Varga László: A Kalmár-féle formulavezérlésű számítógép, Rendszerprogramok elmélete és gyakorlata, Akadémiai Kiadó Budapest, 1978, részlet, 385-404.

4.13.1-es fejezet részletesen ismerteti a Kalmár-féle formulavezérlésű számítógépet. Három feladat is kapcsolódik a fejezethez.

4.13.1. A KALMÁR-FÉLE FORMULAVEZÉRLÉSŰ SZÁMÍTÓGÉP

A Kalmár-féle formulavezérlésű számítógép a következő öt főrészből áll:

1. bemeneti egység,

2. memória,

3. vezérlőegység,

4. aritmetikai egység,

5. kimeneti egység.

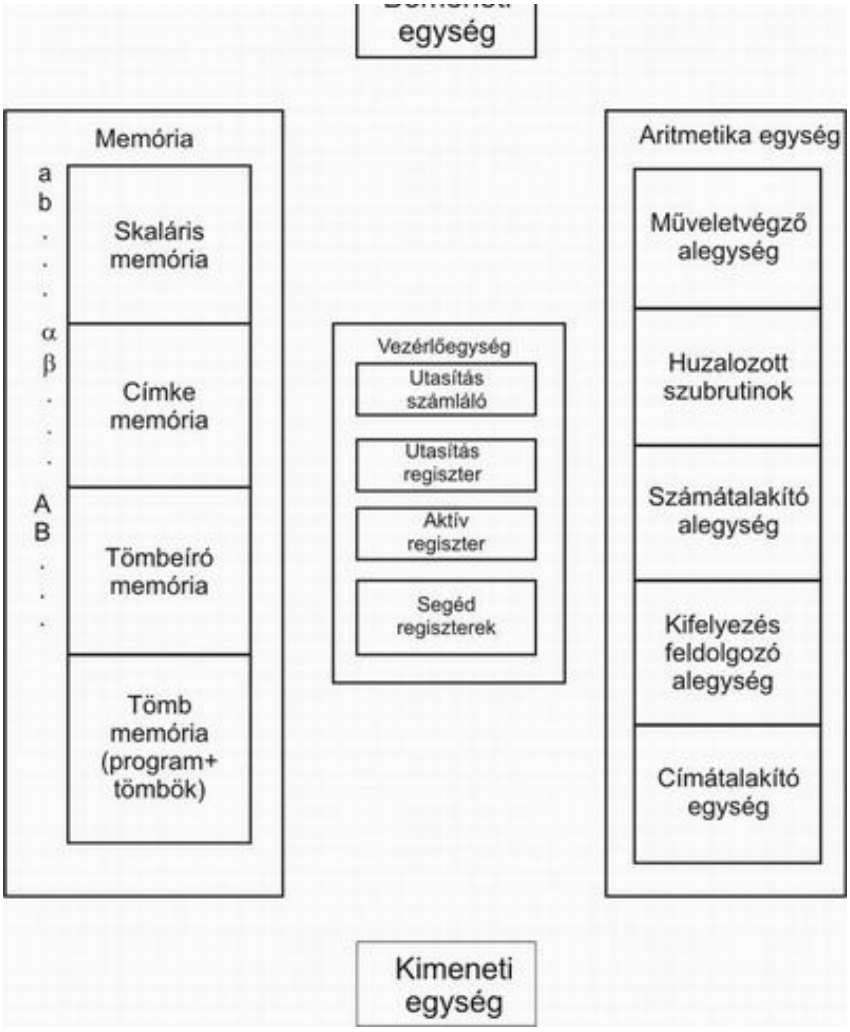
Ezeknek az egységeknek a részleteit a *4.36. ábra* mutatja.

A programot szimbolikus formában valamilyen adathordozóra - például lyukszalagra - lyukasztjuk le. A program az alapszimbólumokból alkotott szöveg. Ezt a szöveget a *bemeneti egység* olvassa be és helyezi el a memóriában, valamilyen alkalmasan megválasztott kódolt formában. Ez képezi a formulavezérlésű gépen a gépikódú programot. A programszöveg minden egyes szimbóluma a gép egy utasítása.

A memória négy részből áll:

1. *A skaláris memória.* Itt kapnak helyet a program skaláris változói. A skaláris változók azonosítóinak belső kódja az azonosítóhoz rendelt memóriaszó címét adja meg. Ez legegyszerűbben úgy valósítható meg, hogy az azonosítók egy betűből állnak - például az ABC kisbetűiből -, és a skaláris memória egy szavát mindig az a betű azonosítja, amelynek belső kódja a szó címével azonos. A tárgyalás során mi ezt a megoldást tételezzük fel. A programozó tehát véges számú (meglehetősen kevés számú) azonosító közül választhat és azoknak a helye a memóriában fix. Ez a gyors címzés megvalósításának egy egyszerű megoldása. Általában a logikai címek leképezését fizikai címekre megfelelő asszociatív tárral valósíthatjuk meg.

Bemeneti



4.36.ábra. A Kalmár-féle formulavezérlésű számítógép legfontosabb komponensei

2. *A címkememória.* Itt kaphatnak helyet a programban definiált címkék értékei. Ismét azt tételezzük fel, hogy a programozónak véges számú címke áll rendelkezésére, és azokhoz egyszer s mindenkorra hozzá van rendelve egy memóriaszó, amely az adott címke értékét tartalmazza, ha azt a címkét a programozó a programjában definiálta. Például tegyük fel, hogy a címkék görög betűk lehetnek és minden görög betűhöz az a memóriaszó van hozzárendelve, amelynek címével a betű belső kódja azonos.

3. A tömbleíró memória. Itt kapnak helyet a programban szereplő tömbök (vektorok, mátrixok) táblázatai. Ezek a táblázatok tartalmazzák azokat az adatokat, amelyeknek alapján a tömb elemei a memóriában megtalálhatók. A táblázatokat a programban szereplő deklarációs utasítások töltik ki és azok a tömb kezdőcímét, a tömb dimenziójára vonatkozó adatokat és a tömb típusának kódját tartalmazzák. A tömbleíró memória annyi „táblázat helyet" tartalmaz, amennyi tömbazonosító használható a programban. *Így* minden tömbazonosító kódjához egyértelműen hozzárendelhetjük a tömböt leíró táblázat címét a memóriában. Például tegyük fel, hogy a tömbazonosítók a nagybetűk lehetnek csak, és azok belső kódjából egy konstans kivonásával kapható meg a megfelelő táblázatelem címe.

4. A tömbmemória. A memóriának ebben a részében helyezi el a bemeneti egység a programot belső ábrázolású, kódolt formában. A program után kapnak helyet a programban szereplő tömbök elemei. A tömböknek természetesen nincsen fix helyük a memóriában, hiszen a tömbök címeit a tömbleíró elemek tartalmazzák.

A vezérlőegység az utasítások végrehajtását vezérli megfelelő regiszterek segítségével. Ez azt jelenti, hogy számlálja az utasításokat, kijelöli azokat a címeket és regisztereket, amelyekkel a műveletet végre kell hajtani, valamint vezérli az utasítások végrehajtásának ütemét.

Az utasításszámláló regiszter mindenkor a program egy alapszimbólumának címét tartalmazza. Ilyen alapszimbólum például egy azonosító, egy számjegy, egy műveleti jel stb. Az alapszimbólum kódja a gép egy utasítása, amely az *utasításregiszterbe* megy át. Ennek a kódnak alapján határozza meg a vezérlőegység azt, hogy az adott esetben mit kell tennie, például melyik memóriaszó tartalmát kell elhelyeznie valamelyik regiszterben. Néhány utasítás végrehajtását a későbbiekben részletesen fogjuk ismertetni.

A konkrét utasítást ezen a gépen. Általában két adat határozza meg. Az egyik adat az alapszimbólum kódja, a másik adat pedig az utasítás indulásakor szerepet játszó regiszter, az ún. *aktív regiszter.* Az aktív regiszter azonosító kódját, címét egy külön regiszter tartalmazza. Egy hagyományos gépen rendszerint egy gépi utasítás

azonosító kódját, címét egy külön regiszter tartalmazza. Egy hagyományos gépen rendszerint egy gépi utasítás önmagában is egyértelműen meghatározza azt, hogy mit kell tenni az utasítás hatására. A formulavezérlésű gépen viszont például egy azonosító kódja önmagában még nem határozhatja meg a műveletet, a hatása attól is függ, hogy milyen kód volt előtte. Például

a+b

esetén b az összeadás elvégzését is eredményezi, míg a

b+a

formula esetén nem. A programnak ezt az „előéletét" rögzíti az aktív regiszter.

A fenti regisztereken kívül a formulavezérlésű gép még számos más regiszterrel is rendelkezik. Ezeknek egy része az aritmetikás egység részének tekinthető, más része viszont a vezérlőegységet egészíti ki bizonyos műveletek elvégzése esetén. Az utóbbiakat *segédregiszteреknek* nevezzük. Ezeket a segédregisztereket használja fel például a vezérlőegység a ciklusutasítás végrehajtásánál.

Az aritmetikai egység öt alegységből áll:

1. *A műveletvégző alegység.* Ez a hagyományos gépeken szokásos aritmetikai egységnek felel meg, amelynek részei a következők:

összeadómű,

szorzómű,

osztómű,

relációk logikai értékét meghatározó mű,

logikai műveletvégző mű,

ekvivalencia-mű.

A műveletvégző alegység megfelelő számú operandus- és eredményregiszterrel rendelkezik.

2. *Huzalozott szubrutinok.* A legfontosabb szabványos függvényeket, mint pl. a hatványozás, gyökvonás, exp x, sin x, cos x stb. a formulavezérlésű gép „ismeri", ezeknek az algoritmusát a felhasználónak nem kell megadnia. A szabványos függvényeket a számítógép huzalozott formában tartalmazza. Ezeket a felhasználó úgy használhatja, mint egy műveletet. A szabványos függvényeket úgy célszerű kódolni, hogy azok csak a műveletvégző alegységet használják. Szükség szerint ezek egymás szolgáltatásait is igénybe vehetik. A huzalozott szubrutinok rendszerét úgy is felfoghatjuk, mint a műveletvégző alegység kiterjesztését, lehetővé téve így összetett műveletek elvégzését is.

3. *Számátalakító alegység.* Ennek az alegységnek a vezérlését a számjegyek, a tizedespont és a számjegy terminátor kódjai vezérlik. Mivel a formulavezérlésű gép bináris számokkal dolgozik, a decimális formában felirt számokat bináris formára kell átalakítani. Ezt a feladatot látja el a számátalakító alegység.

4. *Kifejezésfeldolgozó alegység.* A kifejezéseket ún. teljesen zárójelezett formában kell felírni. Például az

*a*b+c*d/e*

formulát a

*(a*b)+((c*d)/e)*

vagy a

*(a*b)+(c*(d/e))*

formában kell felírni. A kifejezés-feldolgozó alegységnek az a feladata, hogy egy teljesen zárójelezett formában felirt kifejezés kiszámítását előkészítse a műveletvégző alegység számára, azaz meghatározza a műveletek végrehajtásának a sorrendjét. A kifejezés-feldolgozó alegység működését a későbbiekben az egyszerű értékadó utasítások feldolgozásának példáján részletesen fogjuk ismertetni. A teljesen zárójelezett forma használata kényelmetlen a programozó számára. Elkészíthető a kifejezés-feldolgozó alegység úgy is, hogy ettől mentesítsük a programozót. Mi azonban a későbbiekben az egyszerűség kedvéért csak a teljesen zárójelezett formula feldolgozását vizsgáljuk. A programnak erre a közbülső formára való áttranszformálásával nem foglalkozunk.

5. Címkiszámító alegység. Ez az alegység tömbdeklaráció és indexes változóval való műveletvégzés esetén lép működésbe. Tömbdeklaráció feldolgozása során a címkiszámító alegység számítja ki és tölti ki a tömbleíró táblázatot. Az indexes változóval végzett művelet esetén pedig a címkiszámító alegység feladata a tömbleíró

táblázatot. Az indexes változóval végzett művelet esetén pedig a címkiszámító alegység feladata a tömbleíró táblázat felhasználásával az adott tömbelem címének az előállítása.

A kimeneti egység feladata az eredményadatoknak a megjelentetése, azaz az eredmények előkészítése a megfelelő perifériális egység számára és a perifériális egységek működésének a vezérlése. Számok megfelelő formában való-kiírásához tartalmazza az átalakító egységeket is behuzalozott szubrutinok formájában.

4.13.1.1. AZ EGYSZERŰ ÉRTÉKADÓ UTASÍTÁSOK FELDOLGOZÁSA

A formulavezérlésű gép működését először az egyszerű értékadó utasítások feldolgozásának példáján szemléltetjük. Az egyszerű értékadó utasítást a következőképpen definiáljuk:

<értékadó utasítás>::=<aritmetikai kifejezés>=><skaláris változó>;

<aritmetikai kifejezés>::=<operandus> | <operandus><operátor><operandus>;

<operandus>::=<skaláris változó> | <operandus><operátor><operandus>;

<operátor>::= + | - | * | / | ;

<skaláris változói>::= a | b | c | . .

Az egyszerű értékadó utasításban tehát csak egy betűvel azonosított skaláris változók fordulnak elő. Indexes változók és konstansok az egyszerű értékadó utasításban nem fordulnak elő. Mint látjuk, az aritmetikai kifejezéseket ún. teljesen zárójelezett formában írjuk fel. Természetes dolog, hogy az értékadás művelete a kifejezés után áll. A kifejezés kiszámítása után kell ugyanis az értékadást végrehajtani és ezt a műveletet írja elő ez a műveletjel.

Példák egyszerű értékadó utasításokra:

a => x

a+b => c

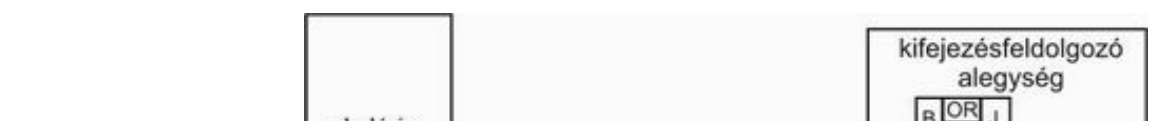
(a+b)*(c-d) => y

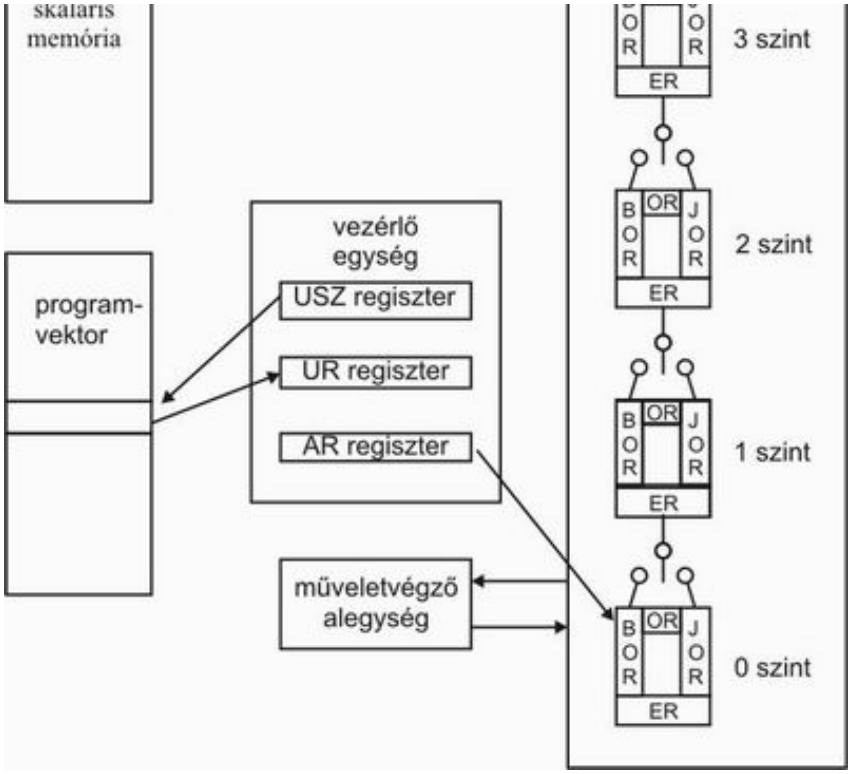
((a+b)*(c/d) x => y

Az egyszerű értékadó utasítások feldolgozására alkalmas regiszternégyesek rendszerét a *4.37. ábra* szemlélteti.

Az ábrán a programvektor tárolására egy négyzetekre osztott szalagot helyeztünk el, amely a tömbmemória része. Mint említettük, a program a tömbmemória elején helyezkedik el. A szalagon a négyzetek az alapjelek kódjának tárolására szolgálnak. Egy négyzetben pontosan egy alapjelkód helyezhető el. Minden egyes alapjelkód a számítógép egy utasítása.

A programvektor betöltésekor az Utasítás Számláló (USz) regiszter annak a memóriaszónak a címét veszi fel, amely a programvektor első szimbólumát tartalmazza. Ezután az USz regiszter tartalma minden utasítás végrehajtása után





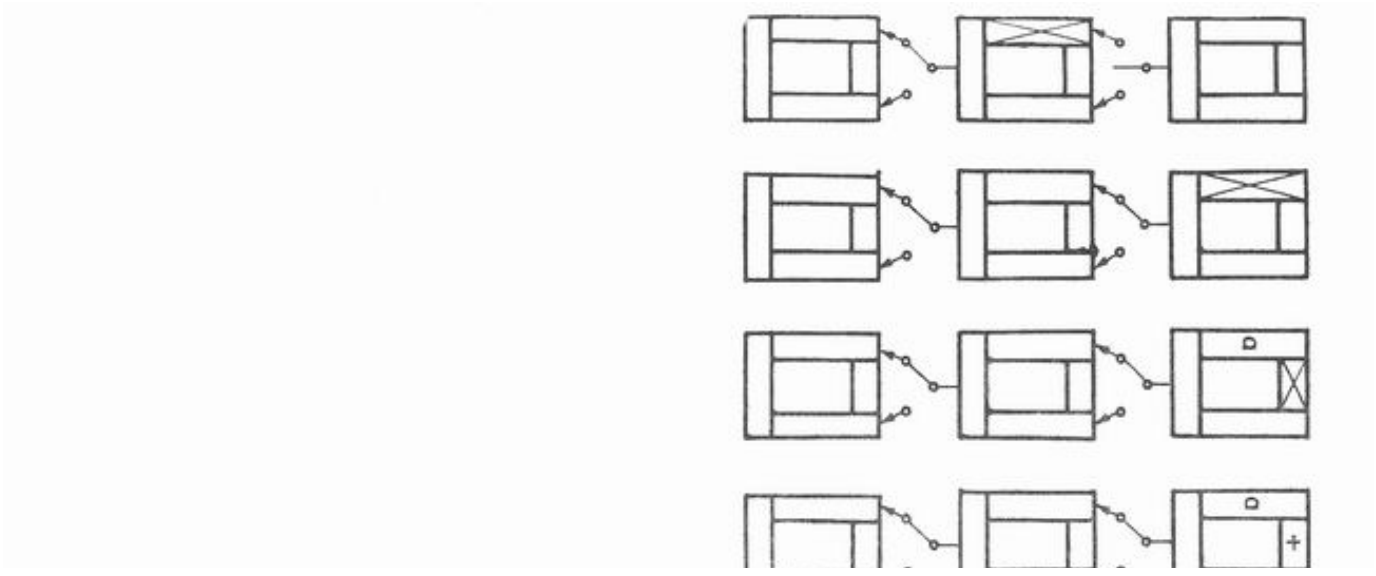
4.37. ábra. *A kifejezés-feldolgozó alegység felépítése*

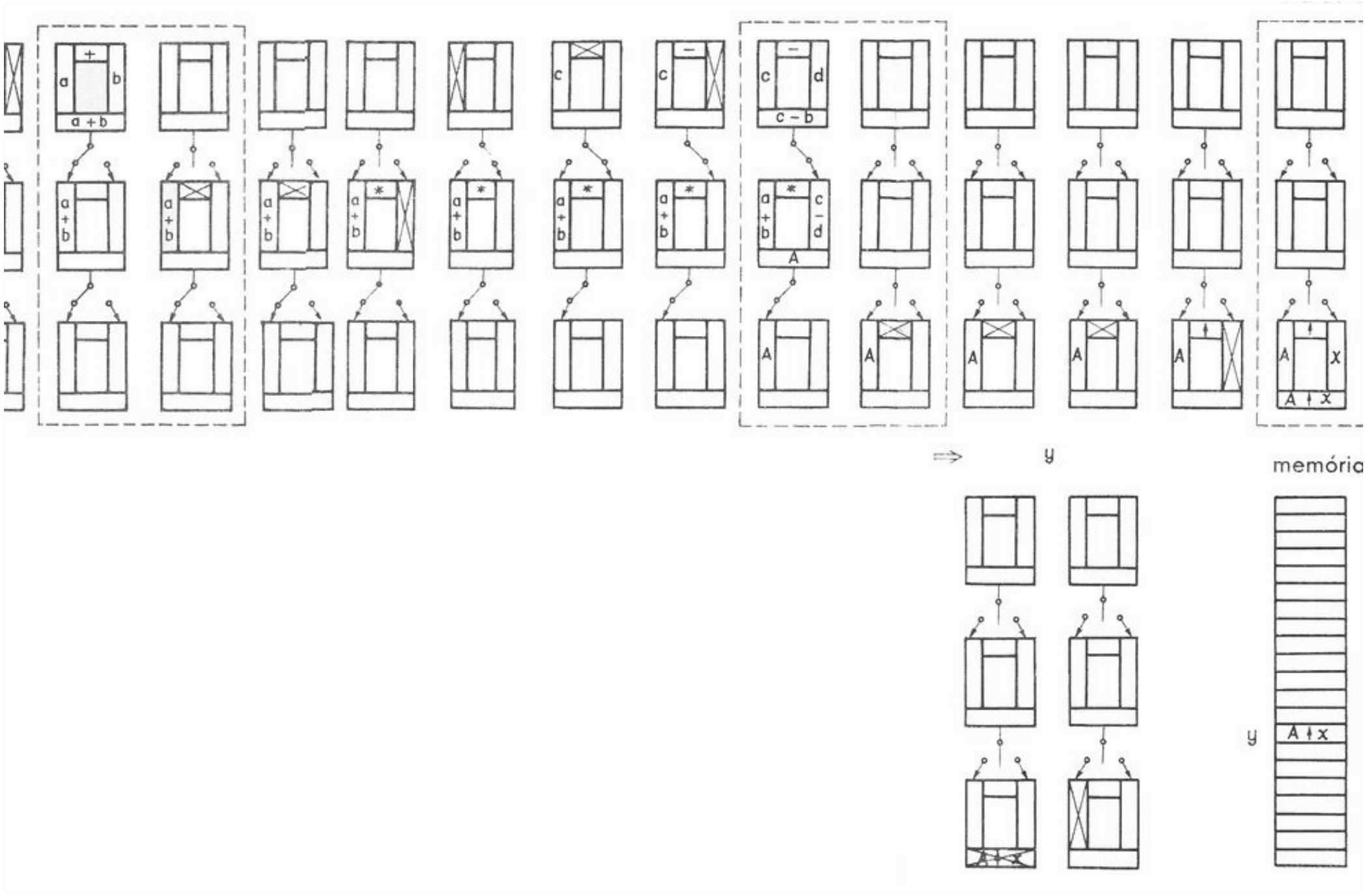
1-gyel nő. Mindig az az alapszimbólum kerül be az Utasítás Rögzítő (UR) regiszterbe, amelynek a címét az USz regiszter tartalmazza. Így az utasítások egymásután, az ábrázolás szerinti sorrendben kerülnek be az UR regiszterbe. Amikor egy utasítás az UR regiszterben elhelyezkedik, azonnal megtörténik annak végrehajtása, Az utasítások végrehajtását a vezérlőegység vezérli. Az egyszerű értékadó utasítás feldolgozása során szükséges utasításokat és azok eredményeit a későbbiekben táblázatosan adjuk meg.

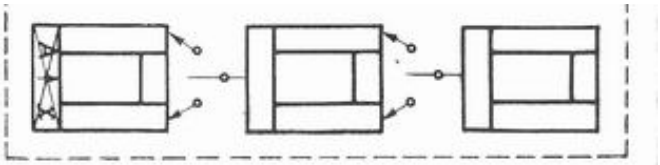
Az egyszerű értékadó utasítás végrehajtása a regiszternégyesekből álló, kifejezés-feldolgozó alegység segítségével történik. A különböző szintű regiszternégyesek, a műveletvégző alegységgel együtt, a teljesen zárójelezett forma különböző szintű (mélységű) aritmetikai műveleteinek elvégzésére szolgálnak. Az aritmetikai műveletek szintjeit a következő példán szemléltetjük

((a+b)*(c/c))^x

2 1 2 0




a
b
+
a + b
c
d
−
c − b
a + b
c − d
A
X
A + X
y
memória



1. melléklet

Egy művelet szintjét a művelet zárójelezettségi mélységének a száma adja meg. Példánkban a művelete, mivel nincs zárójelbe zárva, 0 szintű művelet. A * művelete egy zárójelbe van bezárva, ezért 1. szintű művelet, míg az egyaránt két zárójelbe zárt + és / műveletek 2. szintű műveletek.

A kifejezés-feldolgozó alegység a gép kiépítettségétől függően, általában a darab regiszternégyest tartalmaz. Esetünkben n=4. Ha a regiszternégyesek száma n, akkor a kifejezésekben maximálisan n-1 mélységben skatulyázhatók zárójelbe a műveletek.

A regiszternégyesek egy-egy Bal-Operandus Regiszterből (BOR), Jobb-Operandus Regiszterből (JOR), Operátor (műveleti jel) Regiszterből (OR), és Eredmény Regiszterből (ER) állnak. Az adott szinten elvégzendő művelet kódját az OR, az operandusokat a BOR, ill. JOR, az eredményt pedig az ER regiszterben tároljuk. Ennek megfelelően az OR regiszter hosszát a gépen rendelkezésre álló műveletek száma határozza meg. A BOR, JOR, ER regiszterek hossza a memória szavának a hosszával egyezik meg. Ezekben a regiszterekben tehát egy-egy szó hosszúságú mennyiséget tárolhatunk.

A regiszternégyes a következőképpen működik: Alapállapotban minden regiszter üres. A vezérlőegység általában először a BOR, ezután az OR, majd a JOR regisztereket tölti ki. Amikor a JOR regiszter tartalma megtelik, a vezérlőegység működésbe hozza az aritmetikai egységet, amely az OR regiszter által meghatározott műveletet a BOR és JOR regiszterek tartalmával elvégzi. A vezérlőegység az eredményt az ER regiszterbe tölti be.

A regiszterek kitöltésének vezérlése az AR regiszter segítségével történik. Ez a regiszter kezdetben a 0. szinten álló regiszternégyes BOR regiszterének a „címét" tartalmazza. Azt a regisztert, amelynek a címét az AR regiszter tartalmazza, *aktív regiszternek* nevezzük. Az AR regiszter értékét az utasítások változtatják meg. Az utasítások hatását később fogjuk megadni.

A struktúra regiszternégyesei több szinten helyezkednek el. A magasabb szinten álló regiszternégyest egy kétágú kapu köti össze az 1-gyel alacsonyabb szinten álló regiszternégyessel. A kétágú kapunak egyszerre csak legfeljebb egyik ága lehet nyitva. Ez azt jelenti, hogy a magasabb szinten álló ER regiszter tartalma automatikusan átmásolódik az 1-gyel alacsonyabb szinten álló BOR és JOR regiszterek közül abba, amelyikkel nyitott kapu köti össze. Egy ilyen átmásolás után a magasabb szinten álló regiszternégyes OR, BOR, JOR és ER regiszterei kiürülnek és a kapu mindkét ága bezárul, az összeköttetés megszakad.

Az utasítások leírásához a k-adik szint regisztereinek a jelölésére bevezetjük a BOR k , OR k , JOR k , ER k indexes regiszter neveket. Az utasítást két adat határozza meg. Az egyik adat az utasításszimbólum kódja, a másik adat pedig az aktív regiszter, azaz az a regiszter, amelynek címét az AR regiszter tartalmazza.

Mint később látni fogjuk, egy kifejezés feldolgozása közben működésbe léphet a címkiszámító alegység. Ilyenkor az indexkifejezések kiszámításához is szükség van a kifejezés-feldolgozó alegység munkájára. Ez azonban a kifejezés-feldolgozó alegységnek egy kicsit más működését kívánja meg. Azt, hogy a kifejezés-feldolgozó alegységnek melyik módban kell működnie, egy VR regiszter mutatja. Amikor a VR regiszter értéke üres, a kifejezés-feldolgozó alegység nem indexkifejezést dolgoz fel. A másik üzemmódot az jellemzi, hogy a VR regiszter nem üres.

Az utasítások a következők:

A művelet kódjele:	*Aktív regiszter:*	*Az utasítás hatása:*
(	BOR k vagy JOR k	Az ER k+1 és az aktív regiszter közötti kapu kinyílik, majd AR a BOR k+1 regiszter címét veszi fel.
<skaláris változó>	BOR k	Annak a memóriaszónak a tartalma, amelynek címe a skaláris változó kódja (az UR regiszterben van), betöltődik BOR k -ba. Ezután AR az OR k címét veszi fel.

<skaláris változó>	JOR k (és a VR regiszter üres)	a) Annak a memóriaszónak a tartalma, amelynek címe a skaláris változó kódja, betöltődik JOR k -ba. A betöltés hatására megindul a k-adik szintű négyes által meghatározott művelet végrehajtása és az utasítások végrehajtása az egymás alatti négyesekben is folytatódik mindaddig, amíg egy I-edik szintről a nyitott kapu a JOR l-1 regiszterhez vezet. b) Ha egy i-edik szintről a nyitott kapu BOR i-1 -hez vezet, akkor AR regiszter OR i-1 címét veszi fel. Ha nincs ilyen i, akkor AR az ER0 címét veszi fel. Ezután az i-1-edik szint felett minden kapu bezárul és az i-1-edik szint felett minden regiszter kiürül. Ha AR regiszter az ER 0 címét veszi fel, akkor ER 0 kivételével minden regiszter kiürül.
<skaláris változó>	ER 0	ER 0 tartalma átmásolódik abba a memóriaszóba, amelynek címe a skaláris változó kódja. Ezután ER 0 kiürül és AR felveszi a BOR 0 címét.
<operátor>	OR k	UR tartalma betöltődik OR k -ba. Ezután AR a JOR k címét veszi fel.
)	bármelyik	Üres utasítás, hatástalan.
?	ER 0	A BOR 0 , OR 0 , JOR 0 regiszterek kiürülnek és AR az ER 0 címét veszi fel. Ezzel jelöljük meg azt, hogy a következő skaláris változó értékadást ír elő.
Ü	OR 0	A BOR 0 tartalma átmásolódik ER 0 -ba, majd BOR 0 kiürül. Az AR regiszter az ER 0 regiszter címét veszi fel. (Ez az eset akkor áll elő, ha a kifejezés egy önmagában álló változónév.)

Minden más utasítás hibás utasítás, azaz a kódvektor szintaktikusan hibás. Könnyű belátni, hogy a szintaktikusan helyes kódvektor összes lehetséges utasítását megadtuk.

Hibás utasításra vezet az is, ha a k+1-edik szintű regiszternégyes nem létezik. Ez akkor következik be, ha a programban a zárójelmélység meghaladja az n darab regiszternégyes által megszabott n-1 maximumot.

A fenti utasításokkal rendelkező gép helyes működését a következőkben kísérjük végig egy példán.

4.27. Példa. Tekintsük az

$$(a+b)^* (c-d)^x \Rightarrow y$$

értékadó utasítást. Kísérjük végig a gép működését az 1. melléklet ábráján a fenti utasítás végrehajtása közben. Csak a regiszternégyesekből álló struktúra állapotait ábrázoljuk egy-egy utasítás végrehajtása után. A két lépésben végrehajtott utasítás mindkét fázisának eredményét külön ábrázoljuk, és a két fázis ábráját egy szaggatott vonallal jelölt mezőbe tesszük. Az ábrákon azt a regisztert, amelynek a címét a művelet végrehajtása után az AR regiszter felveszi, átlósan áthúztuk. Az ábránkon az

$$A= (a+b)^*(c-d)$$

jelölést vezettük be.

4.13.1.2. SZÁMÁTALAKÍTÁS

A programban a számokat tízes számrendszerben adjuk meg. A gép bináris számokkal dolgozik, ezért mint már említettük, szükség van egy olyan egységre, amely a decimális alakú számot bináris alakúra alakítja át. A következőkben a fixpontos formában felírt előjel nélküli valós számok átalakítóját ismertetjük.

Legyen a konstansok formája a következő:

<konstans>::=<szám><konstans vége>;

<szám>::=<természetes szám> |

<természetes szám><tizedes pont><természetes szám>;

<természetes szám>::=<számjegy> | <természetes szám><számjegy>;

<számjegy>::=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9;

<tizedes ponti>::= . ;

<konstans vége>::= 3 ;

A számátalakító alegység a kifejezés feldolgozó alegységre támaszkodva látja el feladatát. A számátalakítónak két új regiszterre van szüksége. Az egyiket *váltószámregiszternek* VSZR, a másikat *számregiszternek* SZR nevezzük. Az átalakítás formuláját a következő példa keretében ismertetjük. Legyen a szám:

712.361,

akkor az átalakítás formulája:

(7* 10+ 1)* 10+ 2+ 3*0.1 + 6*0.01 + 1*0.001

Az egészrész átalakítása tehát a Horner-elrendezés szerint történik.

Az átalakítás a fenti regiszterek segítségével és a műveletvégző alegység felhasználásával a következőképpen végezhető el:

Alaphelyzetben legyen az SZR regiszter tartalma 0, a VSZR regiszter tartalma pedig legyen 10. Tegyük fel, hogy a számjegyek belső kódja azok érte-évei azonos. Ekkor az átalakító utasításai a következők lehetnek:

A művelet kódjele:	*Az aktív regiszter:*	*Az utasítás hatása:*
<számjegy>	Nem játszik szerepet	a) Ha VSZR tartalma 10, akkor VSZR és SZR regiszterek tartalmát összeszorozzuk, majd az így kapott eredményt megszorozzuk az UR regiszter tartalmával. Az eredményt az SZR regiszter veszi fel. Az aktív regiszter nem változik. b) Ha VSZR tartalma nem 10, akkor a VSZR tartalmát megszorozzuk 1/10-del és az eredményt a VSZR regiszterbe töltjük be. Ezután összeszorozzuk a VSZR és UR regiszterek tartalmát. Az eredményhez hozzáadjuk az SZR tartalmát. Az így kapott végösszeget az SZR regiszterbe töltjük be. Az aktív regiszter változatlan.
<tizedes pont>	Nem játszik szerepet	A VSZR regiszterbe az 1 értéket töltjük be. Az aktív regiszter változatlan.
<konstans vége>	BOR k	Az SZR tartalmát betöltjük a BOR k regiszterbe, az SZR és a VSZR regiszterek alaphelyzetbe kerülnek.

		SZR és a VSZR regiszterek alaphelyzetbe kerülnek és az AR az OR k regiszter címét veszi fel.
<konstans vége>	JOR k	Az SZR tartalmát betöltjük a JOR k regiszterbe, az SZR és a VSZR regisztereket alaphelyzetbe állítjuk. A JOR k betöltése után megindul a k-adik szinten meghatározott művelet végrehajtása az előző pontban megismert módon.

Elvileg a számátalakításhoz is felhasználható lenne az egyszerű értékadó utasításoknál szereplő regiszternégyesekből álló struktúra. Amikor azonban egy kifejezésben szereplő konstansnak, mint operandusnak az átalakítása a feladat, akkor célszerűbb az átalakításhoz szükséges műveleteket a fent bevezetett VSZR és SZR regiszterek segítségével elvégezni.

4.28. Példa. Kísérjük végig a számátalakítást a 37.24 esetében. A regiszterek állapotai ekkor egymás után a következők:

VSZR: 10 10

\+ +

UR: 3 7

* = * =

SZR: 0 3 37

1/10

*

VSZR: 10 1 1/10

*

ÚR: 2

\+ =

SZR: 37 37 37.2

1/10

*

VSZR: 1/10 1/100 10

*

ÚR: 4 ,_,

+ =

SZR: 37.2 37.24 0

BOR k vagy JOR k : 37.24

4.13.1.3. TÖMBDEKLARÁCIÓ ÉS INDEXES VÁLTOZÓ FELDOLGOZÁSA

A tömb elemeit a memóriában indexük szerint növekvő sorrendben helyezzük el. Kétdimenziós tömb esetén ez megfelel a tömb sorfolytonos elhelyezésének.

Korábban láttuk, hogy ha a tömb elemei a következők:

A i1i2…ir, i 1 =0, 1, …, n 1 -1,

i 2 =0, 1, …, n 2 -1,

.

.

.

i r =0, 1, …, n r -1,

akkor

$$\mathrm{loc}(\mathrm{A}_{i_1 i_2 \ldots i_r}) = \mathrm{loc}(\mathrm{A}_{00\ldots0}) + \sum_{k=1} i_k * b_k,$$

ahol

$$b_k = \prod_{j=k+1}^{r} n_j, \; ha \;\; k \leq r-1,$$

és

b r =1.

A továbbiakban tegyük fel, hogy a formulavezérlésű gépen a tömbök indexeinek alsó korlátja 0, az n1 , n2, …, nr egész értékű aritmetikai kifejezések, és minden elem egy szót foglal el a memóriában. Ezeket a megszorításokat csak azért tesszük, hogy a formuláink egyszerűbbek legyenek, de az ismertetendő címkiszámító alegység kisebb módosítással általánosabb feladat megoldására is alkalmas.

A tömbdeklaráció feldolgozása során a címkiszámító alegység feladata a tömbleíró táblázat kitöltése. A tömbleíró táblázat szerkezete most a következő lehet:

a tömb első elemének címe; *b0 ; b1 ; …; bn*

ahol n formulavezérlésű gép nyelvében a tömbökre nézve megengedett legnagyobb dimenziószám ; *b1* = bl , *b2* = b2 , . . . , *br* = br , továbbá *bi* = 0, ha i > r,

$$\beta_0 = b_0 = \prod_{l=1}^{r} n_l$$

a tömb helyfoglalása *és r* a tömb dimenziószáma.

Legyen a deklaráció formája a következő:

$$\alpha \Rightarrow \otimes\beta \Rightarrow \otimes\ldots \Rightarrow \otimes \Rightarrow \gamma =\rangle \Omega,$$

ahol alfa, ß, …, gamma a felső indexhatárokat meghatározó kifejezések és omega a tömb azonosítója.

Legyen a tömbleíró táblázat hossza n+2 és tegyük fel, hogy a tömbleíró táblázat első elemének a címe a tömbazonosító belső kódjának n+2-szerese.

Vezessük be a következő regisztereket:

A szabadrekesz regiszter, SZRR. Ebben a regiszterben állítjuk elő a tömbmemória mindenkori első szabad rekeszének a címét. A programvektor betöltése után ez a rekesz a programvektor utáni első szabad rekesz címét veszi fel. Ez a vezérlőegység egyik segédregisztere.

A címlépés regiszterek CLR 0 , CLR 1 , ..., CLR n . A CLR 0 a tömb helyfoglalásának, a CLR 1 , ..., CLR r pedig rendre a b 1 , b 2 , ..., b r mennyiségek tárolására szolgál *(r L n).*

A gép indulásakor és minden egyes tömbdeklaráció feldolgozása után CLR 0 értéke 1, CLR 1 , CLR 2 , ..., CLR n értéke pedig 0.

A címregiszter CR. Ezt a regisztert jelenleg egy segédregiszterként használjuk. A deklaráció feldolgozásának utasításai a következők:

Az utasítás kódja:	*Az aktív regiszter:*	*Az utasítás hatása:*
Ä	ER 0	Legyen k az a legkisebb index, amelyre CLR k > 0. A CLR 0 , CLR 1 , ..., CLR k regiszterek tartalma megszorzódik az ER 0 regiszter tartalmával, majd CLR k+1 az 1 értéket veszi fel. ER 0 kiürül és az AR regiszter a BOR 0 regiszter címét veszi fel.
=⟩	ER 0	Legyen k az a legkisebb index, amelyre CLR k > 0. A CLR 0 , CLR 1 , ..., CLR k regiszterek tartalma megszorzódik az ER 0 regiszter tartalmával, majd az ER 0 regiszter kiürül. A CR regiszter átveszi az SZRR regiszter tartalmát. Az SZRR regiszter és a CLR 0 regiszter összege betöltődik az SZRR regiszterbe. Az AR regiszter felveszi a CR regiszter kódját. (Ezzel jelöljük meg azt, hogy a tömbnév most más utasítás lesz, mint egy indexes változóra való hivatkozás esetén.)

4.29. Példa. Legyen a deklaráció a következő:

$$n \Rightarrow \otimes m \Rightarrow = \rangle A,$$

ahol *n,* ill. *m* értéke rendre 2, ill. 10. Kísérjük végig az UR, ER 0 , CR, CLR 0 , CLR 1 , CLR 2 , CLR 3 , SZRR rekeszek tartalmának változását a deklaráció feldolgozása során. Legyen az induláskor SZRR tartalma 1000. A regiszterek tartalmát a kijelölt utasítás végrehajtása után ábrázoljuk:

UR :	n	=>	OX	m
ER 0	-	2	-	-
CR :	0	0	0	0
CLR:	1	1	2	2
CLR 1 :	0	0	1	1
CLR 2 :	0	0	0	0
CLR 3 :	0	0	0	0
SZRR :	1000	1000	1000	1000
UR:	=>	$= \rangle$	5*A	A tömbleíró táblázat:
ER 0 :	10	-	--	
CR:	0	1000	0	
CLR 0 :	2	20	1	1000
CLR 1 :	1	10	0	20
CLR 2 :	0	1	0	10
CLR 3 :	0	0	0	1
SZRR:	1000	1020	1020	0

Tegyük fel ezek után azt, hogy az aritmetikai kifejezés tömbelemeket is tartalmazhat operandusként. Legyen tehát

<változó>::=<skaláris változó> | <indexes változó> ;

<skaláris változó>::=<kisbetű>;

<indexes változó>::=<tömbnév>[<indexlista>];

<tömbnév>::=<nagybetű>;

<tömbnév>::=<nagybetű> ;

<indexlista>::=<index> | <indexlista><index-vessző><index>;

<index>::=<aritmetikai kifejezés>

és nézzük meg, hogyan tudja a címkiszámító alegység a tömbelem címét meghatározni a

$$\mathrm{loc}(A_{00\ldots0}) = \sum_{k=1}^{r} i_k * b_k$$

formula alapján. A formula kiszámítására célszerű magát a kifejezés-feldolgozó alegységet felhasználni. Ez úgy történhet meg, hogy amikor a kifejezés-feldolgozó alegység egy kifejezés feldolgozása közben egy tömbelemhez ér, akkor felfüggeszti a kifejezés feldolgozását és a magasabban levő regiszternégyesek felhasználásával áttér a fenti formula feldolgozására, azaz a tömbelem címének a meghatározására, majd ezután folytatja a kifejezés további feldolgozását. Ezt segíti elő a már korábban megismert VR regiszter.

Tegyük fel, hogy az l-edik szintű regiszternégyes működése közben szükségessé válik egy tömbelem címének a kiszámítása. Ekkor az előbb elmondottak szerint az l+1-edik szintű, az l+2-edik szintű. . . regiszternégyesekből álló kifejezés-feldolgozó egységet kell felhasználni a fenti formula kiszámítására. Ez a kifejezés-feldolgozó egység működésének a

<skaláris> JOR k

utasítás és a számátalakító-egység működésének a

<konstans vége> JOR k

utasítás esetén kisebb módosítását kívánja meg. Ennek a módosításnak az ismertetésére azonban már nem térünk vissza.

A címkiszámító alegység működése a következő:

Az utasítás kódja:	*Az aktív regiszter:*	*Az utasítás hatása:*
<tömbnév>	BOR k	UR•-UR~(n+2).
	JOR k vagy ER k	Abból a tömbleíró táblázatból, amelynek címe UR regiszterben van, a táblaelemek értékei rendre átmásolódnak a CR, CLR 1 , CLR 2 , . . ., CLR n regiszterekbe. Az AR regiszter tartalma átmásolódik VR regiszterbe. AR regiszter ezután a BOR k+1 regiszter címét veszi fel.
[		Üres utasítás, hatástalan.
<index vessző>	ER k	Ez az az eset, amikor az index értéke egy kifejezés kiszámításának eredményeként állt elő. CR tartalmához hozzáadódik CLR 1 tartalmának és ER k tartalmának a szorzata. Ezután a CLR i regiszter értékét felveszi a CLR i-1 regiszter, rendre i=2, 3, ..., n mellett és a CLR n regiszterbe 0 kerül. A BOR k , OR k , JOR k és ER k kiürül. Az aktív regiszter BORk címét veszi fel.
<index vessző>	OR k	Ez az az eset, amikor az index egy konstans, vagy egy skaláris változó. CR tartalmához most a CLR1 és a BOR k regiszterek tartalmának a szorzata adódik hozzá. Ezután a CLR i értékét

		felveszi a CLR i-1 regiszter tartalmát i=2, 3, ..., n mellett, és a CLR n regiszter 0 értéket vesz fel. A BOR k regiszter kiürül és AR felveszi a BOR k címét.
]	ER k	A CR regiszter tartalmához hozzáadódik a CLR 1 és az ER k regiszterek tartalmának a szorzata. A BOR k , OR k , JOR k és ER k regiszterek tartalma kiürül. VR tartalma áttöltődik AR-be és VR kiürül. A CR regiszter tartalma átmásolódik az UR regiszterbe és a kifejezés feldolgozó alegység ismét működésbe lép úgy, mintha az UR regiszterben egy skaláris változó címkódja lenne.
]	OR k	A CR regiszter tartalmához hozzáadódik a CLR 1 és a BOR k regiszterek tartalmának szorzata. A BOR k regiszter kiürül. A VR regiszter tartalma visszatöltődik AR-be és VR kiürül. A CR regiszter tartalma átmásolódik az ÚR regiszterbe és a kifejezés-feldolgozó alegység ismét működésbe lép úgy, mintha az UR regiszterben egy skaláris változó címkódja lenne.

Az utóbbi két utasítás végrehajtása után a CLR0 regiszter tartalma 1, a CLR1, CLR2, . . ., CLRn, CR regiszterek tartalma pedig 0 lesz.

4.13.1.4. VEZÉRLÉSÁTADÁS

A programot a bemeneti egység helyezi el a memóriának a tömbök számára fenntartott részén. **A** program fizikai *végét* egy speciális alapszimbólum jelzi. Ennek a szimbólumnak a beolvasása után az utasításszámláló USZ regiszter annak a szónak a címét veszi fel, amely a programvektor első alapszimbólumát tartalmazza. Ezzel a program végrehajtásra kész állapotba kerül.

A program végrehajtása közben, minden olyan utasítás végrehajtásakor, amely nem ír e16 vezérlésátadást, az USZ regiszter tartalmát a vezérlőegység 1-gyel növeli. Ilyenkor a programvektor szimbólumait a vezérlőegység a fizikai sorrendnek megfelelően egymás után tölti be az utasítás UR regiszterbe és hajtja végre. Ha a fizikai sorrendtől a programozó el akar térni, akkor szüksége van vezérlésátadó utasításra és olyan utasításra, amellyel a programvektor egyes szimbólumait szimbolikus címmel láthatja el.

Legyenek a vezérlésátadó utasítások a formulavezérlésű számítógép nyelvén a következő alakúak:

<természetes szám><címke vége>,

ahol a <természetes szám> egy címke, a <címke vége> szimbólum pedig különbözik a <konstans vége> szimbólumtól. Az utasításnak ez a formája összhangban van a formulavezérlésű számítógép eddig megismert utasításaival. Például az értékadó utasításnál is a kifejezést követi a => szimbólum, amely megmondja, hogy mit kell csinálni a kifejezés értékével. Itt pedig a <címke vége> tulajdonképpen az utasítás, amely azt írja elő, hogy a természetes szám> által meghatározott címre kell átadni a vezérlést. Olyan ez, mintha a FORTRAN-ban a vezérlésátadást a fordított sorrendben írnánk le, pl.:

20 GOTO

A címke az utasítás címét indirekt módon határozza meg. A címke mindig annak a memóriaszámnak a címe, amely a címkével szimbolikusan jelölt utasításcímet tartalmazza. (Itt jegyezzük meg, hogy mivel a vezérlőegység minden egyes utasítás végrehajtása után az USZ tartalmát 1-gyel növeli, ezért a címke által meghatározott szóban azonosított utasítás címe helyett az azt megelőző címet célszerű elhelyezni.) Ilyen célra a címkememória szavai állnak a programozó rendelkezésére, amelyeket a programozó a címkét definiáló utasítással tölti ki. Természetesen a programozó csak annyi szót tölt ki, azaz annyi címkének ad értéket, amennyire a programjában

szüksége van. A címkedefiníció formája legyen az, hogy a címket a címkezárójelbe téve elhelyezzük az előtt az utasítás előtt, amelyet vele azonosítani akarunk:

{<természetes szám>}<azonosított utasítás>

A címke értéke tehát annak a programszimbólumnak (utasításnak) a címe, amely fizikai sorrendben a definíció után áll.

Tegyük fel, hogy egy utasításhoz legfeljebb egy címke tartozhat csak, és természetesen ugyanazzal a címkével egy másik utasítás nem címkézhető meg. A címkedefiníció feldolgozása nagyon egyszerű. A címke kezdőzárójelének nincs jelentősége, csak a program jobb áttekinthetőségét szolgálja.

A <természetes szám> átalakítását a számátalakító alegység végzi el. A címkebezárójel hatására a vezérlőegység a számátalakító által kiszámított című szóba betölti az utasításszámláló értékét.

A vezérlésátadó utasítás feldolgozása szintén nagyon egyszerű akkor, ha a címke által meghatározott memóriaszó nem üres. Ebben az esetben a megadott szó érté két csupán át kell tölteni az USZ regiszterbe. Ez a helyzet minden predefinit címke esetében. A bonyolultabb feladat akkor lép fel, amikor a címke által meghatározott szó üres. Ekkor a gépnek át kell mennie egy kereső üzemmódba és meg kell keresnie a kérdéses címke értékét. (A problémát úgy is meg lehet oldani, hogy a gép mindig kereső üzemmódban indul, meghatározza a programban definiált összes címke értékét, majd a program végét jelző szimbólum hatására átmegy normális üzemmódba és megkezdi a program utasításainak a végrehajtását. Mi azonban most nem ezt a megoldást fogjuk ismertetni.)

Vezessük be a kereséshez a KR segédregisztert. A címkék és vezérlésátadó utasítások feldolgozása a következőképpen történhet:

Az utasítás kódja:	*Az aktív regiszter:*	*Az utasítás hatása:*
<címke vége>	nem UR	Jelölje k a számátalakító alegység SZR regiszterének tartalmát. Ha a k-adik szó tartalma nem üres (a címke predefinit), akkor a k-adik rekesz tartalma betöltődik az USZ regiszterbe, a számátalakító alapállapotot vesz fel és az aktív regiszter változatlan marad. Ha a k-adik szó tartalma üres, akkor a KR regiszter a k értéket veszi fel, egy segédregiszterbe kimentődik az aktív regiszter címe és az AR regiszter az UR regiszter címét veszi fel.
nem {	UR	Üres utasítás. Egy postdefinit címke keresésekor fordul elő.
{	UR	Az AR regiszter a KR regiszter címét veszi fel.
{	nem UR	Üres utasítás. Ez nem kereső üzemmódban fordul elő.
}	KR	a) Ha az SZR regiszter tartalma megegyezik a KR tartalmával, akkor az USZ regiszter tartalma betöltődik abba a szóba, amelynek címe SZR tartalma. A KR regiszter kiürül, a számátalakító alapállapotot vesz fel és az AR regiszter felveszi a címke-vége utasításkor kimentett címet. b) Ha az SZR regiszter tartalma nem azonos a KR regiszter tartalmával, akkor az USZ regiszter tartalma betöltődik abba a szóba, amelynek címe az SZR tartalma, a számátalakító alapállapotba kerül és az AR regiszter az UR regiszter címét veszi fel. Tehát további keresésre van szükség.
}	nem KR	Az USZ tartalma betöltődik abba a szóba, amelynek címe az SZR regiszterben van. A számátalakító alapállapotba kerül. Az aktív regiszter továbbra is változatlan.

4.13.1.5. A CIKLUSUTASÍTÁS FELDOLGOZÁSA

A következőkben nézzük meg, milyen segédregiszterekkel kell a gépünket kiegészíteni ahhoz, hogy az alkalmas legyen a ciklusutasítás végrehajtására is. Tegyük fel, hogy a ciklusutasítás formája a következő:

<ciklusutasítás>::= **for** <ciluspараméter><kezdőérték kif.>

step <lépés kif.> **until** <végérték kif.>

do <ciklusmag> **cend** ;

<ciluspараméter>::=<skaláris változó>;

<kezdőérték kif.>::=<aritmetikai kifejezés>;

<lépés kif.>::=<aritmetikai kifejezés>;

<végérték kif.>::=<aritmetikai kifejezés>;

<ciklusmag>::=<utasítás> | <ciklusmag> = ; <utasítás>

Példa egy ciklusutasításra:

for *i* 1 **step** 1 **until** 2*n **do** a+ *i* =>a **cend**

Mint látni fogjuk, a feldolgozáshoz célszerű bevezetni a ciklusmag végét jelentő **cend** szimbólumot.

A ciklusutasítás feldolgozásához kézenfekvő bevezetni a ciluspараméter címének, a lépésköznek, a végértéknek és a ciklusmag kezdőcímének a tárolására egy regiszternégyest. Mivel azonban a ciklusok egymásba skatulyázását is megengedjük, ezért annyi ilyen regiszternégyesre van szükségünk, amekkora ciklusmélységet engedjük meg. Ekkor viszont szükségünk van még egy regiszterre, amely mindenkor az aktuális ciklusregiszter négyesének a címét tartalmazza. A felsorolt regisztereknek a szerepe részletesebben a következő:

A ciluspараméter-regiszter (PR). A ciluspараméter címét tartalmazza a ciklusutasítás végrehajtása folyamán. A PR regiszter a <ciklus paraméter> utasítás hatására kap értéket.

A lépésközregiszter _(PLR). A PLR regiszter a lépéskifejezés értékének az őrzésére szolgál. A lépéskifejezés értékének kiszámítása után az **until** utasítás hatására kap értéket.

A végértékregiszter (VÉR). A VÉR regiszter a <végérték kifejezés> értékének a megőrzésére szolgál. A <végérték kifejezés> értékének kiszámítása után a **do** utasítás végrehajtásának eredményeképpen kap értéket.

A ciklusmag kezdőcímének regisztere (CKR). A CKR regiszterben tároljuk a ciklusmag első alapjelének a címét. Ez a regiszter ugyancsak a **do** utasítás végrehajtásakor kap értéket.

A ciklusmélység-számláló (CSZ). A ciklusregiszter négyesei is egy hierarchikus struktúrát alkotnak. A o-dik szinten álló regiszternégyes a külső ciluspараméterеinek a megőrzésére szolgál. Minden a ciklus kezdetét jelző **for** utasításhoz a soron következő szint regiszternégyesei tartoznak. A mindenkor aktuális regiszternégyes „címét”, azaz szintszámát őrzi a CSZ regiszter. A CSZ regiszter értéke kezdetben -1, minden **for** utasítás 1-gyel nő és minden **cend** utasítás hatására 1-gyel csökken.

A továbbiakban a k-adik szinthez tartozó regiszterek megnevezésére a PR k , PLR k , VÉR k , CRK k elnevezéseket használjuk. A ciklusutasítás feldolgozásának utasításai részletesen a következők:

A művelet kódjele:	*Az aktív regiszter:*	*Az utasítás hatása:*
for	BOR 0	A CSZ regiszter tartalma 1-gyel megnövekszik. Ennek a regiszternek a hossza úgy van megállapítva, hogy túlcsordulása ciklusmélység túlcsordulást okoz. Az AR regiszter a megnövelt CSZ regiszter értékét veszi fel. Azaz PR k lesz aktív, ahol *k* a CSZ regiszter tartalma.
<skaláris változó>	PR k	Az ÚR regiszter tartalma, amely most a skaláris változó címét tartalmazza, betöltődik PR k -ba. A

változó címét tartalmazza, betöltődik PR k -ba. A ciklusparaméter értékének a kiszámításához az AR regiszter a BOR 0 regiszter címét veszi fel.

step	ER 0	Az ER 0 regiszter ekkor a kezdőérték kifejezés értékét tartalmazza, ezért az utasítás hatása az, hogy ER 0 értéke átmásolódik abba a memóriaszóba, amelynek címét az aktuális PR k tartalmazza. Ezután a BOR 0 , OR 0 , JOR 0 és ER 0 regiszterek tartalma kiürül. Az AR regiszter a BOR 0 regiszter címét veszi fel.
step	OR 0	Ez az eset akkor áll elő, amikor a kezdőérték kifejezés értékét a BOR 0 regiszter tartalmazza. Ezért ilyenkor a BOR 0 tartalma tárolódik abba a memóriaszóban, amelynek címét az aktuális PR k regiszter tartalmazza. Ezután a BOR 0 tartalma kiürül és AR a BOR 0 címét veszi fel.
until	ER 0	Ekkor a lépéskifejezés értéke az ER 0 regiszterben van. Az ER 0 regiszter értékét kell tehát átmásolni az aktuális PLR k regiszterbe. Ezután a BOR 0 , OR 0 , JOR 0 és ER 0 regiszterek tartalma kiürül. Az AR regiszter a BOR 0 címét veszi fel.
until	OR 0	Az eredmény most BOR 0 -ban van. A BOR 0 tartalma betöltődik az aktuális PLR k regiszterbe, majd BOR 0 kiürül és AR felveszi a BOR 0 regiszter címét.
do	ER 0	Most az ER 0 regiszterben a végérték kifejezés értéke állt elő, ezért az utasítás hatása az, hogy az ER 0 tartalma átmásolódik az aktuális VÉR k regiszterbe, az USZ regiszter tartalma megőrződik az aktuális CKR k regiszterben, majd a BOR 0 , OR 0 , JOR 0 , ER 0 regiszterek tartalma kiürül. Az AR regiszter BOR 0 címét veszi fel.
do	OR 0	**A** BOR 0 regiszter tartalma átmásolódik VÉR k regiszterbe, USZ tartalma megőrződik az aktuális CKR k regiszterben, a BOR 0 tartalma kiürül és AR felveszi a BOR 0 regiszter címét.
cend	BOR 0	a) Ha annak a szónak a tartalmához, amelynek címét az aktuális PR k tartalmazza, hozzáadjuk az aktuális PLR k tartalmát és eredményül az aktuális VÉR k tartalmánál nem nagyobb értéket kapunk, akkor annak a szónak a tartalmához, amelynek a címe a PR k regiszterben van, hozzáadódik a PLR k tartalma, CKR k regiszter tartalma átmásolódik az USZ regiszterbe és AR a BOR 0 regiszter címét veszi fel. b) Ha annak a szónak a tartalmához, amelynek címét az aktuális PR k tartalmazza, hozzáadjuk az aktuális PLR k tartalmát és eredményül a VÉR k regiszter tartalmánál nagyobb értéket kapunk, akkor a CSZ regiszter tartalma 1-gyel csökken és az AR regiszter BOR 0 címét veszi fel. Ha a CSZ tartalma a művelet után -1-nél kisebb, akkor hibajelzés következik, mert több **cend** utasítás van, mint **for** utasítás.

Feladatok:

4.15. Kísérjük végig a formulavezérlésű gép kifejezésfeldolgozó alegységének működését a következő értékadó utasítások feldolgozása során:

$$a+((b/c)-d) \Rightarrow x,$$
$$((a-b)/d)-e \Rightarrow x$$

4.16. Kísérjük végig a formulavezérlésű gép működését a következő számok feldolgozása során:

0.321 3

22.0 3

4.17. Kísérjük végig a formulavezérlésű gép megfelelő regisztereiben az értékek változását a következő deklarációs utasítások feldolgozása során:

$$n-m \Rightarrow \otimes n \Rightarrow = \rangle X$$
$$n*m \Rightarrow = \rangle Y$$

IRODALOM:

4.1. Bakos, T.: COBOL programozási nyelv. Műszaki Könyvkiadó (1974)

4.12. Lőcs, Gy., Sarkadi, N., Szlankó, J.: A BASIC programozási nyelv. Műszaki Könyvkiadó (1976)

4.13. Lőcs, Gy.: Az ALGOL 60 programozási nyelv. Műszaki Könyvkiadó (1967)

4.14. Lőcs, Gy., Vigassy, J.: FORTRAN programozási nyelv. Műszaki Könyvkiadó (1973)

4.30. Rákosi, M.: PL/1 programozási nyelv. Műszaki Könyvkiadó (1974)