talál a generáló-kártyákon,
d/ a copy funkciónál a keresett modull nincs a könyvtárban.

Értékelés

Az ELOPLI program jelentősége abból áll, hogy lerövidíti a programozási időt. Egy rendszer kidolgozásakor valamennyi programban a strukturák és file deklarációk azonosak lesznek és ezeket csak egy programozónak kell megirnia, a többiek átvehetik a nekik szükséges részeket.

A programok szervezőinek is csökken a munkája, mivel a strukturákra csak hivatkozniuk kell, de nem kell megadni a szerkezetüket. A file-ok esetében szintén csak a file-nevét és a feldolgozási módját /input, output, update/ kell megadniuk. Az átvett programoknál teljesen ki van küszöbölve a lyukasztásból eredő hiba. Az ELOPLI felhasználásával készülő programhoz kevesebb kártyát kell lyukasztani, mint a hagyományosan irt programokhoz. A szintaktikus hibák száma is kevesebb lesz, mivel az átvett programrészek már tesztelve vannak. A fáziskönyvtárba való katalogizálás után az egész JOB-ot elhelyezhetjük a forráskönyvtárba, igy a kártyák tárolása feleslegessé válik.

Az ELOPLI program jelentősége, hogy kevesebb munkával, kisebb gépidő-felhasználással, /a szervezéstől a program megirásáig/ rövidebb idő alatt lehet a programokat elkésziteni.

Az ELOPLI gépidő-szükséglete 10 kártya/ másodperc. Ez ugy értendő, hogy ha a tényleges PL/I program 200 kártyából áll /ebben benne vannak a generált és a copy funkció által bemásolt kártyák is/, az ELOPLI futási ideje ca 20 másodperc lesz.

Az ELOPLI program kifejlesztés három ember/hónapot vett igénybe, ca 5 /IBM 360/40/ gépóra felhasználásával. A ráforditott munkaidő és gépóra ca fél év alatt térül meg.

Az ELOPLI program változtatás nélkül felhasználható FORTRAN programok esetében is a COPY funkció ellátására.

DOS JOB STREAM GENERÁLÓ PROGRAMCSOMAG

Zsadányi Pál
Vegyipari Számitástechnikai Fejlesztési Társulás

ELŐZMÉNYEK

Több, mint két éves IBM DOS tapasztalatokat elemezve azt deritettük ki, hogy a DOS kellemetlen hátrányai közé tartozik az OS-hez képest a sok vezérlőkártyával történő bajlódás a gépteremben, valamint a kártyán tárolt programok, JOB-ok megbizhatatlan állapota. Gépidőkihasználtsági szempontok pedig eleve azt sugallják, hogy a viszonylag leülepedett anyagokat gyorsan elérhető módon mágneslemezen érdemes tárolni. A DOS alaprendszer azonban rossz hatásfokkal tudja csak kihasználni az ebből származó előnyöket, mert egy sereg ujabb bonyodalmat okozó JOB CONTROL utasitást igényel, és az operátorra is nagyobb feladat hárul akár szalagos, akár lemezes rendszer file-oknál. Ezenkivül nem enged meg dinamikát, változásokat JOB CONTROL szinten lehet csak elérni operátori trükkökkel.

Kerestünk tehát olyan IBM megoldást, ami ezen segitene. Ezt hamarosan meg is találtuk a POWER akkor megjelent legujabb változatában, amely a forrásnyelvi könyvtárban elhelyezett anyagokat képes beilleszteni a JOB áramba, mégpedig módositásokkal. A POWER egyéb igen jó tulajdonságai mellett ezt találtuk a leghasznosabbnak. A POWER rendszer azonban igen képzett operátorokat igényel, és a bérleti viszonyokkal együtt ez nem tette egyenlőre lehetővé, hogy ezt használatba vegyük.

Időközben értesültünk egy olyan megoldásról, ami szintén az általunk kivánt cél irányába hat, de jóval parlagibb módon PL/I szinten beprogramozható megoldás. Ez a megoldás a FÜTI-től származik, és a cimben szereplő elnevezéssel illették. A fő elve: a kártyákat azonositómezeik segitségével indexszekvenciális file-ban tárolni és ehhez volt egy betöltő, egy karbantartó és egy interaktiv lekérdező program /amennyiben a konzolról lehetett megadni a kért feldolgozási menetet/. A megoldás szintén statikus volt. Mi átvettük ugyan a programokat is, az elven tul, de azokat gyakorlatilag nem használtuk, hanem azonnal irányt vettünk egy komplex programcsomag létrehozására, amely ezen az elven egy dinamikus megoldást hoz létre, és megoldja a különböző szerviz funkciókat is.

CÉLOK, ELVEK, FUNKCIÓK

A programrendszer megtervezésekor elhatározott célok:

JOB-ok és programok tárolásának megoldása mágneslemezen ugy, hogy azok állapotát ne bolygassák a napi igények, tehát mindig jól dokumentált állapotban legyenek.

A tárolt anyagokból össze lehessen állitani egy a napi igényekhez igazodó JOB áramot /POWER SLI,DATA funkció/.

A tárolt anyagokon felmerülő változások belövésére ugy legyen alkalmas a rendszer, hogy a lekérdező menetben erre a célra használandó kártyák változtatás nélkül legyenek alkalmasak a módositások végleges végrehajtásához. / Ez a POWER-hez képest is előrelépés !/

Az előbbiek következményeként is jelentkező módon: megszabadulni a kártyatömegekkel történő bajlódástól, biztonságosabbá tenni a futtatáselőkészitést, lehetővé tenni a program és JOB módositások belövését anélkül, hogy közben a változtatás tárgyát képező anyagok megbizhatatlan állapotba kerülnének.

ELVEK:

Alapvetően megkülönböztetünk programokat és JOB-okat, ezért ezekhez két külön MASTER file-t rendelünk hozzá.

Alacsonyabb sebességü gépekre számitva előkészitésnél, nem alkalmazzuk a forrásnyelvi könyvtárban meghonositott komprimált tárolást.

A kivánt anyagok elérésére indexszekvenciális technikát alkalmazunk, ami magasszintü nyelven történő programozást tesz lehetővé. A programokat magasszintü nyelven irjuk meg, mert a stratégiai cél világossága mellett még taktikai problémák várhatók.

A funkciókat fokozatosan bonyolitjuk a programokban, hogy minél hamarabb tapasztalatokra tegyünk szert a rendszer komplex müködésével kapcsolatban.

A programokat az adatfeldolgozás különböző tipusfeladatainak megfelelően választjuk meg. /File-ok betöltése, listázása, karbantartása, mentése, lekérdezése, és speciális feladatok/.

A tárolt adatok azonositóinak képzésénél házi szabványokhoz alkalmazkodunk.

JOB-ok esetén bizonyos JOB sorozatokat lehessen egyetlen paraméterrel hivni, és inkább az egy JOB, egy JOB-step elvet alkalmazni a futtatást zavaró körülményekből eredő veszteségek csökkentésére.

A funkciók az elveknek megfelelően programonként különbözőek:

JØ1
Index-szekvenciális file betöltő program, amely kártyáról, vagy mágnesszalagról vehet inputot a betöltéshez.

JØ2
JOB-STREAM master és program file listázó program paraméterezhető listázásra.

JØ3
Master file update program, pillanatnyilag kártyáról.

JØ4
Master file kimentő program paraméterezhető módon mágnesszalagra vagy kártyára történő kimentéshez esetleges átnevezési, ujraszámozási lehetőséggel.

JØ5
Lekérdező program JOB STREAM generálásra lekérdező paraméterkártyák, módositó kártyák és adatkártyák segitségével. A keletkező JOB sorozatot egy mágneslemez file-ba helyezi el, amely utána lemezes rendszerinput-file-ként /SYSIN/ használható. Bizonyos multiprogramozási problémákat is megold bizonyos erőforráskiosztási konvenciók alapján.

JØ6
Segédprogram félbehagyott munka folytatásának megkönnyitésére. A JOB STREAM file-ból kihagyja a megadott sorszámu JOB előtti JOB-okat.Ezután a munka folytatható.

JØ7

Program azonositóképzésre, sorszámozásra, mágnesszalagra történő felvitelre, betöltés előkészitésre.

JØ8

Speciális program a programfile-ban található programok verziószámainak kiirására.

JOB STREAM GENERÁLÁSA

A JØ5 program olvassa a lekérdező paraméterkártyákat, amelyek szerkezete:

Oszlop	Tartalom
1-4	*▼$$ paraméter azonositó /POWER is/
5-8	lekérdező funkció
9	mindig blank legyen
10-71	paraméter mező
72	folytatás jel /jelenleg potenciálisan/
73-80	érdektelen

A lehetséges lekérdező funkciók:

	Funkció	Hatás
0.	üres	folytatás kártya
* 1.	/*	END OF FILE kártya
* 2.	/&	END OF JOB kártya
* 3.	SLI	JOB/ eljárás bemásolási utasitás
4.	PLI	program bemásolási utasitás
5.	EOU	bemásolási módositások vége
* 6.	DATA	adatazonositási és bemásolási utasitás
* 7.	END	adatok vége
8.	CAT	szabványos katalogizálás utasitás
+ 9.	TEST	tesztfutás generálási utasitás
+10.	BLNK	üres kártya
+11.	ALRM	ALARM utasitás
+12.	EOJ	egybefüggő hivások vége

* POWER-ben is használt
\+ Az előadás irásakor még csak tervezett.

Az 1. és 2. funkciók hatására a generált file-ban a megfelelő /* és /& kártyák generálása történik.
A 3. funkció paramétere egy JOB STREAM MASTER-file-ban található JOB vagy JOB sorozat /u.n.eljárás/ azonositója. Az eljárásazonositó 3 karakteres, a JOB azonositó 5 karakteres. Ebből a házi szabvány szerint az első a feladatcsalád azonositója, kettő sz eljárás azonositója, kettő az eljáráson belüli JOB azonositója, de ezeket a program nem figyeli igy. Egy SLI kártyát módositó kártyák követnek, amelyek vagy közbeszuródnak sorszámuk szerint, vagy felülbirálják a MASTER file megfelelő azonositóju teljes kártyáját /ha a módositó csupa blank a 72.pozicióig, akkor nem engedélyezi a bemásolást/. A módositó kártyák végét az 5. funkcióval jelzett EOU /END OF UPDATE/ jelzi.

A 4.funkció paramétere egy PROGRAM file-ban található program azonositója, amely utolsó négy /sorszám/ pozicióján Ø-kal, első négy pozicióján blank-el egészül ki jobbról. Az igy keletkező azonositóval rendelkező rekordnak kell lennie a program file-ban. További paraméter a tervek szerint a program tipusa. Jelenleg csak PL/I programok lehivására van kiépitve ez a funkció. A módositás elve ugyanaz mint az SLI funkciónál, és szintén EOU kártya jelzi a módositások végét. A program végét a PROGRAM file-ban egy /* $$/*-ot tartalmazó rekord jelzi.

A 6. és 7. funkciót eredetileg a POWER-ben használatos funkcióra terveztük, de a jelenleg megvalósitott állapotban egyszerü bemásolást okoz az adatok végét jelző 7. funkcióig.

A 8. CAT funkció paramétere egy program azonositója, amely a jelenlegi állapot szerint PL/I program, és hatására egy modul könyvtáron át megszervezett forditás és LINK JOB generálódik, amely a program változtatások kizárásával történő katalogizálását okozza szabvány BG és FG területre, szabványos fázisnévvel. Egy belőtt módositás UPDATE-je után fontos lehetőség. Itt megjegyezzük, hogy a PLI menetben hivott programba még bemásolódik egy olyan utasitás, ami a program elindulásakor PROBAFUTAS üzenettel jelzi, hogy nem szabványos menetben forditott program futásáról van szó.

A 9. TEST funkciót egy program forditás-futás menetben történő tesztfutásának automatikus megszervezésére terveztük.

A 10. BLNK-et egy üres kártya beiktatására terveztük.

A 11. és 12. funkciót arra terveztük, hogy megjelölhessük a lekérdező áramban az összetartozó szakaszokat /ameddig hiba esetén át kell ugrani a lekérdező adatokat/ illetve leállitani a generálást, ha a generálás már értelmetlen tovább.

TAPASZTALATOK

A rendszer 1974 októbere óta üzemel a lekérdező program folyamatos fejlesztése mellett és igen alaposan megkönnyitette a futtatás-előkészitést.

TOVÁBBFEJLESZTÉS

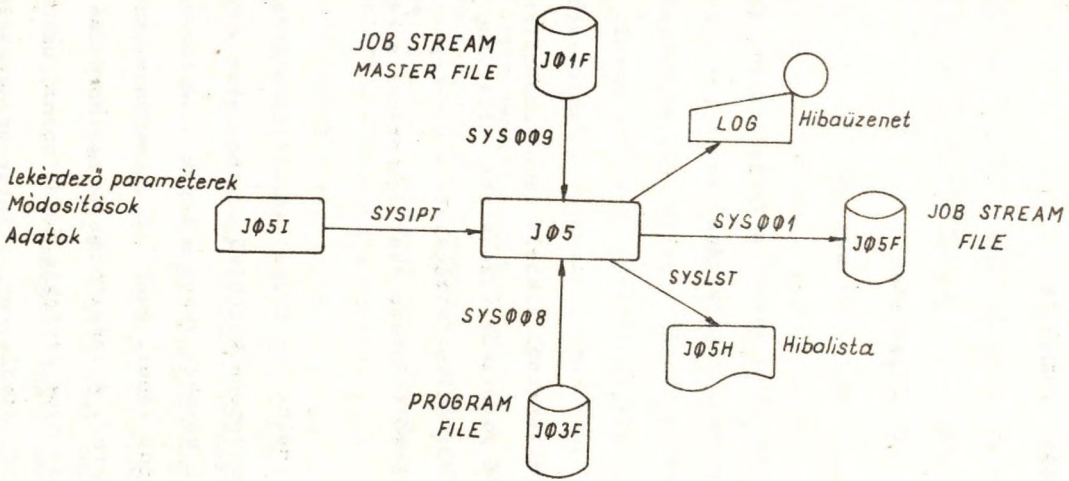
A programrendszert tovább kivánjuk fejleszteni. Egyenlőre magasszintó programozási nyelven, mert még vannak kialakulatlan részletek. Megjegyzendő, hogy ez az állapot meg-

könnyitheti a rendszer adaptációját más felhasználóknál, ha igényeik eltérnek a programrendszer koncepciójától.

Javithatna a rendszer müködésén, ha nem index-szekvenciális file-kezelést használnánk, hanem a BOMP /BASTEI/-ban használatos kontroll-szekvenciális /láncolt/ file kezelést. Az index-szekvenciális technika ugyanis csak átszervezési menetben tudja elviselni az olyan beszurásokat, amelyek egy ponton nagy tömegben jelentkeznek. Uj programok felvétele azonban éppen ilyen problémákat vet fel.
Az algoritmusok kialakulása után célszerü lesz a rendszert ujrairni nagyobb érdeklődés esetén ESZR szabványok szerint.

Alacsonyszintü nyelven elképzelhető POWER DATA és QUEUE file-ok generálása is, amelyek aztán átadhatók egy POWER rendszernek. /WARM START/.

Volna értelme egy ON-LINE lekérdező programnak is kevesebb lekérdező funkcióval, módositások kizárásával.
Sok apró gépkezelői tevékenységet könnyitő JOB-ot lehetne igy kezelni.



JOB STREAM generálás

KISSZÁMITÓGÉP IZOTÓPDIAGNOSZTIKAI SOFTWARE RENDSZERE

Csernay László, Csirik János, Makay Árpád, Máté Eörs
Szegedi Orvostudományi Egyetem és József Attila Tudományegyetem

A RENDSZER FELADATA

Az orvosi gyakorlatban egyre inkább elterjed az un. izotópdiagnosztikai eljárások alkalmazása. Ennek során a beteg szervezetébe sugárzó izotópot juttatnak, amely bizonyos idő elteltével a vizgálni kivánt szervben dusul. Az un. /állódetektoros/ gammakamera bizonyos időintervallum /a képidő/ alatt a beütésekből sikbeli képet szolgáltat. Egyetlen ilyen képből /statikus vizsgálat/, illetőleg képek egy sorozatából /dinamikus vizsgálat/ a szerv alakjára, sőt müködésére vonatkozó információkat nyerhetünk a képkiértékelés során.

A számitógépeket használhatjuk csupán képkiértékelésre, ebben az esetben a gammakamera közbülső adathordozón /pl. lyukszalag/ rögziti a képeket /vagy a képek előállitásához elegendő un. list- módu képet, amely az egyes beütések x,y koordinátáinak sorozata/. A számitógép az adathordozón kapott képeket képes bizonyos mértékig feldolgozni, az orvos számára vizuálisan /pl. display-en, printer-en/ megjeleniteni. A korszerü módszer azonban a gammakamera és a szá-

mitógép közti on-line kapcsolat. A kamera direkt vonalon a számitógép memóriájában állitja össze a képeket, amelyek folyamatosan háttértárolón raktározódnak el a későbbi feldolgozás céljaira.

Rendszerünk a hazai gyártmányu TPAi kisszámitógépre épül, amelyhez a gammakamera on-line kapcsolódik. A kamera 64x64-es képet szolgáltat, amelyeket szines és fekete-fehér képernyőn is meg tud jeleniteni. A képek tárolása diszk-en történik, a képek archiválása /mágneszalagon vagy más, nagykapacitásu és olcsó tárolón/ még megoldatlan kérdés.

A software-rendszer elsősorban azt a célt szolgálja, hogy izotópdiagnosztikai laboratóriumok rutin munkája zökkenőmentes legyen. Folyamatos betegvizsgálatot tesz lehetővé a laboratórium munkarendje szerint amely, asszisztensi munka marad. A rendszer előkészíti a felvételeket kiértékelésre, amelyet a kiértékelést végző orvos bármikor elkezdhet. A kiértékelés során az orvos rendelkezésére állanak azok a feldolgozó rutinok, amelyek a képek transzformálását, többféle megjelenitését stb. végzik. Ezekből az elemi rutinokból a laboratórium saját profiljának megfelelő komplett értékelő programokat is összeállithat, amelyeket a számitógép esetleg már a képfeldolgozás előkészitő szakaszában elvégez.

A rendszer és a kezelő /orvos vagy asszisztens/ kapcsolatában arra törekedtünk, hogy csak a vizsgálattal, illetőleg a kiértékeléssel kapcsolatos kérdések szerepeljenek a konverzációkban. A laboratóriumi munkarend szerint a software rendszer önállóan igyekszik szervezni saját munkáját.

A számitógép kihasználhatóságának fokát a software-rendszer több lehetőség biztositásával is emeli. A képek kiértékelésre való előkészitését a vizsgálat után közvetlenül, a vizsgálatok szüneteiben végzi. A számitógép fennmaradó kapacitása off-line izotópdiagnosztikai vizsgálatokra, továbbá a klinikán, kórházban stb. felmerülő egyéb munkákra felhasználható. Ilyen és ehhez hasonló feladatok számára a software nyitott: lehetőség van ujabb feldolgozó programok rendszerbe illesztésére.

A KAMERA KISZOLGÁLÁSA

A vizsgálat megkezdése előtt, a rendszer és a kezelő közötti konverzáció során kell megadni a felvételhez szükséges paramétereket. A konverzációt a számitógép irányitja olymódon, hogy /a korábbi válaszokat figyelembevéve/ kérdéseket tesz fel. A beteg adatain kivül a vizsgálat célját /amely részben meghatározza az előfeldolgozás és képkiértékelés menetét/, a kivánt képszámot /esetleg több, maximálisan három képidővel/ kell megadnunk. A válaszokat a számitógép ellenőrzi a kezelési tévedések, illetőleg az igények teljesithetősége szempontjából.

A felvétel időszakában a kamera munkáját a software irányitja. A memóriában egyszerre két kép számára van hely. Az egyikbe a kamera folyamatosan tölti a beütések x,y koordinátáit /list módu begyüjtés/ illetve az x,y koordinátáknak megfelelő rekeszben számlálja a beütéseket /frame módu begyüjtés/ mig a másikból a kész kép diszk-re irása folyik. A memóriablokkok előkészitését, illetőleg az előválasztott képidőnek megfelelő blokk-átkapcsolásokat a felvételt vezérlő rutin végzi.

A vizsgálat megkezdése után lehetőség van az ujrakezdésre. Ez, az un. előperiódus teszi lehetővé a beteg jobb elhelyezését, lassu dusulás esetén az érdektelen képek törlését stb. Az előválasztott képszám begyüjtése után is folytatódik a vizsgálat /amig a számitógép diszk-kapacitása ezt lehetővé teszi/ azért, hogy a vizsgálatot ne kelljen megismételni az előkonverzáció során elkövetett esetleges hibák miatt.

A felvétel alatt az elkészült /vagy az éppen készülő/ képek display-en követhetők.

A vizsgálat befejeztével a képeket a számitógép katalogizálja és elkezdi az előfeldogozást végző programok egymás utáni végrehajtását. Az előfeldolgozással párhuzamosan kész a következő vizsgálattal kapcsolatos konverzációt folytatni.

A standard előfeldolgozó programok közül megemlitjük a list-módu képek frame-módu képekké történő transzformációját, az inhomogenitás-korrekciót /a kamera felvevőrendszerének x,y koordináta szerint érzékenysége miatt/ és a holtidő korrekciót /amely szintén a kamera érzékenységéből származó tényező/. További előfeldolgozást a vizsgálatot megelőző konverzáció során a vizsgálat céljának megjelölésével implicit módon, vagy feldolgozó rutinok /programok/ megnevezésével explicit módon kérhetünk.

KÉPFELDOLGOZÁS

Amint az eddigiekből is kiderült, az előfeldolgozott képek kiértékelését a gépkezelő bármikor kezdeményezheti. Ezzel zárul le az un. vizsgálati periódus, amikor a betegekről egymás után készülnek a felvételek. /Természetesen a gépkezelő egy feldolgozás után ujra felvételt kezdeményezhet./ Képfeldolgozást a számitógép is kérhet, ha a felgyült képek lehetetlenné teszik a további vizsgálatokat.

A feldolgozó programok modul-rendszerüek. Egy-egy modul egy-egy elemi feladatot végez: kép-kijelzés, egy kép transzformációja, több kép szummációja, részterület /ROI/ kiválasztása a képből, időgörbék előállitása és elemzése stb. A feldolgozó program néhány ilyen modulból és ezen modulok végrehajtási sorrendjét /ismétlés, ciklizálás, feltételes végrehajtás lehetséges/ előiró, az un. felügyelő-programnak szóló kontrol-információból áll. A modulokat

a felügyelő program inditja vagy közvetlenül az előző lefutása után, vagy a kiértékelést végző orvos felszólitására /igy lehetőség van pl. a képek kellő ideig tartó megfigyelésére/. /Az igy kialakitott rendszer biztositja tehát azt, hogy meglévő modulokból uj feldolgozó programot egyetlen kontrol-szó összeállitásával kaphatunk./

Ugyancsak a kontrol-szó tartalmazhat arra vonatkozólag is utasitást, hogy a program első moduljának inditása előtt a kezelővel konverzációt kell folytatni. A konverzáció menete hasonló a vizsgálatot megelőzővel, az általa nyert információk a feldolgozáshoz szükséges paraméterek, azonkivül a feldolgozás menetét is befolyásolhatják. /Megjegyezzük, hogy a vizsgálatot megelőző konverzáció és az előfeldolgozás során kapott információk a feldolgozás alatt rendelkezésre állnak./ A feldolgozás közben igy hosszabb konverzációra nincs szükség, az előre rögzitett menetrend szerint folyhat a kiértékelés.

A felügyelő-program a modulok számára bizonyos alapvető tevékenységeket elvégez:

- ha a modul egy képen dolgozik, ezt a modul inditása előtt a memóriába hozza;
- kép-sorozatok elemeinek egymás után memóriába - töltését kérheti a modul;
- diszk-en közbülső /munka-/képek tárolását és ujbóli memóriábatöltését végzi;

- a kép- és görbe-megjelenités fizikai tevékenységeit elvégzi;
- közvetit a gépkezelő és a modul között;
- pseudo-utasitásokat /pl. lebegőpontos aritmetika/ hajt végre.

A RENDSZER BŐVITÉSE

A programkönyvtár felépitése lehetővé teszi ujabb modulok beépitését. Ezekből és a régi modulokból standard kontrol szavak összeszerkesztésével ujabb feldolgozó programokat nyerhetünk.

A képek archiválása fogja biztositani azt a tanuló-anyagot, amelyen felépithető és kipróbálható egy automatikus kép-felismerő rendszer.

TRANZ-TRAN 3 - UJ ÁRAMKÖRANALIZIS PROGRAMRENDSZER A TPA-i SZÁMITÓGÉPRE

dr. Székely Vladimir
dr. Tarnay Kálmán
Rencz Márta
Baji Pál

Budapesti Müszaki Egyetem
Elektronikus Eszközök Tanszék

A Központi Fizikai Kutatóintézet által gyártott TPA-i számitógépet kiterjedten használják hazánkban, sok müködik belőle az elektronikai ipar és oktatás területén. Felhasználásuk jelentős területe a tervezési munka számitógépes segitése. Az egyik jellegzetes elektronikai tervezési feladat az áramkörtervezés -- ehhez jelentős segitséget adhat a számitógépes áramkörszimuláció. Ezért merült fel az igénye egy, a TPA-i gépen müködő, univerzális (tehát a felmerülő feladatok széles körében használható) áramköranalizis programnak.

Tanszékünk, a Budapesti Müszaki Egyetem Elektronikus Eszközök Tanszéke a KFKI megbizásából az 1974-75 időszakban kidolgozta a TPA-i számitógép általános rendeltetésü áramköranalizis programját, a TRANZ-TRAN 3-at. E munkáról kivánunk előadásunkban beszámolni.

Az áramkör-szimuláció problémáját közepes- vagy nagygépet követelő feladatként tartják nyilván; az ismert analizisprogramok általában ilyenek. E programok kifejezetten kisgépes megfelelőjét kivánva létrehozni, kettős célt állitottunk magunk elé:

a/ a program a TPA-i gépnek a lehetőségek határain belül legkisebb konfigurációját igényelje,

b/ ugyanakkor szolgáltatásaiban ne maradjon el számottevően a hasonló rendeltetésü, közepes gépeken működő analizisprogramokhoz képest.

Kétségtelen, hogy e két követelmény szöges ellentétben van egymással -- aminek feloldása egy vonatkozásban nem is sikerülhetett. Ez a vizsgálható áramkör mérete, ami elsősorban a rendelkezésre álló memóriaterület függvénye. Programunk kb. feleakkora hálózatot képes analizálni, mint az átlagos nagygépes analizisprogramok. Ez még mindig elegendő közepes méretü diszkrét elemes hálózatok, hibrid integrált áramkörök, néhány müveleti erősitőt tartalmazó áramkörök stb. vizsgálatára. Hangsulyozzuk ugyanakkor, hogy minden, a használat kényelmét érintő kérdésben, analizis-lehetőségben, dokumentálási formában, opcióban programunk alig marad el a "nagy" programoktól.

Munkánk bázisa a tanszékünkön az előző évek során kifejlesztett TRANZ-TRAN 2 nagygépes analizisprogram volt [1]. Ez a program 1969-72-ben készült, és jelenleg az ország több számitóközpontjában hozzáférhető (ICT-1905, RAZDAN-3 és CDC-3300 gépeken). Modellei és algoritmusai mögött öt év tapasztalatai állnak. A TPA-i gépre kifejlesztett TRANZ-TRAN 3 a legtöbb vonatkozásban e nagy program kivonatolt változatának tekinthető. Igy a kisgépes változat mintegy örökölte az algoritmusok és modellek kiérleltségét, megbizhatóságát.

TULAJDONSÁGOK, SZOLGÁLTATÁSOK

A program nemlineáris, tanzisztoros hálózatok analizisére, szimulációjára alkalmas. Segitségével a következő vizsgálatokat végezhetjük [2] :

- egyenáramu (DC) analizis,
- transzfer karakterisztika számitás,
- tranziens analizis,
- kisjelü váltakozóáramu (AC) analizis,

- egyenáramu érzékenység vizsgálat.

A program beépitett modellekkel rendelkezik félvezető diódákra, NPN és PNP bipoláris tranzisztorokra, továbbá n és p csatornás térvezérelt és MOS tranzisztorokra. Figyelembe veszi a félvezetőeszközök paramétereinek hőmérséklet-függését. Lehetőséget ad arra, hogy további modelleket és részáramköröket áramköri modulként definiáljunk. Ezek később név szerint hivhatók a vizsgálandó áramkörben. Az áramköri modulok egymásbaskatulyázási mélysége tetszőleges.

A program használatához szükséges minimális konfiguráció:

- TPA-i központi egység 16 K memóriával
- teletype
- lyukszalagolvasó és perforátor.

E kiépitésen max. 3o csomópontot, 6o ágat tartalmazó áramkörök analizálhatók. A program azonos teljesitőképesség mellett kényelmesebb, gyorsabb futtatást és dokumentálást biztosit a sornyomtatóval és disc-kel is felszerelt gépeken.

Az univerzális áramköranalizis programoknál kialakult gyakorlatot [3] követve, a program egy könnyen elsajátitható, felhasználó-orientált nyelven várja az analizisfeladatot. A beolvasó szegmens sokoldalu szintaktikus ellenőrzést végez a feladaton, hiba esetén részletes diagnosztikát biztosit /kb. 5o-féle hibajelzés/.

ALGORITMUSOK

A program első lépésként felépiti az analizálandó áramkör ekvivalens hálózatát. Ezzel a feladatot egyszerübb, néhány fajta elemet tartalmazó hálózat vizsgálatára vezeti vissza. Utóbbi már közvetlenül feldolgozható a megoldó algoritmus által.

A program minden algoritmusa csomóponti potenciál módszerrel dolgozik. Az egyes analizisek egy-egy jellegzetes matematikai feladat megoldását igénylik:

Az egyenáramu analizis sokismeretlenes nemlineáris egyenletrendszer megoldása. A program Newton-Raphson algo-

ritmussal számol, de az iterációs lépés nagyságát célszerüen korrigálja [4] . A csomóponti feszültségek vektorát V_j-vel jelölve, az

$$E_j = A_{ij}\, I_i(V_j) \qquad (1)$$

kifejezés adja a csomóponti hibaáramokat /tehát a csomóponti Kirchhoff törvénytől való eltérést/, ahol $I_i(V_j)$ az i-edik ág nemlineáris egyenlete és A_{ij} a hálózat csomópont-ág incidenciamátrixa. A feltételezett V_j csomóponti feszültségekre a nemlineáris ágak differenciális vezetése és igy a teljes hálózat Y_{ij} differenciális admittanciamátrixa megállapitható. Ezzel a ΔV_j Newton-Raphson lépés:

$$-E_i = Y_{ij}\, \Delta V_j \quad . \qquad (2)$$

E korrekciós vektort egy R faktorral szorozva képezzük a következő iteráció csomóponti feszültségeit

$$V_i' = V_i + R \cdot \Delta V_i \quad , \qquad (3)$$

ahol $R \leqslant 1$ és értékét a V_i és $V_i + \Delta V_i$-hez tartozó hibaáramok átlagából határozzuk meg.

A transzfer karakterisztika számitás és az érzékenység vizsgálat voltaképpen DC analizisek sorozata. Említést csak az érdemel, hogy előbbinél a program minden analizis kezdő V_j vektorát kvadratikus extrapolációval képzi az előző pontok megoldásából; ez jelentősen gyorsitja a konvergenciát.

A nemlineáris tranziens analizis nemlineáris, időfüggő differenciálegyenlet-rendszer megoldását igényli. Az egyenletek száma a csomópontokéval egyező, tehát akár 3o is lehet. A feladat itt a kisgéppel való megoldhatóság határán van. Az algoritmus időtartományban integrálja a differenciálegyenletrendszert, visszafelé lépő Euler formulát alkal-

mazva. Az integrálás során hatékony, 1:1024 átfogású lépésközszabályozás biztositja a mindenkor optimális lépésközt.

A kisjelü váltakozóáramu analizis a hálózat linearizált helyettesitőképe alapján számolja a frekvenciafüggő tulajdonságokat. Matematikailag az alábbi lineáris, komplex együtthatós egyenletrendszer megoldása a feladat:

$$I_i = \left(G_{ik} + j\omega C_{ik} \right) V_k \quad , \tag{4}$$

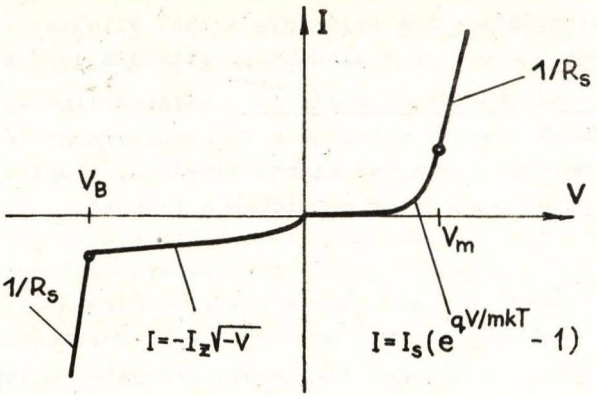
ahol G_{ik}, ill. ωC_{ik} a hálózat admittanciamátrixának valós, ill. képzetes része, I_i a gerjesztések árama és V_k az ismeretlen csomóponti feszültségek vektora. A fő probléma itt a memóriaigény. A komplex együtthatók miatt az admittanciamátrix helyfoglalása megkétszereződne, ami már nem megengedhető. Ezért kikötöttük, hogy a gerjesztési áram oszlopvektor elemei mind valósak. Ezzel

$$\begin{aligned} \mathrm{Re}\, I_i &= \left(G_{ij} + \omega^2 C_{ik} G^{-1}_{km} C_{mj} \right) \mathrm{Re}\, V_j \quad , \\ \mathrm{Im}\, V_k &= -\omega\, G^{-1}_{ki}\, C_{ij}\, \mathrm{Re}\, V_j \quad . \end{aligned} \tag{5}$$

A fenti egyenletekben szereplő mátrixok mindegyike felépithető egyetlen mátrix területén (az első sorban előforduló hármas mátrix-szorzatnál ez nem magától értetődő, de a hálózat kapacitásaiból álló C_{ik} mátrix speciális strukturája lehetőséget ad rá). Ezzel az AC analizis adatmező igényét a többi analizisével egyenlő mértékre tudtuk leszoritani.

MODELLEK

Az eszközmodellek alapeleme a pn átmenet (félvezetődióda) modell. A dióda egyenáramu karakterisztikáját négy tartományban irjuk le (1.ábra) :



1. ábra

- kis nyitófeszültségekre ideális karakterisztika, a kT/q érték m-szeres korrekciójával:

$$I = I_s \left(e^{qV/mkT} - 1 \right), \qquad (6a)$$

- a $V_m = \frac{mkT}{q} \ln\left(\frac{mkT}{q R_s I_s}\right)$ határ fölötti nyitófeszültségekre az R_s soros ellenállásnak megfelelő lineáris karakterisztika

$$I = \frac{V + mkT/q - V_m}{R_s} - I_s , \qquad (6b)$$

(V_m fenti választása a két görbe deriválttal folytonos illesztését biztositja),

- kis zárófeszültségeknél a térfogati generációt is figyelembe vesszük:

$$I = - I_z \sqrt{- V} , \qquad (6c)$$

- a V_B letörési feszültségnél nagyobb zárófeszültségeknél $1/R_s$ meredekségü letörési karakterisztikával számolunk:

$$I = - I_z \sqrt{- V_B} + \frac{V - V_B}{R_s} . \qquad (6d)$$

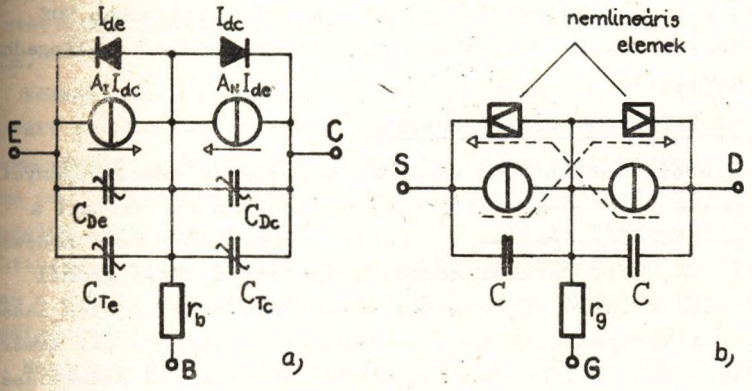
A pn átmenet váltakozóáramu és tranziens tulajdonságait a C_T tértöltés és C_D diffuziós kapacitás irja le:

$$C_T = C_{To} (V_D - V)^{-1/2} \quad ; \quad C_D = \tau \frac{dI}{dV} \quad , \qquad (7)$$

ahol V_D az átmenet diffuziós potenciálja.

A program az I_s, I_z és mkT/q paramétereket a deklarált környezeti hőmérsékletnek megfelelően módositja.

A bipoláris tranzisztorokat a 2a. ábra szerinti Ebers-Moll modell irja le.



2. ábra

Az EB és CB átmenet leirására a fent közölt pn átmenet modell szolgál. A normál irányu áramerősités munkapont- és hőmérsékletfüggő. A_N-nek az EB átmenet I_{de} áramától való függését az

$$A_N = \frac{A_{max}}{1 + a\left(V_{EB}\left(I_{de}\right) - U_m\right)^2} \qquad (8)$$

összefüggéssel számoljuk, ami ekvivalens a bevált

$$A_N = \frac{A_{max}}{1 + b \cdot \ln^2\left(I_{de}/I_m\right)}$$

formulával — de az ln függvény hiánya miatt kiértékelése sokkal gyorsabb.

A tervezérelt tranzisztorokat a program a 2b. ábrán látható, Ebers-Moll jellegü modellel irja le. A feltüntetett nemlineáris elem karakterisztikája:

$$I = \begin{cases} I_o (V - V_T)^2 & \text{ha} \quad V \geq V_T \\ 0 & \text{ha} \quad V < V_T \end{cases} \qquad (9)$$

ahol V_T a küszöbfeszültség. Ez a modell elsősorban MOS tranzisztorok leirására szolgál, de JFET-ekre is elfogadható közelités.

A KISGÉPRE ALKALMAZÁS PROBLÉMÁI

Negyedére csökkenteni egy analizisprogram memóriaigényét, ugyanakkor többé-kevésbé változatlan szinten tartani a teljesitőképességet -- e kettős célkitüzés nagy nehézségeket okozott. További gondot jelentett a TPA-i gépnél software szinten végrehajtott lebegőpontos müveletek lassusága. Algoritmusaink kiválasztásánál igy mind a kis memóriaigényt, mind a lebegőpontos müveletek számának redukálását szem előtt kellett tartanunk. A lebegőpontos számok viszonylag kicsiny szóhossza (27 bites mantissza) is okozott problémákat.

A memóriagondokat a program két részre osztásával némileg enyhiteni tudtuk. A két rész:

Part 1. COMPILER,

Part 2. CIRCUIT ANALYSIS.

E felbontásra az adott lehetőséget, hogy a program által elvégzendő feladatok is élesen két csoportra válnak.

Part 1. végzi a feladat utasitásainak szintaktikus vizsgálatát. Ha a feladat szintaktikusan helyes, beiktatja a hivott áramköri modulokat, felépiti a tranzisztor helyettesitőképeket, majd mind az áramkör leirását, mind az analizisutasitásokat a program belső leirására forditja.

Part 2. végzi a COMPILER szegmens által előfeldolgozott feladatok analizisét és az eredmények dokumentálását. A kisgép-problémák ebben a második szegmensben jelentkeztek a legkritikusabban.

Nagy körültekintéssel kellett megválasztanunk azokat a numerikus algoritmusokat, melyek mind memóriafelhasználás, mind futásidő szempontjából a leginkább elfogadhatók. A választott algoritmusokban minden analizisnél csupán egyetlen kétdimenziós mátrix épül fel ténylegesen a memóriában, éspedig az Y_{ij} differenciális admittanciamátrix, vagy az (5) egyenlet összetettebb mátrixai. A (2) és (5a) egyenletrendszerek megoldása, továbbá az (5a) -ban szükséges mátrixinvertálás konvencionális Gauss-Jordan módszerrel történik.

Felmerülhet egy másik alternativa, ami a memóriaigényt tovább csökkentené. Az Y_{ij} mátrix meglehetősen üres, igy szóba jöhetne a komprimált tárolás. Ez esetben viszont sajnos nem alkalmazhatnánk az egyszerü Gauss-Jordan egyenletmegoldó algoritmust. Mivel az előforduló mátrixok betöltöttsége 2o-5o % körüli, az elérhető nyereség nem áll arányban a sparse mátrix technika nagyobb programhosszával és időigényével.

Megjegyzendő ugyanakkor, hogy az egyenletrendszer megoldásakor egy módon kihasználjuk a mátrix ürességét. A Gauss-Jordan megoldórutin minden lebegőpontos összeadás és szorzás előtt megvizsgálja az argumentumokat. Ha bármelyik is zérus, az eredmény triviális -- ilyenkor a müvelet nem kerül tényleges elvégzésre.

A 27 bites lebegőpontos mantisszahosszuság sajnos korlátozza az eredmények pontosságát. 1 ohm minimális ág-ellenállás mellett az eredmények addig elfogadhatók, ameddig a csomópontok és a föld közötti impedancia $1o^7$ - $1o^8$ ohm alatt marad. Ez a korlátozás tulajdonképpen feloldható volna egy nagyobb szóhosszuságú lebegőpontos szubrutincsomaggal, de csak a memóriaigény és a futási idő jelentős növelése árán. Szerintünk a jelenlegi megoldás a legel-

fogadhatóbb kompromisszum.

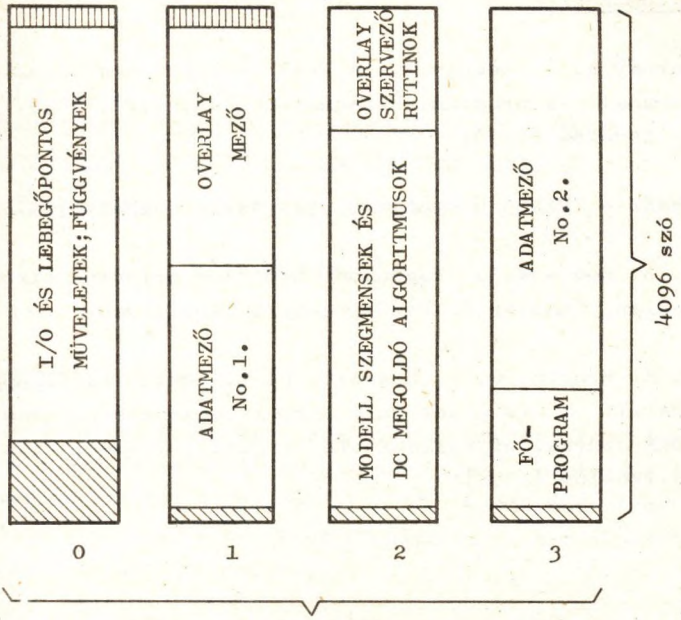
A 3. ábrán a CIRCUIT ANALYSIS szegmens memóriakiosztását látjuk. A TPA-i gép memóriája 4 K-s modulokban szerveződik. A 0. modult a lebegőpontos és I/O rutinok, valamint a függvények foglalják el. A többi három modul 58 %-a programmező, 42 %-a adatmező. A programszegmensek egy része mágneslemezen tárolódik, s csak szükség esetén töltődik a kijelölt overlay területre. Még az overlay lehetőség kihasználása mellett is igen nagy gondot kellett forditanunk a program effektiv kódolására. Egy-egy analizisfeladat algoritmusának ugyanis mindig teljes egészében a központi memóriában kell lennie; az analizis közbeni szegmens-csere megengedhetetlenül növelné a futásidőt.

A mágneslemez háttértárral nem rendelkező TPA-i kiépitések számára a program második részének két változatát készitettük el (A ill. B változat). A COMPILER által forditott feladatot az igényelt analizisfajtától függően az egyikkel vagy a másikkal kell futtatni.

- x -

A bemutatott analizisprogram sok tervezési problémánál reális alternativa a nagygépi programokkal szemben. A "helyben lévő" kis gép könnyü, gyors hozzáférhetősége feltétlenül előny. Az analizis költsége kisebb vagy egyenlő a nagygépen végzett analizisével. A futási idők természetszerüen nagyobbak, de nem elfogadhatatlanok. Például egy 30 csomópontos lineáris hálózat DC analizise kb. 30 másodperc, mig egy nemlineárisé 2-8 perc. A 30 csomópontos hálózat AC analizise egy frekvencián 3-4 perc.

A TPA-i gépre készült TRANZ-TRAN analizisprogram használatbavételére igen gyorsan sor került. Már most, néhány hónappal a fejlesztési munka lezárása után több iparvállalat ill. egyetemi intézet használja a programot. Ugyanakkor a továbbfejlesztés kérdésével is foglalkozunk: célkitüzésünk a program interaktiv, grafikus displayt használó változatának kialakitása.



Az overlay mező rétegei:

- transzfer karakterisztika számitás és tranziens analizis
- AC analizis
- érzékenység vizsgálat
- DC dokumentálás

3. ábra

IRODALOMJEGYZÉK

1 Tarnay K. - Székely V.: A TRANZ - TRAN nemlineáris áramköranalizis program, Hiradástechnika, V.24, No.9, pp.257-264 (1973)

2 TRANZ - TRAN 3 használati utasitás, Budapest, (1975)

3 F. F. Kuo - W. G. Magnuson: Computer oriented circuit design, Chapter 8, Prentice-Hall, Inc. (1969)

4 F. H. Branin, G. R. Hogsett, R. L. Lunde, L. E. Kugel: ECAP-II, a new electronic circuit analysis program, IEEE Journal of Solid-State Circuits, V.SC-6, No.4, pp.146-166 (1971)

PARCIÁLIS DIFFERENCIÁL EGYENLETRENDSZERT ÁLTALÁNOS FELTÉTELEK MELLETT MEGOLDÓ PROGRAM RENDSZER

/ DIFFERENCIA MÓDSZER ALTALANOS PROGRAMOZÁSA /

Szőts Miklós

SZAMGÉP

BEVEZETÉS

Az előadasban a cimben szereplő program rendszer készitése során felvetődött programszervezési problémákról kivanunk szólni. A feladat altal felvetett numerikus matematikai vagy algoritmus problémákat - még oly érdekesek is - nem tárgyaljuk.

A program rendszer általanos feladata a differencia módszer alkalmazásával peremérték feladatokat megoldani. A megoldás menetében a differencia egyenletrendszer felállitása a döntő kérdés, ezért ennek a problémának általános megoldását tartjuk a rendszer magjának. A rendszert rugalmasan kivánjuk fejleszteni, s fokozatosan kiterjeszteni a megoldott feladatok sorát.

A RENDSZER ELEMEI

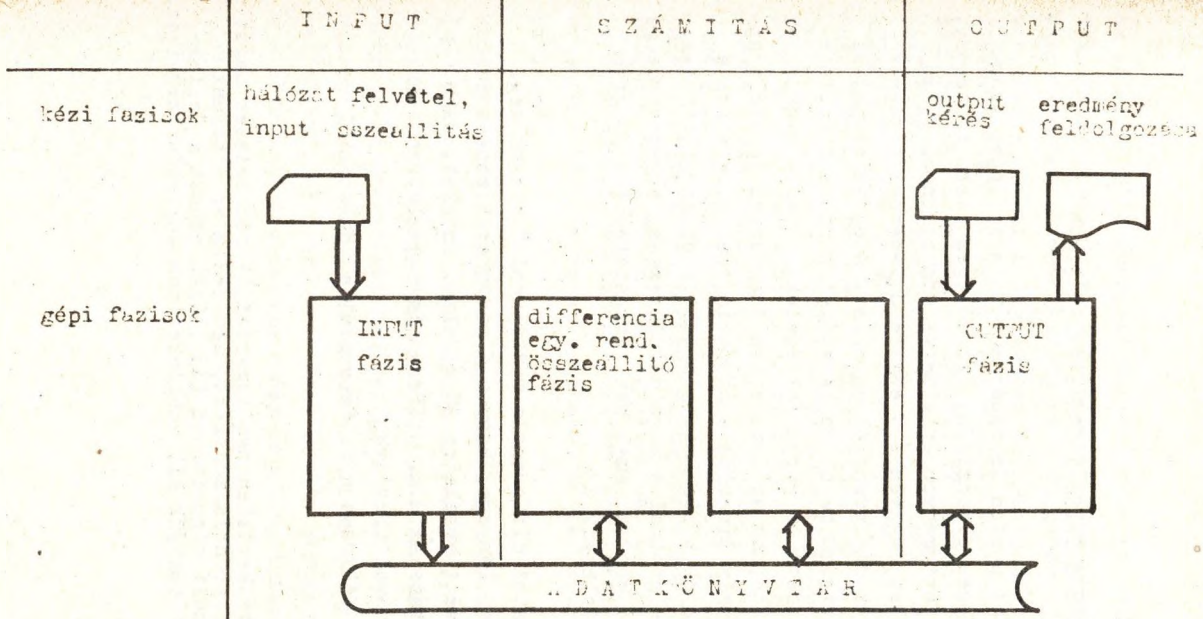
A használatban felvetődő egyes feladatoknak feleltetünk meg programokat. A megoldás módszerének logikájából fakadó nagyobb részfeladatoknak fázisokat. /A fázis elnevezést a Siemens 4004 rendszer terminológiája szerint használjuk: önállóan betölthető, futtatható egység./ Ezen belül a tipikus részfeladatokat könyvtárositott modulokként irjuk meg.

Az alapfeladat: tetszés szerinti tartományokon értelmezett lineáris differenciál egyenletrendszerek megoldása. Az ilyen feladatu programok felépitésének sémája az 1. ábrán látható.

Input

Több input fázist tervezünk. Természetesen szükséges lesz egy teljesen általános input, amellyel minden számitható feladat adatai megadhatók lesznek. Ezeknek modulokra való bontását igen célszerüen kell megtervezni, hogy ujra felhasználásukkal könnyen lehessen speciális input programokat készíteni. Ezek egyes gyakorlatban előforduló feladatokra, vagy feladat tipusok megoldásához fognak szolgálni.

Lehetséges több fázisból álló input elkészitése is. Itt "előtét" fázisokra gondolunk, amelyek az általános input fázis bemenő adatait állitják elő speciális esetekben. Ugyancsak valószinü cserélhető modulok felhasználása, amelyeket esetenként irnánk meg, s például a differenciál egyenletek együtthatóit, vagy osztáspontok koordinátáit számolnák.



1. ábra

Parciális differenciál egyenletrendszer megoldás menete

Differencia egyenletrendszer felirása

A feladat pontos leirása: adott tetszőleges számu, tetszőleges dimenzióju, tetszőleges alaku tartomány. Minden tartományon értelmezve van bizonyos számu ismeretlen függvény, s ezekre ugyanennyi lineáris parciális differenciál egyenlet. Adottak ezenkivül az egyértelmü megoldás létezését biztositó feltételek /kezdeti és perem feltételek/. Ezek a megkötések vonatkozhatnak az ismeretlen függvények deriváltjainak bármely lineáris kombinációjára, s lehetnek a tartományok közös határain csatlakozási feltételek. Adott feladatnál a megoldás létezése, egyértelmüsége elméleti kérdés, amelynek vizsgálatával a program nem foglalkozik.

A feladatot differencia módszerrel * oldjuk meg. A módszer lényege, hogy a függvények bizonyos pontokon felvett értékeire ad közelitő értéket. Ezek kiszámitása az u.n. differencia egyenletrendszer megoldásával történik, amelyek felirása a differenciál egyenletekből

* A következőkben nem ismertetjük részletesen magát a differencia módszert, sem pedig az általunk használt algoritmust [4] . Csak azokat a jellemzőket emeljük ki, amelyeket mondanivalónk megkiván.

az ismeretlen függvények interpolációs polinomokkal való helyettesitésével történik. Az általunk használt algoritmusban ez nemcsak magára a függvény értékére vonatkozik, hanem a függvények deriváltjainak bármely lineáris kombinációjára. Ezeket továbbiakban egyszerűen kombinációknak nevezzük. Egy részük ismeretlen - a differencia egyenletrendszer ismeretlenei - ezeket ismeretleneknek nevezzük. Minden ismeretlen kombinációhoz hozzárendelhetünk egy differencia egyenletet, azaz egyik parciális differenciál egyenletet a tartomány egy pontja koordinátáinak behelyettesitésével. Rövidség kedvéért ezt behelyettesitett differenciál egyenletnek nevezzük. Az igy kapott behelyettesitett differenciál egyenlet-ismeretlen párokat sorba rendezzük, tehát mindkettőhöz tartozni fog egy sorszám. Ez fogja meghatározni a differencia egyenletrendszer szerkezetét. A hozzárendelésnek nincs egyértelmü módja, elvileg tetszőleges.

A differencia egyenletek együtthatóinak számitására az [4]-ben található algoritmust használjuk. Az ott leirt módszer adott pont-környezet esetén különösebb program szervezési nehézséget nem vet fel. Annak meghatározása viszont, hogy melyik egyenlethez milyen környezetet rendeljünk, az igazi probléma. Ez adatstruktura kérdés, s a következő szakaszban ennek megoldását tárgyaljuk.

Egyenletrendszer megoldás

A kapott differencia egyenletrendszer nem szimmetrikus, ritka kitöltésü matrix, általában sáv strukturával, s maga a sáv is ritka. Legcélszerübb a kódolt tárolásu együttható mátrixu egyenletrendszer megoldasára irni fázist. Már meglévő, és más tárolási módot megkivánó egyenletrendszer megoldó fázisok - vagy modulok - szintén felhasználhatók. Ez esetben szükség van az együtthatómátrix tárolásának átalakitását végző modul elkészitésére.

Output

Az output fázis közvetlenül fogad inputot. Ebben határozhatjuk meg, milyen outputot kivánunk. Tervezett output fázisunk képes lesz a tartomány tetszőleges pontján tetszőleges kombináció szolgáltatására. Ha az adatkönyvtárt megőrizzük, a feldolgozás folyamán felmerült további igények kielégitéséhez elegendő lesz az output fázist futtatni. Hasonlóan az inputhoz, itt is egy teljesen általános és néhány speciális fázis megirására számitunk.

Az egyes fázisok cseréjével minimális munkával uj programok hozhatók létre. Már utaltunk az input, output fázisok cseréjére, de ki szeretnénk emelni, hogy ezzel megoldható pl. a nem konstans együtthatós egyenletek számitása / input fázis cseréje /, sőt a rácsvonal módszer alkalmazása / egyenletrendszer megoldás helyettesitése sajátérték feladat megoldásával /.

ADATKEZELÉS

A fent ecsetelt lehetőségek alapfeltétele az adatkönyvtár rugalmas és általános szerkezete. Mivel a differencia egyenletrendszert előállító fázis a rendszer "fix pontjának" tekinthető, elsősorban ehhez kell igazodnia.

Majdnem szabályos hálózat

A differencia módszer hagyományos használata szabályosan felvett hálón alapszik, ahol a hálópontokon felvett függvényértékek az ismeretlenek. Célkitüzésünk - az általános alaku tartomány figyelembevétele - ennek ellentmond, és - más körülményekkel együtt - lehetetlenné teszi. A kombinációkat bevezetve szükségünk van valamilyen strukturára, amely segitségével ki tudjuk majd választani a "szomszédosokat".

Tehát a tetszés szerint képzett kombinációkat, illetve, mivel minden kombinációhoz hozzárendelhető egy pont *, ezeket a pontokat olyan strukturába kell rendezni, hogy értelmezhető legyen a "szomszédos" reláció, amely bizonyos irányba való közelséget, sőt "legközelebbiséget" fejez ki. Az eddig használt gyakorlatban ilyen struktura a szabályos hálózat. Nézzük meg ennek tulajdonságait,

* Az itt emlitett hozzárendelés teljesen természetes, ha a kombináció minden eleme ugyanazon pont koordinátáinak behelyettesitésével adódik. Ha nem, egyik pontot önkényesen hozzárendelhetjük. Később látjuk, hogy mindegy, melyiket. Általában több kombinációt rendelünk egy ponthoz. Ennek a hozzárendelésnek itt csak az a célja, hogy a téma geometriai jellegü tárgyalása zavartalan legyen.

s ezeket változtassuk oly módon, hogy nem szabályos eloszlásu pontokra is alkalmazható strukturét kapjunk.

Szabályos hálót kapunk, ha egy n dimenziós tartományban - amelyet koordináta tengelyekkel párhuzamos hipersikok határolnak - minden koordináta irányra felveszünk néhány x_i = constans egyenletü hipersikot. Ezek metszéspontjai adják a hálózat pontjait. /Kétdimenziós esetben az x_1 és x_2 tengelyekkel párhuzamos egyenesek./ Matematikailag a következőképpen fogalmazhatjuk meg: minden koordináta irányban a hipersikok egyenletében szereplő koordináta értékekhez növekvő sorrendben indexeket rendelünk egytől kezdve. Ezen indexek halmazát jelöljük I_i vel, ahol az i alsó index a koordináta irányt jelzi. Ezeket az indexeket használva a tényleges koordináták helyett a hálózat minden pontja jellemezhető ezek rendezett n -esével:

$\langle i_1, i_2, \ldots i_n \rangle$, ahol $i_1 \in I_1$, $i_2 \in I_2, \ldots i_n \in I_n$.

Ezek halmaza definició szerint a I_i halmazok Descartes-féle /direkt/ szorzata: $H = \overset{n}{\underset{i=1}{X}} I_i$.

Szomszédosnak nevezünk két pontot az l-edik irányban, ha a két pontot reprezentáló rendezett szám n-es az l -edik elem kivételével megegyezik, s a két l-edik elem eltérése egymástól 1, tehát $\langle i_1, i_2, \ldots i_n \rangle$ és $\langle j_1, j_2, \ldots j_n \rangle$ szomszédos l-edik irányban, ha $i_k = j_k$ minden $1 \leq k < l$, $l < k \leq$ n-re és $i_l = j_l \pm 1$.
Ha két pontot szomszédosnak mondunk, ha valamilyen irány szerint szomszédosak. Mód nyilik a környezet fogalmának definiálására is. Az x_0 pont környezetének nevezzük a pontok olyan halmazát: $\{x_i\}$, ahol minden $x_j \in \{x_i\}$ -hez

találhatók $\{x_i\}$ olyan elemei, hogy van $\langle x_j, x_{i1}, x_{i2}, \ldots x_o \rangle$ sorozat, ahol az egymásra következő pontok szomszédosak.

A következőkre jutottunk: szabályos hálózat indexhalmazok direkt szorzata, ahol minden indexhalmaz egy koordináta iránynak felel meg, az indexhalmaz elemei pedig a megfelelő koordináta irányra merőleges hipersikok koordinátáinak indexei. Magát a matematikai felépitést és a szomszédos reláció értelmezését megtartva, csak egyes elemek jelentésén módositva egy feltételeinknek megfelelő strukturához juthatunk.

A változtatások a következők:

1./ Az egyes indexek nem a koordináta tengelyre merőleges hipersikot reprezentálnak, hanem attól kevéssé eltérő hiperfelületet.

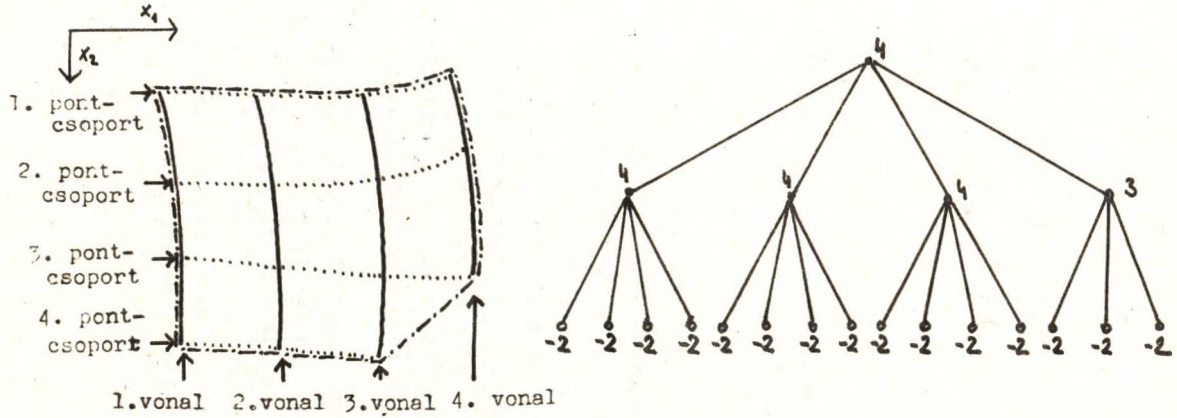
2./ A hálózat elemei - a rendezett szám n-esek, másszóval az indexsorozatok - nem magát a hiperfelületek metszéspontjait reprezentálják, hanem ahhoz közel eső pontok csoportját, azaz az azokhoz rendelt kombinációk halmazát. A továbbiakban nem jelent különbséget, hogy egy ilyen kombináció halmaz egy ponthoz tartozik, vagy egy pontcsoport pontjaihoz. Az itt bevezetett feltétel teszi lehetővé a hálózat sürítését egy pont körül.

A fentieknek megfelelő hálózatot majdnem-szabályos hálózatnak nevezzük. Mindkét kikötést igen szabadon, pontatlanul fogalmaztuk. A "kevéssé eltérő" és "közel eső" fogalmakat matematikailag is tudnánk definiálni, megfelelő normát találni rájuk, azonban ezt fölösleges-

nek éreztük. A lényeg az, hogy a szabályos hálózat torzitása olyan fokig mehet, mig a szomszédos reláció kimeriti a szomszédos szó szemléletes jelentését.

Logikai adatstruktura

A logikai adatstrukturánk az adatcsoportok egymásba skatulyázott, hierarchikus felosztásai, amely egy fa alaku gráffal jól szemléltethető [3]. A gráf minden csomópontja megfelel egy adatcsoportnak, s a csomópontokat követő csomópontok határozzák meg az adatcsoport felosztását. A fa végpontjai a legkisebb adatcsoportok, a fizikai adatstrukturát ezek szerint célszerü szervezni, azaz ezek hozzáférését biztositani. Konkrétan:
A kiindulás a teljes tartomány összes adata. Ezt az első koordináta irány szerint indexelt n-1 dimenziós hiperfelületek adataira bontjuk szét. Ezután egy n-1 dimenziós hiperfelületet tekintjük, ennek adatait a második koordináta irány szerint indexelt n - 2 dimenziós hiperfelület adataira bontjuk. Ezt az eljárást folytatjuk, amig magukat a 0 dimenziós pontcsoportok adatcsoportjait kapjuk. Konkrét példát láthatunk a 2. ábrán. Az ily módon kapott fa n+1 szintből áll. A végpontokon csatlakoznak hozzá az egyes pontcsoportok /kombináció csoportok/ részfái. Ezeket nem részletezzük, mivel felépitésük a matematikai algoritmushoz kötődik. A 2. ábrán a végcsomópontokhoz rendelt -2 index jelzi a továbbépités szükségességét. Hogy ez az adatstruktura adekvát, az is bizonyitja, hogy az egyes pontcsoportok eléréséhez szükséges indexsorozatok megegyeznek az előző pontban a pontcsoportokhoz rendelt rendezett szám n-esekkel.



tartomány határa

a felosztas x_1 szerint indexelt vonalai

a felosztás x_2 szerint indexelt vonalai

2. ábra
Példa majdnem-szabályos hálózat fájára

Fizikai adatstruktura

A fent leirt logikai adatstruktura megvalósitására két lehetőséget ismertetünk:

Random file

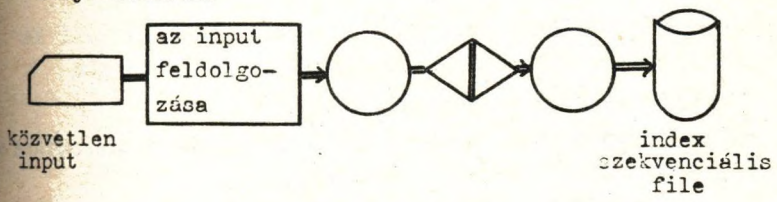
Az adatstrukturát jellemző fa végpontjaihoz tartozó adatcsoportokat random file-ban tároljuk. A randomizáló rutin a fát leiró hierarchikus tartalomjegyzék kezeléséből áll [3].

Index szekvenciális file

Azt a felismerést használjuk fel, amelyet az előző pont végén hangsulyoztunk: minden adatcsoportot egyértelműen jellemez a hozzá vezető indexsor - ezt választhatjuk a rekord kulcsának. / Természetesen most is a fa végpontjaihoz rendelt adatcsoportokat tároljuk az egyes rekordokban./ Ez az indexsor a feldolgozás során mindig adott, s konkrét jelentéssel bir, igy a rekordok nyerése nem jelent problémát.

Az adatkönyvtár felépitése egyszerűbb, mint az előző esetben a tartalomjegyzék kreálása. Először egy szekvenciális file-ba helyezzük a rekordokat az inputnak megfelelő sorrendben, majd - ha szükséges - rendezzük, s az igy kapott szekvenciális file lesz az index-szekvenciális file előállitásának inputja.

Folyamatábrán:



ÖSSZEFOGLALÁS

Előadásunkkal a következőket kivántuk kiemelni:

- Készithető általános program rendszer parciális differenciál egyenletrendszerek megoldására, ugy, hogy nem kötjük meg a tartomány alakját, dimenzióját, valamint figyelembe tudunk venni kapcsolódó tartományok illeszkedési feltételeit.

- Kellően bonyolult program rendszer tervezése csak szigoruan strukturália gondolkodás alapján hatásos, az egyes fázisok, modulok cserélhetősége figyelembevételével.

- Az egyes elemek cserélhetősége elsősorban az adatstruktura megtervezésétől függ. Az u.n. tudományos számitásoknál is előtérbe kerül az adatkezelés.

Irodalom jegyzék

[1] Holnapy D. - Szőts M.:
Peremérték feladatok megoldása Hermite-interpoláción alapuló differencia módszerrel.
Számológép. II.évf. 1. 5-14 /1972/

[2] Szőts Miklós - Varga Gabriella:
Lineáris parciális differenciál egyenletrendszer számitása majdnem-szabályos hálózat felett.
SZÁMGÉP kiadvány 1974

[3] Szőts Miklós:
Müszaki tervezésre orientált programrendszer modulelvü kiépitése.
Programozási rendszerek '72 konferencia előadása.

[4] Modul rendszer általános leirása
SZÁMGÉP kiadvány 1974

AZ ICL 1900-AS SLAM-PROGRAMCSOMAG ADAPTÁCIÓJA EGY FOLYTONOS MŰSZAKI RENDSZER SZIMULÁCIÓJÁRA

Füle Károly
NIM Ipargazdasági és Üzemszervezési Intézet

Az előadás tárgya az ICL 1900-as gépek SLAM-elnevezésü programcsomagjának adaptációja, különös tekintettel egy komplex müszaki szimulációs feladat megoldására.

A SLAM-rendszer folytonos rendszerek szimulációjának segéd-eszköze digitális számológépen. Erre utal a rendszer angol elnevezése - Simulation Language for Analogue Modelling - is, amelynek kezdőbetüiből a SLAM-betüszó származik.

A SLAM-rendszer adaptációja során számos müszaki, matematikai, és /az ipari dinamika köréből vett/ gazdasági feladat modellezését, programozását és számitását végeztük el. Ezek az ujjgyakorlatok lehetővé tették, hogy egy komplex müszaki feladat megoldását is napirendre tüzzük. A megoldás során számos nehézséget kellett áthidalnunk, közben viszont a SLAM-rendszer realisztikusabb értékeléséhez is eljutottunk.

Az előadás az alábbi témaköröket öleli fel:
- Először vázlatosan ismertetjük az eszközt, azaz a SLAM-programcsomagot.
- A következőkben röviden vázoljuk a feladatot, azaz a konkrét müszaki rendszert.
- Végül - kissé részletesebben - beszámolunk a megoldásról, azaz a müszaki szimuláció számitástechnikai kidolgozásáról. Miközben a kidolgozás általánosabb tapasztalatairól beszámolunk, egyszersmind elvégezzük a SLAM-rendszer átfogó értékelését is.

1. A SLAM-programcsomag ismertetése

A digitális számológépen megvalósitott analóg szimulációnak - röviden a digitális-analóg szimulációnak - kétféle változata ismeretes:

- Analóg számológép szimulátor program esetében a digitális számológépen kész programot futtatunk, és a program bemeneti adataiként adjuk meg a szimulációs feladat kapcsolási rajzát.
- Analóg szimulációs nyelv esetében a digitális számológépen csak forditóprogram áll rendelkezésre. A megoldandó feladatra a szimulációs nyelven programot kell irni. A megirt programot a forditóprogrammal forditjuk le gépi nyelvü programmá, vagy más forditóprogram által értelmezhető programmá.

A SLAM-nyelv olyan analóg szimulációs nyelv, amelyet a hozzá tartozó transzlátor-program FORTRAN-nyelvre fordit le.

- A SLAM-nyelv ezért úgy tekinthető, mint egy a FORTRAN-nyelv fölé rendelt nyelv, legalábbis a vonatkozó kézikönyv szerint. Ez elvileg azt jelenti, hogy a SLAM-nyelv egyrészt magasszintü nyelv, másrészt bizonyos szempontokból még a FORTRAN-nyelvnél is magasabb szintü nyelv.
- A SLAM-nyelv ugyanakkor erősen korlátozza a FORTRAN-nyelv magasszintü lehetőségeinek használatát. Más kérdés, hogy a folytonos szimuláció hagyományos feladataihoz ilyesmire nem is nagyon van szükség.

A leggyakoribb SLAM-utasitás az értékadó utasitás. Formátuma tökéletesen megegyezik a FORTRAN-nyelvü értékadó utasitásával. A baloldalon álló kimeneti változó értékét a jobboldalon szereplő bemeneti változók és állandók értékéből számitjuk ki, aritmetikai müveleteken keresztül. A SLAM-nyelvü értékadó utasitás rendszerint több aritmetikai müveletet fog össze, szemben az analóg kapcsolási rajzzal, ahol minden egyes aritmetikai művelet eredménye explicite is előáll.

A SLAM-nyelvü program első közelitésben értékadó utasitások rendezetlen halmaza. Az utasitások felirási sorrendje közömbös. Az utasitások végrehajtási sorrendjét a transzlátor-program határozza meg. Az automatikus sorbarendezés során minden utasitás olyan helyre kerül, hogy összes bemeneti változója már előbb értéket kapjon a megfelelő értékadó utasitások baloldalán.

A folytonos szimulációra jellemző, hogy a modell többszörösen hurkolt - zárt, illetve visszacsatolt - rendszer. A rendszer akkor szabályos, ha minden egyes hurokban előfordul legalább egy integráló elem. A SLAM-nyelvben az integrálás az INTGRL-nevü valós függvény hivásán keresztül történik, az értékadó utasitás jobboldalát képező aritmetikai kifejezésben. Az integráló függvény akár többszörösen is egymásba skatulyázható. Az igy adódó program rendkivül tömör, alig hasonlit az analóg szimulációhoz készitett kapcsolási rajzokhoz. Az integráló függvény hivását tartalmazó értékadó utasitást a transzlátor-program nem is hagyja egyben, hanem - munkaváltozók bevezetésével - több értékadó utasitást képez belőlük. Automatikus sorbarendezésre az igy képzett értékadó utasitások kerülnek, azzal a feltételezéssel, hogy az egyes integrál-értékek mindig az elsők között kapnak értéket.

A SLAM-nyelvü program - a digitális számológépen történő számitás szekvenciális jellegét visszatükrözve és kihasználva - alapvetően három tartományból /region/ áll:

- A bevezető tartomány/initial region/ a futás legelején és egyetlen egyszer kerül sorra. Feladata: a program állandóinak beállitása vagy beolvasása, kezdeti értékadások különféle szimulációs statisztikákhoz, cimlap és fejléc nyomtatása, stb.
- A dinamikus tartomány/dynamic region/ a futás során ciklikusan hajtódik végre, a befejezési feltétel teljesüléséig. Minden egyes ciklus elején meghatározódik az összes integrál uj értéke, kivéve az első lépést, ahol ez az uj érték a kezdeti értékkel egyenlő. A ciklus többi

része lényegében az összes integrandus uj értékének meghatározása az értékadó utasitások megfelelően sorbarendezett sorozata szerint.

- A befejező tartomány/terminal region/ a futás legvégén é egyetlen egyszer kerül sorra. Feladata: a szimulációs st tisztikák kinyomtatása, esetleg az eredmények alapján op malizációs számitás elvégzése, a program állandóinak mód tása, és az egész szimuláció megismételtetése, egészen a optimális megoldás eléréséig.

A SLAM-nyelvü program alkalmas nagyüzemi igényü szimuláció vizsgálatok elvégzésére is.

- A program állhat több szegmensből is; az elsőt master sze mensnek nevezzük. Ezek a szegmensek egymástól független s mulációkat folytathatnak le. A szegmens első utasitásai d gitális programrészletet is képezhetnek, utána a tulajdon képpeni analóg program bármelyik tartománya el is maradhat Igy a master szegmens ugy is megirható, mint egy tisztán gitális vezérlő program, az alárendelt analóg programok mü désének összehangolására.
- A tulajdonképpeni analóg programban - a SLAM-szegmens hán tartományán belül - ismét irhatunk digitális programrészl teket az un. NOSORT-blokkokban. Igy az analóg programnak esetleg egy komplett digitális szimulációt is alárendelhe tünk.

2. A folytonos müszaki rendszer ismertetése

A szimulációra bocsátott folytonos müszaki rendszer komplex tását az alábbiakban kivánjuk érzékeltetni:

- A rendszerben termodinamikai állapotváltozásokat és áraml technikai mozgásokat kisérünk figyelemmel.
- A rendszerben háromféle közeg - viz, vizgőz és levegő - v részt, és ezek a közegek egymással keverednek. Az ilyen r szert háromfázisu rendszernek is nevezzük.

- A rendszerbeli közegek helyiségekben tartózkodnak, és a helyiségek közötti nyilásokon keresztül áramolnak. A helyiségek és nyilások geometriai rendszerét általánosan n-elemü és m-kapcsolatu rendszernek tekintjük, ahol n és m értéke változhat.
- A rendszer kezdetben fennálló egyensulyi állapotát nagynyomásu és magas hőmérsékletü viz robbanásszerü betörése bontja meg. A bekövetkező termodinamikai és áramlástechnikai folyamatokban ezért az állapotjelzők igen széles tartományban változnak, és a változási sebesség is igen nagy. Ezek a tényezők a müszaki rendszer realisztikus modellezését rendkivül megnehezitik.

A teljesség kedvéért meg kell emlitenünk, hogy a szimuláció jelenleg holtponton áll. Az összes számitástechnikai nehézséget sikerült áthidalnunk. A számitási eredmények tükrében azonban ugy tünik, hogy a modell egyelőre a benne szükségszerüen eszközölt müszaki közelitések áldozata. A közelitések eredőjeként ugyanis müszakilag nem értékelhető eredmények születtek. Pontosabb összefüggések viszont egyrészt nem is állnak rendelkezésre, másrészt igen megnövelnék a már amugyis nagy forditási-futási időt.

3. Az adaptáció tapasztalatai

Az adaptáció során szerzett fontosabb tapasztalatainkat az alábbi szempontok köré csoportositjuk:

- Az integrálás megoldása;
- A makrók irása és használata;
- Az automatikus sorbarendezés.

3.1. Az integrálás megoldása

A digitális gépen történő analóg szimulációtalán legkritikusabb pontja a numerikus integrálás. Szinte azt is mondhatnók, hogy egy jó szimulációs nyelvtől nem is várunk mást, mint a numerikus integrálás lehető legtökéletesebb megoldását.

Általánosságban elmondhatjuk, hogy a SLAM-programnyelv által nyujtott integrálási rendszer rendkivül ragalmas, nem hagy kivánnivalót maga után. A SLAM-rendszerü integrálás fontosabb jellemzői a következők:

- A numerikus integrálást az INTGRL-nevü valós függvény hivása utján végeztethetjük el. A függvény két paramétere - az integrandus és a kezdeti érték - tetszőleges valós aritmetikai kifejezés lehet. Maga a függvény is hivható aritmetikai kifejezésben, és más függvényekbe - sőt önmagába is - beágyazható.
- A numerikus integrálás - az integranduson és a kezdeti értéken tul - még számos paraméterrel rendelkezik. Ilyen paraméterek: az integrálás számitási algoritmusa, a lépésköz/a kommunikációs intervallum/ nagysága, a relativ hiba százalékos értéke, és még több más érték. Ezeket a paramétereket nem függvényhivásonként tüntetjük fel, hanem több hiváshoz együttesen rendeljük hozzá. A dinamikus tartományon belül un. derivativ szakciókat képzünk, és az egyes derivativ szekciókon belül a bennük előforduló függvényhivásokhoz közösen adjuk meg az egyéb paramétereket az erre a célra szolgáló un. INTINF-blokkban. Az egyes paramétereknek feltételezett /default/ értékük is van, csak az ezektől eltérő értékeket kell megadnunk. A paraméterek értékét a program egy-egy változóján keresztül is megadhatjuk, ilyenkor a paraméter értéke futás közben változhat is. Érdemes kiemelni, hogy az integrálás 6 különféle módszer szerint történhet/trapéz-módszer, Simpson-szabály, fix és változó lépésközü Runge-Kutta, predictor-corrector, Adams-Moulton/, és ez a módszer is változhat a futás közben.
- A numerikus integrálás miatt a SLAM-program dinamikus tartománya tulajdonképpen sürübben hajtódik végre, mint amilyen sürün az előirt lépésköz szerint kellene. Egyes esetekben ez kellemetlenül megnövelheti a számitási időt, a pontosságot viszont lényegében nem fokozza. Erre példa lehet az un. függ-

vénygenerálás, azaz az adott értéktáblázat alapján történő lineáris interpoláció. Az ilyen számitásokat a derivativ szekciókból eleve kiemelhetjük, és közöttük un. paralel szekciókban foglalhatjuk.

3.2. A makrók irása és használata

A digitális-analóg szimuláció talán legkényelmetlenebb tulajdonsága az, hogy a programozónak rendkivül sokat kell irnia. Ez a megállapitás a SLAM-programnyelv esetében is igaz, jóllehet ez a nyelv - a FORTRAN-bázisnyelv révén - tulajdonképpen magasszintü.

A SLAM-programnyelv esetében a makrók irása és használata hivatott a programozó irásbeli munkájának csökkentésére.

- A SLAM-makrók irása és használata szempontjából a SLAM-programnyelven megirt programot inkább csak karaktersorozatnak kell tekintenünk. A makrótörzs karaktersorozata bármit tartalmazhat. A makró hivása is bárhol megtörténhet, ahol egyébként utasitás kezdődnék. A makróhivás helyére a makrótörzs karaktersorozata minden további nélkül behelyettesitődik. Érdemi szintaktikai vizsgálatra sem a hivó program, sem a makró nem kerül, csak a belőlük előállitott program. Hasonló megfontolások érvényesek a makróparaméterek használatára is. A makró hivásakor az aktuális paraméter gyanánt álló karaktersorozat előbb minden további nélkül behelyettesitődik a formális paraméter által jelzett heylekre a makró törzsében, és azután történik a makrótörzs behelyettesitése a hivó programba. A formális paraméter bármilyen karaktersorozatot képviselhet, illetve az aktuális paraméter gyanánt bármilyen karaktersorozat állhat. Külön is megemlitjük, hogy a makrók egymásba is skatulyázhatók, maximum lo szinten.
- A SLAM-makrók szerepével kapcsolatban érdemes külön is kihangsulyozni, hogy a behelyettesitett makrók számottevően növelik a tárgyprogram terjedelmét, Vezérlésátadási-adatátviteli problémák viszont nincsenek. Emiatt a SLAM-makrók

alapvetően különböznek a FORTRAN-szubrutinoktól, vagy az ALGOL-eljárásoktól

- Itt emlitjük meg, hogy a SLAM-programban NOSORT-blokkon belül minden további nélkül hivható FORTRAN-szubrutin is. A FORTRAN-nyelvü szubrutint külön szubrutin szegmensként forditjuk le a SLAM-forrásprogramból generált FORTRAN-nyelvü master szegmenssel együtt. Az adatátviteli problémákhoz érdemes megjegyezni, hogy a SLAM-rendszer megengedi a FORTRAN-nyelvü COMMON-utasitások használatát/bár azokat nem elemzi/.

3.3. Az automatikus sorbarendezés

A digitális-analóg szimuláció talán leginkább vitatható pontja az automatikus sorbarendezés. Szinte az is igaz, hogy az automatikus sorbarendezésnek legalább annyi hátránya van, mint amennyi előnye.

Mi az automatikus sorbarendezés szerepe? A digitális-analóg szimuláció összefüggéseiből számitási diagramot rajzolhatunk. Ez a számitási diagram elsősorban az analóg-analóg szimulációnál szokásos kapcsolási rajzhoz hasonlit, és nem a digitális gépek hagyományos programozásánál megszokott blokkdiagramhoz. Az analóg gépek müveleti egységeinek összekapcsolása a kapcsolási rajz alapján történik. A gép párhuzamos müködéséből következően sorrendi probléma a program végrehajtása során nem lehet. Nyilvánvalóan az egyes müveleti egységek összekapcsolása - a programozás - is tetszőleges sorrendben történhet. A digitális gépek viszont szekvenciális müködésüek, csak egymás után végezhetik el azt, amit az analóg gépek egyszerre oldanak meg. A végrehajtási sorrendet az összefüggések felirási sorrendje képviseli. Ésszerünek látszik, hogy az analóg-analóg szimulációnál megszokott előnyt a digitális-analóg szimulációnál is biztositsuk. Ezért a forditóprogram - igy a SLAM-rendszer transzlátor-programja is - magára vállalja az utasitások sorbarendezését. A sorbarendezés

logikai alapja az, hogy minden egyes értékadó utasitás jobboldalán csak olyan változók szerepelhetnek, amelyek előzőleg már előfordultak egy másik értékadó utasitás baloldalán.

Mi az automatikus sorbarendezés hátulütője? Az egyes értékadó utasitások végrehajtási sorrendjét a transzlátor-programnak kell felismernie, és az utasitásokat ennek megfelelően át kell rendeznie. A programszerü felismerés eleve kizárja a magasszintü nyelvek számos lehetőségének igénybevételét. A SLAM-programban is lényegében csak értékadó utasitások irhatók, jobboldalukon többé-kevésbé egyszerü aritmetikai kifejezésekkel. Vezérlésátadás, ciklus, szubrutinhivás,stb. értelemszerüen nem - illetve csak rendkivül körülményesen - programozható. Ezeknek a magasszintü lehetőségeknek a kizárása mellett alighe mondható egy nyelv magasszintünek. A legkellemetlenebb következmény viszont a tömbök használatának majdnem teljes kizárása. A SLAM-nyelvü program lényegében csak egyszerü változókkal tud dolgozni. Még állandó indexü tömbelemek sem használhatók. Ebben az esetben az automatikus sorbarendezés még elvégezhető volna, ugyanakkor ez számos - itt nem részletezett-előny forrása lehetne.

Az automatikus sorbarendezésről - az elvégzett kisérletek alapján - a következőket lehet még elmondani:

- Az automatikus sorbarendezés legfontosabb előnye talán az, hogy a sorbarendezés egyben ellenőrzés is. Ha a transzlátor program hibát nem jelzett, biztosak lehetünk abban, hogy minden egyes változó uj értékének számitásához formálisan megadtunk egy összefüggést is, és abban is, hogy ezek az összefüggések formálisan a helyes sorrendben kerülnek számitásra. Más kérdés, hogy ez a program egészére kiterjedő formális ellenőrzés a programozó szemében gyakran elegendőnek tünhet, és a csak általa végezhető tartalmi ellenőrzés esetleg elmarad.
- Az automatikus sorbarendezés előnye továbbá az is, hogy végülis a programozónak nem kell az összefüggéseket a vég-

rehajtási sorrendben fölirnia. Aki irt már digitális gépre bonyolultabb analóg programot, az tudja, hogy az ilyen sorbarendezés nem is olyan egyszerü feladat. Más kérdés, hogy ez a program egészére kiterjedő szabadság a programozó részéről szabadosságba is átcsaphat, és az egyes összefüggéseken belül is pontatlan megadásban jelentkezhet.

- Az automatikus sorbarendezés egyik legfontosabb hátránya éppen az, hogy egyetlen sorrendi-logikai hiba esetén is az összes programváltozóra nézve fog hibát jelezni a transzlátorprogram. Ez végülis majdnem annyi, mintha nem is jelezne hibát. Saját tapasztalatunk: 226 kijelzett hibát tüntettünk el egyetlen hiba kijavitásával, de a valódi hiba megtalálásáig csaknem az egész programot át kellett néznünk.
- Az automatikus sorbarendezés másik hátránya az, hogy ez a művelet igen sok időt vesz igénybe a számológépen is. Saját tapasztalatunk: az ICL 1903a tipusu gépen a SLAM-transzláció kereken 8 percig tartott, mig a generált FORTRAN-program fordítása mindössze csak 1,5 perc volt. Az ilyen hosszu transzlációs idő igen megdrágitja az amugyis időigényes szimulációt, hiszen a forrásprogramon a bejátszás során sokszor kell javitani.

Összefoglalásként elmondhatjuk: sokkal előnyösebb a folytonos szimulációt SLAM-programnyelven programozni, mint analóg számológépre ültetni, vagy analóg számológép szimulátor programmal elvégeztetni, de a SLAM-programnyelv magasszintüsége - a FORTRAN-bázisnyelv és számos lehetőség ellenére is - véleményünk szerint vitatható.

PORTÁBILIS PROGRAMOZÁSI RENDSZER IMPLEMENTÁLÁSÁNAK TAPASZTALATAI

Márton János
Központi Statisztikai Hivatal

Bevezetés

A portabilitás software rendszerek azon tulajdonságát jelenti, hogy mennyi nehézséggel vihetők át egyik számitógép környezetből a másikba. Egy rendszert akkor tekintünk portábilisnak, ha sokkal kevesebb munkával tudjuk egy új környezetbe átvinni, mint ott újra előállitani /kódolni/.

A gyakorlat, úgy is mint az elmélet próbaköve, késik a portabilitási problémára adódó válasszal. Ennek fő oka az, hogy túlságosan sok lépésben dönthető csak el egy software rendszer portabilitására vonatkozó kijelentés igazsága. Általában olyan portábilis rendszereket ismerünk, amelyek valójában csak demonstrációi valamely portabilitási elképzelés helyességének. Természetes, hogy az ilyen rendszerek jó eredményekkel dicsekedhetnek, hiszen olyan körülmények között próbálják ki őket, amelyek lényegében kisérletiek.

Ezért úgy véljük, hasznos lehet egy olyan rendszer bemutatása, amely nemcsak célul tüzte ki a portabilitást, hanem azzal létező felhasználók valódi igényeit kivánja kielégiteni. E felhasználók a különböző államok

statisztikai hivatalai, a feladat pedig a statisztikai adatfeldolgozás problémáinak átfogó megoldása. A statisztikai hivatalok gépi felszereltsége eléggé változatos, ugyanakkor nagyon is hasonló természetü gondokkal küzdenek. Ez az az eset tehát, amikor egy megfelelő software portábilis mivoltát jól ki lehet használni.

A rendszer elnevezése Integrated Statistical Information System /ISIS/. A pozsonyi Computing Research Centre /CRC/ az ENSZ megbizásából fejlesztette ki. Az ISIS alapötlete az európai statisztikusok 1970-es konferenciáján született. 1971-ben nyujtotta be a CRC az ISIS tervezett tulajdonságaira vonatkozó ajánlását. Ebben már kifejtik az ISIS portábilitásának szükségességét. Azóta elkészült a rendszer első változata és a KSH az első idegen felhasználó, aki éles feladatok megoldására kivánja felhasználni.

A poszonyi CRC-ben az ISIS-t egy CDC 3300 tipusú számitógépen készitették el. A KSH-nak a CDC-től tökéletesen idegen IBM 370/155 gépére való átültetés az ISIS portabilitásának főpróbája. Ez a munka most folyik, már vannak tapasztalataink és az ISIS-nek egyes részei már müködnek az új környezeti feltételek között.

1. Az ISIS feladata

Előljáróban szükségesnek látszik vázlatosan felsorolni azokat a lehetőségeket, amelyeket a felhasználó az ISIS-ben megkap. Ez főleg azért fontos, mert igy képet alkothatunk magunknak az ISIS tervezésekor negoldott feladat méretéről és bonyolultságáról.

Az ISIS lényegében egy nagy adatkezelő rendszer, amely a statisztikai adatfeldolgozás természetének megfelelően különbözik a hasonló rendszerektől. Alapvető tulajdonságai a portabilitás és a modularitás.

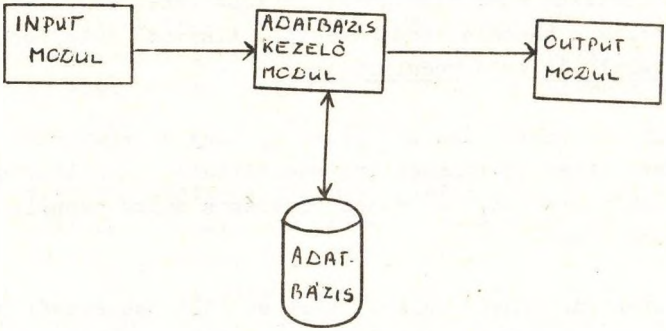
Az ISIS portabilitása azt jelenti, hogy a legkülönbözőbb környezetekben is könnyen implementálható. A gyakorlatban a tervezők szerint, az ehhez szükséges munka mennyisége 3-6 emberhónap.

A modularitás annyit jelent, hogy az ISIS összetevői önállóan is működőképes egységek, és a felhasználó dönthet el, hogy melyikre tart igényt és melyikre nem. Természetesen csak az összes modul együttes használata biztositja automatikusan a részfolyamatok összhangját.

Az ISIS három modulból áll:
- input modul
- adatbázis kezelő modul
- output modul.

A moduláris felépités lehetővé teszi, hogy az ISIS-ben kétféle adatfolyamot valósitsunk meg:

- Az egyikben az adatok, miután megjárták az input modul hibaellenőrző és javitó cilusait, az adatbázis kezelő modulon keresztül az adatbázisba kerülnek. Innen különféle jellemzők alapján visszakereshetők és az output modullal feldolgozva elégithetik ki a felhasználó adatigényét.



- A másikban az input modullal ellenőrzött és javitott adatok megmaradnak az adatbázison kivüli szerkezetben /ún. külső file-ok/. Ezt a szerkezetet leirjuk az ISIS e célra szolgáló eszközeivel és a leirás alapján az output modul ugyanúgy tudja kezelni, mintha az adatbázisból érkező adatokról lenne szó.

- Ez a két lehetőség több, mint amennyit a szokványos rendszerek megengednek és nagyon fontos a statisztika szempontjából, ahol általában az adatok töredékrészét érdemes csak adatbázisba szervezni /pl. népszámlálási adatok/.

Az ISIS a nyelvek szempontjából is nagyon rugalmas.

Van saját felhasználói nyelve, amely tulajdonképpen célnyelvek rendszere, de a célon belül a magasszintü nyelvekben megszokott általános lehetőségeket is biztositja. Ugyanakkor lehetséges az is, hogy az ISIS kivánt eljárásait valamely más nyelvü /pl. COBOL/ programba beágyazva használjuk.

Összefoglalva megállapithatjuk, hogy egy bonyolult, rugalmas és nagyteljesitményü rendszerről van szó. A rendszer tervezése és kivitelezése sok és értékes tapasztalatot eredményezett.

2. Az ISIS megvalósitása

A CRC-nek nyújtott ENSZ támogatás határidőre szólt és ez nagymértékben sürgette a rendszer elkészitését. Olyan technikát kellett tehát alkalmazni a rendszer kivitelezésénél, amely lehetővé teszi, hogy rövid idő alatt a sokirányú követelményt kielégitő és hibamentes software-t lehessen előállitani.

A kivitelezés során úgy tünt, hogy a struktúrált programozás és az ennek megfelelő munkaszervezés biztositja a leggyorsabb és a legmegbizhatóbb eredményt.

A struktúrált programozás, mint módszer, megfelelő, de kellett találni egy olyan leiró nyelvet, amely eleget tesz az alábbi követelményeknek:

- géptől független, magasszintü nyelv;
- ne rontsa a programok hatékonyságát 1.2 faktornál jobban az adott gépi nyelven /asszemblerben/ irt programokhoz viszonyitva;

- könnyitse a munkaszervezést: legyen érthető, olvasható, öndokumentáló és rendelkezzen megfelelő hibafelfedő képességgel;
- egyszerü forditóprogramja legyen, amelyet a rendszer egészénél sokkal könnyebb megvalósitani;
- megfelelő legyen rendszerprogramok irására is, azaz rendelkezzen minél több adat és utasitás struktúrával;
- alkalmazkodjon a struktúrált programozás követelményeihez /pl. ne legyen benne GOTO tipusú ugró utasitás/.

A választás az újabban befutó PASCAL nyelvre esett, amelynek általánossága a PL/I-éhez hasonló, de annál sokkal egyszerübb szerkezetü.

A PASCAL azon tulajdonsága, hogy a forditóprogramját önmagával is le lehet irni, döntő jelentőségü a portabilitás szempontjából. Ezen kivül is úgy tünik, hogy igen jól bevált és igazolta a hozzáfüzött reményeket. Egyedül a hatékonyságot illetően támadható, de majd látni fogjuk, hogy ennek fokozására is van lehetőség.

3. Az ISIS technikai felépitése

A felhasználói lehetőségeket a portabilitási követelmény gyakorlatilag nem befolyásolta. Érezhetően hatott azonban a megvalósitás elveire és döntéseire és különösképpen meghatározó jelentőségü a rendszer technikai felépitését tekintve.

Technikailag az ISIS-t kétféle szempont alapján tudjuk elemezni. Az egyik szempont szerint úgy bonthatjuk fel

a rendszert /teljesen eltekintve a moduláris elrendezéstől/, hogy annak két része:

- az algoritmusokat megvalósitó részek összessége és
- a környezettel való kapcsolatot biztositó részek összessége.

Ez utóbbi részt még tovább bonthatjuk;

- input/output müveleteket végző részre és
- programok /részfolyamatok, munkaegységek, "task"-ok/ közötti kommunikációt biztositó részre.

Ez a felosztás lényegében egybeesik a portabilitás megoldási lehetőségeinek megoszlásával. Amig az algoritmus rész portabilitása a leginkább kiforrott, addig csak kisérleteznek a könyezeti kapcsolatot jelentő rész portabilitásával. Bár ma még az ISIS-ben ezt a feladatot a célkörnyezet asszemblerében kell elvégezni, biztató kezdeményezés az Environment Language /EL/ kifejlesztése, amellyel tovább fokozódhat majd a rendszer portabilitása.

Egy másik technikai szempont szerint két alrendszerre oszthatjuk a rendszert, amelyek között a használatban szoros a kapcsolat, de a megvalósitásukban alkalmazható módszerek gyökeresen különböznek. Tulajdonképpen kétféle nézetét látjuk itt az ISIS-nek:

- az ISIS, mint nyelvi rendszer és
- az ISIS, mint eljárások összessége.

Az ISIS alapötletét 197o-ben megfogalmazó szakemberek tulajdonképpen azzal a céllal gyültek össze, hogy rögzitsék egy statisztikai feladatok megoldására alkalmas

célnyelv követelményeit. Az ISIS ezt az elképzelést beolvasztotta és mindhárom moduljánál alkalmazta. A modulok egyenként jelentenek egy-egy nyelvet, de ezek a nyelvek rokon szerkezetüek és a jövőben az a tervezők szándéka, hogy egyesitsék őket. Bár a munkaszervezést könnyebbé tette az elkülönült nyelvek és forditóprogramok rendszere, bonyolitja az új környezetbe történő átvitelt.

Itt is tovább bonthatjuk a nyelveket feldolgozó forditóprogramokat:

- szintaktikai elemző részre, valamint
- szemantikai rutinok csoportjára.

Ez a felbontás azért érinti a portabilitást, mert amig az eljárások és szemantikai rutinok leirása a megszokott leiró nyelvvel /PASCAL/ történt, addig a szintaktikai elemzőket egy Translator Writing System-mel /TWS/ a nyelvtanból közvetlenül állithatjuk elő. Maga a TWS PASCAL-ban iródott, de igy is jelentősen csökkenti az átvitelhez szükséges induló programszöveg mennyiségét, s igy emeli a portabilitás szintjét.

4. Az ISIS implementálásának eszközei és módjai

A software rendszer öt nagyobb egységből áll:

1. A PASCAL leiró nyelv forditóprogramja
2. A TWS rendszer
3. Az input modul nyelvének forditóprogramja /INLAN/
4. Az output modul nyelvének forditóprogramja /OUTLAN/
5. Az adatbázis kezelő rendszer

Előlegezve ide lehet még sorolni az EL környezetleiró nyelvet, valamint azokat a kiterjesztéseket, amelyek előfeldolgozóként /preprocesszor/ biztositják az ISIS eljárások beágyazását valamilyen általános célú nyelvbe.

Az ISIS bármely összetevője PASCAL, TWS vagy asszembler nyelven áll rendelkezésre. Ennek alapján kétféle módon lehet átvinni az ISIS-t egy új környezetbe:

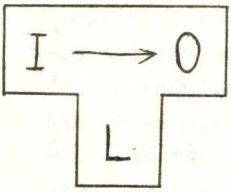
I. Installáló eszközök segitségével.
Ilyenkor a legelső installálási lépés a PASCAL forditóprogram felélesztése az új környezetben. Célszerü rögtön le is ellenőrizni a PASCAL-ban irt forditóprogrammal. Ezt követően lehet a PASCAL-ban irt eljárásokat és szemantikai rutinokat, valamint a TWS rendszert átvinni. Végül az új TWS segitségével elő kell állitani a megfelelő nyelvek szintaktikai elemzőit. Végül mindaddig nem müködik a rendszer, amig nincsenek meg a kivánt asszembler rutinok.

II. Installáló eszközök nélkül.
Ilyenkor nem került át sem a PASCAL forditóprogram, sem a TWS az új környezetbe, csupán azt tesszük lehetővé, hogy a tárgykód generálás az új környezet követelményeihez igazodjon. Ebben az esetben azonban az átvitelhez szükséges futásokat a kiinduló környezetben kell végrehajtani. Az asszembler rutinok itt is szükségesek.

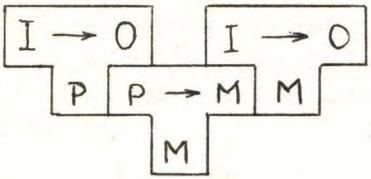
Az installáció tulajdonképpen nem egyéb, mint a megfelelő programkönyvtárak felállitása és feltöltése. Célszerü az első módot választani, mert ezzel a későbbiekben is biztositjuk, hogy a rendszer alkalmazkodni tudjon a változó felhasználói igényekhez.

5. Az ISIS portabilitásának kulcsa

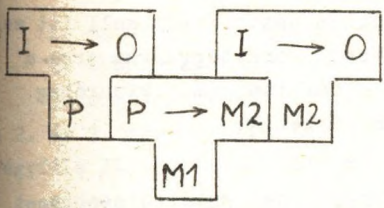
A továbbiakban bemutatni kivánjuk, hogy melyek azok a belső szerkezeti megoldások, amelyek a PASCAL forditóprogramot és ezen keresztül az egész ISIS rendszert olyan nagy mértékben függetlenitik a környezettől. Előbb azonban fel kell hivnunk a figyelmet arra, hogy a nyelvi rendszer portabilitása nem azonos jelentőségü az eljárásokéval. Ezt nagyon könnyen meg lehet világitani a T diagramok segitségével.



Jelentsen ez az ábra egy olyan véges állapotú gépet, amely egy meghatározott belső állapotból kiindulva az egy irányban mozgó input szalagról leolvasott jelsorozat alapján megfelelő output jelsorozatot állit elő. Mint tudjuk ez a gép analóg lehet egy programmal, amelyet L nyelven irtak. Lényegtelen most, hogy az input és output programszöveg-e vagy adat.

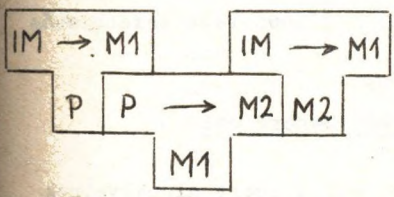


Ez az ábra most a P /PASCAL/ nyelven irt eljárások müködését ábrázolja. A P programot M gépi nyelvvé alakitó gépi kódú program teszi lehetővé, hogy az eljárás M gépen megvalósuljon.



Ez az ábra azt a helyzetet mutatja, amikor a P programot az M2 gép nyelvére forditjuk. Ugyanazon inputból azonos output készül. A portabilitás megoldott az M1 gépen generált megfelelő kód segitségével.

Vegyük most példának az INLAN programot, amely tudvalevőleg az input modul IM nyelvét forditja le az M1 gép nyelvére. Ha megváltoztatjuk az IM-et leiró PASCAL forditóprogramot, akkor a helyzet az alábbi lesz:



Olyan forditóprogramot kapunk a célkörnyezetben, amely még mindig a kiinduló környezetben szükséges tárgykódot generálja.

A PASCAL forditóprogram egyedi problémája átalakult tehát az n nyelv, m gép közismert problémájává. A megoldás klasszikus érvényü: közbeeső nyelv, amellyel az m.n forditóprogram szám m+n-re csökkenthető. Az ISIS-nek ezt a valójában névtelen közbeeső nyelvét IL-nek /Intermediate Language/ jelölhetjük.

Az IL megválasztása döntő hatást gyakorol a portabilitásra. Kétirányú követelménynek kell eleget tennie:

- géptől független nyelvnek kell lennie, amellyel az összes gépi kód tulajdonságai leirhatók, és
- gépközeli nyelvnek kell lennie, hogy az ujrakódolása minél kevesebb munkát jelentsen.

Lényegében egy géptől független asszemblert kell itt megvalósitani. A PASCAL és a többi forditóprogram erre a nyelvre fordit. A tárgykód előállitása már kizárólag ennek a nyelvnek az alapján történik.
Mit jelent ez?

- az IL-ről az adott gépi kódra forditó programot minden installációnál ujra kell programozni;
- az IL átprogramozásával az összes PASCAL-ban irt programrész átvihető az új rendszerbe.

Itt utalunk arra is, hogy a portabilitás általános megoldásának tekinthető "bootstrap" technika valósul meg az IL alkalmazásával. Az IL tulajdonképpen az az absztrakt gép, amely a valódi géphez legközelebb eső szinten megvalósul.

6. Az IL és a tárgykód generátor.

Az előirt terjedelem miatt csak a nyelv alapelveinek ismertetésére vállalkozhatunk.
Alapvetően arra készült, hogy a PASCAL követelményeit kielégitse. Mivel a felhasználói nyelvek is sokban a PASCAL-ra hasonlitanak, ez az általánositás még nem okoz gondot. Lehet, hogy más nyelveknél már több probléma lenne, ezt még nem próbáltuk ki. Mindenesetre ránézésre kevés korlátozást mutat.

Az algoritmusok leirása az input sorozatban az ún: "polish notation" alakban történik, mégpedig a "postfix" változat szerint, vagyis a müvelet kódja mindig követi az operandusokat. Az értelmezést egy hallatlanul egyszerü verem technikával lehet megoldani. Az utasitásokat számokkal jeleniti meg. Az IBM 360 gépi kódját előállitó IL "forditóprogram" irása két emberhónapnyi munkát kivánt.

Az IL utasitások felépitése:

1. az utasitás tipusát meghatározó kulcsszó /kód/
2. tipus megadás
3. cim vagy érték.

A 2. és 3. rész elmaradhat.

Négyféle IL utasitás létezik:

- cimzést, cimmódositást előidéző utasitások,
- a kifejezések és a függvények kiértékelésére szolgáló utasitások,
- vezérlő utasitások,
- a kezdő és végpontok jelzésére szolgáló utasitások.

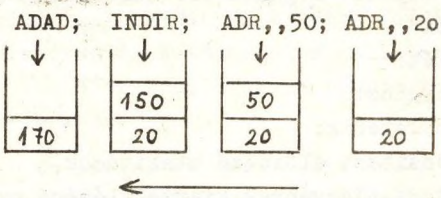
Jó példa az IL szerkezetére a cimzéssel kapcsolatos utasitások csoportja:

```
adrexp :: = ADR,, adr |
            adrexp; INDIR |
            adrexp; exp; MULI,,len |
            adrexp; adrexp; ADAD
```

/A kisbetüs szavak nem terminális, a nagybetüsök a terminális szimbólumok. A vesszők mezőket, a pontosvesszők utasitásokat választanak el./

Az ADR utasitás egyszerüen szolgáltat egy cimet; az INDIR az indirekt cimzést, a MULI az indexelt cimzést szolgálja, az ADAD pedig az egyszerü cimszámitást valósitja meg.

Pl:



legyen ez a bemenő jelsorozat és /5o/ = 15o, azaz az ötvenes cim tartalma 15o.

Az IL-ről gépi kódra fordító program hatékonyságán múlik az egész rendszer hatékonysága. Ez elég fontos kérdés, mivel igen nagy rendszerről van szó. A CRC tapasztalatai szerint nagyon jó hatékonyságú célkódot lehet generálni az IL alapján. Ők a FORTRAN-nal végeztek összevetést és kiderült, hogy kb. azonos méretek mellett a PASCAL-ból származó tárgyprogramok még gyorsabbak is valamivel. Ez azonban egy igen fejlett, gondosan optimalizált PASCAL változat.

7. Az ISIS IBM 360/OS környezetben

A KSH-ban az előbb felsorolt öt részből négy jelenleg müködik. Az adatbázis kezelő modul az egyetlen kivétel, amelyben még elég sok asszembler rutin van.

A fő probléma az átvitelnél az volt, hogy OS/MFT-ben dolgozunk és az ISIS nagyobbnak bizonyult, mint az üzemszerüen használatos partició méretek /110K byte/. A PASCAL fordítóprogram helyigénye 12o-13o K, az input modul nyelvé /INLAN/ 180K byte, az output modul nyelve még ennél is nagyobb tárterületet igényel. Érdekes, hogy problémát okozott a változók kezdőértéke is, mivel ez számos rendszerben bináris nulla, mig OS/MFT-ben meghatározatlan.

Meg kell állapitanunk azt is, hogy nem csak az IL fel-

dolgozót kellett ujraprogramozni, hanem a PASCAL-ról IL-re forditóban is meg kellett változtatni kb. ötven sort.

Véleményünk szerint néhány optimalizáló ciklus után jól használható PASCAL forditóprogramhoz jutunk. Ha az optimalizálást a célnyelvi programra vonatkoztatjuk, akkor az öndefiniálás miatt a forditóprogram is javul.

Végső következtetésünk az, hogy a portabilitás ezen az úton megvalósitható, nem vezet a rendszer elbonyolódásához és az a csekély hátrány, ami a csökkent hatékonyságból fakad, gondos munkával megszüntethető.

Felhasznált irodalom

1. Problems of portability, CRC Bratislava, May 1972.

2. ISIS Software Version 1.0, General Information Manual, CRC Bratislava, August 1974.

3. A. Mráz: ISIS Software Version 1.I, Installation Manual, CRC Bratislava, August 1974.

4. B. Šuster: PASCAL-B Version 3.0, User Manual, CRC Bratislava, August 1974.

5. P.J. Voda: The transfer of PASCAL to the IBM 360 System. A test of the portability of the ISIS software. Proc. on ISIS Working Seminar 1972, CRC Bratislava.

6. J. Gajdošik: Vnutorný Jazyk, /kézirat/ CRC Bratislava, é.n.

EGY IBM ADATBÁZISKEZELŐRENDSZER ÉS TÖBB SZÖVEGES INFORMÁCIÓTÁROLÓ ÉS VISSZAKERESŐ RENDSZER ADAPTÁLÁSI ÉS ALKALMAZÁSI KISÉRLETE

Balogh Zoltán
INFELOR Rendszertechnikai Vállalat

A szöveges információk számitógépes kezelése,feldolgozása területén több éves kutatómunka áll mögöttünk. Eddigi munkáinkat az állami finanszirozással megindult KSH számitógépalkalmazási kutatási-fejlesztési program keretében végeztük./ld.irodalom/

Az IBM Magyarországi Kft. segitségével nyolc programcsomagot rendeltünk,rendszersoftware szakembereink közremüködésével néhányat üzembehelyeztünk. A rendelkezésre álló dokumentáció feldolgozása mellett minta, majd élő adatokkal sikeres próbákat tettünk. Ezen programcsomagok fő alkalmazási területeként a közepes és nagyméretü szakkönyvtárak,dokumentációs központok jelölhetők meg. Szakirodalmi dokumentumok nyilvántartására,ad hoc és SDI /Selective Dissemination of Information/ tipusu visszakeresésére,bibliográfiák és katalógusok készitésére alkalmasak.

A fenti programcsomagok másik csoportja dokumentumok szedési,szerkesztési munkáinak automatizálásában adhat jelentős segitséget. A felmerült akadályok miatt ezek üzembehelyezését egyelőre el kellett halasztani.

Az Államigazgatási Számitógépes Szolgálat /ÁSZSZ/ létrehozásának előkészitő munkái során általános adatbáziskezelő rendszer tervezési és alkalmazási tapasztalatszerzésére volt szükség felhasználói és felhasználókat koordináló szinten. Az IBM Information Management System nevü rendszerének demonstrációs célu alkalmazási kisérletét hajtottuk végre. Ehhez kapcsolódik beszámolóm utolsó része.

A szöveges információkezelés témakörében alkalmazási kisérletre kijelölt programcsomagok:

1.Information Retrieval Management System /IRMS/

Deszkriptorokkal feltárt dokumentumok bibliográfiai leirásainak,esetleg kivonatainak tárolására és Boole-operátorokkal összeállitott kérdésekre való visszakeresésére alkalmas. Lemezes tárolással dolgozik,beépitett, csak kis részben aktivan használt tezaurusszal rendelkezik.

2.TEXT-PAC normál szöveges információ feldolgozó, visszakereső és kurrens információválogató rendszer

A rendszer nagyméretü elsősorban mágnesszalagos dokumentumtárak kialakitására,kurrens és retrospektiv visszakeresésére alkalmas a tárolt szöveges információk minden elemére kiterjedően. Ötféle index és bibliográfia készitése is lehetséges ideértve a cimek KWOC indexét is.

3.Document Processing System /DPS/

Lemezorientált nagymértékben integrált dokumentum-

visszakereső rendszer. Input formatum meghatározását a felhasználónak kell definiálni meglehetősen tág határok között. A visszakeresésnél a többszintü Boole-kifejezésekben megadott szövegszavak szinonima és ekvivalencia szólistákkal automatikusan bővithetők. A lekérdezés a tárolt dokumentumleiró szövegek szavain kivül formatizált mezők tartalmára is vonatkozhat.

4.HYPHENATION/360

Szövegszerkesztő rendszerekben a sorvégi szóelválasztást szótaghatáron végzi a kivételes esetek figyelembevételével.

5.COMPOSITION/360

Szövegek szedése és szerkesztése az előirt formai követelményeknek megfelelően.

6.TEXT/360

Szövegfeldolgozó,szerkesztő és kamerakész nyomtató rendszer.

7.KWIC/360

KWIC és KWOC tipusu indexek,bibliográfiák és katalógusok készitésére alkalmas programcsomag.Ujabb változata információvisszakeresésre is használható.

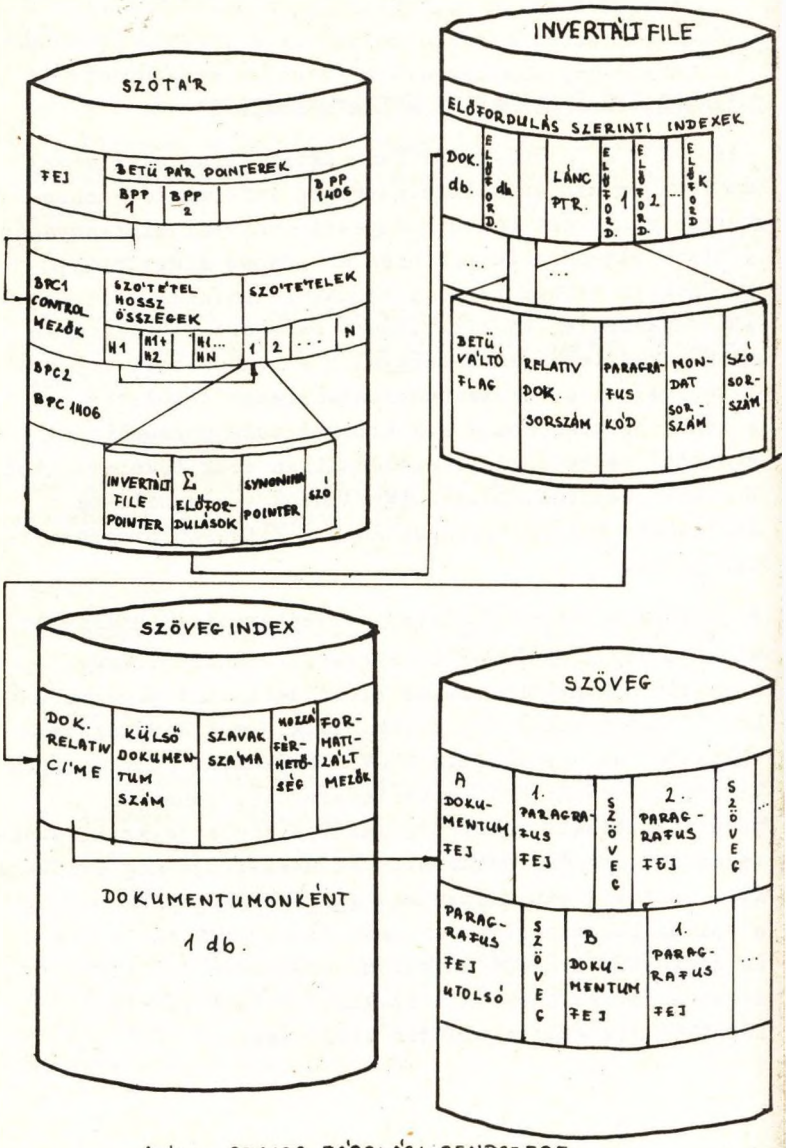
8.Generalized OS/360 I/O szubrutinok

Számos általánosan használt I/O müvelet elvégzését teszi lehetővé hatékony módon PL/1 és BAL nyelvü programoknál. Hibaelemző és dump lehetőségek biztositásával segiti a programozókat.

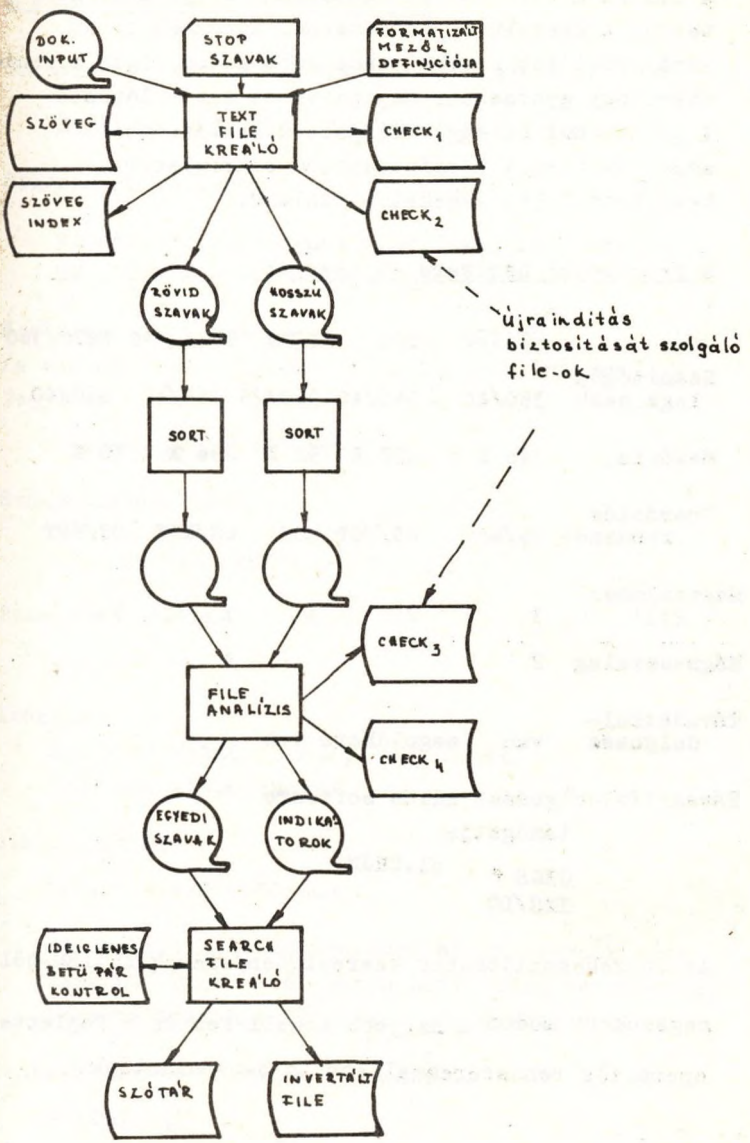
Storage And Information Retrieval System

Az I B M a Budapesti Nemzetközi Vásár alkalmával mutatta be ezt az online szöveges információ visszakereső rendszert három budapesti könyvtár mintaanyagán. A mintafile-ok előkészitésén és számos kisérleti kérdezésen vettem részt. A rendszer installálását az IBM szakemberei végezték. Ennek ellenére előadásomban néhány részletkérdést tárgyalok a rendszerrel kapcsolatban, mivel a dokumentumvisszakeresés területén a többi programcsomag sok értékes tulajdonságát egyesiti, technológiai megoldásaiban azok tökéletesitett formáira ismerhetünk. A nagyvolumenü szalagfile-ok lekérdezésénél jól kiegészülhet a TEXT-PAC programcsomaggal.

Az 1.ábra a STAIRS rendszer szövegtárolási rendszerét mutatja be. A SZÖVEG-FILE szerkezete mutatja, hogy a tárolt dokumentum számos előre definiált paragrafusból állhat például a szerzők, cim, kivonat, tárgyszavak választhatók igy külön. A SZÖVEGINDEX-FILE a dokumentum leirásából rendezést, továbbiszelekciót lehetővé tevő formatizált mezők tárolását biztositja. Az illetéktelen hozzáférést dokumentum szinten oldja meg az összes file biztositásán tulmenően. Az INVERTÁLT-FILE a szavak pozicionális információi mellett még a kis és nagybetüs irásmód megkülönböztetésére is lehetőséget ad. A SZOTÁR-FILE a szavak elérését az első két betü vizsgálata alapján oldja meg.



1. ábra STAIRS TÁROLÁSI RENDSZERE



2. ábra STAIRS KERESŐ FILE ÉPITÉSE

A 2.ábra a file-ok létrehozásának folyamatát mutatja. A kiemelt 15 karakternél hosszabb és az annál rövidebb szavak rendezését külön végzi a rendszer,hogy gyorsabban hajtsa végre ezt a lépést. A géphibából és egyéb kényszerü leállásokból adódó esetleges ujrafuttatások elkerülésére beépitett lehetőségeket tartalmaz.

A rendszerek hardware környezete

	STAIRS	DPS	IRMS	TEXT-PAC	KWIC/360
Számitógép legkisebb	360/40	360/40	360/25	360/40	360/40
Memória	24o K	128 K	32 K	256 K	70 K
Operációs rendszer	OS/MFT	OS/PCP	DOS	OS/MFT	OS/MFT
Mágneslemez 2311	3	2	3	1	1
Mágnesszalag	2			4	
Távadatfeldolgozás	van	megoldható		van	

Távadatfeldolgozást külön software támogatja

CICS pl.CRJE
IMS/DC

Az összehasonlitásban szereplő rendszerek az IBM-től megszokott módon a nagyobb modelleken és a fejlettebb operációs rendszerek mellett is használhatóak.

A rendszerek output lehetőségei

Információvisszakeresés adhoc kérdésekre:

STAIRS, IRMS, DPS, TEXT-PAC, KWIC/36o

SDI/szelektiv információ szétosztás/

mindegyik rendszernél megoldható, legjobban a TEXT-PAC esetében van kiépitve.

Konkordancia szójegyzék:
/a szavakat pozicióikkal együtt leiró információk jegyzéke/

STAIRS, DPS, TEXT-PAC

Szógyakoriségi listák:

STAIRS, DPS, TEXT-PAC, KWIC/36o

Tezaurusz listák:

IRMS

Indexek:
KWIC/36o /KWIC index/,/KWOC index/
TEXT-PAC /KWOC index/

Bibliográfiák:

DPS,TEXT-PAC,KWIC/36o

/tárgyszavak,dokumentumazonositók,szerzők neve szerint/ Erre a feladatra az INDICAT rendszer a legalkalmasabb,a DPS csak igen kevéssé.

Statisztikák a rendszer filejairól:

STAIRS,IRMS

KÉRDEZÉSI LEHETŐSÉGEK ÖSSZEHASONLÍTÁSA

Rendszerek:	S.I.D.T.K.
Kérdés eleme deszkriptor:	+ + + + +
Kérdés eleme a szöveg bármely szava:	+ + +
Szövegszó jobbról csonkítva:	+ + +
pl: PROGRAM§ megfelel a program szónak illetve bármely képzett alakjának	
Szövegszónál hosszabb és/vagy rövidebb:	+
pl: PROGRAM/+3/ megfelel a program szó maximum 3 karakteres ragozott stb. formáinak	
A kérdés elemének bővítése szinonim szavakkal automatikusan:	+ +
A kérdés elemének automatikus bővítése ekvivalensnek tekintett szavakkal: /a tezaurusz burkolt használata lehet/	+
A szövegösszefüggést figyelembevevő operátorok:	+ + +
pl: X ADJ Y x és y szavak szomszédok	
X WITH Y egy mondaton belül	
X PAR Y egy paragrafusban	
Boole operátorok: AND,OR,NOT,XOR	+ + + + +
Sulyozási logika /relevancia küszöb/:	+ +

	S.I.D.T.K.
Maszkolásos keresés:	+ +
pl: 1□□6 karaktersorozat 2.és3. pozíciója bármi lehet	
Scan jellegü keresés:	+
pl: SC 'AB' az AB karakterpár bármilyen pozición előfordulhat	
Formatizált mező keresése:	+ + +
Keresési tartomány behatárolása:	+ + + +
A keresés kényelmét szolgáló további lehetőségek	
Listázás online/offline választással:	+ +
Listázásban szereplő mezők választása:	+ + +
Kis és nagybetü megkülönböztetése:	+ + + +
Szótári információ kérdezhetősége: /tezaurusz passziv használata/	+ + +
Kérdés tárolása,visszahivatkozás rá:	+ + +
Rendezett output: /formatizált mezőre/	+
Utószerkesztés:	+
Output relevancia sorrendben:	+
Dokumentum elérési száma gépi adathordozón mikrofilm kereséshez:	+ +
Rendszerhasználat elszámolási adatai sornyomtatón:	+

A programcsomaginstallálási munka legfontosabb lépései:

1. Kiválasztás,megvalósithatóság vizsgálata:
   - HW,SW korlátok;
   - A rendszer funkciói teljesitik-e a követelményeket;
   - A tervezett rendszer információstrukturája behelyettesithető legyen a programcsomag információstrukturájába;
   - Installálás idő,anyagi-,személyi,gépidő lehetőségei.
2. Beszerzés /program,teljes dokumentáció/
3. Compilálás /katalógizálás,file-ok létesitése/
4. Tesztelés
5. Kiegészités,honositás
6. Szervezési környezet,adatok biztositása
7. Üzemeltetés és karbantartás

A DPS,TEXT-PAC,KWIC/360 rendszerek néhány fontosabb installálási tapasztalata

A DPS rendszer az OS/360 egy valamivel régebbi verziójára iródott,igy a megadott minta JOB sorozat csak kis lépésekben, job-lépésenként volt futtatható.

A KWIC/360 programcsomagban egy-két apróbb hiba sokáig meggátolta az összes opció kipróbálását, néhánynapos programfejtés és nyomozás után viszont sikerült kijavitani.

A TEXT-PAC rendszer mintegy 33 jobját meglehetősen nehéz kezelni. Nagy memóriaigénye és a rendszer hibák elleni kevéssé védett volta miatt komoly nehézségeket okoz.

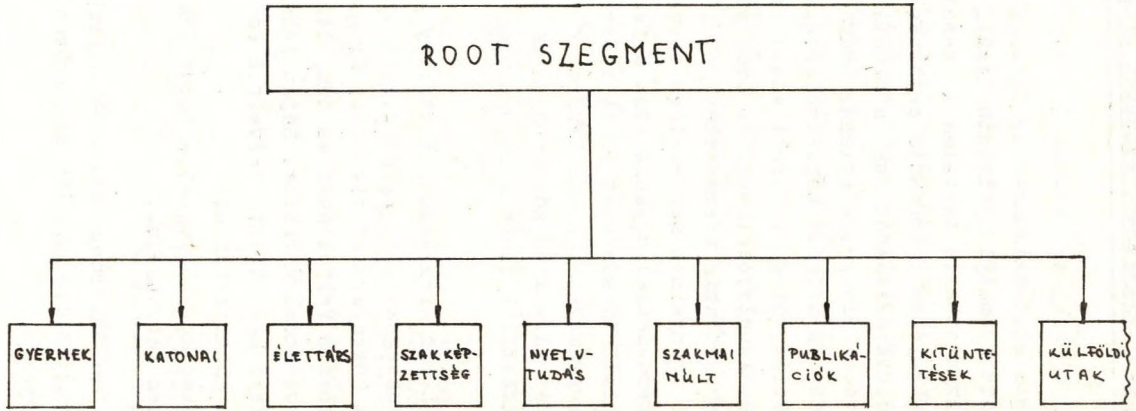
Az IMS adatbáziskezelő rendszer alkalmazási kísérlete

Kisérleti adatbázisunkat az INFELOR dolgozoinak sokoldalu személyi nyilvántartására kivántuk kiépiteni. Az adatbázis tartalma a személyhez tapadó összes lényeges információ együttese.A dolgozók hozzátartozóinak adataiból csak a szociális körülmények ismeretéhez elengedhetetlenül szükséges adatokat vettük tekintetbe.A dolgozó képzettsége,szakmai multja, publikációi,tömegszervezeti adatai mellett természetesen a munkahely,bér,beosztás szokásos adatai szerepeltek. Az adatbázis szerkezetét /3. ábra/ a lekérdezés várható igényeinek megfelelően terveztük.Egyszerü IQF lekérdezéssel ugyanis csak a faszerkezetben egy uton található adatokat lehet lekérdezni. Saját lekérdező programunk és a GIS lekérdező rendszer/ld. irodalom/ már bármilyen adatkombináció lekérdezésére képes, ezek elkészülte azonban időben csak később volt várható.

Az adatbázis kreálásához 3 utility program megirása volt szükséges./file kreálás,adatbázis definició, program hozzáférését biztositó-Program Control Block/ Az adatbázis feltöltését és dumpolását 2 darab PL/1 programmal végeztük. Saját lekérdező programunkat interaktiv használatra terveztük és szintén PL/1 nyelven készitettük el.
A rendszer kreáló részét a bécsi IBM 370/145 -ös gépén sikeresen lefuttattuk.

A rendszer lekérdező részének kipróbálására egyelőre nem került sor,ez az IMS Magyarországi installálásának függvénye.

3. ábra IS ADATBÁZIS SZERKEZETE



Irodalom:

A programcsomagok dokumentációi

1. IBM System/360 Information Retrieval and Management System/ IRMS/ Application description manual, 1970,/GH-19-0012-1/ - magyarul is
2. TEXT-PAC, S/360 Normal Text Information Processing,Retrieval and Current Information Selection System, 1968,/360D-07.7.020/- magyarul is
3. IBM System/360 Document Processing System /DPS/ Application desription, 1967,/GH-20-0315-0/
4. IBM SYstem/360 and System/370 /OS/ Storage and Information Retrieval System /STAIRS/ General Information Manual/GH-12-5107-2/ 1973.
5. KWIC/360 Program description
6. Information Management System/360, Version 2. General Information Manual,1972,/GH-20-0765-3/
7. Interactive Query Facility /IQF/ for IMS/360, Version 2. General Information Manual,1974, /GH-20-1074-2/
8. Generalized Information System/Virtual Storage /GIS/VS/ General Information Manual,1974, /GH-20-9035-1/
9. Siemens GOLEM/2 Ein System zur Wiedergewinnung von Information - Verfahrenbeschreibung
10. CII. MITRAL

Az INFELOR munkatársainak publikációi

11. Számitógép alkalmazása a jogalkotásban és a jogalkalmazásban - kutatási jelentés I-II.köt. INF 1078/72

12. Balogh Zoltán
Szöveges információvisszakeresés
= Információ Elektronika 1974.1.sz.

13. Szövegfeldolgozó programcsomagok installálása
- belső tanulmány INF 1724/1974

14. Az IBM TEXT-PAC nevü szöveges információfeldolgozási programcsomagjának alkalmazási leirása
- belső tanulmány INF 1473/1975

15. Az IBM IMS adatbáziskezelő rendszerének alkalmazása - belső tanulmány 1975.

16. Szöveges információkezelés Bp.1975.
/ INFELOR Közlemények 11./

17. Balogh Zoltán
Könyvtárgépesités I-II.
= Tudományos és Müszaki Tájékoztatás
- megjelenés alatt

SÁMÁN ADATBÁZISKEZELŐ RENDSZER ALKALMAZÁSA SZÜKSÉGLET-SZÁMITÁSRA

Erényi Vilmos
NIK Ipargazdasági és Üzemszervezési Intézet

BEVEZETÉS

A SZÁMOK, MÁV és a NIK IGÜSZI a SÁMÁN létrehozásakor megállapodott abban, hogy külön-külön különböző tipusu alkalmazásokkal vizsgálják az adatbáziskezelő rendszert. Az ismertetésre kerülő alkalmazás egy lehetséges megoldás a sok közül, melynek célja: adatbázisra épülő komplex termelésirányitási rendszer kialakitása.

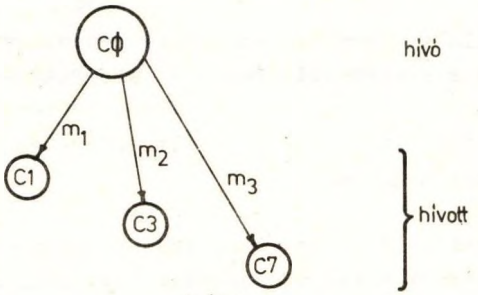
Egyetlen vállalat termelésirányitási rendszere sem nélkülözheti a szükségletszámitást, amelynek elsődleges feladata:
a gyártani kivánt végtermék /ek./ alkatrész- és anyagszükségletének meghatározása.

A szükségletszámitás nettó vagy bruttó, aszerint, hogy figyelembe veszik a raktárkészletet vagy nem. A gyakorlatban használatos elnevezések még a szükségletszámitásra: családfa- illetve darabjegyzéklebontás.

1. SZÜKSÉGLETSZÁMITÁSI MODELL

Egy termék előállitásához szükséges anyagok, alkatrészek és félkész termékek, valamint beépülési mennyiségeik felsorolását darabjegyzéknek nevezzük. A darabjegyzék modellje: irányitott fa-tipusu gráf, ahol a csomópontok az anyagokat, alkatrészeket, termékeket, az irányitott élek a közöttük lévő tartalmazási relációkat, az élek multiplicitása a beépülési mennyiséget jelenti. Az előállitandó terméket szokás hivónak, az ehhez szükségeseket pedig hivotttaknak nevezni. Az irányitott élek a hivóból indulnak ki.
A csomópontból kiinduló élek száma - nagyobb

mint kettő - a hivóhoz közvetlenül tartozó hivottak számát mutatja.
Az anyagok, alkatrészek, termékek azonositóját szokás cikkszámnak, rajzkódszámnak /stb/ nevezni. A továbbiakban azonos értelemben használjuk a hivó,illetve hivott csomópont vagy cikkszám fogalmat, nem téve különbséget, hogy anyagról, alkatrészről vagy termékről van szó.



1. ábra

CØ cikkszám darabjegyzéke: irányitott fa

Ilyen darabjegyzékekből épül fel egy vállalat gyártmánycsaládfája, amely több szintből állhat pl.: gépsor, gép, szerelvény, szerelék, alkatrész, kereskedelmi áru, anyag vagy csak egyszerüen késztermék, félkész termék, anyag.
Ezen szintek szerint szokás a cikkszámot az u.n. szintszámmal kibőviteni, már csak azért is, hogy a cikkszám gyártmánycsaládfában elfoglalt helyét is mutassa. A szintszám azonkivül, hogy reprezentálja

a gyártmánycsaládfa hierarchiáját, rendkivüli módon megkönnyiti annak kezelését, karbantartását is végezzük ezeket manuálisan vagy számitógéppel. A szintszám létrehozása egyébként nem jelent különösebb nehézséget, de itt erre nem térünk ki.
A gyártmánycsaládfa tulajdonságai hierarchikus, hálótipusu gráffal irhatók le. A gráf hierarchikus, mert minden részgráfja irányitott fa, igy nem tartalmazhat körutat. A gráf hálótipusu, mert egy csomópontba több mint egy él futhat. Ezen befutó élek száma - melyet nevezzünk egyszerüen nyilszámnak - mutatja, hogy hány csomóponthoz tartozik az illető csomópont. Megforditva az élek irányitását azt is láthatjuk, mely csomópontokba. Igaz továbbá, hogy végtermék - csak hivó csomópont - nyilszáma zérus.

CSOMÓPONTOK TIPUSAI	HIVOTTAK SZÁMA	NYILSZÁM
CSAK HIVO	≥ 2	$= 0$
HIVO ÉS HIVOTT	≥ 2	≥ 1
CSAK HIVOTT	$=0$	≥ 1

2.ábra

Összefüggés a csomópont tipusa, valamint az élek tipusa és száma között.

A szükségletszámitási modell szerkezetileg három - egymással szorosan összefüggő - részre tagolható

INFORMÁCIÓSTRUKTURA a gyártmánycsaládfa tulajdonságait leiró hierarchikus hálótipusu gráf.

ADATSTRUKTURA, amely egyrészt az anyagok, alkatrészek és termékek szükségletszámitás szempontjából lényeges jellemzőit, másrészt a darabjegyzékeket tartalmazza.

TÁROLÁSI STRUKTURA közvetlen elérésü tárolón kijelölt, alkalmasan választott file-ok együttese.

A három struktura együtt alkotja a müködő szükségletszámitási modellt, amelyre a továbbiakban DARABJEGYZÉKSTRUKTURA névvel hivatkozunk.

2. DARABJEGYZÉKSTRUKTURA FUNKCIÓI

1. Bruttó szükséglet számitása
2. Nettó szükséglet számitása
3. Közvetlen beépülés
4. Közvetett beépülés
5. Karbantartás /csomópontok és élek törlése, beszurása, módositása/
6. Listák, illetve DUMP-ok készitése darabjegyzékenként

3. DARABJEGYZÉKSTRUKTURA létrehozása SÁMÁN segitségével.

Két file-t generáltunk a DTORZS és DRBJ nevüt.

1. DTORZS

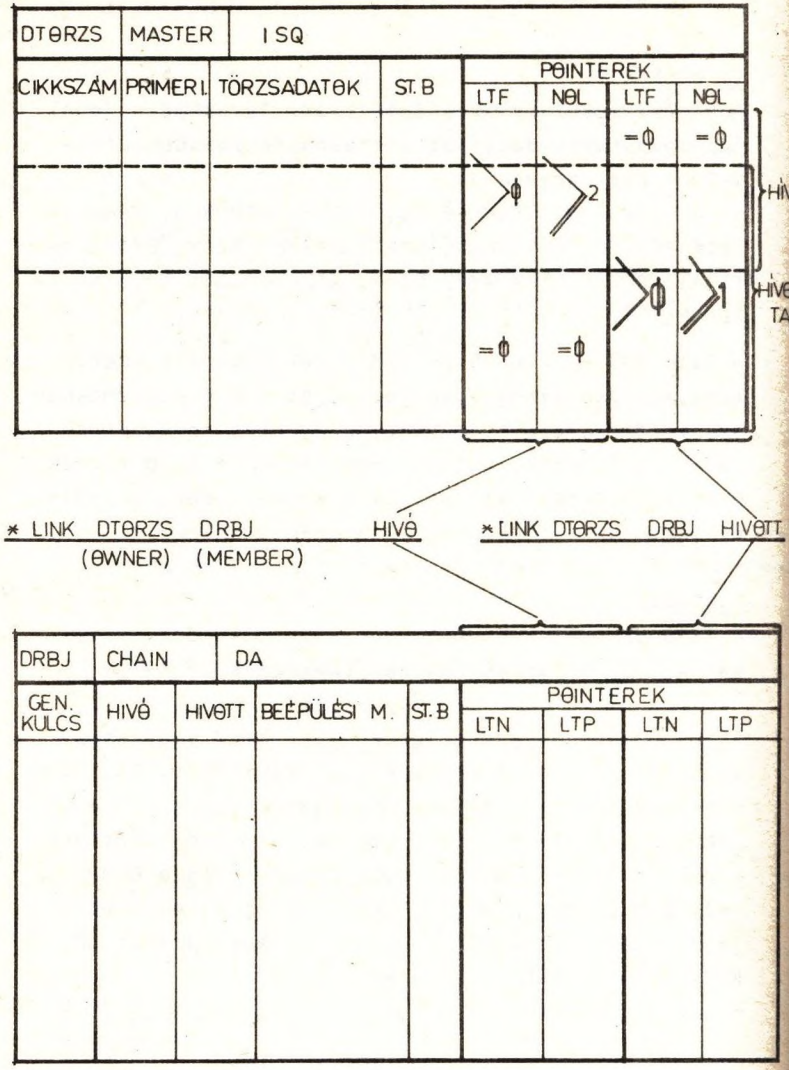
A cikkszámokat és szükségletszámitás szempontjából lényeges törzsadataikat tartalmazza /minden cikkszámot csak egyszer/.
Pl.: CIKKSZÁM, MEGNEVEZÉS, MÉRET, MINŐSÉG, MENNYISÉGI EGYSÉG, EGYSÉGÁR, STATISZTIKAI SZÁM, STATISZTIKAI SZORZÓ, FŐKÖNYVI SZÁM, GAZDÁLKODÓI KÓD, KÖLTSÉGHELY

A file MASTER tipusu és indexszekvenciális szervezésü. Ez utóbbiból következik, hogy a cikkszámoknak szintszámra szigoruan monoton növekvő rendezettségünek kell lenni. Azaz a legmagasabb szintü termék szintszám értéke kisebb, mint bármely ebbe beépülő cikkszámé.

2. DRBJ

A file az összes hivó hivott kapcsolatpárt tartalmazza, CHAIN tipusu, direkt szervezésü.

A "HIVO" egyedi kapcsolatnév /3. ábra/ segitségével hoztuk létre a DARABJEGYZÉKSTRUKTURÁT. Pl. csak a termékeknek és félkésztermékeknek van darabjegyzékük ezért LTF mutatja, hol van az első hivottjuk a DRBJ file-ban, a következő hivottjukra a DRBJ-beli LTN mutat. A DTORZS-beli NOL érték a hivottjaik száma. A "HIVOTT" kapcsolat magyarázatát ld. KÖZVETLEN BEÉPÜLÉS c. fejezetben.



3. ábra

DTORZS és DRBJ kapcsolatai.

4. BRUTTÓ SZÜKSÉGLETSZÁMITÁS ALGORITMUSA

Mindaddig nem határozható meg egy cikkszám szükséglete, amig ennél magasabb szintü cikkszámok szükséglete nem meghatározott.

1. Legyen a DTORZS file-ban a PRIMER IGÉNY valamennyi CIKKSZÁM-ra zérus, illetve a rendelt mennyiség / >o/. Az igy meghatározott primer igényt nevezzük aktuális primer igénynek.

2. Vegyük a DTORZS következő /első/ cikkszámát,

/2.1/ ha NOL értéke zérus, vége az algoritmusnak,

/2.2/ ha nem és a PRIMER IGÉNY zérus, visszatérünk 2-re,

/2.3/ ha NOL és a PRIMER IGÉNY nem zérus, 3 következik.

3. Az LTF segitségével emeljük ki DRBJ-ből az első hivottat.

4. Beépülési mennyiségét szorozzuk meg a hivó cikkszám aktuális primer igényével és a hivottat cikkszáma - DTORZS elsődleges kulcsa - segitségével emeljük ki DTORZS-ból és aktuális primer igényét növeljük az előbb számitott szükséglettel!

5. A 4-ben szereplő, hivott LTN értékének vizsgálata:

/5.1/ LTN >o kiemeljük a következő hivottat és 4. következik

LTN = o LTO - a hivó - szerint visszatérünk DTORZS-be és a 2 következik.

5. NETTÓ SZÜKSÉGLET

1. A primer igény legyen a cikkszámonkénti aktuális raktárkészlet negativ előjellel, ha van raktáron és zérus, ha nincs.

2. Azoknak a cikkszámoknak a primer igényét, amelyekből rendeltek növeljük a rendelt mennyiséggel. Igy negativ, illetve zérus a primer igény, ha nem kell gyártani, mert van raktáron elegendő, és pozitiv a primer igény, ha kell gyártani. Ezen előkészités után a BRUTTÓ szükséglet meghatározása következik a 2 lépéstől.

-Mindkét algoritmus befejeződésekor azonnal nyomtatásra, illetve további számitásokra készen áll a DTORZS file, hiszen minden cikkszám szükséglete és törzsadata egy rekordban van.

-Továbbá lehetőség van a SÁMÁN-ban arra is, hogy a példában emlitett DTORZS-rekord /ld. 3. fejezet/ FŐKÖNYVI SZÁM, GAZDÁLKODÓI KÓD, KÖLTSÉGHELY adatatait másodlagos kulcsoknak tekintsük az adatbázis generálásakor és ezen szempontok szerint azonnal lehet a táblákat is nyomtatni.

-Mindkét algoritmus számára közömbös, hogy hány és milyen létező cikkszámból rendeltek, tehát lehetséges akár egyetlen termék szükségletét meghatrozni.

-Lehetőség van továbbá karbantartási müvelet nélkül olyan részgráfok ideiglenes törlésére, melyek hivója egyben hivott is. Legyen az emlitett hivó

primer igénye az a negativ szám, amelynek abszolut értéke megegyezik az emlitett hivó aktuális primer igényével, amikor bruttó vagy nettó szükségletének meghatározására éppen sor kerül. T.i. ekkor aktuális primer igénye zérussá válik.

-Lehet tervezésre is használni az algoritmust. Pl.: szeretnénk létrehozni a DTORZS-ben eddig nem szereplő terméket, ekkor olyan primer igényeket irunk a beépiteni kivánt cikkszámokhoz, amilyen beépülési mennyiségeket tervezünk és fentiek szerint előkalkulációt is végezhetünk.

6. KÖZVETLEN BEÉPÜLÉS

Feladat: adott cikkszám melyik darabjegyzékben és mekkora beépülési mennyiséggel szerepel.

Megoldás: még egy kapcsolatot definiálunk a DTORZS és DRBJ file-ok között /3. ábra/.

A HIVOTT kapcsolatban ismét a DTORZS az OWNER és DRBJ a MEMBER.

A pointerek jelentése:

DTORZS
- LTF: hol fordul elő a hivott, hivottként a DRBJ-ben
- NOL: hányszor

DRBJ
- LTN: hol a következő hivott
- LTP: hol az előző hivott

Ezek ismeretében nem probléma a feladat megoldása.

7. KÖZVETETT BEÉPÜLÉS

Feladat: mely cikkszámokat érintené közvetlenül és közvetve adott cikkszám megszünése.

Megoldás: a közvetlen beépülés többszöri alkalmazása.

Ezzel a DARABJEGYZÉK STRUKTURA négy funkcióját megismertettük.

A karbantartás ismertetése meghaladná a cikk terjedelmét, a LISTA és DUMP- készitès - darabjegyzékenként illetve termékenként - ismertetése pedig lényegében nem mondana ujabbat, ezért mindkettőtől eltekintünk.

8. ÖSSZEFOGLALÁS:

Befejezésként fel kell hivnunk a figyelmet ismételten arra, hogy a SÁMÁN host-language tipusu adatbáziskezelő rendszer ezért ASSEMBLER, COBOL, PL/1 vagy FORTRAN nyelven is irhatjuk az előzőleg ismertetett 4 funkciót. A SÁMÁN-nak ez a tulajdonsága igen előnyös, hiszen minden felhasználó saját igénye szerint alakithatja ki szükségletszámitási rendszerét. Pl.: helyettesitő termékek automatikus felhasználása.
Megnyugtatásként csak annyit, hogy a szükségletszámitási algoritmus összesen 6 DBAM és 5 FORTRAN utasitás volt!!!
Az ismertetett szükségletszámitási modell általánosithatósága:

1. a megoldás felhasználható minden olyan feladat megoldására, melynek modellje hierarchikus, hálótipusa gráf. Továbbá igaz, hogy a gráf nem változik sokszor és lényegesen, azaz a modell használata gyakoribb, mint változtatása.
Pl.: a termelésirányitás további területei; személyzeti nyilvántartások; mezőgazdaságban pedig az állattenyésztési genetikai modellek.

2. a megoldás példaként szolgálhat termelésirányitási adatbázis kiépitésének kezdeti lépéseire.

A TOPOGRÁFIAI ADATBÁZISOK KEZELÉSÉNEK PROBLÉMÁI

Szörényi Miklós
Közlekedési és Távközlési Müszaki Főiskola

A közelmultban kezdődött el, és jelenleg is folyik az első, Magyarország egészére kiterjedő topográfiai adatbázis létrehozása. Ezzel kapcsolatban igen sok részfeladat merült fel, melyek megoldásába több intézmény is bekapcsolódott. Az adatbázis fejlesztésén tulmenően igen sürgetővé vált a felhasználási lehetőségek felkutatása, és az adatbázis felhasználhatóságának elvégzett munkákon keresztüli bizonyitása.
Az első két ilyen jellegü feladat a Széchenyi-hegyen lévő TV adótornyokból való geometriai átláthatóság vizsgálata, valamint az Interszputnyik Nemzetközi Ürtávközlési Rendszer és Szervezet magyarországi földi állomása interferenciális zavarokkal szembeni védelmi körzetének meghatározása volt. Ezen két feladat számitógépen történő megoldása a programtervezési és programozási munkáinak elvégzésével együtt reám hárult.
Előadásomban az eddigi tapasztalatokról számolok be, és ezalapján megpróbálom összefoglalni azokat a problémákat, melyek a topográfiai adatbázisok kezelésével kapcsolatosak lehetnek. Nem törekedek teljességre, mert ez egyrészt meghaladja az előadás kereteit, másrészt pedig a konkrét tapasztalatok a meglévő adatbázis jellemző tulajdonságaihoz kapcsolódnak, igy a más felépitésü adatbázisok kezelésének problémáiról csak az irodalmi hivatkozások szintjén tudnék szólni.

Először a rendelkezésre álló adathalmaz legfőbb jellemzőit mutatom be.

Az adott területet egységes rendszer szerint készitett un. mintaelemek sokaságára osztották fel. Egy mintaelemről a figyelembe veendő topográfiai adatokat kigyüjtötték és lyukszalagra rögzitették. Több egymás melletti mintaelemet egy blokkba foglaltak össze, és a pozicionáló adatokat csak a blokkhoz rendelték. Egy mintaelem topográfiai helyét a blokk pozicionáló adataiból és a mintaelem blokkon belüli sorrendiségéből lehet megállapitani. Az adathalmaz méreteiről jellemzésként csak annyit, hogy a mintaelemek száma meghaladja a két milliót.
Mint a bevezetőben emlitettem, az adatbázis jelenleg is fejlesztés alatt áll, ezért csak a lyukszalagon rögzitett alapadatokat használhattuk a kitüzött feladatok megoldásakor, és igy magunknak kellett megteremteni a kezelő rendszer felhasználásra kerülő elemeit. Az első feladat a rendelkezésre álló adathalmaz hibamentessé tétele volt, mivel a lyukszalagra való rögzités elég nagy hibaszázalékkal történt. Ezt a munkát, mely jellegét tekintve rutinfeladat, nem részletezem. A feladatot már ebben a fázisában is, de a későbbiekben még inkább nehezitette az a körülmény, hogy a rendelkezésre álló számitógép igen szük konfigurációju volt: csak 4 db 64K-szó kapacitásu fix dobtárolóval rendelkezett háttértárolóként. Igy a kijavitott adathalmazra is csak lyukszalagos formájában támaszkodhattunk, és minden adatátrendezési tranzakciót csak ebből kiindulva lehetett végrehajtani.

A következőkben a feladatok megoldása érdekében létrehozott adatbázis kezelő elemeket ismertetem.

ÖSSZEFÜGGŐ TERÜLETEK ELŐÁLLITÁSA

A bevezetőben emlitett feladatokhoz legelőször azokat az

adatokat kellett összeválogatni, melyek a kijelölt területre vonatkoztak. A feldolgozáshoz a szükséges adatok mátrix adatszerkezet szerinti tárolását tartottuk célravezetőnek. Ehhez elkészitettük a blokkok pozicionáló adatai és a háttértároló megfelelő fizikai cimei közötti leképező eljárásokat, melyek segitségével a szükséges adatokat blokkonkénti beolvasás után a megfelelő cimü rekeszekbe tudtuk elhelyezni.

A következő lépés a mátrixba ugyan belekerült, de a feldolgozáshoz nem szükséges adatok - pl. országhatáron tuli területek stb. - megjelölése volt. Ezt a háttértárolón lévő szavak megfelelő, és a lényeges információk által fel nem használt, bitjének 1-re állitásával értük el. Például a magasságadatok egészre kerekitett tárolásánál egy adathoz elegendő 10 bit, ugyanis a Kékes magassága kisebb 1023-nál, igy egy 24 bites szóban éppen elég "felesleges" bit marad.

A következő probléma az volt, hogy mindkét munkánál a magasságjellegü adatokat nem lehetett közvetlenül felhasználni, mivel a hullámterjedési és a geometriai átláthatósági számitásoknál a Föld görbültségét nagyobb távolságok esetén figyelembe kellett venni. A magasságok módositásának megállapitása a centrumbeli érintősikhoz viszonyitott "lehajlás" kiszámitásának felhasználásával történt.

TALAJPROFIL REKONSTRUKCIÓK

A feladataink megoldásához elengedhetetlen volt a megfelelő talajprofil rekonstrukciós eljárások kidolgozása, melyek a két végpontot összekötő egyenes, nagyobb távolságok esetén a megfelelő gömbi főkör menti meghatározott módon elhelyezett talppontokra jellemző topográfiai adatokat szolgáltatja.

Nemlineáris interpoláció esetén egy topográfiai adatot a kérdéses pontot körülvevő mintaelemek adataiból egy z=f/x,y/ illeszkedő felület meghatározása után számitottuk ki.
Lineáris interpoláció esetén egy vizsgált pont adatait a két szomszédos mező adataiból határoztuk meg. Ezeket a szomszédos mezőket vehetjük a kérdéses pont sorából, oszlopából, vagy az átlósan szomszédosak közül.
Adateltolásos módszer esetén a kérdéses egyenes közelében fekvő - a két végpontot összekötő egyenes által "surolt" mintaelemek középpontjára vonatkoztatott - tereppontokat betöltuk az egyenesre.
Mindegyik módszernek közös vonása, hogy a mintaelemek adatait a középpontjukra vonatkoztatva értelmezi. Az első módszert csak akkor használtuk, ha egyetlen talajprofilt kellett valamilyen célból rekonstruálni, a második és a harmadik módszert összefüggő, nagyobb területre való feldolgozáskor alkalmaztuk. Egy feldolgozásnál mindig figyelembe kell venni a tereppontok darabszámát, egy adat elérési idejét, a feldolgozás pontosságigényét, a feldolgozás számitási lépéseinek darabszámát, és ezeknek megfelelően kell választanunk az egyes talajprofil rekonstrukciós eljárások közül.
Speciális talajprofil rekonstrukció a talajprofil konvex burkának meghatározása. Konvex burokra a hullámterjedési számitásoknál volt szükség. Az alkalmazott eljárás az előző talajprofil rekonstrukciós eljárásokkal kombinálva egy verem adatszerkezetben állitja elő a konvex burok szögpontjait beleértve a kezdő és a végpontot is. Az eljárás lényegét a következő algoritmus tükrözi:
Az algoritmus leirásánál használt veremkezelő kivesz és berak müveletek értelmezése közismert.

```
ciklus aktuális:= a következő tereppont magassága;
    ha a veremben már van legalább két elem akkor
       kezd kivesz/utolsó/; kivesz/előző/;
            ciklus utolsó:=előző; kivesz/előző/;
            amig az aktuális az előző és utolsó egyenese
                 fölé nem kerül, vagy a verem üres;
            berak/előző/;
       vége;
    berak/aktuális/;
amig a végpontot el nem hagyjuk;
```

ÖSSZEFÜGGŐ TERÜLET VÉGIGJÁRÁSA

Mindkét felhasználói munkánknál egy összefüggő területet kellett végigjárni, és minden egyes mintaelemre a kijelölt centrum és a kérdéses mintaelem közötti talajprofil segitségével állapitottuk meg a kivánt jellemzőket. A végigjárásra két módszert alkalmaztunk.

A szervezés szempontjából egyszerübb az adatmátrix sorfolytonos végigjárása, amikoris minden egyes mintaelemhez talajprofilt rekonstruálunk valamilyen módszerrel. Ez az eljárás már a lineáris interpoláció alkalmazásával is igen időigényes, mert az adott számitógépen - annak ellenére, hogy megfelelő laptechnikával a háttértárolóhoz fordulások számát minimálisra csökkentettük - egy Budapest nagyságu terület feldolgozása, mely több mint 30000 mintaelemet tartalmazott, körülbelül 10 órás gépidőt vett igénybe. Nagyobb területek végigjárására ez a módszer nem alkalmazható, mert a talajprofil rekonstrukciók számitási és adatelérési lépésszáma miatt a gépidő egy négyzetes mátrix elrendezésü adathalmaz esetén a lineáris méretek köbével arányos.

Bonyolultabb ugyan az ugynevezett sugárszerű végigjárás

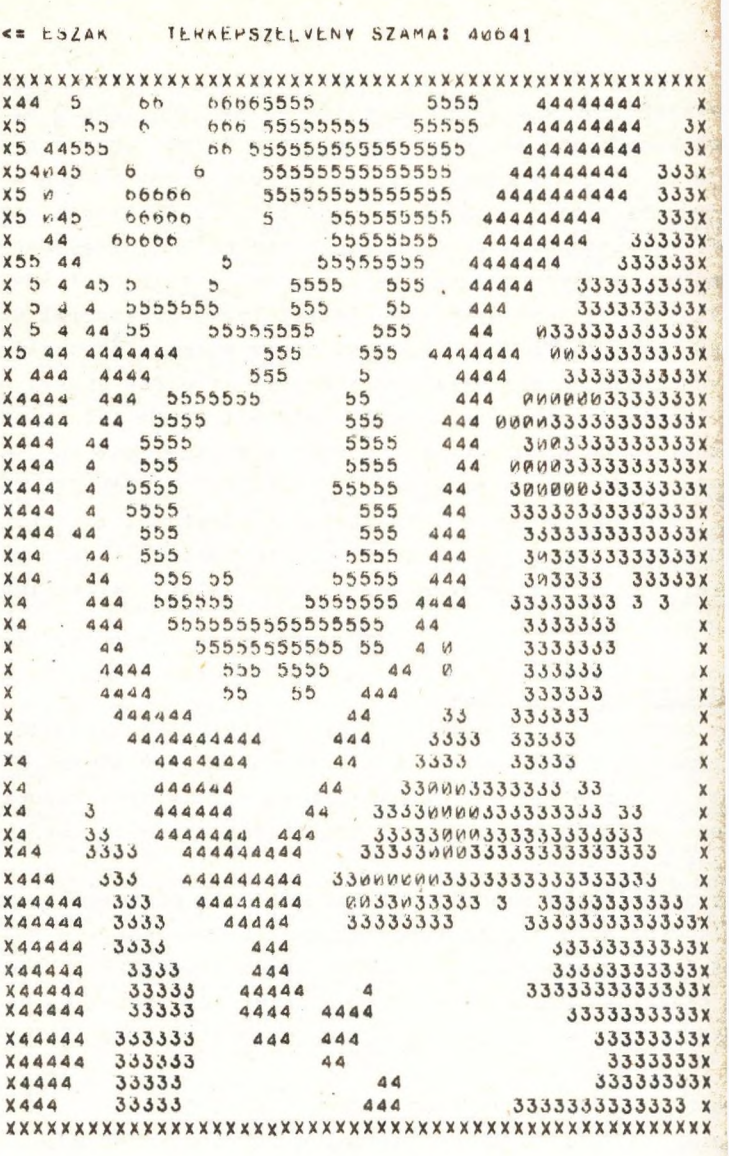
szervezése, de gépidőben jelentős csökkentés érhető el. Ennek alapötlete az, hogy megfelelően "sürü" sugársor surolja az összes mintaelemet, és igy egy sugár mentén a centrumból kifelé elindulva a már eddig előállitott talajprofil segitségével elegendő csak azokra a mintaelemekre számolni, amelyek ujonnan kerültek be. A talajprofil előállitását célszerü ekkor az adateltolásos módszerrel végezni, mert megfelelő - az adatmátrix soraira és oszlopaira vonatkozó - pointerek segitségével az előző sugárt számon lehet tartani, és igy ennek adatait ismét fel lehet használni, ha az uj sugár ugyanazt a mintaelemet surolja. Természetesen ezt a módszert is megfelelő laptechnikával kell kombinálni. Igy egy félmillió mintaelemet tartalmazó területet a hozzátartozó bonyolult felhasználói számitásokkal együtt is nem egészen 30 gépóra alatt fel tudtunk dolgozni.

MEGJELENITÉSI ELJÁRÁSOK

Nagy területre vonatkozó akár elsődleges, akár számitott információk numerikus megjelenitésekor a nagy tömegü adat miatt gyakorlatilag áttekinthetetlen számhalmazt kapunk. Éppen ezért fontos szerepe lehet minden vizuális megjelenitési módszernek.

A megjelenitésre szánt adattipus értékeit célszerü intervallumokba sorolni. Az intervallumok egymásutánja a felhasználási szempontoktól függően lehet egyenlőközü, vagy más beosztásu, de az intervallumba sorolást lehetőleg egy előzetes vizsgálat alapján megállapitott relativ gyakoriság figyelembe vételével készitett bináris fa segitségével végezzük el, igy a gépidőt csökkenteni tudjuk.

A feldolgozandó területet a leporelló méreteiből adódó korlátozásoknak megfelelő lapokra kell felosztanunk. Ez



1.sz. ábra

lehet egy szabványos méretü térképszelvény is. Egy ilyen megjelenitési formát láthatunk az 1. ábrán, ahol minden második intervallumba kerülő értékeknek a "space" felel meg. Ha ezekhez a ki nem töltött helyekre is értékes karaktert nyomtatnánk, elvesztenénk a vizuális áttekinthetőséget.
Ha az adott területen az értékek gradiense elég nagy, akkor könnyen előfordulhat, hogy az egymás mellé kerülő, de más-más intervallumhoz tartozó spacekről nem tudjuk hovátartozásukat eldönteni. Ezt a "komplemens" térkép elkészitésével kerültük el, melynél a spaceket most azokhoz az intervallumokhoz rendeltük, melyekhez az előzőkben értékes karakter tartozott.

Az eddig elmondottakat összefoglalva egy topográfiai adathalmaz csak a megfelelő kezelő rendszerekkel együtt válik adatbázissá. Az általánosan használt és közismert adatbázis kezelő elemeken tul a következő müveleteket és eljárásokat kell a kezelő rendszer elemévé tenni:
a./ Összefüggő területek előállitására szolgáló eljárások
b./ Két meghatározott tereppont közötti talajprofilt rekonstruáló eljárások, a különféle megjelenitési - pl. plotteres - megoldásokkal együtt.
c./ Egy-egy adattipus értékeit térképszelvényes formában megjelenitő eljárások. Ide tartoznak azok az eljárások is, melyek meghatározott számu egymás mellett elhelyezkedő mintaelemet összevonni képesek. Igy ugyan durvább felosztáshoz jutunk, de nagyobb területről tudunk átfogó képet adni. Erre azért van szükség, mert a félmillió mintaelemet tartalmazó munkánknál például az 1.sz. ábrán látható méretü térképszelvényeket összeillesztve már igen nagy - 1.75x3 m nagyságu térképekhez jutottunk.

A felsoroltakon kivül, melyek nélkül bármelyik topográfiai adatbázis alkalmazhatósága igen nehézkes lenne - természetesen még további, valamilyen más általános felhasználhatóságu eljárások is a kezelő rendszer elemeivé tehetők.

Irodalom.

1./ Györki-Majténvi: Az adatbázis kezelés problémái
SKV 1974

2./ Czigány-Kántor: Ürtávközlő földi állomás helykijelölő számitásai
PKI Közleményei XIV/1. 3-28.

3./ Dr. Villányi: Ürtávközlési földi állomás épül Magyarországon
Posta xXVI/9. 258-262

4./ R.Edwards-J.Durkin: Computer prediction of service areas for v.h.f. mobile radio networks
PROC. IEE, Vol. 116, No 9, Sept 1969. 1493-1500

5./ Czigány: A magyarországi ürtávközlő földi állomás létesitése
KTMF Tudományos Ülésszak 1975. 191-196

6./ Szörényi: Digitális térmodellek hasznositásának lehetőségei a hirközlésben
KTMF Tud. Ülésszak 1975. 211-214

INTEGRÁLT TERMELÉSIRÁNYÍTÁSI RENDSZER HAZAI GYÁRTÁSÚ KISSZÁMÍTÓGÉPRE (INTERTIP)

Novotny Lászlóné
Kohó- és Gépipargazdasági, Szervezési és Számítástechnikai Intézet

A Videoton Számítógépgyára által gyártott kisszámítógépek termelésirányítással kapcsolatos alkalmazói software kialakítására az INFELOR Rendszertechnikai Vállalat és a KGM ISZSZI tette meg az első lépéseket. Az INFELOR MM rendszere lehetővé tette, hogy olyan vállalati feladatokat, amelyek állományfelépítésre, karbantartásra, rendezésre, válogatásra, az állományból történő kiiratásra alapulnak, egyszerűen lehessen megoldani.

A KGM ISZSZI a Videoton Számítógépgyára részére 1971-ben egy VT 1010B számítógépre alkalmas számítógépes termelésirányítási rendszer kidolgozását kezdte meg. 1974-ben a rendszert R10 számítógépre dolgozta át, és a programok már erre a számítógépre készültek el. Az így kialakított programrendszer a PROGRAMOZÁSI RENDSZEREK '75 Konferencia egy másik előadása keretében kerül ismertetésre.

Ezeknek a tapasztalatoknak a birtokában 1974-ben kezdte el az Intézet egyik munkacsoportja - amelynek én is tagja vagyok - egy integrált termelésirányítási programrendszer kidolgozását.

Célkitűzésünk az, hogy a hazai kisszámítógép önálló alkalmazása érdekében olyan programrendszert dolgozzunk ki, amely rendszerében integrált, alkalmazása a hardware lehetőségek figyelembe vételével modulárisan történhet a felhasználó vállalat szervezettségi szinvonala, igényei szerint és amely kompatibilis a nagyobb ESZR gépek programrendszereivel.

Felvetődhet az a kérdés, hogy vajon miért nem a sok helyen

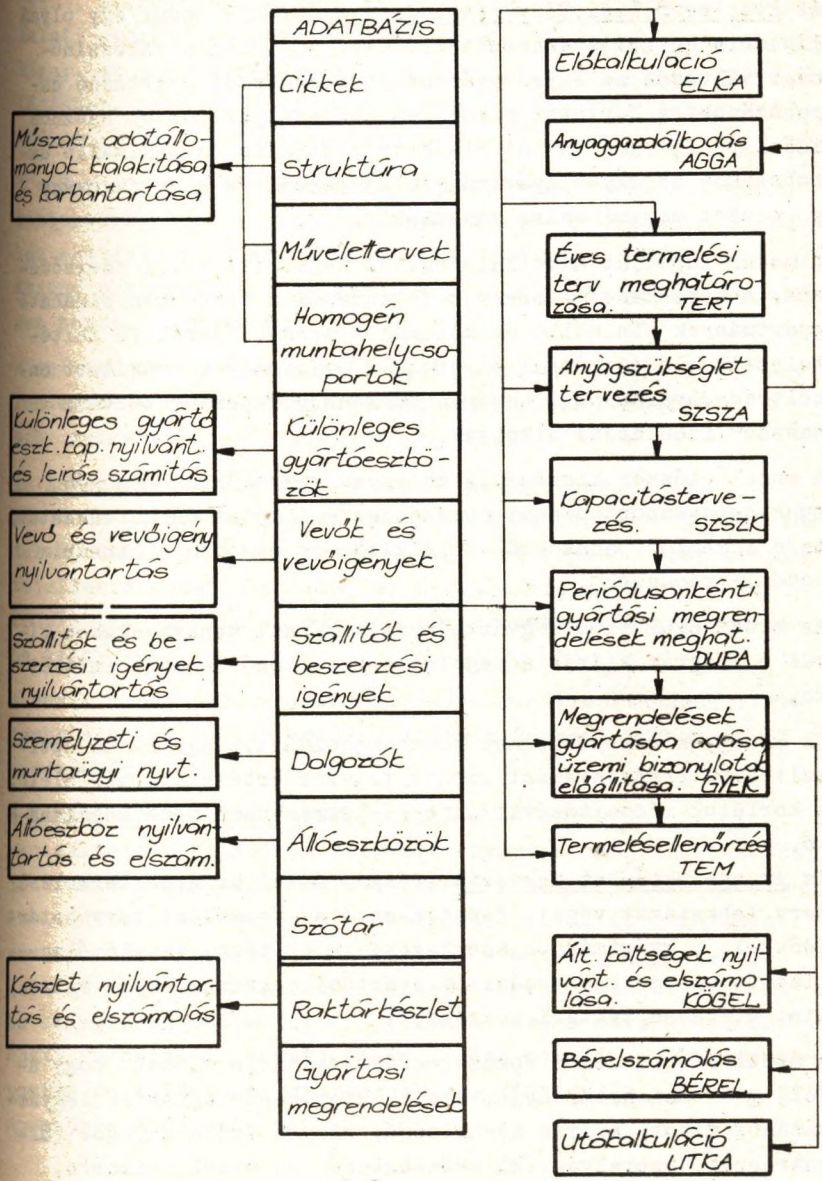
alkalmazott, ismert számítógépes cégek által kidolgozott termelésirányítási programcsomagok közül próbáltunk kiválasztani egyet, és azt a hazai kisszámítógépre áttenni.

Természetesen ismerjük ezeket a programcsomagokat és ahol lehet, az alkalmazott megoldásokat fel is használjuk a rendszerünkben. Ezek a programcsomagok sok olyan feladatot megoldanak, amely a mi gyártó-szerelő vállalataink többségének technológiai és szervezettségi szinvonalán jelenleg nem követelmény. Ilyenek pl. a napi gyártási program kiadása, a gyártási út optimalizálása, a tényleges gyártás és a gyártási program összehasonlítása alapján történő számítógépes programmódosítás, stb. Ugyanakkor nem tartalmaznak megoldást sok olyan feladatra, amelyet adott hazai vállalatnál éppen munkaigényességénél fogva célszerű számítógéppel elvégeztetni. Ilyenek pl. a főkönyvi elszámolások, statisztikai kimutatások elkészítése a KGM, a KSH, a KGM MTTI, stb. felé, az állóeszközök értékcsökkenési leírásának számítása, a saját gyártású termékek elő- és utókalkulációja, az elszámolóár karbantartása, stb. A most ismertetésre kerülő programrendszer - többek között - ezeket a feladatokat is megoldja.

Az Integrált Termelésirányítási Programrendszer (röviden INTERTIP) moduláris felépítésű rendszer. Szerkezeti felépítése, adatbázisának és moduljainak kapcsolata az ábrán látható.

A rendszer gerincét a közvetlen termelésirányítási feladat megoldására alkalmas modulok alkotják, de - éppen a széleskörű alkalmazhatóság érdekében - tartalmaz a tervezéssel, elszámolásokkal és egyéb, pl. nyilvántartásokkal kapcsolatos feladatok megoldására alkalmas modulokat is.

Jelen előadás keretei között nincs lehetőség a teljes rendszer részletes bemutatására, ezért most csak a lényegesebb modulok működését fogom ismertetni.



Az adatbázis és a modulok főbb kapcsolatai

Az Éves Termelési Terv meghatározása (TERT) modul egy olyan lineáris egyenletrendszert old meg, amelyben a valószínűségi változók az egyes gyártmányféleségekből gyártandó darabszámok, a korlátok a homogén munkahelycsoportok hasznos időalapjai, valamint az eladható minimális és maximális darabszámok az egyes gyártmányféleségekből és a célfüggvény a fedezet maximálására vonatkozik.

A modul inputját a vállalat által megkötött vevői szerződések, a kereskedelmi szervek jelentései a tervévben eladható gyártmányok minimális és maximális mennyiségéről (a feltételezett eladási árral együtt), a gyártmányok szűkített önköltsége, valamint a homogén munkahelycsoportok tervévi hasznos időalapjai alkotják.

A modul először kiszámítja az egyes gyártmányok homogén munkahelycsoportonkénti normaóraszükségletét és a fedezetet, majd a meglevő adatokból felállítja és megodja a lineáris egyenletrendszert.

Az eredményül kapott gyártmányfajtánkénti mennyiségeken kívül a program kiírja a megoldáshoz tartozó fedezet értékét is.

Ha az egyenletrendszernek nincs megoldása, vagy ha az optimalizálás eredményeként kapott fedezet értéke nem megfelelő, a korlátok módosításával az egyenletrendszer újra megoldható.

Az Anyagszükséglet Tervezés (SZSZA) modul az éves termelési terv lebontását végzi. Inputja az éves termelési terv határidőkkel, a gyártmányok szerkezeti felépítése, az előzőleg kiadott szabad beszerzési és gyártási megrendelések, valamint a szabad raktári készletek.

A modul diszpozíciós fokozatonként számítja ki azt, hogy adott gyártási határidejű, adott mennyiségű végtermék legyártásához mikor, milyen mennyiségű, milyen fajta anyagok (alkatrészek, szerelvények) szükségesek, és miből, mikorra, milyen mennyiségű anyagot kell beszerezni, illetve alkat-

részt, szerelvényt gyártani. (A diszpoziciós fokozat a vállalati gyártmánystruktúra adott elemének jellemző tulajdonsága; azt a legalacsonyabb szintet jelenti a gyártmány-családfában, amelyen az illető elem a vállalat teljes gyártmánystruktúráját tekintve előfordul.)

A modul a számítás során figyelembe veszi a selejt és a biztonsági készlet miatti készletigényt (gyártási megrendelés esetén), valamint a raktárban levő szabad készletet, a szabad beszerzési és gyártási megrendeléseket és a gazdaságos sorozatnagyságot.

A számítás eredménye a Határidős Tervezett Beszerzési Megrendelések és a Határidős Tervezett Gyártási Megrendelések, melyeket kinyomtatva is megkapunk.

Az Anyaggazdálkodás (AGGA) modul az SZSZA modul által megadott Határidős Tervezett Beszerzési Megrendelésekből kiindulva megállapitja az egyes anyagokhoz tartozó beszerzési tételnagyságokat a minimális, a maximális és a biztonsági készletek, valamint a várható selejt nagyságának figyelembe vételével.

A modul a beszerzési tételnagyságok megállapitásánál figyelembe veszi a raktárforgalom alapján végzett ABC analizis eredményét is.

A Kapacitástervezés (SZSZK) modul az SZSZA modul által szolgáltatott Határidős Tervezett Gyártási Megrendelésekből indul ki. A gyártási megrendelésekhez hozzárendeli a műveleti időket, a homogén munkahelycsoportokat és a különleges gyártóeszközöket; meghatározza a műveletcsoportokhoz tartozó időszükségleteket, valamint egy tervezett kezdési és befejezési időpontot (figyelembe véve a műveletcsoportok közti várakozási időket), majd minden homogén munkahelycsoportra és különleges gyártóeszközre kiszámítja a tervezett normaóra-terhelést.

A modul által nyomtatott tabló a homogén munkahelycsoportok periódusonkénti hasznos időalapját és a tervezett terhelést egymás mellett mutatja meg, feltüntetve az esetleges különleges gyártóeszköz-hiányt is. A terhelési kimutatás alapján intézkedések tehetők a hiányzó gyártóeszköz előállítására, a homogén munkahelycsoporton hiányzó normaórák pótlására vagy a felesleges idő lekötésére.

A Periódusonkénti Gyártási Megrendelések Meghatározása (DUPA) modul az SZSZK modul által szolgáltatott homogén munkahelycsoportonkénti periódus-terheléseket a meglevő hasznos időalap-értékekhez próbálja közeliteni helyettesitő technológia és/vagy helyettesítő homogén munkahelycsoport alkalmazásával, vagy ha ez nem vezet megfelelő eredményre, a műveletcsoportok közti várakozási idők csökkentésével.

A számitás eredménye egy új Tervezett Gyártási Megrendelések állomány, amely az előzőnél reálisabb inditási határidőket tartalmaz, valamint egy új Terhelési Kimutatás. Ez a tabló a homogén munkahelycsoportokra vonatkozó terheléseket az előző Terhelési Kimutatáshoz képest bekövetkezett változásokkal együtt mutatja ki.

A Gyártáselőkészités (GYEK) modul a Tervezett Gyártási Megrendelésekből indul ki. Bizonyos időszak, pl. 9 periódus (egy negyedév) tervezett gyártási megrendeléseire ellenőrzi, hogy a szükséges anyagok (alkatrészek, szerelvények) rendelkezésre állnak-e, illetve feltehetően rendelkezésre fognak-e állni. Ezt a mennyiséget "foglalt"-nak nyilvánitja, majd a gyártáshoz szükséges bizonylatok (gyártási tasak, anyagutalvány, munkautalvány, szerszámutalvány, rajzszámjegyzék) adatait összeállitja és kinyomtatja.

A Tervezett Gyártási Megrendelések ellenőrzött és kinyomtatott részét a Nyitott Gyártási Megrendelések közé viszi át, és az üzem részére az időszakra (pl. az emlitett 9 periódus-

ra) vonatkozó gyártási programot nyomtat ki, melyben fel van tüntetve, hogy az adott költséghelynek miből, mikorra, milyen mennyiséget kell gyártania.

A Termelésellenőrzés (TEM) modul a Nyitott Gyártási Megrendeléseket az elkészült gyártási megrendelésekkel hasonlitja össze és szükség esetén Eltérési Jegyzéket készit.

Az Előkalkuláció (ELKA) modul a saját gyártású alkatrészekre, szerelvényekre és végtermékekre számitja ki az ún. szűkített önköltséget, amely a következő tételekből áll: közvetlen anyagköltség, közvetlen bérköltség, a közvetlen bérek közterhei, gyártási különköltség, értékesitési különköltség, üzemi általános költség és selejtveszteség.

A modul tervezett adatokkal számol; először a cikkek közül kiválasztja azokat, amelyekhez a legnagyobb diszpozíciós fokozat tartozik, majd az alkatrészekre vonatkozóan egyenként kiszámítja a szűkített önköltség tételeit.

A közvetlen anyagköltséget a program az alkatrészhez tartozó anyagmennyiség és az adott anyagra megállapított elszámolóár szorzásával, majd a szorzatok összegzésével, a közvetlen bérköltséget a műveletterven szereplő műveleti idő és a művelethez tartozó átlagbér szorzásával, majd a szorzatok összegzésével határozza meg. A közvetlen bérek közterhét a közvetlen bérből egyszerű szorzással számitja.

Az alkatrészhez tartozó gyártási különköltséget a műveletterven meghatározott különleges gyártóeszközök egységnyi leírási költségeinek összegzésével, az üzemi általános költséget a művelet elvégzésének helyéhez megállapított rezsikulcs és a közvetlen bér szorzásával, majd a szorzatok összegzésével számítja ki a program.

Az értékesítési különköltség összegét a program nem számitja, hanem az adott alkatrészhez tartozó adatot háttértárolóról hívja meg.

Ezután következik az előző költségtételek összegezése, és a selejtveszteség értékének az összeg és az alkatrészhez tartozó átlagos selejtszázalék szorzásával történő kiszámítása, majd a szűkített önköltség értékének megállapítása.

A program tárolja a kalkulációs tételeket és azok összegét valamennyi legnagyobb diszpozíciós fokozattal rendelkező alkatrészre, majd veszi az eggyel kisebb diszpozíciós fokozattal rendelkező cikkszámokat.

Ezek vagy alkatrészek, vagy szerelvények lehetnek. Ha alkatrészről van szó, akkor a számítás az előzőekben leírtak szerint történik; ha szerelvényről van szó, akkor az egyes költségtételek két részből tevődnek össze: az egyik a beépülő alkatrész és/vagy szerelvény megfelelő költségtételei, a másik pedig magára a szerelvényre vonatkozó költségtételek, amelyek kiszámítása ugyancsak az előzőekben leírtaknak megfelelően megy végbe.

Az eddig ismertetett modulok a termelés tervezésével és végrehajtásával voltak kapcsolatban. Az ezután következő modulok a termeléssel kapcsolatos nyilvántartások és elszámolások körébe tartoznak.

Az Utókalkuláció (UTKA) modul az ún. szűkített önköltségszintig számítja ki az egy-egy munkaszámra jutó költségeket, amelynek tételei megegyeznek az Előkalkuláció modulnál felsorolt költségtételekkel.

A program az elszámolt anyag-, munka- és szerszámutalványokból indul ki. Ellenőrzi, hogy a munkaszámra kiadott valamennyi anyag- és munkautalvány (a selejt pótlására szolgáló anyag- és munkautalvány is) elszámolásra, a szerszám visszavételezésre került-e. Ha igen, összegezi a munkaszámra kiadott összes anyag értékét elszámolóáron, (a selejt miatti pótanyagokat elkülönítve ugyanolyan módon) és a munkautalványokhoz tartozó bérköltségeket, (elkülönítve a selejt miatti pót-munkautalványokhoz tartozó bérköltsége-

ket.

A munkaszámonkénti közvetlen bérköltségekhez kiszámítja a közvetlen bérek közterheit is.

A gyártási és értékesítési különköltség számításánál a külön modulban kiszámított, a munkaszámra vonatkozó gyártóeszközleírást illetve értékesítési különköltséget az UTKA munkaszámonként összegzi.

Az üzemi általános költség számítása az üzemekre megállapított rezsikulcsok és a munkaszámhoz üzemenként tartozó közvetlen bérköltség szorzásával és a szorzatok összegezésével történik.

A selejtveszteség számításánál a modul összegzi a selejt miatti pótanyag- és pót-munkautalványokhoz tartozó költségeket.

A modul a kapott költségtételeket munkaszámonként kinyomtatja és összegzi.

A Bérelszámolás (BÉREL) modul a lineáris teljesítménybéres és az időbéres (órabéres és havidíjas) dolgozók bérelszámolását végzi. (A rendszer kiegészíthető a nem-lineáris teljesítménybéres dolgozók bérelszámolásával.)

A modul a lineáris teljesítménybéres dolgozók elszámolásánál az elszámolt munkautalványokból és a munkaidő-elszámolási bizonylatból, az időbéres dolgozók elszámolásánál a munkaidő-elszámolási bizonylatból indul ki. A munkaidő-elszámolási bizonylat megadja a műszakonkénti, munkaszámonkénti óraráfordításokat, távollét esetén a hiányzás okát olyan kódszámmal megjelölve, amely lehetővé teszi a kiegészítő fizetések elszámolását. A modul a rendszeres bérpótlékokat és levonásokat a Dolgozók Törzsállományából, a rövidebb időszakra vonatkozó bérpótlékokat és levonásokat pedig külön bizonylatról kapja.

Az Általános Költségek Nyilvántartása és Elszámolása (KÖGEL) modul a rendszer indulásánál a költséghelyenként érvényes rezsikulcsokat tartalmazza. Az év folyamán a modul gyüjti az egy-egy költséghelyre terhelt általános költségeket (az adatokat input bizonylatról, illetve az Állóeszközök Nyilvántartása és Elszámolása modul által szolgáltatott költséghelyenkénti értékcsökkenési leírás-adatokat tartalmazó állományból veszi) és a költséghelyenkénti közvetlen munkabéreket.

Év végén új rezsikulcsokat számít a költséghelyre összegyüjtött általános költségek és közvetlen munkabérek hányadosának képzésével, majd az új rezsikulcsokról kimutatást készit.

A Különleges Gyártóeszköz Kapacitás-nyilvántartás és Leírásszámítás modul a Különleges Gyártóeszközök állományából kiindulva kiszámitja a különleges gyártóeszközök periódusonkénti hasznos időalapját, a különleges gyártóeszközöknek a raktárból gyártás céljára történt kivételezése és visszavételezése esetén a gyártóeszköz értékleírásának összegét és aktuális nettó értékét. A periódusonkénti hasznos időalap-adatokat a Kapacitástervezés modul, a gyártóeszköz-leírás értékét az Utókalkuláció modul, az aktuális nettó értéket a Készletnyilvántartás és Elszámolás modul használja fel, de ezek az adatok tabló formájában is kinyomtathatók.

Az Állóeszköznyilvántartás és Elszámolás modul leltár alapján állóeszközállományt hoz létre, az állóeszközökre vonatkozó mozgásadatokkal az állományt aktualizálja és negyedévenként illetve év végén kiszámítja az egyes állóeszközökre vonatkozó értékcsökkenési leírást és az állóeszközök nettó értékét, valamint statisztikákat készit.

A Készletnyilvántartás és Elszámolás modul a raktári készleteket a készletforgalmi állomány segítségével naprakészen

tartja. A raktári készletmozgásokat mennyiségben és értékben havonta főkönyvi számlaszámonként kiírja.

Az ismertetett programrendszer kidolgozása most folyik. Természetesen tisztában vagyunk azzal, hogy ilyen termelésirányítási programrendszer kidolgozása nem egyszerű feladat. A rendszer kidolgozását egyszerűbb modulokkal kezdtük és úgy folytatjuk, hogy az egymásra épülő modulok lehetőleg egymás után kerüljenek kidolgozásra. Úgy tervezzük, hogy az elkészült modulokat néhány vállalatnál - a többi modul kidolgozásával párhuzamosan - először külön-külön vezetjük be, és így már menet közben tudomást szerzünk az esetleges hibákról. A szükséges módosításokat a vállalatnál alkalmazott modulra épülő másik modulnál már figyelembe tudjuk venni.

- . -

SZÁMITÓGÉPPEL SEGITETT TERMELÉSIRÁNYITÁSI RENDSZER SOFTWARE-JE

Gacsádi Lórándné
MTA Számitástechnikai és Automatizálási Kutató Intézet

1. Feladat kitüzése

1.1. A SzTAKI Müszaki Osztályán kifejlesztett számitógépes Termelés Irányitási Rendszer /TIR/, amely az első függelékben kerül ismertetésre számos software problémát vetett fel, amelyek megoldásáról ad tájékoztatást ez az előadás.

1.2. A feladat tulajdonképpen rendkivül egyszerüen megfogalmazható: Listát adni mindennap, minden mühely számára, azokról az alkatrészekről, melyek kezdésének minden feltétele adott.
Tekintettel az adottságokra és a követelményekre azámos speciális megoldást alkalmaztunk, amelyek ismertetése nem lesz érdektelen.
Adottság : a./ NOVA 1200 /DOS/ kisszámitógép a házban
b./ 2,5 Mbyte disk háttér tárolóval
c./ 16 Kszó központi memóriával

Követelmény :
a./ az operativ futásidő nem haladhatja meg az egy órát
b./ a rendszer legyen illeszthető más alrendszerekhez.

2. Megoldás

2.1. A követelményeket és adottságokat ugy kellett közös nevezőre hozni, hogy az eredmény optimális legyen.
A megoldásig a következő lépésekben jutottunk el.

2.2. A NOVA 1200-nak FORTRAN, ALGOL, ASSEMBLER és BASIC compilere van. Miután minden TIR nagy adathalmaz kezelését igényli, természetesen a COBOL nyelv lett volna legalkalmasabb a rendszer megirására, ennek hiányában a FORTRAN mellett döntöttünk.

2.3. A file szervezésnél a következő tényezők játszottak szerepet:

a./ Rendelkezésünkre áll egy 2,5 Mbyte-os disc amelyet teljes egészében felhasználhatunk.

b./ Az operativ futásidő leröviditése rendkivül fontos szempont.

c./ Egyszerre maximum 200 hálóterv adatait kell nyilvántartanunk.

d./ A különböző adatok aktivitási szintje pontosan definiálható.

e./ Az 1. Függelékben részletezett 4 adatcsoport mindegyike különböző rekord hosszuságot igényelt.

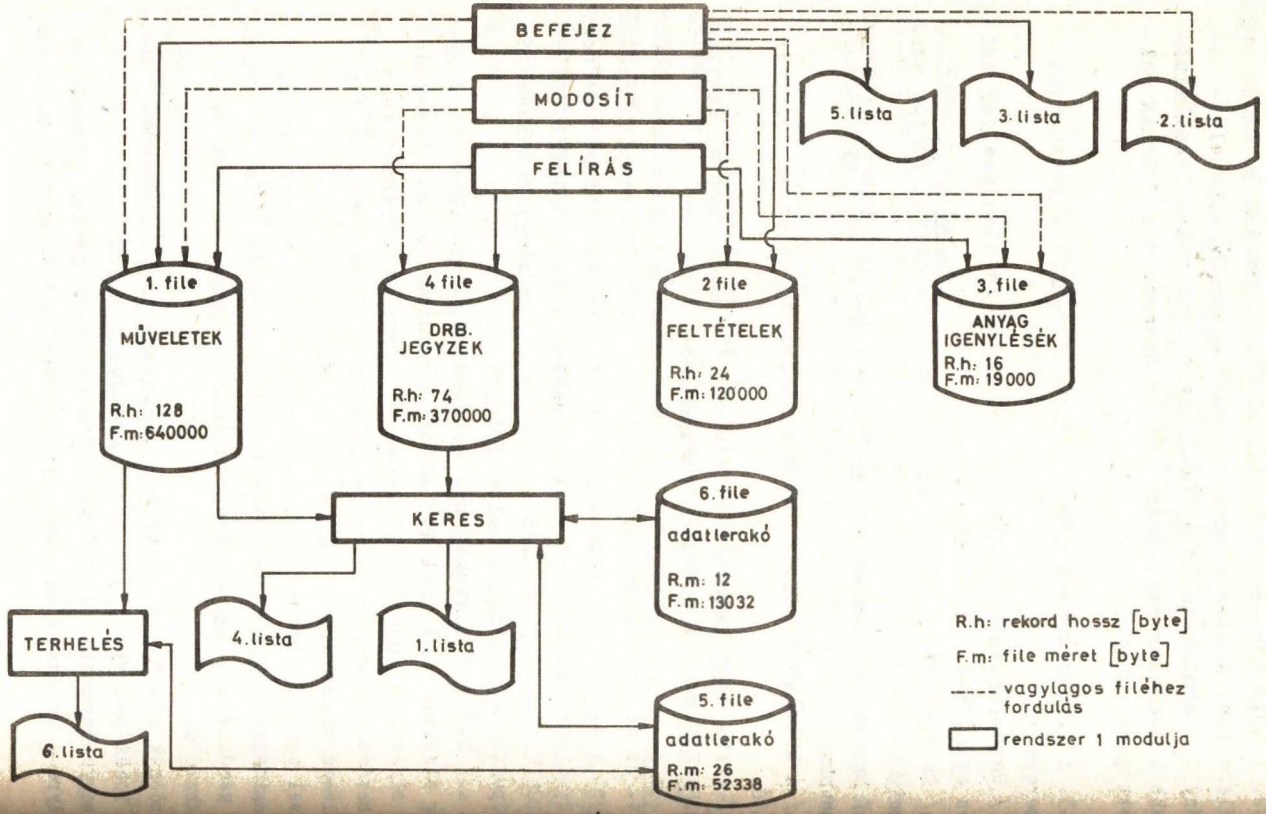
2.4. Az adatoknak az 1.ábra szerinti szétválasztását egyrészt a d./ és e./ pontok tették indokoltá, másrészt az hogy a rendszert felépitő 5 modul /1-ábrán a FELIRÁS,BEFEJEZ, MÓDOSIT, KERES, TERHELÉS nevü modulok/ mindegyike különböző időpontokban fordul a különböző file-khez.

Az a./ b./ c./ pontokat ugy értékeltük, hogy ilyan viszonylag alacsony szám mellett /200/ egy durva /rough/ indextábla elfér a központi memóriában és gyors hozzáférést tesz lehetővé egy-egy hálótervhez, ha egy hálótervnek a 4 fix file-re vonatkozó 4 indexét megadjuk.

Mivel egy hálótervet 1 file-én átlagban 30 rekord ir le, /maximális rekordméret: 128 byte/ ez szintén elfér a központi memóriában, igy szekvenciálisan is gyors keresés valósitható meg.

Mindezek miatt a 4 adatfile-t hálótervenként indexelve, durva index-szekvenciális hozzáférésüvé tettük.

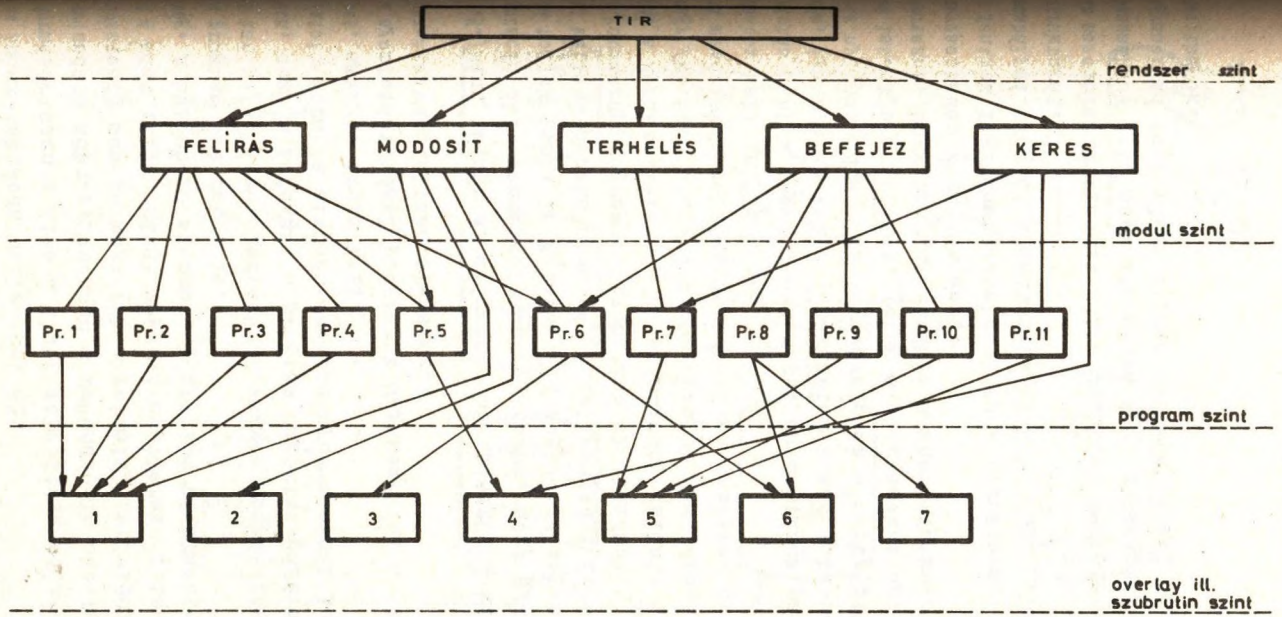
A hagyományostól eltérő megoldás az, hogy néhány adatot, amelyre a hálóterv minden alkatrészénél szükség van, szintén


BEFEJEZ
MODOSÍT
FELÍRÁS
5. lista
3. lista
2. lista
1. file
MŰVELETEK
R.h: 128
F.m: 640000
4 file
DRB. JEGYZÉK
R.h: 74
F.m: 370000
2 file
FELTÉTELEK
R.h: 24
F.m: 120000
3. file
ANYAG IGÉNYLÉSÉK
R.h: 16
F.m: 19000
KERES
6. file
adatlerakó
R.m: 12
F.m: 13032
TERHELÉS
4. lista
1. lista
6. lista
5. file
adatlerakó
R.m: 26
F.m: 52338
R.h: rekord hossz [byte]
F.m: file méret [byte]
vagylagos filehez fordulás
rendszer 1 modulja

1.ábra.

az index-táblában helyeztünk el.
Természetesen a file-k ilyen felépitése miatt előfordulnak redundáns adatok, azonban a redundancia okozta területi veszteség messze megtérül a futásidő lerövidülése által.
A 4 adat file-n kivül még két adatlerakó file-re volt szükség. A különböző szempontok szerint kiválasztott adatokat a program ide rakja le.
E két file direktszervezésü, vegyes hozzáféréssel, olymódon, hogy a lerakást a program direkthozzáféréssel végzi /adott formula alapján egy bizonyos adatból leképzi a cimet, amelyre irnia kell./
A nyomtatást végző programok ugyanezekről a file-kről index-szekvenciálisan olvasnak.

2.5. Mint az már az előzményekből kiderült, kisgépes konfiguráció állt rendelkezésünkre. Ahhoz, hogy ezt a rendszert rá tudjuk tenni 16 Kszavas /32 Kbyte-os/ gépre maximálisan ki kellett használni a gép által nyujtott lehetőségeket és a korszerü programozástechnikai eljárásokat.
Már a rendszertervezésnél számolnunk kellett azzal, hogy a nagy /kb.6ooo utasitásos/ programok nem férnek be a központi memóriába, tehát a meglehetősen nehézkes overlay megoldáshoz kell folyamodnunk.
Miután a NOVA overlay-nél nincs automatikus paraméterátadás, igy nyitni kellett paraméter átadó file-ket, amelyeknek nyitása-zárása és természetesen a memóriatükör készitése lassitja a futaást, tehát a programokat ugy kellett megtervezni, hogy overlay hivásra a lehetőségekhez képest legkevesebbszer kerüljön sor.
A fentiek figyelembevételével és a strukturált programozás szabályai szerint alakitottuk ki a 2. ábra szerinti szintstrukturát.
Az assembler rutinokat külön szintnek tekintettük és ugy használtuk fel, hogy más gépre történő adaptáció esetén ugyanolyan funkciókat ellátó szubrutinokkal legyenek helyettesithetők.


TIR
rendszer szint
FELIRÁS
MODOSIT
TERHELÉS
BEFEJEZ
KERES
modul szint
Pr. 1
Pr. 2
Pr. 3
Pr. 4
Pr. 5
Pr. 6
Pr. 7
Pr. 8
Pr. 9
Pr. 10
Pr. 11
program szint
1
2
3
4
5
6
7
overlay ill.
szubrutin szint
FORTRAN-ból hivható assembler rutinok
Pr. program

2. ábra.

A konkrét termelésirányitási feladatot végző programokon kivül számos tesztelő és néhány, a rendszer karbantartásához szükséges program készült /összesen 34 db./

2.6. A további bővités, illetve más alrendszerek illesztésének lehetőségét biztositottuk.
Ehhez azonban szükség volt arra, hogy az illesztendő alrendszerek által végzett funkció már a tervezésnél ismert legyen.
A vezetés által vázolt nagyvonalu koncepció tartalmazta a célkitüzéseket, ezért tudtunk konkrét, nem általános megoldásokat alkalmazni /pl. a számitógépes anyaggazdálkodás beindulása esetére fenntartunk hálótervenként 6o byte-t, ez pillanatnyilag üres/.
A rendszer 1975. április 7.-e óta üzemel, bármikor megtekinthető.
Az out-put-okat a 2. Függelék tartalmazza.

2.7. Konfiguráció igénye: 16 Kszó központi memória
2,5 Mbyte disc
kártyaolvasó
sornyomtató /8o karakteres/
mágnesszalag

Gépidő igénye: 6o perc operativ futásidő
15 perc az adatok mágnesszalagra történő mentése
1o-6o perc esetenként változóan az uj adatok bevitele ill. a módositása.

Végezetül annyit jegyzünk meg, hogy a rendszer alkalmas nagyobb termelő egységek irányitására is, a bővitésnek a disc kapacitás szab határt.

1. Függelék a szervezeti rend, az üzemeltetés, a modulok funkcióinak és a file-k konkrét tartalmának leirását tartalmazza.
2.Függelék az output-ok leirását és a próbaüzem tapasztalatait foglalja magába.

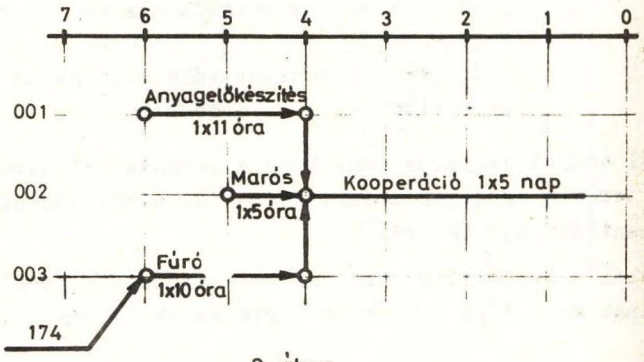
1.Függelék

1.Szervezeti rend

A SzTAKI műhelyeinek termelésére jellemző a termékek rendkivül sokfélesége és a viszonylag alacsony legyártandó darabszám. A fenti két tényből következően mind nehezebbé vált az alkatrészek munkábavételének sorrendjét helyesen megállapitani, készültségüket nyilvántartani.
Az előadásban ismertetett rendszer alapvető célja e feladatnak számitógéppel történő megoldása.
Ez a rendszer az itt ismertetésre kerülő szervezeti rend keretein belül valósulhatott meg.

Abban a szerencsés helyzetben voltunk, hogy Pólya Endre osztályvezető vezetése alatt már évekkel előbb bevezetésre került a hálótervezési módszer az osztályon, igy a szükséges szervezeti átalakitások, melyet Perjés Miklós szervezett, nem voltak jelentősek.

A korábbi gyakorlatnak megfelelően a szerkesztésekből gyártásra leadott dokumentáció alapján a gyártáselőkészités technologizál és hálótervet készit /3.ábra/. Ez a hálóterv tartalmaz minden olyan adatot, amelyekre a gyártásnál / a rajzokon kivül/ szükség van.



3. ábra.

A 001, 002, 003 az alkatrész azonositó, a 003-hoz mutató nyil fölött látható 174 az alkatrészhez rendelt anyagigénylési száma. Az ábra felső részében elhelyezett számsor napokat jelent. Ebből egyértelmüen leolvasható pl. hogy a 001 alkatrész első müveletét a véghatáridő előtt 6 nappal kell elkezdeni. A véghatáridő / befejezési dátum / változtatása tehát nem érinti az alkatrészek leirását, mert ott csak azt kell megadni, hogy a mindenkori befejezési dátumhoz képest az adott müveletet hány nappal korábban kell elkezdeni.

A számitógép számára a hálótervekről szintén a gyártáselőkészités tölti ki az előrenyomtatott kódlapokat, amelyek ne.. sematikus rajza a 4. ábrán látható.

A gép minden nap 12^{30}-kor ad uj listát a mühelyek számára sürgösségi sorrendben azokról a müveletekről, amelyek elkezdésének mind feltétele teljesült /vagy anyag, szerszám stb./.

A rövidtávu tervezés szabályainak megfelelően a lista 2 napra elegendő munkát tartalmaz. A kiadandó munka mennyiségéről a termelésprogramozó dönt. Egyedül neki áll jogában a gép által diktált sorrendet esetleg megváltoztatni.

Az elkezdett ill. elkészült munkáról másnap 11^{00}-kor minden munkahely készrejelent. Igy az uj lista már a megváltozott adatoknak megfelelően készül.

2.1. A rendszer 5 modulja által végzett funkciók:

FELIRÁS: ellenőrzi, felviszi az adatokat, karbantartja a lefoglalt, ill. felszabadult helyeket.

KERES: napi listát készit a munkahelyek számra.

BEFEJEZ: az adatok naprahozását végzi a napi készrejelentések alapján.

TERHELÉS: a munkahelyek teljes terhelését adja meg az összes bentlévő munka alapján, hálótervenként.

MÓDOSIT: lehetőséget ad bármelyik adat megváltoztatására. Interaktiv kapcsolatot létesit a termelés programo-

zó és a gép között. Egyedül a termelés programozónak áll jogában bármilyen okból kifolyólag módositani.

2.2. A négy adatfile tartalma:

MÜVELETI file: egy rekordba egy alkatrésznek tiz müveletét leiró információn kivül a legyártandó darabszám és a legkorábbi kezdés kerül. Egy müveletet hat adat ir le /lásd. 4.ábra/ ugymint munkahely kód, becsült müveleti-idő, feltétel kapcsoló, igény és kapcsoló, dolgozók száma, és a legkésőbbi kezdés ideje.
Az emlitett két kapcsoló állásától függ, hogy egy alkatrész kezdhető-e vagy sem. A kapcsolók átállitását a BEFEJEZ végzi, a pillanatnyi állapotnak megfelelően.

FELTÉTEL file: egy rekord egy alkatrész egy müveletének feltételeit tartalmazza, maximálisan ötöt. Ha egy müveletnek ötnél több feltétele van, akkor a továbbiak számára uj rekordot kell nyitni.

IGÉNYLÉS file: egy alkatrész egy müveletéhez tartozó anyagigénylési számok kerülnek egy rekordba, maximálisan öt, a továbbiak számára szintén uj rekordot kell nyitni.

DRB JEGYZÉK file: az alkatrész megnevezését, rajzszámát, darabolási méretét és induló darabszámát tartalmazza.
A rekord 3 alfanumerikus mezőből áll, ezért alkalmas bármilyen, az alkatrészre vonatkozó megjegyzés tárolására is.
Mint látható a legmagasabb aktivitási szintü file rekord méretét ugy választottuk meg, hogy az a gép számára optimálisan kezelhető legyen.

2. Függelék

1. A rendszer legfontosabb outputjai: /9 ábra/

- munkaprogram a mühelyek számára /9.ábra/
- lista az anyagbeszerzés számára azokról az anyagigénylési számokról, amelyeknek a kért határideje lejárt /8.ábra/

- munkaidő elszámolás naponta, homogén munkahelyenként. /5.ábra/
- terhelési tábla /lo. ábra/
- kooperációs megrendelés /11.ábra/
- munkaidő ráforditás hálótervenként /6.ábra/
- elkészült hálótervre forditott össz munkaóra /7.ábra/

E listák közül az 5., 8., 11., 9. ábrán láthatókat minden nap szolgáltatja a gép. A 7-en láthatót automatikusan egy hálóterv elkészültével, a 6. és lo-en lévőket esetenként külön kérésre.

1.2. A próbaüzem 1975. márc. 3-án indult meg. A várakozással ellentétben a műhelyekben igen kedvező fogadtatásra talált. Az egyhónapos próbaüzem alatt előjött hibát könnyen lehetett javitani, igy április 7-én kedvező tapasztalatok után, leállás nélkül indult meg az élesüzem.

Kezdetben még manuálisan is készült munkaprogram, ez azonban rövidesen feleslegesnek bizonyult. Fennakadás egyszer fordult elő hardware meghibásodás miatt. Jelenleg a software dokumentálás folyik.
A rendszer most 11 homogén munkahely irányitását végzi, gyorsan, pontosan.
Eddigi tapasztalatok szerint a gyártmányok átfutási ideje lerövidült, kevesebb a határidő reklamáció, mert az igért határidőt nagy valószinüséggel tudják tartani.
Rendkivül rugalmas irányitást ad a tekintetben, hogy egy hálóterv prioritásának megváltoztatása igen egyszerű. Másrészről nincs semmilyen korlát állitva a hálóterv méreteinek. /Eddig a legnagyobb hálóterv 22o alkatrészből állt./

MAI DATUM: 1975. 6. 5.

MUNKAIDO RAFORDITAS HALOTERVENKENT

SZAM	AGE	ESZ	MAR	KSS	HFR	GRA	LAK	MUSZIEMU	NYAK	KOUP
133	2	0	0	0	0	0	0	317	0	0
314	1	0	5	0	0	0	0	1001 256	0	0
462	3	121	76	41	0	0	51	1211	0	0 19
4681	1	0	0	0	0	0	0	0 2151	0	0

6.ábra

133-AS HALOTERVRE FORDITOTT IDO

SZAM	AGE	ESZ	MAR	KSS	HFR	GRA	LAK	MUSZIEMU	NYAK	KOUP
133	2	0	0	0	0	0	0	0 317	0	0

7.ábra

****
* MTA-SZTAKI *
****

MAI DATUM: 1975. 6. 5.

LISTA AZ ANYAGBESZERZES SZAMARA

FELHIVJUK AZ ANYAGBESZERZES FIGYELMET, HOGY AZ ALABBI IGENYLESI SZAMOK KERT HATARIDEJE LEJART:

IGENYLESEK KESZHEJELENTESE 1

TASAKSZAM:	IGENYLESI SZAM:	BEJOTT?
500	10	
473	34	
550	39	

8.ábra

MŰVELETEK

Azonosító	Darb. sz.	Kezdési dátum	Művelet leírása					
			Mhely kód	Művelet idő	Feltétel k.	igényl. kapcs.	Fő	Kezdés napokban
625001	1	-	AGE	10	0	0	0	6

FELTÉTELEK

Azonosító	Műv. szám	Feltételek azonositói és müveleti száma									
		azonosito	m. sz.	azonosító	m. sz.	azonosító	m. sz.	azonosító	m. sz.	azonosító	m. sz.
002	02	001	01	003	01						

IGÉNYLÉSEK

Azonosító	Műv. szám	Igényl. hat.idő	Igényl. szám	Igényl. szám	Igényl. szám	Igényl. szám	Igényl. szám
003	01	5.05.1[illegible]	174	-	-	-	-

DRB.JEGYZÉK

A dokumentációban szereplő darabjegyzék kerül változtatás nélkül lyukasztásra.

4.ábra

MUNKAIDŐ ELSZÁMOLÁS [illegible] DÁTUM:1975.6.3.

HOMOGÉN MUNKAHELYEK	MUNKA[illegible]
ANYAGELŐKÉSZITÉS	69
ESZTERGÁLYOS MŰHELY	80
MARÓS MŰHELY	48
KÜSZÖRÜSÖK	4
HELYZETFURÓS	8
GRAVIROZÓ	0
LAKATOS MŰHELY	86
MŰSZERÉSZ MŰHELY	69
ELEKTROMŰSZERÉSZEK	217
NYÁK LABOR	0

133-AS HÁLÓTERV TÖRÖLVE

5.ábra

MTA SZTAKI

MUNKAPROGRAM: MECH. MŰSZERÉSZNEK 1 NAPRA
DATUM: 1975.06.03. 8. LAP

ALKATR AZONO-SITO	MEGNEVEZÉS RAJZSZÁM ANYAGMINŐSÉG V TIPUS DARABOLÁSI MERET ELOK DB	INDU LO DB	KOV MUV H	MUV H	RÁFORD IDO JO DB	MUV IDO	FO	FELTE-TEL ALK;AZ	TAR TAL IDO	K A T	KOV MUV DB
595003	FEDOLEMEZ FELSO 312-22-2	4	SZ	MUS		10	1		4	9	1
493065	FUL 410-016-2 M2H LM2X 100X100	6	KKD	MUS		6	1		4	8	2
600008	ELLENALLASTARTO 409-011-1 TEXTILBAKELIT LM2X100X100	3	SZ	MUS		3	1		4	0	1

9.ábra

MTA SZTAKI

TERMELÉSI TABLA TASAKONKENT 1975.06.03..
KOSZORUSNEK 1.LAP

A-SAK	DAT	KEZ DETT	6.3 -IG	6.3 -TOL	6.9 -TOL	6.16 -TOL	6.23 -TOL	6.30 -TOL	7.7 -TOL	7.14 -TOL	7.21 -TOL	7.28 -TOL	8.04 -TOL	8.11 -TOL	8.18 -TOL
7o	7.31								20	28					
ÖSSZESEN		0	0	0	0	0	0	0	0	28					
GONGYOLT		0	0	0	0	0	0	0	20	48	48	48	48	48	48

10.ábra

************** KOOPERACIOS MEGRENDELES: AL.ELOX BARMELY SZIN
* MTA SZTAKI *
************* DATUM: 1975.06.08. 1. LAP

ALKATR. AZONO- SITO	MEGNEVEZES	RAJZSZAM	INDU LO DB	KOV MUV M	MUV IDO NAP	TAR IDO	MUV H	BEERKEZETT DARAB
554013	ANYA	PA-8	1		6	-1	KKA	
	0 0 ALMGSI							
	AT25X25							
562001	FENYMERO CSO	150-489	1		6	-17	KKA	
	0 0 ALMGSI							
	AT50X200							
562005	KUPAK	150-493	1		6	-17	KKA	
	0 0 AIMGSI							
	AT50X15							
562004	TARTO CSO	150-492	1		6	-20	KKA	
	0 0 AI99,5							
	LM1X150X100							
568003	KUPAK	150-496	1		6	2	KKA	
	0 0 ALMGSI							
	AT60X20							
568004	HAZ	150-497	1		6	2	KKA	
	0 0 ALMGSI							
	AT60X210							
568007	TARTO	150-500	1		6	2	KKA	
	0 0 AL99,5							
	LM2X100X200							
575010	HUTOBORDA	410-15,54-2	1	MA	6	-9	KKA	
	0 0 1772 HUTOBORDA							
	32X59,5X130							

11.ábra

Irodalomjegyzék

1 BISAD Vállalati információ rendszerek elemzése és tervezése. Statisztikai Kiadó Vállalat Budapest, 1976.

2 Disc Operating System /Date General Corporation 093-000048-06/ Southboro 1971.

3 File szervezési eljárások. SzÁMOK, Budapest, 1970.

4 FORTRAN reference manuel /Data General Corporation 093-000053-04/ Southboro, 1971.

5 File Directory Check Program /Data General Corporation 093-000071-00/ Southboro, 1972.

6 Hálótervezési módszerek, KSH Országos Ügyvitelgépesitési Felügyelet. Budapest, 1968.

7 Kapcsolt indexü szekvenciális hozzáférésü file kezelő rendszer. MTA Számitástechnikai Központja, Budapest, 1972.

8 Dr.Nagykálnai Endre: A vállalati rendszerszervezés gyakorlata. Statisztikai Kiadó Vállalat, Budapest, 1973.

HÁZGYÁRI TERMELÉSIRÁNYITÁSI RENDSZER ADATBÁZISA

/Egy konkrét megvalósult alkalmazás tapasztalatai/

Szécsi Károly és Kaszás Tibor
FÜTI Programfejlesztési Osztály

1./ A PROBLÉMA FELVETŐDIK:

Egy házépitőkombinát /HK/ azzal az igénnyel kereste meg Irodánkat, hogy ajánljunk számitógépes megoldást a házgyári adminisztráció megsegitésére.

A megkeresés előzménye egyrészt a FÜTI és a HK közötti hagyományos adatfeldolgozási kaposolat volt, másrészt a megrendelőnek a tipizálási törekvése, amely egy teljes termék-katalógus összeállitását eredményezte.
A termék-katalógus egészen a szekció tipusokig /egy léposőházhoz kaposolódó épületrész/ tartalmazza valamennyi termék konszignációját. Az igy keletkezett adathalmaz manuális kezelése viszont már keresztülvihetetlen.

2./ A TÉMÁHOZ KAPCSOLÓDÓ FOGALMAK:

A Házgyár telepitett épitőipari üzem, funkciója az, hogy az épitőipari vállalat előregyártott épület-összeszerelő részlegét meghatározott összetételben különböző térelemekkel, panelekkel és gépészeti szerelvényekkel lássa el.
A kivitelezési folyamatot a megrendelés inditja meg /minden megrendelés kap egy külön munkaszámot/. Tipizált szekciók esetén a megrendelő csak azt közli, hogy milyen tipus szekcióból, milyen

mennyiséget óhajt a megrendelt épületbe elhelyeztetni.
A munkaszám időszakosan változó adat.
A termék-katalógus legmagasabb szervezettségü eleme a tipus szekció, összetétele állandó. A tipus szekciók a katalógus szerint házgyári késztermékekből állnak /térelemekből, panelekből, gépészeti szerelvényekből, stb./
A házgyári késztermékek - hasonlóan a tipus szekciókhoz - félkésztermékekből állnak, a félkésztermékek, további félkésztermékekből és/vagy alapanyagokból.
Tehát a munkaszámok, szekció tipusok, házgyári késztermékek, házgyári félkésztermékek és alapanyagok egymáshoz való viszonya hierarchikus. A felsorolt fogalmakra a továbbiakban az "elem" gyüjtőfogalommal is fogunk hivatkozni.
Egy elem - a munkaszámok és alapanyagok kivételével - kétféle minőségben is megjelenik a hierarchiában; mint befogadó /apa/, és mint beépülő /fiú/.
A munkaszámok csak befogadóként jelennek meg, és a hierarchia legmagasabb szintjét jelentik. Az alapanyagok csak beépülőként jelennek meg, és a hierarchia legalacsonyabb szintjét képviselik.
A hierarchiából elvileg kiemelve egy apát és a fiait egy családnak nevezzük. Egy család annyi összefüggést /befogadási - beépülési kapcsolat/ jelent, ahány fiú van a családban.
A hierarchiával kapcsolatban lehet még szintekről beszélni: egy szintet alkotnak azok az elemek, amelyeket azonos számú befogadási - beépülési kapcsolat választ el a hierarchia legmagasabb szintjétől a munkaszámoktól. Egy elem azonban több szinten is előfordulhat, az ilyen elemeket a legmélyebb beépülési szintjükkel célszerü megjelölni.

3./ A TERMELÉS IRÁNYITÁSI RENDSZERE:

A HK igénye az volt, hogy a megrendelések és a katalógus adatainak közlése után Irodánk a következő információkat szolgáltassa:

a./ meghatározott tervidőszakra a házgyár alapanyagszükségletének kimutatása,
b./ időszakos kész- és félkésztermék szükséglet üzemi szinten,
c./ kész- és félkésztermékek árképzése, költségtényezők szerinti bontásban.

A feladat megfogalmazása során a továbbfejlesztés igénye is felvetődött; az üzemi szintü programozás, a megrendelésekkel összekapcsolt anyaggazdálkodás, az üzemi irányitással összekapcsolt munkás nyilvántartás, a helyszini szerelések programozása, a panel-szállitás programozása stb., vagyis egy olyan komplex termelés tervező, programozó és kivitelezést ellenőrző rendszer, amelyet már termelésirányitásnak lehet tekinteni.
Ilyen rendszer bevezetésére azonban csak fokozatosan kerülhet sor. Tehát az első léposő kialakitásánál arra kellett törekedni, hogy az önmagában is hasznos legyen, terjessze el a számitógépes módszerek alkalmazása környezetében szükséges gondolkodásmódot, legyen hatékony, és kiterjeszthető a további termelésirányitási feladatok végrehajtása irányában.

4./ A KONKRÉT FELADAT, AZ ADATBÁZIS

A leirt megfontolások és igények /a.,b.,c./ eredményeképpen egy kezdeti adatbázis létesitését, és ehhez kapcsolódó visszanyerő programok elkészité-

sét irányoztuk elő első lépcsőben.

I. A feladat megoldásához képféle tipusú információ tárolására van szükség:

a./ az elem tipusú halmazba sorolhatók a munkaszámok, a szekciók, a házgyári szintü késztermékek /térelem, panel, gépészeti szerelvény stb./, félkésztermékek és alapanyagok,

b./ a beépülési-befogadási kapcsolat tipusú halmazba sorolhatók a munkaszám konszignáció, a szekció konszignáció és az összes többi termék konszignáció.

A tárolt információk változékonyságuk szempontjából két csoportba sorolhatók:

a./ rendszeresen, időszakonként változó adatok: a munkaszám és a munkaszám konszignáció,

b./ állandó jellegü, de a technológiai fejlesztés, változás következtében rendszertelenül változó adatok:
szekció tipusok és szekció konszignációk, termékek és termék konszignációk.

Az idő dimenziója - az első lépcső adatbázisában - csak a munkaszám elemben jelenik meg: kezdési és befejezési dátum formájában.

II. Az adatkezelést illetően a rendszernek rendkivül rugalmasnak kell lennie, ugyanis a várható adatváltozás negyedévenként 10-15 %-os. Az adatok egymáshoz való viszonyát a beépülési- befogadási kapcsolat jellemzi /forrása a konszignáció/, az adatszervezésnek ezt a hierarchikus, többszintü kapcsolódást kell tükröznie, vagyis például egy munkaszám ismeretében el kell érni a hozzátartozó tipus szekciókat és azok darab-

számát az illető munkaszámban, stb.

III. Az adathalmaz méretei:

A tájékozódás időszakában a következő becsléseket fogadtuk el az egyes adattipusokra vonatkozóan:

A./ "elem" adatok: - munkaszám: 100 db
- szekció típus: 30 db
- kész-, félkésztermék, alapanyag: 2 000 db

B./ befogadás-beépülés adatok:
- munkaszám-szekció: 600 db
- szekció-késztermék: 4 500 db
- késztermék-félkésztermék, félkésztermék alapanyag: 12 000 db.

5./ AZ IBM BOMP PROGRAMCSOMAG LEHETŐSÉGEI

I. A BOMP az IBM cég által készített adatkezelő rendszer, közvetlen hozzáférésü tárolókon elhelyezkedő adatállományok kezelésére, szervezésére, ahol az adatok kaposolata hierarchikus. Pontosabban olyan félkész programok halmaza, amelyekből a felhasználó az adott lehetőségek keretén belül kialakithatja saját file-kezelő programjait. A felhasználó szabványositott paraméterkártyákon irja le az általa tervezett adatbank file-jainak számát és jellegét, a file-ok közötti összefüggéseket, és az egyes file-okban tervezett rekordképet, stb.
A paraméterkártyákat ezután a felhasználó beolvastatja a BOMP vezérlő-leszabó programjával, amely a félkész progrmok utasitás halmazából, mint input adathalmazból leszabja, kialakitja

a paraméterkártyák alapján meghatározott konkrét programok utasitás sorozatait.

A BOMP háromféle típusú adat file szervezését és kezelését biztositja:

a./ A törzsadat file-ok, amelyek az elemek valamilyen kulcs szerinti tárolására alkalmasak. /Ha egy gráfhoz hasonlitanánk a teljes adathalmazt, akkor az elemek lennének a gráfok csúcsai./

b./ Az összefüggés /pl. beépülés-befogadási összefüggés/ adat file-ok vagy más megfogalmazásban a láno-file-ok, amelyek a fenti elemek közötti kapcsolatok tárolására alkalmasak. /A gráf hasonlatban a gráf élei./

c./ Az alárendelt törzsadat file-ok, amelyek az elemekhez opcionálisan kapcsolódó bővebb információ tárolására alkalmasak. /A gráf hasonlatban azt lehetne mondani, hogy ezek az adatok a csúcsok "árnyékai". Nem szükséges okvetlenül, hogy minden csúcsnak legyen "árynéka"./

A törzsfile-ok elsődlegesek az összefüggésfile-okkal szemben. Két törzsfile csak egy összefüggésfile-on keresztül teremthet kapcsolatot. Minden alárendelt törzsfile egy és csak egy törzsfile-hoz kapcsolódhat. A file-ok darabszámát csak az korlátozza, hogy összesen nem lehet több mint húsz file.

/Legyen egy konkrét példa a következő:

Egy törzsállomány a hallgatók törzsadatait tartalmazza, egy másik törzsállomány a tanszékek adatait, ezt a két törzsállományt összekapcsolja egy össze-

függés állomány, amely azt mutatja, hogy melyik hallgató melyik tanszéken, vagy tanszékeken készíti a diplomatervét és mi a beadási határidő.

Egy alárendelt törzsfile pedig minden hallgatóhoz kapcsolódóan tartalmazza a teljes feladat kiirást./

II. A BOMP kezelése alatt létrehozott és karbantartott file-ok egyéb irása és olvasása is csak BOMP rutinok használatával oldható meg.

Az adatok közötti logikai összefüggések tárolását cim-láncok valósitják meg. Minden BOMP fileban elhelyezett rekordnak 4 byte-os cime van, amely a hagyományos 9 byte-os cim információit tartalmazza süritett formában. A cim-lánc úgy keletkezik, hogy egy adott rekord, adott mezőjében /amely az adott cim-lánc részére van kijelölve/, az adott logikai sorban /láncban/, az ezután a rekord után következő rekord 4 byte-os cime van. A cim-lánc végét egy "END." bejegyzés jelzi. A cim-lánc minden egyes eleme egy rekord és egy következő rekord cime.
A cim-láncok következő tipusai lehetségesek:

a./ egy törzsfile-on belül a kulcs szerinti logikailag egymást követő rekordokat összekötő cim-lánc, a legkisebb kulcsútól indulóan a legnagyobb kulcsig,

b./ egy törzsrekordból kiinduló lánc, amely egy összefüggés állományba megy át, és ott azokat a rekordokat kapcsolja sorba, amelyek a kiinduló rekordra vonatkozó információt tartalmaznak,

c./ a törzsfile-okon belül minden egyes rekordból kiindulóan, az első szabad hely cime a törzsfile-hoz való rekord hozzáadás részére, ha a hozzáadandó rekord kulosa éppen az egyes rekordé után következik a logikai sorban.

A cimláncok minden változtatásnál felhasználói beavatkozás nélkül aktuális állapotra állitódnak. A láncfile-ok közvetlenül nem érhetők el, csak valamelyik törzsfile egy rekordjából kiindulva. A láncfile-ok rekordjai közvetlenül csak az általunk összekapcsolt két törzsrekord kapcsolatára jellemző adatokat /pl.: beépülő mennyiség/ tartalmazzák. A további információ közvetett, és a láncrekordbeli cimekben van.

III. A BOMP adatszervezése mágneslemezes adattárolást feltételez.

A. A törzsfile-ok szervezése módositott indexszekvenciális szervezés. Kulcs használata kötelező, a rekordhossz fix, az állomány blokkolt, a rekord sorrend elsődlegesen kulcs növekvő. Az index háromszintü. Az általános túlcsordulási terület az elsődleges adatterület végétől visszafelé jelölődik ki. Uj rekord közbeszúrásnál csak a cimlánc aktualizálódik. /az a. és c. tipusú cimlánc/

B. A láncfile-ok közvetlen elérésüek, kulcs itt nincsen értelmezve. Hozzáférési szempont a láncrekord cime. A rekordhossz fix, az állomány blokkolt. Az adatterület töltése az összes cilinder felhasználásával fokozatosan történik.

A BOMP file-ok rekord felépitése speciális: a felhasználói adatmezőket megelőzi egy "prefix", ahol a lánc-cim mezők helyezkednek el és egyéb BOMP információk.

6./ AZ R-20-AS GÉP

A fent leirtak szerint adott volt a feladat és a software eszköz, ehhez járult hardware elemként a a FÜTI R-20-as gépe.

A konfiguráció -

128 K Byte Központi tár, 5 db 7,5 M Byte kapacitású lemez, 4 db szalagegység, 2 db kártyaolvasó és 2 db sornyomtató kielégitette az adathalmaz és a BOMP támasztotta követelményeket. Az alkalmazott DOS operációs rendszer támogatja a BOMP adatkezelő rendszert.

7./ A KONKRÉT SZERVEZÉS

A rendszerterveket a FÜTI Rendszerszervezési osztálya készitette a HK szervezőivel szoros együttmüködésben.

A rendszerterv tartalma a következő:

A feladat ismertetése.

1. Az alrendszer adatállományai:
   11. Az alrendszer törzsállománya
   12. Az alrendszer láncállománya.

2. Az alrendszer bizonylatai:
   21. Törzsadat bizonylat
   22. Beépülési kapcsolat bizonylat
   23. Szekció konszignációs bizonylat

24. Fuvardij bizonylat
25. Normaóra bizonylat
26. Építési bizonylat /munkaszám konszignáció/

3. Az alrendszer táblái:
   31. E /ellenőrző/ táblák:
      311. 1E Törzsadatállomány ellenőrző tábla,
      312. 2 E Láncállomány ellenőrző tábla,
   32. L /lista/ táblák:
      321. 1L Termékek és alapanyagok felsorolása cikkszám szerinti sorrendben,
      322. 2 L Termékekbe egyszintű beépülések felsorolása,
      323. 3L Termékekbe valamennyi szinten beépülők felsorolása,
      324. 4L Szekciókba beépülő késztermékek felsorolása,
   33. T /terv/ táblák:
      331. 1T Üzemi késztermékek konszignációja és éves termelési terv,
      332. 2T Üzemenkénti félkésztermék konszignáció és éves termelési terv,
      333. 3T Üzemenkénti késztermék termelési terv épületenkénti bontásban,
   34. A /alapanyagszükségleti/ tábla;
   35. ÁK /árkalkulációs/ táblák:
      351. 1ÁK Késztermékek árkalkulációs táblája,
      352. 2ÁK Üzemi késztermékek árkalkulációs táblája,
      353. 2ÁK A késztermékek anyagszükségletének részletezése,
      354. 2ÁK Az üzemi kész- és félkésztermékek anyagköltségeinek részletezése,
      355. 3ÁK Kalkulációs tábla magyarázó szintek,

356. 4ÁK Szekcióba beépülő termékek árkalkulációja,

4. A mellékletek:

41. Kártyatervek,

42. Rekordtervek,

43. Mozgásszám táblázat,

44. Lyukasztási utasitás,

45. Blokkdiagramok.

8./ AZ ADAPTÁLT BOMP RENDSZER

A. A kiépitett adatbázis a következő állományokat tartalmazza:

1./ A törzsállomány /Part Number Master File/, amely a házgyári termelés körében szereplő valamennyi munkaszámra, szekció tipusra, késztermékre /térelem, panel, szerelvény stb./, félkésztermékre és alapanyagra vonatkozóan tartalmaz egy-egy rekordot.
A rekordok kulcsa a munkaszám, a szekció azonosító kód, illetve a cikkszám. A file jelenleg kb. 4000 rekordot tartalmaz.

2./ A láncállomány /Product Structure Chain File/, amely minden beépülési - befogadási kapcsolatra - amely a törzs két-két rekordja között fennáll - tartalmaz egy-egy rekordot. Részletezve:

a./ minden egyes munkaszámra vonatkozóan annyi rekordot, ahány különböző tipusú szekció a munkaszámba beépül,

b./ minden alapanyagra vonatkozóan annyi rekordot, ahány különböző kész- és félkésztermék, stb. az illető alapanyagot befogadja,

c./ és a munkaszámok és alapanyagok kivételével az összes többi törzsadatra vonatkozóan annyi rekordot, ahány tényleges beépülési-befogadási kapcsolat létezik közöttük.

A rekordoknak nincsen kulcsuk. A file jelenleg kb. 10.000 rekordot tartalmaz.

B. Az egyes állományok rekord felépítése a következő:

1./ Törzs logikai rekord /PN/ felépítése:

a./ lánc-cím a logikai hozzáadáshoz /PN-ben/,
b./ PN kulcs /cikkszám/szerk.azonosító/munkaszám/,
c./ az első befogadásra vonatkozó láncrekord címe /PS-ben/,
d./ a befogadott elemek száma,
e./ az első beépülésre vonatkozó láncrekord címe /PS-ben/,
f./ a beépülések száma,
g./ a legalacsonyabb beépülés szintje,
h./ a logikailag /kulcs szerint/ következő PN rekord címe,
i./ a logikailag következő rekord kulcsának ellenőrző része,
j./ használati aktivitás számláló,
k./ felhasználói adatmezők.

2./ Lánc logikai rekord /PS/ felépítése:

a./ a beépülő törzsrekord címe a PN-ben,
b./ a beépülő törzsrekord kulcsának összehasonlító része,
c./ a következő befogadásra vonatkozó láncrekord címe a PS-ben,

d./ a befogadó törzsrekord címe a PN-ben,
e./ a befogadó törzsrekord kulcsának összehasonlító része,
f./ a következő beépülésre vonatkozó láncrekord címe a PS-ben,
g./ felhasználói adatmező: egységnyi befogadóra vonatkozó mennyiség a beépülőből.

C. Az adatbázis létrehozásához és használatához a következő programokat használjuk:
1./ BOMP programok:
- vezérlő-funkció választó,
- törzsfile kreáló,
- láncfile kreáló,
- általános karbantartó,
- visszanyerő programok.
2./ Felhasználói programok:
- adat előkészitő,
- lista készitő,
- számitást végző programok.

./ AZ ÁRKÉPZÉSI ALRENDSZER

Az árképzés a BOMP rendszer lehetőségét - a hierarchikus adattárolást - használja fel, de annak nem szerves része. Az árképzési rendszer az adatbázis felhasználásával a FÜTI önálló szervezése és megvalósitása.

Az árképzés fázisai a következők voltak:
1./ Az alapanyagok árát a FÜTI Cikklista Törzsállományból átvettük.
2./ Az alapanyagok cikkcsoportjaira a HK fuvardijat adott fel.
3./ A kész- és félkészterméekekre ugyancsak a HK bérköltséget adott fel.

4./ Ezután az érképzés szintről-szintre először csak alapanyagokat befogadó cikkek árának kiszámításával, stb. - elvégezhető. Megőrzésre került törzsrekordonként a közvetlen anyagköltség és a kibocsátási ár.

10./ BOMP ADAPTÁLÁSI TAPASZTALATOK

A BOMP rendszer lehetőségei lényegében lefedték az adott problémát. A rendszer adaptálása a kézikönyv alapján nagyobb zökkenők nélkül megoldható volt. A rendszer az előirányzott funkciókat tökéletesen végrehajtja.
A kialakitott adatbázis felhasználói programokkal is könnyen használható, ha a programozó ismeri és betartja a hozzáférési szabályokat.

11./ ÉRTÉKELÉS

A rendszer kifejlesztése beváltotta a hozzáfüzött reményeket. Az adathalmaz karbantartása lényegesen egyszerübb és gyorsabb, mint azt felhasználói programokkal meg lehetett volna oldani. Hasonlóan az adathozzáférés is kevesebb időt igényel, mint a hagyományosan szervezett adatfeldolgozásokban.
A BOMP rendszer hatékonyságának lemérése kontrolrendszer hiányában még nem történt meg. A FÜTI azonban tervbevette, hogy az Épitőipari Költségvetéseket készitő rendszer adatbázisát is átszervezzük a BOMP rendszerbe, ennek kapcsán lesz alkalom hatékonyság vizsgálatra.
A kifejlesztett adatbázis termelésirányitási kérdések megválaszolására alkalmas, a továbbfejlesztés alapjául szolgálhat.

A teljes termelésirányitási információs rendszer kifejlesztéséhez és bevezetéséhez azonban az szükséges, hogy a számitástechnikában elterjedjenek azok a megoldások, amelyek az információs rendszer adatállományának azonnali aktualizálását teszik lehetővé, és minden vezetői szinten fejlődjék a számitástechnikai ismeretek és a számitástechnika alkalmazása iránti igény.
A rendszer alkalmazása már megkezdődött.
A rendszer számos olyan információt szolgáltat, amely hagyományos módszerrel egyáltalán nem, vagy csak rendkivüli nagy erőfeszitéssel állitható elő.
A rendszer információinak célszerü felhasználása szervezettebbé és eredményesebbé teheti a jelenlegi házgyári termelésirányitási folyamatot.

Megkezdődött az erőforrások felhasználását tervező alrendszer készitése, amely bizonyos szempontok szerinti optimális felhasználási programokat fog készitení.

Budapest, 1975. június 10.

SYDAC

/System for Data Conversion/

Királyné, Török Éva - Kőváriné, Lábady Marianna
MTA Központi Fizikai Kutató Intézet

BEVEZETÉS

A KFKI-ban 1969 óta foglalkozunk a kisszámitógépes mágnesszalag technika alkalmazásához szükséges software-rendszerek fejlesztésével. Hazai és külföldi konferenciákon ennek a munkának több fázisáról beszámoltunk már.

A harmadik generációs hardware-technika a kisszámitógépek nagyobb mértékü elterjedését, szélesebb körü alkalmazását tette lehetővé, és ezáltal fokozott mértékben vetette fel uj software-rendszerek fejlesztésének kérdéseit.

Ma már nem szokatlan például az az igény, hogy az adatfeldolgozás területén, a nagyszámitógépes rendszerek mellett, kisszámitógépek is képesek legyenek önálló feladatmegoldásra. - A kisszámitógépes adatfeldolgozás racionalitását a feldolgozandó adattömegek növekedése, a szocialista országok mágnesszalag-periféria gyártásának /ESzR/ fejlődése, a viszonylag alacsony beruházási költségek, valamint a kisszámitógépeken elérhető magasszintü nyelvek és fejlett operációs rendszerek bizonyitják. A TPA-1001/i kisszámitógépre kidolgozott adatfeldolgozó rendszerek, a kisszámitógép lehetőségeit meghaladó volumenü feladatok esetén, hatékony együttmüködést biztositanak a nagyszámitógépes konfigurációkkal.

Ebben az előadásban ismertetett programrendszer elsősorban a lassu perifériáknak kigéprendszerben való müködtetésével javitja a nagygép hatásfokát /nagy adatmennyiségü ügyviteli feldolgozások során a lassu periféria igényes feladatok akadályozzák leginkább a központi egység teljesitőképességének kihasználását/.

A kapcsolatot a gyors feldolgozásra alkalmas mágnesszalagos adathordozó biztositja.

PROGRAMRENDSZER RENDELTETÉSE ÉS TULAJDONSÁGAI

Mivel hazánkban gomba módra sokasodnak az ESzR konfigurációk, közelfekvőnek látszott a SYDAC programrendszert olyan mágnesszalagok feldolgozásával, illetve előkészitésével "megbizni", melyeken az információ strukturája ESzR standard-nek felel meg.

Az ESzR-IBM kompatibilitás miatt azonban a SYDAC az IBM konfigurációk off-line kiszolgálására is alkalmas.

A SYDAC megtervezésénél nagy sullyal vettük figyelembe:
- a/ a felhasználók rendszerint szükre szabott anyagi lehetőségeit,
- b/ a kisszámitógépes konfigurációk sokféleségét,
- c/ a későbbi fejlesztési igényeket,
- d/ a könnyen kezelhetőséget.

Ezért a programrendszer:
- a/ moduláris felépitésü,
- b/ általános I/O rendszerrel rendelkezik,
- c/ univerzális konvertáló egysége van,
- d/ konzolról vezérelhető.

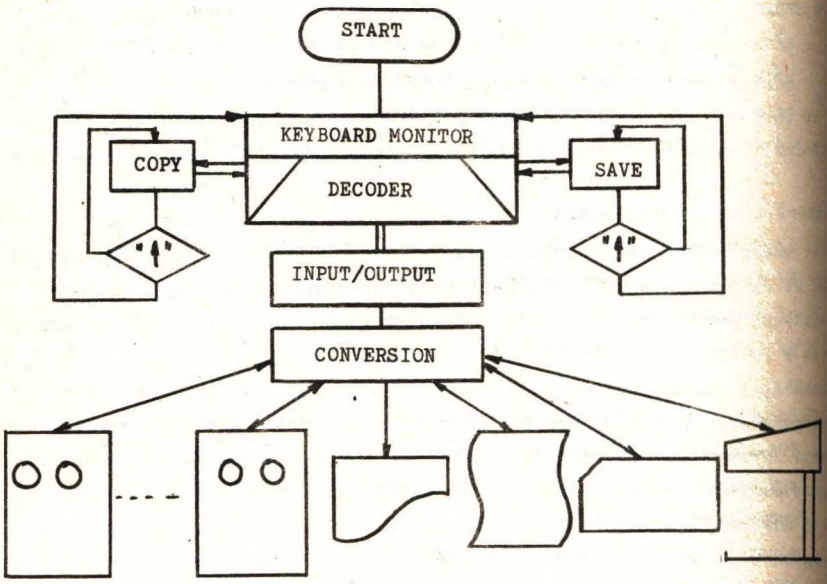
Működéséhez szükséges minimális hardware kiépitettség:
- TPA-1001/i 8K operativ memória
- egy mágnesszalag egység /9 csatornás/
- gyors szalagolvasó
- sornyomtató/gyors szalaglyukasztó
- ASR-33 teletype.

A fenti konfiguráció további mágnesszalag egységekkel /max. 8 db/ és lassu input/output /kártya, plotter stb/ perifériákkal az operativ memória növelése nélkül bővithető. Diszk és hétcsatornás mágnesszalag egységek illesztése azonban már memória bővitést is igényel /max.32K/.

ÁLTALÁNOS FELÉPITÉSE

A SYDAC programrendszert inicializáló KEYBOARD MONITOR - az operátorral folytatott kezdeti párbeszéd után - azonnal átadja a vezérlést a kiválasztott rendszerprogramnak.

A rendszerprogramok - COPY, SAVE - müködését a konzolról kijelölt I/O perifériák mellé felirható opciós karakterek és egyéb paraméterek /file szám, file-azonositó, kötet-azonositó stb/ specifikálják. Ezeknek az információknak az értelmezése, ellenőrzése és adminisztrálása a DECODER feladatai közé tartozik.



Az adattranszfert a KONVERTÁLÓ rendszer közbeiktatásával az INPUT/ /OUTPUT programcsomag végzi. A SYDAC mai változata DKOI-ASCII, illetőleg ASCII-DKOI kódkonverziót hajt végre, de a programrendszerben alkalmazott konvertálási eljárás tetszőleges kódkészletü periféria illesztését engedi meg a konfigurációba.

Az INPUT/OUTPUT programcsomag periféria-független programozási lehetőséget biztosit azáltal, hogy a perifériák fizikai és logikai kezeléséhez szükséges szubrutinokat egyaránt tartalmazza.

A SYDAC a következő logikai perifériákat használja:

nyitás
zárás
funkció /karakter input/output/

A logikai perifériákhoz rendelendő fizikai periféria neveket konzolról lehet megadni, a hozzárendelést az I/O programcsomag végzi.

A SYDAC periféria készlete:

Mn - mágnesszalag /n=a mágnesszalag egység száma/
L - sornyomtató
P - gyors szalaglyukasztó
R - gyors szalagolvasó
C - konzol

RENDSZERPROGRAMOK

A mágnesszalagok ESzR/IBM szabványa azontul, hogy cimke nélküli és cimkézett formátumot is megenged, fizikai és logikai egységek tekintetében különböző felépitésü file-ok kezelésére ad alkalmat.

Ezeket a lehetőségeket, a standard követelmények betartása mellett, mindkét rendszerprogram biztositja a felhasználók számára. A rendszerprogramok feladatai közé tartozik továbbá néhány speciális felhasználói igény kielégitése is.

COPY

Az ESzR/IBM operációs rendszerek programjai által mágnesszalagra mentett adatok, eredménylisták, programlisták vagy egyéb információk megjelenitését elsődleges periférián - a COPY rendszerprogram végzi.

A program inputja mindig mágnesszalag, outputja sornyomtató vagy opcionálisan más periféria lehet. A mágnesszalag jellegével és a másolás menetével kapcsolatos operációs kivánságok a perifériák kijelö-

lésével egyidejüleg, egy opciós listával adhatók meg.

A COPY rendszerprogram opciói

Input periféria után:

/L Jelöli, hogy cimkézett mágnesszalag kerül feldolgozásra.

/E Opcióval a mágnesszalag teljes egészének másolása kérhető.

Output periféria után:

/L Opciós karakter hatására az információs file-lal együtt a hozzátartozó első kezdőcimke is megjelenik az output periférián.

/C Opcióval közvetlenül a file-másolat elé maximálisan 70 karakteres felhasználói megjegyzés irható a konzolról.

Az opciós lista elfogadása és értelmezése után a Decoder az opciós karakterektől függően különböző kérdésekre vár választ. Ezek a kérdések a kötet-azonositóra, a file nevére vagy számára, a file rekordjainak tipusára, a file blokkjainak és rekordjainak hosszuságára vonatkoznak. A válaszok értelmezése után a Decoder visszaadja a vezérlést a rendszerprogramnak. A kijelölt akció végrehajtását a COPY vagy az "END OF TAPE" üzenettel /ha /E opció van/, vagy az ↑ karakter kigépelésével jelzi.

Az "END OF TAPE" után a rendszerprogram visszatér a Monitorba, lehetőséget adva ezzel a felhasználónak az ujabb feladat kijelölésére. A ↑ karakter a munka más természetü folytatására ad alkalmat. A program várakozó hurokba kerül, ahonnan a következő karakterkombinációk valamelyikének hatására lép ki:

CTRL/M Hatására az utolsóként kijelölt file-ról további másolat készül. /Valamely file-ról tetszőleges számu másolat állitható elő./

CTRL/N Karakterkombinációval a mágnesszalagon fizikailag az éppen feldolgozott file után lévő, bármely más file-ról kérhető másolat.

CTRL/C Segitségével a vezérlés visszadaható a Monitornak.

SAVE

A SAVE rendszerprogram elsődleges adathordozók adatait konvertálja és írja át mágnesszalagra, ESzR/IBM standard szerinti cimke nélküli vagy cimkézett strukturával.

A program inputja gyors szalagolvasó vagy opcionálisan más periféria, outputja mindig csak mágnesszalag lehet. Az I/O periféria megjelölésén kivül egy feladat specifikálása opciós karakterek és a Decoder kérdéseire adott válaszok segitségével lehetséges.

A Copy rendszerprogramnál leirtakkal ellentétben a SAVE program kizárólag csak output perifériát követő opciós lista karaktereit fogadja el.

A SAVE rendszerprogram opciói

Output periféria után:

/L Opciós karakter cimkézett mágnesszalag szerkesztését jelöli ki, és a file első kezdőcimkéje az input perifériáról kell, hogy a memóriába kerüljön.

/C Ugyancsak cimkézett mágnesszalag formátumot jelöl, de ebben az esetben a file első kezdőcimkéjének szerkesztését maga a program végzi a konzolról kapott paraméterekből.

/N Opcióval kérheti a felhasználó, hogy az adatrögzités a mágnesszalagra annak fizikai kezdetétől /BOT/ történjék. Egyébként a soron következő file az előzőleg már felirt információ megőrzésével kerül a mágnesszalagra.

A perifériák és opciós lista elfogadása után a Decoder ugyanugy, mint a Copy-nál, további paraméterek után "kérdez". A válaszok értelmezése után a vezérlés automatikusan visszaadódik a rendszerprogramnak, hogy a kijelölt feladatot elvégezze.

Az akció sikeres befejezését a program ↑ karakter kinyomtatásával jelzi. Ezután a felhasználónak módja van a mágnesszalag szerkesztést folytatni, lezárni vagy visszatérni a Monitorba.

CTRL/N Hatására a SAVE program felkészül a következő adatfile fogadására.
Az ujabb file "folytatólagosan" kerül a mágnesszalagra.

CTRL/E Karakterkombinációval kérheti a felhasználó a kötet lezárását /kötet végcimkét ir a program/.

CTRL/C Hatására a vezérlés visszaadódik a Monitornak. /Információ lezárás történik./

A felhasználó munkáját számos hiba és tájékoztató, illetve irányitó jellegü üzenet segiti mindkét rendszerprogramban.

A SYDAC programrendszer, moduláris felépitésénél fogva, tetszőleges bővitési-fejlesztési lehetőséget biztosit. További perifériák fizikai handler-einek, valamint más célu rendszerprogramoknak a beépitésével az off-line funkciók egyre szélesebb körét ellátó, processzor-vezérelt konverter állomások létrehozása válik lehetővé.

FORTRAN MATEMATIKAI PROGRAMKÖNYVTÁR FEJLESZTÉSE

Ivanyos Lajos, dr. Sima Dezső, Utassy Sándor
KKVMF Számitástechnikai Tanszék

CÉLKITÜZÉS

A FORTRAN nyelvü matematikai programkönyvtár kidolgozásával a gyakorlatban legtöbbször előforduló általános matematikai jellegü problémák megoldásához kivántunk segitséget nyujtani a VT 1010 B számitógépek felhasználóinak.

Jelenleg az alábbi témakörökben állnak rendelkezésre könyvtári rutinok:

- Lineáris egyenletrendszerek megoldása
- Vektor- és mátrixmüveletek
- Polinommüveletek, gyökmeghatározás
- Interpoláció, approximáció
- Numerikus integrálás
- Differenciálegyenletek és egyenletrendszerek megoldása

A rutinok a lényegében FORTRAN II szintnek megfelelő BASIC FORTRAN 1010 B gépi reprezentációban készültek, de néhány kisebb módositással minden FORTRAN reprezentációban, igy az R10-en is futtathatók.

A szubrutinok mellett kidolgozásra kerültek a szubrutinokat kezelő /tesztelő/ főprogramok, és igy a könyvtári rutinok alkalmazásával nem csak a szükséges matematikai

és programozási ráfordítás csökken jelentős mértékben, hanem minimális programozási ismeretekkel is lehetőség van viszonylag bonyolult feladatok megoldására.

SZEMPONTOK

Társzükséglet

A feltételezett, minimálisan szükséges 16 K-s központi egység és teletype konfiguráció esetén a felhasználó rendelkezésére álló tárterület mintegy 8 Kbyte. Ez a tény egyrészt a könyvtári rutinok hosszára adott 5-6 Kbyte-os korlátot, másrészt felhivta a figyelmet a tárterülettel való gondos gazdálkodásra.

A fentiek alapján az algoritmusok realizálásánál a
- kis futási idő,
- kis helyszükséglet
követelmények közül a társszükséglet minimalizálását tekintettük elsődleges szempontnak.

Egyszerü kezelhetőség

A programkönyvtárt ugy alakitottuk ki, hogy az a feltételezett minimál konfiguráció esetén adódó lyukszalagrezidens formában is jól kezelhető legyen. A szubrutinok egymásra épülése esetén a direkt hivás helyett általában előnyben részesitettük a kérdéses rutin beépitését.

Dimenzionálás

A VT 1010 B BASIC FORTRAN reprezentáció nem tesz lehetővé dinamikus tömbméret megadást, ezért a rutinokban kezelt tömbök méreteit rögziteni kellett. A dimenziók rögzitésénél több szempontot kellett figyelembe venni:

- a felhasználó szemszögéből kivánatos lenne minél nagyobb dimenziók megadása,
- növekvő tömbméreteknél jelentősen növekszik a társzükséglet,
- az elvégzendő müveletek számának lineáris vagy kvadratikus növekedése miatt jelentősen megnő a futási idő,
- a felhalmozódó kerekitési hibák miatt számábrázolási pontosság és tartomány is korlátot jelent a dimenziók növelésénél.

A fenti meggondolások alapján a rutinok maximálisan 10x10-es dimenzióra készültek.

Változatok

Számos feladatnál a módszer pontosságával és a szubrutin futási idejével kapcsolatos problémák csak ugy voltak megoldhatók, ha két különböző algoritmus alapján készült szubrutin: az egyik kevésbé pontos, vagy általános, de gyorsabb, a másik nagyobb pontosságu, vagy szélesebb feladatkörre alkalmas, de lassubb. Sok esetben nem a módszerek alkalmazhatósági korlátaiba, hanem az adott FORTRAN reprezentáció szabta korlátokba ütköztünk, amikor pontosabb eredmények elérését tüztük célul. Ez a korlát például magasabb foku polinomok gyökeinek meghatározásánál a számábrázolási tartomány, többszörös gyökök meghatározásánál a számábrázolási pontosság volt.

Megjegyezzük, hogy az R10 FORTRAN reprezentációban a kivánt pontosság eléréséhez szinte elkerülhetetlen a dupla pontosságu ábrázolás.

EREDMÉNYEK

A fejlesztés során az alábbi szubrutinok kerültek kidolgozásra:

SGEL	Lineáris egyenletrendszer megoldás a Gauss eliminációval
SINVS	Lineáris egyenletrendszer megoldás a simplex módszert alkalmazó mátrix invertálással
VCOMM	Több vektor lineáris kombinációja
MINVS	Mátrix invertálás simplex eljárással
MINVGJ	Mátrix invertálás Gauss-Jordan eljárással
PDIV	Polinom osztás
PDIVQ	Maximálisan másodfoku gyök leválasztása
PVAL	Polinom helyettesitési értéke valós, vagy komplex helyen
PVALD	Polinom és deriváltak helyettesitési értéke valós helyen
PCOEF	Polinom együtthatók meghatározása valós vagy konjugált komplex gyökökből
RREG	Függvényszegmensként adott egyváltozós függvény valós gyökei
RQUAD	Másodfoku egyenlet gyökei
RNR	Polinom gyökeinek meghatározása a Newton-Raphson módszerrel
RNRE	Polinom gyökeinek meghatározása a Newton-Raphson módszerrel, dinamikus normálással
RBA	Polinom gyökeinek meghatározása a Bairstowe módszerrel
ECP	Mátrix sajátértékeinek meghatározása a Danyiljevszkij-módszerrel

IPNE	Ekvidisztans alappontokra illeszkedő interpolációs polinom együtthatóinak meghatározása
IPGEN	Tetszőlegesen megadott alappontokra illeszkedő interpolációs polinom együtthatóinak meghatározása
ALQPN	Sulyozott pontsorozat approximálása korlátozott fokszámu polinommal
AFFT	Periódikus függvény gyors Fourier vagy inverz transzformációja
ALQLN	Sulyozott pontsorozat approximálása logaritmus függvénnyel
ALQEX	Sulyozott pontsorozat approximálása exponenciális függvénnyel
GRAPR	Egy vagy két függvény értékeinek kinyomtatása és rajzoltatása nyomtatón
INTRI	Ekvidisztans alappontokban adott függvény határozott integráljának kiszámitása Richardson finomitással
INTRO	Ekvidisztans alappontokban adott függvény határozott integráljának kiszámitása Romberg módszerrel
DIERK	Függvényszegmensként adott elsőrendü differenciálegyenletek megoldása Runge-Kutta módszerrel
DIEHA	Állandó együtthatóju lineáris elsőrendü differenciálegyenlet megoldása Hamming módszerrel
DIHOR	Állandó együtthatóju lineáris magasabbrendü differenciálegyenlet megoldása Runge-Kutta módszerrel

DISYS Elsőrendü differenciálegyenlet-rendszer megoldása Hamming módszerrel

A szubrutinok és főprogramok kidolgozásában a KKVMF Számitástechnikai Tanszékének munkatársai vettek részt.

VT-1010 B MÁGNESLEMEZES RENDEZŐ MODUL

Bartos György

Volán Tröszt Elektronika

A VOLÁN TRÖSZT ELEKTRONIKA VT-1010 B számitógépét, amely Székesfehérváron a VOLÁN 14.sz. Vállalatnál van felállitva, VIDOS-II rendszerben programozzuk.
A gyártól kapott mágnesszalagos rendező program csak a francia MONITOR rendszerben használható hatékonyan, ezért tartottuk szükségesnek egy VIDOS-II kompatibilis, gyors /mágneslemezes/ rendező programot készitени.

A rendező modul jellemzői:

- lehetőséget nyujt a felhasználó részére olyan programok irására, amelyek egy rendkivül gyors adatrendezést is magukba foglalnak,
- VIDOS-II kompatibilis,
- max. 10 db. rendezési kulcs, növekvő vagy csökkenő irányban,
- változó hosszuságu rekordokat kezel /lehetőséget ad az adattömöritésre/,
- min. 1, max. 4 mágneslemez használható /rendszerlemez is lehet/,
- felhasználói programba könnyen beilleszthető,
- a modul hossza 10 lap /Overlay szerkezetü/ a minimum szükséges munkaterület nagysága függ a maximális rekordhosszuságtól.

$$\text{min munkaterület} > \left(\frac{\text{max rekordhossz}}{256} + 2 \right) \cdot 1024 \text{ byte}$$

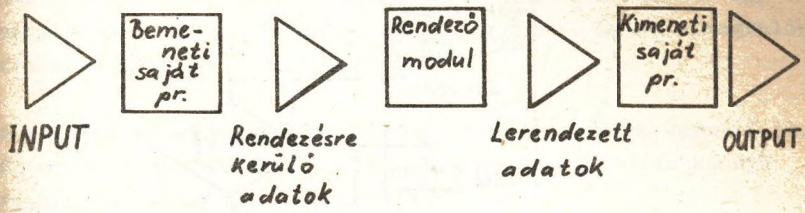
ha max rekordhossz $<$ 256, akkor 8 lap,
ha 256 $\leq$ max rekordhossz $<$ 512, akkor 12 lap,

- a rendezés gyorsasága függ a munkaterület nagyságától, /Overlay szerkezetü felhasználói programnál nagyobb hely biztositható/,
- a rendezés elve
  memóriában a kapott rekordokból turnus módszerrel stringeket képez. A stringeket a rendelkezésre álló lemezterület elejétől irja lemezre. Minden stringről stringrekordot készit, amely tartalmazza a string kezdet lemezcimét és a stringben lévő "legjobb" kulcsot. A stringrekordokat a rendelkezésre álló lemezterület végétől, visszafelé irja fel. Külön string összeválogatási menet nincs.

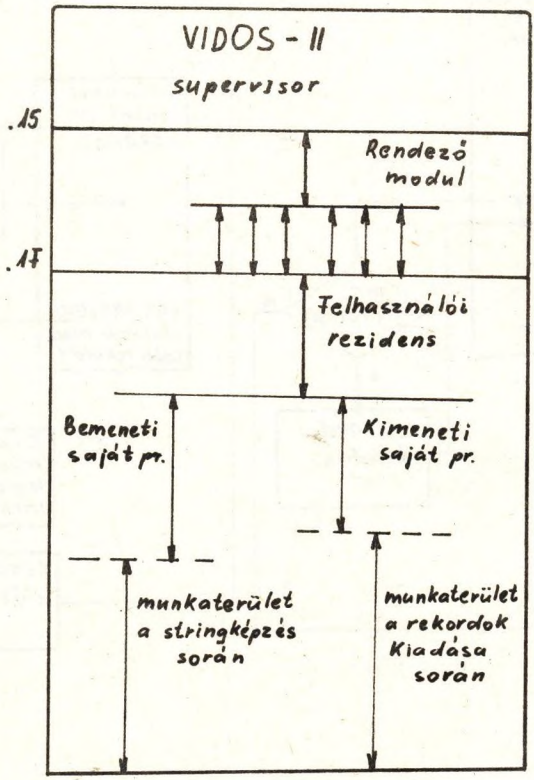
  A rendezésből kivitel során a stringrekordok alapján szintén turnus módszerrel választja ki a kiadásra kerülő rekordot,
- ujrainditási lehetőséget biztosit a kimeneti program kezdetéről,
- a felhasználót részletes hibaüzenetekkel segiti.

A továbbiakban néhány ábra mutatja be a rendező modul müködését.

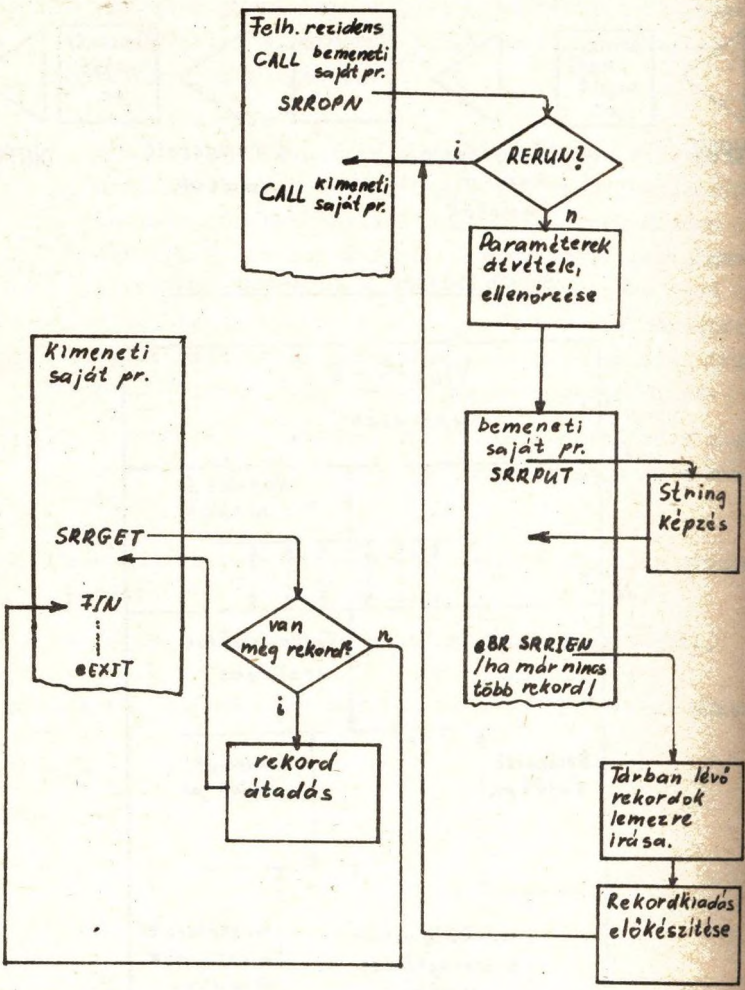
Adatok áramlása a rendezés során



Tárfelosztás a rendezés során



Felhasználói saját program és a rendező modul kapcsolata



A rendező modul terve ugy készült, hogy lehetőséget biztosit a bővitésre. Mágnesszalagot is felhasználhatna munkatárolóként, természetesen ez a gyorsaság rovására menne, de igy nagytömegü adat rendezését tudná elvégezni. Ennek a megvalósitására még nem került sor.

A mágneslemezes rendező modul ez év februárjában készült el, és azóta több felhasználói programban bizonyitotta könnyü felhasználhatóságát, gyorsaságát.

EGY MÓDSZER A SOFTWARE DOKUMENTÁLÁSÁRA ÉS ANNAK EGY SZÁMITÓGÉPES MEGVALÓSITÁSA

Rosta János
Számitástechnikai Koordinációs Intézet

BEVEZETÉS

Aki software fejlesztéssel foglalkozik, jól tudja, hogy ahhoz, hogy egy adott célra megtervezett programot vagy programrendszert üzemszerüen használni lehessen, távolról sem elegendő az adott programot vagy programrendszert megtervezni, elkésziteni, kipróbálni és adathordozón /papirszalag, kártya, diszk-file stb./ tárolni.

A software-sek tudják, hogy egy program - feltéve, hogy nem néhány utasitásról van szó - soha nincs "teljesen készen". A programban az üzembehelyezés után is előfordulhatnak rejtett hibák, hiányosságok, amelyek csak a rendszeres használat során derülnek ki, ezért fel kell készülni arra, hogy a programot az üzembehelyezés után is javitani, módositani kell. Nyilván nem szabad számitani arra, hogy a program készitője mindig rendelkezésre áll erre a célra.

Az a hardware "környezet", számitógépes konfiguráció, amelyre a programot eredetileg tervezték is sokszor változik az idők folyamán. Az is gyakran előfordul, hogy egy adott célra készült programot különböző helyeken, különböző konfigurációkon kell használni.

Maga a cél, amelyre a programot tervezték is módosulhat, bővülhet a későbbiek folyamán. Nagyon gazdaságtalan megoldás lenne ilyenkor az egész programot ujra megirni.

Egy adott programnak, programrendszernek vannak olyan önálló moduljai, szubrutinjai, amelyek felhasználhatók egy teljesen más programhoz is. Például egy lebegőpontos aritmetika tesztprogramhoz megirt véletlenszám generáló modul felhasználható tucatnyi más programban, például egy "Monte-

-Carlo" módszert megvalósitó programban.

Nyilvánvaló tehát, hogy egy adott programhoz vagy programrendszerhez, amelyet a továbbiakban software témának fogunk nevezni, a legkülönbözőbb célu és formáju dokumentumokat kell elkésziteni pl. programlistát, különböző szintü program leirásokat, használati utasitást, kézikönyvet, müködési blokkdiagramokat és folyamatábrákat stb.

Mindezeknek közös vonása, hogy szemben a programmal, amely egy eszmei alkotás, ezek "objektumok", tárgyi formájuk van: lyukszalag, egy mágnesszalagnak egy jól meghatározott része, egy vagy több ROM tok stb. Ezeket a továbbiak során software terméknek vagy dokumentációs elemeknek fogjuk hivni.

Ha az adott software elkészülésének a teljes folyamatát tekintjük, akkor a fenti software termékekhez egy sor olyan objektum csatlakozik, amely a software elkészitésének folyamatát tükrözi: célkitüzés, tervcél, specifikáció stb. és egy sor olyan, amely a téma "utóéletét" tükrözi; átvételi jegyzőkönyv, minőségi bizonyitvány, fejlesztési kézikönyv stb. Ha a program ujabb változata vagy kibocsátása készül, természetszerüleg ujabb software termékek készülnek.

Természetesen egy adott software témához tartozó sokféle információ csak akkor használható jól, ha az információk valamilyen módon rendszerezve állnak rendelkezésre és az információk gyüjtése, valamint az információkhoz való hozzáférés gyors, kényelmes módon történik. Igy szükség van olyan módszerre, amely javaslatot tesz arra, hogy egy adott tipusu software alkotáshoz milyen dokumentumokat kell és milyeneket ajánlatos elkésziteni, továbbá amely megteremti a lehetőséget arra, hogy a dokumentumokat információ strukturába lehessen rendezni. Szükség van olyan számitógépes rendszerre is, amely a fenti módszer szerinti dokumentumok összegyüjtését, valamint a dokumentumokról és a dokumentu-

mok strukturájáról való tájékozódást gyorssá és kényelmessé teszi.

A SOFTWARE DOKUMENTÁCIÓ RENDSZERE

A Számitástechnikai Koordinációs Intézet Hardware Laboratóriumában még 1973 végén készült el egy módszer, amellyel a bevezetésben foglaltakat igyekeztünk megvalósitani. A továbbiakban mind a módszert, mind pedig a számitógépes rendszert SWDR-nek fogjuk nevezni.

A módszer részletesen felsorolja a software témához tartozó elképzelhető software termékeket, mégpedig a software alkotás készitése alatt keletkező, valamint a kész software-hez tartozó termékeket. A software alkotásokat a következő alap-szintek valamelyikébe sorolja:

- rendszer
- programrendszer
- program
- modul-csomag
- modul

Az SWDR előirja, hogy az egyes alapszintekhez tartozó software alkotásokhoz minimálisan milyen software termékeket kell elkésziteni a software kidolgozása során és milyeneket a kész software-hez.
Itt példaképpen felsoroljuk, hogy egy programnak minősithető kész software alkotás minimálisan milyen software termékekből áll:

- Törzslap
- Az alkalmazás leirása
- Müszaki leirás
- Programozói kézikönyv
- Operátori kézikönyv
- Utasitás lista
- Forrásnyelvü adathordozó

- Memóriakép adathordozó /bináris/
- Dokumentáció jegyzék

Az SWDR szerint minden software termékhez tartozik
- egy kód
- egy nyilvántartási lap /Software Lap, a továbbiakban SL/
- egy reprodukálásra alkalmas törzspéldány.

A software termék kódja

A software termék kódja a következő szerkezetü:

AAAA.BBBBB-CC.DD-EE.FF

- A kód "A" mezeje a számitástechnikai rendszer azonositója. Rendszerint a számitógéptipust jelöli, amely a hardware konfiguráció központjában áll /például R-12/.
- A kód "B" mezeje a software termék dokumentációs sorszáma. Öt jegyü szám, minden software termékhez külön sorszám tartozik.
- A kód "C" mezeje a változatszám, amely a funkcionálisan alapvetően megegyező, de részletekben eltérő és különkülön használható dokumentációs elemek megkülönböztetésére szolgál.
- A kód "D" mezeje a kibocsátási /release/ szám. Ezzel különböztetjük meg azokat a dokumentum elemeket, amelyeket eredetileg nem terveztünk be és amelyek szolgáltatásaikban nem adnak ujat /tipikusan ilyenek az eredetileg rejtett hibát tartalmazó programok javitott kibocsátásai, ésszerübb megoldások stb/.
- A kód "E" mezeje a technikai kód, amely a software termék tipusát jelöli meg.

Néhány példa:

CK	Célkitüzés
TC	Tervcél
TL	Törzslap
15	Programozói kézikönyv

LI Utasitáslista

- A kód "F" mezeje a kötetszám.
  Itt különböztetjük meg egymástól egy software termék /pl. rendszerprogramozói kézikönyv/ különböző köteteit.

Nyilvántartási lap /SL/

Egy software termék Software Lap-ja három különböző részből áll: a software termékre jellemző információk, strukturális információk és módositások.

A software termékre jellemző információk

Itt kell elhelyezni a dokumentum sorszámát, nevét, kódját, tárolási helyét, a készitő nevét, a dokumentum kiadásának dátumát, az SL elkészülésének dátumát, a dokumentálásért felelős személy aláirását.

Strukturális információk

Az SWDR kötelezően előirja, hogy egy adott software témához tartozó, azonos release ill. változatszámmal rendelkező dokumentum-elemek "fa" strukturát alkossanak.
Ezt a strukturát támogatja az SL "Törzslap kódja" ill. "Csatlakozó lapok sorszáma és kódja" bejegyzése.
Egy SL kitöltésekor a "Törzslap kódja" rovatba kell bevezetni azt a - a "fa" struktura definicója miatt egyértelmü - SL kódot, amelynek a szóbanforgó SL az /egyik/ ága, a "Csatlakozó lapok sorszáma és kódja" rovatba pedig az adott SL összes ágát. Ugy is fogalmazhatjuk, hogy egy SL törzslapja az SL "felett" elhelyezkedő szomszédos SL, az SL csatlakozó lapja/i/ pedig az SL "alatt" elhelyezkedő szomszédos SL/ek/.

Ha egy software terméknek több változata vagy kibocsátása létezik, akkor mint láttuk ezeket a kód változatszám ill. kibocsátás szám részében meg kell különböztetnünk. Az egymással ilyen kapcsolatban álló software termékek összetartozását mutatja az SL "Erre hivatkozó lapok sorszáma és

kódja" ill. az "Ebből hivatkozott lapok sorszáma és kódja" rovata. Például, ha az X sorszámu SL-el leirt software terméknek készül egy uj változata, amely Y regisztrálási számot kap, akkor az Y sorszámu SL "Ebből hivatkozott lapok sorszáma és kódja" rovatába be kell irni az X sorszámu SL kódját és forditva, az X sorszámu lapra fel kell venni az Y sorszámu SL kódját az "Erre hivatkozó lapok sorszáma és kódja" rovatba. Ilymódon egy software téma teljes strukturája nem is "fa" hanem "erdő", amelynek különböző "fái" az azonos változat /vagy release/-hez tartozó software termékek.

Módositások

Ha egy Software Lapon, az őt tartalmazó struktura bővülése, megváltozása következtében uj, pótlólagos bejegyzést kell tenni, ezt az SL "Módositások" részébe kell bevezetni. Itt kell tehát gyüjteni azokat a strukturális információkat, amelyek az adott SL kiadása után keletkeztek.

SWDR: SZÁMITÓGÉPES SOFTWARE DOKUMENTÁCIÓ RENDSZER

1974 elején készült el az SzKI Hardware Laboratóriumában az SWDR első számitógépes reprezentációja, amely a software termékek az előző fejezetben ismertetett módon rendszerezett információit tárolja, különböző szempontok szerint visszakeresi és karbantartja. A rendszer továbbfejlesztett változatát 1975 áprilisa óta kisérleti üzem-ben általánosan használják az egész Intézetben.

A KONFIGURÁCIÓ

A használt konfiguráció a következő:

- R-10 számitógép /12 Kszó operativ memória/
- 2 db MOM diszk /1600 Kbyte kapacitás/
- 1 db MOM lyukszalag egység
- 1 db line printer
- 1 db display-konzol
- 3 db tároló display-terminál

- 3 db TAM-200 modempár
- 3 db egy érpáros telefonvonal

A rendszer szolgáltatásai az Intézet három fő telephelyén elhelyezett távoli display-terminálokon vehető igénybe. Az alkalmazott adatátviteli eljárás aszinkron, a rendszer - az SzKI Hardware Rendszertechnikai Laboratóriuma által készitett - aszinkron monitorral kibővitett OS-10 operációs rendszer irányitása alatt fut.

Itt jegyzem meg, hogy tervbevettük az SWDR rendszer átalakitását egy nagyobb konfigurációra.
A tervezett rendszer R-12 számitógépen müködik majd OS-12 operációs rendszer irányitása alatt. A használt háttértár egy 5 Mbyte kapacitásu EC-5052-es diszk lesz.

Az SWDR szolgáltatásai

Az SWDR lényegileg egy könyvtárkezelő programrendszer; a software termékekről készült Software Lapok könyvtárának kezelő programja.
Az SL-ek a diszkeken helyezkednek el, tömör formátumban; egy SL-et tartalmazó diszkterület változó hosszuságu.

A rendszer szolgáltatásai három csoportba oszthatók:
- Információ visszakeresés
  Ide soroljuk mindazokat a szolgáltatásokat, amelyekkel minősitetlen információkat mindenki számára hozzáférhetővé teszünk. Ebbe a csoportba tartoznak a következő utasitások:
  SEARCH, PRINT, SEARCH-STRUCT, PRINT-STRUCT
- Információba beavatkozás
  Az ebbe a csoportba tartozó utasitások hatására a diszken tárolt információ megváltozik. Tekintve, hogy szakszerütlen kezelés esetén az SL könyvtár maradandó károsodást szenvedhet, az információba avatkozó utasitásokat software uton védjük, igy azokat csak az arra illetékes

személy használhatja. Ezen utasitások csak akkor hatásosak, ha kiadásuk előtt a kezelő egy " titkos kódot " ad meg, egyébként a rendszer nem fogadja el az utasitást. Ide tartozó utasitások: GEN, REGEN.

- Karbantartási utasitások
  Ebbe a csoportba tartozó utasitásokkal lehet az SL-ek könyvtárát karbantartani /könyvtárat kimenteni lyukszalagra, visszaolvasni, régebbi állapotot korszerüsiteni stb/. Ezen utasitások kiadása természetszerüleg csak azon a display-terminálon engedélyezett, amely helyileg a központi egység mellett helyezkedik el, igy ezeket az utasitásokat a rendszer nem fogadja el, ha a másik két display-terminál valamelyikéről adták ki.
  Karbantartási utasitások: PUNCH, COMPARE, END, READ, UPDATE.

Az SWDR utasitásai

SEARCH /keresés/
Erre a parancsra az SWDR a következő funkciókat tudja végezni.

- Ha a parancs után 5 decimális számjegyben egy SL sorszámát adjuk meg, akkor SWDR az adott sorszámu SL-et kikeresi a diszkről és megjeleniti a display-terminálon.
- Ha a parancs után 4 karakterben valamely számitástechnikai rendszer azonositóját adjuk meg, akkor SWDR kikeresi a fenti azonositónak megfelelő SL-ek kódjait és azokat sorban a display-terminálon megjeleniti.
- Ha a parancs után két karakterben technikai kódot adunk meg, akkor SWDR kikeresi mindazon SL-ek kódjait, amelyek technikai kódja a fentivel megegyezik és azokat sorban a display-terminálon megjeleniti.

Az SWDR gépi nyilvántartó rendszere nem tesz semmilyen megkötést a dokumentum technikai kódjára ill. számitástech-

nikai rendszer azonositójára.

PRINT /nyomtatás/
Erre a parancsra SWDR a következő funkciókat tudja elvégezni:

- Ha a parancs után 5 decimális számjeggyel egy software lap sorszámát adjuk meg, akkor SWDR ezt a software lapot a sornyomtatón kinyomtatja.
- Ha a parancs után 2 karakterben technikai kódot adunk meg /lásd fent/, akkor SWDR mindazon SL-ek kódját, melyek technikai kódja az adottal megegyezik, sornyomtatón kinyomtatja.
- Ha a parancs után 4 karakterrel számitástechnikai rendszer azonositót adunk meg, akkor SWDR mindazon SL-ek kódját, amelyekben a számitástechnikai rendszer azonositója az adottal megegyezik, a sornyomtatón kinyomtatja. Az SWDR a line printer papiron megjeleniti, hogy a PRINT utasitást melyik display-n adták ki. Ez megkönnyiti az egy nap alatt különböző helyről kért listák szétválogatását.

SEARCH-STRUCT /struktura keresés/
A parancs után 5 decimális számjegyben egy SL sorszámot adhatunk meg. Hatására SWDR megkeresi a megadott sorszámu SL-et, majd ennek a strukturára vonatkozó információi segitségével - fokozatosan - felderiti az egész software téma strukturáját és azt a file kezelő rendszereknél szokásos formátumban megjeleniti a display-terminálon.

Mivel egy software témához az előző fejezetben leirt módon több "fa" tartozhat, amelyek közti kapcsolatot a hivatkozások /"Erre hivatkozó lapok", "Ebből hivatkozott lapok"/ irják le, ezért a SEARCH-STRUCT utasitás feltárja az egymás mellé rendelt "fa" strukturák közti kapcsolatot is. A parancs hatására tehát megjelennek azon "fa" strukturának

elemei /kódjai/, amelyek között szerepel a parancsban megadott sorszámu SL kódja is egymás alá irva olymódon, hogy minden egyes csatlakozó lapon keresztüli kapcsolatot a kódnak eggyel jobbra való eltolása jelképez. Azon SL kódoknál, amelyekre vagy amelyikből hivatkozás történik, nyillal szemléltetve a hivatkozás irányát, a hivatkozó vagy hivatkozott SL sorszámát is feltüntetjük. Ilymódon - a nyilakkal megjelölt sorszámokat figyelve - több SEARCH--STRUCT utasitással az egész software téma strukturáját felderithetjük.

PRINT-STRUCT /struktura kinyomtatás/

Erre a parancsra SWDR a következő funkciót végzi:
A parancs után megadott 5 decimális számjeggyel meghatározott sorszámu SL-hez tartozó információ struktúrát SWDR a sornyomtatón kinyomtatja.
A kinyomtatásra az előző utasitás leirásában foglaltak érvényesek.

GEN /generálás/

Az utasitás uj software lap előállitására szolgál. A rendszer az előállitandó SL sorszámát automatikusan generálja, olymódon, az eddig létező legmagasabb sorszám+1 lesz az uj SL sorszáma.

A GEN utasitás hatására rendre megjelennek a Software Lap rovatai és minden egyes rovatnév megadása után az SWDR várakozik, hogy a felhasználó a display-terminál klaviaturájáról gépelje be az információt, amelyet az illető rovatba el kiván helyezni. Minden rovat kitöltését # karakter leütésével kell befejezni.

A GEN utasitás közben az adott válaszokról automatikusan lyukszalag másolat készül, amelyet a később ismertetendő UPDATE utasitás használ.
A GEN utasitás az SL előállitása és diszkre való felvite-

le után u.n. "automatikus módositást" hajt végre.
Ez a következőt jelenti: Az SL bizonyos rovatainak kitöltése - a Software Dokumentáció Rendszere szabályai szerint - más SL-ek módositását teszi szükségessé. Ha pl. a 00053 sorszámu SL "Törzslap száma" rovatában a 00020 bejegyzés szerepel, az azt jelenti, hogy a 00020 sorszámu SL csatlakozó lapjai között kell szerepelnie a 00053 sorszámu SL-nek. Ha ez nem szerepel, akkor az SL "Módositások" részében be kell vezetni. Ugyanigy, ha egy SL-ből történik egy hivatkozás egy másik SL-re, akkor ezen utóbbi SL-ben az előző SL sorszámát és kódját "erre hivatkozó lap"-ként fel kell tüntetni.

Az "automatikus módositás" szolgáltatás az ilyen tipusu funkciókat automatikusan elvégzi és a felhasználót csak akkor értesiti, ha valamilyen rendellenesség következett be, pl. valamelyik SL számára kijelölt módositás terület betelt.

REGEN /ujragenerálás/
Az utasitás már meglévő SL-ek ujraelőállitására szolgál.

Két változatban használható:
- regenerálás; ekkor az egész SL ujrairható, az SL ujonnan adott válaszokat tartalmazza.
- aktualizálás; ekkor azok a rovatok, amelyekre válaszként kizárólag a "#" karaktert adjuk változatlanul maradnak, azok a rovatok, amelyekre uj választ adunk az uj válaszokat tartalmazzák. Ezzel a változattal lehet az SL-et aktualizálni, a hiányzó rovatokat /tipikusan dátum, aláirás/ pótolni.

Az utasitás müködése egyebekben megegyezik a GEN utasitás leirásában foglaltakkal.

PUNCH /lyukasztás/
Az utasitás hatására a teljes SL könyvtár lyukszalagra ke-

rül tömör formátumban.

COMPARE /összehasonlitás/
Az utasitás a PUNCH utasitással nyert lyukszalag helyességét ellenőrzi.

END
Hatására SWDR müködését befejezi és a vezérlést átadja az operációs rendszernek.

READ /beolvasás/
Beolvassa a lyukszalagon tárolt SL könyvtárt és elhelyezi a diszken.

UPDATE /korszerüsités/
A parancs hatására SWDR az információs rendszerbe való beavatkozásról /GEN, REGEN/ szóló lyukszalagokat beolvassa és a bennük foglalt módositásokat végigvezeti. Ez a parancs a diszk meghibásodása utáni regenerálásnál használható.

FÜGGELÉK: Az SWDR hibaüzenetei

ILLEGAL INSTRUCTION	hibás parancs
NONEXISTENT PAGE: Y	az "Y" sorszámu SL nem létezik
NONEXISTENT TECHNICAL CODE	nincs ilyen technikai kód
NONEXISTENT SYSTEM	nincs ilyen rendszerazonositó
UNDEFINED REQUEST	hibás paraméter
PUNCH ERROR IN TAPE	lyukasztási hiba
NO PLACE FOR MOD. IN Y	nincs hely módositásra az Y sorszámu SL-ben.

AZ M-51 ÉS M-52 MIKROGÉP SZIMULÁCIÓS RENDSZERE R-10 SZÁMITÓGÉPEN

Ebergényi Kálmán
Számitástechnikai Koordinációs Intézet

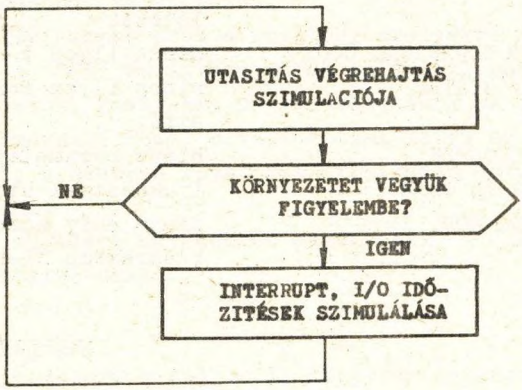
AZ M-51 ÉS M-52 MIKROGÉP PROGRAMJÁNAK SZIMULÁLT ÉS NYOMKÖVETETT FUTTATÁSA R-10 SZÁMITÓGÉPEN

Szimuláció

Az M-51 mikrogép programjának szimulálása alatt nem egyszerü utasitás-szimulációt értünk, hanem jól körülhatárolt és kézbentartható konfiguráció szimulációt, a valódi időt is beleértve.

A konfigurációt itt csak mint a program környezetét vizsgáljuk, csak azokat a jellemzőit, amikkel a program kapcsolatot tart, vagy kapcsolatba kerül.

A program és a környezet kapcsolatának sémája az alábbi ábra szerinti:



Nyomkövetés

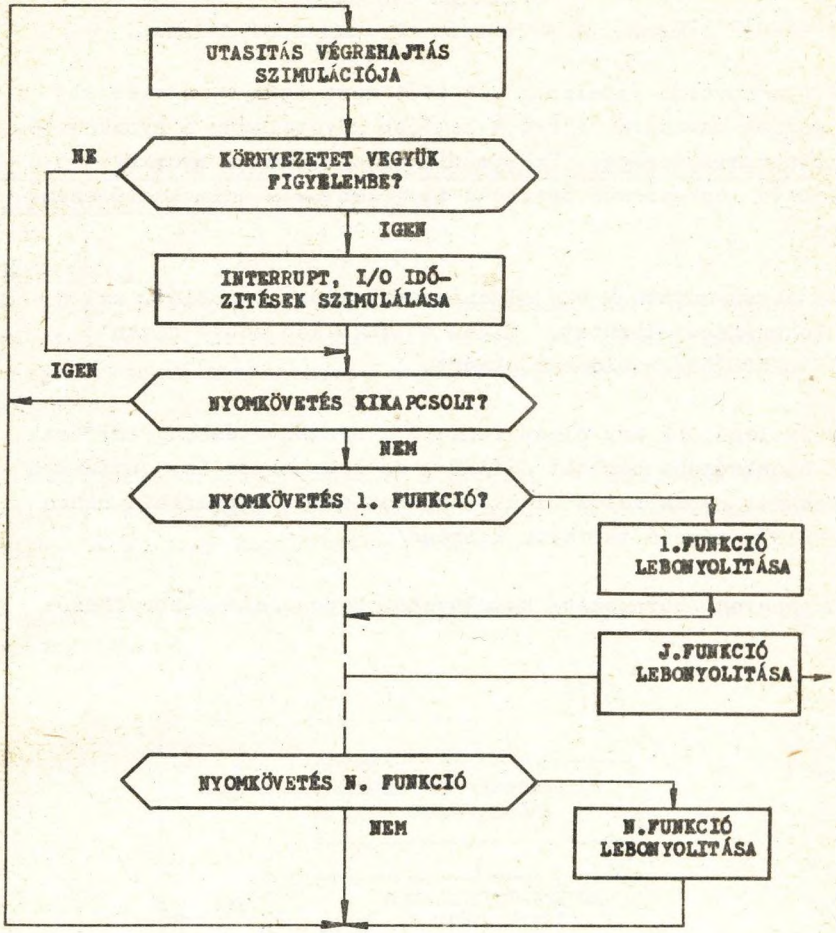
A nyomkövetés lehetőséget ad a program vagy a program/környezet rendszer viselkedésének statikus vagy dinamikus vizsgálatára.

A nyomkövetés általában sok időt vesz igénybe a vizsgált program rovására, ezért a legfőbb követelmény a nyomkövetés kikapcsolhatósága. Kikapcsolt állapotban a nyomkövetés a lehető legkevesebb overhead-et jelentse a szimuláció számára.

Az ábrán mutatunk egy olyan nyomkövetési funkciót, ami a blokksémából kimutat, ilyen a megállás adott címen /BREAKPOINT/, példának okáért.

Lesz legalább egy olyan funkció a nyomkövetésben, ami csak a blokksémába bemutat /START/, de lesz olyan is, aminek a konkrét nyomkövetés blokksémájához semmi kapcsolata nincs /PRINT, tároló tartalom kiírása/.

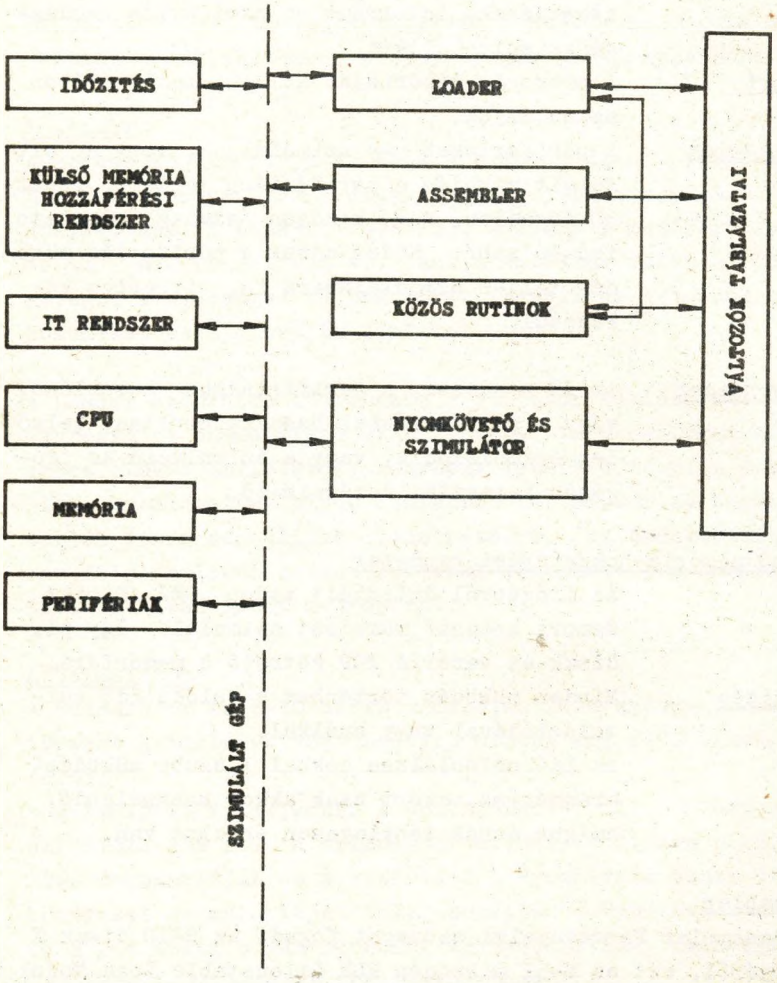
A program /környezet/ nyomkövetés kapcsolata a következő:


UTASITÁS VÉGREHAJTÁS SZIMULÁCIÓJA
NE
KÖRNYEZETET VEGYÜK FIGYELEMBE?
IGEN
INTERRUPT, I/O IDŐ-ZITÉSEK SZIMULÁLÁSA
IGEN
NYOMKÖVETÉS KIKAPCSOLT?
NEM
NYOMKÖVETÉS 1. FUNKCIÓ?
1.FUNKCIÓ LEBONYOLITÁSA
J.FUNKCIÓ LEBONYOLITÁSA
NYOMKÖVETÉS N. FUNKCIÓ
NEM
N.FUNKCIÓ LEBONYOLITÁSA

AZ R-10/M-51, M-52 SOFTWARE RENDSZER KONCEPCIÓJA

A Szimulátor és Nyomkövető az Assembler-rel és Loader-rel együtt szerves egészet képez, amennyiben közös táblázatokat és rutinokat használnak.

A programok összefüggésének sematikus vázlata a következő:



A szimulált gép

M-51 szimulációjában a következő főbb szimulációs egységeket különitjük el, mint az ábrán is látható.

CPU — Ide soroljuk a központi egység utasitáskészletének, regisztereinek, statikus bitjeinek, stackjének, interrupt és perifériás rendszerének szimulálását.

Memória — A mikrogép memóriáját az R-10 memóriájában szimuláljuk.

Perifériák — A perifériákat ugy szimuláljuk, hogy a szimulált müködés a perifériára jellemző inputot fogadjon, vagy kiadjon /szalag, nyomtató, irógép stb/. Módot adnak a perifériás müködés teljes szimulálására is, irógépen keresztül.

IT rendszer — Az IT rendszer, a perifériákhoz hasonlóan, vagy a perifériákkal összekapcsoltan, belsőleg szimulálódik, vagy a szimuláció az irógépen keresztül vezérelhető.

Külső memória-hozzáférési rendszer

Az irógépről definiált módon DMA /Direct Memory Access/ müködést szimulál. Igy pl. diszk-et vezérlő IOP köthető a memóriára.

Időzités — Minden müködés történhet a valódi idő szimulációjával vagy anélkül.
Az idő szimulálása sokkal lassubb müködést eredményez, ezért csak akkor használandó, amikor annak ténylegesen értelme van.

Assembler

Az Assembler Forrásnyelvi szöveget fogad, az R-10 diszk X mezőjéről, ezt az M-51 mikrogép RLM Relocatable Load Modul formátumu gépi kódjába leforditja, a szükséges táblázatokat

elkészíti, az RLM szöveget a diszk Z mezőjébe helyezi el. Egymásután elhelyezés céljából az Assembler, többször ujrainditva, hosszu programokat képes forditani. Az egyes programok azonos cimkéket használhatnak, de egy forditáson belül is lehetséges azonosan kódolt, lokális változók bevezetése.

Feltételes forditás direktivákkal a szöveg különféle munka-kópiái hozhatók létre, a hibakeresés megkönnyitésére.

A Loader

A Loader fő funkciója az M-51 Assembler által leforditott RLM formátumu szövegek összetöltése, gépi utasitásokká való átalakitása. A Loader a szöveget a diszk Z mezőjéről veszi és az Y mezőre helyezi el. Az összetöltés során a Loader elvégzi a szükséges relokációkat, kezeli az externál szimbólumokat és a globális cimkéket.

A Loader ezenfelül létrehozza a változók táblázatát, ami egyrészt az összetöltés után hasznos információkat ad az egyes cimkék hollétéről, másrészt lehetővé teszi a szimbólikus nyomkövetést olyan esetekben is, amikor az Assembler táblák éppen nem állnak rendelkezésre. Az összetöltés során feloldatlanul maradt externál cimkéket is a Loader táblázataiban lehet felfedezni.

A Szimulátor

A Szimulátor az R-10 diszk Y mezőjén lévő szöveget utasitásként szimulálva végrehajtja, mintha egy M-51 mikrogép volna.
A szimuláció kiterjedhet a konfiguráció, ill. a környezet szimulálására is. A szimulátor müködését minden utasitás után megszakitjuk és a vezérlést a Nyomkövető veszi át. A környezet szimulációját tulajdonképpen a Nyomkövető végzi, de a két modul olyan szorosan egybeépül, hogy leghelyesebb Nyomkövető-Szimulátor programról beszélni.

A Nyomkövető

A Nyomkövető egy speciális programot interpretálva futtató program, nyomkövető nyelven megirva.
A Nyomkövetés célja a szimuláció teljes mértékü kézbentartása, a különféle szimulációs opciók vezérlése.
A Nyomkövető nyelvben statikus parancsok adhatók a szimulált program vizsgálatára, dinamikus parancsok a szimuláció futás közbeni ellenőrzésére és szervezési /számitási/ parancsok járulékos müveletek végzésére.

Az Editor

Az Editor szimbólikus szövegek irására, szerkesztésére való. Egyformán használható a szimulálandó és a nyomkövető programok kezelésére.

RENDSZER PARANCSOK ÖSSZEFOGLALÁSA

* INITIATE	- iniciálás
* ASSEMBLE	- Assembler meginditása, vagy ujrainditása
* LOAD	- RLM szövegek összecsatolása
* OUTPUT <program-név>	- kimásolás diszkről
* INPUT	- diszkre felvitel
* EDIT <forrás szöveg megjelölése>	- szimbólikus szöveg szerkesztése, hivása
* TRACE	- nyomkövetést meginditja
* INTERRUPT	- a rendszer visszanyeri a vezérlést
* CONTINUE	- a megszakitott programot folytatjuk.

INITIATE

A rendszer iniciálását jelenti.

- A diszken alaphelyzetbe állnak vissza SYMPR, BINPR, RLMPR, TRCPR pointerek.
- Az Assemblerben törlődnek a változó-táblák.

- A Loaderben törlődnek a változó-táblák.
- A Szimulátor-Nyomkövetőben törlődnek a változó-táblák.

Az R-10/M-51 Software Rendszer Assemblere fordítások között is megőrzi az összes szimbólumot, megkülönböztetve őket egymástól a program nevével, amelyikben előfordultak. Igy több programban azonos nevü változók is lehetnek, mindegyiket szimbólikusan is lehet kezelni, anélkül, hogy összekeverednének, kivéve, ha a programok azonos nevüek, vagy megnevezetlenek. Megnevezetlen programok összetöltése esetén az egy fordításhoz tartozó szimbólumokat a program-táblázatok pointerei határolják.

ASSEMBLE

Az Assembler inditását jelenti.

- A diszken SYMPR az X mező elejére áll vissza, mert az Assembler uj forrásszöveget kapott.
- Az Assembler változó táblái törlődnek, kivéve a globális változókat. A táblák törlése annyit jelent, hogy az Assembler uj táblák kitöltését kezdi meg, megkülönböztetve az uj változókat a régiektől a fordítás sorszámával.
- Az Assembler az X mezőn talált szöveget forditani kezdi, az elejétől egészen addig, mig egy file-vég jelzést nem talál. Ha a szövegben előfordul az .END direktiva, az egy fordítási modul végét jelzi. Egy fordítás során több modul is fordítható egyszerre.
- A fordítás eredményeként az Assembler RLM szöveget helyez el a diszk Z mezőjére, folytatólagosan.

LOAD

A relokáló LOADER inditását jelenti.

- A diszken az RLMPR pointer a Z mező elejére áll és innen RLM formátumu rekordokat véve a programokat összetölti a diszk Y mezőjére, annak elejétől kezdve. Igy egy ujabb LOAD müvelet a bináris memóriaképet teljesen fel tudja ujitani.

- A LOADER felépiti a saját táblázatait müködése közben. Ig program szimulációját nem feltétlenül kell, hogy fordità előzze meg.

OUTPUT

Programok kimásolását végzi a diszknél. A parancsnak paramé terek is adhatók

- OUTPUT Z kimásolja a teljes Z mezőt, illetve az ott lévő összes programot. A formátum RLM.
- OUTPUT Z, <program név> kimásolja a névvel megjelölt programot.
- OUTPUT Y kimásolja az Y mezőn található programot. A formátum abszolut.
- Hiányparaméter = OUTPUT Y

INPUT

Programokat lehet vele a diszkre felvinni. A parancs paraméterei lehetnek

- INPUT Z felirja a szöveget a diszk Z mezőjére, folytatólagosan.
- INPUT Y felirja a szöveget a diszk Y mezőjére, elejétől kezdve, ill. ahová szükséges.
- Hiányparaméter = INPUT Y

EDIT

Forrásnyelvü szövegek kezelését lehet vele végezni /beolvasás, kiirás, javitás stb/.

A parancsnak paraméterei is vannak:

- EDIT S a szimulálandó program forrásszövegét tekinti szövegnek a diszk X mezőjén.
- EDIT N a nyomkövető program forrásszövegét tekinti szövegnek.
- Hiányparaméter = EDIT S

TRACE

A nyomkövető programot inditja meg. A nyomkövetőnek adott

azonnal végrehajtásu u.n. statikus parancsokkal, vagy a szimulált programot futása közben, ellenőrző u.n. dinamikus parancsokkal teljes HW-SW mikrogép rendszerek szimulálhatók, ellenőrizhetők, tesztelhetők, szimulált real-time körülmények között.
A nyomkövető hivni tudja a szimulátort, egy-egy utasitás szimulálására, de a nyomkövető program végrehajtását is tudja vezérelni.

INTERRUPT
Ilyen input szöveg adásával a szervező programnak visszaadható a vezérlés.

CONTINUE
Interrupt paranccsal megszakitott müködés folytatását inditja meg.

STATISZTIKAI KIADÓ VÁLLALAT
Felelős vezető: Kecskés József igazgató
Terjedelem: 20,9 (A/5) iv Nyomdaüzem - 75-1349-7