8. hét Véges állapotú gépek

8.1. Bevezetés: A véges állapotú gépek definíciója

Korábbi előadásokon már tisztáztuk a logikai hálózat fogalmát.

Logikai hálózatnak nevezzük azokat a rendszereket, melyeknek bemeneti illetve kimeneti jelei logikai jelek, a kimeneti jeleket a bemeneti jelek függvényében többé-kevésbé bonyolult logikai műveletsorozat eredményeként állítják elő.

Ezeknek a logikai hálózatoknak két típusa van:

1. **Kombinációs hálózatok**

Kombinációs hálózatoknak nevezzük azokat a logikai hálózatokat, melyeknek kimeneti jelei csak a bemeneti jelek pillanatnyi értékétől függnek. Ezek a hálózatok éppen ezért „Emlékezet" nélküli hálózatok.

Ez persze azt is jelenti, hogy ezek a hálózatok csak a pillanatnyi bemenetekkel és a pillanatnyi kimenetekkel jellemezhetőek, tehát a rendszer egyrészt csak e kettővel rendelkezik, másrészt nem tudjuk a kimeneti érték ismeretében a pillanatnyi bemenetet meghatározni.

1. ábra A kombinációs hálózat

2. **Sorrendi hálózatok**

Sorrendi (szekvenciális) hálózatoknak nevezzük azokat a logikai hálózatokat, melyek kimeneti jelei nemcsak a pillanatnyi bemeneti jelkombinációtól függnek, hanem attól is, hogy korábban milyen bemeneti jelkombinációk voltak. Ezek a hálózatok tehát emlékezettel rendelkeznek. Ez csak úgy valósítható meg, ha a hálózatban visszacsatolás van.

Ez azt is jelenti, hogy a kimeneti kombinációit csak akkor tudjuk meghatározni, ha ismerjük a

- pillanatnyi bemeneti kombinációt
- a pillanatnyi bemeneti kombinációt megelőző bemeneti kombinációkat

- és ezen bemeneti kombinációk sorrendjét

Ez önmagában azt jelenti, hogy ezekben a rendszerek minden egyes bemeneti kombináció felépítésének hatására elő kell állítania a egy olyan szekunder kombinációt, amely a soron következő bemeneti kombináció mellett a hálózat előéletét, azaz a hálózatra hatott korábbi bemeneti kombinációkat és azok fellépésének sorrendjét. hivatott képviselni. Ezek a szekunder kombinációk jelentik a sorrendi hálózat emlékező elemét.

Így egy adott időpillanatban az aktuális bemeneti kombinációja fellépéskor fennálló szekunder kombinációval együtt létrehozza a kimeneti kombinációt, valamint az új szekunder kombinációt.. Ez utóbbi fogja az aktuális bemeneti kombináció hatását is előéletként képviselni majd a soron következő bemeneti kombináció fellépésekor.

A sorrendi hálózatnak éppen ezért ún. rekurzív módon kell gondoskodnia a szekunder kombinációk módosításáról.

A szekunder kombinációkat a fenti szerepük miatt a sorrendi hálózat állapotainak nevezik, és ezek a logikai változók: állapotjellemzők.

2. ábra A sorrendi hálózat

8.1.1. A sorrendi hálózatok leírási módjai

Alapvetően a sorrendi hálózatokat is többféle módon lehet ábrázolni. Ez az ábrázolás lehetséges:

- függvénnyel,
- különböző modellekkel,
- állapottáblával,
- állapotgráffal.

8.1.2. A sorrendi hálózatot leíró függvény

Mivel a sorrendi hálózatok esetében a működés során adott bemeneti kombináció mellett létrejövő szekunder kombinációk a soron következő bemeneti kombinációval együtt hozzák létre a soron következő kimeneti és szekunder kombinációt a hálózat működését két leképezéssel, függvénnyel tudjuk felírni.

$$f_y = (A, Q) \Rightarrow Y$$

$$f_{Qn+1} = (A, Q) \Rightarrow Q^{n+1}$$

ahol

A: bemeneti kombinációk halmaza

Y: a kimeneti kombinációk halmaza

Q: a bemenetre pillanatnyilag visszajutott szekunder kombinációk, azaz a pillanatnyi állapotok halmaza

Q_{n+1}: A és Q által létrehozott soron következő szekunder kombinációk, azaz a következő állapotok halmaza

f_y: a kimeneti kombinációt előállító leképezése

f_q: a szekunder kombinációt előállító leképezés

Mivel minden kialakuló szekunder kombinációról feltételezzük hogy visszajut a bemenetre, ezért a pillanatnyi és a következő állapotok, azaz szekunder kombinációk halmaza tulajdonképpen ugyanaz a halmaz, melyet állapothalmaznak nevezzük. A q és Q jelölésbeli megkülönböztetésének csupán az a szerepe, hogy a sorrendi hálózat működésének egyes fázisait vagyis az állapotváltozások menetét szemléltesse.

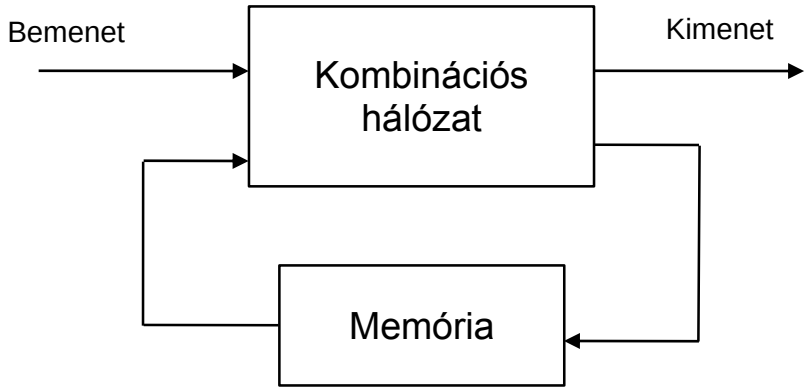
Tehát összefoglalva:

- Sorrendi (szekvenciális) hálózatoknak nevezzük azokat a logikai hálózatokat, melyek kimeneti jelei nemcsak a pillanatnyi bemeneti jelkombinációtól függnek, hanem attól is, hogy korábban milyen bemeneti jelkombinációk voltak
- „Emlékezettel" (memóriával) rendelkező hálózat
- Ugyanazon bemeneti kombinációhoz más-más kimeneti kombináció tartozhat, a szekunder változók aktuális értékétől függően.
- A szekunder változók értékét a korábbi bemeneti kombinációk és azok sorrendje is befolyásolja

Mindezekből az is következik, hogy előző állapotuktól függően különböző módon reagálnak a bemenetükre . Az eképpen működő rendszerek az ún. Véges állapotú automaták (Finite State Machines – FSM) Ezek a rendszerek valamiképpen tehát emlékezniük kell a korábbi bemenetekre ill. a bemenetek sorrendjére (ez tulajdonképpen a belső állapotváltozók alapján fog megvalósuln)i.

Ezért a rendszerben két dolognak meg kell lennie

- visszacsatolás
- memória
-



3. ábra A sorrendi hálózat modellje

A működés folyamata tehát leírható oly módon, hogy tudjuk, a hálózat belső állapotát a szekunder változók értéke határozza meg. A szekunder változók száma viszont megadja a lehetséges állapotok maximális számát. Ugyanakkor fontos azt is leszögezni, hogy nem feltétlenül jön létre minden lehetséges állapot. Bekapcsoláskor a sorrendi hálózat a szekunder változók kezdeti értékeinek megfelelő állapotban van. A bemenő kombinációk változásának hatására a rendszer viszont újabb

állapotba kerülhet, további bemeneti változások hatására pedig még újabb, vagy akár korábbi állapotokba is ugorhat. Azonos bemenő jelre adott esetben más-más szekunder változó és kimeneti kombináció tartozhat. A sorrendi hálózat aktuális állapota pedig megadja a rendszer előéletét.

Egy n hosszúságú bemeneti sorozat (szekvencia hatására) n hosszúságú belső állapot (szekunder változó) szekvencia jön létre, és n hosszúságú kimeneti szekvencia generálódik.

Ha n véges: véges sorrendi automatáról beszélünk. Ezeket véges állapotú gépeknek (FSM: Finale State Machine) is hívjuk.

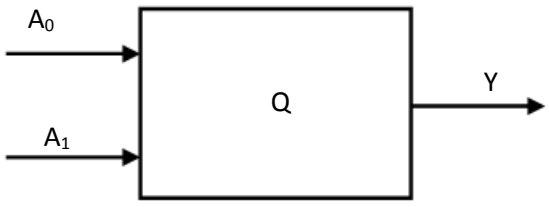
Néhány példát már ismerünk. Ismételjük át őket!

8.1.3. Italautomata

Legyen az általunk vizsgált rendszer egy italautomata, amelyről az alábbi dolgokat tudjuk:

- 150 Ft egy üdítő
- A gép 50 és 100 Ft-os érmét fogad el,
- és visszaad

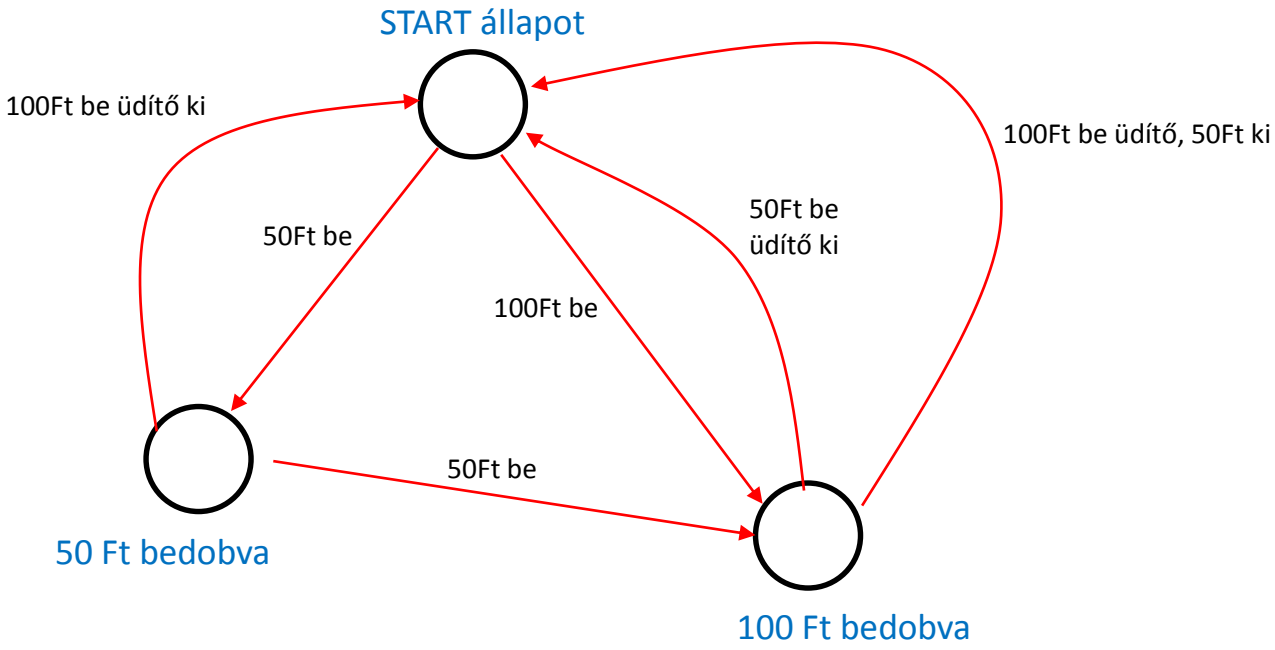
A rendszer felírható a következő módon:



4. ábra Az italautomata elméleti modellje

Belső állapotok száma 3, hiszen vagy kezdeti állapotban vagyunk, vagy bedobtuk valamelyik pénzérmét, és el kell dönteni, hogyan tovább.

Ezt a folyamatot egy gráffal lehet szemléltetni.



5. ábra Az italautomata gráfja

Kódoljuk le a lehetőségeket:

Bemenetek:

A_1: 100Ft;

A_0: 50 Ft

Kimenetek;

Y_1: üdítő ki;

Y_0: 50 Ft ki;

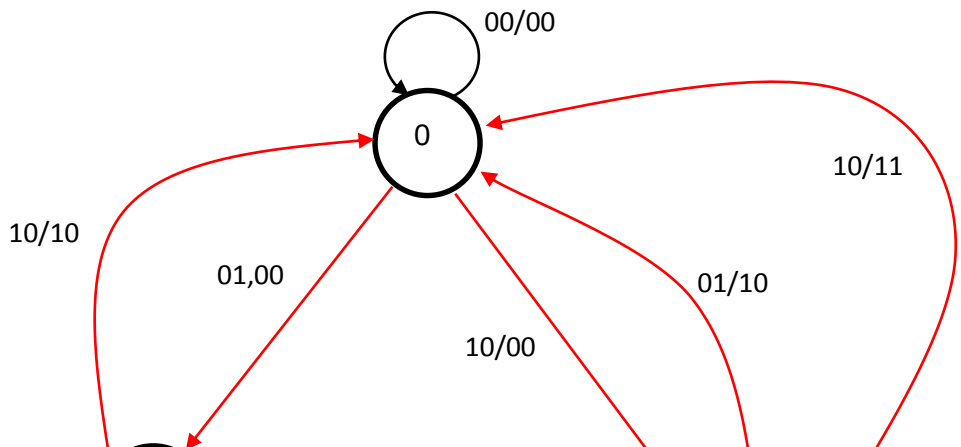
Belső állapotok:

Start állapot: 00,

Bedobtunk 100 Ft-ot : 01,

Bedobtunk 50 Ft-ot: 10.

A kódolás után, a bináris kódokat ráírhatjuk a gráf éleire.



6. ábra Az italautomata gráfja

A rendszer működését táblázatos formában is meg tudjuk adni:

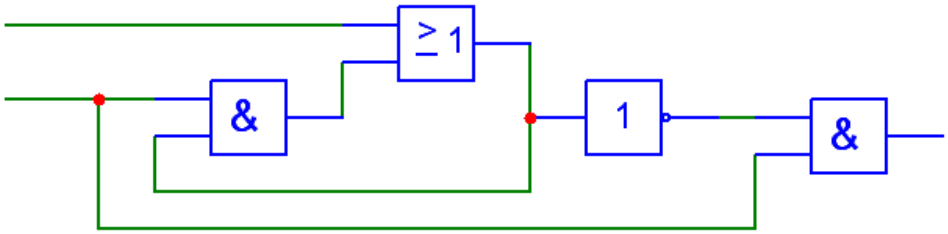
Előző állapot	Bemenet (A_1A_0) 100/50Ft			
	00	**01**	**10**	**11**
0 (0 Ft)	**0**	**2**	**1**	**x**
1 (100 Ft)	**1**	**0**	**0**	**x**
2 (50 Ft)	**2**	**1**	**0**	**x**

7. ábra Az italautomata állapotai

Az 11 állapot jelzi a 150 Ft bedobását, és az automata kiadja az üdítőt.

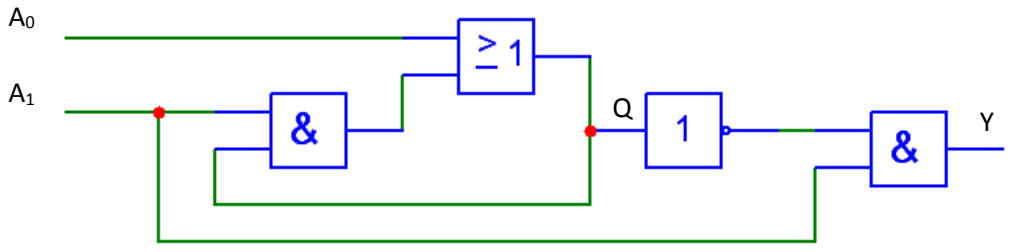
8.1.4. Hálózat

Legyen egy olyan hálózatunk, ahol a hálózatban visszacsatolás van.



8. ábra A példahálózat

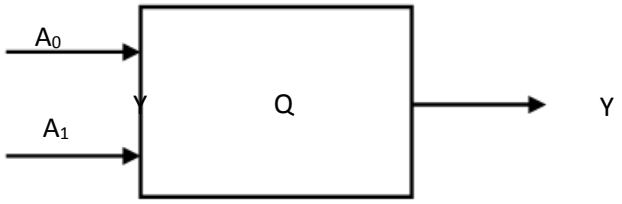
A kimenet ebben az esetben nem csak a bemenetektől függ, hanem a VAGY kapu kimenetén előzőleg észlelt logikai értéktől is, ezért a hálózat egyenletének felírásához szükség van egy közbenső (belső) változóra is.



9. ábra A példahálózat ki - és bemenetei

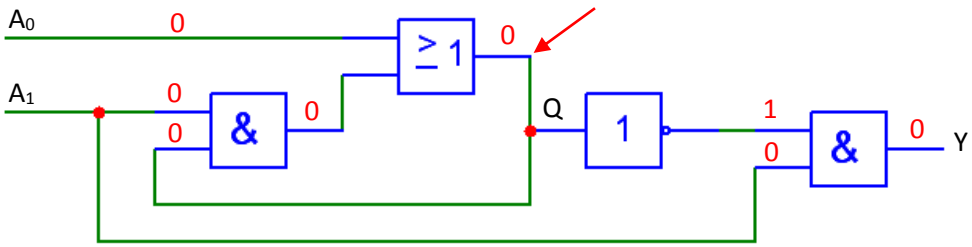
A **Q_{n+1}-gyel jelölve a belső változó aktuális, és Q-val az előző értékét**, logikai függvénykapcsolat írható fel a közbenső változóra és a kimenetre is.

$$Q_{n+1} = A_0 + A_1 Q$$
$$Y = A_1 \cdot \bar{Q}$$



10. ábra A hálózat blokksémája

Tegyük fel, hogy kezdetben

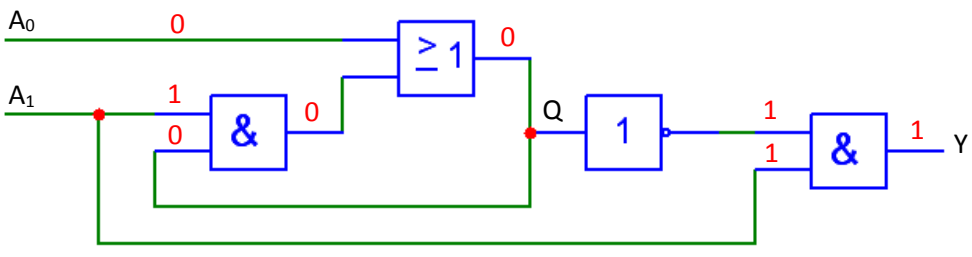


11. ábra Kezdeti állapot

A_0	A_1	Q	Y
0	0	0	0

12. ábra A kezdeti állapot táblázatba foglalva

Tegyük fel, hogy a kezdeti állapotból változás történik: A_1 értéke valamilyen okból 1 lesz.

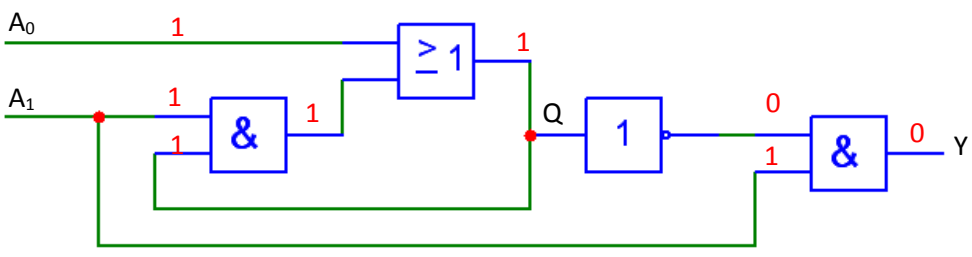


13. ábra Az első lépés

A_0	A_1	Q	Y
0	0	0	0
0	1	0	1

14. ábra Az első lépés táblázatba foglalva

Következő lépésként az A_0 is 1 értéket fog felvenni.

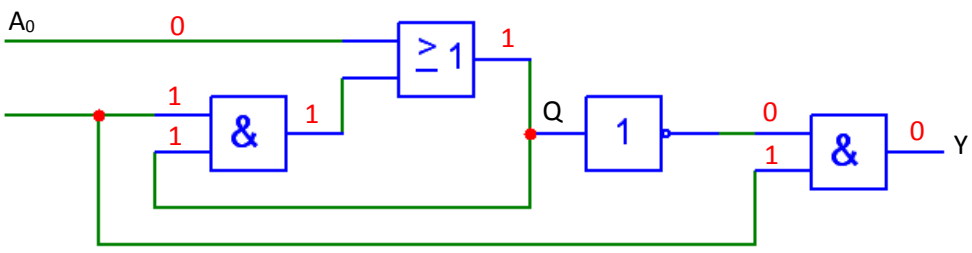


15. ábra A második lépés

A_0	A_1	Q	Y
0	0	0	0
0	1	0	1
1	1	0	0

16. ábra A második lépés táblázata

Harmadik lépésünk legyen megint az, hogy az A_0 értéke legyen nulla.



A_1

17. ábra A harmadik lépés

A_0	A_1	Q	Y
0	0	0	0
0	1	0	1
1	1	0	0
0	1	0	0

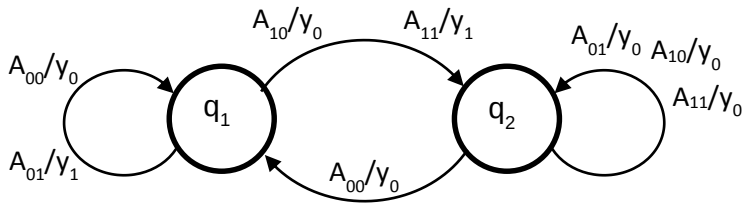
18. ábra A harmadik lépés

Most viszont érdemes visszatekintenünk a harmadik lépés után kapott táblázatra. Jól látható, hogy két alkalommal volt ugyanaz a bemenetünk, de a kimenetek eltérnek egymástól! Ez azért valósulhat meg, mert a hálózat kimenete nem csak a bemenetektől, hanem a hálózat előző Q állapotától is függ. Ez a jelenség mutatja, hogy az adott hálózatban a visszacsatolás mellett, emlékezettel is rendelkezik.

A_0	A_1	Q	Y
0	0	0	0
0	**1**	**0**	**1**
1	1	0	0
0	**1**	**0**	**0**

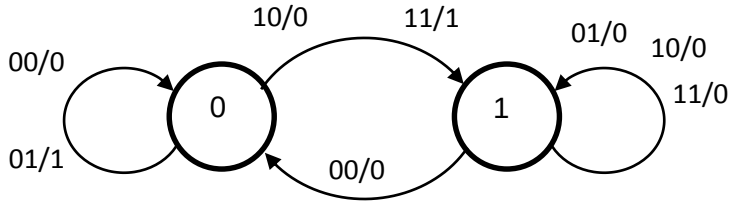
19. ábra Azonos a bemenet, de a kimenet mégis különbözik.

Mivel ennek a hálózatnak két belső állapota van, ezért gráfként két csomóponttal írható fel.



20. ábra A rendszer gráfja

Ha a megfelelő értékeket behelyettesítjük:



21. ábra A rendszer gráfja

Az állapottábla táblázatos formában adja meg, hogy adott bemeneti kombinációk hatására mely állapotból mely állapotba ugrik a rendszer. A kimenet alakulását is ebben a táblázatban írhatjuk fel. A kombinációs hálózatoknál használt igazságtáblázathoz hasonló szerkezetű táblázatról van szó.

Q	A_0	A_1	Q_{n+1}	Y
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

22. ábra A hálózat állapottáblája

8.2. A 7 szegmenses kijelző

A 7 szegmenses kijelző tulajdonképpen egy alap építőelem, egy jel- dekódoló. Mit is jelenet ez?

Fontos kérdés, hogy milyen építőelemekből építhetjük meg a valós rendszereket. Mi az, ami számunkra rendelkezésre áll. Ezen építőkövek ismerete azért is fontos, mert ezek segítségével nagyobb és bonyolultabb rendszereket is fel tudunk építeni.

Ezek az építőelemek már nem tekinthetők „alap építőkövek”-nek, hiszen *a korábban megismert logikai kapuk megfelelő kombinációjából épülnek fel.*

Az építőelemek típusai a következők:

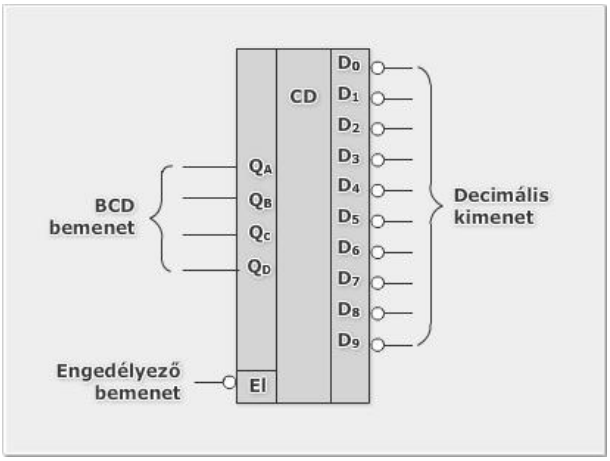
- Kódolók, Dekódolók
- Adatút-választók
- Aritmetikai egységek

8.2.1. Kódolók

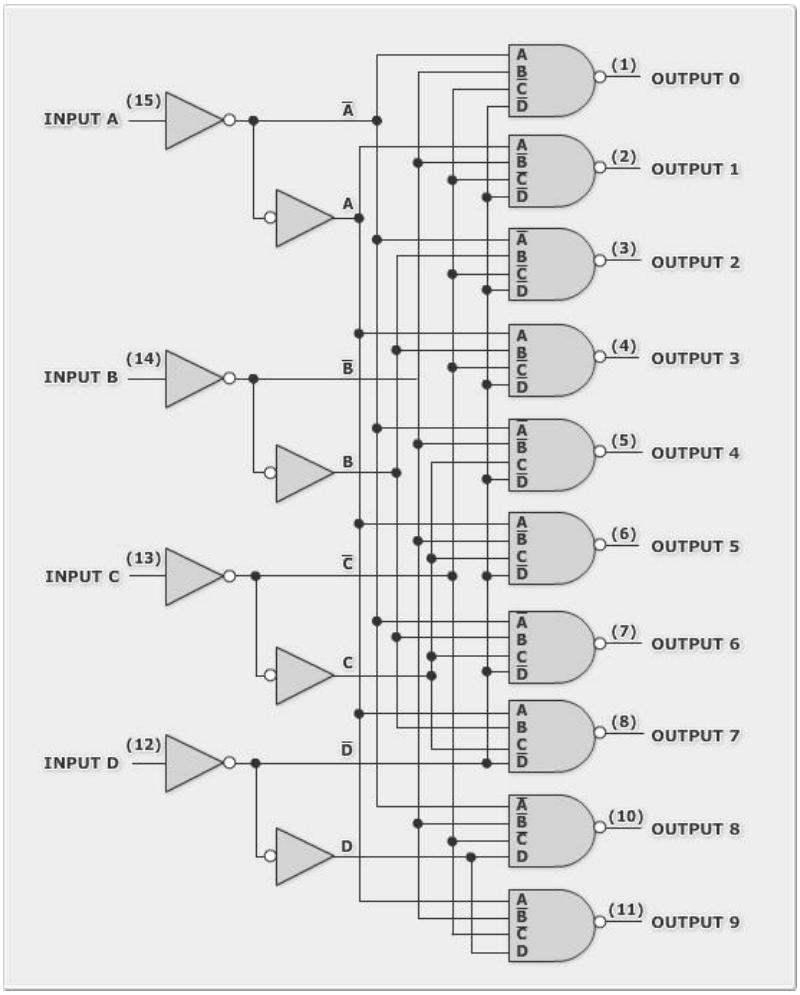
Kódolóknak, dekódolóknak vagy kódátalakítóknak azokat a kapuáramkörökből felépített logikai hálózatokat nevezzük, amelyek az információt az egyik kódrendszerből a másik kódrendszerbe alakítják át.

8.2.1.1. Kódolók

Ilyen például a BCD decimális kódoló-dekódoló áramkör. Mint tudjuk, a bináris rendszerben 4 helyiértékkel 16-ig lehet a számokat leírni. A BCD decimális dekódolóban nem használjuk ki mind a 16 lehetőséget (ezt a hexadecimális rendszer teszi meg), mert csak 10 decimális számjegy ábrázolására van szükség. Az áramkör úgy működik, hogy a megfelelő BCD-kódra az áramkör kimeneti oldalán a megfelelő decimális kimeneten logikai 1 (igen) jelenik meg, míg a többi decimális szám kimenete zérus (logikai nem) lesz. Az áramkör része szokott lenni az engedélyező bemenet, ami azt jelenti, hogy az áramkör csak akkor lesz aktív, ha erre a bemenetre logikai igent kapcsolunk. Az ábrán látható engedélyező bemenetnél megjelenő kis körszimbólum egy jelinvertálást jelent, ezért az engedélyezés csak akkor történik meg, ha erre a bemenetre logikai nem értéket kapcsolunk.

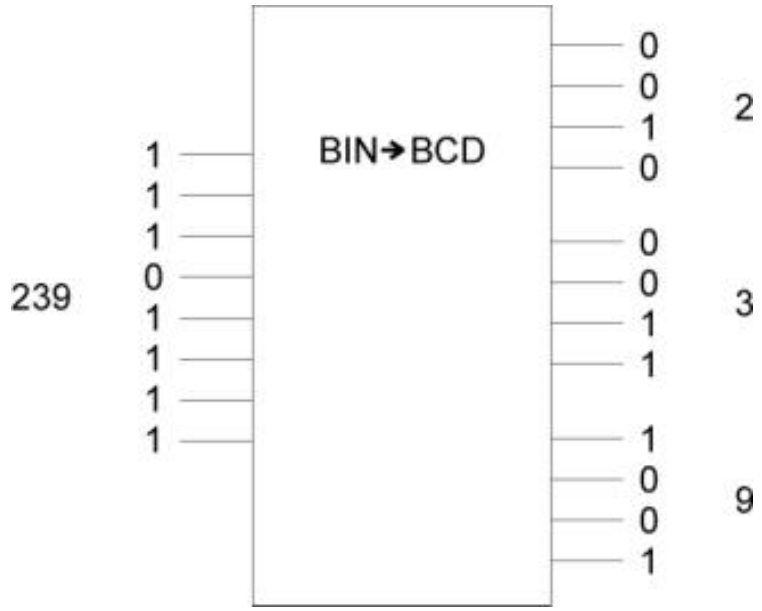


23. ábra BCD - Decimális konverter



24. ábra BCD - Decimális konverter

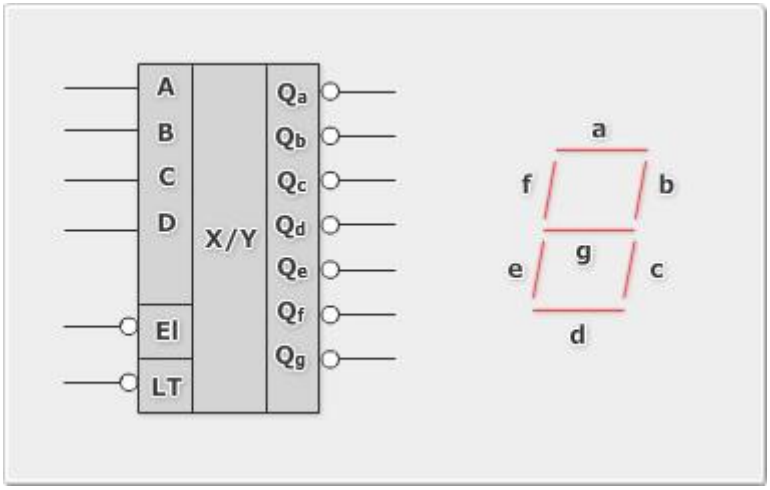
Jó példa ugyanerre az átalakításra a Bináris – BCD átalakító, melynek fontos szerepe van a binaris számok megjelenítésében például. Segítségével lehet a 239-es szám binaris alakját 3db 4 bites tárolású számmá alakítani.



25. ábra Bináris - BCD konverter

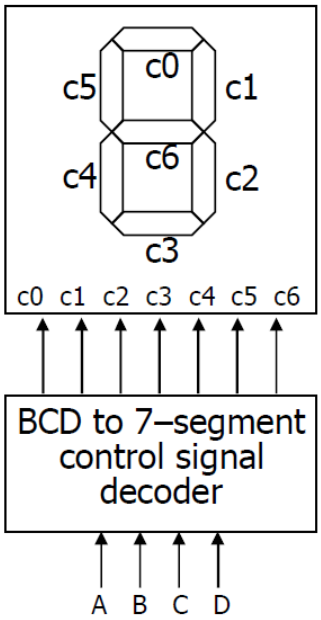
8.2.1.2. Dekódolók

Következő példa a kódoló-dekódoló áramkörökre a BCD hétszegmenses kijelzőt meghajtó áramkör. A hétszegmenses kijelzőket a digitális technikában sok helyen használjuk, elsősorban a decimális számok kijelzésére.



26. ábra 7 szegmenses kijelző - dekódoló

A kódátalakító sémáját az ábra mutatja. A 4 bemenethez itt 7 kimenet tartozik. A kimenetekre akkor jut logikai igen, amit a kimeneteknél ábrázolt a kis körszimbólum, ami itt is invertálás műveletet jelent logikai nem értékre változtat, ha az aktuális szegmensnek világítania kell. A hétszegmenses kijelző állapotait az ábra mutatja.



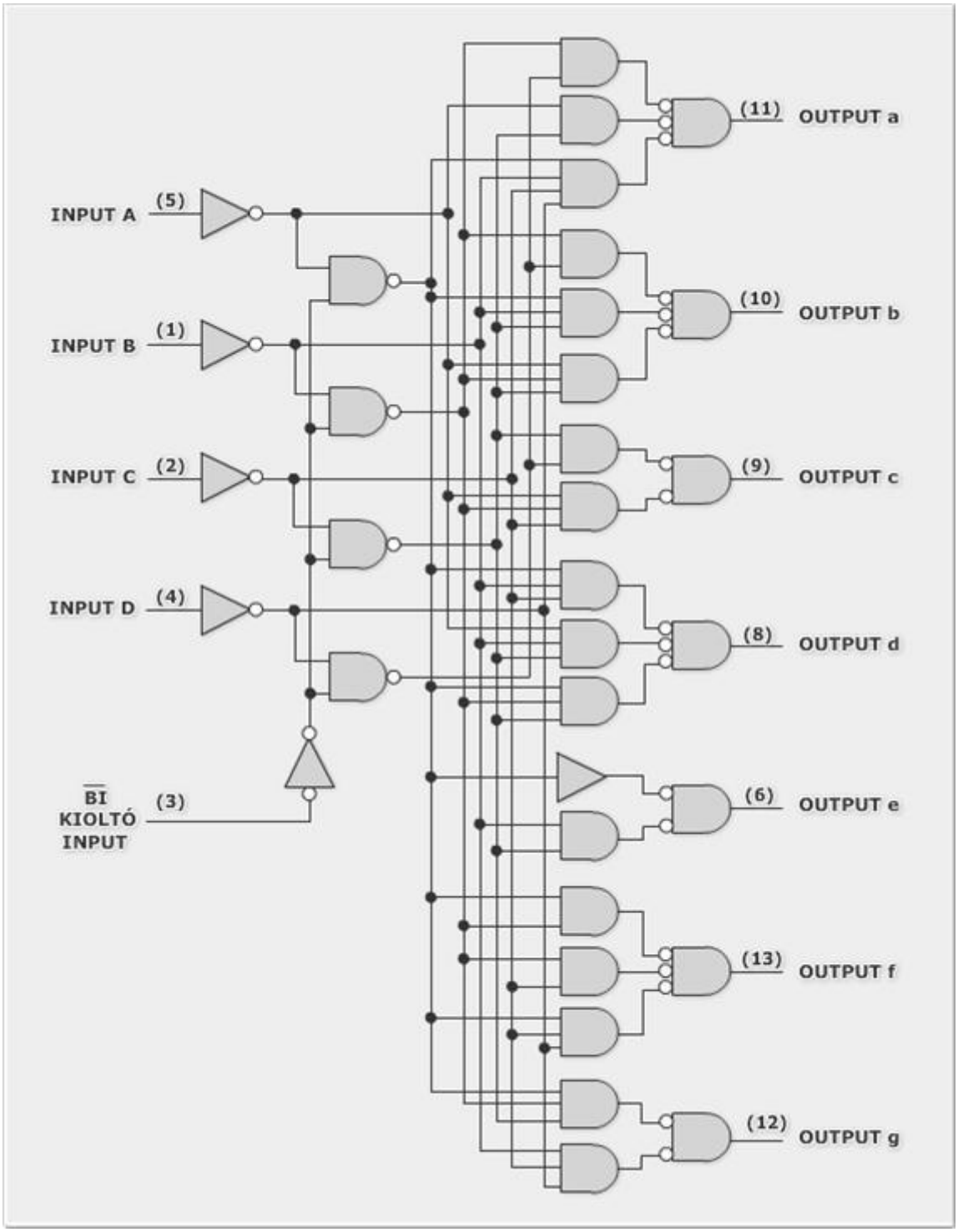
27. ábra 7 szegmenses kijelző

Az igazságtáblázatot az láthatjuk. Például a 0 kijelzésénél a g szegmensen kívül mindegyik szegmensnek világítania kell. Megjegyezzük, hogy az X jelölés azt jelenti, hogy ekkor a logikai változó értéke közömbös, nincs befolyása a kimenetre.

Funkcionális táblázat (T3)													
decimális szám, vagy funkció						kimenet							
	D	C	B	A	$\overline{BI}$	a	b	c	d	e	f	g
0	L	L	L	L	H	H	H	H	H	H	H	L
1	L	L	L	H	H	L	H	H	L	L	L	L
2	L	L	H	L	H	H	H	L	H	H	L	H
3	L	L	H	H	H	H	H	H	H	L	L	H
4	L	H	L	L	H	L	H	H	L	L	H	H
5	L	H	L	H	H	H	L	H	H	L	H	H
6	L	H	H	L	H	L	L	H	H	H	H	H
7	L	H	H	H	H	H	H	H	L	L	L	L
8	H	L	L	L	H	H	H	H	H	H	H	H
9	H	L	L	H	H	H	H	H	L	L	H	H
10	H	L	H	L	H	L	L	L	H	H	L	H
11	H	L	H	H	H	L	L	H	H	L	L	H
12	H	H	L	L	H	L	H	L	L	L	H	H
13	H	H	L	H	H	H	L	L	H	L	H	H
14	H	H	H	L	H	L	L	L	H	H	H	H
15	H	H	H	H	H	L	L	L	L	L	L	L
BI	X	X	X	X	X	L	L	L	L	L	L	L

1.2.1.6. ábra Forrás: Texas Instruments

Az, hogy ez az áramkör is kombinációs hálózat, a következő ábrából jól látható, az áramkör kapukból épül fel, és memóriaelemet nem tartalmaz.

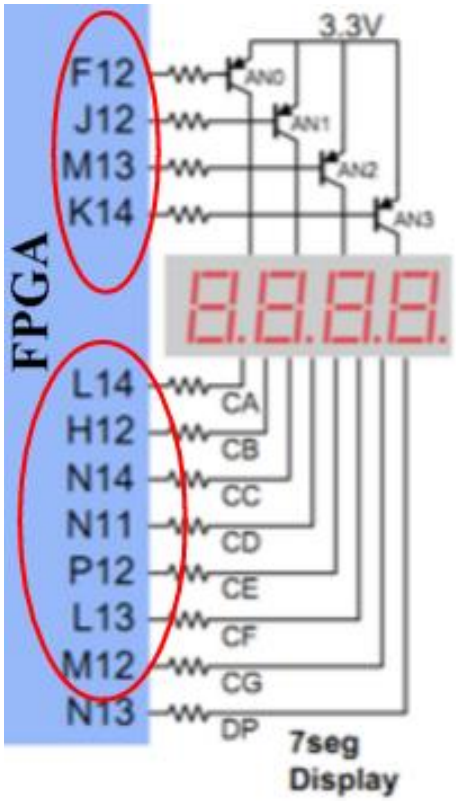


Forrás: Texas Instruments

28. ábra A 7 segmenses kijelző kapcsolási rajza

8.2.2. A 7 szegmenses kijelző működése

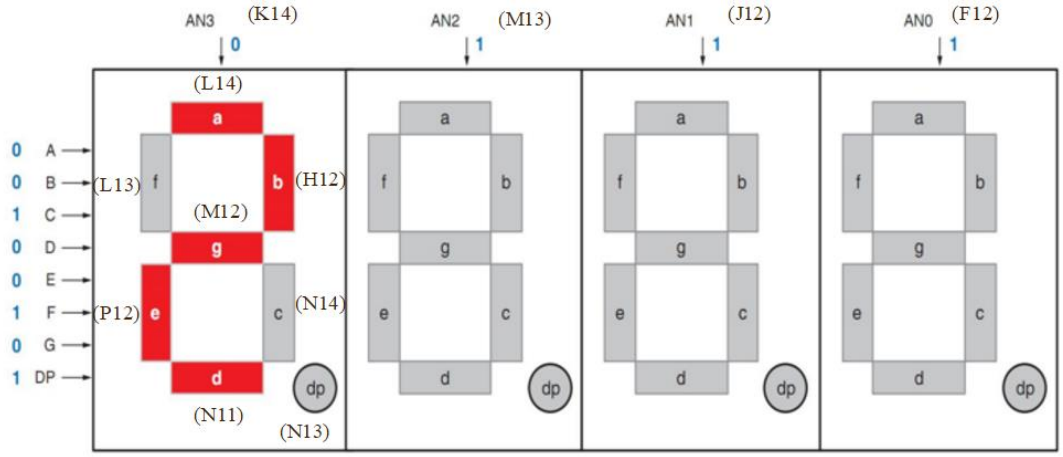
A kijelzés csak akkor működik, ha a „blanking” bemenetre logikai igen értékű jelet kapcsolunk.



29. ábra A 7 segmenses kijelző kapcsolási rajza

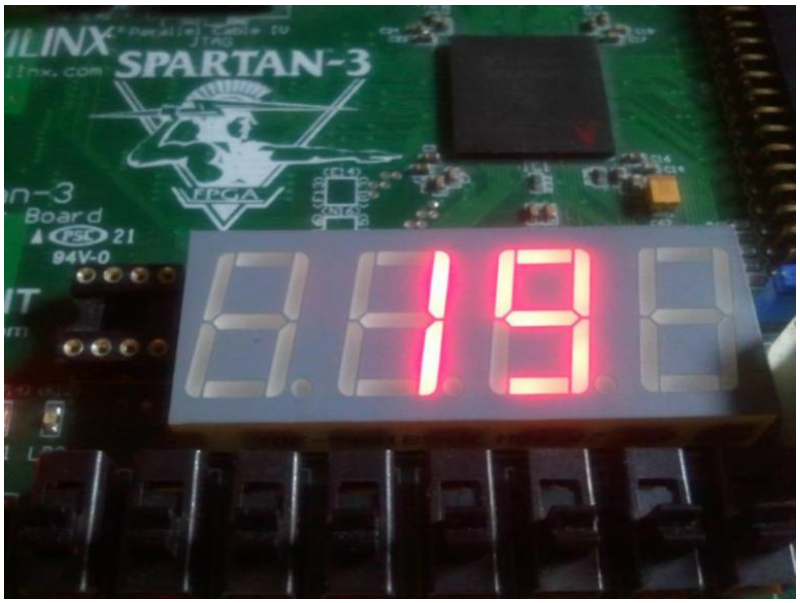
A 7 szegmens vezérlése úgy történik, a panelen levő négy kijelző vaamennyi szegmensét egyszerre vezéreljük , azaz mindre ugyanaz a jel jut ugyanabban az időpillanatban. Ahhoz, hogy a megfelelő kijelzőn lássuk, amit szeretnénkegy engedélyező jelre van szükség (AN). Ennek mindig logikai o értéke jelenti az aktív állapotot.

A 7 szegmenst a megkülönböztethetőség kedvéért betűkkel jelöljük: a, b, c,d, ef, g és a pont dp. A kigyújtott szegmens szintén logikai nullát kap.

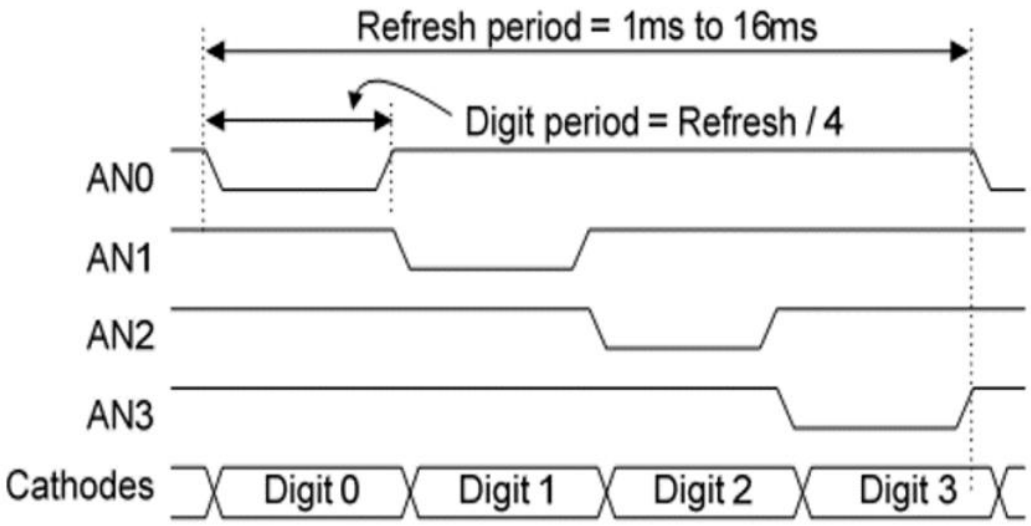


30. ábra A 7 segmenses kijelző szegmensei

Ahhoz, hogy minden kijelzőn más értéket tudjunk megjeleníteni, fontos, hogy a kijelzők között a rendelkezésre álló időt megosztjuk, szeletekre szeparáljuk. Az időszelet hosszát úg kell megválasztani, hogy az emberi szem folyamatosnak lássa a működését (25/s).Ezért a vezérlő órajelre 1 ms- ként van szükségünk.

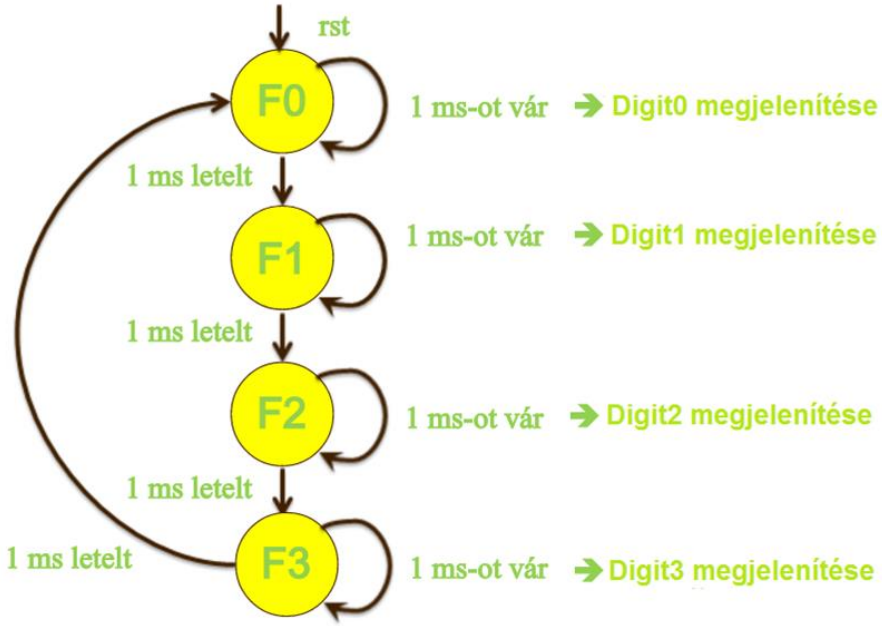


31. ábra A 7 segmenses kijelző szegmensei



32. ábra A 7 segmenses kijelző időszeletei

Az egyes digitek kiválasztása és a hozzájuk tartozó érték megjelenítése a rendszer egy-egy állapota, ezért ez a rendszer is tekinthető FSM-nek.



33. ábra A 7 segmenses kijelző mint FSM

Hogyan mérjük meg az 1 ms-ot?

A rendszerórajel (szinkronjel) 50 MHz frekvenciájú, ez pontosan 20 ns!

Kezdjük el számolni a rendszerórajel periódusokat, minden 50 000. pontosan 1 ms távolságra van egymástól (20ns*50 000 = 1ms).

1 s megmérése (minden 50 000 000. órajel periódus 1s távolságra van).

34. ábra A vezérlő órajel előállítása

8.3. Microprocesszorok : A CPU

8.3.1. A CPU definíciója

A CPU (angol: Central Processing Unit – központi feldolgozóegység) más néven processzor ill. mikroprocesszor, a számítógép „agya”, azon egysége, amely az utasítások értelmezését és végrehajtását vezérli, félvezetős kivitelezésű, összetett elektronikus áramkör. Magyarra többféleképpen fordítják, így pl. a központi végrehajtó egység, központi feldolgozó egység, központi feldolgozó processzor, vagy egyszerűen processzor kifejezések is elterjedtek. Míg a processzor fogalma már korábban ismert volt, a mikroprocesszor megjelenését csak a félvezetős technológia és az integrált áramkörök kifejlesztése tette lehetővé, az 1970-es évek elején. A processzor alatt általában mikroprocesszort értünk, régebben a processzor sok különálló áramkör volt, ám a mikroprocesszorral sikerült a legfontosabb komponenseket egyetlen szilíciumlapkára integrálni.

A mikroprocesszor egy vagy több szilícium kristályra integrált, néhány ezertől több tízmillió számú tranzisztort tartalmazó integrált áramkör, amelyben további részegységek különíthetők el, rendelkezik az adatok ki- és beviteléhez szükséges sínrendszerrel, és rendelkezik egy utasításkészlettel, amelynek utasításait képes végrehajtani. A bemeneti eszközök segítségével kódolt információkat feldolgozza, majd az eredményt a kimeneti eszközök felé továbbítja, melyek ezeket az adatokat információvá alakítják vissza. A világ első mikroprocesszorát (TMS 1000) az amerikai Texas Instruments fejlesztette ki 1971-ben, ezt követte az Intel 4004-es processzora, majd további cégek gyártmányai.

8.3.2. A CPU története

Az első mikroprocesszor az 1971-ben megjelent 4 bites szóhosszúságú Intel 4004 volt. Később több sikeres 8 bites sorozat jelent meg több gyártó részéről (Intel 8008, 8080, 8085, Zilog Z80, Motorola 6800, MOS Technology 6502). A 80-as évektől kezdve megnőtt a processzorok szóhossza (Intel 8086 (az IBM PC és PC/XT processzora): 16 bit (20-bites címtartomány), Intel 80286 (a PC/AT processzora): 16 bit (24 bites címtartomány) – 1982, Intel 80386: 32 bit – 1985) az órajel folyamatos növekedése mellett.

Az első mikroprocesszor a Texas Instruments által 1971. szeptember 17-én bemutatott, a TMS 1000 sorozatba tartozó TMS1802NC jelű processzor volt. Röviddel ezután, 1971. november 15-én jelent meg az Intel cég Intel 4004 jelű processzora. Mindkettő 4 bites szóhosszúságú eszköz volt, azonban a Texas csipje maszkprogramozott memóriát (beépített alkalmazást) és ki-/bemeneti csatornákat is tartalmazott. Később több sikeres 8 bites sorozat jelent meg több gyártó részéről (Intel 8008, 8080, 8085, Zilog Z80, Motorola 6800, MOS Technology 6502).

A 8 bites mikroprocesszorok már az 1970-es évek elején megjelentek. A további fejlődés is a szóhossz növelésének irányába mutatott: 1973-ban megjelent az első 16 bites processzor (National Semiconductor IMP-16), de az 1970-es években a mikroprocesszorok túlnyomó többségének szóhossza 8 bites maradt. Az 1970-es évek vége felé jelentek meg nagyobb számban a 16 bites szóhosszúságú processzorok: 1978: Intel 8086, 1979: Motorola 68000. A 80-as évektől kezdve tovább nőtt a szóhossz: 1980-ban megjelent az AT&T Bell Labs BELLMAC-32A processzora, ez volt az első 32 bites processzor, ezután egyre több 32 bites eszköz jelent meg, így a Motorola sikeres m68k processzorcsaládja, a DEC, Sun és Intel is megjelentet 32 bites processzorokat. Az 1980-as évek közepétől kezdenek megjelenni az eleinte szintén 32 bites RISC elvű processzorok, pl. a Sun (SPARC), MIPS, Hewlett-Packard (PA-RISC), AMD (29k) és ARM architektúrák, de az Intel és a Motorola is gyártott RISC-alapú processzorokat. A szóhossz növekedése mellett megfigyelhető a komplexitás növekedése: a többcsipes eszközöket egyetlen áramkörbe integrált eszközök váltják fel, és a processzorok egyre több tranzisztort tartalmaznak, így az 1980-as évek közepén már elérték a (több-)százezres, az évtized végére pedig már a milliós nagyságrendet is. Ezzel együtt a processzorok órajele is folyamatosan nőtt.

Az x86-64 a x86-os architektúra 64 bites leszármazottja. Az x86-64 utasításkészlet támogatja Intel x86-os architektúráját, ezt az Advanced Micro Devices (AMD) tervezte, majd átnevezte AMD64-re. Ezt az architektúrát az Intel lemásolta és Intel 64-nek nevezte el (régebben Yamhill, Clackamas Technológia, CT, IA-32e és EM64T neveken volt ismert)[1]. Ez vezetett a hétköznapi nyelvben az x86-64 vagy x64 elnevezések használatához, mint gyártó-független fogalmakhoz, amikor a két közel azonos kivitelezésű architektúrára hivatkozunk.

A processzorok kronológiája

60-as, 70-es évek

1964. április 7. – az IBM bejelenti a System/360 számítógéprendszer-családot, ezen belül a Model 30, 40, 50, 60, 62, és 70 jelű gépeket. Jellemzőik: 2-es komplemens bináris aritmetika, 8 bites bájt, 32 bites gépi szó, EBCDIC karakterkészlet, lebegőpontos számformátum, 16-32 db. 32 bites regiszter, min. 4 kB memória. A CPU mikrokódolt, tranzisztoros-nyomtatott áramkörös különálló egység.

1967 – A Texas Instruments feltalálja a kézi elektronikus kalkulátort.

1970. június – Central Air Data Computer MP944 csipkészlet: 20 bites, többcsipes integrált katonai repülésvezérlő rendszer, amelyet korai vadászrepülőgépekben használtak, esélyes a világ első CPU-ja címre, de ennek a megítélése nem egyértelmű.

1970. június 30. – Megjelenik az IBM System/370 számítógépcsalád: 128 bites lebegőpontos aritmetika, 32 bites címzés. A CPU mikrocsipekből áll, több nyomtatott áramköri panelen.

1971 – A Texas Instruments feltalálja az egycsipes mikroszámítógépet.

1971. szeptember – megjelenik a Texas Instruments TMS 1000: 4 bites egycsipes mikrokontroller, a világ első egycsipes számítógépe,[3] ezeket a csipeket kalkulátorokban alkalmazták.

1971. november 15. – megjelenik a 4004-es, az Intel első 4 bites processzora.[4]

1972 – Elkészül az Intel 4040 (az Intel 4004 utódja) – 4 bites processzor, BCD utasításokkal.

1972. április 1. – megjelenik az Intel 8008, világ első 8 bites mikroprocesszora. Tranzisztorszáma: 3500.[5]

1972 – Rockwell PPS-4: 4 bites PMOS technológiájú CPU, az Intel 4004 konkurense.[6]

1973 eleje – Az első többcsipes 16 bites mikroprocesszor megjelenése: a National Semiconductor IMP-16 jelű terméke.[7] (5 csip, 4 bites regiszterek és ALU, CROM)

1973. szeptember 4. – A Texas Instruments a világon elsőként, szabadalmat kap az egycsipes mikroprocesszorra, Gary Boone találmányára.

1974. április 1. – Megjelenik az Intel 8080, az első valóban használható 8 bites CPU. Tranzisztorszáma: 4500.[8]

1974 közepe – A Motorola 8 bites CPU-ja, az MC6800. Tranzisztorainak száma 4100.

1975. január – A National Semiconductor bemutatja első egycsipes 16 bites mikroprocesszorát, a PACE-t,[9] – ez az első ilyen típusú mikroprocesszor, ami üzleti forgalomba került. Ezt később ennek NMOS változata követte, a INS8900.

1975 – Az IBM System/4 Pi számítógépcsalád megjelenése – ez a sorozat a System/360 folytatása, hibatűrő, sugárzásvédett, többprocesszoros, 32- vagy 16 bites rendszerek, harci repülőgépeken, bombázókban és az űrsiklókban szerepeltek.

1975 – Az AMD elindítja Am2900-as csipsorozatát, ezek első példánya az Am2901: 4 bites bitszelet-technikát használó ALU

1975 – A Fairchild Semiconductor bemutatja F8 jelű 8 bites CPU-ját, melyet játékgépekben, szintetizátorokban használnak majd

1975. szeptember – a MOS Technology bemutatja 6502 számú 8 bites CPU-ját, amely az Apple I és Apple II processzora. Az eszköz 3510 tranzisztort tartalmaz.

1976 júniusa – Megjelenik a Texas Instruments TMS9900 – az egyik első valódi 16 bites mikroprocesszor.[10]

1976 – RCA 1802, avagy CDP 1802 – az RCA által fejlesztett 8 bites CMOS mikroprocesszor, a Voyager, Viking, Galileo űrszondák processzora.

1976. július – Megjelenik a Zilog Z80 processzor, a világ egyik legelterjedtebb 8 bites processzora.

1977 – Az Intel kibocsátja a 8085 8 bites mikroprocesszorát (ebbe a családba tartozik a Sojourner marsjáró 80C85 processzora is).

1977 – Az AT&T Bell Laboratories bemutatja a BELLMAC-8 mikroprocesszort: ez egy 8 bites, 16 bites címzéssel rendelkező processzor, 5 mikronos CMOS technológiával készült.[11]

1978. június 8. – A 16 bites Intel 8086 megjelenése

1979 – Megjelenik a Motorola 68000, az első 16/32 bites CISC processzor, az Amiga, Apple, Atari és Macintosh gépek processzora.

1979 – Zilog Z8000: 16 bites processzor, nem Z80-kompatibilis, 8-, 16- és 64 bites regisztereket használhat.

1979. június 1. – Az Intel 8088 16 bites processzor, az első IBM PC-k processzora.

80-as évek

1980 – A MOS Technology befejezi a 6510 CPU fejlesztését – ez a Commodore 64 számítógépek CPU-ja.

1980 nyara – elkészül az IBM 801 processzor prototípusa: az első RISC processzor.

1980 – AT&T Bell Labs BELLMAC-32A – az első egycsipes, teljesen 32 bites CPU.

1980 – David Patterson a kaliforniai Berkeley Egyetemen elindítja a RISC projektet, amely a RISC I és RISC II processzorokhoz vezetett 1981-ben. Tőle ered a RISC kifejezés.

1980 – Az Intel bejelenti a 8087-es lebegőpontos koprocesszort. Ez a 8086, 8088, 80186 és 80188 processzorokkal működik együtt, teljesítménye kb. 50 000 FLOPS.

1981. január 1. – Az Intel iAPX 432 bemutatása: az Intel első 32 bites processzora.

1981 – Elkészül az IBM ROMP (Research (Office Products Division) Micro Processor) processzora: ez egy 10 MHz órajelű 32 bites RISC processzor, később az IBM RT/PC gépeiben és lézerprinterekben használták.

1982 – Az Intel 80186 bemutatása

1982 – Motorola 68008: a Motorola 8/16/32 bites mikroprocesszora, a Motorola 68000 egy változata, 8 bites külső adatbusszal. A Sinclair QL személyi számítógépben volt ilyen.

1982. február 1. – Az Intel 80286 bemutatása

1982. február – Az AMD licencszerződést köt az Intellel 8086 és 8088 processzorok gyártására.

1982 – Az AMD Am286 processzorokat is gyárt, az Intel licencszerződés keretében.

1983 – Az Acorn Computers Ltd. megkezdi az ARM architektúra tervezését. A cél egy egyszerű 32 bites RISC processzor kifejlesztése.

1984 – Elkészül a Western Design Center (WDC) 16 bites mikroprocesszora, a WDC 65816 ill. WDC 65802 – teljesen kompatibilis a MOS Technology 6502-vel, a 65802 még tokozásban is megegyezik vele. Ezek a processzorok az Apple IIGS gépekbe kerültek. A SNES játékkonzol Ricoh 5A22 CPU-ja egy módosított WDC 65C816.

1984 – Az INMOS angol mikroelektronikai cég megjelenteti az első transputereket. Ezek párhuzamos működésre szánt mikrokontroller-szerű processzorok, 16, 32 és 64 bites változatok készültek belőlük.

1985 – A Hitachi által tervezett 68HC000 bemutatása – ez a Motorola 68000 CMOS változata.

1985 – DEC MicroVAX 78032 – VAX utasításkészletű processzor, amit a DEC VAX gépeiben használtak.

1985 – SUN SPARC (Scalable Processor Architecture) – a Sun Microsystems által tervezett RISC jellegű processzor-architektúra, a Sun workstationokban való használatra.

1985 – R2000, a MIPS Computer Systems MIPS (Microprocessor without Interlocked Pipeline Stages) architektúrájú 32 bites RISC mikroprocesszora

1985. április 26. – Elkészül az ARM1, az Acorn első működő ARM processzora.

1985. október 17. – Megjelenik az i386DX, az x86-os processzorcsalád első 32 bites tagja.

1986 – az Intel felmondja a licencszerződést az AMD-vel, és elutasítja az i386 architektúra átadását. Ld. 1991.

1986 – Elkészül az Acorn ARM2, a világ talán legegyszerűbb, használható 32 bites mikroprocesszora.

1986 – Z80000 a Zilog 32 bites processzora.

1986 – A Hewlett-Packard elkészíti az első PA-RISC architektúrájú processzorát, a 6 csipből álló TS1-et.

1987 – Az AMD kifejleszti a Sonyval a CMOS technológiát

1987 – A HP elkészíti PA-RISC processzorainak CMOS verzóit, a 8 MHz-es CS1-et és az NMOS technológiájú, gyorsabb NS1-et.

1987 – Az AT&T CRISP (C-language Reduced Instruction Set Processor) processzora.

1988 – MIPS R3000

1988. április 5. – Az Intel bemutatja az i960 (alias 80960) architektúrát, ami egy Berkeley RISC-en alapuló 32 bites processzor-architektúra.

1988. április – A Motorola 88000 (röviden m88k) család: a Motorola saját fejlesztésű Harvard architektúrájú RISC processzor-tervezete. Első elkészült tagja a 88100; 32 bites RISC processzor, órajele 33MHz.[12]

1988 – Elkészül az AMD 29000, másképp 29K, az AMD első RISC alapú 32 bites mikroprocesszora, a 29000-es sorozat első tagja. A Berkeley RISC designra épül, hasonlóan a SUN SPARC és az Intel i960 architektúrához.

1989 – PA-7000, a Hewlett-Packard PA-RISC architektúráján alapuló 32 bites processzor.

1989. február 27. – Intel i860 avagy 80860 – az Intel első szuperskalár processzora, RISC 32/64 architektúra.

1989. július – Intel i960CA az i960 architektúra első tiszta RISC megvalósítása, egycsipes szuperskalár RISC implementáció.

1989 – Elkészül az Acorn ARM3 (a csipben 4k adat- és utasítás-gyorsítótár is van),[13] valamint az ARM2as, az ARM statikus verziója.

1989 – Az Intel 80486 bemutatása

90-es évek

1990 – Az IBM POWER1 architektúra bemutatása – az IBM RS/6000 gépek processzora, 32 bites, többcsipes, kétutas szuperskalár RISC CPU, a modellek órajele (20-)25-62.5 MHz között lehet.

1990. november – Megalakul az ARM Ltd., az ARM processzorok fejlesztése elválik az Acorn Computers Ltd.-től.

1991 – Elkészül az ARM6, az Apple és ARM Ltd. közös fejlesztése (ARM4 és ARM5 nem készült, a számozást 6-tól folytatták).[14]

1991. március – Az AMD bemutatja az Am386 microprocesszor családot, ami az AMD saját i386 klónja[15]; ezzel megtöri az Intel monopóliumát.

1991 – MIPS R4000 – a MIPS első 64 bites mikroprocesszora.

1991 – Az Apple, IBM és Motorola – az AIM szövetség – elkezdik a PowerPC 600-as processzorcsalád tervezését.

1992. február 25. – A DEC Alpha 21064 processzorának bejelentése - 64 bites RISC architektúra. Ez volt a legnagyobb teljesítményű processzor 1993-ig, az IBM POWER2 megjelenéséig.

1992. október – Elkészülnek az első PowerPC 601 processzor prototípusok.

1992. október 12. – Az AT&T bejelenti az ATT 92010 – más néven Hobbit – processzort.[16] Ez egy 32 bites, 20–30MHz órajelű processzor volt, 3 kB-os utasítás-előolvasó tárral (instruction prefetch buffer) és 32 elemű utasítás-gyorsítótárral rendelkezett, regiszterei nem voltak, állítólag 37 VAX MIPS tejlesítményre volt képes. Az AT&T CRISP architektúra megvalósítása, nyelvspecifikus, C nyelvű programok futtatására volt optimalizálva.[17]

1992 – A Hitachi megkezdi SuperH RISC processzorcsaládjának fejlesztését, aminek az első tagjai az SH1 és SH2 processzorok, melyeket beágyazott rendszerekbe szánnak (1994,[18]).

1993. március 22. – Az Intel Pentium bemutatása: 32 bites processzor.

1993. április – Az AMD Am486 mikroprocesszor-család megjelenése.

1993 – Az IBM POWER2 (eredetileg RIOS2 nevű) processzorok kibocsátása: 55 - 71.5 MHz, javított POWER1 felépítés, 8 csipes.

1993. ősz – Az IBM-nél elkészül a PowerPC 601 (IBM PPC601, a Motorolánál MPC601 néven), a 32 bites alap PowerPC utasításkészletű processzorok első generációja. 50 - 80 MHz közötti órajelet használ, RS/6000 workstation ('93) és Power Macintosh gépekben ('94) alkalmazzák.

1994 – Az AT&T 92020 Hobbit processzora, a 92010 továbbfejlesztése, 6 kB-os utasítástárral. Az AT&T korai PDA-jában, az EO Communicator-ban működött.

1994 – MIPS R8000 – a MIPS első szuperskalár mikroprocesszora.

1994 – ARM Ltd., ARM7 processzor (ARMv3, ARMv4T, 0-60 MHz, 8KB cache lehet)[19]

1994 – A PowerPC 603 és PowerPC 604 megjelenése: a 603 az első teljes 32 bites PowerPC architektúra implementáció; a 604: szuperskalár, 6 fokozatú futószalag, órajele 100 - 180 MHz.

1994 – Az IBM kísérleti PowerPC 615 processzora: 32/64 bites PowerPC és 32 bites x86 utasításokat is képes volt végrehajtani, akár vegyesen is.

1995 – A DEC licencelt ARM6 technológián alapuló StrongARM projektjének első eredménye az SA-110 ARMv4 architektúrájú processzor. Fogyasztása 233 MHz órajelnél 1 watt.

1995 – A NEC VR4300 egy MIPS R4300i-en alapuló 64 bites RISC mikroprocesszor, MIPS I, MIPS II, MIPS III utasításkészlettel, a Nintendo 64 játékkonzol processzora.

1995 – PowerPC 602 – a Motorola és IBM játékkonzolokba szánt, redukált PowerPC 603, órajel: 50 - 80 MHz.

1995 – Az IBM kibocsátja a Cobra vagy A10 processzort AS/400 rendszerekben – egycsipes processzor, 50-77 MHz órajellel.

1995 – Megjelenik a Sun 64 bites UltraSPARC processzora, SIMD multimédiás utasításokkal; órajele 143–200 MHz, támogatja a többprocesszoros rendszerekben való alkalmazást is

1995 – Az Intel Pentium Pro bemutatása

1996. január – MIPS R10000, avagy "T5" – MIPS IV utasításkészletű mikroprocesszor, szuperszámítógépekben való használatra.

1996 – ARM Ltd., ARM8 processzor (ARMv4. 0-72 MHz, 8KB cache, 5 fokozatú futószalag)[20]

1996 – Az AMD K5 processzor bemutatása.

1996 – Az IBM P2SC (POWER2 Super Chip) bemutatása, a POWER2 utódja, annak egycsipes megvalósítása, órajele 135 MHz. Az IBM Deep Blue számítógép, amely 1997-ben legyőzte Garri Kaszparovot, 30 db P2SC processzort tartalmazott.

1996 – PowerPC 603q - egy független PowerPC 603 kompatibilis processzor, a Quantum Effect Devices (QED) gyártmánya.

1996 – Az IBM nagy teljesítményű, többcsipes 4-utas SMP egysége: Muskie, A25 vagy A30, AS/400 gépekbe. Órajel: 125–154 MHz.

1997 – Bemutatják az AMD K6 processzort (Intel Pentium II ekvivalens, szuperskalár, órajel 166–300 MHz).

1997 vége – ARM Ltd., ARM9 processzor (ARMv4T, ARMv5TE, 0–200 MHz, dual cache, TDMI)[21]

1997 – Az IBM RS64 vagy Apache processzora: 64 bites PowerPC RISC processzor, RS/6000 és AS/400 gépekben szerepel, ismert még PowerPC 625 és A35 néven.

1997 – Sun picoJava I és picoJava II – a Sun Java- azaz nyelvspecifikus processzorai, amelyek közvetlenül hajtják végre a Java bytekódot.[22]

1997 – Az Intel Pentium II bemutatása.

1998. február 5. – Az IBM a világon elsőként demonstrál egy kísérleti CMOS mikroprocesszort, amely 1000 MHz fölötti órajellel működik. Ekkoriban az általános órajel 300 MHz alatt jár.[23]

1998. október – ARM Ltd., ARM10 processzor (ARMv5TE, 300MHz-1.2GHz, dual cache, 6 fokozatú futószalag)[24]

1998 – IBM RS64-II vagy Northstar (262 MHz).

1998. október 5. – Az IBM POWER3 processzora: 32/64 bites PowerPC utasításkészletet, valamint POWER2 alternatív utasításkészletet megvalósító architektúra. (Órajel: 200–450 MHz)

1998 – Az Intel Celeron bemutatása.

1999. június 23. – Az AMD K7, azaz az AMD Athlon – hetedik generációs x86 típusú processzor bemutatása.

1999 – IBM RS64-III vagy Pulsar (450 MHz).

1997 – Az Intel Pentium III bemutatása.
2000-től

2000 – Az AMD az Athlon XP és Duron processzorokkal jelent meg ebben az évben. Az IBM két különböző architektúrájú processzort jelentetett meg: az RS64-IV, más néven Sstar, processzort (ez egy többszálú futtattást támogató processzor, órajele 600 MHz, később 750 MHz), a z900 64 bites, az IBM zSeries sorozat szervereibe való processzort. Megjelent még a Fujitsu SPARC64 IV és az Intel ekkor még 32 bites Pentium 4-es processzorának első szériája.

2000. január 19. – a Transmeta cég bejelenti új processzorcsaládját, melynek kódneve Crusoe. Ez egy az Intel x86 architektúrával kompatibilis, kis fogyasztású, 128 bites VLIW végrehajtású processzor. Első modellje a 700 MHz-es TM3120 volt.[25]

2000. augusztus 23. – Intel XScale: ARM architektúrán alapuló 32 bites RISC mikroprocesszor.

2000. október – a Transmeta kibocsátja a 600 MHz-es Crusoe processzort (VLIW - nagyon hosszú utasításszót használó, alacsony fogyasztású processzor, hatékonyan képes emulálni egyéb processzorokat).

2000. november 20. – Megjelenik az Intel híres Pentium 4-es processzora (ekkor 32 bites, CISC, 1,5 GHz órajellel).

2001 – Az IBM POWER4 processzora: 64 bites PowerPC architektúra, többmagos processzor: két magot tartalmaz egy házban. Eredetileg 1,1 és 1,3 GHz órajellel működik, egy továbbfejlesztett POWER4+ modell elérte az 1,9 GHz-et is.

2001 közepén az Intel kibocsátotta új Itanium sorozatának első tagját, a Merced kódnevű processzort. Az Itanium architektúra 64 bites VLIW, eredetileg az Intel és HP által közösen fejlesztett EPIC-ből (explicitly parallel instruction computing) származó felépítés, a Merced ennek az első megvalósítása. Ezt a családot nagyteljesítményű szerverekbe szánták, képes emulálni az x86 utasításokat is. Kezdetben teljesítménye a középkategóriás RISC processzorokéhoz járt közel.

2001 júliusában megjelenik az SGI MIPS R14000-es (500 MHz-es RISC) processzora: 64 bites architektúra, amely lefelé kompatibilis az előző MIPS modellek 32 bites architektúrájával.

2001. október 9. – Az AMD bemutatja Athlon XP és MP processzorait.

2001 decembere – Elkészülnek a Fujitsu SPARC64 V processzorának első példányai. Ez egy négyszeres kibocsátású szuperskalár, sorrenden kívüli végrehajtású processzor, a 64 bites SPARC V9 architektúrát valósítja meg, órajele 1,1-től 1,35 GHz-ig terjed. A Fujitsu PRIMEPOWER szervereibe építették.

2002 – Megjelenik az Intel Itanium 2 processzora. A 2002-es első változat kódneve McKinley, amelyet az Intel a Hewlett-Packard-dal közösen fejlesztett ki, nagyteljesítményű szerverekben történő felhasználás céljaira. A 64 bites, az Intel Itanium utasításkészlet-architektúrát implementáló processzor 180 nm-es technológiával készült, 221 millió tranzisztorából csak 25 millió alkotta a logikát. Változatai 900 és 1000 MHz-es órajelekkel jelentek meg.

2003. március 12-én megjelent az Intel Pentium M sorozat első tagja, a Banias kódnevű modell.

2003 júliusában: Megjelenik az IBM PowerPC 970 processzora, elsőként az Apple Power Mac G5 modellben. Ez egy ötödik generációs PowerPC processzor, 64 bites, de visszafelé kompatibilis a 32 bites PowerPC architektúrával. Órajele 1,6-től 2,0 GHz-ig terjed.

2003. szeptember 23. – az AMD kibocsátja az Athlon 64-et, nyolcadik generációs AMD64-architektúrájú mikroprocesszorát.

2004 – A Transmeta 1,6 GHz-es Efficeon 256 bites, második generációs VLIW processzora.

2004. február 2-án az Intel bejelentette az első Prescott-maggal szerelt Pentium 4 processzorait, ezzel a Pentium is bekerült a 64 bites processzorok közé.

2005. november 14. – Megjelenik a Sun Niagara kódnevű processzora, másként az UltraSPARC T1: egy alacsony fogyasztású szerverprocesszor, 4, 6 és 8-magos kiépítésben készül, órajele 1,0-tól 1,4 GHz-ig terjed.

2006 – Megjelennek az IBM, Sony és Toshiba együttműködésben készülő Cell processzor első példányai.

2007 – Az Intel Core 2 Duo 2,5 GHz.

2007 – A Sun Niagara 2 avagy UltraSPARC T2 processzora: 4, 6 és 8-magos kiépítésben készül, 64 szálat futtat párhuzamosan, órajele 1,2-tól 1,6 GHz-ig terjed minden szálon. Minden magja külön FPU-t tartalmaz.

2008 – Az AMD Opteron Dual-Core 8222 2 GHz Socket F processzor.

2010 – Az Oracle Corporation (korábban Sun Microsystems) SPARC T3 processzora, másként UltraSPARC T3: 8 vagy 16 magos processzor, órajele 1,67 GHz, maximum 512 szálat futtat.

2010. július 22. – Az IBM kibocsátja a z196 processzort: 4 magos nagyszámítógépekbe szánt szerverprocesszor, CISC, órajele 3,8-tól 5,2 GHz-ig terjedhet.

A vezérlési, aritmetikai és logikai, illetve egyéb feldolgozási műveleteket elvégző központi egységet CPU-nak (Central Processing Unit, központi feldolgozó egység) hívjuk való kommunikációra, illesztésre

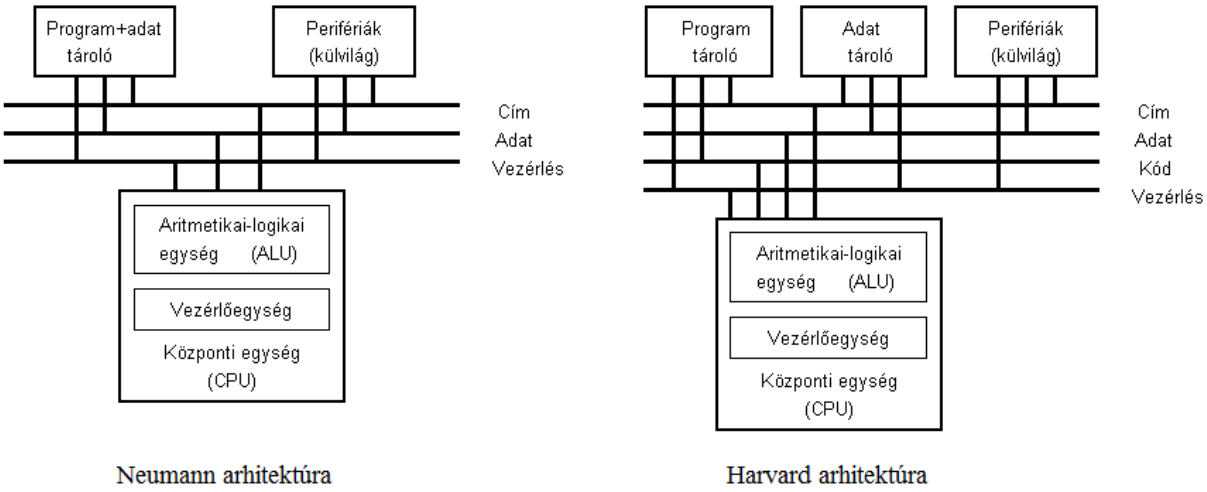
Az utasításkészlet, vagy utasításkészlet-architektúra (Instruction Set Architecture, ISA) a számítógép-architektúra programozáshoz kötődő része, ami magába foglalja a bitszélességet, a használható natív adattípusokat, gépi kódú utasításokat, regisztereket, címzési módokat, memóriaarchitektúrát, megszakítás- és kivételkezelést és a külső I/O-t. Az ISA specifikációjához tartoznak az opkódok (a gépi kód) és az adott processzor natív parancsai.

Az utasításkészlet-architektúra különbözik a mikroarchitektúrától, ami az utasításkészletet megvalósító processzortervezési technikák összessége. Nagyon különböző mikroarchitektúrájú számítógépeknek is megegyezhet az utasításkészletük. Például az Intel Pentium és az AMD Athlon az x86 utasításkészlet csaknem azonos változatát valósítja meg, de radikálisan különböző belső felépítéssel.

Egyes virtuális gépek utasításkészlete támogatja a Smalltalk, Java virtuális gép vagy a Microsoft Common Language Runtime-jának a bájtkódját oly módon, hogy a gyakrabban végrehajtott kódrészleteket natív gépi kódra fordítják, a kevésbé gyakoriakat pedig interpretálva futtatják (lásd Just-in-time compilation). A Transmeta hasonlóan valósította meg az x86 utasításkészlet futtatását VLIW processzorokon.

8.3.3. Számítógépes architektúrák

A mikroszámítógépek fejlődése során két architektúra terjedt el. Az egyik a Neumann architektúra, itt ugyanabban a memóriában van a programkód, mint az adatok. Ilyen architektúrájúak például az INTEL x86-os processzorok. A Harvard architektúra esetében külön memória áll rendelkezésre az adatok és az utasításkód tárolására (mikrokontrollerek, PIC család).



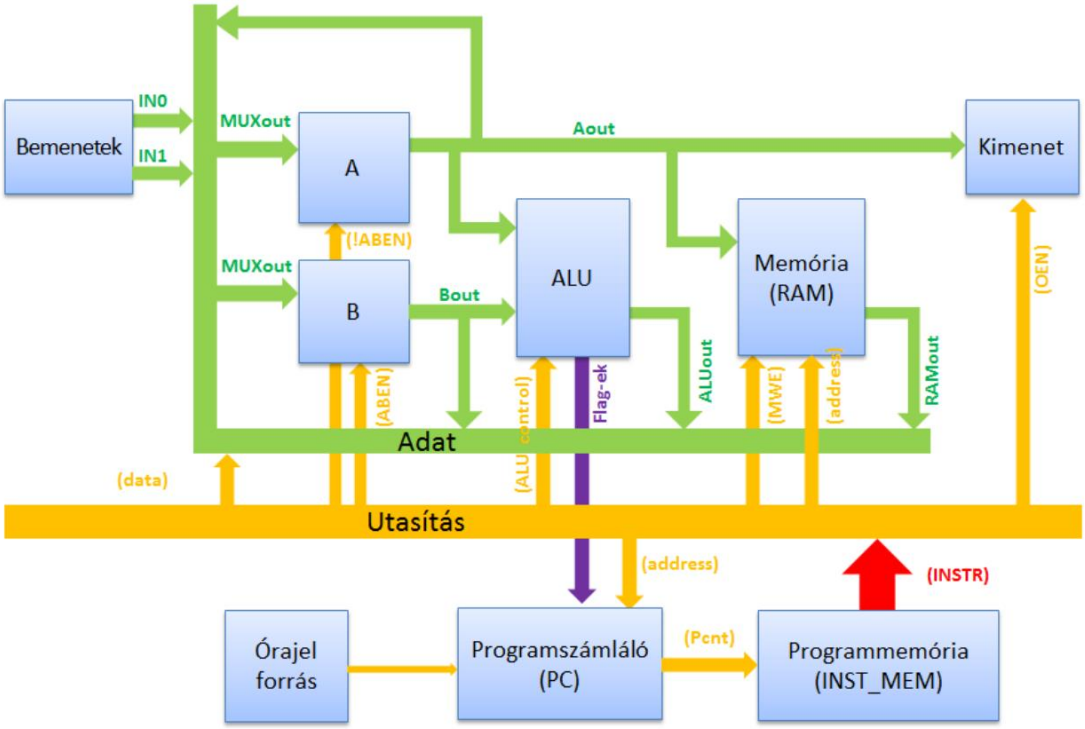
35. ábra - Neumann és Harvard számítógép architektúra

Neumann elvek:

- a kettes számrendszer alkalmazása
- teljes mértékben elektronikus elven működő számítógép

- központi vezérlő egység
- aritmetikai egység alkalmazása
- belső program- és adattárolás, programvezérlés

8.3.4. A CPU részei



36. ábra Az egyszerüsített CPU

Az CPU több részből áll:

1) **ALU:** (Arithmetic and Logical Unit – Aritmetikai és Logikai Egység). A processzor alapvető alkotórésze, ami alapvető matematikai és logikai műveleteket hajt végre. Sebessége növelhető egy koprocesszor (FPU, Floating Point Unit, lebegőpontos műveleteket végző egység) beépítésével. Az FPU korábban külön részegység volt, manapság a processzorok mindegyike beépítve tartalmazza. CPU "kalkulátora" tehát. Néhány alapvető műveletet képes végrehajtani:
   - Összeadás, kivonás, átvitel bitek kezelése
   - Fixpontos szorzás osztás
   - Logikai műveletek
   - Léptetések, bitek mozgatása jobbra/balra
   - Lebegőpontos aritmetikai műveletek
2) **AGU:** (Address Generation Unit) - a címszámító egység, feladata a programutasításokban található címek leképezése a főtár fizikai címeire és a tárolóvédelmi hibák felismerése.
3) **CU: (**Control Unit a.m. vezérlőegység vagy vezérlőáramkör). Ez szervezi, ütemezi a processzor egész munkáját. Például lehívja a memóriából a soron következő utasítást, értelmezi és végrehajtatja azt, majd meghatározza a következő utasítás címét. Tehát Utasítás dekódoló és végrehajtó egység. Irányítja és ütemezi az összes többi egység működését. Az adatokat vagy címeket a megfelelő útvonalon vezeti végig, hogy a kívánt helyre eljussanak Ha szükséges, beindítja az ALU valamelyik műveletvégző áramkörét.
4) **Regiszter (Register):** A regiszter a processzorba beépített nagyon gyors elérésű, kis méretű memória. A regiszterek addig (ideiglenesen) tárolják az információkat, utasításokat, amíg a processzor dolgozik velük. A mai gépekben 32/64 bit méretű regiszterek vannak. A processzor adatbuszai mindig akkorák, amekkora a regiszterének a mérete, így egyszerre tudja az adatot betölteni ide. Például egy 32 bites regisztert egy 32 bites busz kapcsol össze a RAM-mal. A regiszterek között nem csak adattároló elemek vannak (bár végső soron mindegyik az), hanem a processzor működéséhez elengedhetetlenül szükséges számlálók, és jelzők is. Ilyen például :

a. **utasításszámláló,** (PC=program counter, IP=instruction pointer) ami mindig a következő végrehajtandó utasítás címét.A PC (Program Counter Register)programszámláló regiszter mely a következő utasítás memóriacímét tartalmazza. Minden utasítás végrehajtása során egyel nő az értéke.
b. utasításregiszter (IR=instruction register), mely a memóriából kiolvasott utasítást tárolja. E kód alapján határozza meg a vezérlőegység az elvégzendő műveletet A memóriából kiolvasott utasítást tartalmazzaés a dekódoló egység értelmezi a tartalmát és ennek megfelelően ad ki vezérlő jeleket a többi egységnek.Ugrás esetén innen kerül a következő utasítás címe a PC-be, memória íráskor illetve olvasáskor ebből a regiszterből jut el a kívánt cím a memóriához (az MA regiszteren keresztül)
c. flagregiszter, amely a processzor működése közben létrejött állapotok jelzőit (igaz, vagy hamis),
d. veremmutató (SP = Stack Pointer) ,
e. és az akkumulátor, (AC) amely pedig a logikai és aritmetikai műveletek egyik operandusát, majd az utasítás végrehajtása után az eredményt tartalmazza. Ideiglenes tárolást (munkamemóriát) biztosít(anak) az ALU számára.
f. MA (Memory Address Register) memória címregiszter. Az MA és MD regiszterek tartják a közvetlen kapcsolatot a memóriával Az MA-ból jut a memória bemeneteire a kiválasztott rekesz címe (adatírás, -olvasás, utasításbeolvasás esetén).
g. MD (Memory Data Register) memória adatregiszter. A memóriából kiolvasott adat közvetlenül ide kerül, illetve a memóriába innen töltjük az adatokat

5) **Buszvezérlő**: A regisztert és más adattárolókat összekötő buszrendszert irányítja. A busz továbbítja az adatokat.
6) **Cache:** A modern processzorok fontos része a cache (gyorsítótár). A cache a processzorba, vagy a processzor környezetébe integrált memória, ami a viszonylag lassú rendszermemória-elérést hivatott kiváltani azoknak a programrészeknek és adatoknak előzetes beolvasásával, amikre a végrehajtásnak közvetlenül szüksége lehet. A mai PC processzorok általában két gyorsítótárat használnak, egy kisebb (és gyorsabb) első szintű (L1) és egy nagyobb másodszintű (L2) cache-t. A gyorsítótár mérete ma már megabyte-os nagyságrendű. A CPU belső tárolóelemei. Írásuk és

olvasásuk sokkal gyorsabb a memóriáknál Segítik a címképzést, tárolnak állapotjellemzőket, státusokat (ezzel a vezérlést segítik) Tartalmuk gyorsan és egyszerűen elérhető a CPU elemei számára.

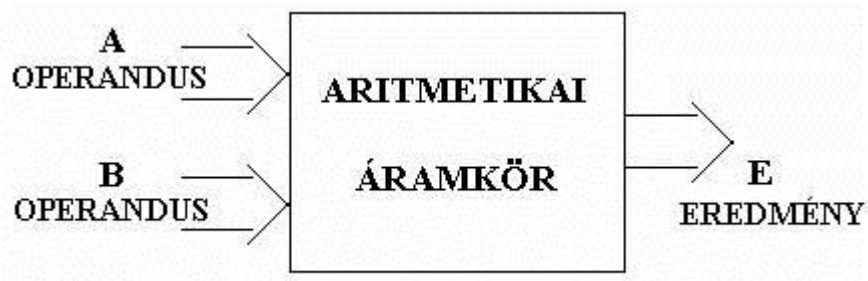
8.3.4.1. ALU

ALU: (Arithmetic and Logical Unit – Aritmetikai és Logikai Egység). A processzor alapvető alkotórésze, ami alapvető matematikai és logikai műveleteket hajt végre. Sebessége növelhető egy koprocesszor (FPU, Floating Point Unit, lebegőpontos műveleteket végző egység) beépítésével. Az FPU korábban külön részegység volt, manapság a processzorok mindegyike beépítve tartalmazza.

Ennek a segítségével hajtjuk végre a műveleteket.

Ebbe a csoportba olyan digitális áramkörök tartoznak, amelyekkel összeadási, kivonási, szorzási, osztási és egyéb műveletek végezhetők el.

A műveletvégzéshez szükséges adatokat (általában bináris, vagy BCD kódú bináris szám) az áramkör bemeneteire vezetjük, és a kimeneten kapjuk meg az eredményt valamilyen kód formájában.



8.3.4.1.1. Digitális komparátor

Vegyünk egy további példát, ahol megtanulhatjuk a kombinációs hálózatok felírását, természetesen a korábban megismert felírási lehetőségekkel.

❖ **Szöveges megfogalmazás**

Legyen egy digitális komparatorunk, melynek az a feladata, hogy két binárisan felírt számot hasonlítson össze. A két szám legyen eltárolva két biten.

❖ **Blokk**

Ebben az esetben a rendszer bemenete az a négy bit, ahol a két számot eltároljuk, kimenete pedig az a három változó, melyek igaz értékeivel képes eldönteni a két szám viszonyát. Erre azért van

szükség, mert a három kimenet állapota csak igaz vagy hamis lehet, egyéb értéket nem vehet fel. Így az igaz állapot „helyzete” mondja meg, hogy mi a két szám viszonya.

Ez azt jelenti, hogy

- Y_0 = 1 ha A > B egyébként 0
- Y_1 = 1 ha A = B egyébként 0
- Y_2 = 1 ha A < B egyébként 0

37. ábra A komparátor.

A rendszerben legyen (Azért, hogy az előbb tanult felírási módot tudjuk követni A_0 = D, A_1=C, B_0= B és B_1= A.)

❖ **Igazságtáblázat**

Ezután felrajzoljuk a rendszer igazságtáblázatát, amely 4 független változót és 3 függő változót tartalmaz, ezért 7 oszlopa lesz, viszont 4 független változójából következik, hogy 16 sorral fog rendelkezni. Felírjuk A, B C, D lehetséges kombinációit, és kitöltjük a táblázatot.

i	A	B	C	D	Y_0	Y_1	Y_2
0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	0	1
3	0	0	1	1	0	0	1
4	0	1	0	0	1	0	0

5	0	1	0	1	0	1	0
6	0	1	1	0	0	0	1
7	0	1	1	1	0	0	1
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	0
11	1	0	1	1	0	0	1
12	1	1	0	0	1	0	0
13	1	1	0	1	1	0	0
14	1	1	1	0	1	0	0
15	1	1	1	1	0	1	0

38. ábraA komparátor igazságtáblázata

❖ **Logikai függvények**

A logikai függvény felírásához csak egy egyenletrendszert veszünk most figyelembe, hogy egyszerűbben fel tudjuk írni a függvényt.

Tekintsük most csak azt az esetet, amikor

– Y_0 = 1 ha A > B egyébként 0

– Ekkor az igazságtáblában az alábbi sorok lesznek fontosak.

i	A	B	C	D	Y_0	Y_1	Y_2
0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	0	1
3	0	0	1	1	0	0	1

Y1	**4**	**0**	**1**	**0**	**0**	**1**	0	0
	5	0	1	0	1	0	1	0
	6	0	1	1	0	0	0	1
	7	0	1	1	1	0	0	1
Y2	**8**	**1**	**0**	**0**	**0**	**1**	0	0
Y3	**9**	**1**	**0**	**0**	**1**	**1**	0	0
	10	1	0	1	0	0	1	0
	11	1	0	1	1	0	0	1
Y4	**12**	**1**	**1**	**0**	**0**	**1**	0	0
Y5	**13**	**1**	**1**	**0**	**1**	**1**	0	0
Y6	**14**	**1**	**1**	**1**	**0**	**1**	0	0
	15	1	1	1	1	0	1	0

Az ebből felírható részegyenletek (soronként meghatározott mintermek):

$$Y1 = \overline{A}B\overline{C}\overline{D}$$

$$Y2 = A\overline{B}\overline{C}\overline{D}$$

$$Y3 = A\overline{B}\overline{C}D$$

$$Y4 = AB\overline{C}\overline{D}$$

$$Y5 = AB\overline{C}D$$

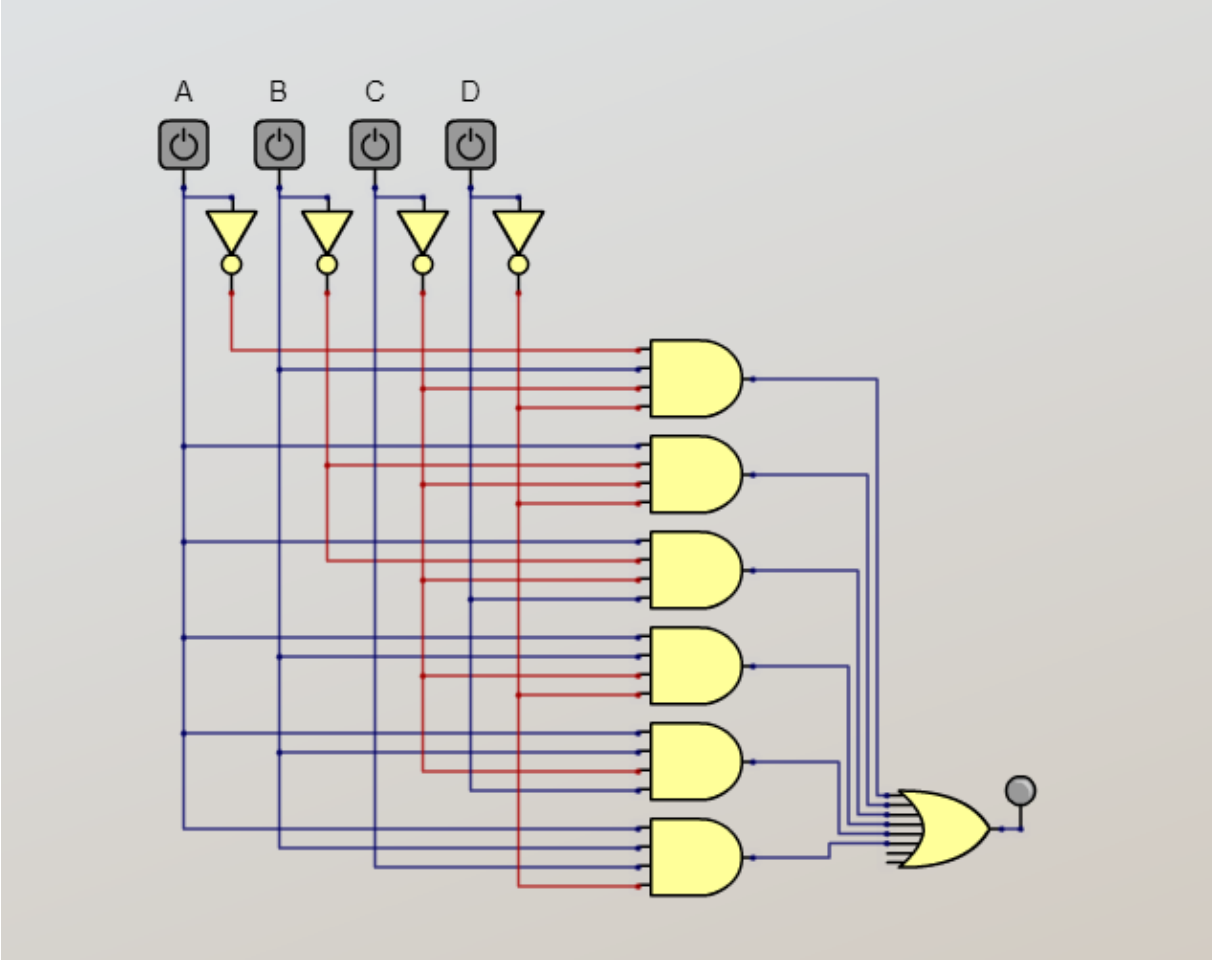
$$Y6 = ABC\overline{D}$$

Tehát az egyenlet:

$$Y = \overline{A}B\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D + AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D}$$

❖ **Kapcsolási rajz**

Ebből az egyenletből felépíthetjük a hálózatot is. Ehhez 6 darab És és egy VAGY kapura van szükségünk. Ezen kívül természetesen minden bemenetnél képeznünk kell a negáltat is.



39. ábraA komparátor kapcsolási rajza

❖ **Karnaugh - tábla**

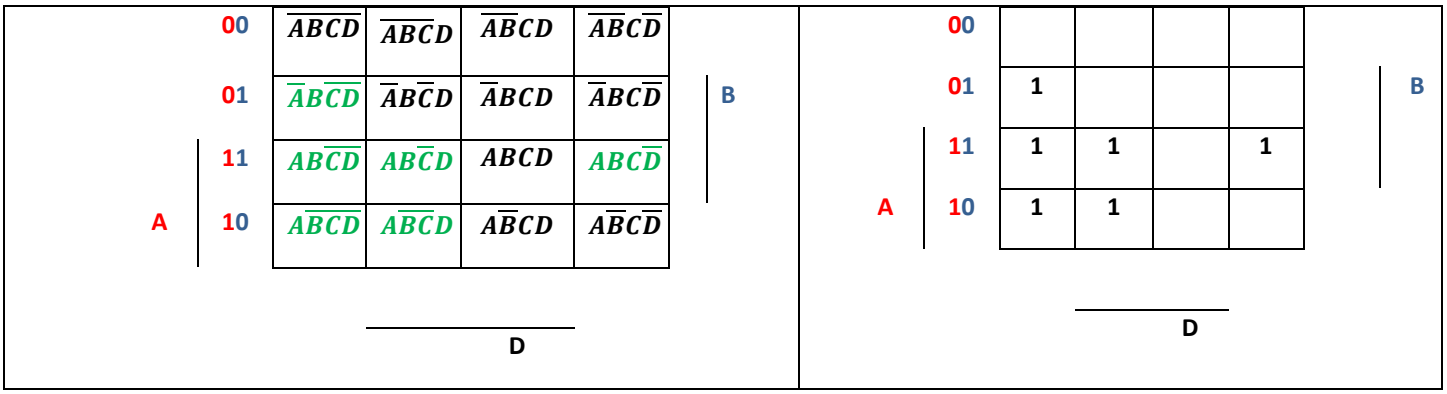
A Karnaugh tábla felírásához szükség van az előbbi egyenletre.

$$Y = \overline{A}B\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}C\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D}$$

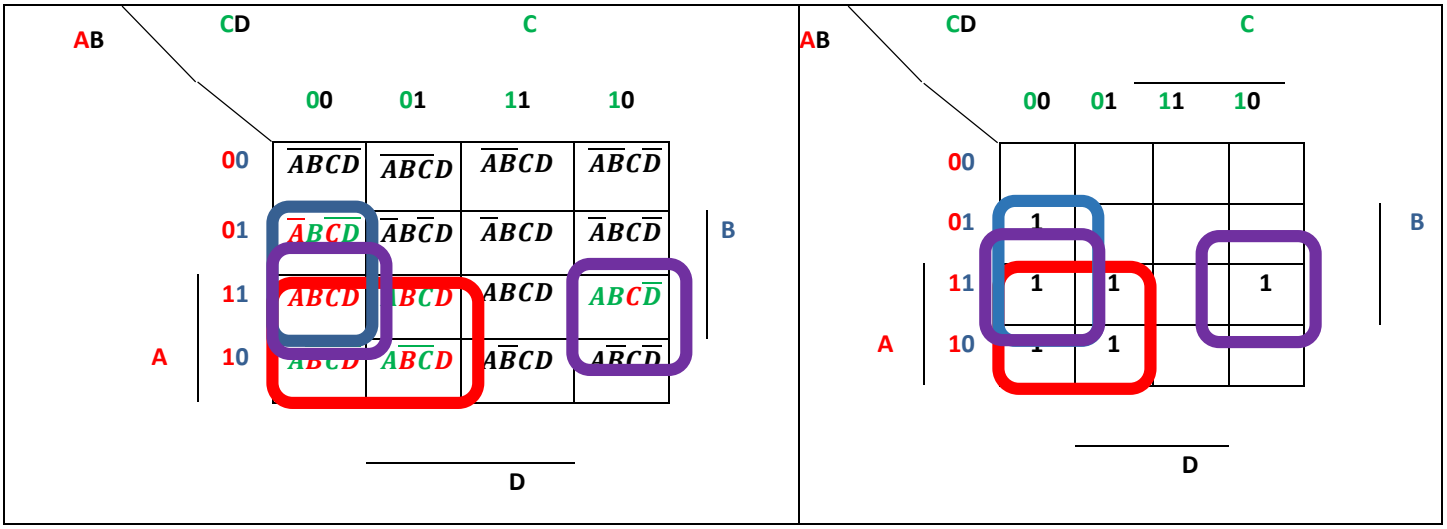
Ennek elemeit keressük meg a táblában, és írunk a helyére egyes értéket.

AB \ CD	00	01	11	10

AB \ CD	00	01	$\overline{11}$	10



Ezután a tábla egyszerűsíthető: (Pirossal a kieső tagok):

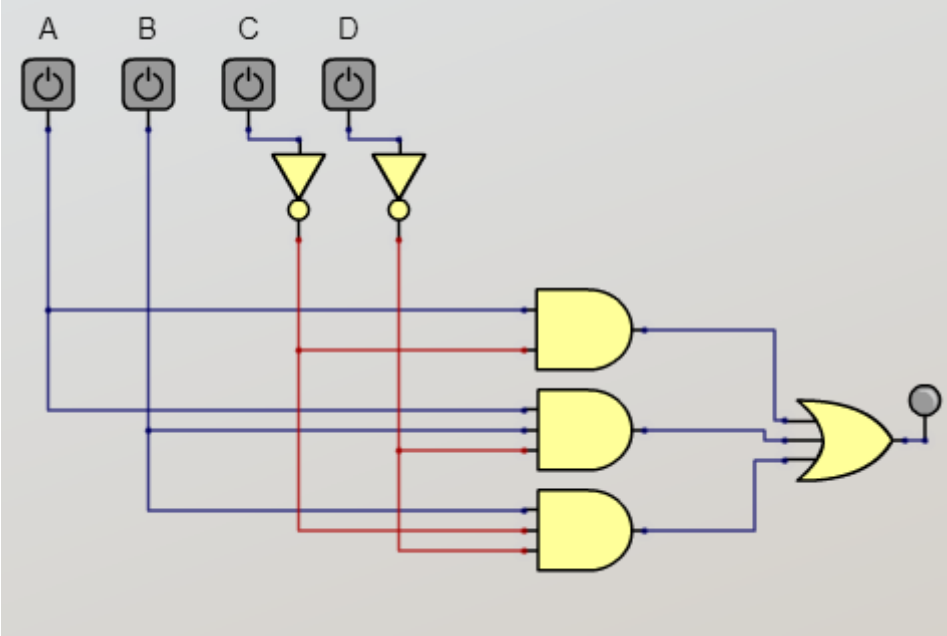


Az egyszerűsítés után kapható egyenlet:

$$Y = \overline{A}B\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}\overline{C}D + AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D}$$

$$Y = A\overline{C} + AB\overline{D} + B\overline{C}\overline{D}$$

Ezzel az egyszerűsítéssel tényleg egy könnyebben kezelhető rendszert kaphatunk: így már csak 3 ÉS kapu és egy VAGY Kapu szükséges a rendszer felépítéséhez. A négy negált helyett pedig csak kettőt használunk.

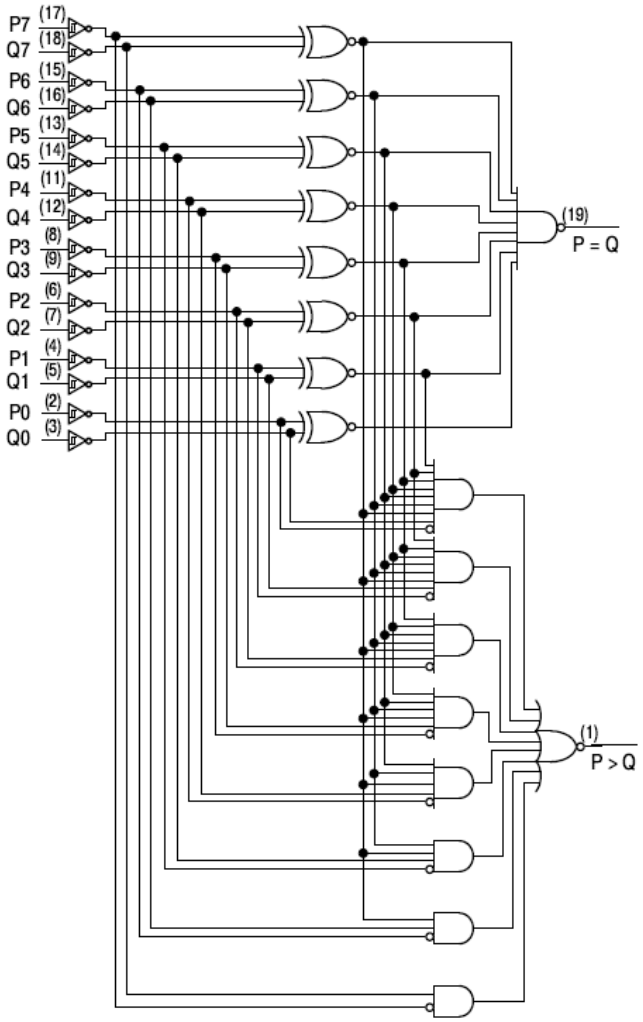


40. ábra Az egyszerűsített kapcsolási rajz

Konkrétan pl. egy két binaris számot összehasonlítását elvégző komparátor a következőképpen valósítható meg.

INPUTS			OUTPUTS	
DATA	ENABLES			
P, Q	$\overline{G}$, $\overline{GT}$	$\overline{G2}$	$\overline{P = Q}$	$\overline{P > Q}$
P = Q	L	L	L	H
P > Q	L	L	H	L
P < Q	L	L	H	H
X	H	H	H	H

41. ábra A digitális komparátor igazságtáblázata (74LS682)



42. ábra A digitális komparátor megvalósítása: (két bináris szám) (74LS682)

8.3.4.1.2. Összeadó

❖ **Az összeadás definíciója**

Olyan áramkörök, amelyek a bemeneteikre kerülő két operandust bitenként összeadják, figyelembe véve az egyes helyi értékeken keletkező átvitelt is, és az eredményt a keletkezett átvitellel együtt megjelenítik a kimenetükön. Ennek leggyakrabban alkalmazott formája a teljes összeadó.

A teljes összeadó megértéséhez fontos a számláló elvének megértése is.

A számlálás során, ha decimálisan írjuk a számokat, 10 darab számjeggyel írjuk le az összes lehetséges mennyiséget. A használt számjegyek 0,1,2,3,4, 5, 6, 7, 8, 9, .

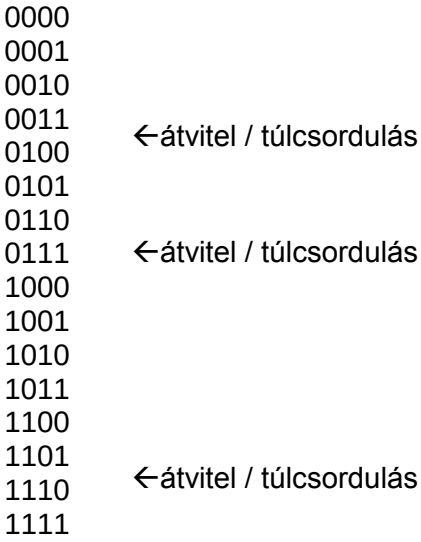
9 után azonban elfogynak a számjegyek. Mi ilyenkor a teendő?

A rendszerben ilyenkor túlcsordulás keletkezik: egyet lépünk a helyiértéken, és újra kezdjük a számlálást.

01
02
03
04
05
06
07
08
09
10 ←átvitel / túlcsordulás
11
12
…

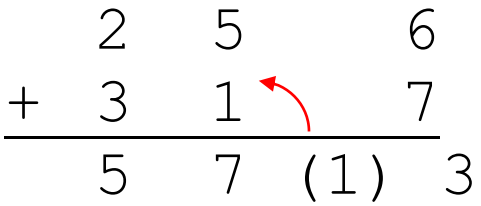
43. ábra A decimális számláló túlcsordulása

Ugyanez történik bináris rendszerben is, azzal a különbséggel, hogy mivel itt csak két számjegyünk van, ezért hamarabb fog jelentkezni a túlcsordulás.



44. ábra a bináris számláló túlcsordulása

Az összeadás során is ezzel a túlcsordulással kell számolni. Ezért a decimális esetben a papíron végzett összeadásnál, jobbról balra haladva sorra összeadjuk az egyes számjegyeket. Ahol kilencnél nagyobb eredményt kapunk, ott hozzáadjuk a maradék egyest az eggyel nagyobb helyiértékű számjegyekhez.



45. ábra Összeadás decimális szám esetén

A bináris esetben pedig ugyanígy járunk el, csak a felhasználható számjegyek 0 és 1. Ahol 1-nél nagyobb eredményt kapunk, ott hozzáadjuk a maradék egyest az eggyel nagyobb helyiértékű számjegyekhez.

```
      1     0     0     1     0     1     0          (74)
+     1     0     0     1     1     1     1         +(79)
-------------------------------------------------------
1  (1)0     0     1  (1) 1  (1) 0  (1)0    1          (153)
```

46. ábra Összeadás kettes számrendszerben

A **teljes összeadónak** 3 bemenete van, amelyekre a két összeadandót és az előző helyiértéken keletkezett átvitelt vezetjük. Ilyen 1 bites modulok felhasználásával készíthetünk több bites összeadó áramköröket.

Mivel akettes számrendszerben történő összeadéás esetén az átviteleket is figyelembe véve:

Összeadás	Példa
0 + 0 = 0 0 + 1 = 1 1 + 0 = 1 1 + 1 = 10	1011_2 $+\quad 11_2$ 1110_2

Kivonás	Példa
0 − 0 = 0 0 − 1 = −1 1 − 0 = 1 1 − 1 = 0	1011_2 $-\quad 111_2$ 100_2

Szorzás	Példa
0 . 0 = 0 0 . 1 = 0 1 . 0 = 0 1 . 1 = 1	1010_2 $\cdot \quad 11_2$ 11110_2

Osztás	Példa
0 / 0 = n.def. 0 / 1 = 0 1 / 0 = n.def. 1 / 1 = 1	1010_2 $/ \quad 10_2$ 101_2

47. ábra A műveletek kettes számrendszerben

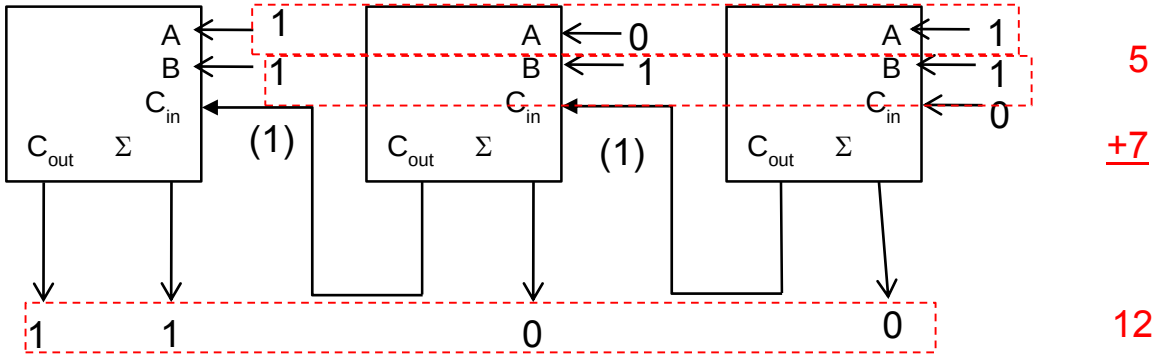
Ezért a rendszer igazságtáblázata a következőképpen írható fel:

C_{in}	A	B	Σ	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

48. ábra Az összeadó igazságtáblázata

Az összeadó (Áramköri felépítése az alábbi ábrán látható.) tehát

- Két bináris szám egy-egy bitjét adja össze
- Figyelembe veszi az eggyel kisebb helyiértékről érkező átvitelt.
- Ha nála is keletkezik átvitel, akkor továbbítja azt.
- Több egybites összeadót összekapcsolva két bármilyen hosszú bináris számot összeadhatunk
- Kettes komplemens kóddal kivonóként is használható, az utolsó átvitel használata nélkül



49. ábra Az teljes összeadó

A teljes összeadó működését tekintve kétféle lehet: soros vagy párhuzamos.

❖ **Soros összeadó**

Az egyes helyiértékeken lévő bitek összeadása időben egymás után történik, kezdve a legkisebb helyiértékkel. Így csak egy 1 bites teljes összeadó áramkörre, regiszterekre és egy 1 bites késleltető tárolóra van szükség. Ezzel a módszerrel viszonylag hosszú idő szükséges egy nagyobb bitszámú művelet elvégzéséhez.

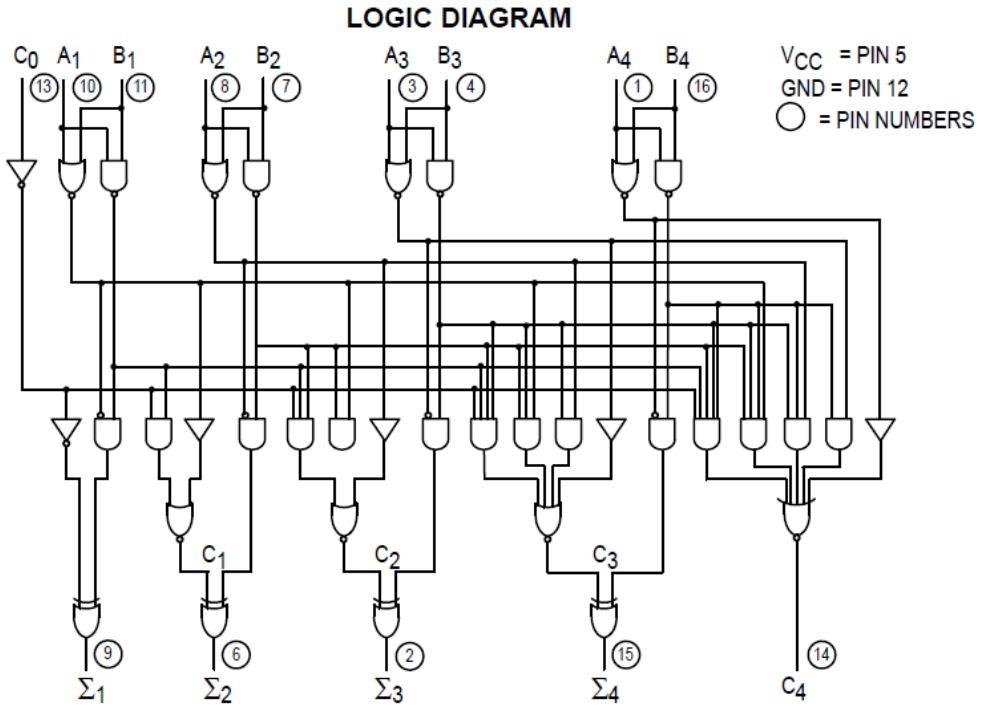
❖ **Párhuzamos összeadó**

Az egyes helyiértékeken lévő bitek összeadása azonos időben történik, de természetesen az átvitelbitek megjelenéséig szükség van valamekkora továbbterjedési időre, vagy valamilyen áramkörre, amely az átvitel bitek létrehozását gyorsítja.

❖ **Példa**

Jó példa az összeadóra a **74LS8 amely egy 4 bites teljes összeadó.**

50. ábra A négy bites öszeadó blokkja



51. ábra A négy bites számláló kapcsolási rajza

	C_0	A_1	A_2	A_3	A_4	B_1	B_2	B_3	B_4	Σ_1	Σ_2	Σ_3	Σ_4	C_4	
Logic Levels	L	L	H	L	H	H	L	L	H	H	H	L	L	H	
Active HIGH	0	0	1	0	1	1	0	0	1	1	1	0	0	1	(10+9 = 19)
Active LOW	1	1	0	1	0	0	1	1	0	0	0	1	1	0	(carry+5+6 = 12)

52. ábra A négy bites számláló igazságtáblázata

8.3.4.1.3. Kivonó

❖ **A kivonás definíciója**

A kivonás művelete matematikailag **a kivonandó szám mínusz egyszeresének hozzáadásával végezhető el.** Erre a 2-es komplemens kódot használják. Az áramköri megoldás lényege az, hogy a kivonandó minden bitjét negáljuk és ehhez még 1-et hozzáadunk. Majd az így keletkezett operandust és a másik operandust vezetjük be egy összeadó áramkörbe. Az így kapott eredmény 2-es komplemens kódban áll rendelkezésre. A kivonandó komplemensét előállíthatjuk párhuzamosan vagy sorosan.

A kivonó áramkörök szintén lehetnek soros vagy párhuzamos működésűek.

❖ **Példa:**

A számítógépek a kivonást a kettes komplemens segítségével végzik. A kisebbítendőhöz hozzá kell adni a kivonandó kettes komplementerét. Ez két lépést jelent. Át kell váltani a kivonandót kettes számrendszerbe, majd a bitjeit át kell billenteni, majd 1-et hozzá kell adni. Az így kapott számot és a kisebbítendő bináris számát össze kell adni. például vonjuk ki a 8-ból az 5-öt.

8-5=3

8: 1000 5: 0101

5 I: 1010 (negálás)

5 II: 1011 (negálthoz adunk 1-et)

Ezeket most össze kell adni:

1000

+1011

_____

10011

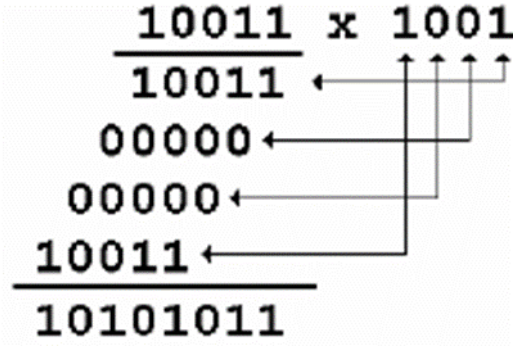
A kapott eredmény az 10011. Itt viszont keletkezik egy felesleges bit, túlcsordulás, amit le kell választani, arra nincs szükség, ezt levágjuk.

1 | 0011

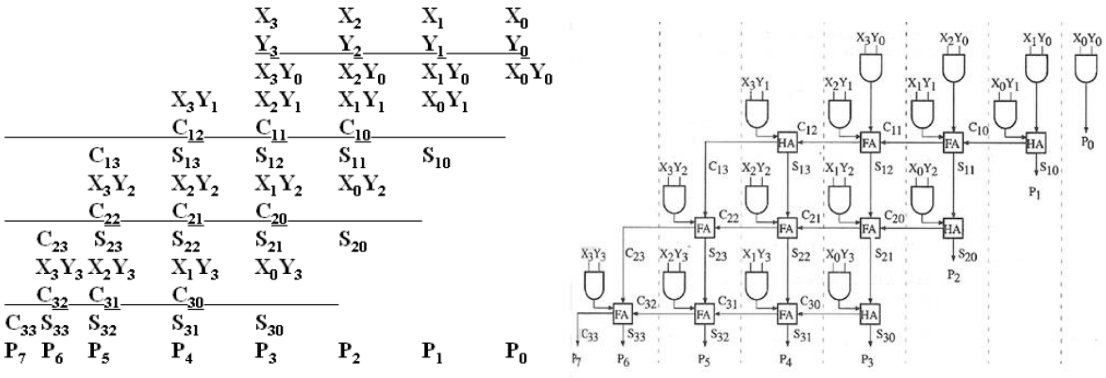
Az így kapott számot, ha vissza váltjuk 10-es számrendszerbe, akkor megkapjuk az eredményt. 0011 = 1+2 = 3

8.3.4.1.4. Szorzás

Két bináris szám összeszorzásánál is ugyanúgy járunk el, mint a decimális esetben: A szorzandót a szorzóval úgy szorozzuk meg, hogy a szorzó mindenegyes helyiértékévek külön elvégezzük a szorzást, és a részeredményeket összeadjuk. A helyiértékek helyességére, ugyanúgy figyelünk, mint decimális esetben. fontos megjegyezni, hogy két N bites szám szorzata egy 2N bites számot eredményez!



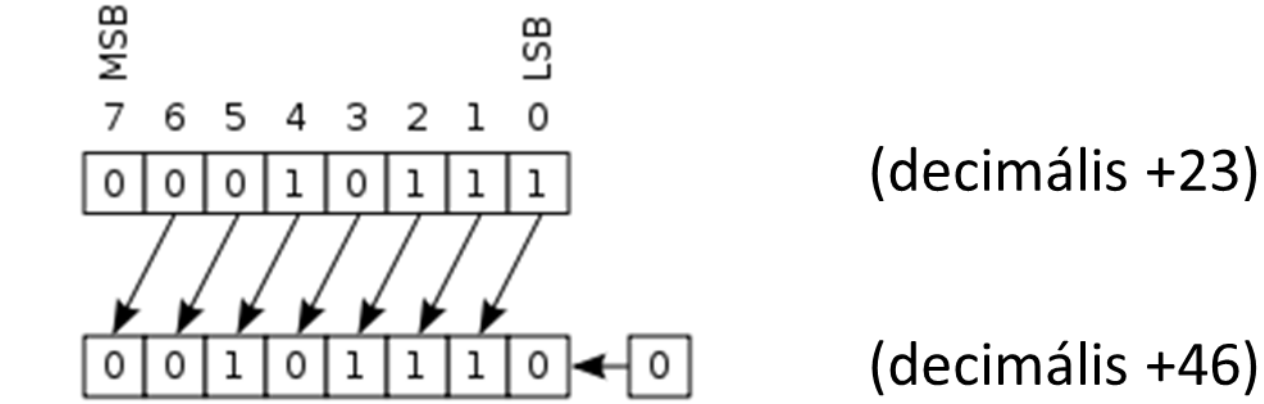
53. ábra A bináris szorzás



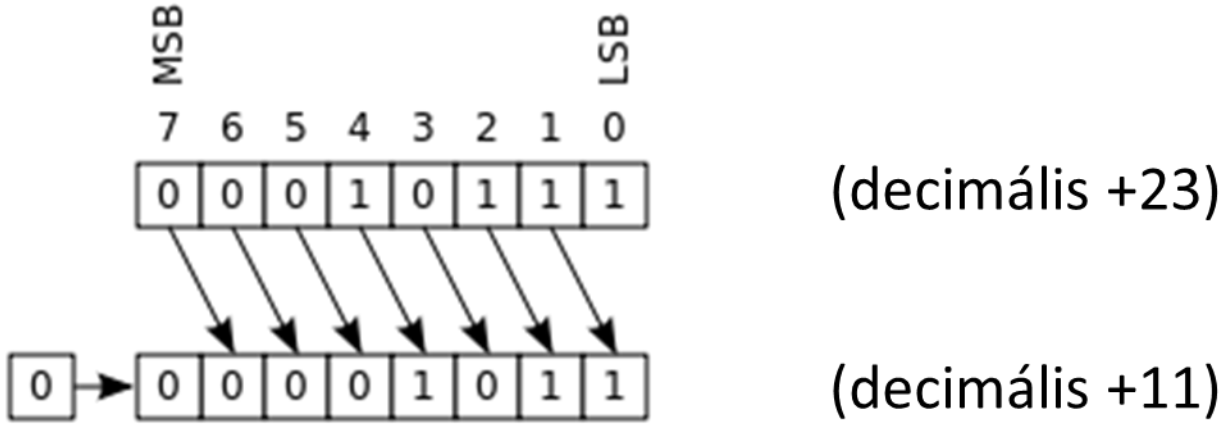
54. ábra A bináris szorzás megvalósítása logikai kapukkal

A szorzás megvalósítása kapukkal mindenképp bonyolultabb művelet, ezért egy egyszerűbb megoldásra kell törekedni. Ezt a lehetőséget a regiszterek kínálják a számunkra. A regiszterek ugyanis jobb, bal és mindkét irányba léptethetők. A léptetés során egy bit (ahogy „toljuk”) kiesik, hogy a megfelelő bitszámunk meglegyen, ezért egy nulla logikai értékkel helyettesítjük, abból az irányból,

melyből eltoltuk. Ugyanakkor fontos, hogy a balra tolás egy 2 – vel való szorzásmnak a jobbra tolás pedig egy kerttővel való osztásnak felel meg.



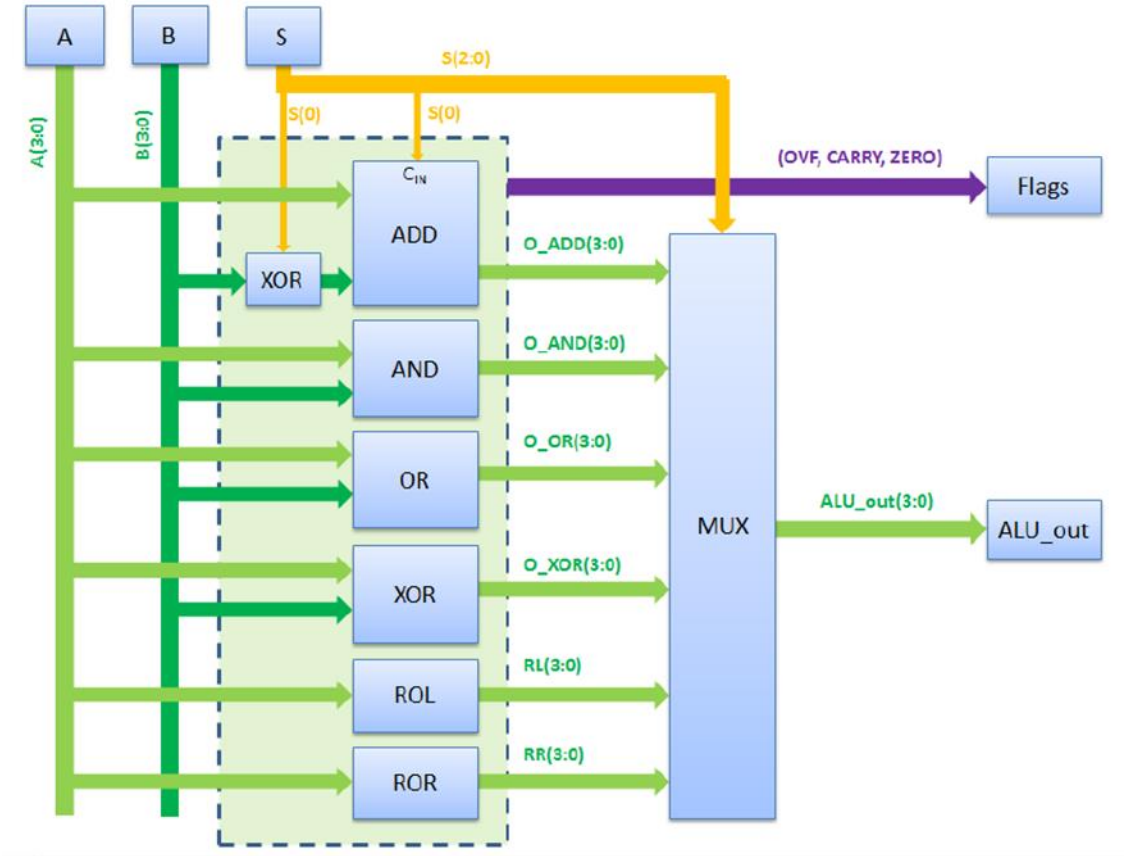
55. ábra Balra shiftelés - kettővel való szorzás



56. ábra Jobbra shiftelés - kettővel való osztás

Mivel ez egy jóval egyszerűbb megvalósítás, ezért a bonyolultabb számításokat is igyekszünk erre visszavezetetni.

8.3.4.1.5. Az ALU részei



57. ábra Az ALU részei (gyakorlaton megvalósított)

Az ALU részei tehát a műveleti elemek:

- osszeadó és kivonó
- ÉS művelet
- Vagy művelet
- XOR művelet
- Balra shiftelés
- Jobbra shiftelés

ADD: Két négybites szám összeadására képes teljes összeadó. Ha kivonást is akarunk végeztetni az összeadóval, akkor a második operandus (B) kettes komplemenést (bitenként negált +1) kell az elsőhöz (A) hozzáadnunk. A negálást exkluzív vagy kapuval, az egyes hozzáadását az első helyi értéken a CIN-re adott egyessel oldhatjuk meg.

AND, OR, XOR: A két négybites bemenet bitenkénti és, vagy, kizáró-vagy kapcsolatát adja eredményül.

ROL, ROR: Az A bemenet egy bittel történő balra, illetve jobbra forgatása.

MUX: A multiplexer a vezérlő jeleknek megfelelő műveletvégző egység kimeneti jelét választja ki. Az ALU-ban minden műveletvégző egység kimenetén megjelenik az adott műveleti eredmény, ezért szükséges a multiplexer, ami csak a megfelelő műveleti eredményt adja a kimenetre. Ez lesz az ALU kimeneti értéke.

Az S szelektjel segítségével választjuk ki a feladatokat:

Kiválasztó			Kimenet	Leírás
S2	S1	S0		
0	0	0	Y = A + B	Az 'A' és 'B' bemenet összege.
0	0	1	Y = A - B	Az 'A' és 'B' bemenet különbsége
0	1	0	Y(i) = A(i) & B(i)	Az 'A' és 'B' bemenet bitenkénti ÉS kapcsolata
0	1	1	Y(2:0) = A(3:1), Y(3) = A(0)	Az 'A' bemenet rotálása jobbra.
1	0	0	Y(i) = A(i) | B(i)	Az 'A' és 'B' bemenet bitenkénti VAGY kapcsolata
1	0	1	Y(3:1) = A(2:0), Y(0) = A(3)	Az 'A' bemenet rotálása balra.
1	1	0	Y(i) = A(i) ^ B(i)	Az 'A' és 'B' bemenet bitenkénti XOR kapcsolata
1	1	1	-	-

58. ábra A 3 bites szelektjel használata

Ezen kívül az ALU rendelkezik ún. FLAG-ekkel. A flagek az ALU belső állapotáról tájékoztatnak.

Overflow: Előjeles túlcsordulás. Előjeles számok esetében az összeadás, vagy a kivonás eredménye nem ábrázolható az adott számtartományon.

Carry: Átvitel előjel nélküli műveleteknél. Az összeadás eredménye nem ábrázolható az adott számtartományban.

Zero: Az ALU művelet eredmény nulla lett.

8.3.4.2. AGU

AGU: (Address Generation Unit) - a címszámító egység, feladata a programutasításokban található címek leképezése a főtár fizikai címeire és a tárolóvédelmi hibák felismerése.

8.3.4.3. CU

(Control Unit a.m. vezérlőegység vagy vezérlőáramkör). Ez szervezi, ütemezi a processzor egész munkáját. Például lehívja a memóriából a soron következő utasítást, értelmezi és végrehajtatja azt, majd meghatározza a következő utasítás címét.

A vezérelt rendszer kívánt algoritmus szerinti működését:

- Felügyelik
- Szervezik

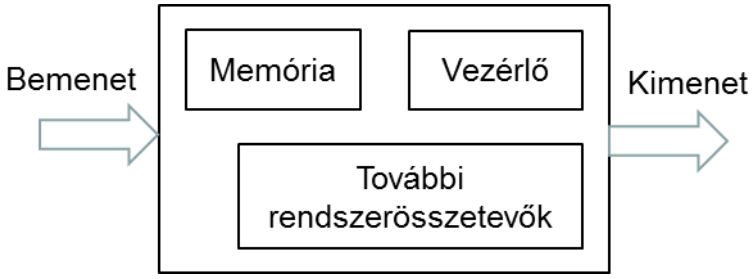
A vezérlő működési programja lehet:

- Fázisregiszteres

Kötött struktúrájú és a fázisregiszter (ez lehet egy számláló is) tárolja a vezérlő belső állapotát A belső állapotnak megfelelően kell előállítani a kimeneti vezérlő jeleket. A módosítás azonban nehézkes

- Mikroprogramozott

A vezérlőt működtető utasítások sorozata (a program) egy memóriában van tárolva. Az aktuális utasítás megadja az adott fázisban végrehajtandó műveletet és a következő utasítás memórián belüli címét.



59. ábra A CU

8.3.4.4. Regiszter

A regiszterek funkciója az **<u>átmeneti tárolás.</u>** Ezért azonos órajelű D tárolókból épül fel. Azért van szükség több tárolóra, mert több bites adatok átmeneti tárolására alkalmas. Ezek az adatok egyrészt lehetnek vezérlő információk, másrészt a regiszterek alkalmasak műveletek operandusainak és eredményének tárolása is.

Mivel a regiszter azonos órajelű D tárolókból épül fel, ezeknek a tárolóknak a működését össze kell hangolni. Ezt egyrészt megteszi a minden tárolón azonos órajel, másrészt a regiszter ún léptetése.

Léptetés alatt azt értjük, hogy a tárolók kimenete egy másik tároló bemenetére csatlakozik, ezért az órajel hatására az információ az egyik tárolóból a másikba képes íródni.

Az ilyen típusú regisztereket ún. léptető, vagy shiftregisztereknek nevezzük.

A léptetés történhet

- jobbra,
- balra,
- mindkét irányba.

A bemeneti jel tehát áthaladva a láncon késleltetve, de változatlanul jelenik meg a kimeneten. A léptetőregiszter bináris jelek tárolására szolgál és minden egyes flip flop, amely a regisztert alkotja, egy bit információ tárolására alkalmas. A léptetőregiszterekben a megfelelő működés érdekében, hogy minden léptetési parancsra egy és csakis egy léptetés történjen, feltétlenül órajelvezérelt flip-flopokat kell alkalmazni. Tudjuk azt is, hogy közös vezérlőjelekkel vannak ellátva, így valamennyi flip-flop egyszerre azonos műveletet végez.

A léptetőregiszterek esetén a soros és párhuzamos beírás és kiolvasás, valamint a kétféle léptetési irány lehetőségének variációival sokféle típus állítható elő. Az ábra az egy irányban (jobbra) léptethető, soros beírású és kiolvasású 4 bites áramkör kapcsolását szemlélteti. Ha az áramkör minden flip-flopját kimeneti kivezetéssel látjuk el, a léptetőregiszter párhuzamos kiolvasásra is alkalmassá válik.

Az áramkör működésekor a soros adatbemenetre érkező 1 bites jel csak egy órajel hatására kerül az első D tároló kimenetére. Minden egyes órajel hatására a következő tároló kimenetén jelenik meg, tehát négy órajel hatására kerül az áramkör kimenetére.

Használjuk fel a flip-flopok statikus beíró és törlő bemeneteit is! Így alkalmassá válik a léptetőregiszter párhuzamos beírásra is. Az ábra egy 4 bites soros/párhuzamos kiolvasású és beírású léptetőregiszter kapcsolási rajzát mutatja be.

Az áramkör működésekor a párhuzamos beírás és a léptetés funkcióját az ÉS-VAGY kapuk választják szét. A beírás előtt szükséges törlés a CLR (clear: törlés) bemeneten keresztül történik logikai 0 szinttel. Beírni az S (set: beírás) bemenetről logikai 0 szinttel lehet.

Ha az üzemmód vezérlő (angolul: MC - Mode Control) bemenetén logikai 0 szint van, a kettős ÉS kapucsoportok kapui közül azok vannak engedélyezve, amelyek a párhuzamos bemenetről veszik az információt. Ha a vezérlőbemenet logikai 1 es szinten van, akkor az ÉS kapucsoportok kapui közül azok vannak engedélyezve, amelyek a szomszédos flip-flop kimenetére csatlakoznak. A bemenő jel a soros bemenetről érkezik.

Összegezve a vezérléseket:

- Párhuzamos beírás MC = 0 esetén lehetséges
- Soros beírás MC = 1 esetén lehetséges

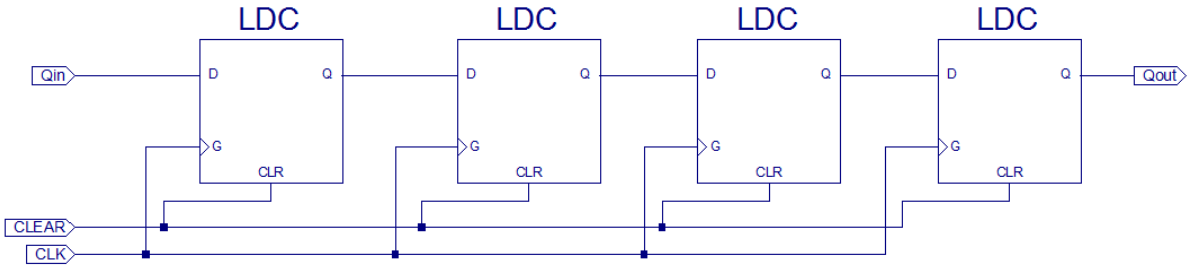
Ne felejtsük el, hogy bármelyik működés csak órajel hatására jön létre.

A léptetőregiszter (shift- vagy helyiérték toló regiszter) flip-flopok olyan lánca, amely lehetővé teszi, hogy a bemenetére adott információ minden egyes órajel hatására egy flip-floppal tovább lépjen. A léptetés irányának a megadására a jobbra, illetve balra megjelöléseket alkalmazzák.

Soros beírásnál és kiolvasásnál a regiszter első és utolsó flip-flopjához lehet hozzáférni. Ebben az esetben szükséges, hogy az információt a regiszterben léptetni lehessen. Ezeket a regisztereket a léptetőregisztereknek nevezzük.

Párhuzamos beírásnál és kiolvasásnál az információt a regiszter minden flip-flopjába egyszerre írják be, illetve onnan egyszerre olvassák ki. Mivel ezeknél a regisztereknél léptetés nem szükséges, a regiszter csak tárolási feladatra alkalmas. Ezeket a típusokat átmeneti tároló vagy közbenső tároló (puffer) regisztereknek nevezik.

8.3.4.4.1. Jobbra léptető shiftregiszter megvalósítása

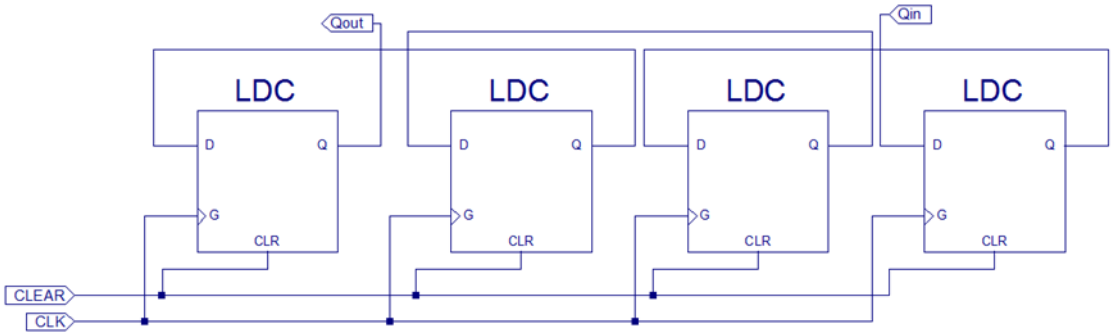


60. ábra A jobbra léptető regiszter

```
Process
Begin
   if (clear = '1') then
      q <= "0000";
   elsif (clk`event and clk = '1') then
      q(2 downto 0) <= q(3 downto 1);
      q(3) <= qin;
   end if;
end;
```

61. ábra A jobbra léptető regiszter kódja

8.3.4.4.2. Balra léptető shift regiszter megvalósítása



62. ábra Léptetőregiszter balra

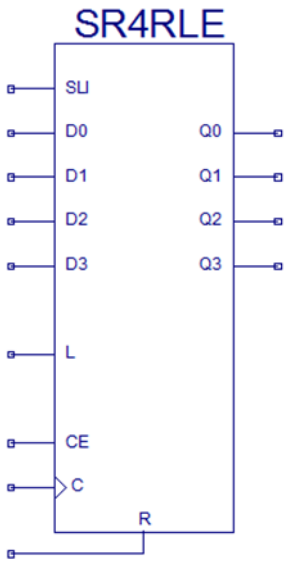
```
        Process
        Begin
          if (clear = ’1’) then
                    q <= ”0000”;
          elsif (clk`event and clk = ’1’) then
                    q(3 downto 1) <= q(2 downto 0);
                    q(0) <= qin;
end if;
        end;
```

63. ábra A balra léptető regiszter kódja

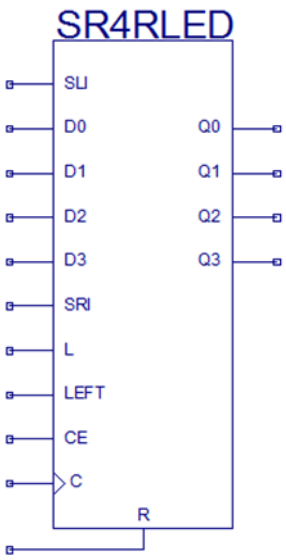
8.3.4.4.3. Mindkét irányba léptető shift regiszter megvalósítása

A jobbra balra léptetés esetén párhuzamos be- és kimenet, és/vagy soros be- kimenet működik. Ezért a regiszter alkalmas soros/párhuzamos átalakításra.

Az alábbi két rajzon két léptetőregiszter látható,



64. ábra Léptetőregiszter 1



65. ábra Léptetőregiszter 2.

ahol

SLI: baloldali soros bemenet

SRI: jobboldali soros bemenet

D: párhuzamos bemenetek

Q: párhuzamos kimenetek

CE: órajel engedélyező bemenet

C: órajel

L: beírás engedélyezés

LEFT: balra/jobbra léptetés

R: szinkron törlés

A regiszter portjait természetesen állapottáblával is felírhatjuk.

Inputs								**Outputs**		
R	**L**	**CE**	**LEFT**	**SLI**	**SRI**	**D3 : D0**	**C**	**Q0**	**Q3**	**Q2 : Q1**

1	X	X	X	X	X	X	↑	0	0	0
0	1	X	X	X	X	D3 : D0	↑	D0	D3	Dn
0	0	0	X	X	X	X	X	No Change	No Change	No Change
0	0	1	1	SLI	X	X	↑	SLI	q2	qn-1
0	0	1	0	X	SRI	X	↑	q1	SRI	qn+1
qn-1 or qn+1 = state of referenced output one setup time prior to active clock transition										

66. ábraA Shift regiszter állapottáblája

8.3.4.4.4. A regiszter felhasználása

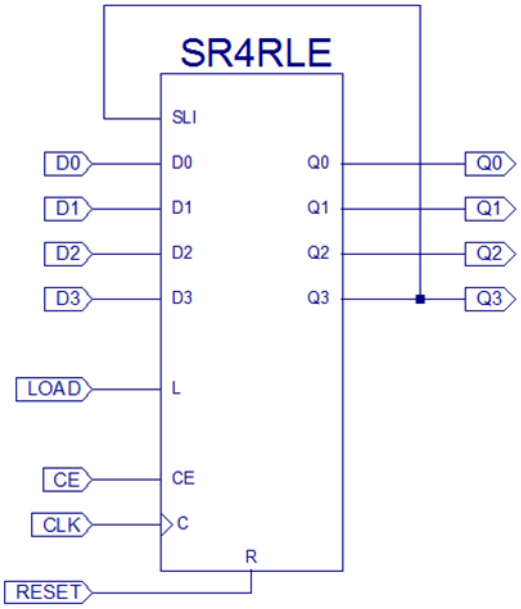
A regisztereket több dologra fel lehet használni. Készíthetünk belőle átmeneti adattárolót, és speciális számlálót, valamint felhasználhatjuk a véletlen számok előállítása során is.

8.3.4.4.5. Adat átmeneti tárolása

A regiszter alap felhasználásának tekinthető a d tárolókon keresztül mozgatott adatok átmeneti tárolás és továbbítása, melyről szó volt az előzőekben.

8.3.4.4.6. A gyűrűs számláló

A gyűrűs számláló speciális számlálónak tekinthető, hiszen egy regiszterrel valósítjuk meg, amely nem számol valójában, hanem működése során adatot mozgat. A gyűrűs számláló megvalósítása során az előző regisztert alakítjuk át egy hurok segítségével, melyben a Q3kimenetetet és az SLI bemenetet kapcsoljuk össze.

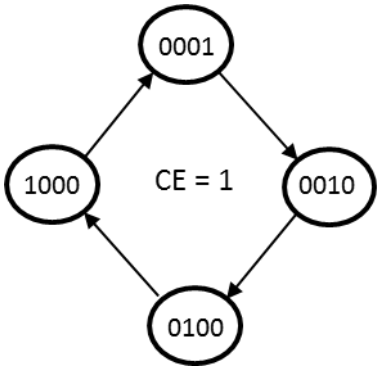


67. ábra A gyűrűs számláló megvalósítása

A működtetés során a LOAD bemenettel a D3-D0 bemeneteket 0001 alaphelyzetbe állítjuk. Az órajel engedélyezése után minden órajel ciklusban az 1-es továbblép a következő helyi értékre.A visszacsatolás miatt 4 ciklus után újra kezdődik a folyamat. A 2n (16) lehetséges állapotból csak n (4) valósul meg (12 tiltott állapot).

Q_3	Q_2	Q_1	Q_0
0	0	0	1
0	0	1	0
0	1	0	0
1	0	0	0
0	0	0	1
0	0	1	0
...			

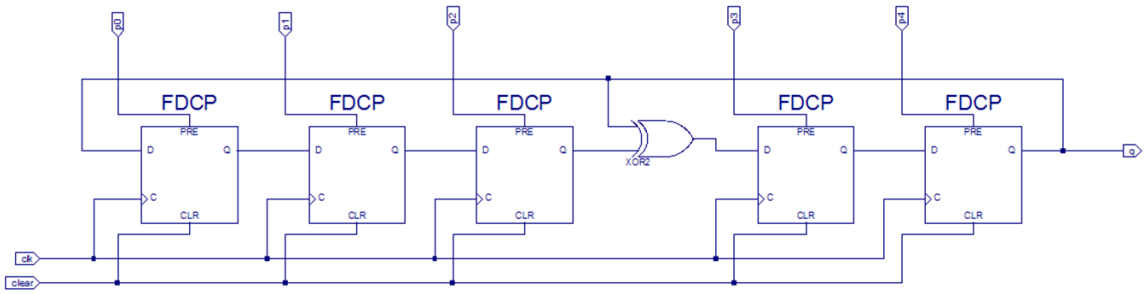
68. ábra a gyűrűs számláló állapottáblája



69. ábra A gyűrűs számláló működése

8.3.4.4.7. Véletlen számgenerátor

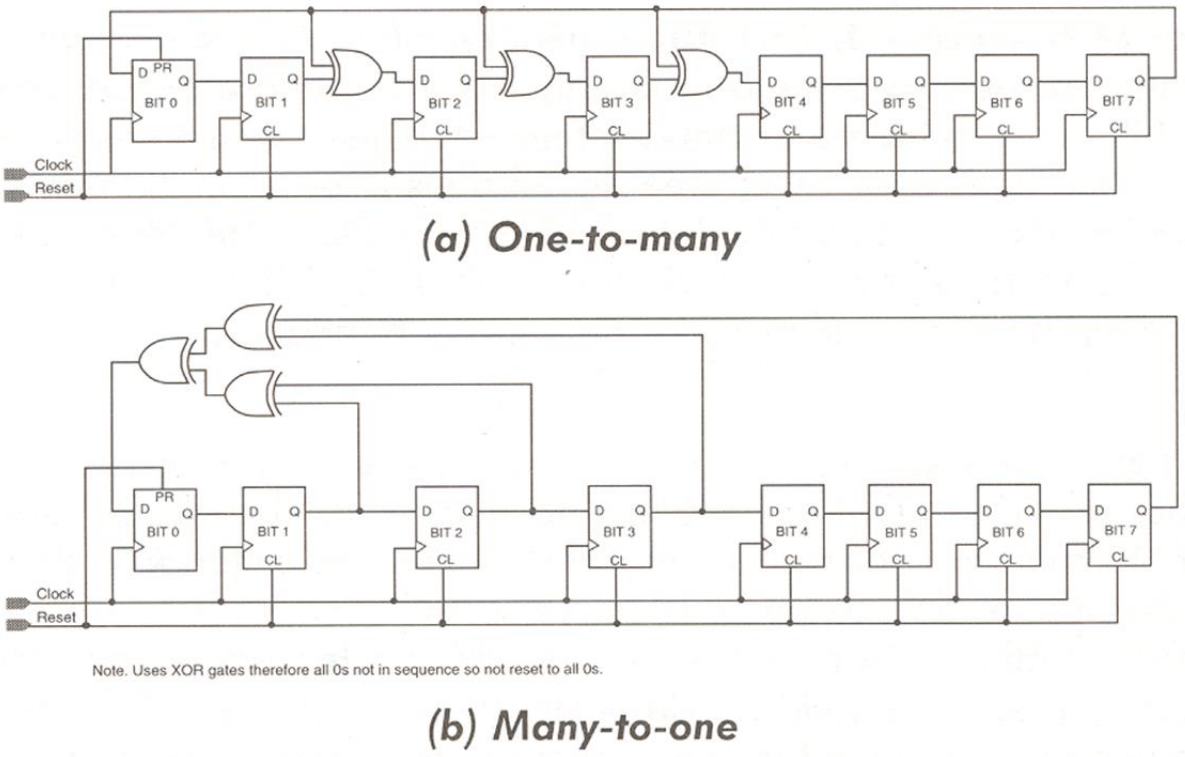
Az álvéletlen szám generátor léptetőregiszterrel (LFSR (Linear Feedback Shift Register) úgy történik, hogy, ha a regiszterek tartalma 0, ez az állapot marad. Nem nulla kezdőállapot után azonban véges hosszúságú periodikus jelet állít elő a kimeneten. A periódus hossz maximum 2n-1 (n a regiszterek száma).



70. ábra Az álvéletlenszám generátora

A generátornak két fajtáját tudjuk megkülönböztetni:

1) Egy irányból készítünk sokat
2) Sokból készítünk keveset



71. ábra A véletlenszámgenerátor

n	XNOR from	n	XNOR from	n	XNOR from	n	XNOR from
3	3,2	45	45,44,42,41	87	87,74	129	129,124
4	4,3	46	46,45,26,25	88	88,87,17,16	130	130,127
5	5,3	47	47,42	89	89,51	131	131,130,84,83
6	6,5	48	48,47,21,20	90	90,89,72,71	132	132,103
7	7,6	49	49,40	91	91,90,8,7	133	133,132,82,81
8	8,6,5,4	50	50,49,24,23	92	92,91,80,79	134	134,77
9	9,5	51	51,50,36,35	93	93,91	135	135,124
10	10,7	52	52,49	94	94,73	136	136,135,11,10
11	11,9	53	53,52,38,37	95	95,84	137	137,116
12	12,6,4,1	54	54,53,18,17	96	96,94,49,47	138	138,137,131,130
13	13,4,3,1	55	55,31	97	97,91	139	139,136,134,131
14	14,5,3,1	56	56,55,35,34	98	98,87	140	140,111
15	15,14	57	57,50	99	99,97,54,52	141	141,140,110,109
16	16,15,13,4	58	58,39	100	100,63	142	142,121
17	17,14	59	59,58,38,37	101	101,100,95,94	143	143,142,123,122
18	18,11	60	60,59	102	102,101,36,35	144	144,143,75,74
19	19,6,2,1	61	61,60,46,45	103	103,94	145	145,93
20	20,17	62	62,61,6,5	104	104,103,94,93	146	146,145,87,86
21	21,19	63	63,62	105	105,89	147	147,146,110,109
22	22,21	64	64,63,61,60	106	106,91	148	148,121

72. ábraA z álvéletlenszám generátor működése

Az álvéletlenszám generátor felhasználható:

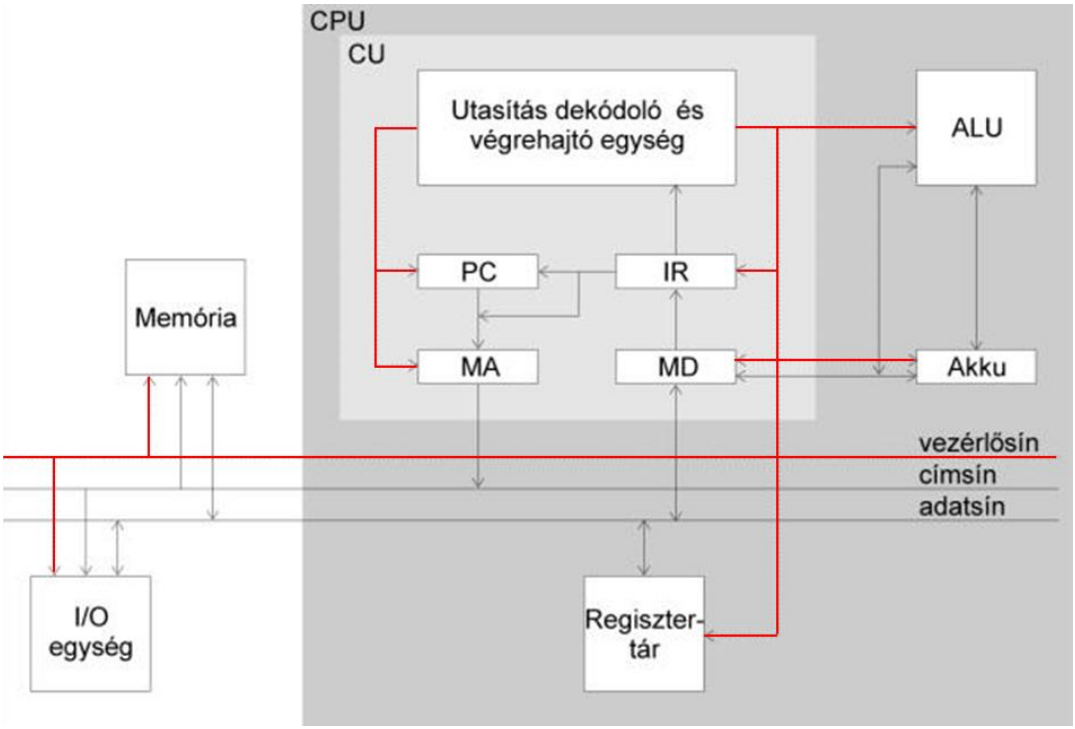
- Bitminta generálásra,

- Titkosításra,
- Hibavédelemre.

8.3.4.5. *Buszvezérlő*

Buszvezérlő: A regisztert és más adattárolókat összekötő buszrendszert irányítja. A busz továbbítja az adatokat.

CPU a hozzá csatlakoztatott memóriaegységekkel, be- és kimeneti egységekkel, regisztertárakkal ezeken keresztül tart kapcsolatot. Több párhuzamos vezeték, melyeken az adatok, a memóriák egyes rekeszeit kiválasztó címek, és egyéb vezérlőjelek utaznak A síneken általában több eszköz osztozik, de egyszerre csak egy használhatja őket.



73. ábra A vezérlő sín

8.3.4.6. *Chase*

Cache: A modern processzorok fontos része a cache (gyorsítótár). A cache a processzorba, vagy a processzor környezetébe integrált memória, ami a viszonylag lassú rendszermemória-elérést hivatott kiváltani azoknak a programrészeknek és adatoknak előzetes beolvasásával, amikre a végrehajtásnak közvetlenül szüksége lehet. A mai PC processzorok általában két gyorsítótárat használnak, egy kisebb

(és gyorsabb) első szintű (L1) és egy nagyobb másodszintű (L2) cache-t. A gyorsítótár mérete ma már megabyte-os nagyságrendű.

8.3.4.7. Memória

A rendszer működése hatékonyabb, ha a memóriában nem csak az utasítások hanem adatok is tárolhatók későbbi felhasználás céljából

- Programmemória
- Adatmemória

Lehet közös memória (Neumann architektúra) vagy fizikailag külön adat és programmemória (Harvard architektúra).

8.3.4.8. Bemenetek- Kimenetek

A bemeneti adatok fogadására

A kimeneti adatok megjelenítésére

A külvilággal kapcsolat

Műveletvégző egységek

A gyakran használt aritmetikai és logikai műveletek végrehajtását célszerű külön erre a célra tervezett egységre bízni

8.3.5. Algoritmusok megvalósítása

Műszaki rendszereket mindig valamilyen feladat megoldása érdekében építünk. A feladatmegoldás általában valamilyen algoritmus szerint történik.

- Mérésadatgyűjtés
- Adatok elemzése (pl. összehasonlítás)
- Aritmetikai, logikai műveletek végzése az adatokon
- Döntéshozatal stb.

Az információfeldolgozás menetét (programját) építjük be a rendszerbe. Erre két megoldás kínálkozik.

8.3.5.1. 1. megoldás: Huzalozott program

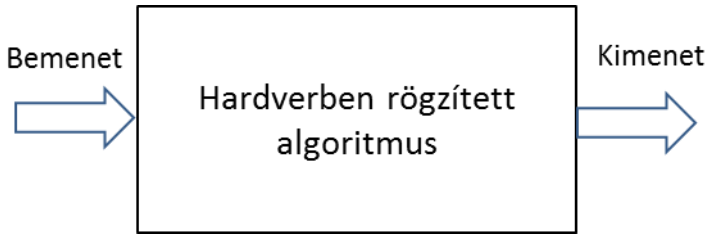
A rendszer összetevői és egymáshoz való kapcsolódásuk, sorrendiségük a hardverben fixen „behuzalozva” jelennek meg.

Előnye:

Egyes részfeladatok párhuzamosan is végrehajthatók (gyors működés)

Hátránya:

A hardver a rögzített struktúra miatt csak az adott feladat megoldására alkalmas.



74. ábra Huzalozott program

8.3.5.2. 2. megoldás: Tárolt program

Az algoritmusnak megfelelő sorrendben, előre letárolt program szerint, egy vezérlő berendezés segítségével aktivizáljuk az egyes műveletvégző egységeket

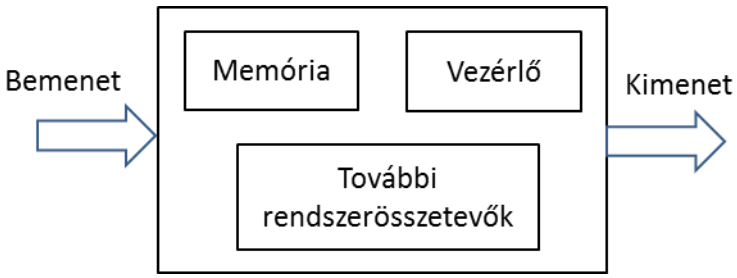
Előnye

Ha megváltoztatjuk a memória tartalmát (a programot) más-más feladatra használhatjuk

Hátránya

A részfeladatok nehezen párhuzamosíthatók (lassúbb működés)

Szekvenciális utasítás végrehajtás



75. ábra Tárolt program

8.3.6. A processzor működése

1. Az utasítás beolvasása a memóriából a processzorba: A memória címtárolójából, az AR-ból (address register - címregiszter) kerül át a processzor címtárolójába az IP-be (instruction pointer). Ezek után a memória adattároló regiszteréből, a DR-ből (data register - adatregiszter) kerülnek át az adatok a processzor adattárolójába, az IR (instruction register)-be.

2. A beolvasott utasítás dekódolása, elemzése: Az ALU az utasítás kódját értelmezi, melyből kiderül milyen műveletet kell elvégeznie, és hogy mennyi adatot kell beolvasni még ahhoz, hogy meghatározhatóak legyenek az operandusok, amelyeken a műveleteket végzi.

3. A művelet végrehajtása, mely eredménye az LR3 segédregiszterbe kerül.

4. Eredmény tárolása: az LR3 segédregiszterből vagy egy másik regiszterbe, vagy a DR-en keresztül a memóriacímre kerül.

5. A következő utasítás címének meghatározása: A szekvenciális program esetében az IP értékének megnövelésével jut el az ALU a következő utasítás címéhez. Ellenkező esetben egy regiszter tartalmazza a következő utasítás címét, melyet a processzor az IP-be ír.

8.3.6.1. Az óra és az órajel

Az óra az egész számítógép működéséhez szükséges ütemet biztosítja. Az óra magában foglal egy kvarckristályt, ami az órajel előállításához szükséges rezgés stabilitását adja. Sebességét Hertzben (Megahertzben) mérjük. Az órajel-generátor néhány száz MHz-es rezgést ad, ezért a processzor órajelének előállításához egy beállítható szorzót alkalmaznak, hogy többféle sebességű processzort is a rendszerbe lehessen építeni.

A processzor részegységei (itt a legalapvetőbb műveleteket végző részegységekre kell gondolni, tehát nem egy olyan nagy egységre, mint például az ALU.), az órajel ütemére végzik feladataikat; amikor egy részegység megkapja az órajelet egy elektronikus jel formájában, akkor elvégzi a soron következő műveletet, amikor megkapja a következő jelet, akkor a következő műveletet végzi el. Egy másodperc alatt egy mai processzor egysége több milliószor kap jelet. Az órajel sebességének így ahhoz az időhöz kell alkalmazkodnia, amennyi időbe telik egy részegységnek a rá kijelölt művelet elvégzése (különben akkor jönne a következő művelet, amikor az előző még feldolgozás alatt van, és ez érthetően problémákat okozna). Ez lényegében azt eredményezheti, hogy a processzor egységeinek a leglassúbb elem miatt kell várakozniuk. Ezt persze különféle megoldásokkal orvosolják.

Ám a műveletet nem szabad összetéveszteni az utasítással, ezek bonyolultsága miatt egy utasítás végrehajtása több órajelciklust is igénybe vehet. Az is lassító tényező, hogy a processzor az adatokat lassabban kapja, mint ahogy fel tudná dolgozni őket, ilyenkor pedig várakoznia kell.

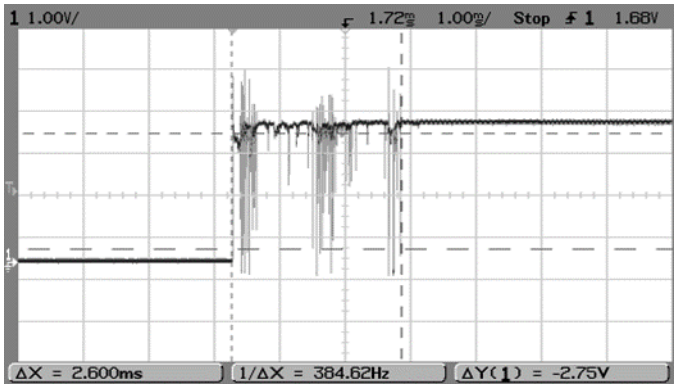
Gépi ciklusnak nevezzük azt az időt, amely alatt a számítógép egy gépi műveletet végre tud hajtani. Egy gépi ciklus általában több órajelütemből áll, az egyes utasítások végrehajtásához szükséges gépi ciklusok száma utasításonként más és más lehet.

A laborfeladat megoldása során kétféle órajelet kell előállítanunk a CPU számára, melyek között egy kiválasztó gomb (OCLK_EN) lenyomásával választhatunk:

A CPU rendes működéséhez: Az FPGA 50 MHz-es rendszer-órajeléből a Clk_divide makró segítségével 10 Hz-es órajelet állítunk elő.

A CPU teszteléséhez: Az órajel fel- és lefutó éleit nyomógombok (UP és DOWN gombok) segítségével állítjuk elő. Ehhez az órajel előállítás során egy frekvenciaosztót használunk, mert a 7 szegmenses kijelzők vezérléséhez is szükség van a megfelelő frekvencia előállítása során.

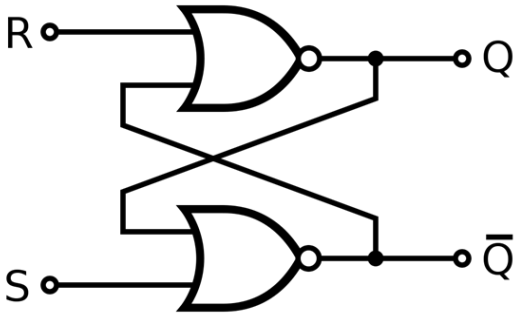
Nyomógomb pergés/prellezés/pattogás jelensége egy nagyon fontos mechanikai problémára hívja fel a figyelmünket a gyakorlaton. A mechanikus kapcsolók és nyomógombok be- illetve kikapcsolásakor az érintkezők többször érintkeznek majd eltávolodnak egymástól. A kapcsolást érzékelő nagysebességű rendszerek ezt úgy tekintik, hogy több gombnyomás/kapcsolás történt egymás után.



76. ábra A pergés

Prellmentesítés (debouncing): Ez több féle módon lehetséges:

- Speciális Hall generátoros vagy vezetőképes gumi nyomógombok alkalmazása
- Schmitt-trigger alkalmazása
- Lenyomás/kapcsolás idejének mérése, pergés idejéig változás letiltása
- Számunkra azonban a legfontosabb: RS –tároló alkalmazása

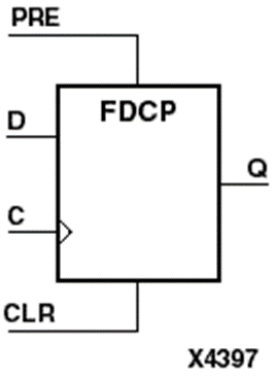


Ez a tároló ugyanis képes az érték megtartására, ezért gátolja a pergés továbbjutását a rendszerbe.

S	R	Q	$\overline{Q}$	
1	0	1	0	
0	0	1	0	S=1 és R=0 után
0	1	0	1	
0	0	0	1	S=0 és R=1 után
1	1	0	0	nem megengedett

A flip-flop tartja az értékét!

A rendszert D- tárolóval fogjuk megvalósítani:



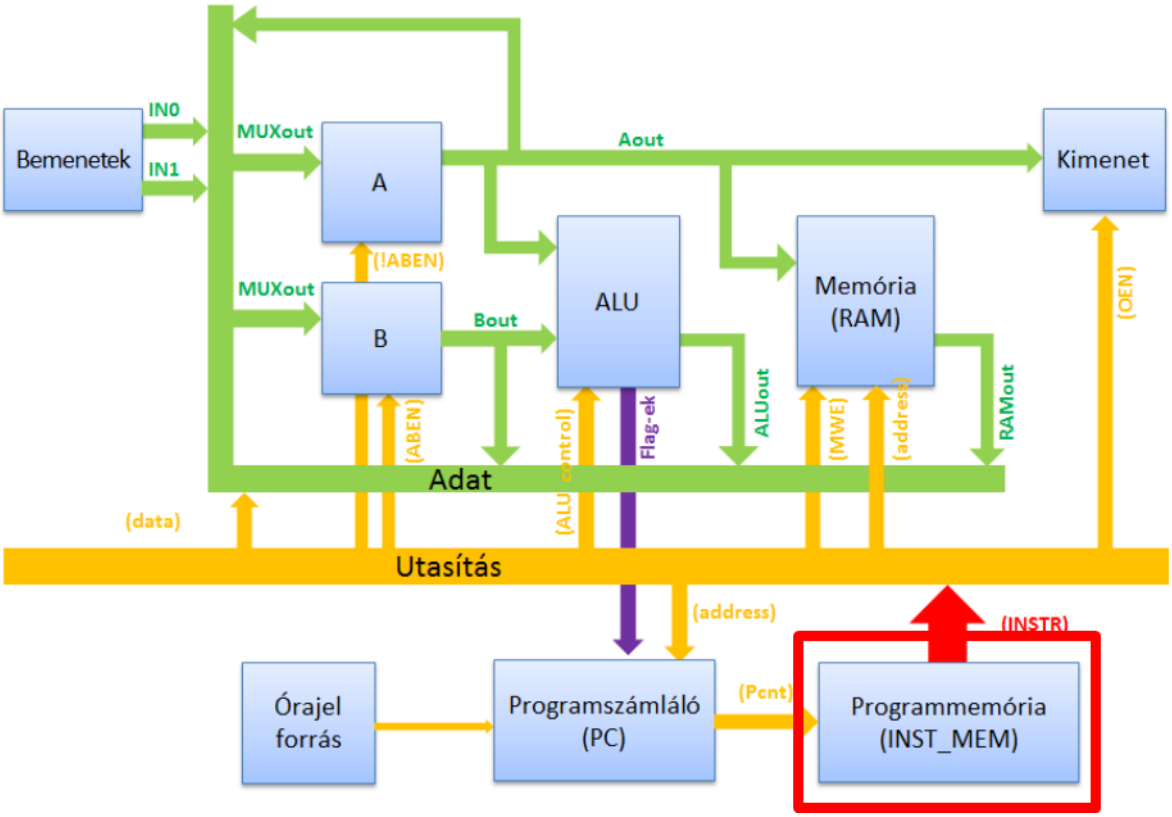
Inputs				Outputs
CLR	PRE	D	C	Q
1	X	X	X	0
0	1	X	X	1
0	0	0	↑	0
0	0	1	↑	1

77. ábra A D tároló működése

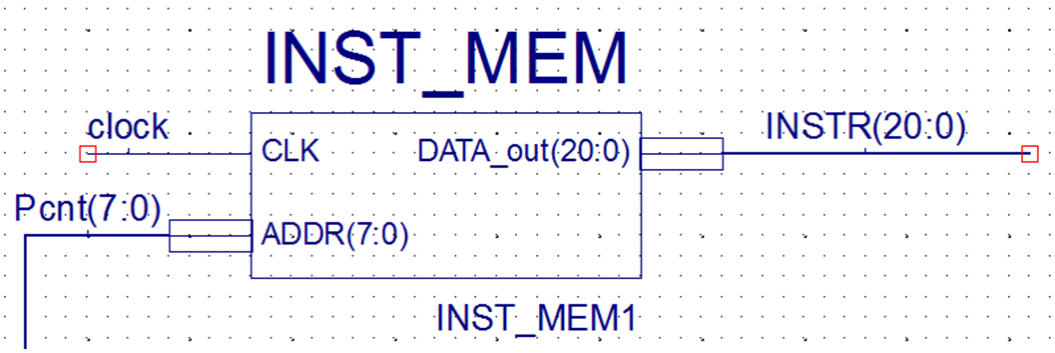
- A CLR (törlés) bemenet törli a tárolót
  (CLR=1 esetén a Q kimenet a többi bemenettől függetlenül 0 lesz)
- A PRE (preset) bemenet 1 értékre állítja a tárolót, ha nincs éppen törlés (CLR=0).
- A C bemenet felfutó élekor vizsgálja a D bemenetet
- A D bemenet 0 értéke esetén a következő C felfutó élnél a Q kimenet 0 lesz, 1 értékekor a következő C felfutó élnél Q kimenet 1 lesz.

8.3.6.2. A processzor utasításkészlete

A processzor által ismert műveletek és utasítások összességét értjük a processzor utasításkészlete alatt. Legelőször a RISC (Reduced Instructions Set Computer) utasításkészletet használták, ez leegyszerűsített, rövid utasításokat tartalmazott. Elsődlegesnek tekintette a sebességet, és az egyszerűséget. Később a CISC-et (Complex Instructions Set Computer) alkalmazták, ez már több, hosszabb utasítást tartalmazott, ám a túl sok, bonyolult utasítás nem bizonyult célravezetőnek, ezért visszatértek a RISC-hez. Ma már persze rengeteg utasításkészlet van, melyben keverednek a RISC, és a CISC irányelvei.



- A programszámláló kimenete (Pcnt busz), ami a programmemória (INST_MEM) címbemenetét adja.
- Az utasításmemória kimenetén (INSTR busz) jelennek meg az utasítások a clock órajel felfutó éle hatására.

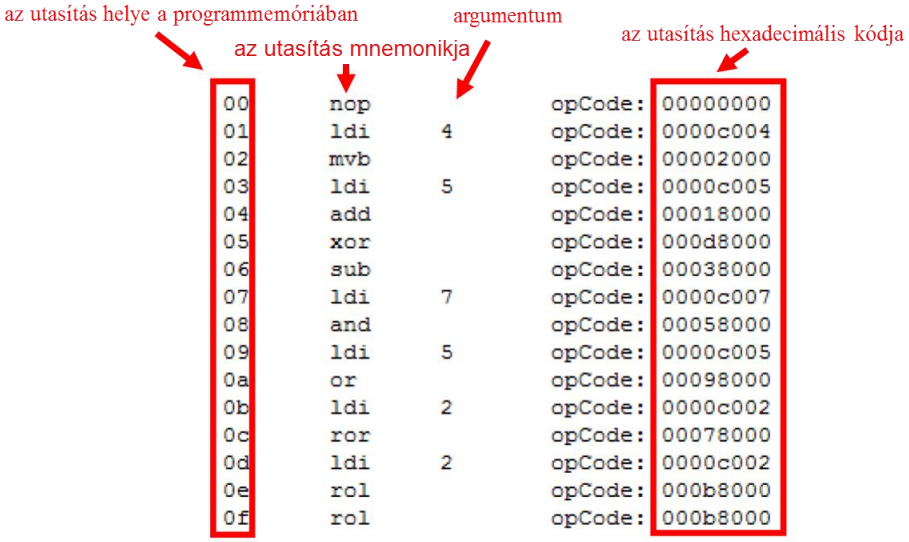


78. ábra A programmemória megvalósítása

A processzor által ismert műveletek és utasítások összességét értjük a processzor utasításkészlete alatt. Legelőször a RISC (Reduced Instructions Set Computer) utasításkészletet használták, ez leegyszerűsített, rövid utasításokat tartalmazott. Elsődlegesnek tekintette a sebességet,

és az egyszerűséget. Később a CISC-et (Complex Instructions Set Computer) alkalmazták, ez már több, hosszabb utasítást tartalmazott, ám a túl sok, bonyolult utasítás nem bizonyult célravezetőnek, ezért visszatértek a RISC-hez. Ma már persze rengeteg utasításkészlet van, melyben keverednek a RISC, és a CISC irányelvei (Pentium, Pentium MMX, SSE 3/4, 3D now!).

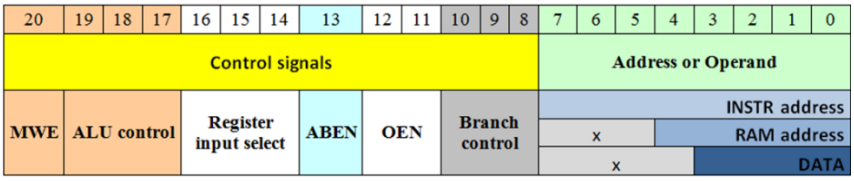
A laborgyakorlaton egy nagyon egyszerű kódot fogunk használni.



79. ábra A program kódja

A CPU utasításokat hajt végre, azaz vezérlőjelek vezérlik a processzor belső egységeit.

Argumentum: Az utasítás típusától függően lehet adat, vagy cím.



A vezérlőjelek egyszerű megadása mindenképpen fontos. Ezért „felhasználó barát módon:”

Példa:

- ‚A’ regiszterbe konstans töltése
- Konstans: k
- Vezérlőjelek: 0x0C000
- Utasításkód: 0x0C000+k

- Mnemonik: LDI k

20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	x	x	x	0	1	1	0	0	x	0	0	0	x	x	x	x	k(3)	k(2)	k(1)	k(0)
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	k(3)	k(2)	k(1)	k(0)

A példaprogramban használt utasításkészlet:

Mnemonic	Operation	Code
LDI k	k->A	0C000
MVA	B->A	4000
MVB	A->B	2000
LD a	mem(a)->A	8000
ST a	mem(a)<-A	100000
ROR	rot A	78000
ROL	rot A	B8000
ADD	A+B	18000
SUB	A-B	38000
AND	A&B	58000

OR	A|B	98000
XOR	A^B	D8000
BR a	a->PC	700
BRZ a	a->PC	100
BRNZ a	a->PC	200
BRC a	a->PC	300
BRNC a	a->PC	400
BRO a	a->PC	500
BRNO a	a->PC	600
IN d	in(d)->A	10000
OUT k	A->out(d)	1000

80. ábra Utasításkészlet

8.3.6.2.1. Az utasítások felépítése

Egy utasítást két részre lehet bontani. A vezérlőjelek az utasítás azon részei, melyek a mikroprocesszor vezérléséért felelnek (pl. az ALU egyes egységeinek vezérlőjelei). A alábbi ábrán ezeket összefoglalóan „Control signals”-nak nevezzük

20	19	18	17	16	15	14	13	12	11	
Control signals										
MWE	ALU control			Register input select			ABEN	OEN		

Az utasítás másik része az operandus, amit a vezérlőjelektől függően lehet adat vagy címként is értelmezni, vagy az utasítás szempontjából nem tartalmaz információt (Address or Operand). Az operandust a következőképpen lehet értelmezni:

INSTR address: 8 bites címet tartalmaz, az ugró utasítások esetén.

RAM address: 5 bites cím, a memória műveletek esetén. RAM-ba írás, vagy RAM-ból olvasás. A maradék 3 bit nem használt (don't care).

DATA: 4 bites adat, a konstans szám regiszterbe töltése esetén. A maradék 4 bit nem használt (don't care).

Mnemonik

A következő táblázat azokat az utasításokat tartalmazza, melyeket a CPU képes értelmezni, és végrehajtani. Vegyük példának egy konstans A regiszterbe való betöltését. Ebben az esetben az utasítás a következőképpen épül fel.

20	19	18	17	16	15	14	13	12	11	10	
0	x	x	x	0	1	1	0	0	x	0	
0	0	0	0	0	1	1	0	0	0	0	

A táblázat első sora a bitszámozást tartalmazza, a második sorában az utasítás általános alakja szerepel. Itt a lényeges vezérlőbitek értéket kaptak (0,1), az utasítás számára lényegtelen biteket x-el jelöltük. Az utolsó sorban az utasítás számára érdektelen biteket nullával helyettesítettük. A 0.-3.-ik helyi értéken egy k szám kerül megadásra, ez lesz a betöltendő konstans. Ha vesszük az utasítást hexadecimális formában 0x0C000+k és hozzáadjuk a konstans négybites számot akkor megkapjuk az utasítás kódját. Az ilyen kódok szemléletesebb jelölésére rövid utasításneveket szoktak megadni, melyeket mnemonikoknak nevezünk. Ez a rövid azonosító az operandussal együtt meghatározza az utasításkódot. A példa esetében: „LDI k ".

8.3.6.2.2. Utasításkészletek osztályozása

Az utasításkészletek osztályozhatók:

- típus szerint (CISC, RISC, VLIW, EPIC)
- bitszélesség szerint (jellemzően 8 bit, 16 bit, 32 bit, 64 bit)
- regiszterek típusa és száma szerint
- címzési módok szerint
- operandusok száma szerint
- aszerint, hogy egy vagy több utasítás egy vagy több adaton végez-e műveletet (Flynn-féle osztályozás)

- ❖ egyéb szempontok lehetnek: van-e verem, megszakításkezelés, kivételkezelés, támogatott-e a hardveres virtualizáció stb.

A komplex utasításkészlettel rendelkező számítógépek (CISC) sok olyan, specializált utasítással bírnak, melyeket a programok csak ritkán használnak. A csökkentett utasításkészletű számítógépeket (RISC) úgy egyszerűsítették le, hogy csak azokat az utasításokat valósították meg bennük, melyek gyakran előfordulnak a programokban; a ritkább műveleteket szubrutinokkal oldják meg, a ritkán használt utasítások nagyobb futási idejét pedig ellensúlyozza az egyszerűbb felépítés. Elméleti fontosságú még a minimális utasításkészletű számítógép (MISC) és az egy utasításos számítógép; ilyenek kereskedelmi forgalomba nem kerültek. További variációk a nagyon hosszú utasításszavú (VLIW) architektúrák, ahol a processzor számos utasítása van egyetlen hosszú utasításszóba belekódolva, és a VLIW utódjának tekintett utasításkészlet nélküli számítástechnika (NISC).

Gépi kód

A gépi kód vagy gépi nyelv diszkrét állításokból vagy utasításokból épül fel. Egy-egy gépi nyelvű utasítás a feldolgozó architektúrán belül az elvégzendő feladaton túl kijelölhet:

- meghatározott regisztereket aritmetika, címzési vagy vezérlési funkciókhoz
- meghatározott memóriacímeket vagy offsetet (eltolást)
- az operandusok értelmezését meghatározó címzési módokat.

A bonyolultabb műveletek ezeknek az egyszerűbb elemek kombinációjaként építhetők fel. A gépi kódú utasítások egymás után hajtódnak végre, vagy ahogy a programlefolyás-vezérlő utasítások meghatározzák.

Utasításfajták

Számos utasításkészletben megtalálhatók a ***<u>következő alapvető műveletek:</u>***

1. Adatkezelő és memóriakezelő műveletek

- értékadás (set): egy regiszter feltöltése egy konstans értékkel

- mozgatás (move): adatmozgatás memóriacímről egy regiszterbe, vagy fordítva. Célja egy regiszter tartalmának vagy egy művelet eredményének eltárolása a memóriában, vagy a tárolt adat betöltése későbbi műveletvégzés céljából.
- írás és olvasás (I/O, read-write) hardvereszközökről; az adatoknak a gépbe történő bevitelét, kiírását, illetve a külső perifériák adatforgalmának vezérlését végzik

2. Az aritmetikai-logikai egységet (ALU) vezérlő utasítások

- két regiszter értékének összeadása, kivonása, szorzása vagy elosztása (add, subtract, multiply, divide), az eredmény egy regiszterbe kerül, a státusregiszter egy vagy több bitjének (flagjének) értékét potenciálisan megváltoztatva
- bitműveletek, két regiszter megfelelő bitjei között végzett logikai és, logikai vagy, vagy egy regiszter bitjein végzett negáció.
- két regiszter értékének összehasonlítása (compare); pl. annak vizsgálata, hogy az egyik értéke kisebb-e a másikénál, netán egyenlőek

3. programlefolyás-vezérlő (vezérlésátadó) utasítások, melyekkel elérhető, hogy az utasítások végrehajtási sorrendje eltérjen azok tárolási sorrendjétől

- feltétel nélküli elágazás a program egy más címére és az ottani utasítások elvégzése
- feltételes elágazás egy másik memóriacímre, ha egy feltétel teljesül
- számított ugrás vagy indirekt elágazás egy másik címre, a következő utasítás címének (a visszatérési címnek) elmentésével (szubrutinhívás)

4. A feldolgozó egység állapotát befolyásoló utasítások a processzor speciális regisztereit módosítják, oly módon, hogy a folyamat további értelmezésére hatnak, például újraindítják vagy elaltatják a processzort, védett módba kapcsolják a feldolgozást stb.
5. Egyéb utasítások, például várakozás vagy NOP (No OPeration), azaz üres utasítás.

Komplex utasítások

Egyes platformok utasításkészlete „komplex” utasításokat is tartalmaz. Egyetlen komplex utasítás elvégzi, ami más számítógépeken sok normál utasítás feladata lenne. Az ilyen utasítások jellemzői, hogy több lépésben hajtódnak végre, számos végrehajtó egységet foglalkoztatnak vagy más módon nagyobb léptékűek az ugyanazon a processzoron futó egyszerű utasításoknál. Néhány példa komplex utasításokra:

1. egyszerre több regiszter mentése a verembe
2. nagy memóriablokkok mozgatása
3. bonyolult és/vagy lebegőpontos aritmetikai műveletek (szögfüggvények, négyzetgyökvonás stb.)
4. atomi test-and-set utasítás (memóriába írás és a korábbi érték visszaadása egy műveletben)
5. utasítások, amik az ALU értékét egy memóriából vett operandussal kombinálják (regiszterérték helyett)

Mostanában népszerű komplex utasításfajták a SIMD-, azaz Single-Instruction Stream Multiple-Data Stream (egy utasításfolyam, több adatfolyam), illetve vektorműveletek, melyek ugyanazt az aritmetikai műveletet végzik el egy időben nagy mennyiségű adaton. A SIMD segítségével lehetséges nagy vektorok vagy mátrixok manipulációja rövid idő alatt. A SIMD utasítások a hang-, kép- és videofeldolgozásban alkalmazott algoritmusok könnyű párhuzamosítását teszik lehetővé. Számos SIMD-implementáció jelent meg a piacon olyan kódneveken, mint MMX, 3DNow! vagy AltiVec.

Speciális célú processzorok, pl. GPU-k is tartalmaznak komplex utasításkészleteket. Mindazonáltal az ilyen egyedi processzorokhoz tartozó komplex utasításkészletek (és a natív assembly nyelv) gyakran nem hozzáférhetők publikusan a gyártói hardver zártságából fakadóan, így a szoftverfejlesztők szabványosított magasabb szintű nyelveken és API-kon keresztül programozhatják csak őket. Az OpenGL virtuális utasításkészlet és virtuális asszembly nyelve (ARB assembly language) és a CUDA példázzák az ilyen, natív utasításkészlet fölött létrehozott hardverabsztrakciós rétegeket.

Egy utasítás részei

Egy utasítás több mezőt tartalmazhat, melyek azonosítják az elvégzendő logikai utasítást, ezen kívül tartalmazhatnak forrás- és/vagy célcímeket, konstans értékeket. A képen a MIPS „Add Immediate” utasítás látható, ami tartalmazhat forrás- és célregisztert és egy kis konstans értéket is.

A hagyományos architektúrák utasításai a következőkből épülnek föl: az elvégzendő utasítást meghatározó opkódból (pl. „add a memóriacím tartalmát egy regiszterhez”) és nulla vagy több operandusból (regiszterek, memóriacímek vagy konstans értékek). Az operandusoknak lehetnek az értelmezésüket meghatározó különböző címzési módjaik vagy fix mezőkön helyezkednek el.

A nagyon hosszú utasításhosszú (VLIW) architektúrákon, amik közé több mikrokód-architektúra tartozik, egyetlen utasítás több egyidejűleg végrehajtandó opkódot és azok operandusát tartalmazhatja.

Egyes egzotikus utasításkészletekben az opkód mező nem létezik (ilyen a Transport Triggered Architecture – TTA – és a Forth virtuális gép), csak operandus(oka)t. Más szokatlan, zéróoperandusú utasításkészletek nem tartalmaznak operandusmezőket, ilyen néhány veremgép, köztük a NOSC

Utasításhossz

Az utasítások hossza vagy mérete az architektúrák között széles tartományban változhat: egyes mikrokontrollerek négy bitjétől a VLIW rendszerek több száz bitjéig. A személyi számítógépek, mainframe-ek és szuperszámítógépek tipikus CPU-nak utasításhossza 8–64 bit között van. Az x86 architektúrában a leghosszabb lehetséges utasítás 15 bájtos (120 bites).[1] Egy utasításkészleten belül az egyes utasítások különböző hosszúak is lehetnek. Bizonyos architektúrákon, jellemzően a legtöbb RISC gépen, az utasítások fix hosszúak, általában az adott architektúra szóhosszától függően. Más architektúrákon az utasítások különböző hosszúak lehetnek, tipikusan bájt vagy félszó többszörösei.

Megjelenítés

Egy programot alkotó utasításokat ritkán adnak meg eredeti, numerikus formájukban. A programozók általában assembly nyelven adják meg őket, vagy még gyakrabban, fordítóprogram állítja elő őket.

Tervezés

Az utasításkészletek tervezése komplex feladat. A mikroprocesszor fejlődésének két fontos állomása kapcsolódik ehhez. Az első a CISC (Complex Instruction Set Computer, komplex utasításkészletű számítógép), ami számos különböző utasítással rendelkezett. Az 1970-es években a mérnökök arra jutottak, hogy számos, ritkán használt utasítás egyszerűen elhagyható lenne. Az eredmény a RISC (Reduced Instruction Set Computer, csökkentett utasításkészletű számítógép) lett, egy kevesebb utasítást használó architektúra. Az egyszerűbb utasításkészlet előnye a magasabb elérhető órajel, csökkent processzorméret és energiafogyasztás. A komplexebb utasításkészlet ugyanakkor segítheti egyes utasítások optimalizálását, javíthatja a gyorsítótár hatékonyságát, egyszerűsítheti a programozást.

Egyes utasításkészletek fejlesztői fenntartottak egy vagy több opkódot egyfajta rendszerhívás vagy szoftvermegszakítás számára. Például a MOS Technology 6502 erre a 00H kódot, a Zilog Z80 a C7,CF,D7,DF,E7,EF,F7,FFH[2] nyolcbájtos kódot, míg a Motorola 68000 az A000..AFFFH közötti kódokat használja erre a célra.

A gyors virtuális gépek megvalósítását nagyban támogatja, ha egy utasításkészlet megfelel a Popek- és Goldberg-féle virtualizációs követelményeknek.

A zavartűrő programozásban (Immunity Aware Programming) is használt NOP-csúszdák megvalósítása egyszerűbb, ha a memória definiálatlan, „nem programozott" állapotát NOP utasításnak értelmezi a processzor.

Többprocesszoros rendszereken könnyebb megvalósítani nem blokkoló szinkronizációs algoritmusokat, ha az utasításkészlet támogat olyan jellegű kombinált utasításokat, mint a fetch-and-add, a load-link/store-conditional (LL/SC) vagy az atomi compare and swap.

- **Utasításkészlet-megvalósítások**

Egy adott utasításkészlet hardveres (mikroarchitekturális) megvalósítása sokféle formát ölthet. Ezek egymással binárisan kompatibilisek, de különböző kompromisszumokat képviselhetnek költség, teljesítmény, energiafelhasználás, méret stb. tekintetében.

Egy processzor mikroarchitektúrájának tervezésekor a mérnökök olyan (gyakran külön megtervezett) bedrótozott elektronikus áramköri blokkokat használnak fel, mint az összeadó, szorzó, számláló áramkörök, regiszterek, ALU-k stb. Valamiféle, a regiszterek közti adatmozgást leíró nyelv (register transfer language) segítségével modellezik az ISA utasításainak dekódolását és kibocsátását. Két fő módja van az utasításkészletet megvalósító vezérlőegység (Control Unit) implementálásának (bár a kettő között számos középút létezik):

1. A korai számítógépek és néhány egyszerűbb RISC számítógép „bedrótozta" a teljes utasításdekódolást és -kibocsátást (és a mikroarchitektúra többi részét is).
2. Más megoldásokban mikrokódrutinok és/vagy táblák valósítják ezt meg – tipikusan ROM és/vagy PLA csipeken (bár korábban különálló RAM-okat is használtak erre a célra).

Néhány új CPU-design is a CPU-n belüli írható RAM-ba vagy flash memóriába fordítja le az utasításkészletet (ilyen a Rekursiv processzor és az Imsys Cjipje),[3] mások egy FPGA-ra

(újrakonfigurálható számítás). Egy korábbi példa a Western Digital MCP-1600, ami különálló ROM modullal rendelkezik a mikrokód tárolására.

Egy utasításkészlet-architektúra szoftveresen is emulálható értelmező program segítségével. Természetesen az interpretáció költsége miatt ez lassabb, mint közvetlenül az emulált hardver-eredetin futtatni a programokat – kivéve, ha az emulátor nagyságrenddel gyorsabb az eredeti platfomnál. Manapság általánosan elterjedt gyakorlat, hogy a gyártók az új ISA-kat vagy mikroarchitektúrákat először szoftveres emuláció útján tesztelik és teszik a szoftverfejlesztők számára elérhetővé, még a hardveres implementáció elkészülte előtt.

Gyakran az implementáció lehetőségei visszahatnak az utasításkészletben alkalmazott utasításokra. Például az utasítás-futószalag sok megvalósításában utasításonként csak egyetlen memóriából való betöltés (load) vagy memóriába mentés (store) lehetséges, ami load-store architektúrához (RISC) vezet.

Egy másik példában az utasítás-futószalag megvalósításának korai módozatai az elágazási késleltetési réshez (delay slot) vezettek.

A nagysebességű digitális jelfeldolgozás követelményei ellentétes irányú hatást fejtenek ki – az utasítások egy bizonyos módon való megvalósításának irányába hatnak. Például, a különböző digitális szűrők kellően gyors lefuttatásához egy tipikus DSP MAC utasítását olyan Harvard-architektúrának kell megvalósítania, amely képes egy utasítást és két adatszót egyidőben betölteni és rendelkezik egy órajelciklus alatt lefutó szorzás-összeadás (multiply–accumulate) műveletre képes bináris szorzó egységgel.

Kódsűrűség

A számítástechnika hőskorában a memória nagyon drága volt, ezért a programkódnak lehetőleg kompaktnak kellett lennie, hogy elférjen a korlátozott memóriában. Egy adott feladat végrehajtására szolgáló programkód teljes hosszúsága, a „kódsűrűség” is fontos jellemzője volt bármely utasításkészletnek. A magas kódsűrűségű számítógépek gyakran komplex utasításokkal rendelkeztek parametrizált visszatérésekre, ciklusok szervezésére stb. (ezért nevezték el őket utólagosan komplex utasításkészletű, azaz CISC számítógépeknek). Azonban a tipikusabb, illetőleg gyakrabban előforduló CISC utasítások csak összekombinálnak egy egyszerű ALU utasítást (mint pl. „ADD”) egy vagy több memóriában lévő operandus elérésével (különböző címzési módok, mint direkt, indirekt, indexelt stb.) segítségével. Egyes architektúrák megengedik, hogy (az eredményt beleértve) két vagy három operandus is memóriában legyen, vagy olyan funkciók elvégzésére képesek, mint pl. az automatikus

mutató-inkrementálás. A szoftveresen megvalósított utasításkészletekben még komplexebb és hathatósabb utasítások találhatók.

A csökkentett utasításkészletű számítógépek (RISC) széles körű elterjedése a gyorsan növekvő méretű memóriaalrendszerek időszakára esett. A RISC-ek feláldozták a kódsűrűséget a hardveres megvalósítás egyszerűsítése érdekében, így a teljesítményt az órajel-frekvencia és a regiszterek számának megnövelésével próbálták elérni. A RISC utasítások tipikusan egyetlen műveletet hajtanak végre, például regiszterek összeadását („ADD") vagy egy memóriacím tartalmának regiszterbe betöltését („LOAD"); általában fix utasításhosszal dolgoznak, míg egy tipikus CISC utasításkészletben számos olyan utasítás van, ami rövidebb ennél a fix hosszúságnál. A fix hosszúságú utasítások kezelése több okból is egyszerűbb a változó hosszúságú utasításokhoz képest (ennek egyik oka, hogy nem kell külön odafigyelni az utasítások gyorsítótár-vonalon vagy virtuálismemória-laphatáron való átnyúlására), így valamivel könnyebb a sebességre való optimalizálás. Mivel azonban a RISC számítógépek adott feladathoz általában több és nagyobb méretű utasítást (így nagyobb méretű kódot) igényelnek, ezért kevésbé optimálisan használják ki a rendelkezésre álló busz-sávszélességet és gyorsítótár-memóriákat.

A minimális utasításkészletű számítógépek (MISC) olyan veremgépek, ahol csak néhány (16-64) különböző utasítás létezik, így több utasítás kódja elfér egyetlen gépi szóban. Az ilyen architektúrák vezérlői egyszerűen, kevés kapuval megvalósíthatók, így könnyen előállíthatók FPGA vagy többmagos formában. A kódsűrűség a RISC-évél összemérhető, a megnövelt utasítássűrűséget ellensúlyozza, hogy a primitívebb utasításokból több darabra van szükség a feladat megoldásához.

A kódsűrűség javításának egyik lehetséges eszköze a futtatható állományok tömörítése (EXE-tömörítés). Az ezzel kapcsolatos kihívásokkal és korlátokkal matematikailag a Kolmogorov-bonyolultság területe foglalkozik.

Az operandusok száma

Az utasításkészletek csoportosíthatók az utasításokban explicit módon meghatározott operandusok maximális száma szerint.

(A következő példákban a, b és c (közvetlen vagy számított) memóriacímek, míg reg1 és a továbbiak regisztereket jelentenek.)

1. 0-operandusú (zérócímű gépek), az úgynevezett veremgépek: minden aritmetikai művelet a verem felső egy vagy két címének felhasználásával megy végbe: push a,

push b, add, pop c. A veremgépeknél a „0-operandusú” vagy „zérócímű” kifejezések az aritmetikai utasításokra vonatkoznak, de valamennyi utasításra nem, hiszen 1-operandusú push és pop utasításokkal érik el a memóriát.

2. 1-operandusú (egycímű gépek), az úgynevezett akkumulátorgépek közé tartoznak a hőskor számítógépei mellett sok kis mikrokontroller is: a legtöbb utasítás egyetlen jobb operandust határoz meg (egy konstanst, regisztert vagy memóriacímet), az implicit bal operandus pedig az akkumulátor (és a célcím, ha az utasításnak van ilyenje): load a, add b, store c. Idetartozó kategória még a praktikus veremgépeké, melyek az aritmetikai utasításoknál gyakran megengednek egyetlen explicit operandust: push a, add b, pop c.
3. 2-operandusú – számos CISC és RISC gép tartozik ehhez a kategóriához:

- CISC – gyakori a load a,reg1; add reg1,b; store reg1,c megoldás olyan gépeken, melyek utasításonként egy memóriaoperandusra vannak korlátozva
- CISC – move a->c; add c+=b.
- RISC – explicit memóriakezelésre van szükség, így az utasítsok: load a,reg1; load b,reg2; add reg1,reg2; store reg2,c

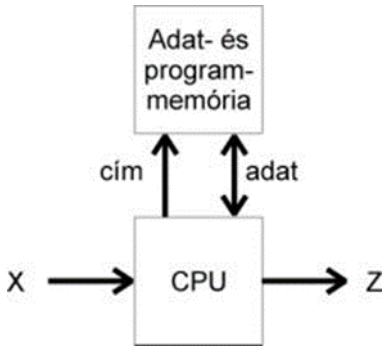
4. 3-operandusú, az adatok jobb újrahasznosítását lehetővé tevő:

- CISC – vagy egyetlen utasítás lesz: add a,b,c, vagy jellemzőbben: move a,reg1; add reg1,b,c, hiszen a legtöbb gép legfeljebb két memóriaoperandust enged meg.
- RISC – az aritmetikai műveletek csak regisztereket használhatnak, ezért explicit, kétoperandusú load/store műveletekre van szükség: load a,reg1; load b,reg2; add reg1+reg2->reg3; store reg3,c; a 2-operandusútól és az 1-operandusútól eltérően ez mind a három felhasznált értéket meghagyja az a, b és c regiszterekben.[4]

5. több operandusú – egyes CISC gépek háromnál több operandusú (regiszter vagy memória) címzési módokat is ismernek, mint a VAX „POLY” polinomkiértékelő utasítása.

Mivel egy háromoperandusú utasítás három regiszterének kódolásához sok bitre van szükség, a 16 bites RISC processzorok kivétel nélkül kétoperandusú gépek, ilyenek pl. az Atmel AVR, a TI MSP430 és az ARM Thumb egyes verziói. A 32 bites RISC processzorok általában háromoperandusú gépek, ilyenek a Power Architecture, a SPARC architektúra, a MIPS, az ARM architektúra és az AVR32 architektúra processzorai. Az utasítások explicit módon meghatároznak egy vagy több operandust

(regisztereket, memóriacímeket vagy konstans értékeket). Ezen túl egyes utasítások egy operandust vagy mindkettőt implicit módon tartalmazzák, implicit regiszterként, vagy pl. az által, hogy a verem tetejéhez nyúlnak hozzá. Ha az operandusok egy része implicit módon meghatározott, az utasítás operandusainak száma kisebb lehet a művelet matematikai értelemben vett aritásánál (változószám). Ha a céloperandus explicit módon meghatározza az eredmény címét, az utasítás operandusainak száma nagyobb lehet a művelet aritásánál. Az operandusok tehát vagy az utasítások opkódjában, vagy az utasítást követő értékek vagy címek formájában vannak meghatározva. Egyes utasításkészletekben különböző utasításokhoz különböző számú operandus tartozhat.



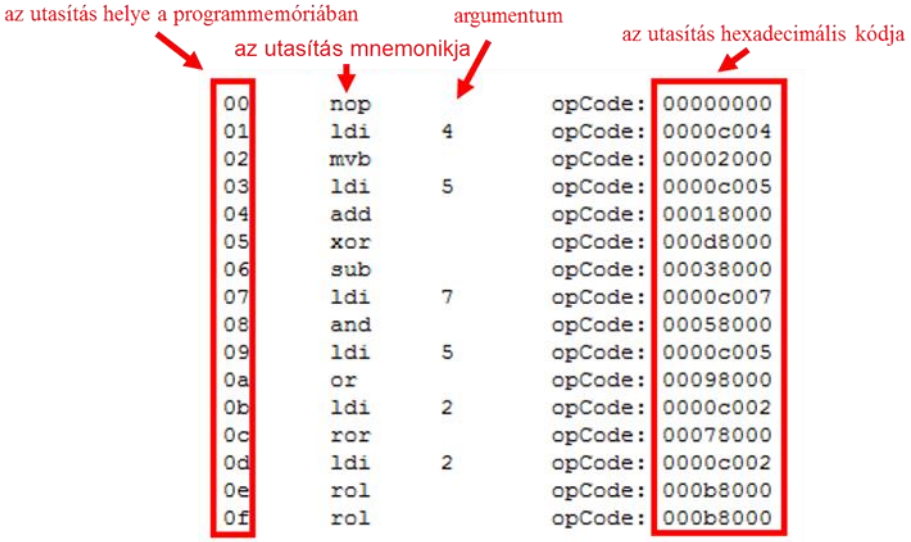
Az utasítások alapján előállítja a processzoron belüli és a processzorhoz kapcsolt külső egységek működéséhez szükséges vezérlőjeleket

8.3.6.2.3. A CPU teljes utasításkészlete

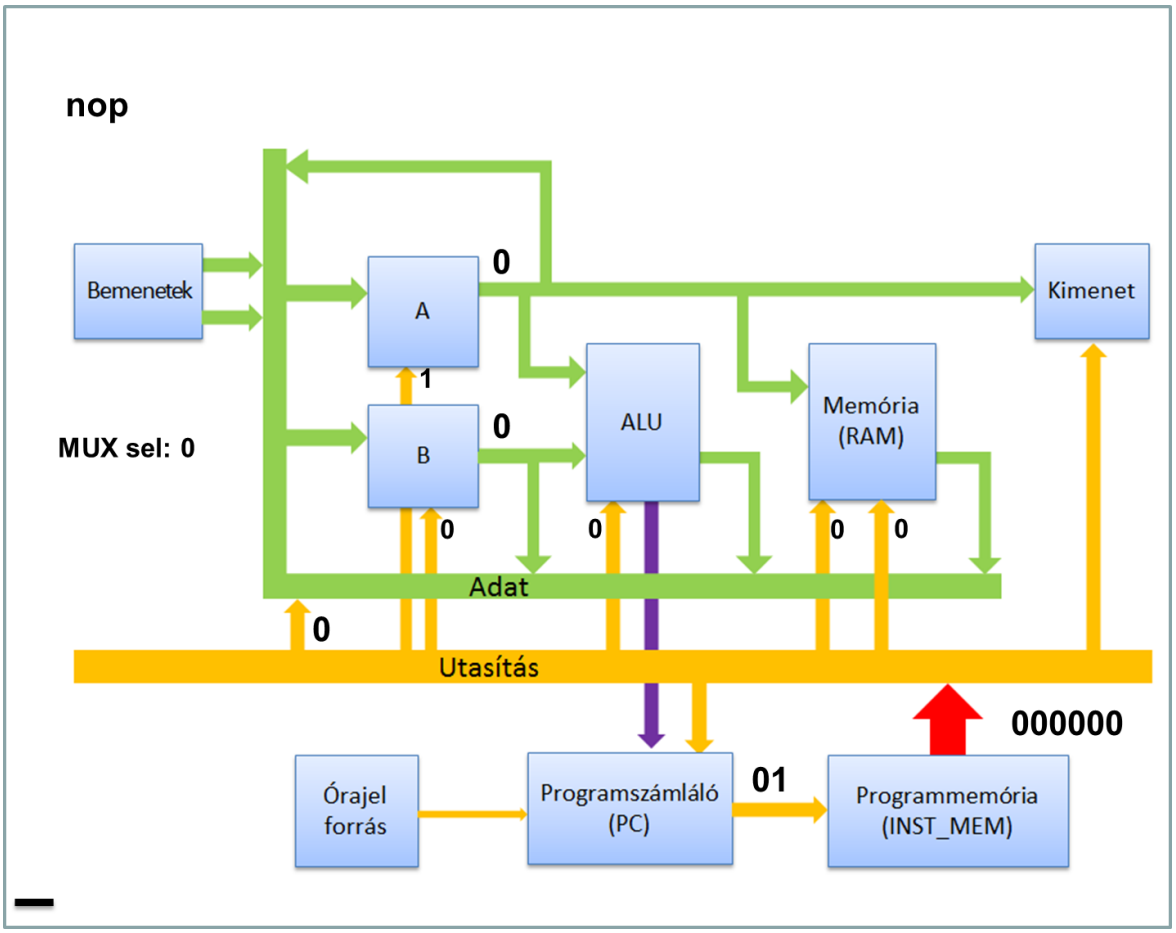
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Mnemonic	Operation	Code
MWE	**ALU control**			**Register input select**			**ABEN**	**OEN**		**Branch control**			**Address or Operand**										
mwe	S2	S1	S0	Aen2	Aen1	Aen0	Ben	Oen	Onb	Br2	Br1	Br0	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0			
0	x	x	x	0	1	1	0	0	x	0	0	0	x	x	x	x	k(3)	k(2)	k(1)	k(0)	LDI k	A=k	0C000
0	x	x	x	0	0	1	0	0	x	0	0	0	x	x	x	x	x	x	x	x	MVA	A=B	4000
0	x	x	x	0	0	0	1	0	x	0	0	0	x	x	x	x	x	x	x	x	MVB	B=A	2000
0	x	x	x	0	1	0	0	0	x	0	0	0	x	x	x	a(4)	a(3)	a(2)	a(1)	a(0)	LD a	A=mem(a)	8000
1	x	x	x	0	0	0	0	0	x	0	0	0	x	x	x	a(4)	a(3)	a(2)	a(1)	a(0)	ST a	mem(a)=A	100000
0	0	1	1	1	1	0	0	0	x	0	0	0	x	x	x	x	x	x	x	x	ROR	A=rot A	78000
0	1	0	1	1	1	0	0	0	x	0	0	0	x	x	x	x	x	x	x	x	ROL	A=rot A	B8000
0	0	0	0	1	1	0	0	0	x	0	0	0	x	x	x	x	x	x	x	x	ADD	A=A+B	18000
0	0	0	1	1	1	0	0	0	x	0	0	0	x	x	x	x	x	x	x	x	SUB	A=A-B	38000
0	0	1	0	1	1	0	0	0	x	0	0	0	x	x	x	x	x	x	x	x	AND	A=A&B	58000
0	1	0	0	1	1	0	0	0	x	0	0	0	x	x	x	x	x	x	x	x	OR	A=A|B	98000
0	1	1	0	1	1	0	0	0	x	0	0	0	x	x	x	x	x	x	x	x	XOR	A=A^B	D8000
0	x	x	x	0	0	0	0	0	x	1	1	1	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	BR a	PC=a	700
0	x	x	x	0	0	0	0	0	x	0	0	1	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	BRZ a	PC=a	100
0	x	x	x	0	0	0	0	0	x	0	1	0	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	BRNZ a	PC=a	200
0	x	x	x	0	0	0	0	0	x	0	1	1	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	BRC a	PC=a	300
0	x	x	x	0	0	0	0	0	x	1	0	0	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	BRNC a	PC=a	400
0	x	x	x	0	0	0	0	0	x	1	0	1	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	BRO a	PC=a	500
0	x	x	x	0	0	0	0	0	x	1	1	0	a(7)	a(6)	a(5)	a(4)	a(3)	a(2)	a(1)	a(0)	BRNO a	PC=a	600
0	x	x	x	1	0	d	0	0	x	0	0	0	x	x	x	x	x	x	x	x	IN d	A=in(d)	10000
0	x	x	x	0	0	0	0	1	d	0	0	0	x	x	x	x	x	x	x	x	OUT d	out(d)=A	1000

8.3.6.2.4. Az írt program működése

Az írt program működését vesszük most cmost végig – soronként.

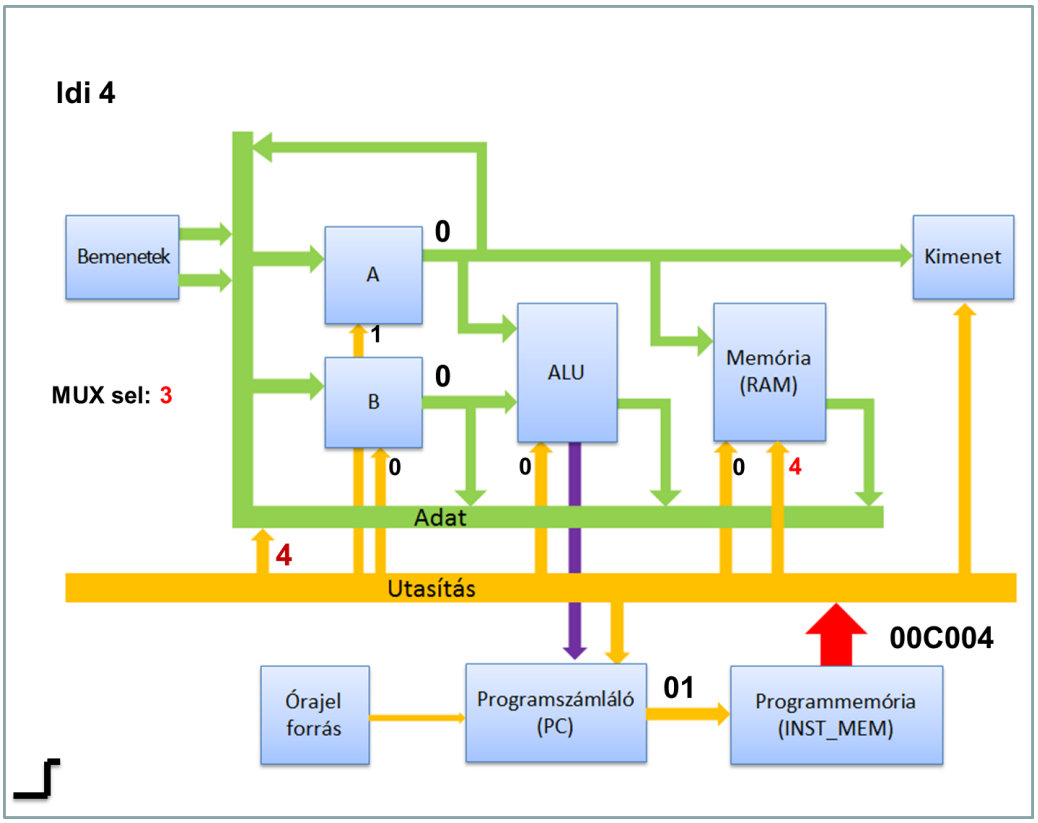


A nop utasítás kinullázza a rendszert (no operation) az új feladat előtt.



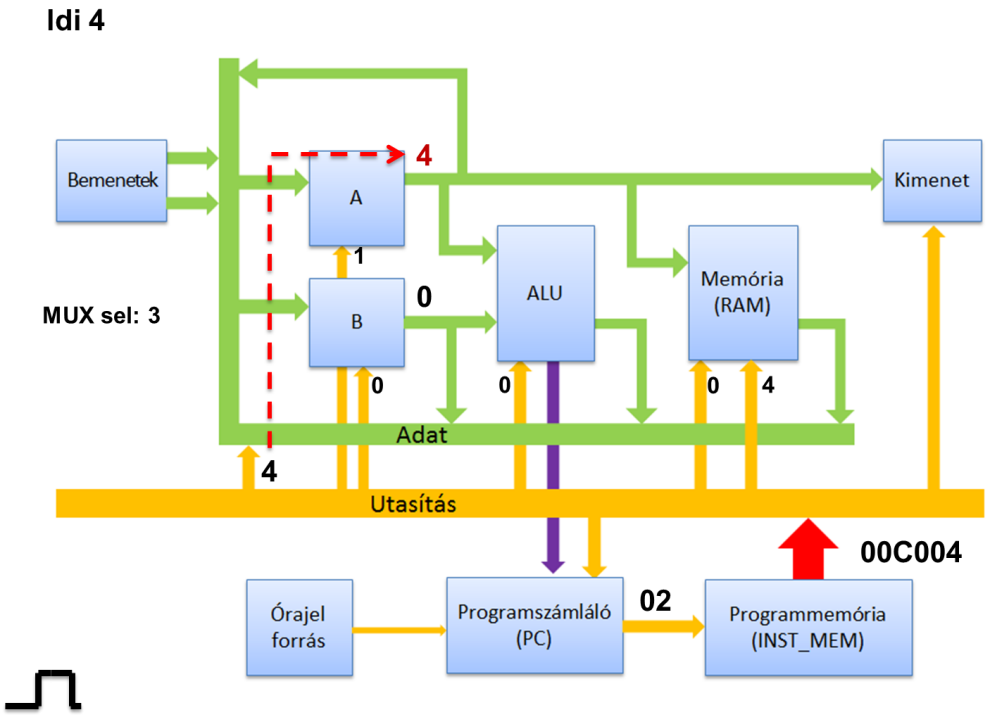
81. ábra 1. lépés : NOP: No Operation

Ezután az órajel felfutó élére beolvasunk egy számot (4). Ezt eltároljuk a memóriában, ill. eljuttatjuk a CPU akkumlátorába – gyorsan elérhető regisztertárolójába (A).



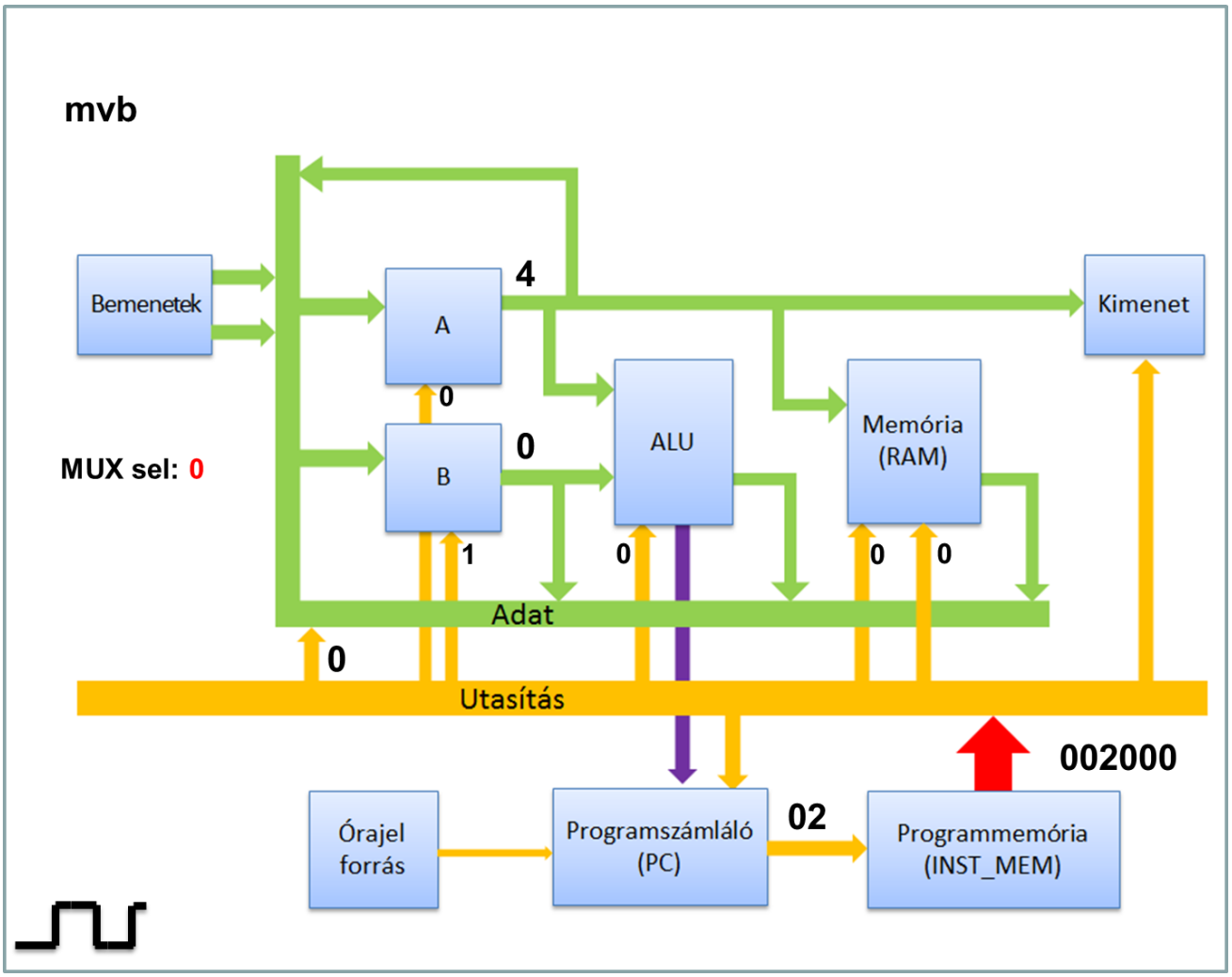
82. ábra Első utasítás: 4 beolvasása

Ezután elvégezzük a tárolást a regiszterben. ill. a programszámláló a lefutó élre ugrik egyet: jöhet majd a második utasítás.

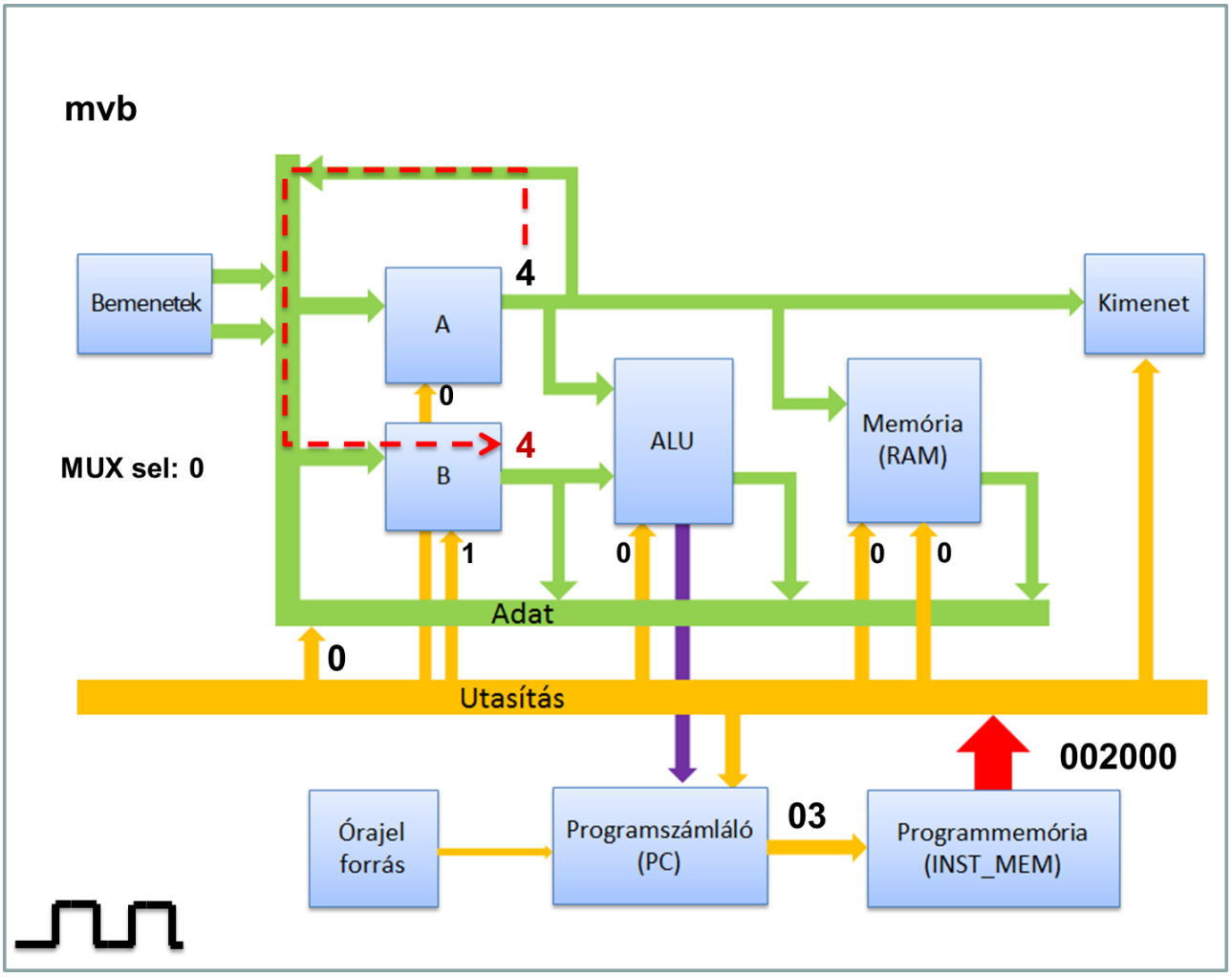


83. ábra Első utasítás: a 4 eltárolása a regiszterben.

A következőkben a tárolt értéket áthelyezzük a másik regiszterbe.

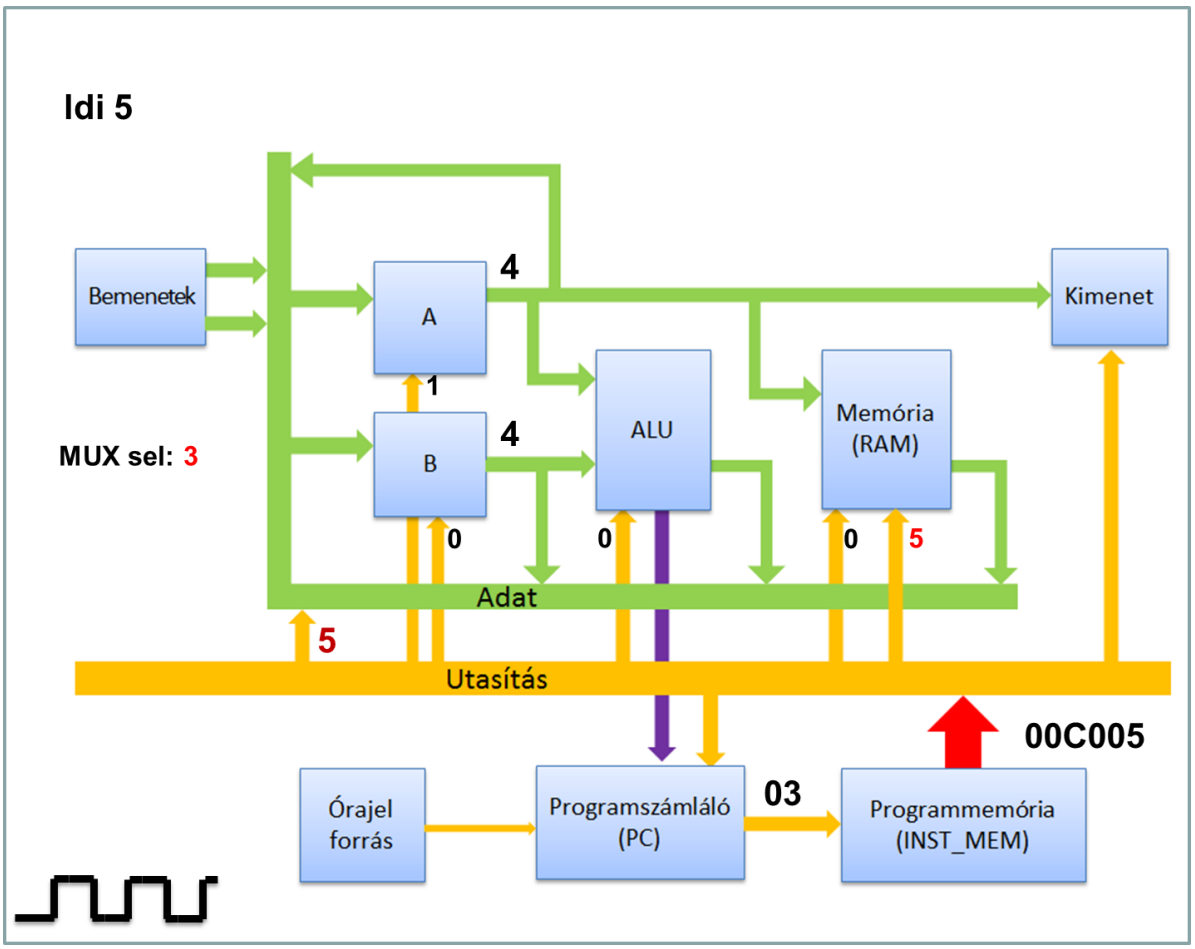


84. ábra Tárolás

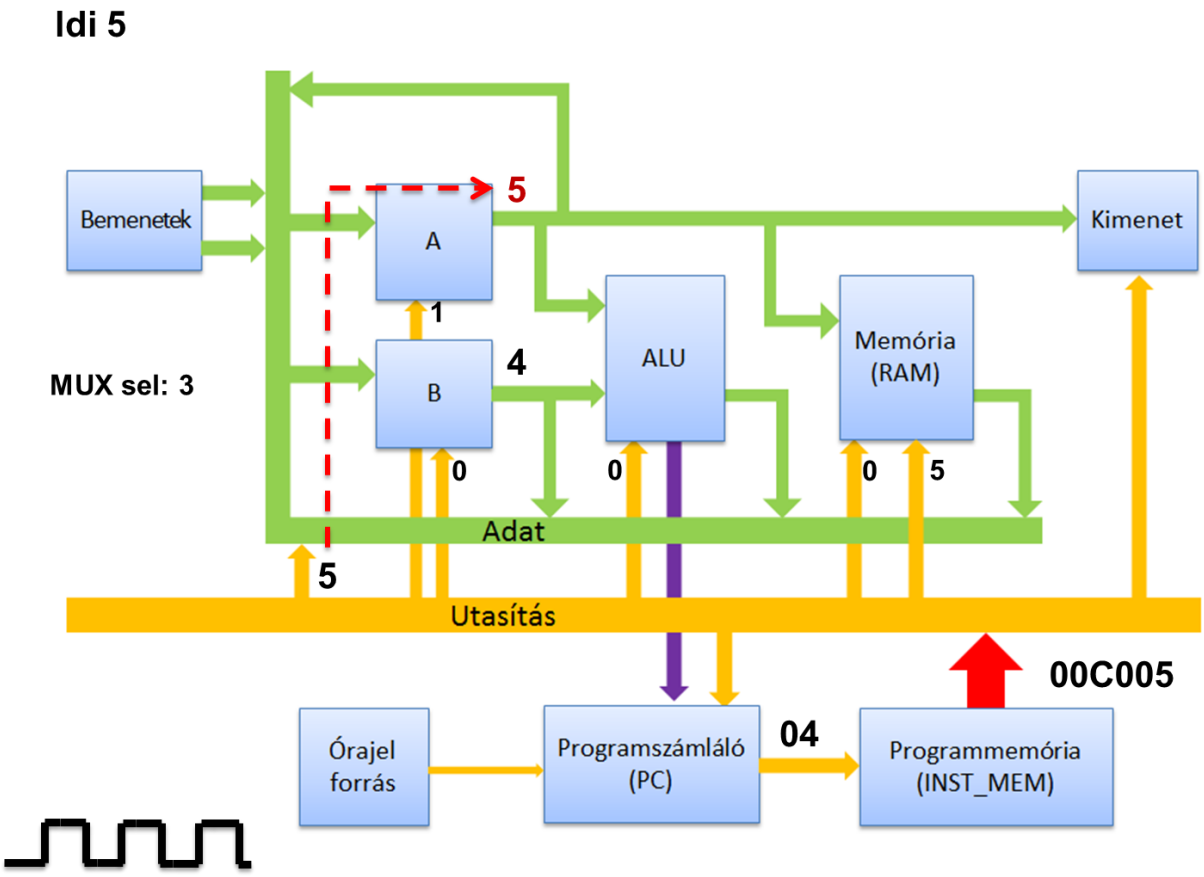


85. ábra A tárolás

A következőkben újra beolvasunk egy értéket (5) és ezt is eltároljuk a regiszterben.

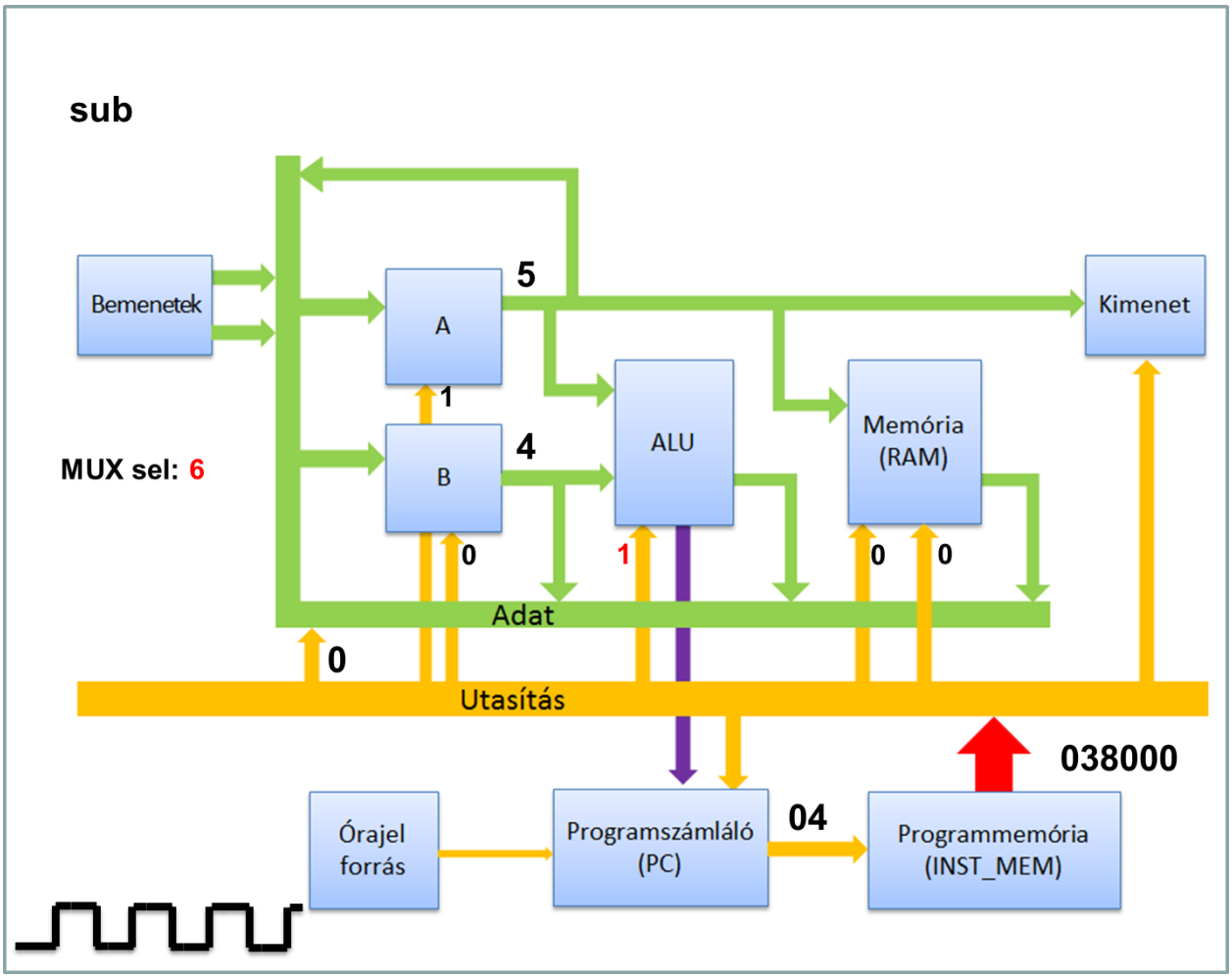


86. ábra Következő érték beolvasása

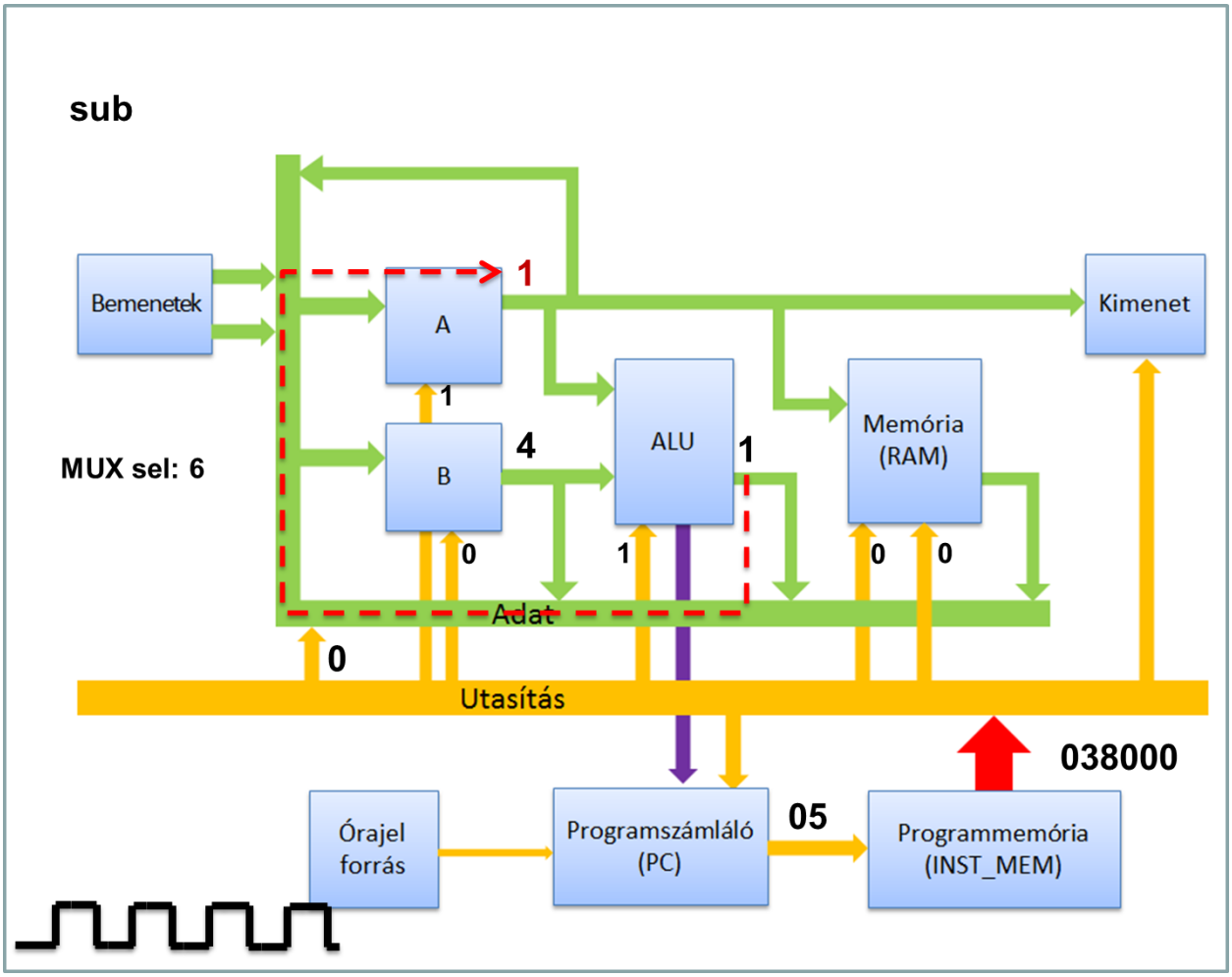


87. ábra Következő érték tárolása

A két értéket ezután kivonjuk egymásból.

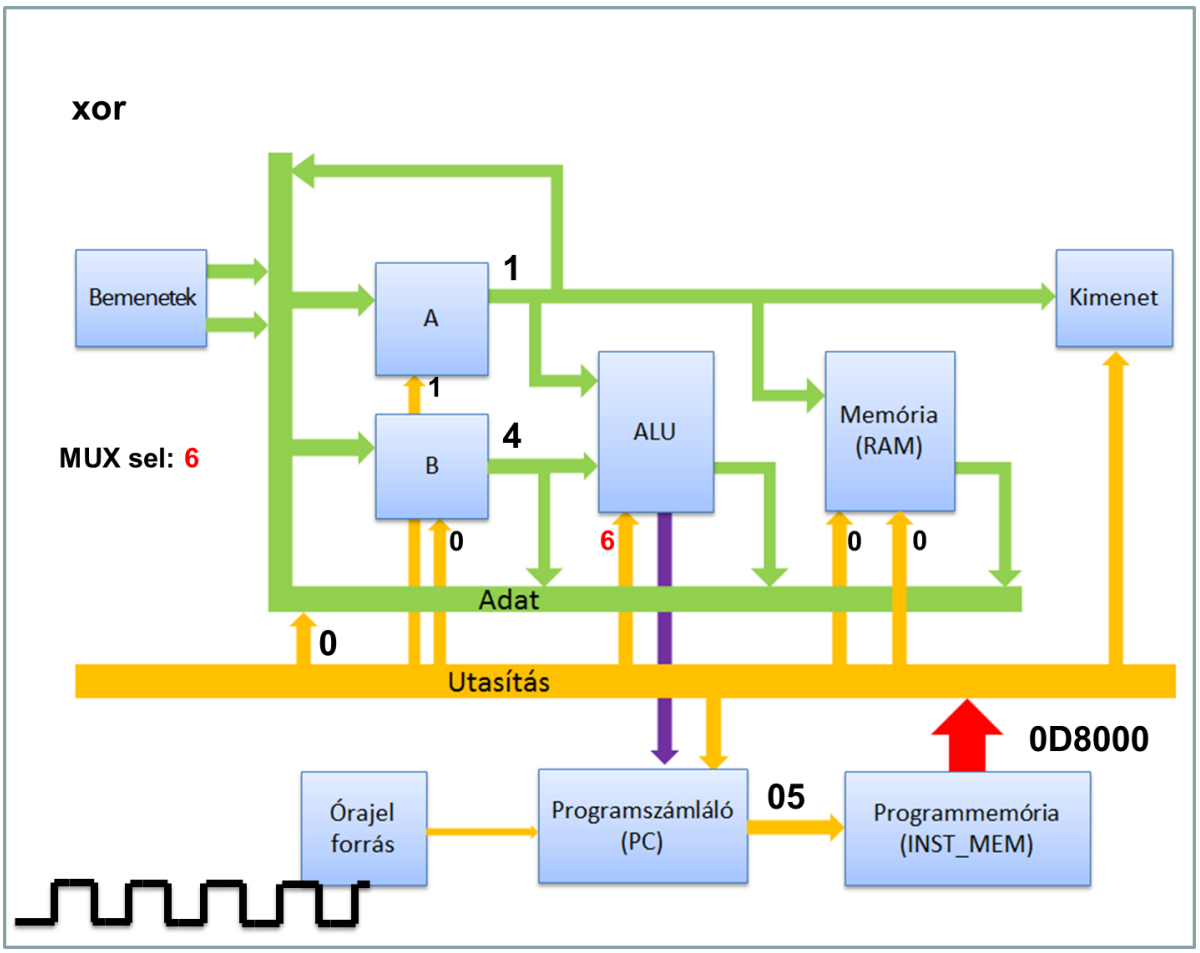


88. ábra A kivonás

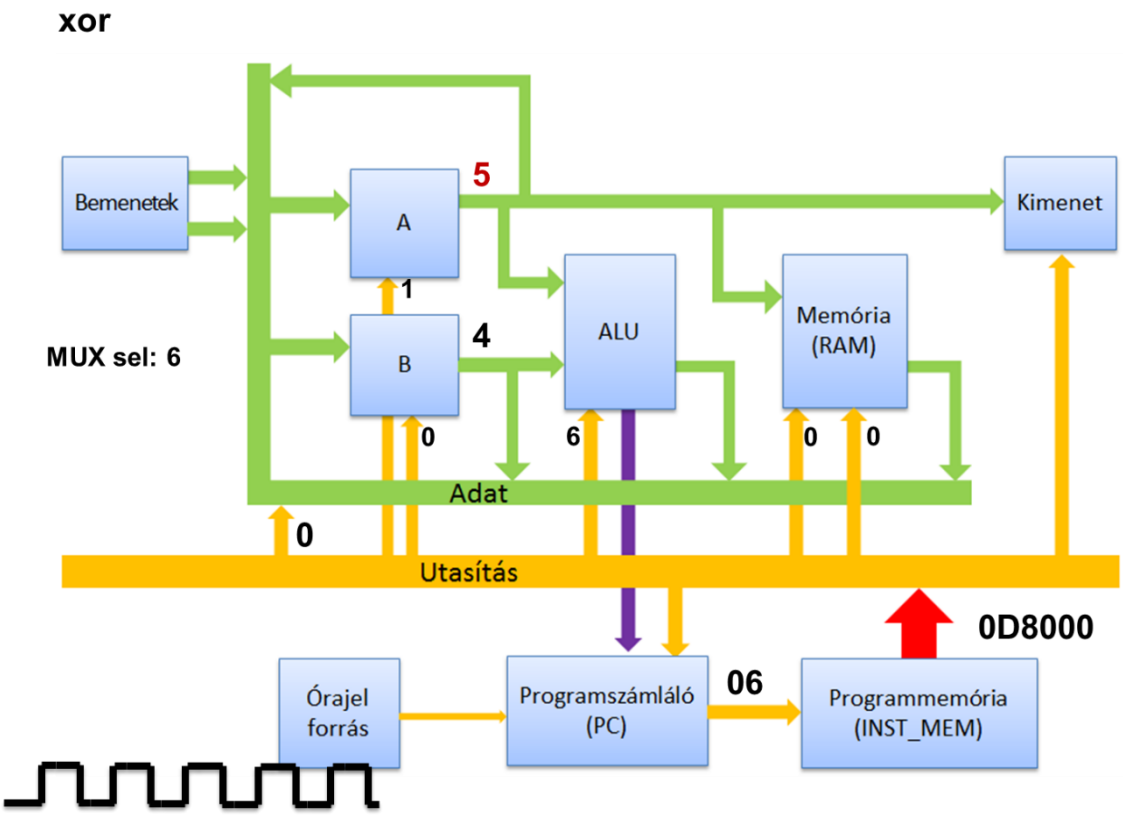


89. ábra Az ALUban elvégzett kivonás eredményének átvezetése

A meglévő értékeinkkel végzünk XOR műveletet.

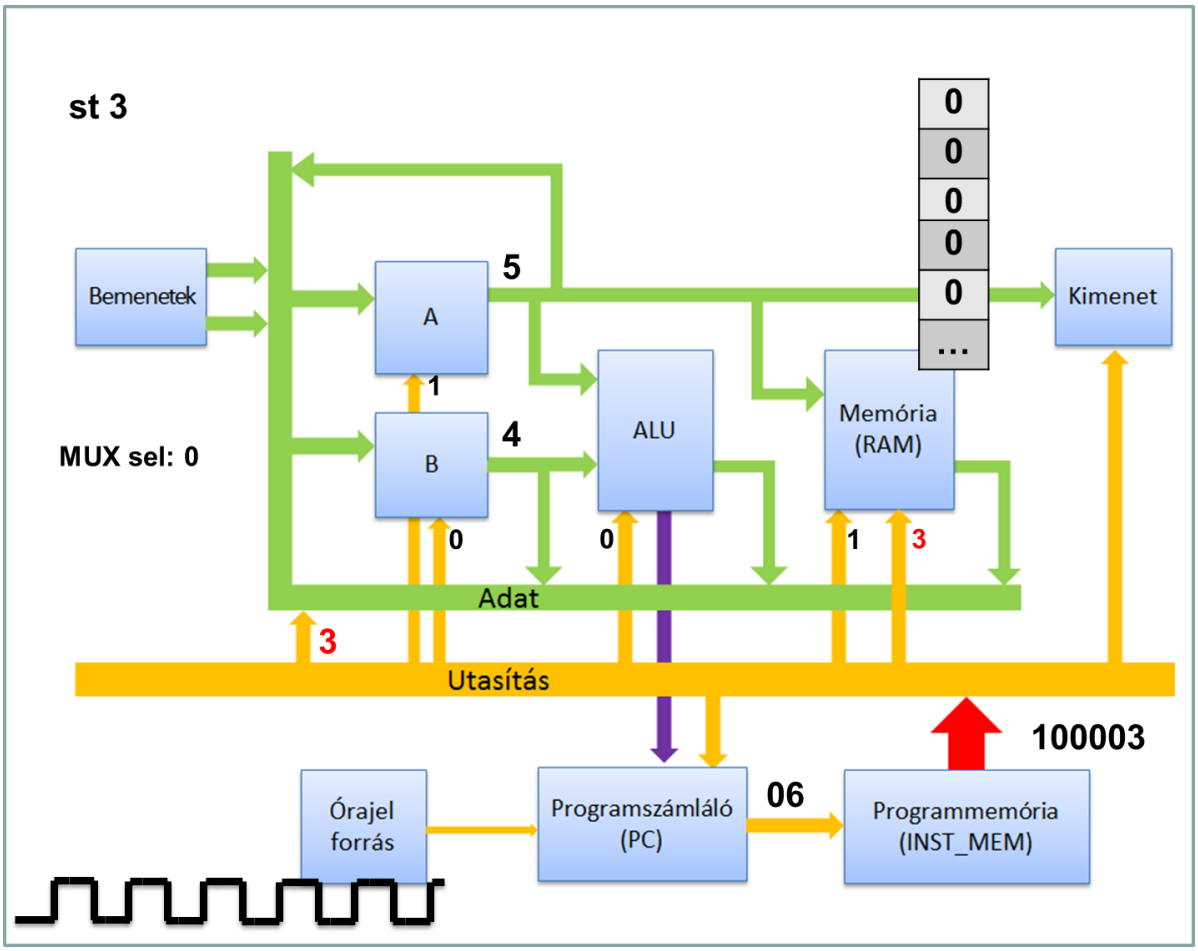


90. ábra Xor Művelet elvégzése

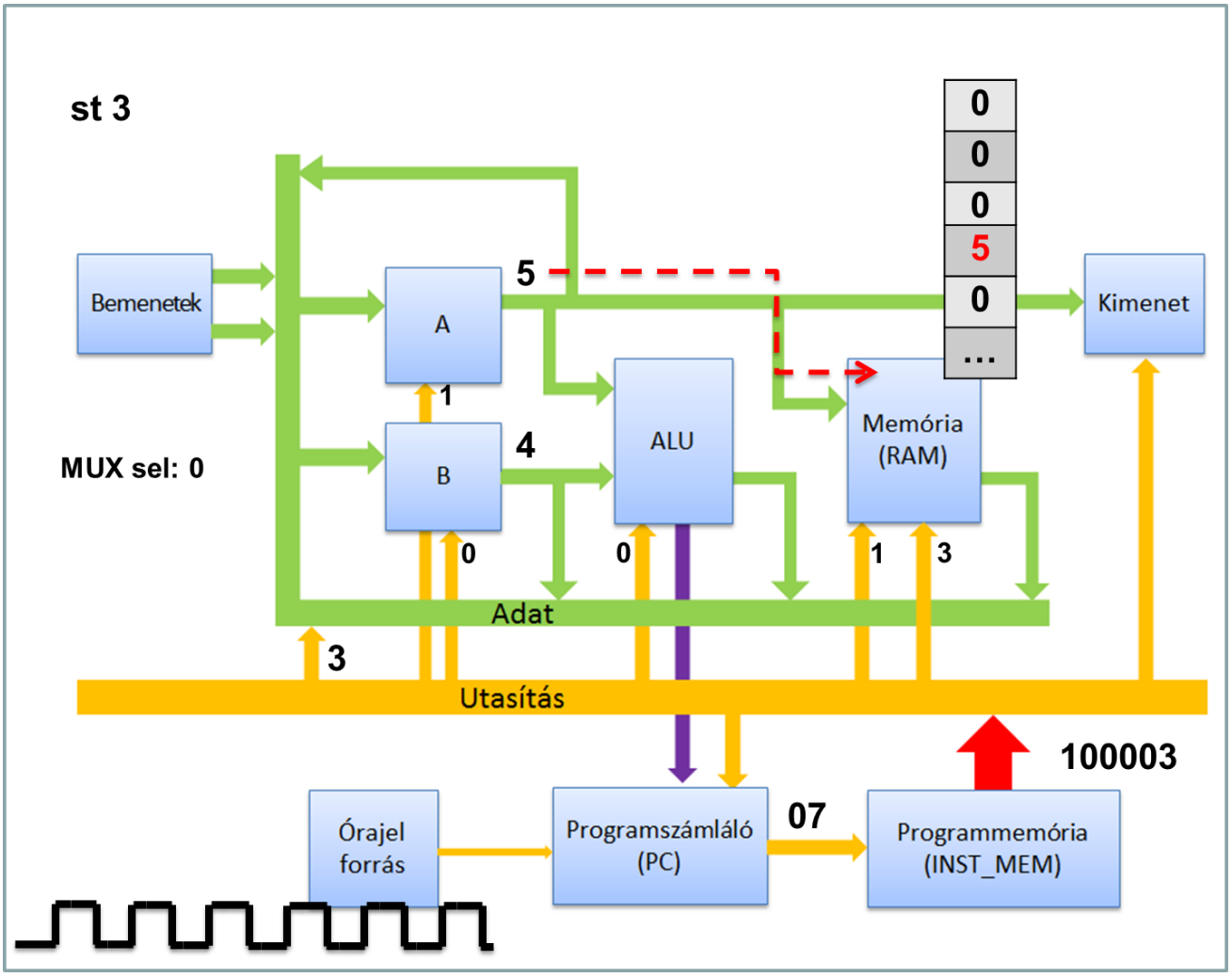


91. ábra A xor művelet elvégzése

A kapott eredményt eltároljuk a memőtriában (még nem a külsőbe, csak CPU RAM-jában).



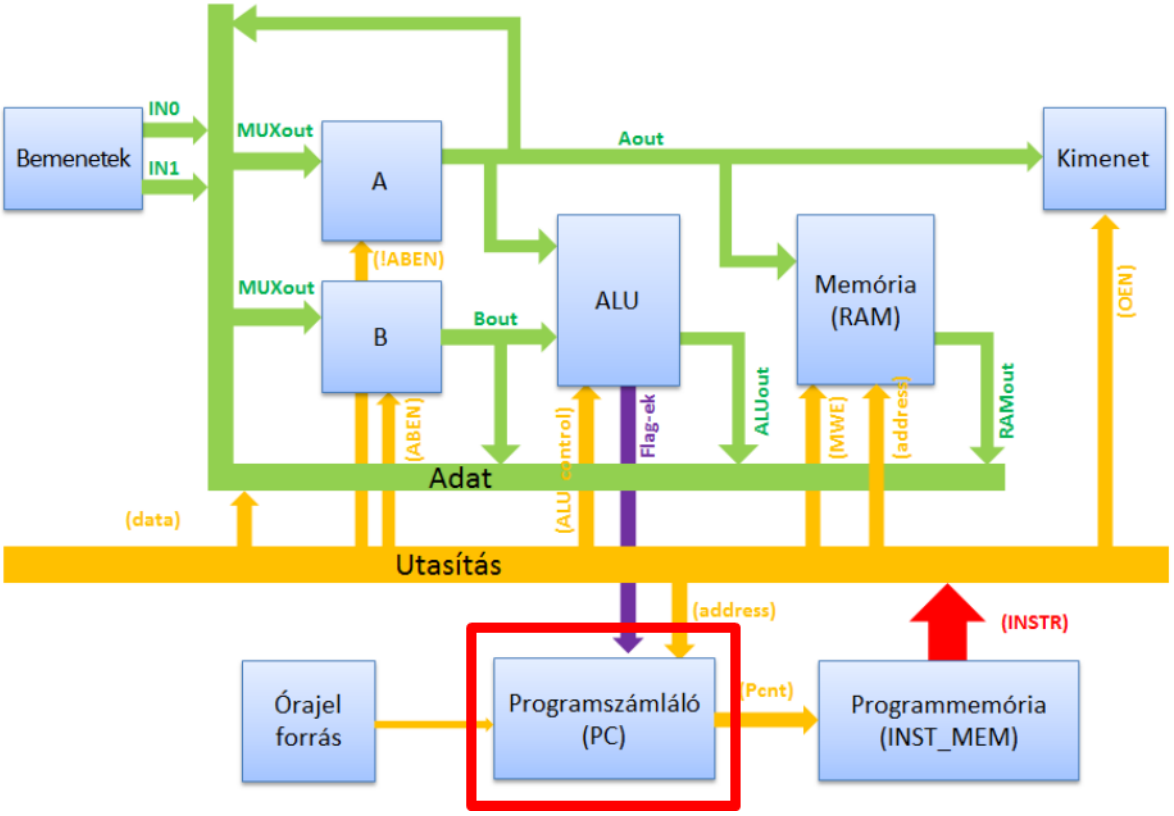
92. ábra Az eredmény tárolása : memória címzés



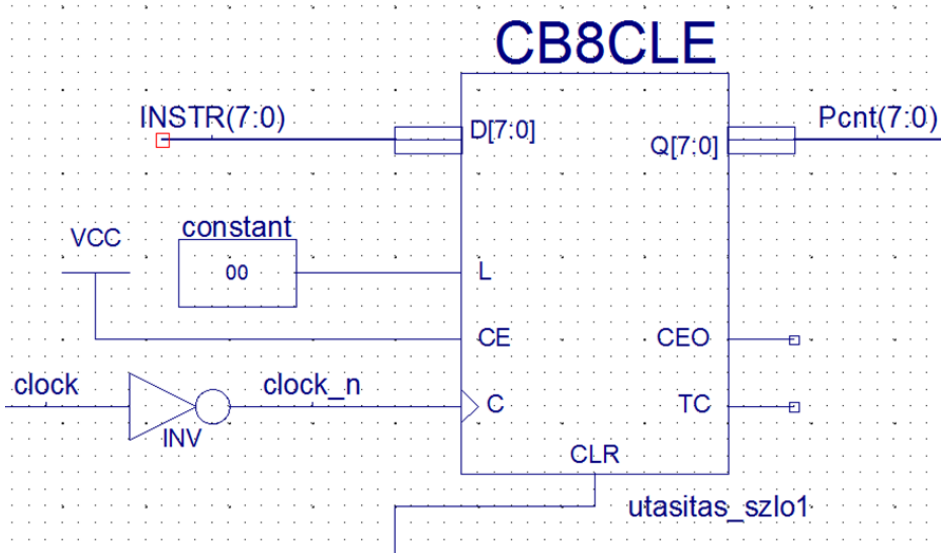
93. ábraAz eredmány tárolása

8.3.6.2.5. A programszámláló:

A programszámláló a program sorait számolja, tartja nyilván.



- 8 bites, tehát max. 256 sora lehet a programnak.
- Az órajel lefutó éle vezérli
- Az ugró utasításokkal egyelőre nem foglalkozunk, ezért az L (load) lábat 0-ra kötjük. Ekkor a Q kimenet a D bemenettől függetlenül minden órajelnél eggyel növekszik. (CLR=1 hatására nullázódik!)

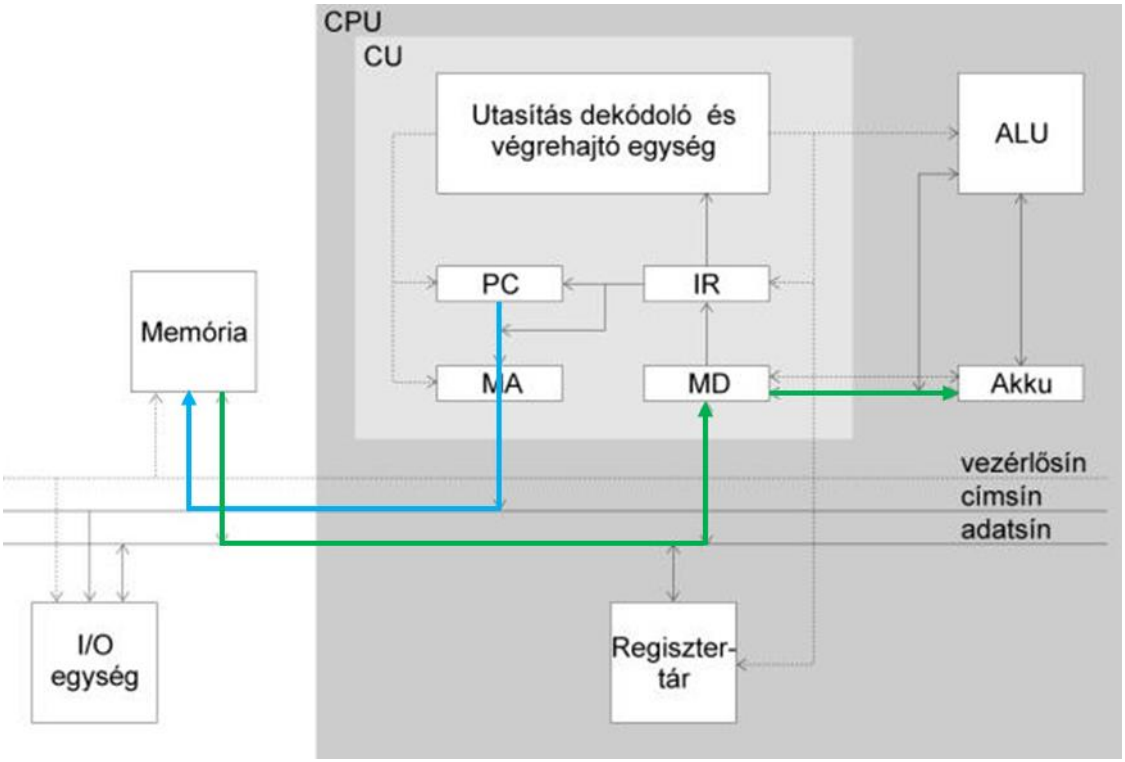


94. ábraA programszámláló

8.3.6.2.6. A CPU működése (példa)

Pl. Adat beírása az akkumulátor regiszterbe

1) lépés: A PC-ből az MA-n keresztül a memória bemeneteire jut az utasítás címe. A memória adatvezetékein megjelenik a rekesz tartalma (vagyis a műveleti kód), az MD-n keresztül az IR-be kerül
2) lépés: Az utasítás dekódoló és végrehajtó egység beolvassa a műveleti kódot, és értelmezi azt
3) lépés: A PC értéke eggyel nő (így az operandus címére mutat)
4) lépés: Az operandus címe a PC-ből a memória bemeneteire jut, majd tárolt érték az MD-n keresztül az akkuba kerül
5) lépés: A PC értéke megint eggyel nő, vagyis a következő utasításra mutat: elkezdődhet annak a végrehajtása



8.3.7. A processzor tokozása

Tokozáson a processzor külső burkát, érintkezőinek kialakítását értjük.

Három elterjedt fajtája van:

1) LGA-tokozás: az előző kialakításokkal szemben a tűsor az alaplapon helyezkedik el, míg a processzoron csak úgynevezett érintőpadok[2] találhatóak.
2) PGA-tokozás: itt a csatlakozók a négyzet alakú tok alján helyezkednek el. Ezen belül is lehet:
    a. CPGA, azaz kerámiatok, vagy
    b. PPGA műanyag tok.
3) SECC-tokozás: a tok inkább egy kazettára hasonlít, az érintkezők (tűk) az alján vannak.

8.3.8. A processzor hűtése

A mai processzorok olyan magas frekvencián dolgoznak, hogy egyszerűen elolvadnának az elektromos áram hőhatása miatt, ezért ezt kell hűtőrendszerrel orvosolni.

Több fajtája létezik:

1) Léghűtéses: A processzorra egy hűtőbordát szerelnek, ami elvonja a hőt, erre pedig egy hűtő-ventilátort, ami hűti a hűtőbordát. Ezt nevezik aktív hűtésnek, passzív hűtésnek nevezik azt a fajta hűtést, ha a ventilátort elhagyják a rendszerből. A hűtőborda és a processzor közé szinte mindig hűtőpasztát tesznek, a jobb hőátadás érdekében. Ez általában alumínium hűtőpaszta.
2) Vízhűtéses: Csövekben vizet cirkuláltatnak, és ezt kötik rá a hűteni kívánt alkatrészre. Teljesen halk, emellett igen hatékony, ám kiépítése bonyolult és drága.

Egyéb hűtési fajták is léteznek, de ezek nem olyan elterjedtek, például:

1) Peltier hűtés: a processzorra egy ún. Peltier elemet raknak, és erre kerül rá egy további hűtő egység. Az elem lényege, hogy a töltés áramlása mellett hőáram alakul ki, amelynek következtében az elem egyik oldaláról a másikra vezeti a hőt → az egyik oldala hideg, míg a másik oldala forró lesz.
2) Hidrogénes hűtés
3) Hőcsöves hűtés
4) Folyékony nitrogénes hűtés

A processzorgyártók különféle módszereket vezettek be arra, hogy ha a CPU nincs terhelés alatt, órajeléből visszavegyen, kisebb teljesítményen dolgozzon, és ezáltal kevesebb hőt termeljen. Ilyen megoldás az AMD Cool 'n Quiet és az Intel SpeedStep technológiája is. Ezeket az eljárásokat főleg hordozható számítógépekben használják.

8.3.9. Processzorgyártók, mai processzortípusok

8.3.9.1. AMD X2 3600 processzor

Az asztali PC-k piacán manapság két nagy processzorgyártó vetekszik egymással, az Intel és az AMD. Az Intel a nagyobb, belőle vált ki az AMD. Mind a két processzorgyártónak nagy részesedése van a videokártyák piacán is. Rajtuk kívül vannak még processzorgyártók ugyan (IBM, Cyrix), de piaci részesedésük a mikroprocesszorok terén igen csekély. A két gyártó főbb processzortípusai:

8.3.9.2. Intel

8.3.9.2.1. Core család

- **Core i7 –**

A jelenlegi asztali csúcskategória, 4 mag/8 szál LGA1155-ös foglalat esetében és 6 mag/12 szál LGA2011-es foglalat esetén.

- **Core i5 –**

4 magos CPU, LGA 1155-ös foglalatba illeszkednek.

- **Core i3 –**

2 magos/2 szálas vagy 2 magos/4 szálas CPU, LGA 1155-ös foglalatba illeszkednek.

- **Core 2 Quad –**

Otthoni gépekbe szánt négymagos processzor, LGA775 foglalatba illeszkedik.

- **Core 2 Duo –**

2 magos, rendkívül jó ár/érték mutatójú, nagy teljesítményű processzor, LGA775 foglalatba illeszkednek.

8.3.9.2.2. Xeon család

- **Xeon E7**

A legnagyobb szerverprocesszorok 10 magos/20 szálas. LGA1567 foglalatba illeszkednek.

- **Xeon E5**

Munkaállomásokba szánt processzorok, 4-8 magszámmal, a legtöbb támogatja az alaplaponkénti 2 vagy több használatát is. LGA2011/LGA1356 foglalatba illeszkednek.

- **Xeon E3**

Munkaállomásokba, mikroszerverekbe szánt processzorok.

- **Pentium 4, Pentium D**

Az Intel előző architektúrára épülő processzorcsaládja, van kétmagos is belőle, a Pentium 4-esek első verziói (Willamette) S423 foglalatba illeszkedtek, második verziói (NorthWood, Prescott 1M) S478

foglalatba illeszkednek, és a Pentium 4-esek legutolsó verziói (Prescott 1M, Prescott 2M és Cedar Mill) LGA775 foglalatba illeszkednek. A Pentium D-k (Pressler) kizárólag LGA775 foglalatba illeszkednek.

- **Celeron**

Mérsékelt árú és teljesítményű processzor, Willamette magosok S478, NorthWood magosok S478, és Prescott magosak pedig S478 illetve LGA 775 foglalatba illeszkednek.

Pentium M (Mobile), Celeron M, Core Solo, Core Duo, Core 2 Duo, mobil gépekbe szánt mérsékelt fogyasztású és hőleadású processzorok.

8.3.9.2.3. AMD

- **FX Bulldozer**

Nyolc magos, Socket AM3+ foglalatba illeszkednek.

- **APU**

Az AMD legutóbbi fejlesztésének egyike, melyet a felvásárolt ATI-val közösen dolgozott ki. Az APU egy újfajta integrált videokártyával ellátott processzort jelent, mely egyaránt alkalmas a multimédiás alkalmazásokra és a közepes igényű játékok futtatására.

- **Opteron**

Szerverprocesszor-család, egyes típusait munkaállomásokban és szuperszámítógépekben is alkalmazzák. 1, 2, 4, 8, 12 és 16 magos típusai is készülnek.

Athlon FX

Csökkentett teljesítményű Opteron processzorok, az FX5x széria egymagos processzor volt, az FX6x széria pedig kétmagos. Az AMD 2007-ben vezette be az AMD 4x4-et, mellyel 4 magos rendszert lehet létrehozni úgy, hogy egy alaplapon 2db processzorfoglalat van. Egyelőre csak az nVidia gyárt hozzá csipkészletet, és csak Socket F(S1207) foglalatban működnek.

- **Phenom X6**

Natív hatmagos processzor

- **Phenom X4 –**

Natív négymagos processzor

- **Phenom X3**

Hárommagos processzor, ami egy olyan Phenom X4-es, aminek a négy magja közül csak három működik, a negyedik mag a gyártás során le lett tiltva, vagy elromlott.

- **Athlon X2**

  Az AMD kétmagos processzora.

- **Athlon64**

  Az AMD híres egymagos 64 bites processzorcsaládja, amely kétmagos változatban is készült.

- **Sempron**

  mérsékelt árú és teljesítményű processzorok.

- **Turion**

  Az AMD mobil processzora.

- **Turion64, Turion64 X2**

  64 bites; illetve kétmagos mobil processzorok.

8.4. Ellenőrző kérdések

1) Definiálja a logikai hálózatok fogalmát!
2) Milyen módon írhatók le a sorrendi hálózatok?
3) Definiálja a véges állapotú gépek fogalmát!
4) Mondjon példát véges állapotú gépekre!
5) Rajzoljon fel egy hét szegmenses kijelzőt! nevezze el a szegmenseket!
6) Milyen típusú építőelem (és miért) a 7 szegmenses kijelző?
7) Hogyan vezérelünk 4 db 7 szegmenses kijelzőt az FPGA-n?
8) Hogyan definiálhatjuk a CPU - t?
9) Hogyan jött létre a CPU? Mi fejlődésének története?
10) Milyen számítógépes architektúrákat ismer? Mi jellemzi ezeket? Melyek a hasonlóságaik és különbségeik? Melyiket hol alkalmazzák?
11) Melyek a CPU részei?
12) Jellemezze az ALUT! Definiálja, nevezze meg a részeit, a be és kimeneteit! Hogyan végzi el az egyes alapműveleteket? Hogyan jelenítjük meg az eredményt?
13) Mit tekintünk flag- nek?
14) Mi az AGU ? Mire és hogyan használjuk?

15) Mi a CU? Mire és hogyan használjuk?
16) Mit nevezünk regiszternek? Milyen típusai vannak? Hogyan használhatjuk fel?
17) Hogyan készíthetünk véletlen szám generátort?
18) Mit jelent a gyűrűs számláló kifejezés?
19) Mi a buszvezérlő?
20) Mi a Chase? Mire és hogyan használjuk? Milyen típusai vannak?
21) Milyen módokon lehet egy algoritmust megvalósítani?
22) Milyen részei vannak a pprocesszor utasításkészletének?
23) Mi a szerepe a programszámlálónak?

8.5. Feladatok

Feladat az eddigi feladatok átnézése és a laborokhoz kiadott anyagok tanulmányozása.

8.6. Irodalom

Kóré László: Digitális elektronika I. (BMF 1121)

Zsom Gyula: Digitális technika I. (Műszaki Könyvkiadó, Budapest, 2000, KVK 49-273/I, ISBN 963 6 1786 6)

Zsom Gyula: Digitális technika II. (Műszaki Könyvkiadó, Budapest, 2000, KVK 49-273/II, ISBN 963 16 1787 4)

Arató Péter: Logikai rendszerek tervezése (Tankönyvkiadó, Budapest, 1990, Műegyetemi Kiadó 2004, 55013)

Zalotay Péter: Digitális technika (http://www.kobakbt.hu/jegyzet/DigitHW.pdf)

Rőmer Mária: Digitális rendszerek áramkörei (Műszaki Könyvkiadó, Budapest, 1989, KVK 49-223)

Rőmer Mária: Digitális technika példatár (KKMF 1105, Budapest 1999)

Matijevics István: Digitális Technika Interaktív példatár (ISBN 978-963-279-528-7 Szegedi Tudományegyetem)

http://www.inf.u–szeged.hu/projectdirs/digipeldatar/digitalis_peldatar.html