VEZÉRLŐEGYSÉGEK

Tartalom

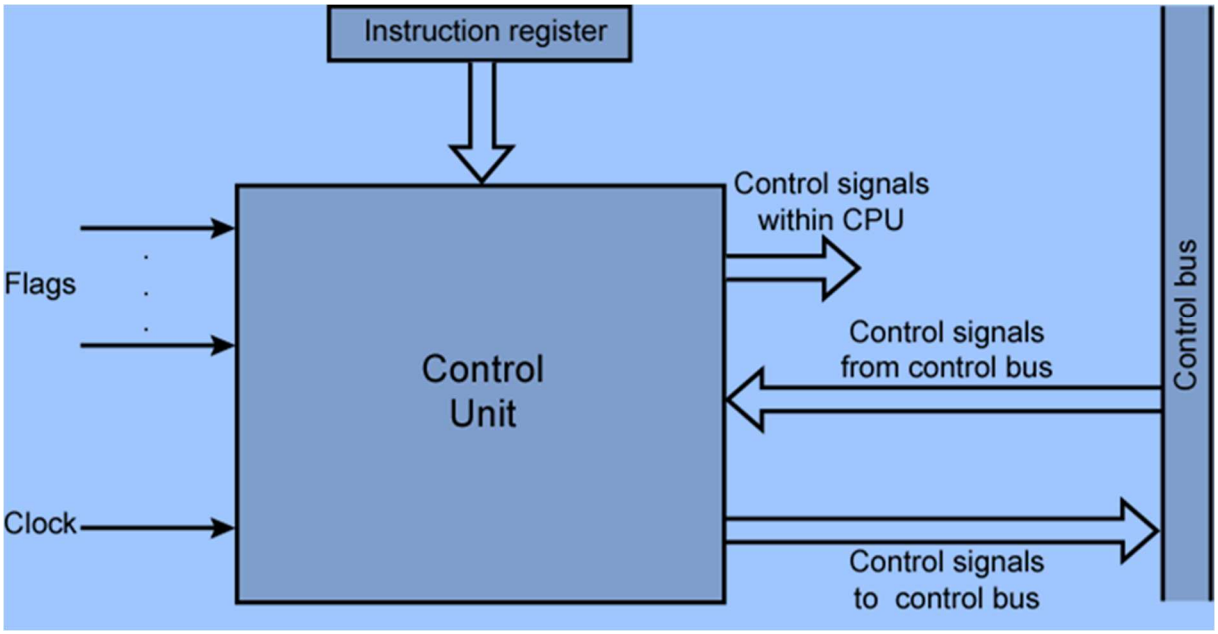
Vezérlőegységek fajtái és jellemzői

A processzor elemei

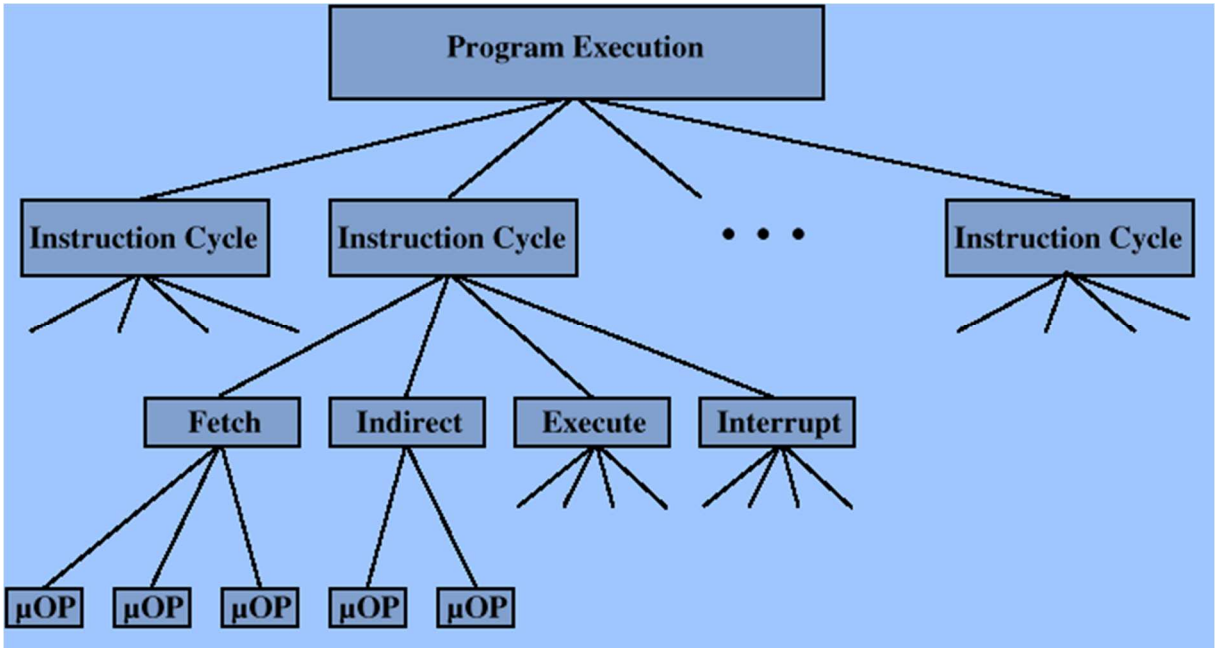
ALU, regiszterek, adat utak, vezérlés

A vezérlés modellje

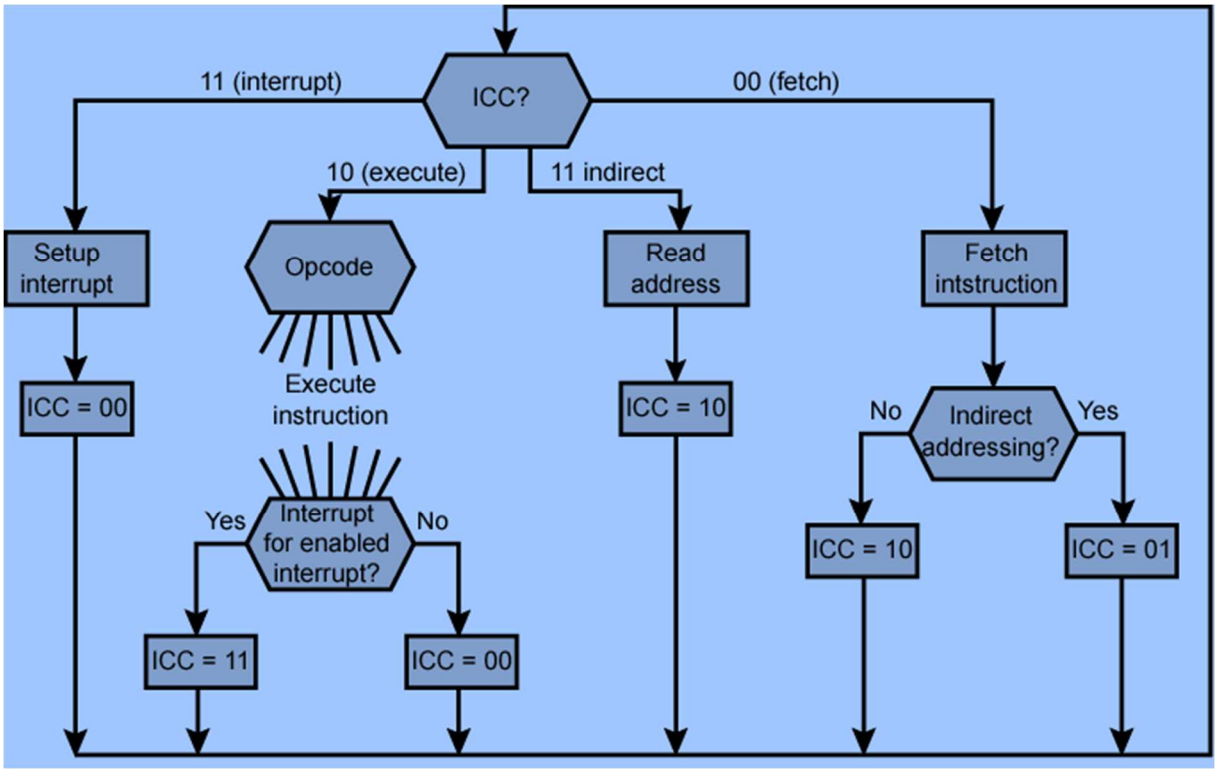
Forrás: - haszn ---- jóó 15_Control Unit -- Instruction cycyle & benne 8085.ppt



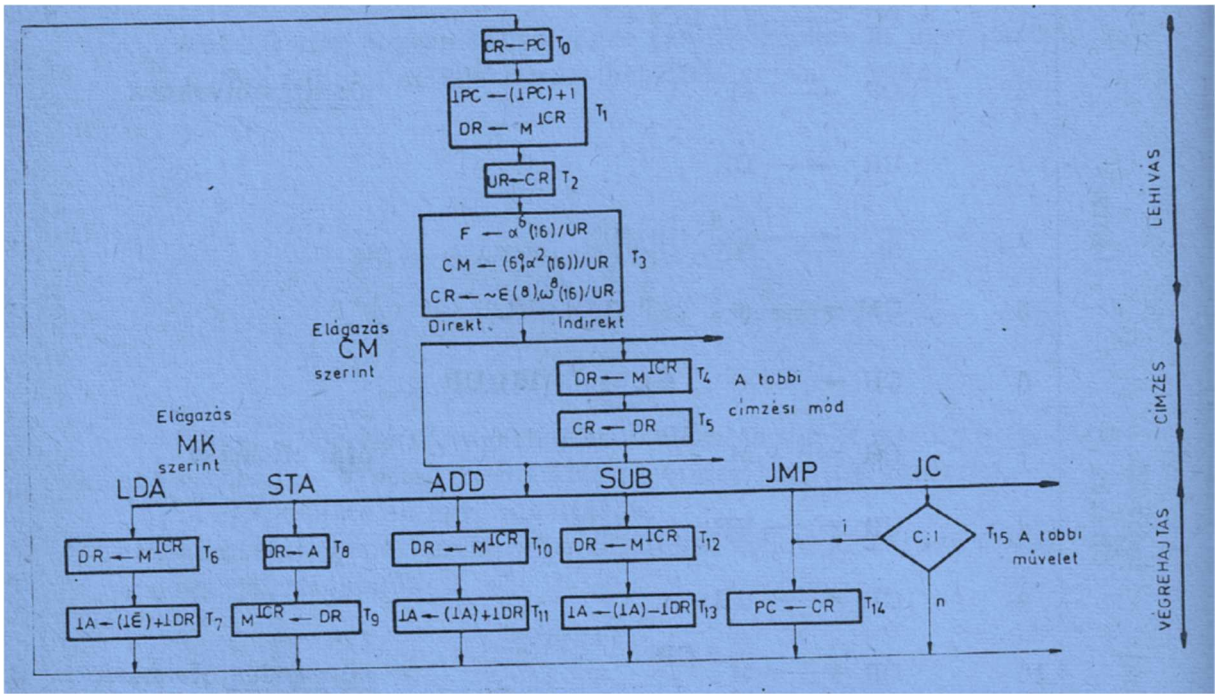
Constituent Elements of Program Execution

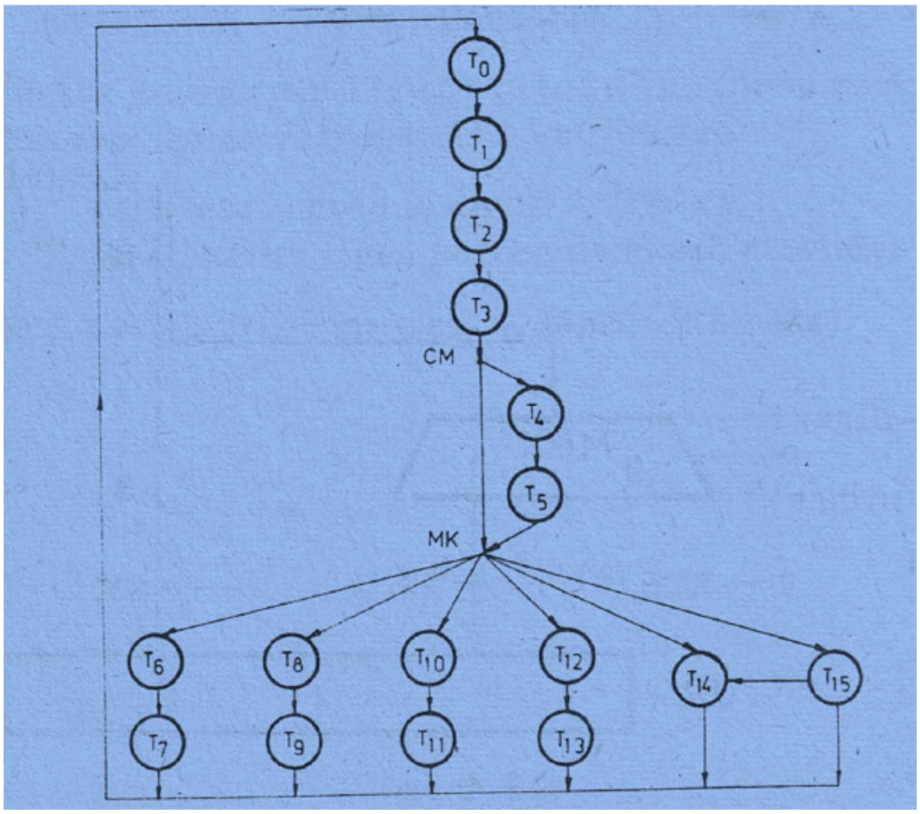


Flowchart for Instruction Cycle

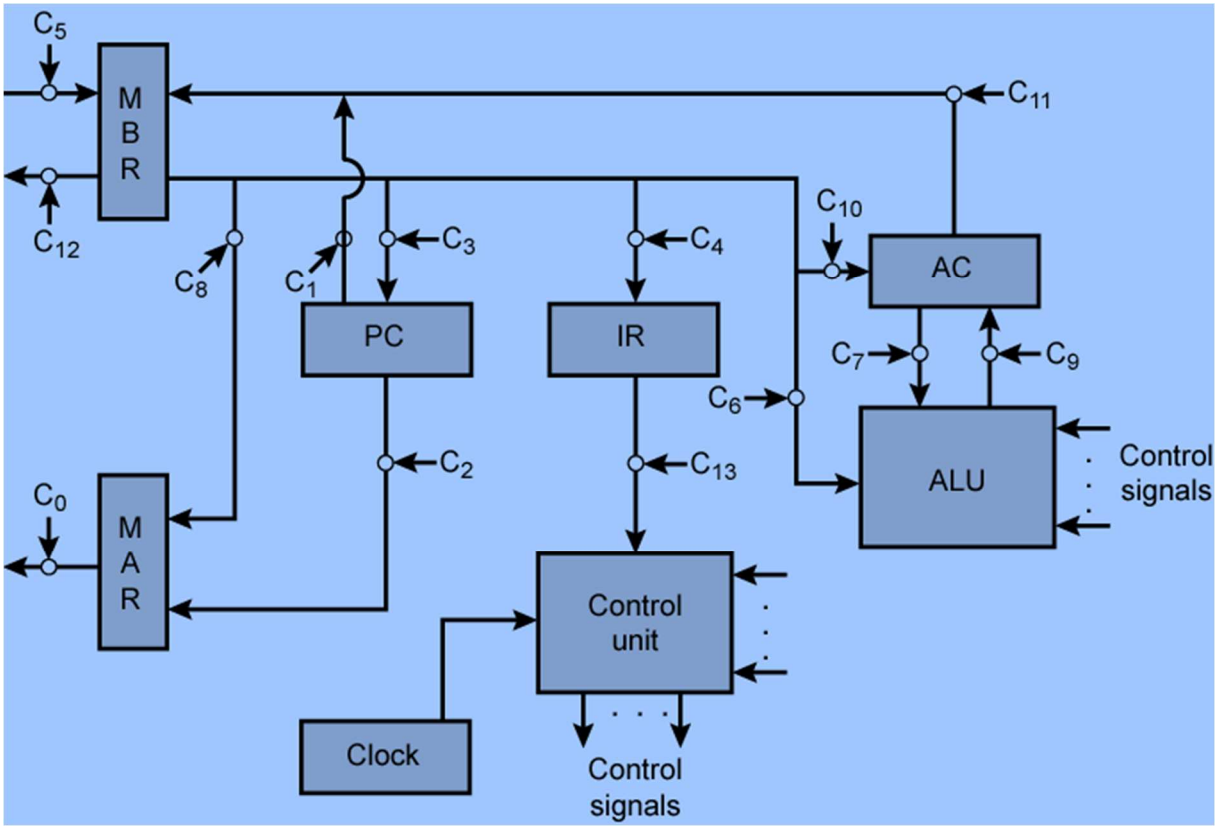


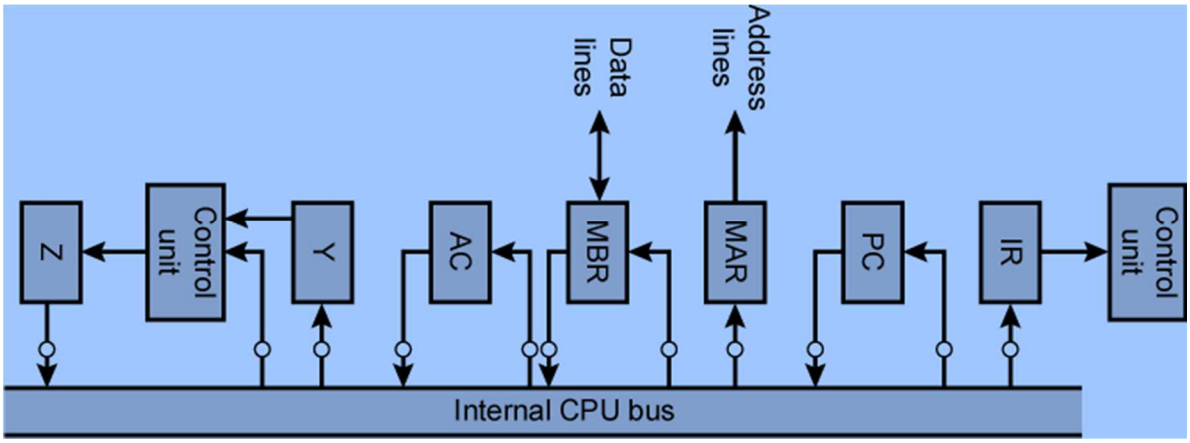
Forrás BohusM

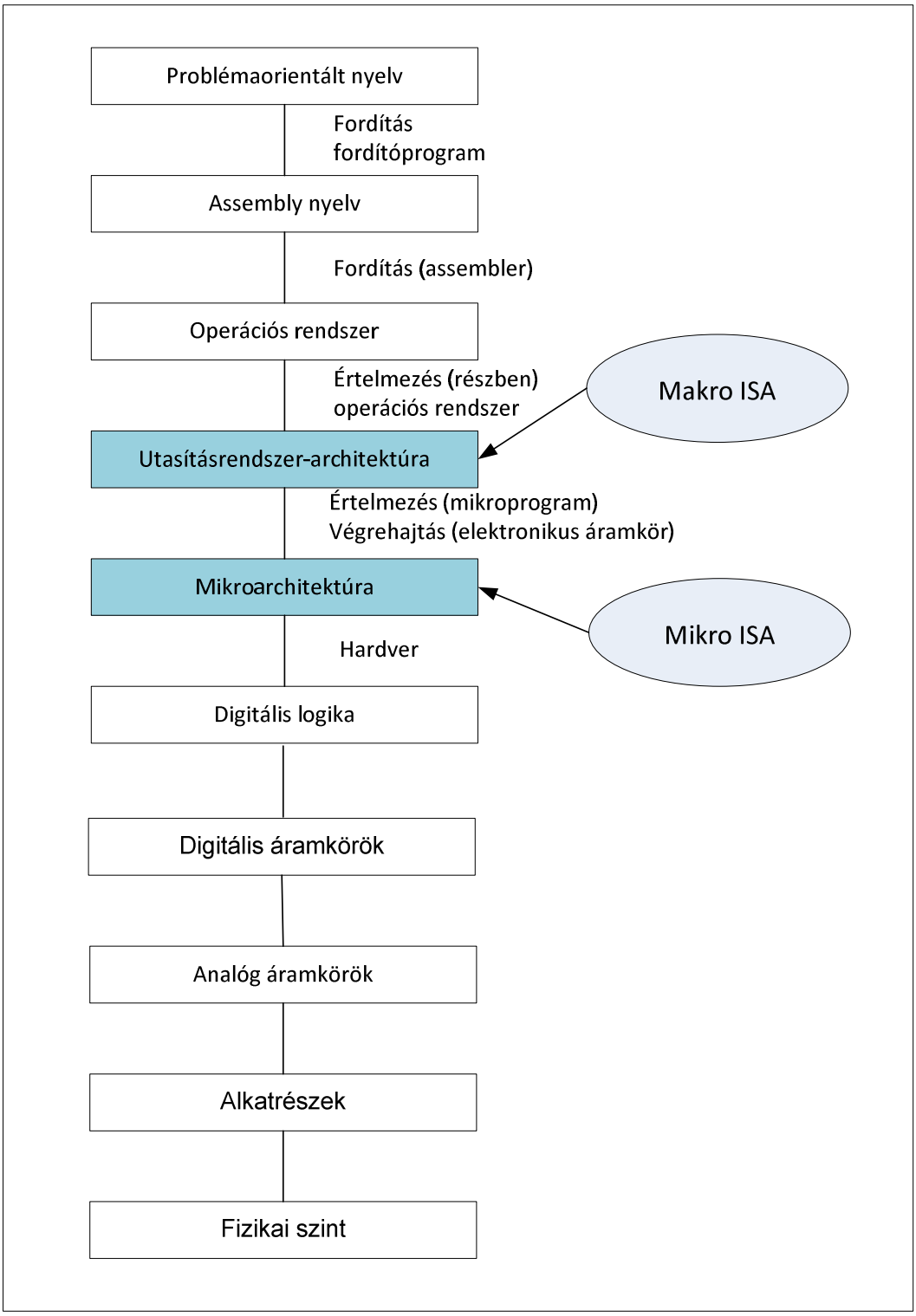




Forrás: - haszn ---- jóó 15_Control Unit -- Instruction cycyle & benne 8085.ppt



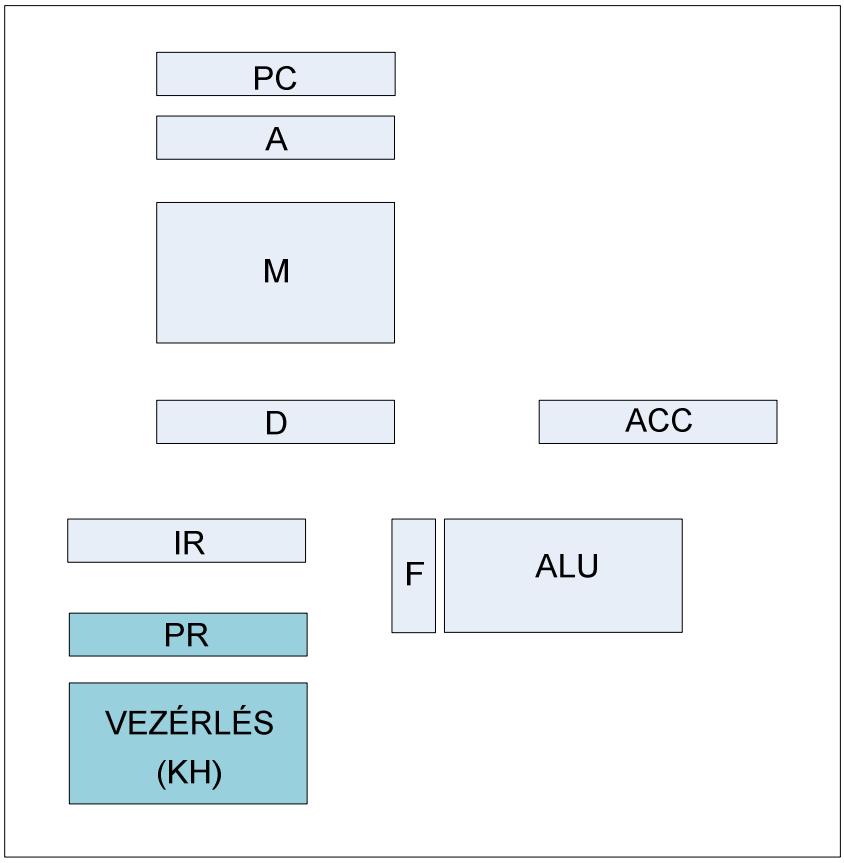

Address lines
Data lines
Control unit
IR
PC
MAR
MBR
AC
Y
Control unit
Z
Internal CPU bus

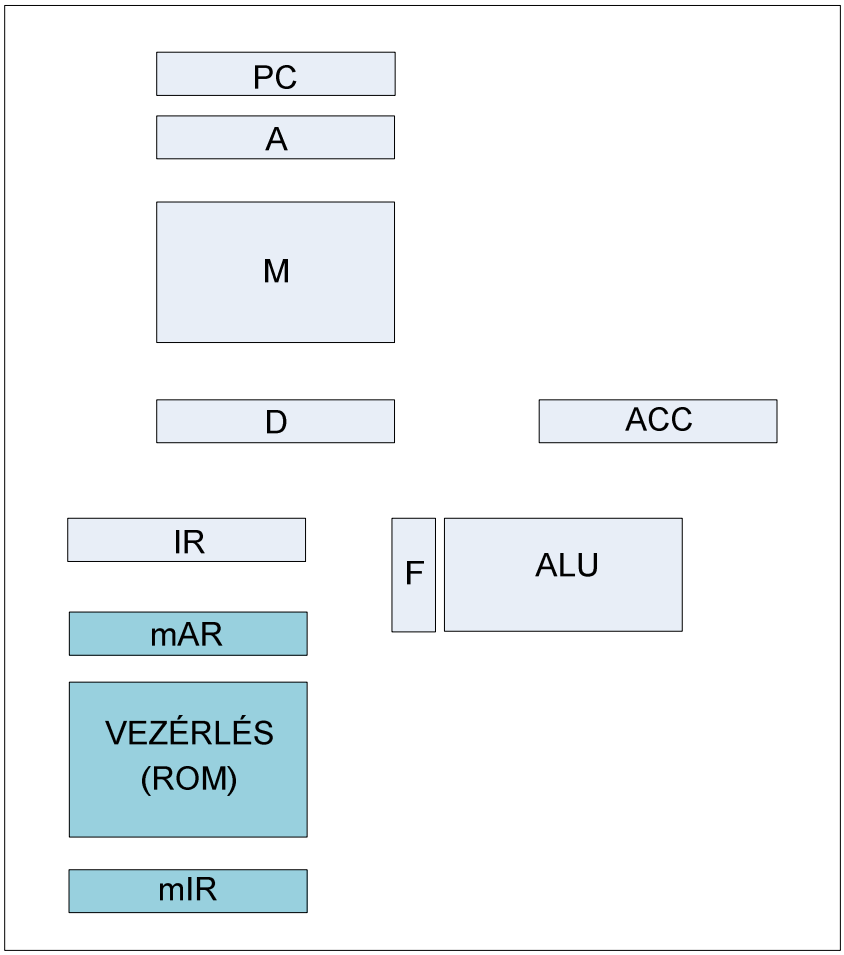

Problémaorientált nyelv
Fordítás
fordítóprogram
Assembly nyelv
Fordítás (assembler)
Operációs rendszer
Értelmezés (részben)
operációs rendszer
Makro ISA
Utasításrendszer-architektúra
Értelmezés (mikroprogram)
Végrehajtás (elektronikus áramkör)
Mikroarchitektúra
Mikro ISA
Hardver
Digitális logika
Digitális áramkörök
Analóg áramkörök
Alkatrészek
Fizikai szint

A vezérlőegységek csoportosítása a tervezés módszere szerint

- A **huzalozott (random) logika** azt jelenti, hogy a vezérlést valamilyen aszinkron vagy szinkron sorrendi hálózattal valósítják meg. Előnye a gyorsaság, hátránya a tervezés bonyolultsága, illetve az, hogy módosítás esetén tipikusan újra kell tervezni.
- **Fázisregiszteres vezérlőegység** esetén a tervezést egy részben kötött struktúra segíti. A hagyományos szinkron sorrendi hálózat tervezéssel szemben itt a tervező nem állapotgráf, hanem folyamatábra alapján dolgozik. A fázisregiszter (ez lehet egy számláló is) tárolja a vezérlő belső állapotát, ennek értékét kell megfelelően átállítani (pl. betöltéssel) illetve ennek értéke alapján kell előállítani a kimeneti vezérlést (dekóderrel, kombinációs hálózattal). A módosítás itt is problémát okoz.
- A **mikroprogramozott vezérlőegység** a legrugalmasabb megoldás, itt módosításkor kedvező esetben csupán a ROM-ban tárolt mikroprogramot kell átírni. Ennek ára az, hogy ez a leglassabb megoldás. Működését, fajtáit és programozását egy kicsit bővebben megismerjük.

Működéséről az előadáson elhangzottakon túl érdeklődőknek ajánljuk: [1]: 95-100. o. Benesóczky

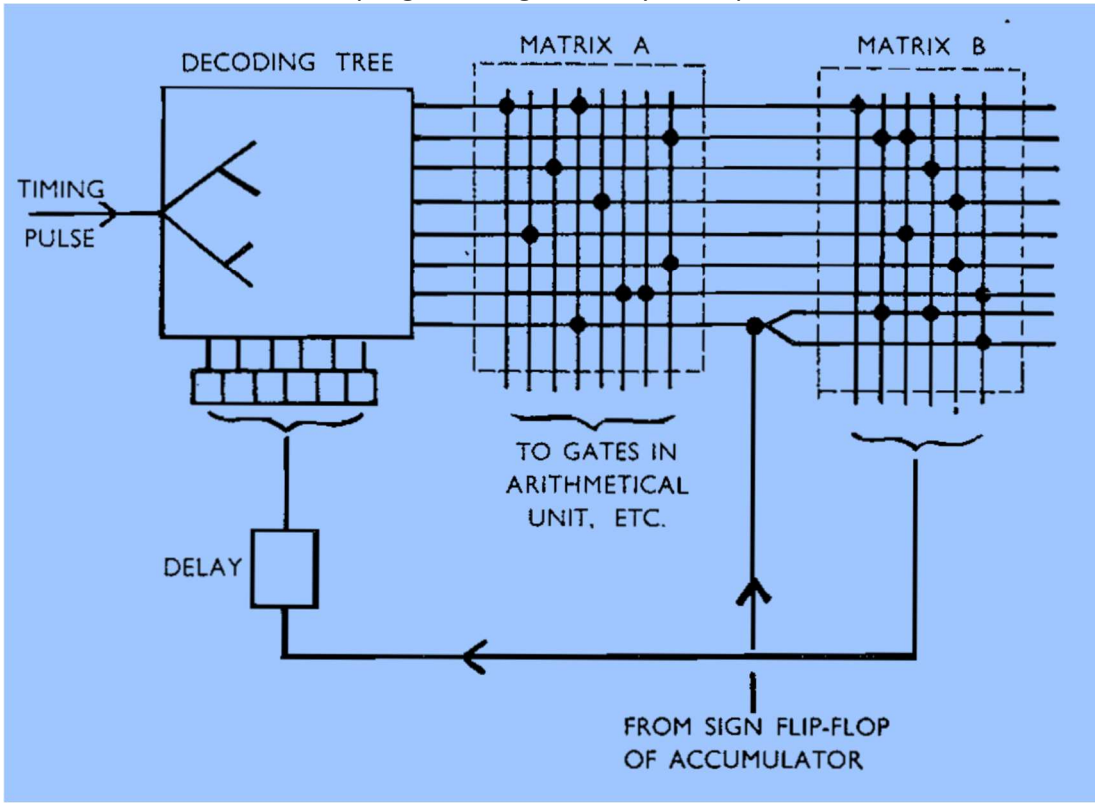



PC
A
M
D
ACC
IR
F
ALU
mAR
VEZÉRLÉS
(ROM)
mIR

A mikroprogramozott vezérlés

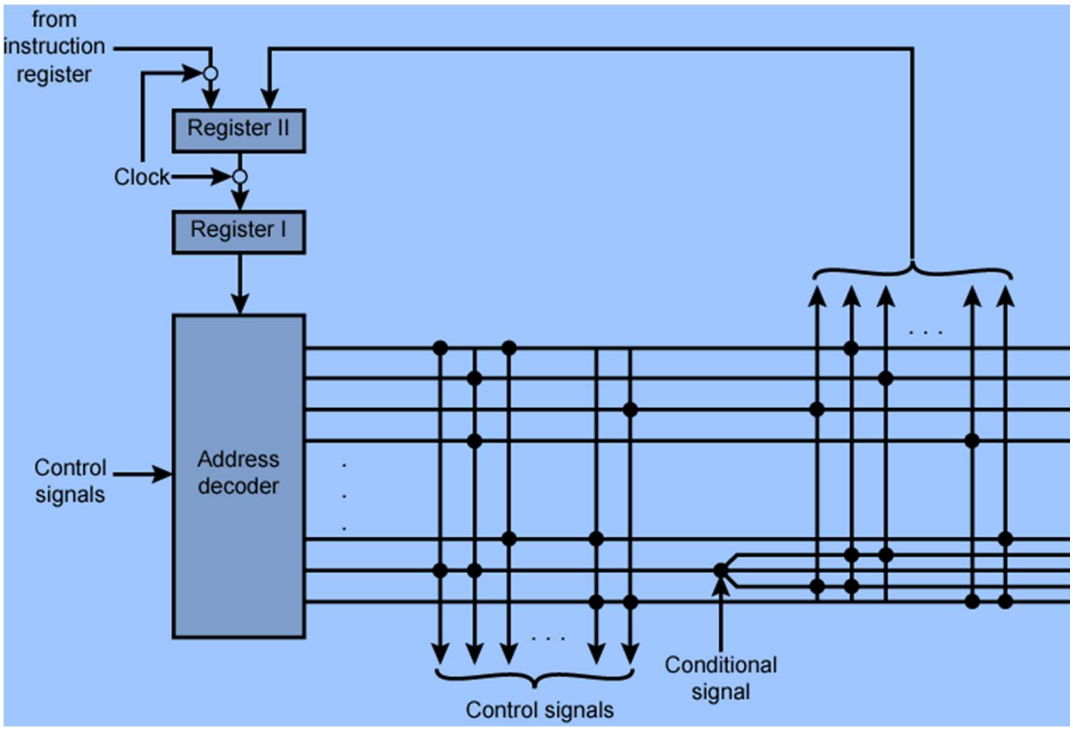
Mikroprogramozás: Wilkes

Forrás: - haszn ---- Wilkes Microprogramming BestWay 1951.pdf

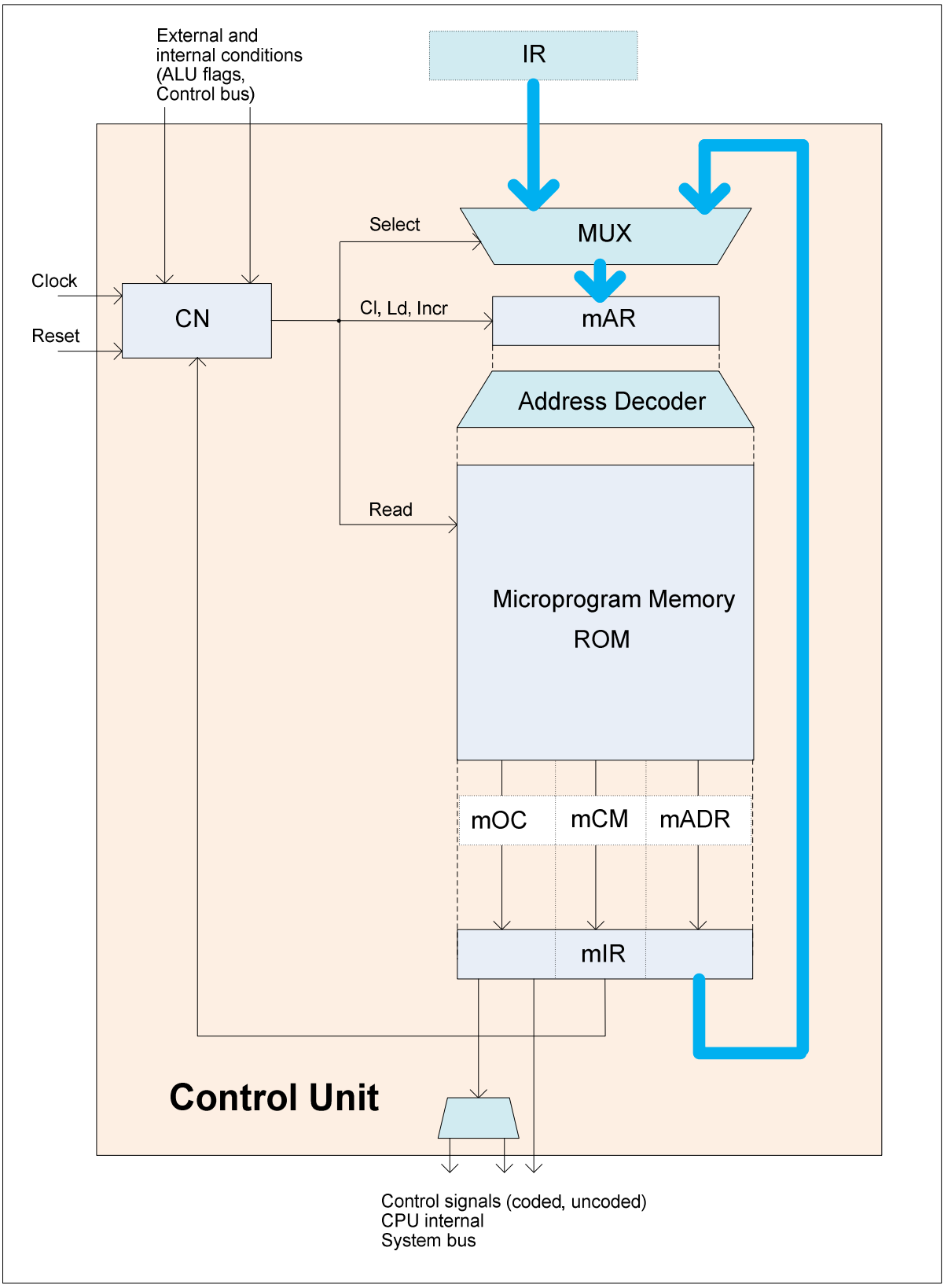


Forrás: - haszn --- 16_Micro-Programmed Control --Putnam .ppt

Wilkes's Microprogrammed Control Unit



A mikroprogramozott vezérlőegység felépítése



Mikroutasítás formátumok

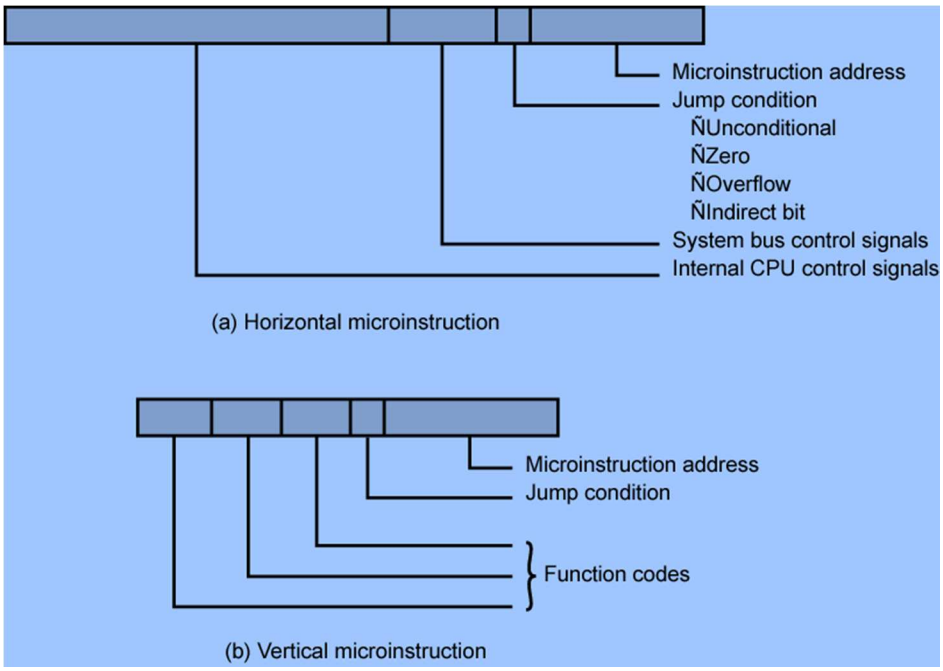
A mikroprogramozott vezérlö ún. **mikroutasításokat hajt végre**. Egy mikroutasítás 3 részből áll:

<műveleti rész>< feltétel rész>< cím rész>

Az *i*-edik állapotban, a mikroutasítás feltétel része kiválasztja a figyelni kívánt feltételt. A címlogika a kiválasztott feltétel, és a mikroutasítás címrésze függvényében előállítja a következő mikroutasítás címét. A mikroutasítás műveleti része adja a kimenetet.

Forrás: - haszn --- 16_Micro-Programmed Control --Putnam .ppt

Typical Microinstruction Formats



mAM =

INC(Increment)

JP(Jump)

JC(Jump Condition)

µMK	µCM	µCÍM
mOC	mAM	mADR

Mikroutasítás formátumok

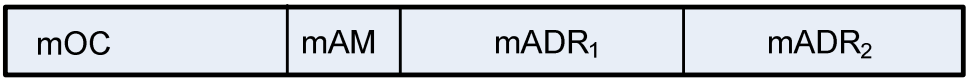
- horizontális

- vegyes

- vertikális

Elágazás - következő mikroutasítás címe

- kettő címmel

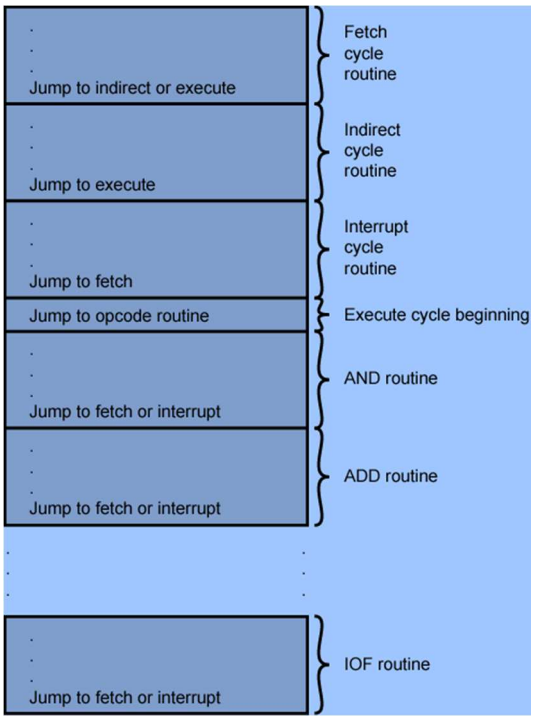


- egy címmel

- változó formátummal

A mikroprogram szerkezete

Forrás: - haszn --- 16_Micro-Programmed Control --Putnam .ppt

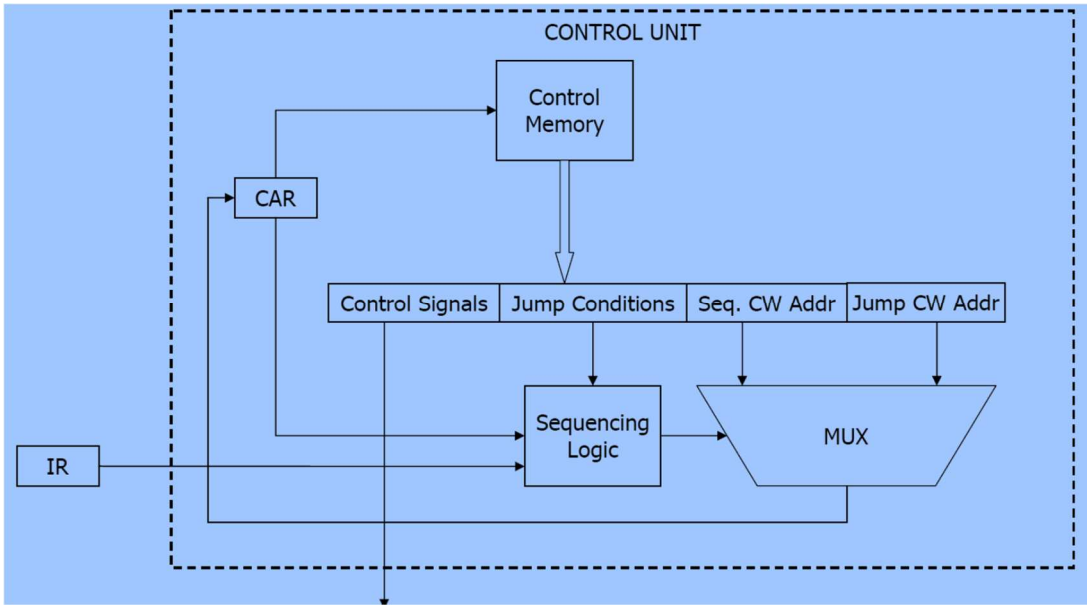


Mikroprogram elágazás

Forrás: - haszn ---- Ch_15_95-rev3-1 Micro-programmed control .pdf

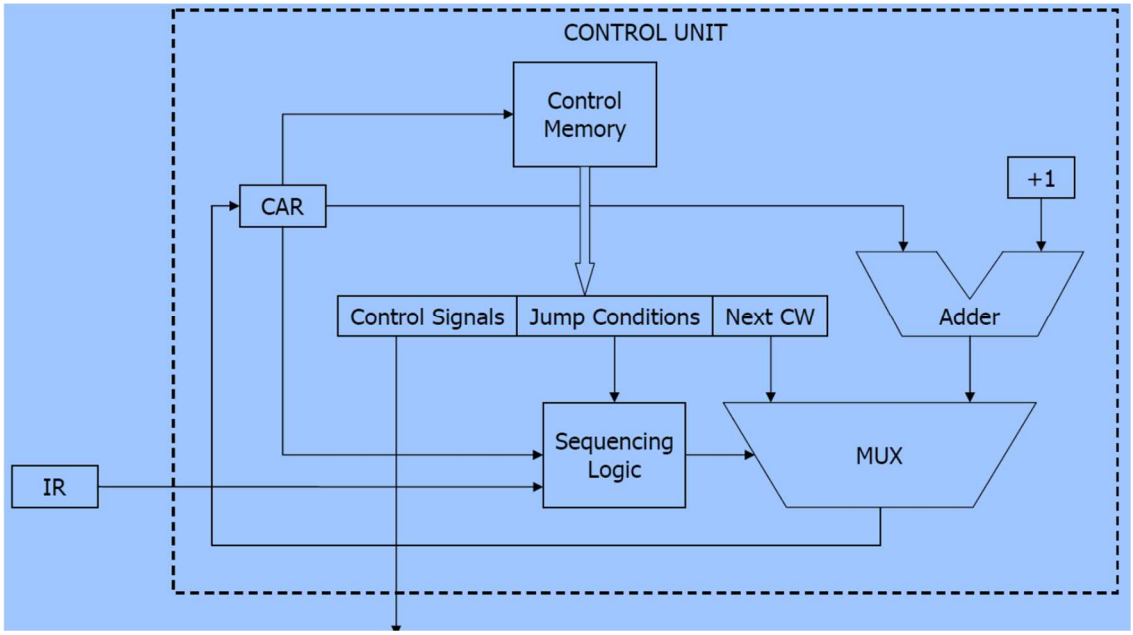
Rev. 3.1 (2006-07) by Enrico Nardelli

Elágazás vezérlés: 2 címmező



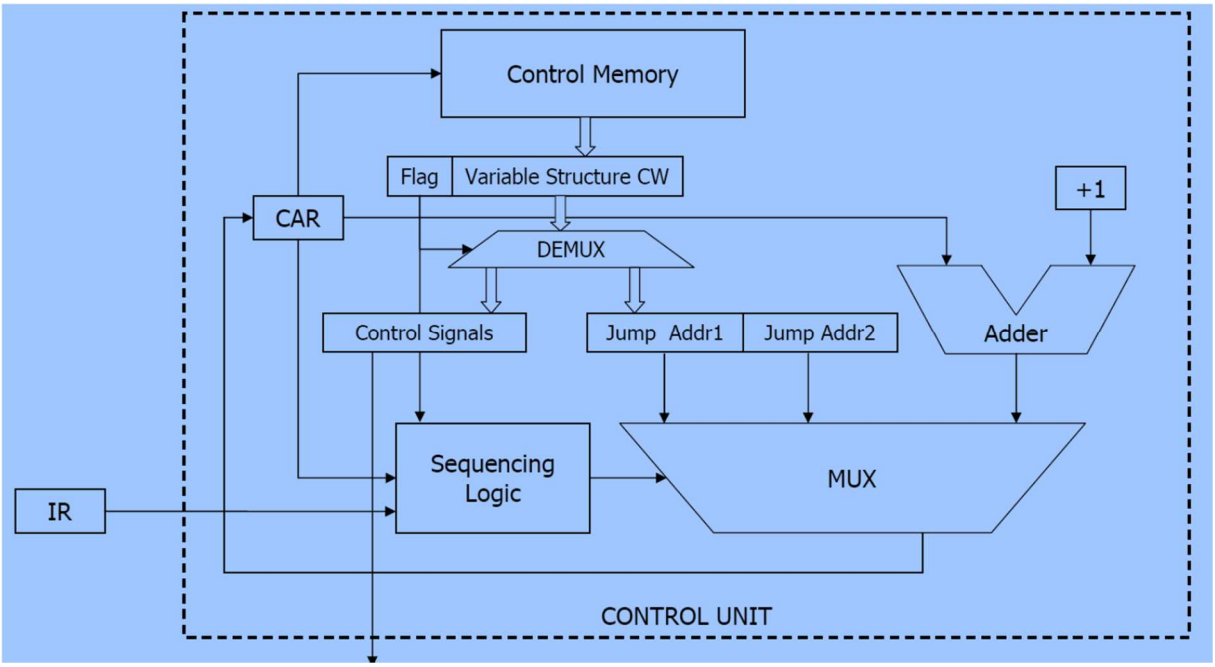
- címek az ugrás és a nem ugrás eseteire
- hosszabb CW

Elágazás vezérlés: 1 címmező



- cím az ugráshoz, nem ugrás estén +1
- rövidebb CW, lassúbb

Elágazás vezérlés: változó utasítás formátum



- csak címinformáció vagy csak vezérlő információ
- hátrány: hosszabb mikroprogram

A mikroutasítás-ciklus

1. mikroutasítás olvasás MM-ből
2. mikroutasítás beolvasása a mIR-be
3. vezérlőjelek generálása a CPU-nak és a rendszersínnek; és információ a következő mADR generáláshoz
4. következő mADR meghatározása az alábbiak szerint
    a. JUMP feltétel és a következő cím az mIR-ben
    b. IR és ALU flagek
    c. a MAR-ban lévő allapot információ
5. a következő mikroutasítás címének betöltése a MAR-ba, amely lehet
    a. a jelenlegi cím +1
    b. ugrás
        i. új mikro-procedúra ugyanabban a gépi utasításban
        ii. új mikro-procedúra egy új gépi utasításban

A huzalozot és a mikroprogramozott vezérlés összehasonlítása

A mikroprogramozott vezérlés

- olcsóbb
- sokkal könnyebb ellenőrizni és módosítani

A huzalozott vezérlés gyorsabb

A mikroprogramozott vezérlés főként CISC architektúrákhoz használatos,

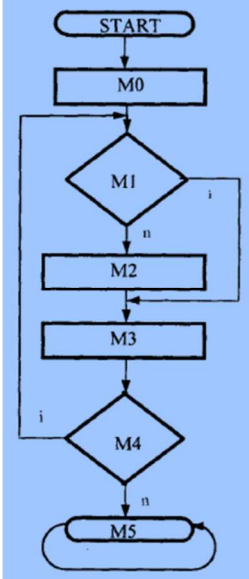
míg RICS architektúrákban huzalozott vezérlést alkalmaznak.

Mikroprogram készítése

Forrás Horvárth L

Illusztratív példaként [HORVÁTH] alapján elkészítjük egy egyszerű folyamatábra mikroprogramját.

Példa feladat:
készítsünk mikroprogramot az alábbi folyamatábrához:



Kódoljuk is a szimbolikusan megadott programot, ahol:

a művelet		kódja
M0	...	0 0 0
M1	...	0 0 1
M2	...	0 1 0
M3	...	0 1 1
M4	...	1 0 0
M5	...	1 0 1
NOP	...	1 1 1

és a címzési

mód		kódja
INC	...	0 0
JP	...	1 1
JC	...	1 0

5.3. ábra A szorzási algoritmus szimbolikus folyamatábrája

ROM ADDR	mOC		mAM		mADR	Code	Comment
	Mnemonic	Code	Mnemonic	Code			
N+0	I0	000	INC	00	-		
N+1	I1	001	JC	10	N+3		
N+2	I2	010	INC	00	-		
N+3	I3	011	INC	00	-		
N+4	I4	100	JC	10	N+1		
N+5	I5	101	JP	11	N+5		

Itt van maga a feladat: bináris szorzás ismételt összeadással

a) Szorzás ismételt összeadással

Írjuk fel a műveletet n bites operandusokkal:

$$A \cdot B = A \cdot (b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_1 2^1 + b_0 2^0)$$

Átalakítva: $A \cdot B = (\ldots(0 + 2^n \cdot A \cdot b_0)2^{-1} + 2^n \cdot A \cdot b_1)2^{-1} + \ldots + 2^n \cdot A \cdot b_{n-1})2^{-1}$

Elemezzük a fenti összefüggést

(i) A műveletet a legkisebb súlyozású b_0 bittel kezdjük.

(ii) Nem a szorzandót, hanem annak n-szer balra léptetett változatát - 2^n -A-t - szorozzuk meg. Ebből következik, hogy a szorzat legkisebb helyértékű bitje n bittel jobbra lesz a szorzandó azonos helyértékű bitjétől.

(iii) A szorzásokat összegzések követik.

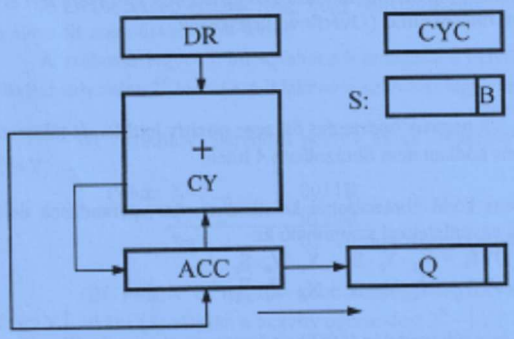
(iv) A részletösszeg tárolót kezdetben törölni kell!

(v) A részletösszeget jobbra kell léptetni. Ezt jelenti a 2^{-1}-vel való szorzás.

(vi) Az algoritmus n lépéses.

(vii) Az algoritmusban nincs előjel kezelés, tehát csak előjel nélküli pozitív számok szorzására alkalmas. (Természetesen az operandusok előjelének antivalenciája megadja a szorzat előjelét.)

Az algoritmushoz tartozó regiszter elrendezés a 3.1. ábrán látható.



3.1 ábra Szorzás ismételt összeadással regiszterelrendezése

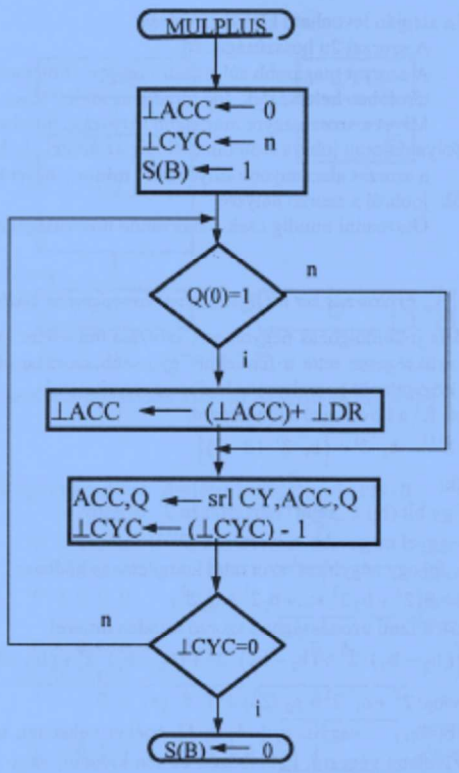
A regiszterek tartalma szorzás *előtt* (amit előre be kell állítani):

DR - szorzandó
Q - szorzó

A regiszterek tartalma szorzás *után*:

ACC,Q - szorzat
DR - szorzandó

A folyamatábra a 3.2. ábrán látható. Az ábrákon megjelenő CYC (**Cycle Counter**) **ciklusszámláló** a szorzó bitjeinek számával (n) töltődik fel a **kezdeti beállítás** állapotában. Az algoritmus minden lépését CYC dekrementálása követi, tehát CYC=0 jelzi a művelet végét. A szorzó áramkör mindenkori állapotát az S (Status) regiszter B (Busy) bitje jelzi. A művelet elején S(B)←1 állításával az eszköz **foglaltságát**, S(B)←0 állapottal az eszköz **üzemkészségét** (Ready) jelzi a külvilág felé.



3.2. ábra Szorzás ismételt összeadással folyamatábrája

Nézzünk egy számpéldát is: A=7 ; B=5 . (A számpéldában tekintsünk minden számot bi-nárisnak, és csak az ettől eltéröeket jelöljük.) Legyen n=3!

	szorzandó	$b_2b_1b_0$ szorzó	
	1 1 1	1 0 1	
Az induló részletösszeg....	0 0 0		
$2^n \cdot A \cdot b_0$ ...	1 1 1		
Első részletösszeg	1 1 1		léptetés jobbra
	0 1 1	1	
$2^n \cdot A \cdot b_1$ ...	0 0 0		
Második részletösszeg ...	0 1 1	1	léptetés jobbra
	0 0 1	1 1	
$2^n \cdot A \cdot b_2$ ...	1 1 1		
Harmadik részletösszeg ...	1 0 0	1 1	léptetés jobbra; *a keletkezett átvitel: belép*
A szorzat	1 0 0	0 1 1	

A számpélda alapján levonható következtetések:

(i) A szorzat 2n hosszúságú lett.

(ii) A szorzat magasabb súlyozású - **major** - bitjei a részletösszeg tárolóban keletkeztek. Ide mindig az átvitel lépett be.

(iii) Mivel a szorzó egyre magasabb súlyozású bitjeire van szükség, a szorzót is célszerű folyamatosan jobbra léptetni. (Mindig az átvitel lép be!)

(iv) a szorzat alacsonyabb súlyozású - **minor** - bitjei folyamatosan beléptethetők jobbról a szorzó helyére.

(vi) Összeadni mindig csak a szorzandó hosszúságában, tehát n biten kellett.

b) Szorzás két bit figyelésével komplemens kódban

A valósidejű digitális jelfeldolgozás nagyon sok szorzási műveletet igényel. Ezen feladatok tömeges elterjedése szükségessé tette a fentieknél gyorsabb szorzási algoritmusok kifejlesztését, melyben a számok közvetlenül komplemens kódban szorozhatók.[4]

Először írjuk fel a következő azonosságot:

$$b_i \cdot 2^i = b_i \cdot 2^{i+1} - b_i \cdot 2^i = \left[b_i \cdot 2^i \cdot (2-1)\right]$$

Értelmezzük:

(i) Egy bit (b_i) a „saját" helyértékén 2^i *negatív,*

(ii) eggyel magasabb helyértéken (2^{i+1}) *pozitív.*

Most írjunk fel egy négybites[5] szorzatot komplemens kódban:

$$A \cdot B = A \cdot (-b_3 2^3 + b_2 2^2 + \ldots + b_1 2^1 + b_0 2^0)$$

Alkalmazzuk a fenti azonosságot a szorzó minden bitjére!

$$A \cdot B = A \cdot \left[(b_2 - b_3) \cdot 2^3 + (b_1 - b_2) \cdot 2^2 + (b_0 - b_1) \cdot 2^1 + (b_{-1} - b_0) \cdot 2^0\right] =$$

$$A \cdot (c_3 \cdot 2^3 + c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0) \qquad (*)$$

ahol: $c_i = b_i - b_{i+1}$ vagyis: $c_i \{-1, 0, 1\}$ értéket vehet fel, tehát azonos b_i , b_{i-1} bitek esetén nem kell műveletet végezni, különbözők esetén kivonni, vagy összeadni kell. A b_{-1} a 2^{-1} súlyozású bit lenne, egészeknél természetesen zérus. Az algoritmust leíró forma (*)-ból:

$$A \cdot B = ((((0 + 2^4 \cdot A \cdot c_0)2^{-1} + 2^4 \cdot A \cdot c_1)2^{-1} + 2^4 \cdot A \cdot c_2)2^{-1} + 2^4 \cdot A \cdot c_3)2^{-1}$$

Az algoritmus a fentiek alapján általánosítva:

(i) A szorzandót n-szer balra kell léptetni.

(ii) A szorzást a legkisebb helyértéken kell kezdeni.

(iii) Kezdetben $b_{-1} = 0$.

(iv) Mindig két szorzóbit figyelésével, hol össze kell adni, hol ki kell vonni a szorzandót és a részletösszeget, hol pedig nem kell műveletet végezni.

(v) A részletösszeget és a szorzót jobbra kell - aritmetikailag - léptetni.

Az algoritmus regiszterelrendezése a 3.3. ábrán, folyamatábrája a 3.4. ábrán látható.

Az egyes regiszterek tartalma szorzás előtt:

	DR - szorzandó	Q - szorzó
Szorzás után:	DR - szorzandó	ACC,Q - szorzat

[4] Az irodalom - felfedezőjéről - Booth-féle kétbit-figyeléses algoritmusnak is nevezi.

[5] A négybites szóhossz nem megy az általánosság rovására!