Szupermikroprocesszorok és alkalmazásaik

VAJDA FERENC
MTA Központi Fizikai Kutató Intézet
Mérés- és Számítástechnikai Kutató Intézet

VAJDA FERENC

a műszaki tudományok doktora, az MTA Központi Fizikai Kutató Intézet tudományos tanácsadója és a BME Műszer- és Méréstechnika Tanszék docense. Fő kutatási területei a számítógépek architektúrája, a mikroprogramozás és a képfeldolgozás. Tagja az Euromicro igazgatótanácsának és a HTE—MATE—NJSzT Mikroprocesszorok Alkalmazása Munkabizottság elnöke.

ÖSSZEFOGLALÁS

Egy rétegezett modell alapján mutatjuk be az új szupermikroprocesszor családok célkitűzéseit és megoldásait. A cikk az utasítás rendszer, az operációs rendszer, a programozási és rendszer szint, valamint a mikroprogramozási és digitális logikai szint számára — a szupermikroprocesszorok által — nyújtott architekturális támogatással foglalkozik.

1. Bevezetés

Napjainkban, amikor a félvezető áramköri technológia már alig korlátozza, inkább csak szolgálja a mikroprocesszor architektúrák implementációját, a mikroprocesszorok fejlődésének új szakaszához érkeztünk el. Az újonnan kifejlesztett — elsősorban 32-bites — mikroprocesszorok jellemzői elérték és bizonyos értelemben túl is haladták a hagyományos nagyszámítógépek tulajdonságait. Ebbe az új kategóriába tartozó mikroprocesszorokat szokás szupermikroprocesszoroknak nevezni. Ennek az új kategóriának a tulajdonságait és lehetőségeit elsősorban az új alkalmazási körülmények definiálják. A mikroprocesszorokat egyre gyakrabban programozzák modern magas szintű nyelveken, amelyek a struktúráit programozást és az összetett adattípusokat egyaránt támogatják. Előtérbe kerültek a bonyolult operációs rendszerek és egyre gyakrabban több processzor együttműködésével oldják meg a feladatokat. A szupermikroprocesszorok architektúráit ós ezek által biztosított tervezési lehetőségeket ezen alkalmazási körülmények szempontjából vizsgáljuk, természetesen csak néhány jellemző példán keresztül [1], [2].

Bár ezt nem tekintjük közvetlen feladatunknak, az 1. táblázatban összefoglaltuk — összehasonlításképpen — a legfontosabb szupermikroprocesszorok néhány jellemzőjét, hogy ezen keresztül is képet kapjunk erről a mikroprocesszor kategóriáról [3], [4], [5], [6].

1. táblázat

A legfontosabb szupermikroprocesszorok néhány összehasonlító adata

Jellemző	Intel iAPX386	Motorola MC68020	National Semiconductor NS32332	AT &T WE 32100	Zilog Z80, 000
Tokon belüli memóriakezelő egység	I	N	N	N	I
Utasítás cache (byte)	N	250	N	256	256
Adat cache	N	N	N	N	I
Címfordító cache	I	N	N	N	I
Cím vezetékek	32	32	32	32	32
Adat vezetékek	32	32	32	32	32
Multiplexeit	N	N	I	N	I
Szegmentálás	I	N	N	I	I
Lapszervezés (byte)	4K	256—32K	512	2K	1K
32 bit regiszterszám	8	16	8	9	16
Max. óra frekvencia (MHZ)	8	16	8	9	16
Pipe fokozatok száma	4	3	3	4	6
Tok csapszám	132	120	84	132	68

Modellek és definíciók

Egy számítógéprendszer nagyon sokféle módon írható le [7], [8], [9]. A számítógép tervezők a a számítógépet csak szerkezeti alapjaiban látják, kizárólag a hardverre koncentrálva. Egy modell, amely az alapvető technológiai változásokat átvészelte, négy szintet használ: rendszer szint, elektronikus áramköri szint, regiszter-átviteli (RT) szint, processzor-tár-kapcsoló (PMS) szint. Mindegyik szintet külön „nyelv" jellemzi, amely az adott szinttel kapcsolatos elemeket és viselkedésük törvényszerűségeit írja le [10], [11], [12].

Beérkezett: 1987. II. 10. (H)

Az előző szerkezeti leírással ellentétben a számítógéptudomány művelői működési nézőpontból vizsgálják a szuper mikroprocesszorokat, számítógép rendszereket. Ennek megfelelően a számítógép értelmezőprogramok rétegeiből épül fel. Az értelmező programot utasítások irányítják és állapotinformációktól (külső és belső) függően működik. Mindegyik szint különböző szintű absztrakciót használ [13], [14]. A különböző működési rétegek közti határok egy számítógép rendszeren belül elkülönített architektúrát határoznak meg. Ezt a leírást funkcionálisnak tekinthetjük, szemben az előző struktúrális leírással.

Természetesen különböző modellek vannak, különböző számú réteggel. Az általunk használt modell felülről kezdve a következő rétegekből (architektúrákból) áll:

1. Rendszer szint architektúrája
2. Programozási nyelv szint architektúrája
3. Operációs rendszer szint architektúrája
4. Utasítás készlet szint architektúrája
5. Mikroprogramozás szint architektúrája
6. Digitális logikai szint architektúrája
7. Elemek szintjének architektúrája

Megkönnyíti egy számítógéprendszer leírását, elemzését, tervezését és használatát, ha azt a szintek hierarhciájának tekintjük. Következésképpen a nem használt szintek átlátszóvá tehetők és belső összetett szerkezetük elrejthető. Ezeket a szinteket vagy rétegeket gyakran cél (target) gépnek, látszólagos (virtuális) gépnek, absztrakt vagy hasonmás (image) gépnek nevezik.

Szemantikus rések

A szemantikus rés kifejezést eredetileg úgy definiálták [15], mint a magasszintű nyelvek és az ez alatt elhelyezkedő számítógép architektúra közti különbség mértékét. Ez az alapdefiníció kiterjeszthető és általánosítható az általunk tárgyalt modell összes rétegére.

A rendszer szint architektúrája és az alatta lévő számítógép architektúra közti rés magában foglalja a multi- ós konkurrens programozás, modularitás, információ elrejtés, hiba keresés és javítás, több processzoros alkalmazások, számítás orientált feladatok stb. támogatásának hiányát. A programozási nyelv szint architektúrája és az alatta elhelyezkedő számítógép architektúra közti rés a számítógép architektúra olyan alapvető tulajdonságainak hiányaival jellemezhető, melyek egy fordítóprogram író feladatát megkönnyítenék, ezáltal téve a fordítóprogramot jobb hatásfokúvá. Ennek a résnek második mértékének tekinthető a magas szintű nyelvek olyan alapelveinek mint blokk struktúrák, eljárások, adat struktúrák és adatreprezentációk stb. ós az alatta elhelyezkedő számítógép architektúra közti különbségek. Harmadik gondolatként — az utasításkészlet összetettsége (beleértve a rendelkezésre álló címzési módokat és az adattípusokat is) képezi napjainkban az éles viták tárgyát [16].

Az operációs rendszer szint szemantikus rése az operációs rendszer szint elvei és az alatta lévő számítógép architektúra közti különbségként definiálható. Ezen a ponton nem szabad megfeledkeznünk az operációs rendszer fő feladatairól [17], úgy mint:

— szolgáltatások nyújtása más programok számára (pl.: tárfelosztás, folyamat szinkronizáció és folyamatok közti kommunikáció),
— más programok megvédése olyan részletektől mint pl.: megszakítás struktúra vagy géphiba,
— „virtuális" tulajdonságok biztosítása erőforrások (mint pl.: processzorok vagy tárak) „átlátszó" megosztására,
— rendszer kezelési eljárások létrehozása és kikényszerítése (olyan területeken, mint adat biztonság vagy üzemezés).

A fő kérdések a tár kezelés (tár szervezés, tár leképzés és védelem a hozzáférés vezérlésével) és a folyamat kezelés (folyamat szinkronizálás és kommunikáció).

A különböző szemantikus rések a számítógéprendszerben sok, nem kívánatos tulajdonságot okozhatnak. Ezek közül soroljuk fel a legfontosabbakat az alábbiakban [15]:

— hatékonysági problémák, melyek következménye ez a nagyszámú utasítás, melyet a fordítóprogramnak generálni kell (és amelyet az architekrurális szintnek értelmeznie kell),
— nagy program méretek (a fent említett probléma más megfogalmazásban),
— program megbízhatatlanság (a nagy program méretek, a „nem természetes" adat típusok és változó kezelés eredménye),
— fordítóprogram összetettsége (a nagy rés csak egy nagyon összetett kódot-generáló résszel hidalható át),
— alacsony szintű programozási termelékenység (mind assembler mind magas szintű nyelv szinten).

Innen kezdve egy mozaik jellegű képet próbálunk adni a mostani szupermikroprocesszorok „magas szintű" nagyszámítógép jellegű architekturális tulajdonságainak fő kérdéseiről A cikk a többszintű modellnek megfelelően van szerkesztve (az elemek szintjének problémái kívül esnek a cikk tárgykörén)

2. Az utasításszint architektúrája

— Fő kérdések:
— az architektúra típusa,
— regiszter szerkezet és szám,
— utasítások típusa és száma,
— címzési módok típusa és száma,
— támogatott adattípusok fajtái,
— teljesítmény mutatók,
— lefelé való kompatibilitás

Utasításkészletek értékelése

Az utasításkészletek értékelésére sok javaslat született [18], [19], [20], [21] Ezek az értékelések legtöbbször mennyiségi alapokon nyugszanak. Ezért megvan arra a lehetőség, hogy visszaéljenek a használatukkal, hiszen az értékelési folyamat a teszt anyag méretéből adódóan kis problémákat megoldandó programokra korlátozódik. Az ilyen

mérési korlátok ellenére legalább részleges alapját képezhetik az utasítás készlet értékelésének. Ezek a „mértékek" többek között a következőkből állnak [22]:

— program méret (statikus mérték),
— fordítási idő (egy tipikus processzor az idő felében fordít),
— futási idő (dinamikus mérték)

A kód hosszúságok és végrehajtási idők mérésére és összehasonlítására a szupermikroprocesszor gyártók és néhány független oktatási intézmény a „benchmark" módszert alkalmazza. Annák ellenére, hogy egy meghatározott program halmazt használtak (az ún Berkley-teszt négy programból áll: Search, Sieve, Puzzle és Ackerman) [19] a mérésnek ez a fajtája nem lehet segítségünkre olyan utasításkészletek és architektúrák megítélésében, melyeket különböző célokra terveztek Egy másik nagyon kérdéses módszer az utasítás használat mérésén és elemzésén alapszik, így kívánja az adott utasítás „hasznosságát" eldönteni. Ennek á módszernek az értéke két cikk [23], [24] végkövetkeztetéseivel demonstrálható, melyek a dinamikus utasításhasználattal foglalkoznak számos program és programozási környezet felhasználásával. Ezek a következtetések a következőkben foglalhatók össze:

— az idő- és frekvencia eloszlások rendkívül aszimmetrikusak, melyek „farka" (kis valószínűségű tartomány) hosszú,
— az eloszlások (néha nagyon) különbözőek,
— a különböző nyelvek és alkalmazások az utasításkészletet különböző módon használják.

Utasításkészlet tervezése egy általános célú számítógépre nehéz válaszutak elé állítja a tervezőket, akik elsősorban technikai ós piaci megfontolások alapján döntenek [22]:

— kompatibilitás (a programok átvihetőségének legfőbb szintje az utasításkészlet, nem csupán a magasabb szintű nyelvek),
— tervezési idő (az univerzális struktúra könnyít a tervezésen, de a legfontosabb teljesítménytényezők árán),
— technológia (az architektúrára ós a teljesítményre gyakorolt közvetlen és közvetett hatásai: chip terület, tokozás, csapok számának korlátozása, jel továbbhaladási késleltetés stb.).

Az utasítás kódolást számos ellentmondó, de ugyanakkor egymásraható tényező befolyásolja:

— ortogonalitás (műveletek, adat típusok és címzési módok között),
— elhelyezési és méret korlátok,
— méretbeli megfontolások (változó vagy állandó formátum).

A Z80,000 utasítások például egy vagy több 32 bites szóba vannak kódolva és páros címeknél kell lenniük a tárban. Elkerülendő a kódolási elégtelenséget, a processzor elő tud venni olyan szórészeket, melyek a fő tár szó határait átfedik [25]. Az NS32032 utasításai akárhány byte hosszúak lehetnek (az alaputasítások 1, 2 vagy 3 byte hosszúak, max. 5 byte-ra való kiterjesztés lehetséges). Utasításkészletének tömörségén javít a regiszter relatív címzési móddal használt változó nagyságú címeltérés (displacement). (Egy eltérés két byte-ja az eltérést kódolja byte-ban, a lehetséges eltérési tartomány 64 ós a teljes címzési tartomány között van) [26].

A nagyobb szóhossz alkalmazásának előnyei a következőkben összegezhetők [3]:

— nagyobb közvetlenül címezhető tár tartomány (32 bit cím = 4 Gbyte tár tartomány),
— megnövekedett teljesítmény (a hosszabb utasítás több információt hordoz),
— megnövekedett funkcionalitás (ortogonális utasítás készlet),
— megjavult pontosság,
— nagyobb fokú biztonság és megbízhatóság,
— szélesebb körű hozzáférés meglévő alkalmazásokhoz (olyan gépekre gondolva, mint pl.: a IBM System 370, vagy a DEC VAX—11 stb.).

Míg a legtöbb 16 bites mikroprocesszorra a DEC PDP—11 architektúra volt nagy hatással, a 32 bites architekrútákat (legfőképp az NS32000 családra gondolva) a VAX—11 architektúra [27] ós részben a korszerű IBM architektúrák befolyásolták (pl.: a Z80,000 string mozgató utasítások) [6]. A sikeresebb architektúra élettartama hosszabb és ugyanakkor folytonos kompatibilitást kíván meg. A kompatibilitásnak az ilyen fájta megkövetelése az evolúciós fejlődés különböző formáiban mutatkozik meg. Az Intel '86 család például olyan tulajdonságokon alapszik, amely a család egyik tagjáról a másik tagjára vándorolva „átöröklődik" (8086, 80186, 80286, 80386). Új funkciókat adnak hozzá az előző verziókhoz képest, de a „konfliktus" megakadályozására az előző tulajdonságok „tetejére" vagy azok „mellé". Mivel az akkori áramköri technológiai sűrűségek még nem tették lehetővé egy 32 bites processzor implementálását, a 16 bites Z8000 először a Z80,000 lefelé kompatibilis családtagjaként épült

2. táblázat

Adattípusok

MC68020	i80286
Bit	Egész
Bit mező (1—32 bit hosszú)	Sorszám (Előjeles bináris)
BCD (Pakolt vagy nem pakolt	Mutató (Szegmens kiválasztó + eltérés)
Egész (Előjeles vagy előjel nélküli)	Lánc (1—64 K byte)
Lebegőpontos	ASCII
	BCD (Pakolt vagy nem pakolt)
	Lebegőpontos (Előjeles 32, 64 vagy 80-bytes valós szám)

NS32032	Z80,000
Bit	Bit
Bit mező (1—32 bit)	Bit mező (1—32 bit)
Egész (Előjeles vagy nélküli)	Egész (Előjeles vagy előjel nélküli)
Pakolt BCD	Pakolt BCD
Veremtár	Veremtár
Láne	Lánc
Lebegőpontós	Lebegőpontos

meg. A National Semiconductor NS32000 (azelőtt NS16000) családjának tagjai 8 bites (NS32008), 16 bites (NS32016) és 32 bites (NS32032) busz interfész egységekkel jelentek meg. A Motorola MC68000 család (68000, 68008, 68010, 68020) újabb tagjainál új funkciókkal és címzési módokkal találkozunk, de az alap utasításkészlet közös. Egy sikeres miniszámítógép kompatibilitása az egy chipes VLSI verzióval őrizhető meg (pl.: μVAX: egy chipes VAX—11 mikroprocesszor).

A 2. táblázatban összefoglaltuk a legfontosabb szupermikroprocesszorok által támogatott adattípusokat.

3. Az operációs rendszer szint architektúrája

Fő kérdések:

— tár szervezés,
— tár kezelés,
— virtuális tár szervezés,
— virtuális tár kezelés,
— folyamat kezelés.

Tár szervezés

Jelenleg a tipikus magasszintű mikroprocesszor alkalmazások memóriaigénye közeledik a mini vagy a nagyszámítógépekéhez is. Ezért van az, hogy a mikroszámítógép architektúrájának egyik legfőbb jellemzője a memória szervezés.

A főbb szempontok a következők:

— a teljes tár architektúra (hogyan néz ki a memória a számítógép programok számára),
— tár leképzés: logikairól fizikai címre fordítás (a tár logikai szerkezetének leképzése fizikai vagy valódi címtartományra, azaz hardverre),
— hozzáférés vezérlés (hozzáférési jog, tárvédelem),
— virtuális tármechanizmus (a fizikai tár méret korlátaitól való megszabadulás).

Tár architektúra

A számítógép fő (fizikai) tárára úgy hivatkoznak és azt úgy szervezik, mint egymást követő tárhelyek (cellák) halmazát. A fizikai címtartományt (más szóval az összes fizikai címek halmazát) a tár rendszer hardvere határozza meg.

A logikai cím azonban egy utasításban kerül meghatározásra és azt a programozó használja. A logikai címtartomány a logikailag lehetséges címek halmaza és az határozza meg a tár architektúrát.

A tár architektúrának két alapvető típusa van: a lineáris és a struktúráit. A lineáris címtartományban a címek a nulla helyen kezdődnek és lineárisan folytatódnak egy felső korlátig, amelyet a logikai cím összes bitjének a száma határoz meg. A szegmentált címtartomány alapjában véve kis lineáris címtartományok összessége. Vannak rendszerek, melyek változó méretű szegmenseken alapulnak és olyanok, melyek állandó méretű egységeken, amelyeket lapoknak nevezünk. A két megoldás kombinációjával találkozunk a szegmentált és lapszervezésű rendszerekben.

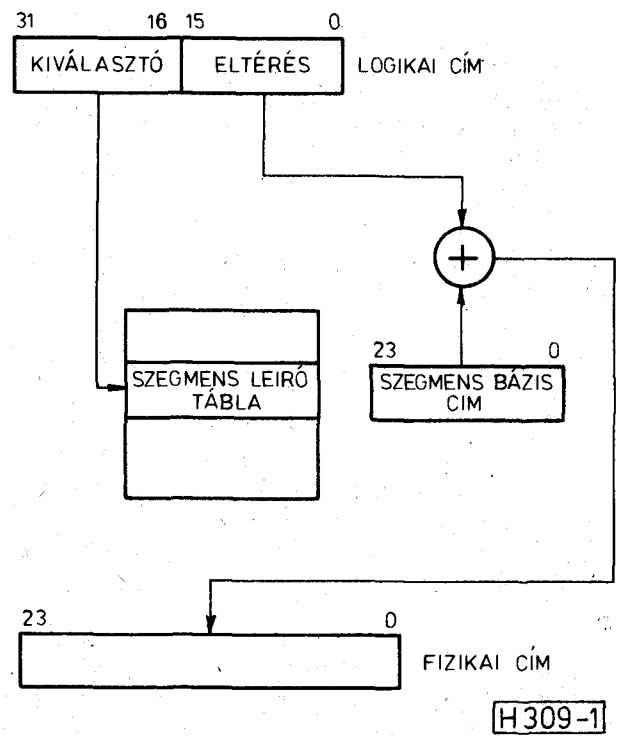
Tár leképzés

A logikai címek fizikai címekkel való megfeleltetésére különböző módszereket dolgoztak ki. A táblázat vezérelt címszámításban a felhasznált táblázatok számát a közvetettségi szint határozza meg. Ez a direkt leképzési eljárás felgyorsítható úgy, ha a teljes leképző táblát (vagy a tár méret korlátai miatt csak egy részét) egy asszociatív tárba helyezik. Ezt a módszert asszociatív leképzésnek nevezik. Mivel az asszociatív tár mérete korlátozott, a direkt és az asszociatív leképzés kombinálható.

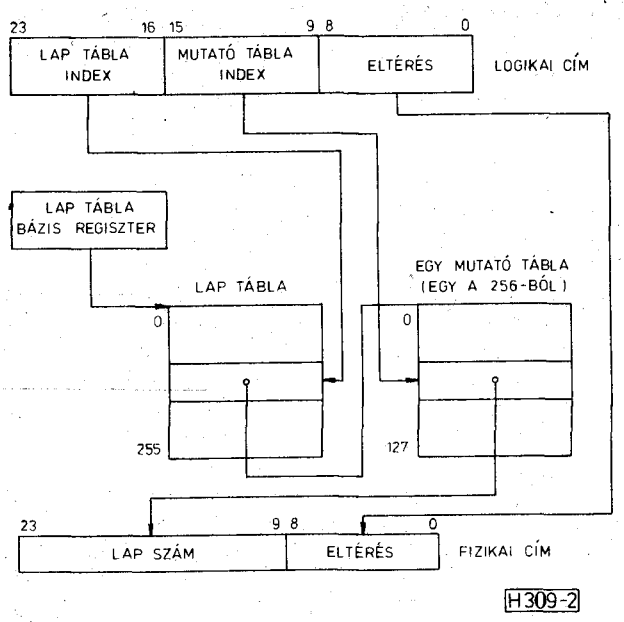
A memória leképzést az összes szupermikroprocesszor támogatja átlátszó, dinamikus, automatikus logikai-fizikai címfordítással. Vannak beépített (a CPU chipen lévő) megoldások, míg más rendszerek kiegészítő (tár kezelő) feldolgozó elemeket használnak.

Az Intel 80286 mikroprocesszor [28] tárát változtatható hosszúságú (max. 64 Kbyte) szegmensek halmazaként szervezték, A két komponensű cím egy szegmens választóból és egy eltérésből áll Az ún. valódi címzési mód kompatibilis a 8086 mikroprocesszor címzési elrendezésével. Ennél a módszernél a processzor 20 bites fizikai címeket generál úgy, hogy egy 20 bites szegmens bázis címet (az alacsony helyértékű négy bit mind nulla) ad össze egy 16 bites eltéréssel.

Védett cím üzemmódban a fizikai cím az 1. ábra szerint számítódik ki. A táblázatokra automatikus hivatkozás történik mindenkor, amikor a szegmens regiszterbe egy kiválasztó töltődik. Négy szegmens regiszter van: kód, veremtár, adat és extra. A szegmens leírók automatikusan betöltődnek egy szegmens leíró cachbe, amikor a megfelelő szegmens regiszterbe egy kiválasztó töltődik. Betöltés után a szegmens összes hivatkozása a



1. ábra. Intel 80286 védett tár címzés

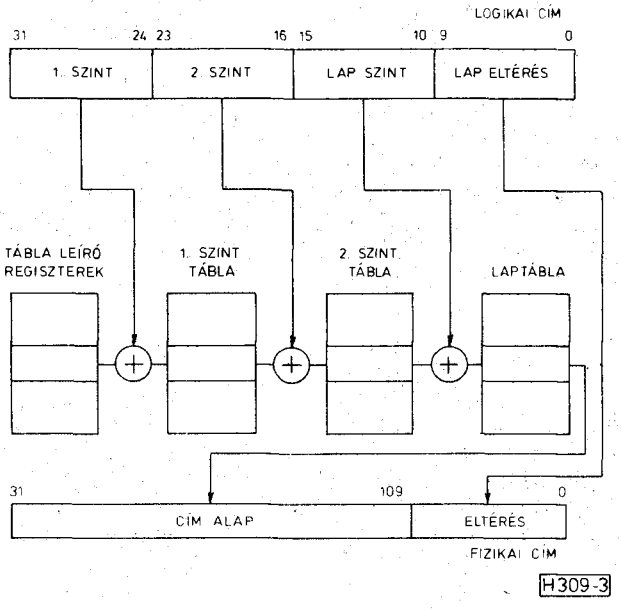


2. ábra. NS32032 címfordítás mechanizmusa

tárolt leírót használja. Ez a programok számára egy átlátszó (nem látható) folyamat. Az NS32032-nek [26] két üzemmódja van (ezeket az egyik lábra adott jellel lehet kiválasztani): cím fordítással vagy anélkül. A cím fordítást egy külön tok valósítja meg (tárkezeló egység: MMU: NS32082), amely szolga processzorként működik a mester központi egységgel. A két egység közötti kommunikációs protokoll a felhasználó számára átlátszó. A rendszer lap-szervezést használ. Mind a logikai, mind a fizikai címtartomány lapokra van osztva (a lap rögzített mérete 512 byte). A címfordítást a memóriában lévő két táblázat végzi: *Lap* táblázat és *Mutató* táblázat. A tárkezelő egység (MMU) két alap laptáblázat regisztert tartalmaz (PTBO a felügyelői program és BTB1 a felhasználói üzemmód számára). Az aktuális PTB az aktuális laptáblázat elejére mutat. A leképzést a 2. ábra vázolja. A Z80,000 [29] mind a szegmentációt, mind a lapszervezést támogatja a CPU-tokon belül lévő tárkezelő egységekkel (harmadik címreprezentációként találkozhatunk a kompakt címzéssel, amely egy 16 bites címzési üzemmód kis konfigurációk számára). Szegmentált reprezentáció esetén a címek vagy egy 15 bites szegmens számból és egy 16 bites szegmens eltérésből állnak, vagy pedig egy 7 bites szegmens számból és egy 24 bites szegmens eltérésből. A lapszervezés a címtartományt 1 Kbyte -os egységekre osztja (melyeket a logikai címtartományban lapnak, a fizikai címtartományba keretnek (frame) hívnak). A címfordításnak három szintje van, amelyet alkalmazkodó (adaptive) formában is használhatunk. A szegmens táblázat leíró regiszterek (négy regiszter van a rendszer utasításokra, rendszer adatokra, normál utasításokra és normál adatokra való hivatkozásokhoz) programozhatóak, hogy szelektálva átugorhassák az első, vagy az első és a második szint táblázatait (ezáltal csökken a hely, ami szükséges a táblázatok tárolására és csökken a fordításhoz szükséges referenciák száma is). A cím fordítási mechanizmust a 3. ábra tartalmazza. Egy asszociatív tár (fordítási puffer) tárolja 16 olyan lapra vonatkozóan az olyan fordítási információkat, melyekre legutóbb hivatkoztunk. Az MC68010 egy MMU-val együtt (tár kezelő egység: MC68451) 24 bites logikai címeket fordít 32 bites fizikai címekre 1024 byte-os lapokat felhasználva. A cím fordítás kétszintű lap leképzésen alapszik. Az MC68020 [30] egy új társprocesszorral rendelkezik, melyet lapszervezós tárkezelő egységnek neveznek (PMMU MC68851) [31], [32]. A kezdeti verziója makrocella tömb alapon készült ós tárkezelés vezérlő a neve: MC68461 [33]. Rendelkezik egy gyors, 64 bemenetű, teljesen asszociatív cache tárral, amely a legutóbb használt lap leírókat tárolja. A 80386 [34] kétszintű lapszervezósi mechanizmust foglal magában, amelynek segítségével kezelni tudja a fizika tárat — minden — a logikai tár szegmensen belül. Logikailag azonos elődeivel, de nagyobb szegmensekkel, fizikai alapcímekkel és méretkorlátokkal rendelkezik [35]. A 3. táblázat az imént tárgyalt szupermikrocsaládok tár architektúráinak főbb tulajdonságait összegzi.

3. táblázat

Típus	Lineáris	Szegmentált	Lapszervezés
8086	×		
80186	×	×	
80286	×	×	
80386	×	×	×
Z80	×		
Z800	×	×	
Z8000	×	×	
Z80,000	×	×	×
NS32008	×		×
NS32016	×		×
NS32032	×		×
MC6800	×		
MC68000	×	×	
MC68010	×	×	
MC68020	×	×	×



3. ábra. Z80,000 címfordítás mechanizmusa

Védelem

Egy számítógép architektúra három fő területen tud támogatni védelmet, úgy mint:

— tár védelem,
— program védelem
— a felhasználó védelme.

Vannak olyan rendszerek, melyek a védelmi szintek egy hierarchiáját valósítják meg (a legjobban privilegizálttól a legkevésbé privilegizáltig). Ezeket a szinteket gyakran gyűrűnek is nevezik. A védelmi tulajdonságok a központi egység üzemmódjaival támogathatók. Felügyeleti üzemmódban például a teljes utasításkészlet rendelkezésre áll, de felhasználói üzemmódban ennek csak egy korlátozott részhalmaza.

A memória védelem hozzáférés vezérléssel valósítható meg a tár leképzés mechanizmusában. A fő kérdés az, hogy mekkora legyen a rendszer által nyújtott védelem „szemcsézettsége". A lapszervezésű rendszerekben a hozzáférés vezérlés természetes egysége a lap. Szegmentált rendszerekben ez az egység a szegmens, míg a kombinált lapszervezésű szegmentált rendszerekben ezek kombinálhatók. A lap alapú rendszerekben minden egyes feladat számára egy-egy külön címtartomány adható (lapoknak egy gyűjteményét alkotva). Most csak a memória védelemre vonatkozó konkrét megoldásokkal foglalkozunk, a program és felhasználó védelemre a rendszerszint architektúrája című fejezetben térünk ki.

Az NS32000 családban a logikai címtartományon belüli tárvédelem alapját a lap mechanizmus szolgálja. A lap és a mutató tábla olyan jellemzőket is tartalmaz (védelmi bitek), melyek jelzik, hogyan lehet a laphoz hozzáférni (lásd 2. ábra). A védelem értelmezése az üzemmódoktól függ. A felügyelő üzemmódban két védelmi szint van (csak olvasható, írható és olvasható), felhasználói üzemmódmódban három (nincs hozzáférés, csak olvasható, irható és olvasható). Szegmens szerű védelem úgy nyújtható, ha a lapok egy halmazának ugyanazok a jellemzői, míg feladatok közötti védelem úgy érhető el, ha minden feladatnak saját laptáblázat készlete van.

A Z80,000 család a többszintű cím fordítási mechanizmusa bármely szintjén nyújt hozzáférési védelmet. A fordítás során minden szinten van hozzáférési védelem mező. Ha egy védelem kiválasztásra kerül, a többi mező figyelmen kívül hagyható, de a döntés átvihető a következő szintre is. A védelmi szintek (nincs hozzáférés, olvasási hozzáférés, írási hozzáférés, végrehajtási hozzáférés) rendszer és normál üzemmódban határozhatók meg.

A 80286 szegmens leírói (1. ábra) a tár felhasználását határozzák meg. Különböző leíró típusok vannak kód, veremtár és adat szegmensek számára. Biztosít továbbá rendszervezérlő leírókat és vezérlósátadó műveleteket (lásd a rendszerszint architektúrája című fejezetet).

Az adat szegmensek vagy csak olvashatók vagy írhatók olvashatók. A szegmensek két irányban terjedhetnek: felfelé az adatszegmensek és lefelé a veremtárat tartalmazó, szegmensek. Az adatszegmens leíró határ-mezője a szegmens utolsó byte-ját azonosítja és az eltéréseket evvel a határral ellenőrzi. A kód szegmens csak végrehajtás vagy végrehajtás és olvasás lehet.

Az MC68010/68020 hozzáférési privilégiumai felügyelői üzemmódban csak olvasás vagy írás/olvasás, felhasználói üzemmódban: nincs hozzáférés, csak olvasás, írás/olvasás.

Virtuális tárkezelés

Virtuális tár kifejezés alatt általában azt a képességet értjük, hogy egy sokkal nagyobb tár tér címezhető, mint ami a fő tárban rendelkezésre áll. Ha a jelölt hely (mely egy laphoz vagy egy szegmenshez tartozik) nincs a memóriában, egy csere művelet kerül végrehajtásra és az operációs rendszer a hiányzó részt betölti a lemezről.

A különböző szupermikro rendszerek különböző szintű architekturális támogatást nyújtanak a csere (swap), „újraindítás" (abbahagyás ós újravégrehajtás) vagy „folytatás" implementálására. Az NS32000 családban az MMU (tárkezelő egység) ellenőrzi a bejövő logikai cím által specifikált státusz biteket a (lap vagy mutató) táblában. Erre a célra minden tételben három bit áll rendelkezésre. Az érvényes bit azt jelzi, hogy a lap a tárban van-e, a „hivatkozott" bit azt mutatja meg, hogy történt-e hivatkozás (azaz írás vagy olvasás) a lapra, míg a „megváltozott" bit azt jelzi, hogy történt-e írás a lapra. Abban az esetben, ha a lap nincs a főtárban, egy lap hiba generálódik, és egy hardver által generált csapda fogja hívni az operációs rendszert, hogy az egy lapcserét hajtson végre. Evvel egy időben az MMU egy jelet küld a központi egységnek, hogy szakítsa meg az éppen esedékes utasítás végrehajtását. A központi egység automatikusan visszaállítja az összes regiszter tartalmát előző állapotukba. Automatikusan megőrződik a programszámláló, a státusz regiszter, a veremtár mutató és számos más regiszter. Amikor az operációs rendszer befejezte a lapcserét, végrehajt egy csapdából visszatérés utasítást ós a végrehajtás folytatódik a megszakított utasításnál (az összes regiszter visszaáll a régi értékére). A lap és mutatótábla másik két bitje is használható a virtuális tár implementációjának támogatására. A lap cserélési algoritmus implementációjakor a „hivatkozott" bit periódikusan ellenőrizhető és törölhető minden bemeneten és ezáltal bepillantást nyerhetünk a lap használati frekvenciára. Lapcserekor a „megváltozott" bit tájékoztatja az operációs rendszert, hogy szükséges-e a lapról egy felfrissített másolatot a mágneslemezre küldeni.

Az MC68010/68020 processzor az utasítás folytatás módszerét használja a virtuális tár implementálására. Amikor hozzáférési hiba generálódik, a gép teljes állapota (minden, a felhasználó által látható, mind az általa láthatatlan rész) megőrződik. Amikor a hibakezelő rutin befejezte munkáját, a processzor ugyanannál a mikroutasításnál (az adott utasításon belül) folytatja az utasítás feldolgozást, mint ahol azt a hiba hatására felfüggesztette. A Z80,000 az utasítás újrakezdés módszerét használja. Minden egyes táblázat helyen négy bit van az operációs rendszer támogatására:

hivatkozott, érvényes, változtatott ós cache-be nem küldhető. Hasonló célokra a 80286-nak két bitje van a szegmens leíró táblázatban: „jelenlevő" bit és „hozzáfért" bit. Bármely olyan címnek a használata, mely nem alakítható egy fizikai memória címmé, egy újraindítható kivételt hoz létre.

A felhasználó szemszögéből nézve az utasítás újrakezdés és utasítás folytatás módszerek egyformák, azonban az implementáció szintjón vannak különbségek. Az újrakezdés módszer feltételezi, hogy a processzor képes helyreállítani, vagy újrakonstruálni a gép utasítás előtti állapotát.

4. A programozási nyelv szint architektúrája

Fő kérdések:

— utasítás formátum és kódolás,
— támogatott magas szintű adattípusok,
— operátorok ós utasításkészlet,
— regiszter készlet,
— HLL címzési módok,
— RISC kontra CISC.

Magas szintű nyelvek támogatása: RISC kontra CISC

Jelenleg a mikroprocesszor architektúrákban két, igazán eltérő módon nyújtanak anrhitektúrális támogatást a magas szintű nyelveknek. Ugyanakkor céljuk ugyanaz vagyis, hogy jobb teljesítményt nyújtsanak mind a magas szintű programozásban, mind pedig az assembler nyelv szintjén. Szinte az összes most megjelent szupermikroprocesszor típus a CISC (Complex Instruction Set Computer: összetett utasításkészletű számítógép) kategóriába tartozik. Az összetett utasításkészlet a magas szintű nyelv támogatásának „irányába" bővült új funkciókkal és architekturális jellemzőkkel. A RISC (Reduced Instruction Set Computer: csökkentett utasításkészletű számítógép) a következő pontokban összegezhető ötleteken és gondolatokon alapszik:

— Az utasításkészlet összes utasítása „primitív", hogy azok egyetlen gépi ciklusban végrehajthatók legyenek.

— Az „egymásba fűzés" (pipelining) az implementáció lényeges elve (az egymásba fűzés sosem szakad meg, ha a késleltetett ugrás elvét alkalmazzuk).

— Az adat hivatkozások túlnyomó többsége memória hozzáférés nélkül valósul meg (nagy regiszter csoport felhasználásával és alkalmazva a „regiszter ablak" elvet az egymást követő eljárások regiszter csoportjainál).

Meggondolva az utóbbi két, a RISC-re nem jellemző tulajdonság előnyeit, nem itélhetjük meg a RISC megközelítés előnyeit egyszerű benchmark programokkal [36]. A HLL programozás támogatására (hogy csökkentsék a szemantikus rést a nyelv szint és a számítógép architektúra között) a szupermikroprocesszorokat magas szintű utasításokkal, rafináltabb címzési módokkal ós adattípusokkal implementálták [37].

HLL adattípusok és címzési módok

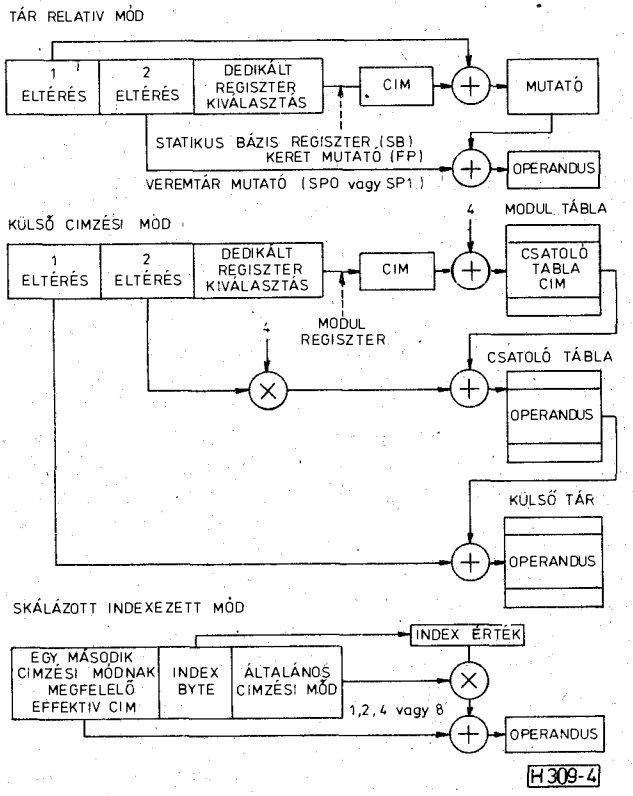
Az NS32032 például az alábbi adattípusokat támogatja:

— egósszámok (előjeles ós előjel nélküli byte, szó, dupla szó),
— lebegőpontos szám (szabványos és hosszú),
— logikai érték (byte, szó, dupla szó),
— BCD (pakolt, pakolatlan byte, szó, dupla szó),
— bitmező (1—32 bit tetszőleges pozícióban),

Az architektúra által támogatott strukturált adat típusok:

— blokk (mozgatás vagy összehasonlítás),
— lánc (mozgatás, összehasonlítás vagy kihagyás),
— tömb (index számítás a skálázott index címzési mód segítségével),
— veremtár (támogatás a verem teteje címzési móddal),
— rekord (a tár relatív címzési segítségével támogatva).

Jellemző példaként a 4. ábrán megadjuk az NS32000 család néhány HLL címzési módját. A tár relatív mód hasznos mutatók kezelésénél és rekordok mezőinek alakításánál (a második eltérés a rekord egy mezőjének az elhelyezkedését határozza meg, melyre egy dupla szó mutat). A külső címzési mód lehetővé teszi modulok áthelyezését csatoló szerkesztő program nélkül. Olyan operandusokhoz való hozzáférésnél használják, melyek külsők az éppen végrehajtott modulhoz képest. Minden egyes modulhoz tarto-



4. ábra. Magas szintű címzési módok az NS32000 családnál

zik egy csatoló tábla. Ez a külső változók abszolut címeit tartalmazza. Ez a címzési mód két eltérést definiál: a külső változó sorszámát és az éppen hivatkozott változó részmezőjétől való eltérést (pl: egy Pascal rekord részmezője). A skálázott index mód kiszámítja az operandus címét az egyik általános célú regiszter segítségével és egy másik címzési mód felhasználásával tömbök elemeinek címzésére használják. A regiszter értéke eggyel, kettővel, néggyel vagy nyolccal szoródik, amikor a tömb elemei byte-ok, szavak, dupla szavak, lebegő pontos számok vagy hosszú lebegő pontos számok.

A RISC kontra CISC kérdés annak alapján dönthető el, hogy egy adott fordítóprogram implementálására mennyire alkalmasak. A hasznos jellemzők a következők [16]:

— Regularítás, amely egy olyan elv, melyet a „legkisebb meglepetés törvényé"-nek is szoktak nevezni.

— Ortogonalítás, ami azt jelenti, hogy az utasítás kódok, a címzési módok és az adattípusok külön tárgyalhatók.

— Összerakhatóság, ami lehetővé teszi azt, hogy az összes jelölés tetszőleges módon használható legyen, azaz minden címzési mód minden operátorral és adattípussal.

Az egyszerű címzési sémájú, nem redundáns operációs kóddal és korlátozott funkcionalitással rendelkező RISC architektúra regularítása jó. A RISC ugyanazokkal az argumentumokkal ortogonális, de az összerakhatóság éppen ellentétes a RISC elvével. Nyilvánvaló, hogy egy CISC architektúra sokkal kevésbé reguláris, mint egy RISC architektúra (mivel egy operandus sok különböző módon címezhető és sok olyan utasítás sorozat van, amellyel egy komplexebb „primitív" funkciót meg lehet valósítani). Az ortogonalítás és az összerakhatóság mértéke a CISC architektúra definíciójából adódóan magas. A RISC a neki tulajdonított jó teljesítményt (gyorsaságot) a vezérlési folyamatot irányító nagyon kis számú döntéssel éri el. Ugyanakkor az összes mennyiségi jellemzőt alacsony nyelvszintű benchmark programok eredményeiből vezették le. Természetesen nincsen elegendő tapasztalatunk a mikro CISC gépekkel kapcsolatosan sem. Figyelembe véve azonban a hasonlóságokat, a VAX 11/780 utasítás használatának tanulmányozása, mérése és elemzése bizonyos kezdetleges nagyságrendi információt nyújthat mind a fordítóprogram végrehajtására, mind a magas szintű programozási nyelvekkel történő felhasználásokra vonatkozóan. Van egy másik mód, ahogyan áthidalható a HLL és a számítógép architektúrája közti szemantikus rés. Ez az ún. magas szintű nyelv architektúra, vagyis amikor a számítógép architektúráját a HLL szintjére emeljük [38]. Ennek a kategóriának egy VLSI verziója egy olyan Pascal processzor, mely közvetlenül hajtja végre a UCSD P kódot [39]. A jövőbeli HLL gépek implementálására mikroprogramozható mikroprocesszorok lesznek használhatók (pl: AM29116 vagy NCR/32 család).

5. A rendszer szint architektúrája

Fő kérdések:

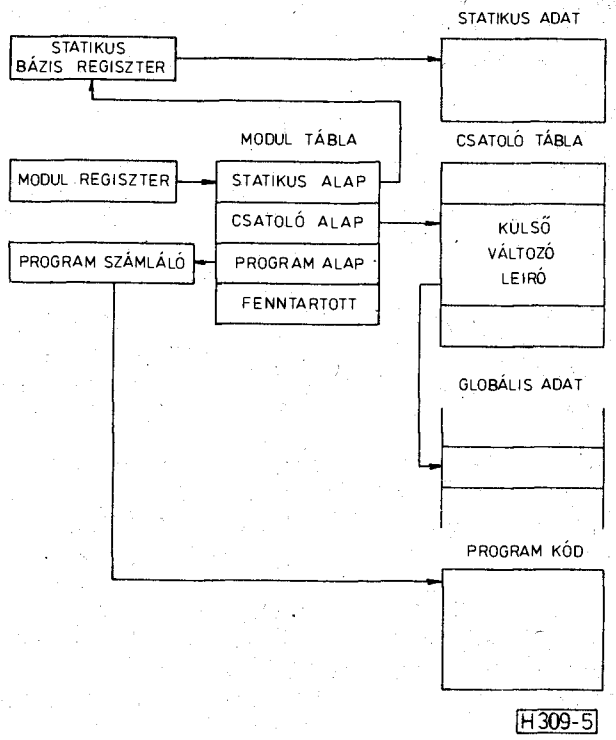
— moduláris szoftver támogatása
— több folyamat feldolgozásának támogatása
— hiba keresés- és javítás támogatása
— többprocesszoros rendszerek támogatása

Moduláris programozás támogatása

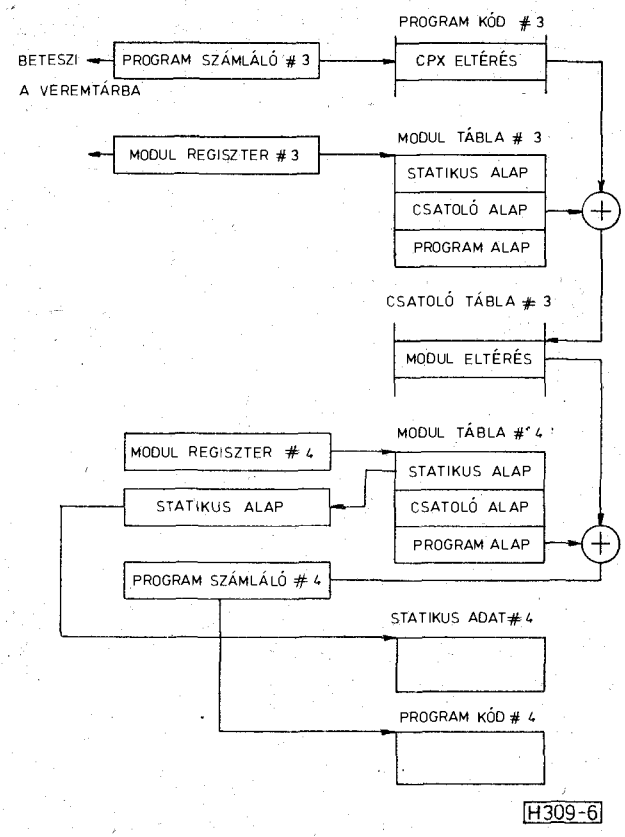
A szoftver megbízhatatlanságok egyik fő okozója a programok összetettsége. A program összetettség kezelésének legfontosabb módszere a moduláris programozás, mely az összetett feladatot alfeladatokra, más szóval modulokra osztja. A moduláris programozást sok magas szintű nyelv támogatja (pl: Pascal, Ada), és a szupermikroprocesszorok is tartalmazzák a moduláris szoftver bizonyos szintű architekturális támogatását. Például az NS3200 családban minden modul három elemből épül fel [37]:

— A program kód pozíció független kódot tartalmaz, mely konnyen áthelyezhető és összekapcsolható (pl: ROM könyvtárak felhasználásával).
— A statikus adat elem a modul globális változóit és adatait tartalmazza (amelyekhez a modulban lévő összes eljárás hozzá tud férni)
— A csatolói tábla a külső változó és eljárás leírókat tartalmazza (a külső címzési mód és a külső eljárás hívás utasítás használja).

A támogatott modul környezetet az 5. ábra mutatja. A modul leíró négy (32 bit-es) értéket tartalmaz a modul minden eleméhez. Betöltés előtt a modulokat nem kell csatolni, de egy csatoló betöltő egyszerűen aktuális értékekkel láthatja el



5. ábra. NS32000 modul környezete



6. ábra. Külső eljárás hívás (CPX) utasítás végrehajtása (NS32000 család)

ós feltöltheti a csatolói táblát is a megfelelő értékekkel. Ha pl: a 3. számú modul akarja hívni mondjuk a 4. számú modult, egy külső eljárás hívás (CXP) kerül végrehajtásra (lásd a 6. ábrát.) A külső eljárás hívás leíróval utasítás lehetővé teszi egy külső eljárás leíró paraméterként történő átadását a hívott modulnak. Egy függvény vagy egy eljárás címe a hívó modul csatolói táblából a veremtárba kerül és ezután a hívó modul használhatja ezt az értéket egy eljárás hívására. A „keret" (frame) mutató regisztert egy eljárás arra használja, hogy hozzáférjen a veremtárban lévő paramétereihez és helyi változóihoz. Az eljárás ezeket az értékeket a keret mutatótól számított rögzített eltérésként címzi. A veremtár változatlanul használható átmeneti tárolásra. A „belépés" és „kilépés" utasítások minimalizálják az eljárás hívások holtidejét. A belépés utasítás megőrzi a hívó modul keret mutatóját a veremtárban és a keret mutató regiszterbe tölti a veremtár mutató értékét. Az eljárás helyi változói számára hely van fenntartva a veremtárban és az eljárás által használt regiszterek a veremtárba olvasódnak. A kilépés utasítás automatikusan visszaállítja a megőrzött erőforrásokat. A visszatérés külső eljárásból utasítás visszaállítja a statikus alapot, a modul regisztert és a hívó eljárás programszámlálóját. A modulokhoz való adathozzáférés a következő módon történik:

— A paraméterek és helyi változók tárolhatók és hozzáférhetők a veremtárban a tár tartomány (Memory Space) vagy a tár relatív címzési módok által (mint a keret mutatótól való pozitív és negatív eltérések).
— Egy modul statikus adataihoz a tár tartomány címzési móddal férhetünk hozzá a statikus alap regiszter felhasználásával (az eltérés mező 1, 2, vagy 4 byte lehet). Mindegyik modul saját statikus adatrésszel rendelkezik. Az architekturális szinten történő védelem nélkülözhetetlen olyan alkalmazásoknál, melyek azt kívánják meg, hogy két vagy több feladat konkurrensen ugyanazt a kódot hajtsa végre (újraindíthatóság biztosítása).
— Olyan operandusok számára, melyek az éppen végrehajtott modul számára külsők, a külső címzési mód használatos.

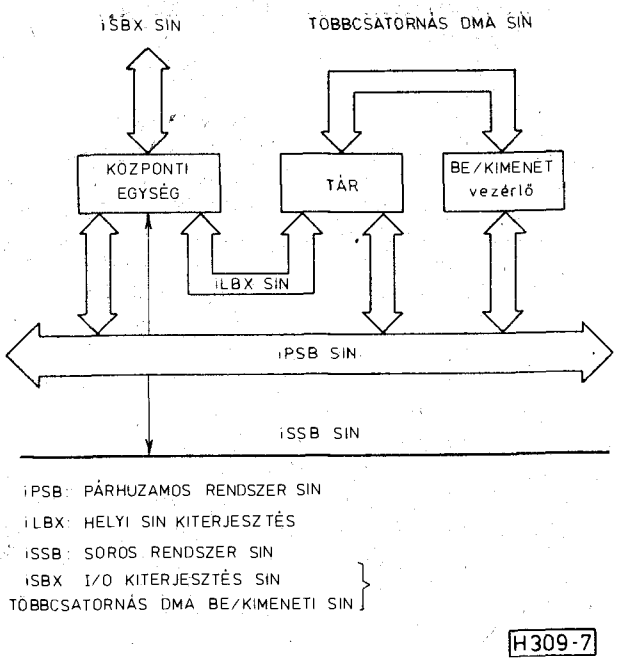
Az Intel 286/386 architektúra támogatja mind a többfelhasználós, mind a többfeladatos rendszereket. A kód és adat szegmens leírókon kívül a védett üzemmód rendszervezérlő leírókat definiál, melyeket hívás, feladat, megszakítás és csapda kapunak (gate) neveznek [29]. Ezek a kapuk közvetettségi szintet biztosítanak a vezérlés átvitel forrása és célja között. A 286/386 négy szintű előjog (priviledge) rendszerrel rendelkezik, amely a felhasználó/felügyelő üzemmód kiterjesztése. Ez vezérli a privilegizált utasításokat, hozzáférés leírókat és a hozzájuk tartozó szegmenseket a feladaton belül. A feladatok elvannak szigetelve különálló leíró táblák segítségével (egy táblázat, melyet globális leíró táblának hívnak, az összes feladat számára rendelkezésre álló leírókat tartalmazza). A négy privilégum szint a következőkre használható:

— 0 szint: operációs rendszer mag (kernel),
— 1 szint: felügyelő program (rendszer szolgáltatások),
— 2 szint: operációs rendszer kiterjesztések (felhasználói, segédprogramok),
— 3 szint: alkalmazások.

Az MC68020 modulok támogatását a modul hívás és a visszatérés utasításokkal végzik. A modul hívás utasítás olyan modul leíróra hivatkozik, mely az adott modulba való belépéshez tartalmaz vezérlő információt. Egy modul veremtár keretet (frame) hoz létre és ebbe tárolja el a modul állapotát. A modulból való visszetérés utasítás visszaállítja az előző modul állapotot a veremtárból és visszatér a hívó modulhoz.

Több folyamat feldolgozása

A mikroszámítógépek világában ma jelentős irányzat a több folyamat (process) egyidejű feldolgozásának fokozott alkalmazása. Ha több processzort is alkalmazunk az első számú fő kérdés a megfelelő architektúra kiválasztása a rendkívül sok MIMD (többszörös utasítás, többszörös adat) ós SIMD (egy utasítás, többszörös adat) architektúra közül. További gondot jelent a lazán csatolt (kommunikációs csatolók által összekapcsolt) és szorosan csatolt (egy sínt, globális vagy elosztott globális tárat használva postaládák segítségével kommunikálva) rendszerek használata. Alapvető kérdés továbbá, hogy a sín struktúrák (egyetlen parallel sín, több-



7. ábra. Mutibus II Sin architektúra

szörös parallel sínek, parallel crossbar sínek, vagy többszörös soros sínek) mester/szolga vagy szimmetrikus szervezés stb. közül mit választunk.

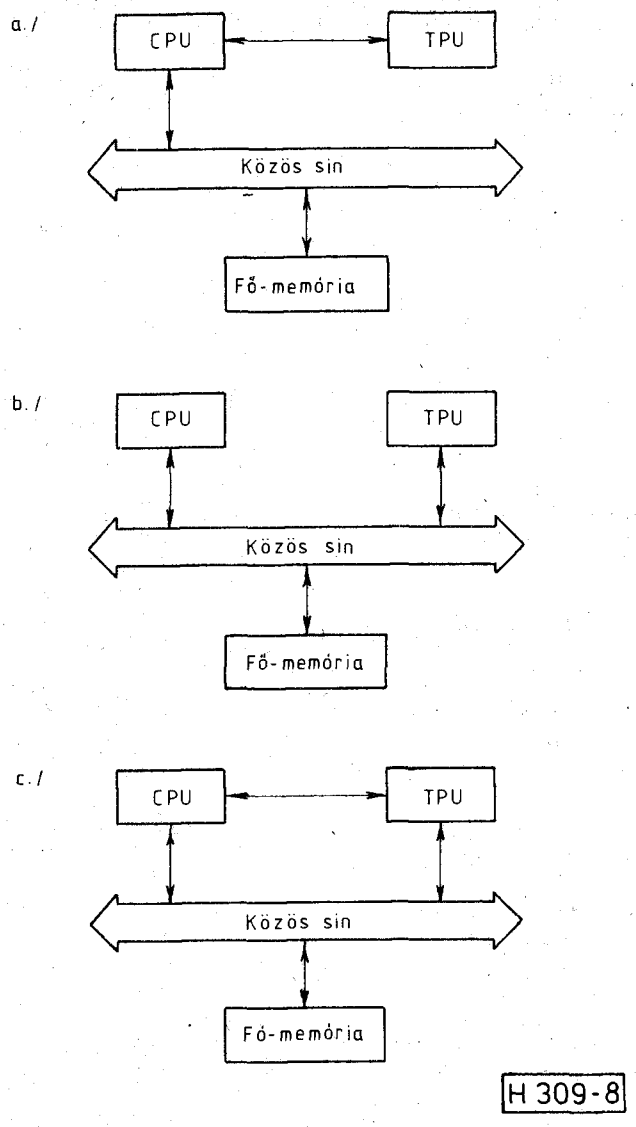
Az összes szupermikroprocesszor fel van készítve a többprocesszoros környezetben történő alkalmazásra és többé kevésbé támogatja is ezt. Minden mikroprocesszor családban vannak áramkörök, melyek a sín versenyhelyzetet oldják fel (pl: Intel 8289 Bus Arbiter), áramkörök a „szabvány" társ- vagy szolga processzorokkal való kommunikációhoz (pl: aritmetikai vagy lebegőpontos feldolgozó egység), DMA (közvetlen tár hozzáférés) átvitel vezérlő vagy egy megszakítás vezérlő. A szupermikroprocesszor családok szerveződhetnek egy szabvány busz architektúra köré, mint az Intel Multibus II (lásd a 7. ábrát) és probléma orientált áramkörök segítségével támogathat kommunikációs protokollokat (pl: Intel LAN: lokális hálózatot támogató tokok a 82586, 82501 Ethernet protokoll számára). Ha több 80286 processzor kritikus erőforrásokat oszt meg (pl: memória,) egy explicit zár (lock) adható bármely utasításhoz — megakadályozandó azt, hogy más potenciális sín mesterek átvegyék a sín vezérlését, mielőtt az utasítás végrehajtás befejeződött. Zár jelek implicit módon a következő helyeken fordulnak elő:

— A leíró táblához való hozzáféréseknél a leíró függetlenségének biztosítására.
— Megszakítás visszaigazolás alatt, amikor a központi egység egy zár jelet ad ki, hogy megakadályozza a kaszkád kapcsolású cím meghajtók és a központi egység cím meghajtói közötti sín versenyhelyzetet egy több mesteres rendszerben.
— A csere (XCHG) utasítás alatt, amely kiválasztott központi egység regiszterek tartalmát cseréli ki a táron belül. (Ez kulcsfontosságú a szemafórok implementálásához).

Az MC68000 család „ellenőrizd és állítsd be" utasítása (TAS) lehetőséget ad az erőforrás védő mechanizmusok implementálására szemafórok segítségével. A TAS utasítás számára a processzor egy megszakíthatatlan táblázat olvasás-változtatás-írás szekvenciát biztosít, mely alatt a sín nem vehető el (ha a hozzáférés alatt egy tár hiba generálódik, a teljes olvasás-változtatás-írás ciklust újra kell kezdeni). A Z80000 központi egysége olyan státusz kóddal rendelkezik, amely jelzi, hogy ez olyan hivatkozás, mely cache-re nem tehető és védve van. A központi egység védelmet (intelock) használ az összekapcsolt csökkentés, összekapcsolt növelés utasításokkal összefüggő adat hivatkozásokra. A központi egység ugyancsak védelmet használ a cím fordítási táblára történő hivatkozásokhoz, amikor a fordítás puffer (Translation Lookaside Buffer) feltöltését végzi.

Többprocesszoros konfigurációk

Minden szuper mikroprocesszor támogatja a több processzoros konfigurációk közti kapcsolatot. Az egyszerű feldolgozó egységek egy (vagy több) kiegészítő feldolgozó egységet használnak, melyet társprocesszornak hívnak. Ezek a központi egységgel összhangban működnek. Ez a megoldás olyan architekturális képességeket biztosít, melyek — figyelembe véve a mai elemkészítési technológiákat — másképp nem lennének elérhetők. A központi egység és a társprocesszor közti kommunikáció egy gyors, jól meghatározott protokoll útján történik, mely a felhasználó számára átlátszó. A különböző rendszerek a társprocesszor (csatolt processzor) elvet különféle módon implementálják (8. ábra). A processzor kiterjesztés interfész fenntartott 1/0 port címeket használ az Intel „processzor kiterjesztés" átviteleknél. A központi egység a tár kezelő és védelmi képességeket a processzor kiterjesztésekre is értelmezi. Meghatározza az operandus kezdő címét és az utasítás irhatóság/olvashatóság státuszát. A 8287 numerikus processzor kiterjesztés (NPX) például ezt az interfészt használja és numerikus számításokat, valamint adatátvitelt hajt végre a központi egység program végrehajtásával egyidejűleg. A Zilog „kiterjesztett feldolgozási architektúrájá"-ban (EPA) egy Z80000 alapú rendszerben max. négy további feldolgozó egység (EPU) használható [29]. A Z8070 aritmitikai feldolgozó például ezt a módszert használja. A Motorola MC68020-ban a „társprocesszor átviteli interfész" ugyancsak megengedi, hogy a társprocesszor (pl: MC68851 lebegőpontos társprocesszor) [32] lebegőpontos utasításokat hajtson végre a központi egységgel konkurrensen. A társprocesszor perifériának használható olyan főprocesszor esetén, mely nem rendelkezik társprocesszor interfésszel (pl: MC68000, MC68010). Ez olyan utasítás sorozatok alkalmazásával érhető el, melyek a központi egységben lévő regiszterek egy csoportjának á felhasználásával megfelelően emulálják a társprocesszor interfész protokollt. A National Semiconductor protokoll esetében a központi egység előre elővesz utasításokat, a sorba állítja azokat és utána vár,



8. ábra. Társprocesszor (TPU) architektúrák
a) Közvetlen csatolás (speciális vonalak, megosztott regiszterek)
b) Csatolás a közös sín és felhasználásával
c) Csatolás *a* közös sín és speciális vezérlő vonalak segítségével

amíg a társprocesszor (pl: NS32081 lebegőpontos egység vagy NS32082 memória kezelő egység) befejezi a végrehajtást. A felhasználó által is definiálhatók társprocesszorok és ezek a jól meghatározott utasítás formátumok, operandus osztályok és kommunikációs protokollok felhasználásával implementálhatók. Ennek a társprocesszor fogalomnak két fő előnye van: amikor az integrációs technológia szintje eléri azt a pontot, a csatolt processzor hardver a központi egység egy tokban fog helyet kapni, nem lesz szükség szoftver változtatásra. A másik előny, hogy a felhasználónak megvan a lehetősége társprocesszor nélküli rendszer építésére szoftver emulátorok felhasználásával. Ebben az esetben a központi egység „csapdát állít" a társprocesszor utasításainál. Később nagyobb teljesítményű rendszerek építhetők különleges chipek hozzáadásával és egyidejüleg az emulátorok eltávolításával.

A szolga processzorok dedikált funkciókat tudnak végrehajtani a központi egységgel aszinkron. A központi egység ós a szolga processzor megoszthat egy helyi sínt, ahol a központi egység a mester. A szolga processzor egy tipikus példája a közvetlen tár hozzáférés (DMA) vezérlő (pl: Zilog Z8016).

A szorosan csatolt többszörös központi egységek független utasítás (vagy) adatfolyamot hajtanak végre és általában egy közös (globális) sínen el helyezkedő megosztott memórián keresztül kommunikálnak alaphelyzetben. Minden egyes központi egység csak a saját sínén mester. Egy külső sín döntőbíró választja ki a globális sín mestert. Egy kettős Port Interfész (pl: Intel 8207 kettős port DRAM vezérlő) kettős hozzáférésű vezérlést biztosít a globális és a helyi sínek között, és ugyanakkor dönt a helyi ós globális sínkérésekről. Számos szabvány ós gyártói sín (pl: multibus, Z-bus, Versa-bus, VME bus, Nubus, Future Bus) használható többprocesszoros rendszerekhez ós ezeket független gyártók kártya szintű termékei támogatják. A szorosan csatolt multiprocesszor rendszereknek számos hátránya van. Mindenek előtt, minden egyes hozzáadott processzor a rendelkezésre álló sín sávszélesség egy egyre kisebb százalékáért verseng, amíg semmi sem marad. Ha az átlagos utasítás végrehajtási idő közel van az átlagos sín ciklusidőhöz, az első mikroprocesszor a rendelkezésre álló sín sávszélesség nagy részét fel fogja használni. A teljesítmény nyilvánvalóan vagy egy hierarchikus tár séma hozzáadásával növelhető, vagy pedig a sín interfész tágításával.

A processzorok egy lazán csatolt hálózata esetén az egész kommunikáció egy több hozzáférésű periférián keresztül történik (pl: Zilog Z8038 FIFO: „elsőnek be elsőnek ki" I/O vezérlő esetén) vagy soros kommunikációs csatornákon keresztül üzenetekkel. Ez a topológia akkor működik különösen jól, amikor a megoldandó probléma nagy mértékben paralellizált és kevés kommunikációra van szüksége.

Hibakeresés és javítás támogatása

Néhány szupermikroprocesszor architektúra a hibakeresést és javítást igen erősen támogatja. A rendelkezésre álló primitívek a program írójának dolgát megkönnyítik, ugyanakkor javítják a felhasználó hibakereső és javító környezetét. A programozási hibák elszigetelésének és javításának architekturális támogatása ugyancsak elősegíti a hibakeresést és javítást. Az NS32032 moduláris szoftver támogatása nem teszi tönkre az alkalmazási szinthez illeszkedő szerkezeti információt és amely ezáltal segít a progléma vezérlésének és adat áramlásának követésében (pl: a nem primitív adatstruktúrák és komplex műveletek, mint például eljárás hívások függőségi viszonyainak helyreállítása, valamint a fontos környezeti vagy szinkronizációs műveletek ós az ezekkel járó hatások).

Az NS32000 család architektúrája támogatja a töréspont meghatározást és a végrehajtás követést. A töréspont csapda (BPT) utasítás kicseréli az

első byte-ot annak az utasításnak a művelet kódjában, amelyet töréspontként megakarunk határozni. Azért, hogy PROM-okban töréspontok beállítása lehetséges legyen, két töréspont regiszter áll rendelkezésre. Minden tár ciklusban a kiválasztott megszakítási pont címe és a cím sín tartalma összehasonlításra kerül. Ha más meghatározott feltételek teljesülnek (a jelzett cím ír és olvas és vagy utasítás elővétel történik) egy nem maszkolható megszakítás generálódik. Mivel ezek a regiszterek a tár kezelő egységben (MMU) vannak, úgy választhatók, hogy vagy a logikai címekre vonatkoznak (a központi egységből), vagy pedig a fizikai címekre (az MMU-ból). A töréspont számláló regiszter az első megszakítási feltételnek eleget tevő egyezések számát határozza meg (interációk egy hurokban) mielőtt egy töréspont jön létre. Az MC68000 család lapszervezett társszervező egysége (PMMU: MC68851) hasonló módon támogatja a töréspont kezelést. Amikor a központi egység (pl: 68020) egy töréspont utasítással találkozik, egy töréspont visszajelzés ciklust hajt végre, a központi egység címtartományába eső adott címet olvasva. A PMMU ezt a címet dekódolja és válaszol egy helyettesítő műveleti kóddal a töréspont számára, vagy pedig sín hiba közbeiktatásával azért, hogy ez utóbbi által illegális utasítás feldolgozást kezdeményezzen. Programozható, hogy a helyettesítő műveleti kódot n-szer hozza létre mielőtt egy kivételt jelez. A Z80,000 központi egysége a töréspont kezelést töréspont utasítással támogatja (BRKPT). Töréspont csapda egy BRKPT utasítás végrehajtáskor jön létre ami egy hibakereső és javító vagy pedig egy ellenőrző programot hív „segítségül".

A végrehajtás követést négy dedikált regiszter (az MMU-ban) és egy „követési csapda bit" a központi egység program státusz regiszterében) támogatja az NS32000 család esetében. A két program követési regiszter az utoljára nem sorban végrehajtott két utasítás címét tartalmazza. Két szekvenciális számláló regiszter tartja nyilván a program végrehatás két változása közti sorrendben lévő utasítások számát. Ha a követés bit be van állítva, az utasítás végén egy követési csapda generálódik. A kivételek közül a követési csapda prioritása a legkisebb, következésképp bármely más csapda vagy megszakítás kérés befejezése megengedett. Az MC68020 követés úgy viselkedik, mint egy nagyon nagy prioritású, belül generált megszakítás minden utasítás végrehajtás után.

6. A mikroprogramozási szint architektúrája

Fő kérdések:

— mikroprogramozás alkalmazások a chipek implementációiban,

— felhasználói szintű mikroprogramozás,

— az utasítás készlet alakítása,

— vertikális migráció,

— bit szeletelt és állandó bit számú mikroproeeszszor családok.

Szinte az összes kereskedelmi forgalomban kapható szupermikroprocesszor belül mikroprogramozva van. A mikroprogramozás előnyei sokrétűek: világosabb architektúra és szerkezet, könnyebb változtatás és ellenőrzés, ezáltal gyorsabb kifejlesztés, jobb teljesítmény. Figyelembe véve a mikroprocesszor felhasználójának szemszögét több mód ós szint van, ahol a felhasználó mikroprogramozás segítségével javítani tud, az architektúrán. A felhasználó által mikroprogramozható mikroprocesszorokban (pl: Texas TMS-7000 család) a mikrokódolás arra használható, hogy alakítsunk az utasításkószleten, ezáltal az adott felhasználás igényeinek hatékonyabban tegyünk eleget (az alakítás azt jelenti, hogy az alap utasításkészlet adott utasításai helyett egyéni, a felhasználó által meghatározott utasítások definiálhatók más, az ún. alap (core) utasításkészletet alkotó utasítások nem változtathatók. Alapvető jellemzők, mint gyorsaság ós programmóret nagy mértékben javítható mikroprogramozás segítségével. Egy másik tényező az algoritmus biztonsága. Ezen kívül a mikrokódolás csökkentheti a rendszerben szükséges járulékos áramköröket. Egy felhasználó által mikroprogramozható mikroprocesszszor család (pl: NCR/32, mely CPC: központi feldolgozó, ATC: cím fordító, STC: rendszer interfész, EAC: bővített aritmetika és VAC: virtuális tárkezelő áramkört tartalmaz [43]) különböző szintű architektúrák megvalósítására (például hagyományos architektúrák: IBM System 370 emulációjára vagy implementációjára, a vertikális migráció módszereinek felhasználásával operációs rendszer primitívek támogatására és implementálására) használható.

A vertikális migráció lehetséges céljai: teljesítmény növelés, az architektúra hatókörének kiterjesztése, strukturális regularítás, és a tulajdonjog védelme. Egy felhasználó által mikroprogramozható mikroprocesszor hatékonyan használható virtuális „magas szintű nyelvű gép" közvetlen mikrokódban történő implementálására (egy vituális Cobol gép például csak 25 Kbyte-ot használ [16]), vagy egy közepes (pseudo) gépi szint (pl: Pascal P-gép) [39] mikrokód segítségével történő megvalósítására. Ezekre a célokra bit szeletelt mikroprocesszorok is használhatók. Ugyanakkor a bit szeletelt (vagy napjainkban byte szeletelt) mikroprogramozható processzor családok (pl: AM29500 család) [42] segítségével speciális processzor architektúrákat tudunk implementálni (pl: tömb processzorok digitális jelfeldolgozó processzorok, komplex aritmetikát igénylő algoritmusok megvalósítása, mint FFT: gyors Fouriertranszformáció, konvolúció, korreláció stb.). A mikroprogramozható mikroprocesszorok (pl: AM 29116) [44] architektúrája és utasításkészlete a nagy teljesítményű periféria vezérlők, mint például grafikus ós kommunikáció vezérlő igényeihez igazítható (bit orientált utasítások, CRC: ciklikus redundancia ellenőrzés generálása stb.). Utasítás sorrend képzőjük olyan megoldásokat tartalmaz, mint többirányú elágaztatás, egymásba ágyazott mikroszubrutinok és mikrohurok vezérlése stb.

7. A digitális logikaszint architektúrája

Fő kérdések:
— hardver interfész vezérlés,
— konfiguráció vezérlés,
— átlapolt működés (pipelining).

Hardver interfész vezérlés

A szupermikroprocesszorok hardver konfigurációinak adott jellemzői különféle módon határozhatók meg. A Z80,000 hardver interfész vezérlő regiszter (HICR) a sín sebességét, a memória adatút szélességét, az automatikus várakozó állapotok számát, globális sín protokoll vezérlést és a központi egység és a társprocesszor átlapolt működését szabhatja meg. Az NS32032 konfiguráció regiszter (CFG) adott külső eszközök (pl: vektoros megszakítás, lebegőpontos aritmetika, tárkezelés, egyedi processzor) jelenlétéről adhat információt a konfigurációt beállító utasítással. Ezen kívül megszabhatja a sín időzítési módot (cím fordítással vagy anélkül). Az MC68020 teljes, 32 bites adat sínnel rendelkezik, de sínét automatikusan ós dinamikusan le tudja csökkenteni ciklusonként 8 vagy 16 bitre (DSACK).

Átlapolt működés

Az átlapolás (pipeling) a teljesítmény növelés egyik eszköze. A Z80,000 központi egysége például egy 6 fokozatú szinkron ,,csővezetékként" van tervezve. Az utasítások sorban haladnak végig a csővezeték minden fokozatán (egy lehetséges kivétel az operandus tárolás fokozata):
— Az utasítás elővétel fokozat megnöveli a programszámlálót és utasítás elővételt kezdeményez.
— Az utasítás dekódolás fokozat az utasítást dekódolja (meghatározza az utasítás hosszot és beállítja a cím számítás vezérlését).
— A cím számítás fokozat egy (közvetlen) operandust kezel vagy egy operandus címét generálja.
— Az operandus elővétel fokozat tár olvasást kezdeményez az operandusok számára.
— A végrehajtás fokozat az utasítás végrehajtását végzi.
— Az operandus tárolás fokozat a feltétel jelzőbiteket állítja be és adatokat tárol a memóriában.

Az NS32032 8 byte hosszú utasítás sorral rendelkezik. A központi egység az utasítás olyan sorrend szerint következő szavát olvassa be az utasítás sorba mindenkor amikor a busz egyébként állna, és a sor még nincs tele (szekvenciális utasítás elővétel). Egy ugrás, elágazás vagy csapda hatására a központi egység előveszi a következő utasítást, miután az utasítás sort kiürítette (nem szekvenciális utasítás elővétel).

Egyszerre három egymást követő utasítás dolgozható fel, mert míg az egyik utasítás végrehajtódik, a következő az utasítás regiszterbe töltődik be és a harmadik a soron halad tovább. A 80286 négy rész egyidejű (átlapolt) működését biztosítja. Ezek a sín interfész, az utasítás dekódoló, végrehajtás és cím fordítás (a 80386 esetében ez hét fokozatból áll). Az NCR/32 mikroprocesszor késleltetett elágaztatás utasításokat használ, számos utasítás végrehajtását engedélyezve ez után. Ebből következően a ,,csőben" tartózkodó utasításokat nem kell ,,kidobni".

A technológia mind közvetett mind közvetlen hatással van az architektúrákra. Korunkban a chip területek korlátozásai megkívánják, hogy egy implementáció egy adott területen elférjen, és a teljesítményt ezért nagy mértékben korlátozhatják annak érdekében, hogy egy előzetesen meghatározott utasításkészletet egy adott chip részbe ,,be lehessen sajtolni". Hogy a rendelkezésre álló csapok száma ne jelentsen korlátozást, a tokozási technológiák nagyon gyorsan fejlődtek. Az elkövetkező ,,szelet" (wafer) technológiák (egy teljes szeletet használnak egy chip helyett) alapvetően megváltoztathatják jövőbeni gondolkodásunkat a processzor architektúrák technológia orientált korlátait illetően [22].

IRODALOM

[1] *Vajda, F.:* Supermicros-Objeetives and Approaches Microprocessing and Microprogramming, 17 (1986), pp. 1—17.
[2] *Vajda, F.:* Supermocroprocessors, Proc. of the Fourth Symposium on Mierocomputer and Microprocessor Applications (Budapest, 15—17 October, 1985), OMIKK—TECHNOINFORM, Budapest, 1985, pp. 226—252.
[3] *Fernandez, E.B.:* Comparion and Evaluation of 32-bit Microprocessors. Proc. Southcon 1984.
[4] *Finker, G.A.:* Full 32-bit Microprocessor: The Generation. Mini-Micro System, August 1983, pp. 187—194.
[5] *Gupta, A. and Toong M.D.:* An Architeetural Comparison of 32-bit Microprocessors, IEEE Micro, February 1983, pp. 9—22.
[6] *Bound, J.:* Architeetural Advances Spur 32-bit Micros. Computer Design, June 1, 1984, pp. 125—133.
[7] *Kavi K.M.:* A Conceptual Framework for the Description and Classification of Computer Architecture. Proc. of the IEEE int. Workshop on Computer System Organization, IEEE Computer Society Press, 1983, pp. 10—19.
[8] *Giloi, W. K.:* Toward a Texonomy of Computer Architecture Based on the Machine Daty Type View. Proc. of the 10th Annual Int. Symposium on Computer Architecture, IEEE Computer Society Press, 1983, pp. 6—15.
[9] *Dasgupta, S.:* Computer Design and Description Languages. Advances in Computer, Vol. 21., Academic Press, 1982. pp. 91—154.
[10] *Bell, G. G. and Newel, A.:* Computer Structures. Reading and Examples. McGraw Hill, New York, 1971.

(Megjegyzés: A szerző 44 tételes irodalomjegyzéket adott cikkéhez, amelynek csak egy részét tudjuk közzé tenni. Kérem azokat a kedves Olvasókat, akiket az irodalomjegyzék további része is érdekel, forduljon közvetlenül a Szerzőhöz. Főszerkesztő.)