ÓBUDAI EGYETEM

Véges állapotú gépek

Steiner Henriette

ÓBUDAI EGYETEM

Logikai hálózat

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.

Logikai hálózat típusai

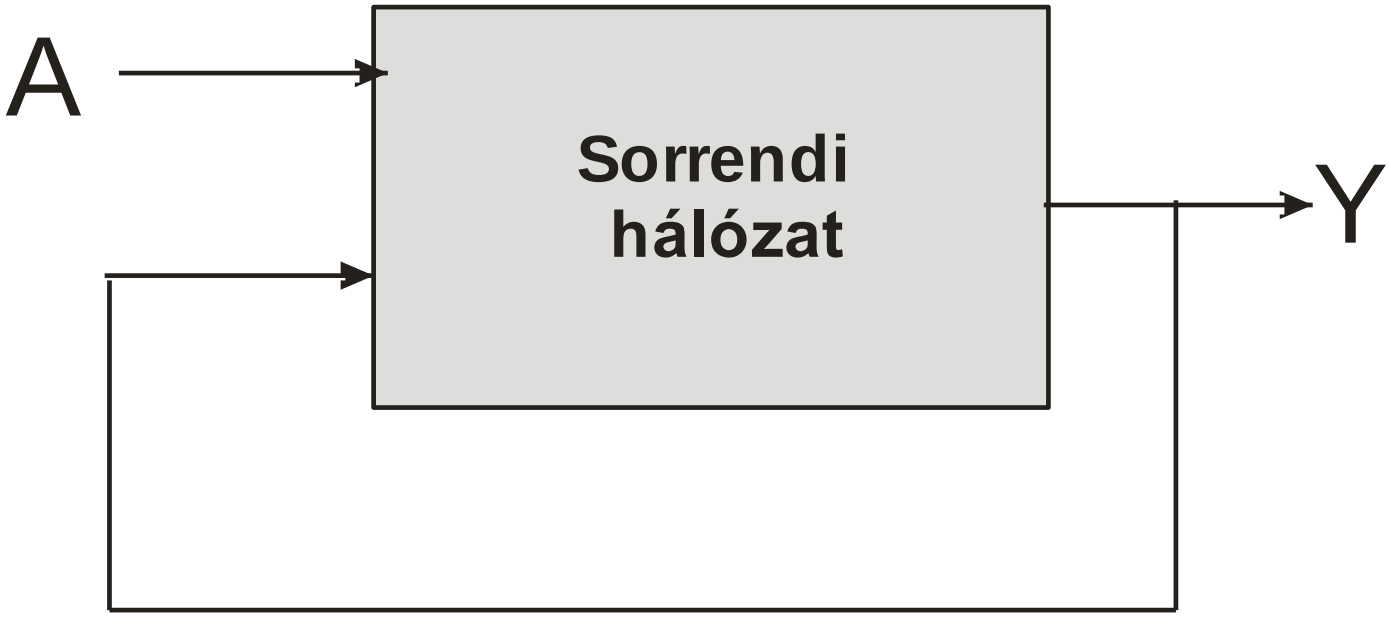
- Kombinációs hálózat
- Sorrendi hálózat

Kombinációs hálózat

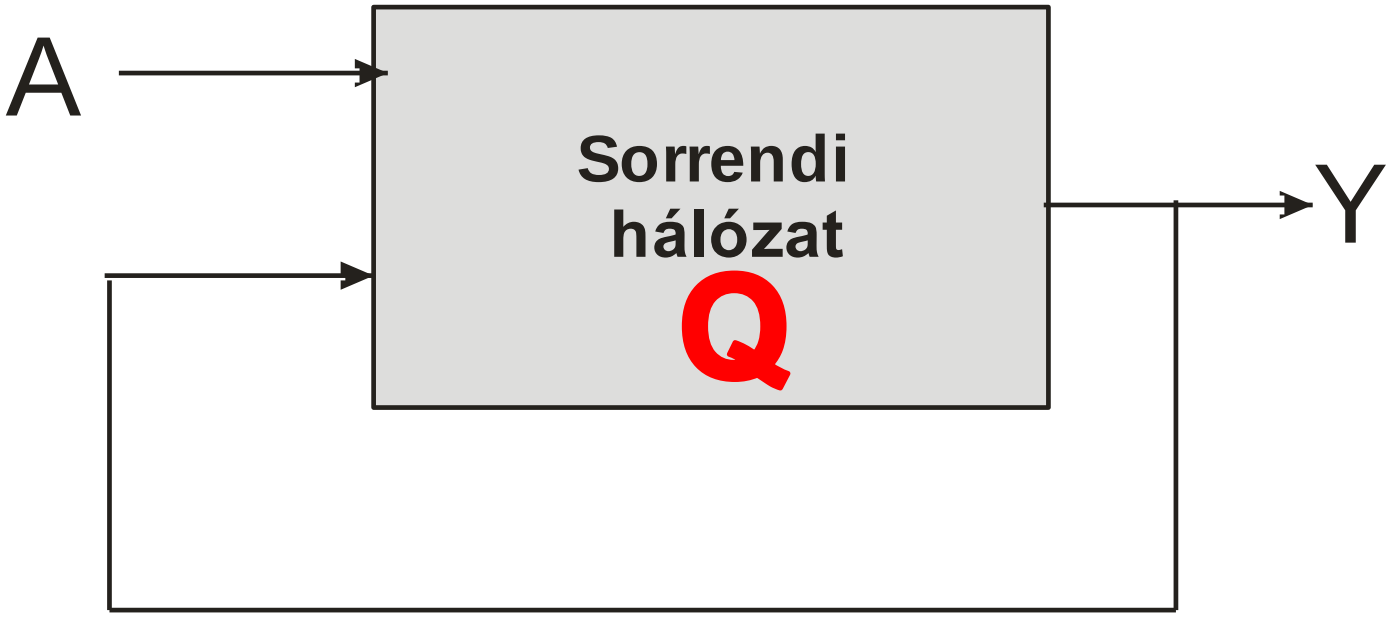
- Kimeneti jelei csak a bemeneti jelek pillanatnyi értékétől függnek
- „Emlékezet” nélküli hálózatok.

Kombinációs hálózat

Sorrendi hálózat



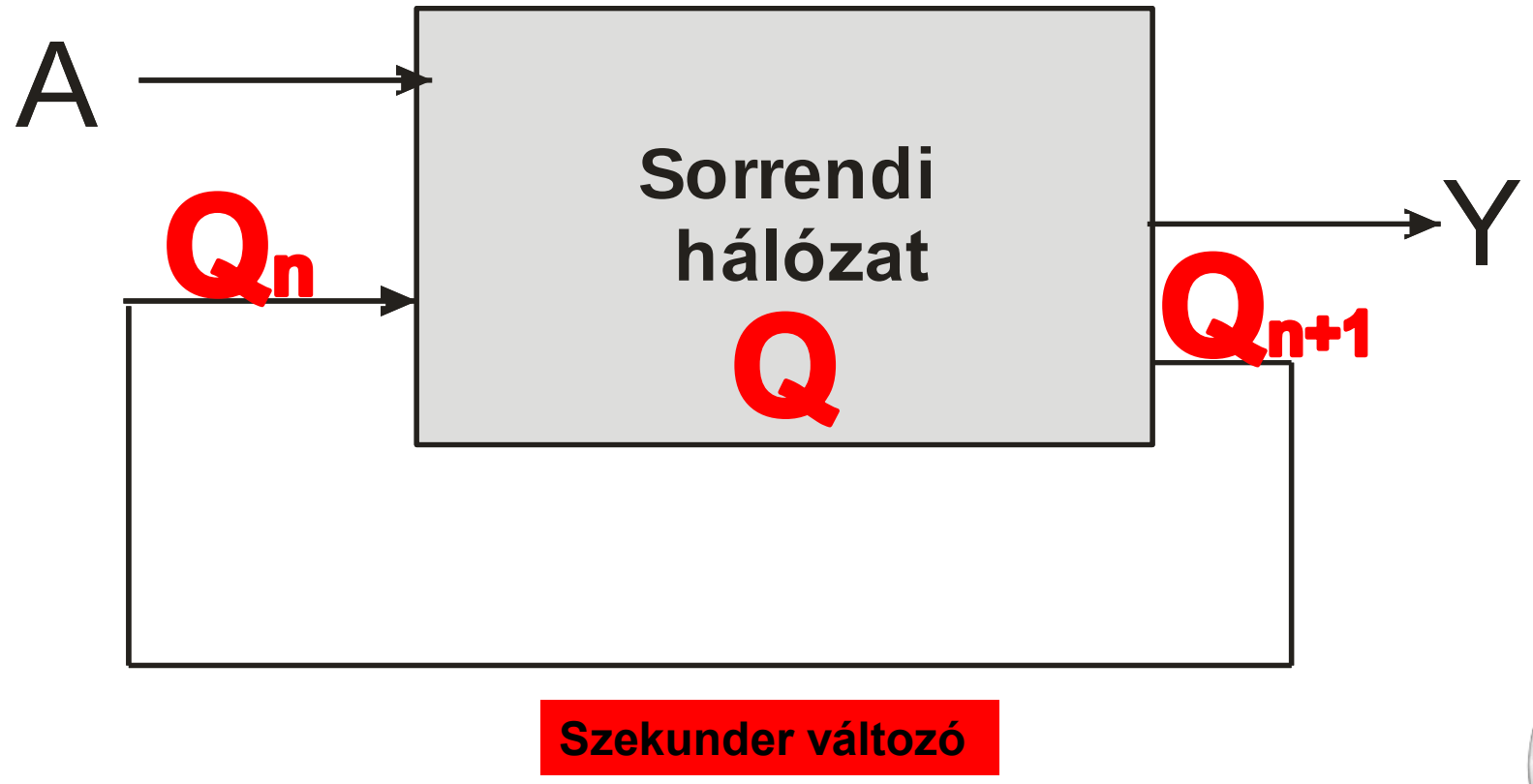
Sorrendi hálózat



Belső állapot

Sorrendi hálózat

Primer változó



A sorrendi hálózat működése

- Egy n hosszúságú bemeneti sorozat (szekvencia)
- n hosszúságú belső állapot (szekunder változó) szekvencia jön létre,
- n hosszúságú kimeneti szekvencia generálódik.
- Ha n véges: **véges sorrendi automatá**ról beszélünk.

A sorrendi hálózatok leírása

- Függvény
- Modell
- Állapottábla
- Állapotgráf

Függvény

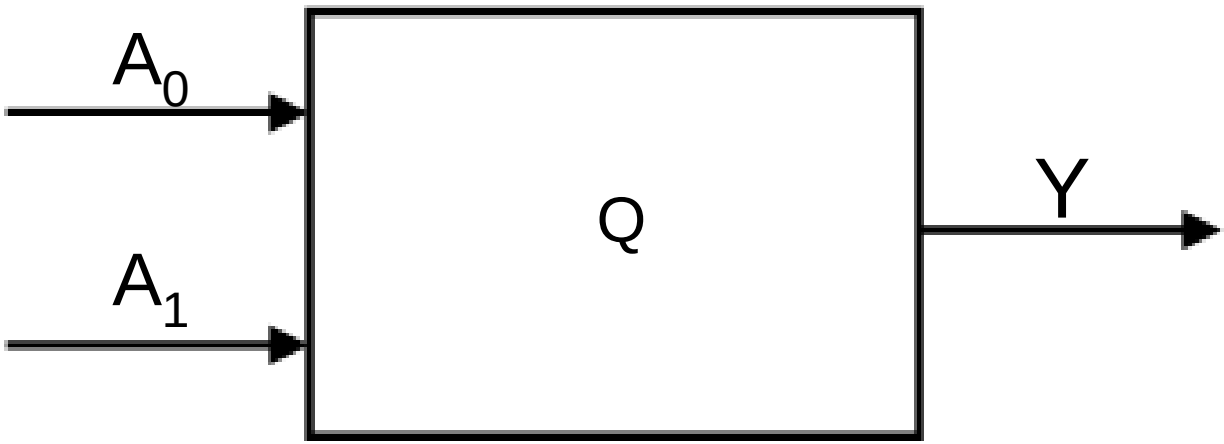
$$f_y = (A, Q) \Rightarrow Y \qquad f_{Qn+1} = (A, Q) \Rightarrow Q^{n+1}$$

Példa 1: Italautomata

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

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

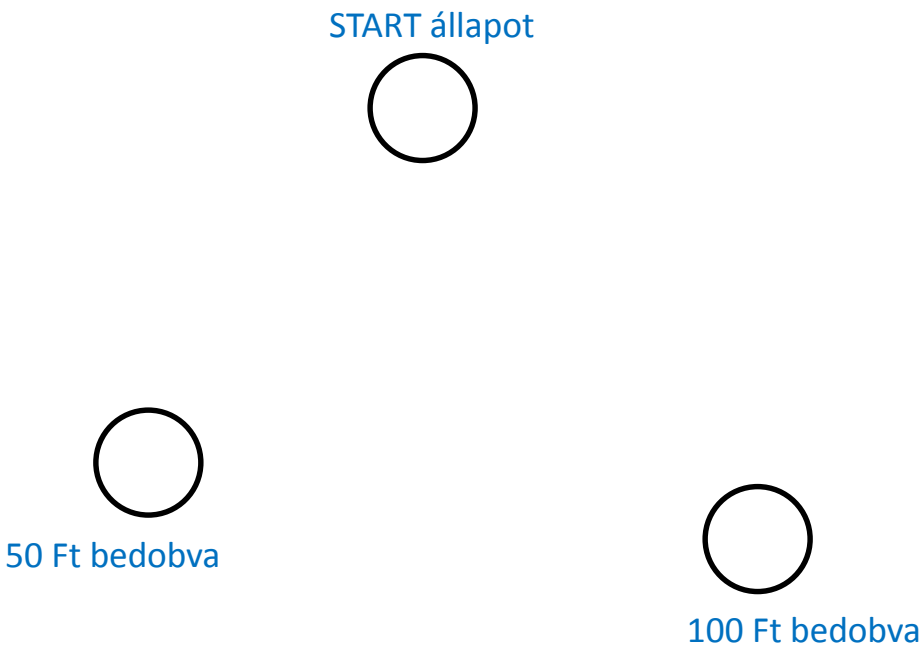
Italautomata



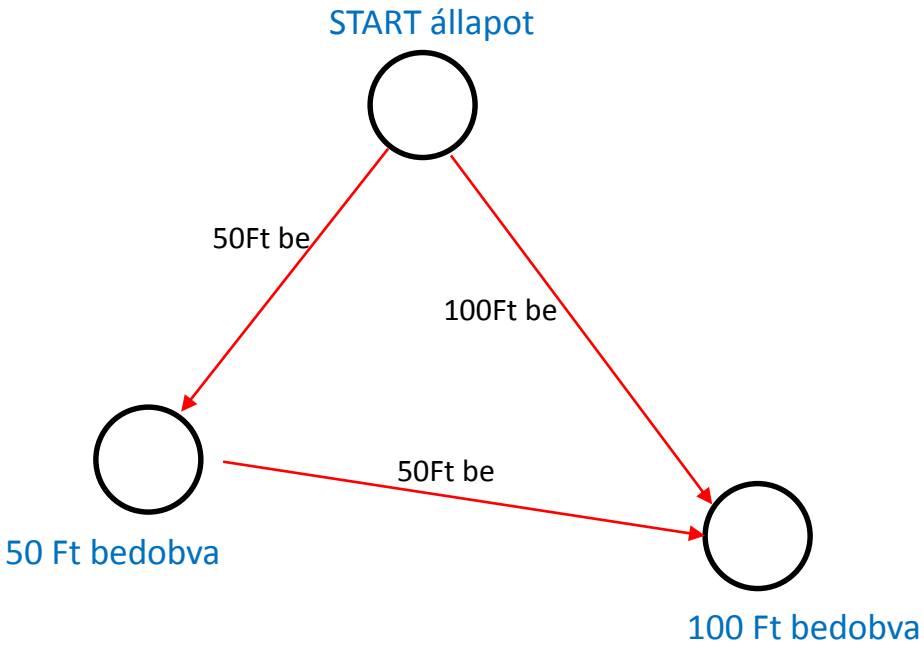
Italautomata

- **Bemenetek:**
  - 100Ft;
  - 50 Ft
- **Kimenetek;**
  - üdítő ki;
  - 50 Ft ki;
- **Belső állapotok:**
  - Start állapot:
  - Bedobtunk 100 Ft-ot : ,
  - Bedobtunk 50 Ft-ot:

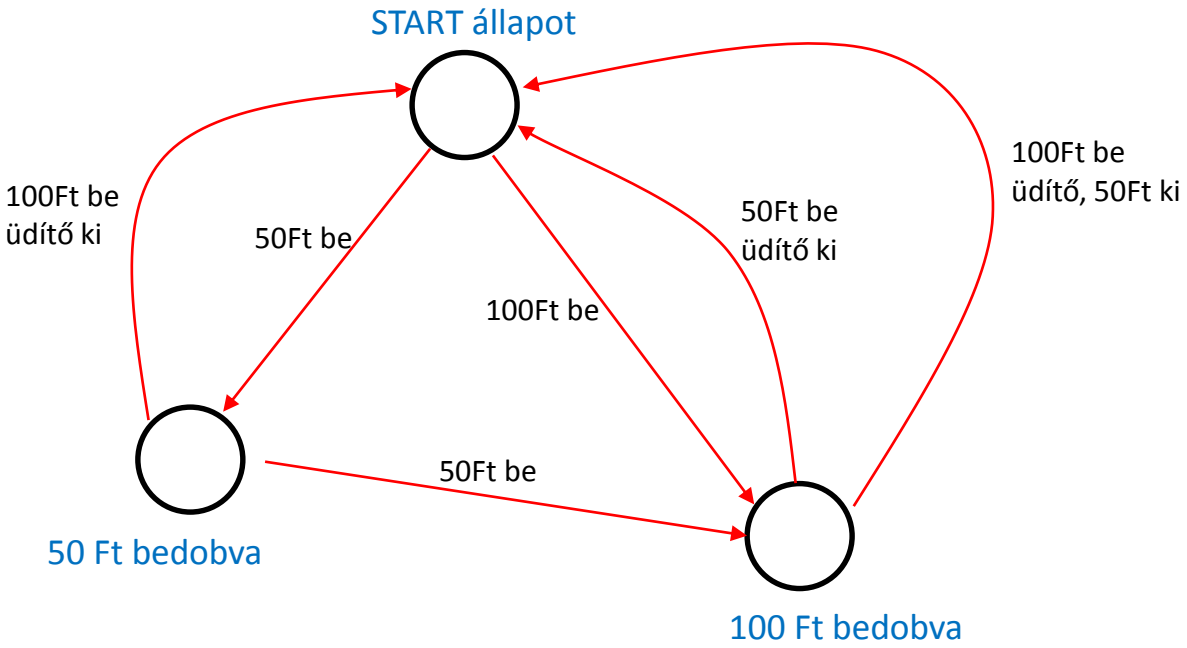
Italautomata



Italautomata



Italautomata



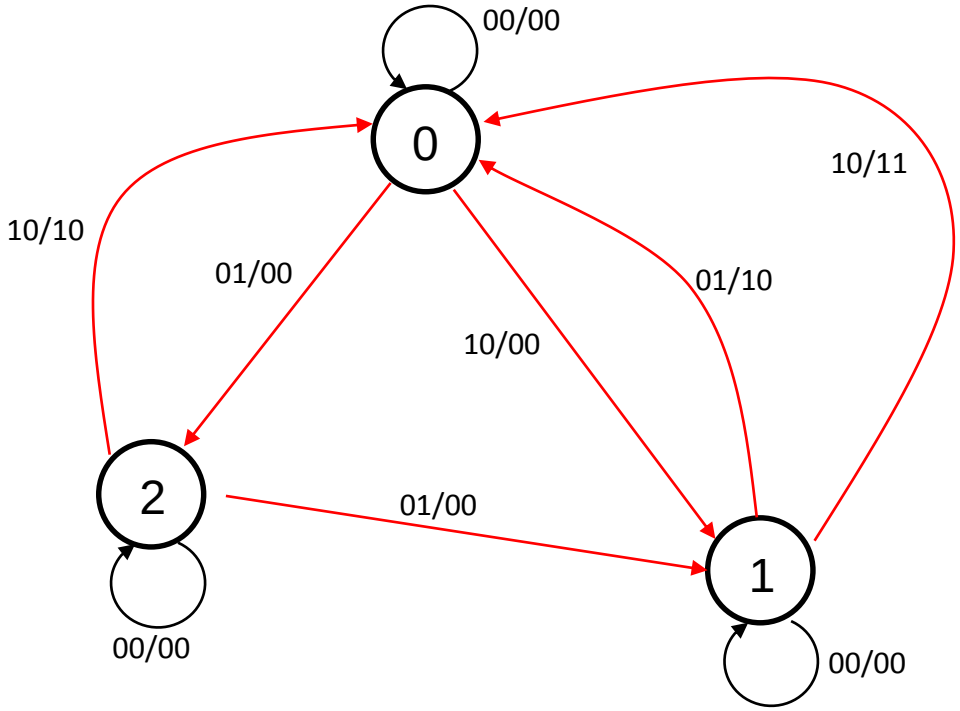
Italautomata

- **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.

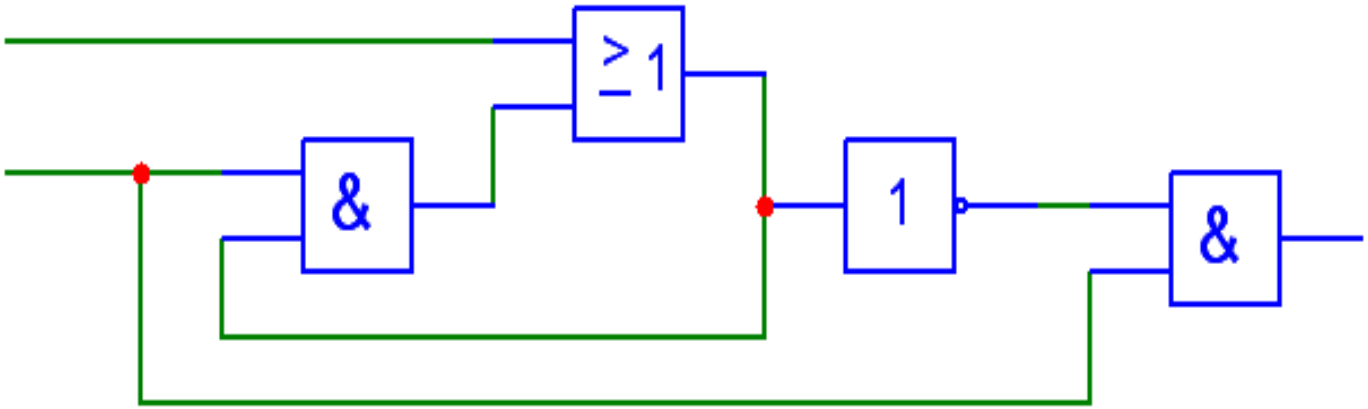
Italautomata - táblázat

Előző állapot	Bemenet (x_1x_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**

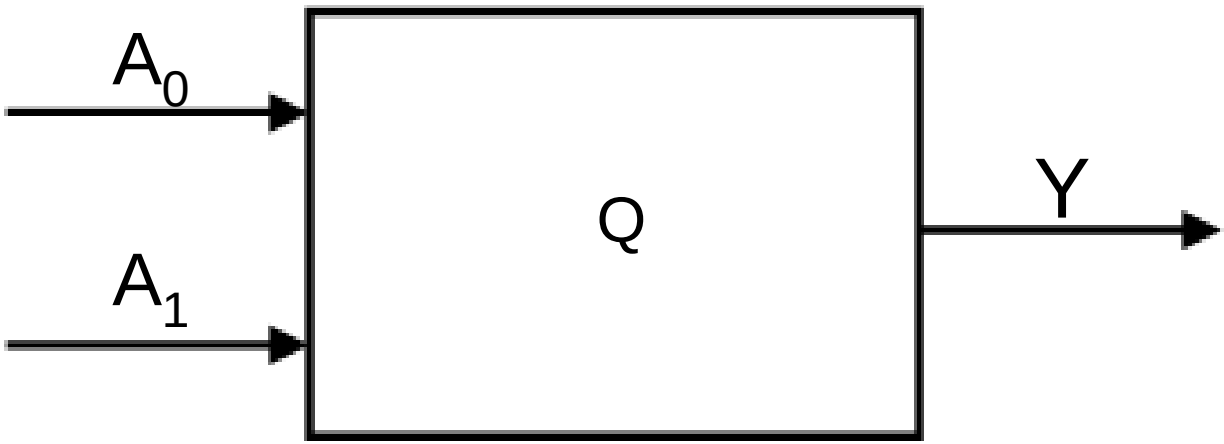
Italautomata



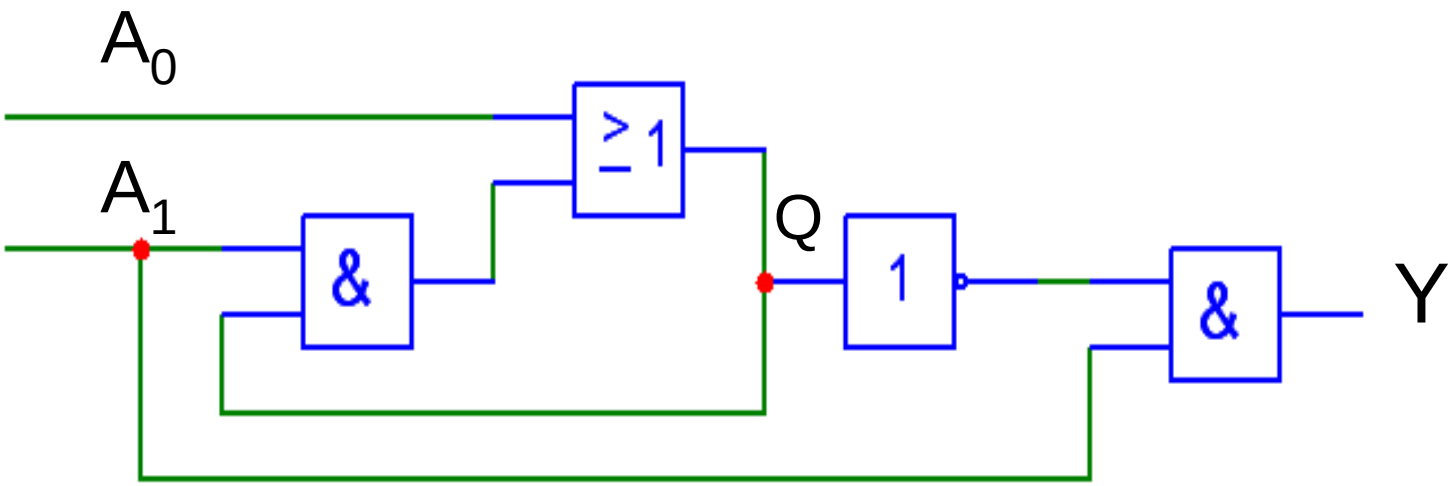
Példa 2: Hálózat



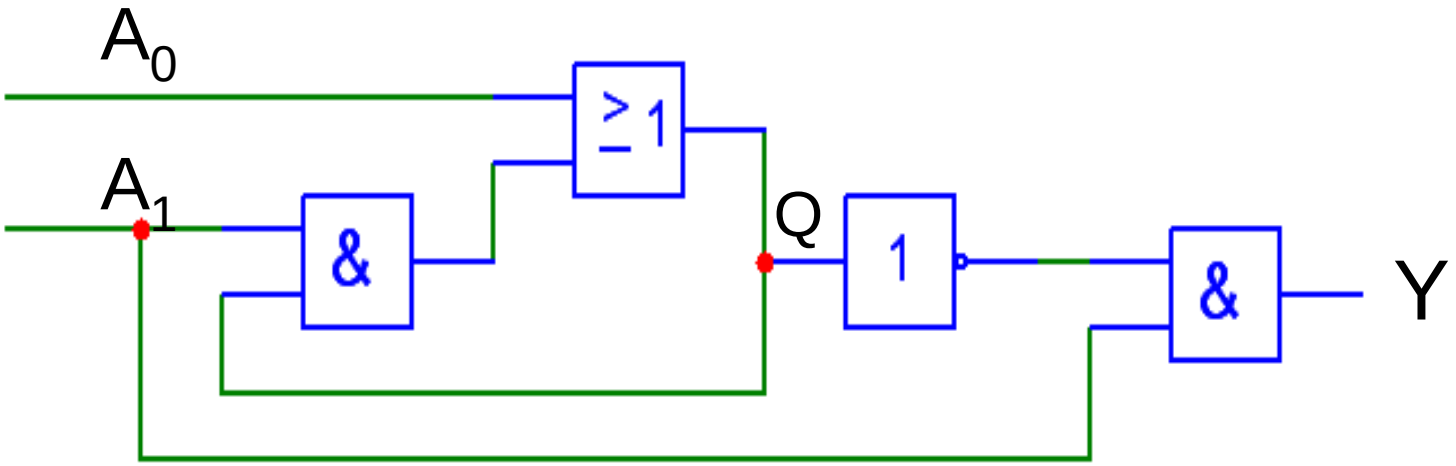
Hálózat



Hálózat



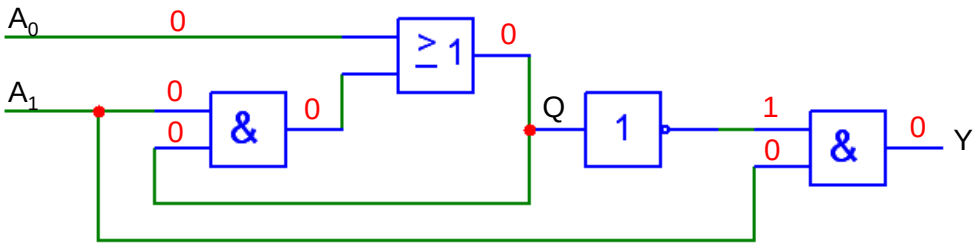
Hálózat



$$Q_{n+1} = A_0 + A_1 Q$$

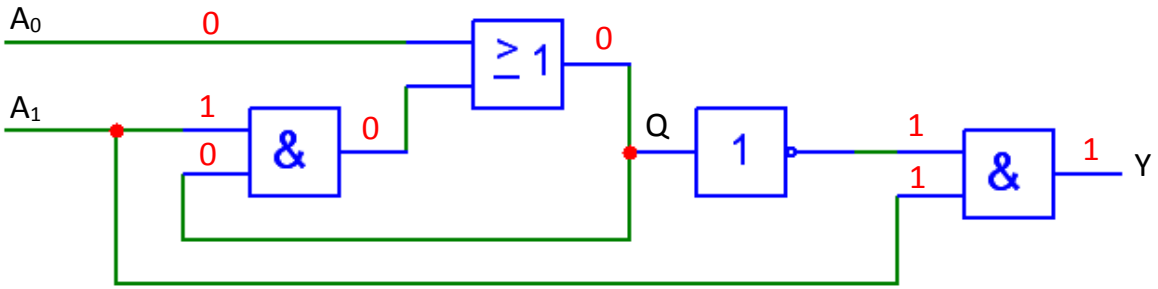
$$Y = A_1 \cdot \bar{Q}$$

Hálózat



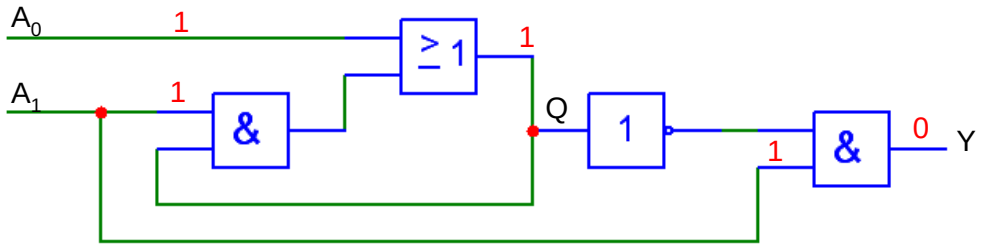
A_0	A_1	Q	Y
0	0	0	0

Hálózat



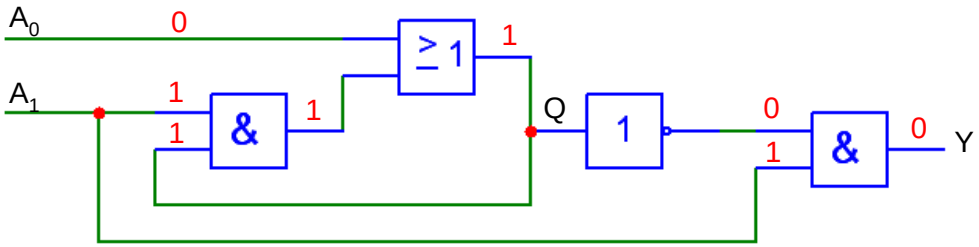
A_0	A_1	Q	Y
0	0	0	0
0	1	0	1

Hálózat



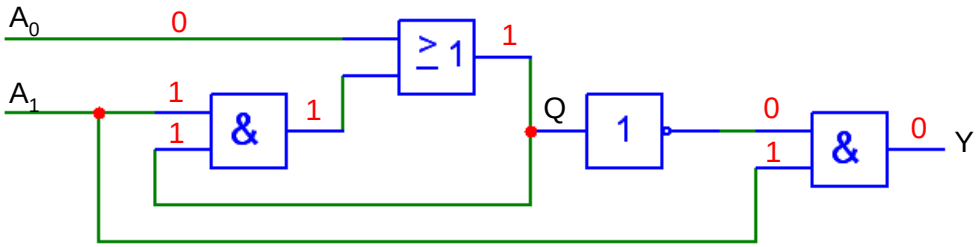
A_0	A_1	Q	Y
0	0	0	0
0	1	0	1
1	1	0	0

Hálózat



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

Hálózat

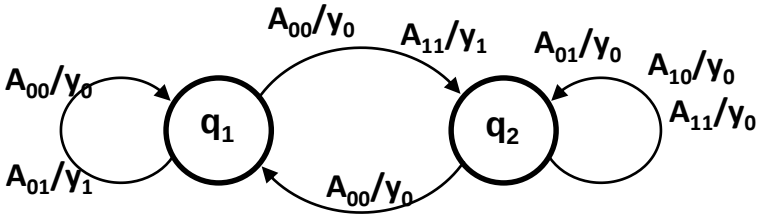


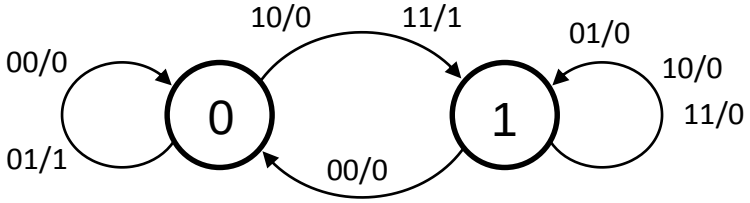
A_0	A_1	Q	Y
0	0	0	0
0	1	0	1
1	1	0	0
0	1	0	0

Hálózat - Tábla

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

Hálózat


A_{00}/y_0
A_{11}/y_1
A_{01}/y_0
A_{10}/y_0
A_{11}/y_0
A_{00}/y_0
A_{01}/y_1
A_{00}/y_0
q_1
q_2


10/0
11/1
01/0
00/0
10/0
11/0
01/1
00/0
0
1

Példa 3:
7 szegmenses kijelző

Építőelemek

- Kódolók, Dekódolók
- Adatút-választók
- Aritmetikai egységek

Építőelemek

- **Kódolók, Dekódolók**
  - Bináris → BCD átalakító (kódoló)
  - BCD → 7 szegmenses kijelző meghajtó (dekódoló)
- **Adatút-választók**
  - Multiplexer
  - Demultiplexer
- **Aritmetikai egységek**
  - Digitális komparátor
  - Összeadó, kivonó

Kódolók

kódoló

Bináris → BCD átalakító (kódoló)

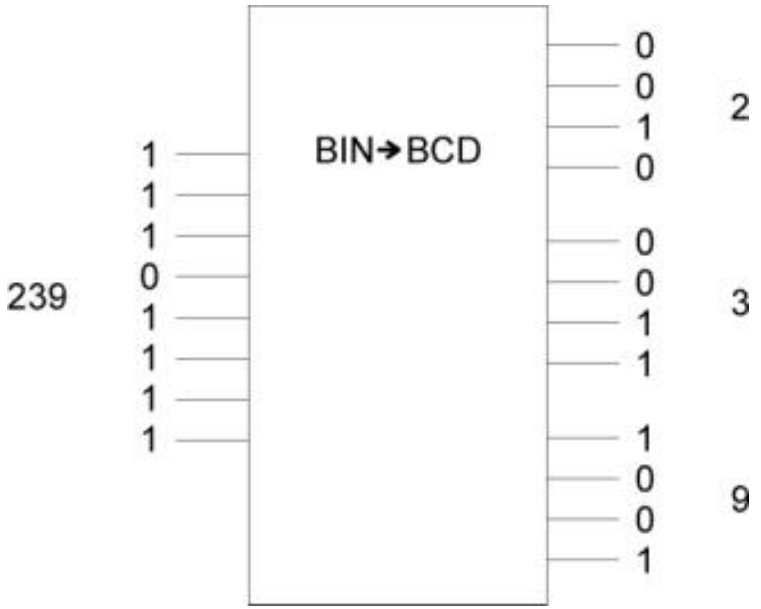
239

1
1
1
0
1
1
1
1

Kódolók

kódoló

Bináris → BCD átalakító (kódoló)



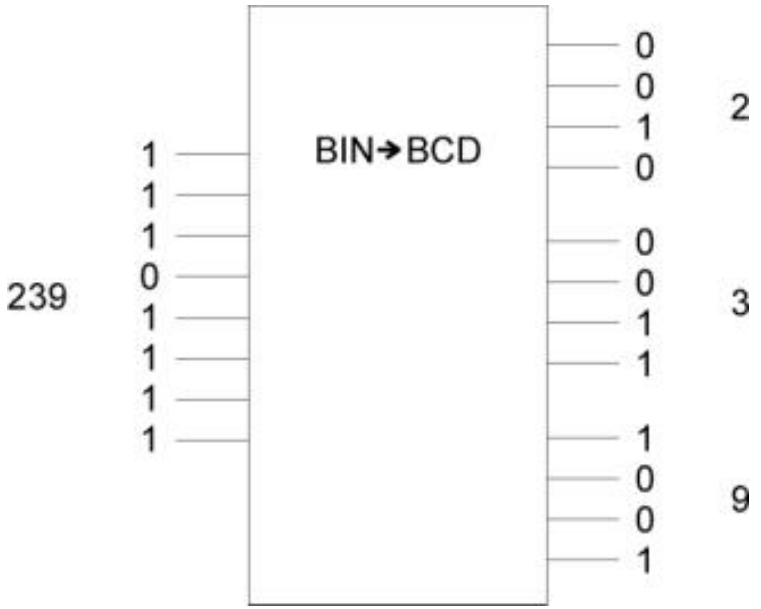
Kódolók

kódoló

Bináris → BCD átalakító (kódoló)

dekódoló

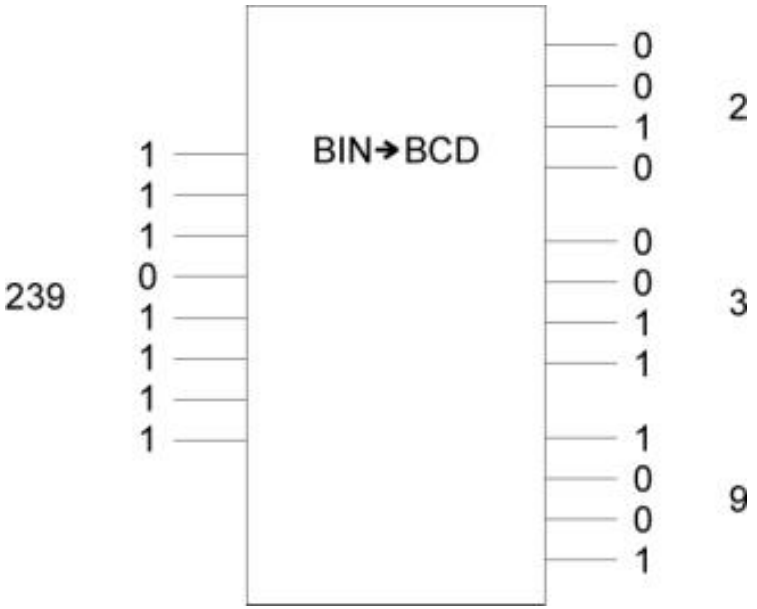
BCD → 7 szegmenses kijelző meghajtó (dekódoló)



Kódolók

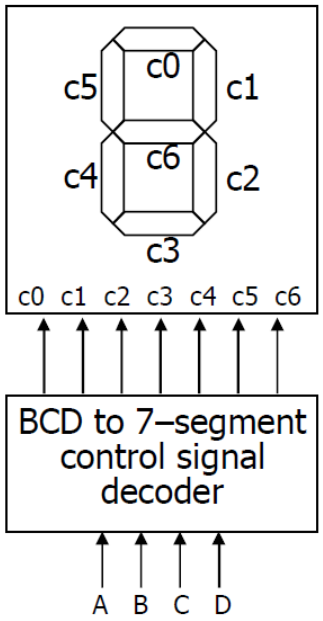
kódoló

Bináris → BCD átalakító (kódoló)



dekódoló

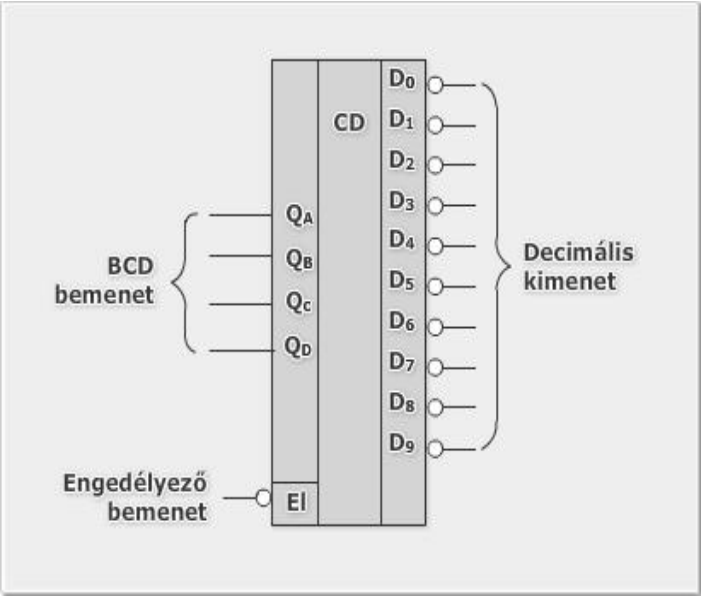
BCD → 7 szegmenses kijelző meghajtó (dekódoló)



Kódolók

kódoló

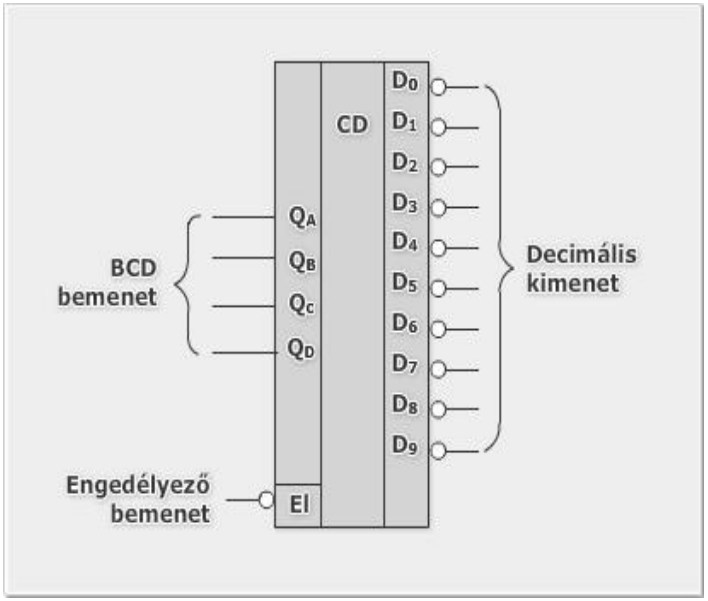
BCD → Decimális átalakító (kódoló)

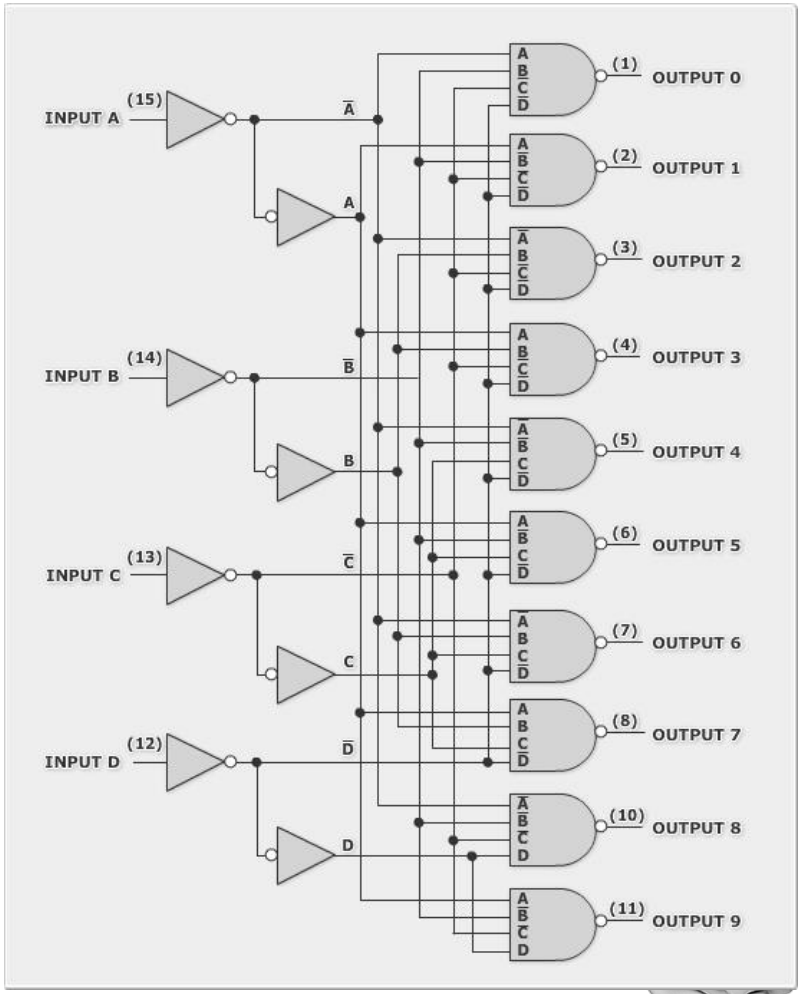


Kódolók

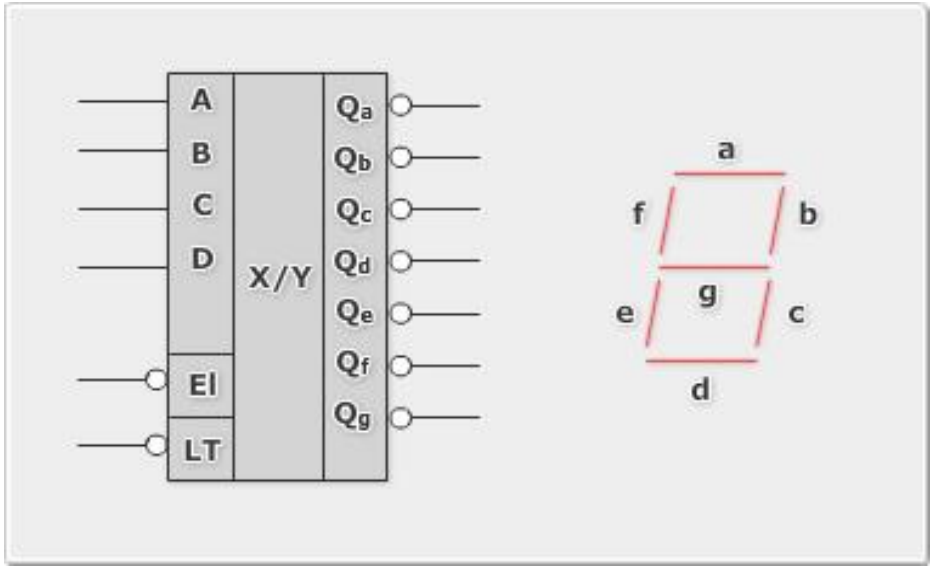
kódoló

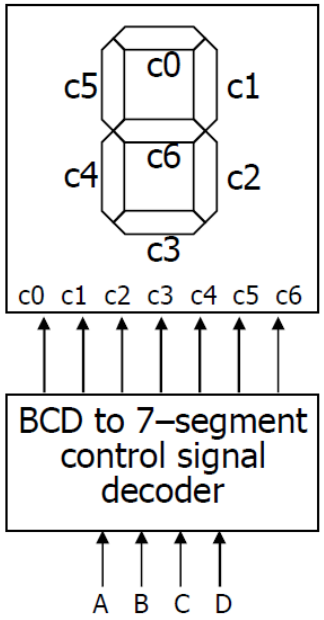
BCD → Decimális átalakító (kódoló)





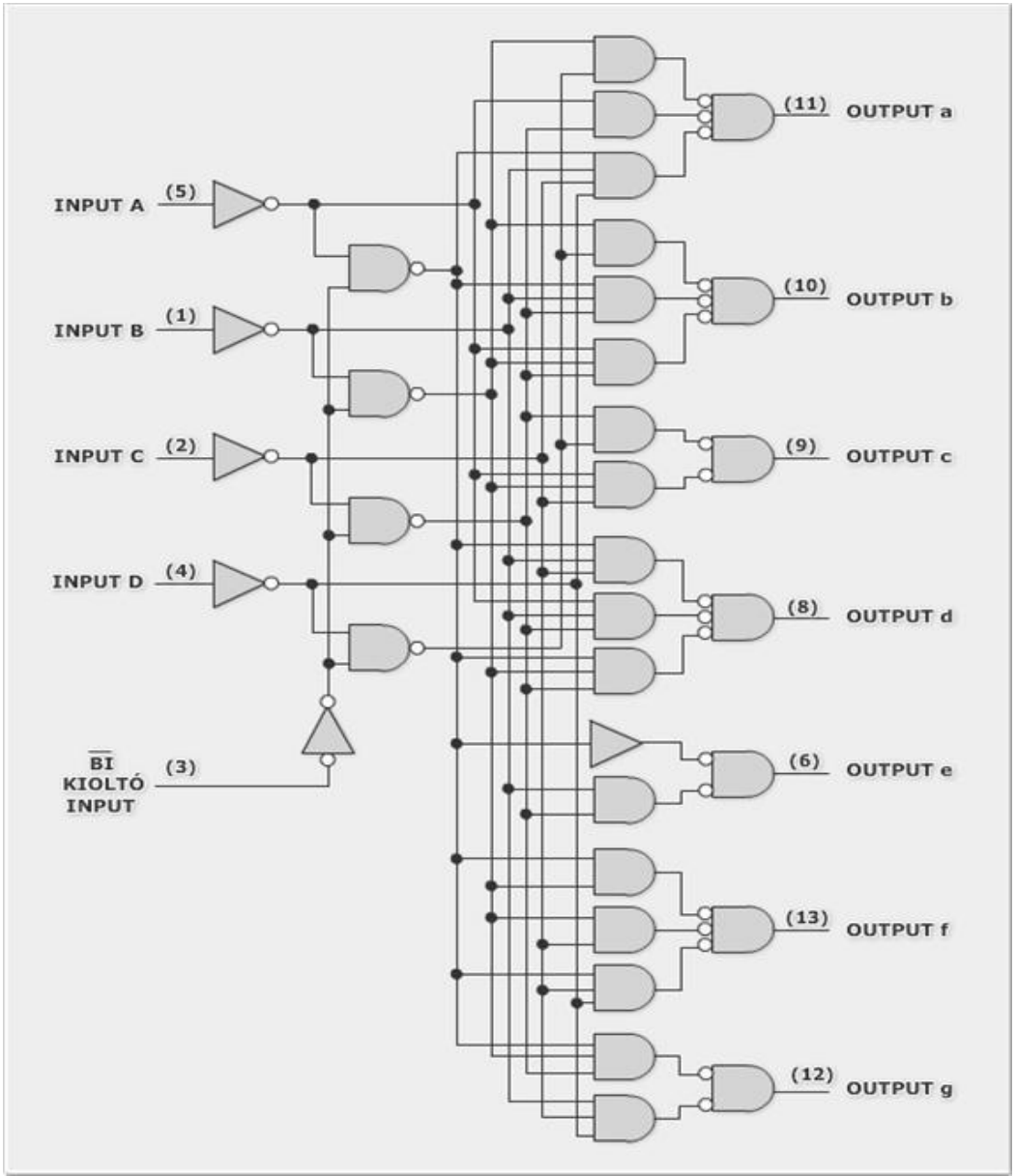
Dekódoló


A
B
C
D
X/Y
EI
LT
Qa
Qb
Qc
Qd
Qe
Qf
Qg
a
f
b
g
e
c
d


c0
c1
c2
c3
c4
c5
c6
c0 c1 c2 c3 c4 c5 c6
BCD to 7–segment control signal decoder
A B C D

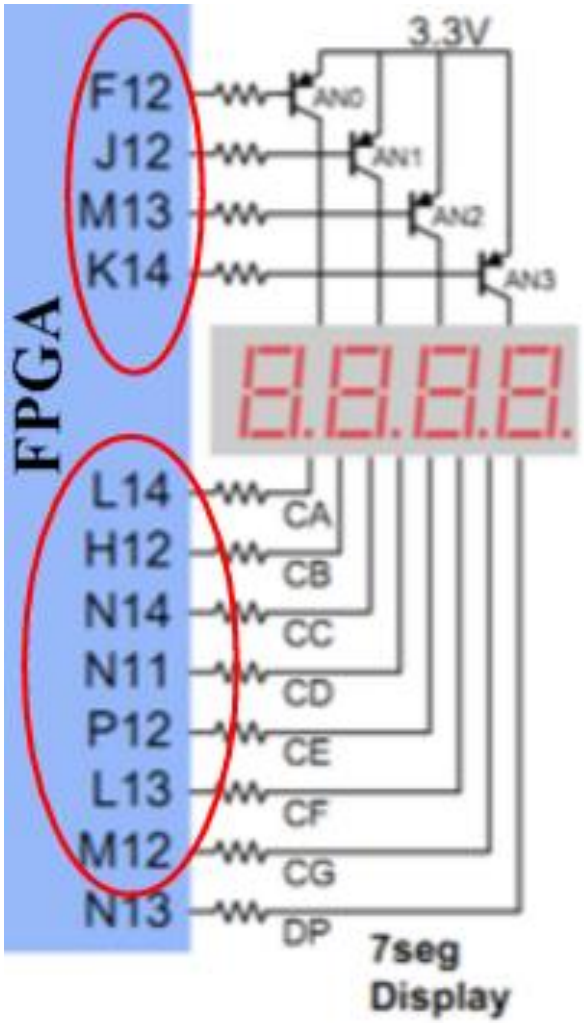

a
f
b
g
e
c
d
A szegmensek azonosítása
A számjegyek kijelzése és a lehetséges kijelzések

Dekódoló


INPUT A
(5)
INPUT B
(1)
INPUT C
(2)
INPUT D
(4)
BI
KIOLTÓ
INPUT
(3)
(11)
OUTPUT a
(10)
OUTPUT b
(9)
OUTPUT c
(8)
OUTPUT d
(6)
OUTPUT e
(13)
OUTPUT f
(12)
OUTPUT g

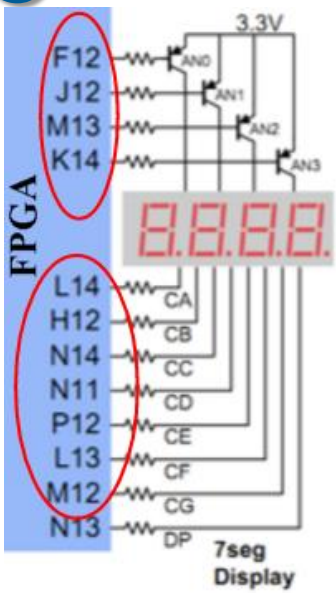
Emlékeztető: A hétszegmenses kijelző

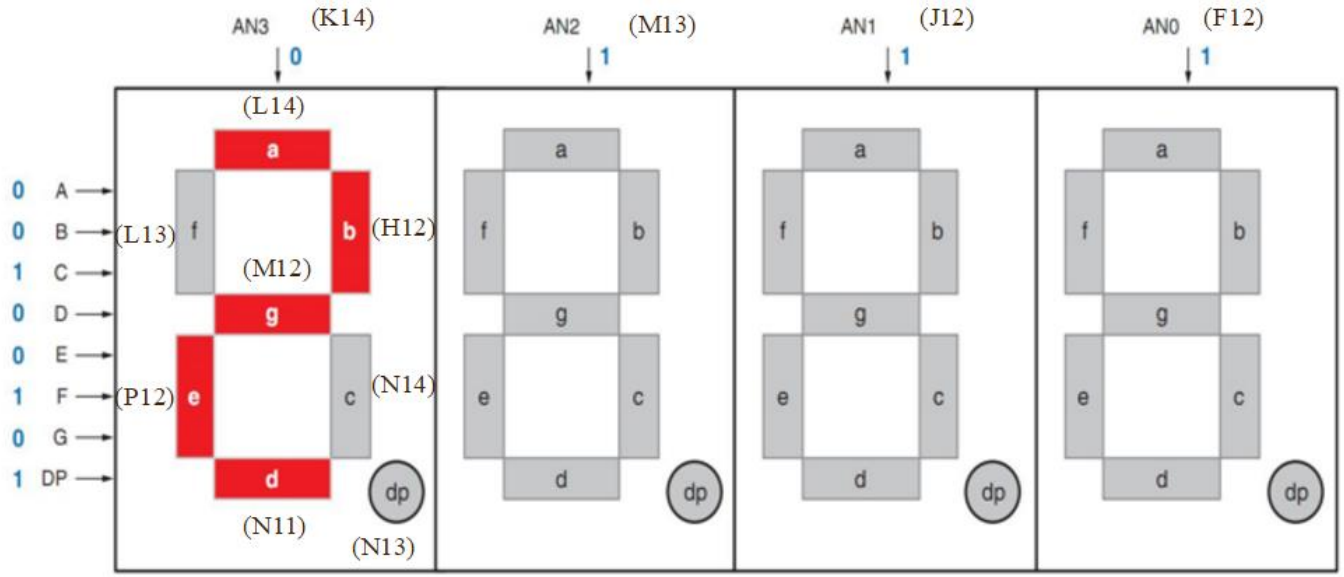
Spartan3 FPGA-nál közös anódos kivitel ⇒ Aktív L szint!!!



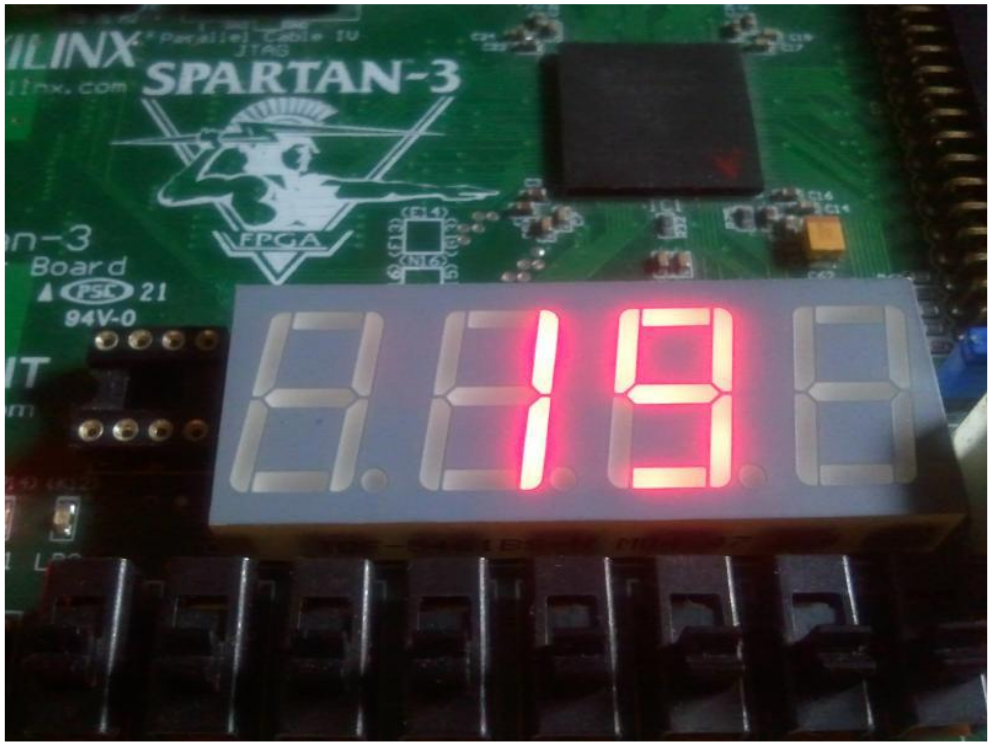
Emlékeztető: A hétszegmenses kijelző

Spartan3 FPGA-nál közös anódos kivitel ⇒ Aktív L szint!!!

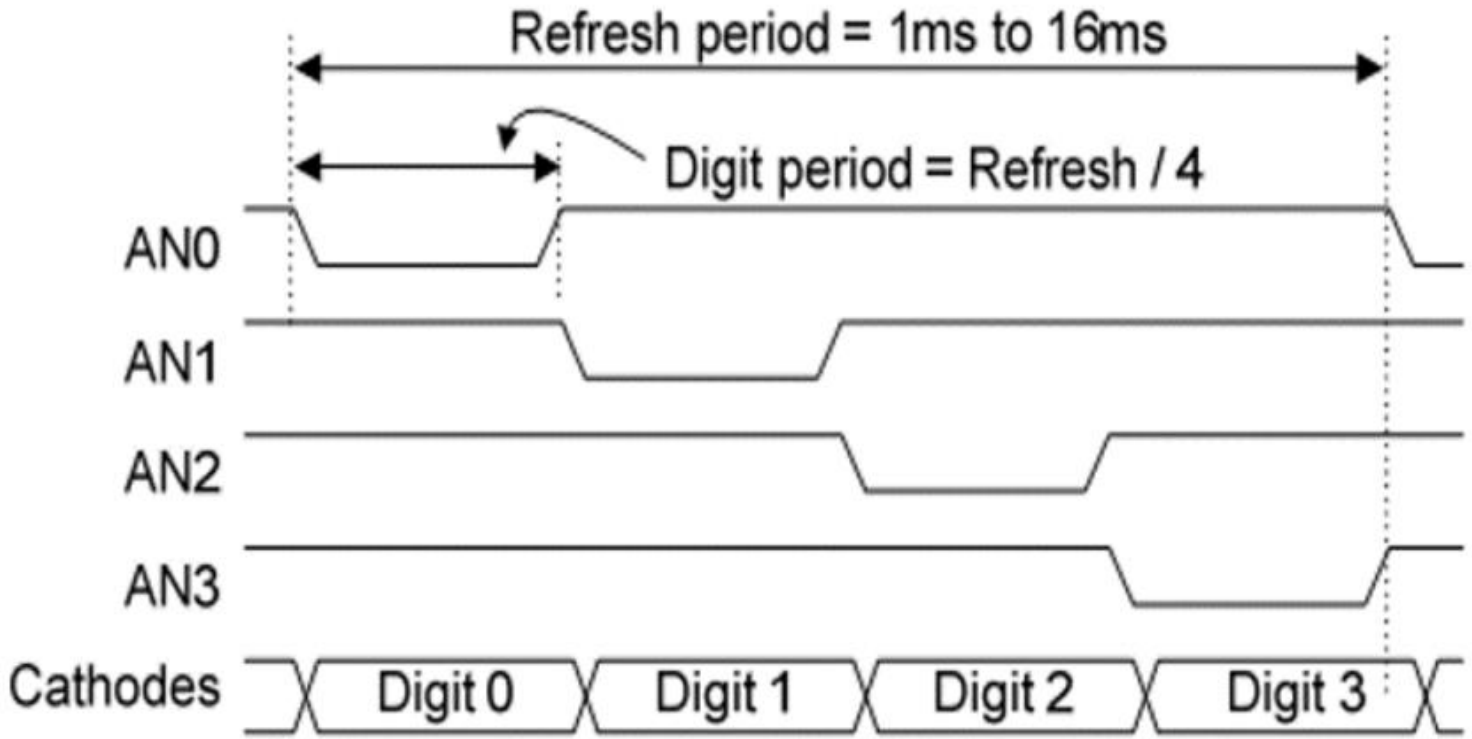




Hogyan jelenítsünk meg különböző értékeket az egyes kijelzőkön?



Megoldás: Multiplex engedélyezés
Minden kijelző kap egy saját időszeletet



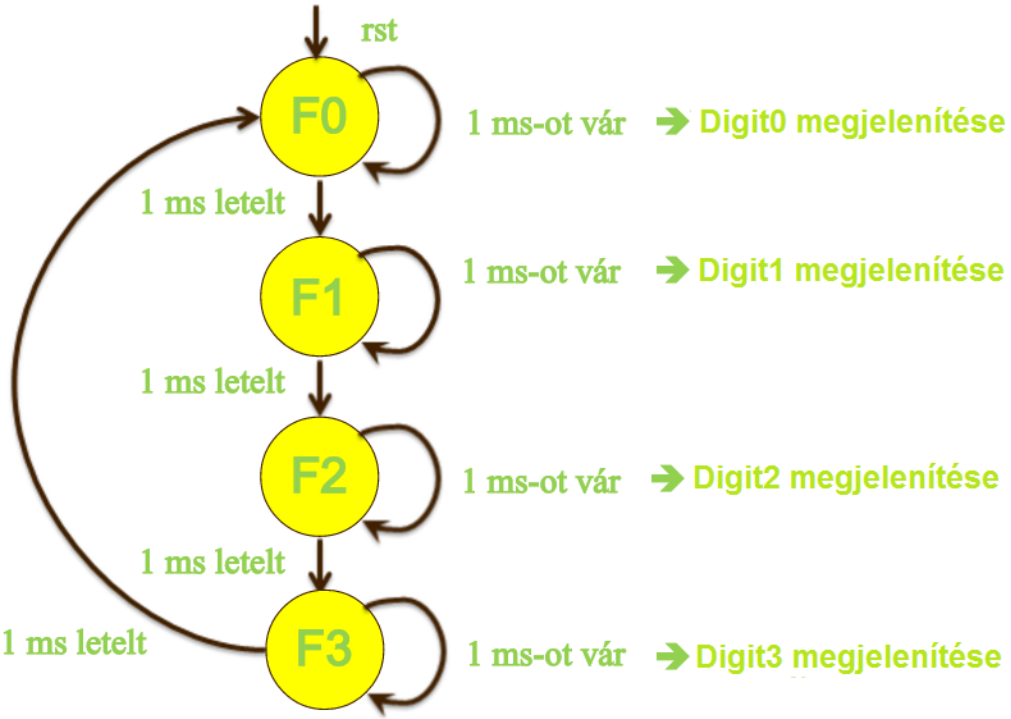
Mekkora legyen az időszelet?

- Ha túl széles az időszelet (> 40 ms), akkor villognak a kijelzők, ez zavaró az emberi szem számára.
- Ha túl keskeny az időszelet (túl nagy a frissítési frekvencia), akkor a kijelző „tehetetlensége” miatt nincs idő arra, hogy az időszelet alatt a tényleges érték beálljon rajta, „összemosódnak” a kijelzett számok.

Tapasztalatok szerint egy-egy digitre néhány ms hosszú időszelet megfelelő.

Az egyes digitek kiválasztása és a hozzájuk tartozó érték megjelenítése a rendszer egy-egy állapota

Véges állapotú gép (Finite State Machine - FSM)

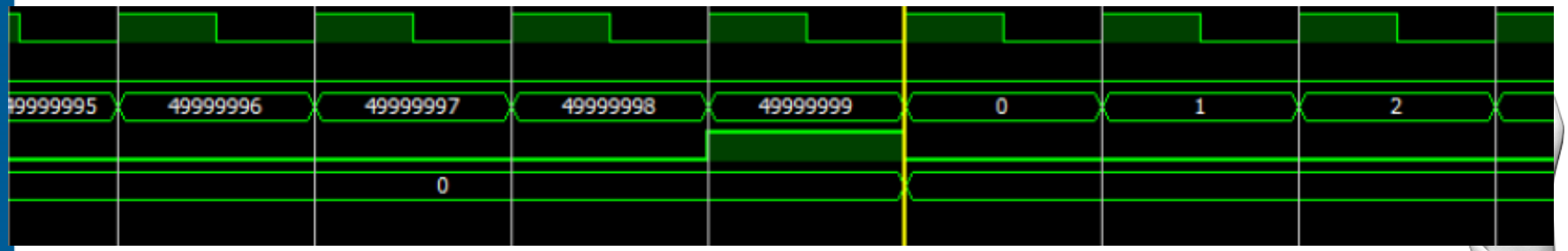


Hogyan mérjük meg az 1 ms-ot?

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):



ÓBUDAI EGYETEM

Példa 4: CPU

Mikroprocesszoros rendszerek

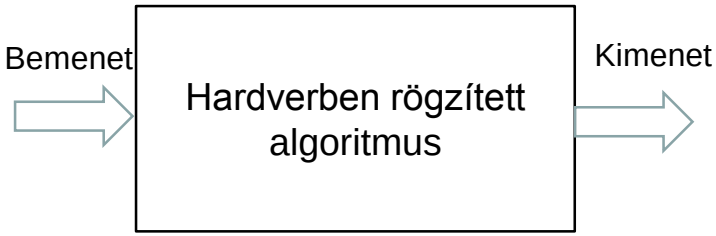
- Algoritmusok megvalósítása
  - Műszaki rendszereket mindig valamilyen feladat megoldása érdekében építünk
  - A feladatmegoldás általában valamilyen algoritmus szerint történik
    - Mérésadatgyűjtés
    - Adatok elemzése (pl. összehasonlítás)
    - Aritmetikai, logikai műveletek végzése az adatokon
    - Döntéshozatal stb.

Mikroprocesszoros rendszerek

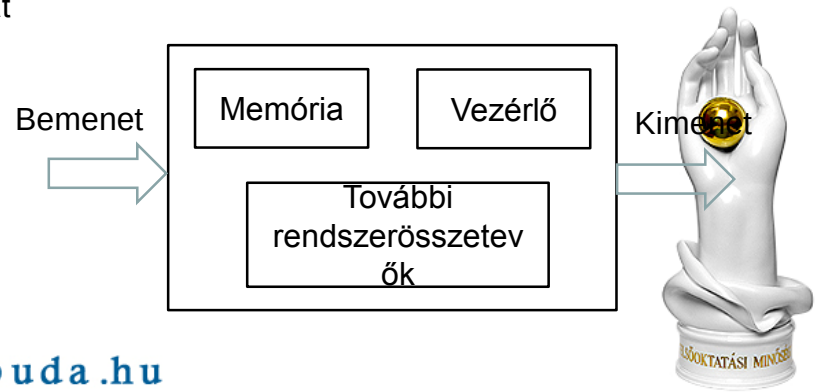
- Az információfeldolgozás menetét (programját) építjük be a rendszerbe
  - 1. megoldás:
    - A rendszer összetevői és egymáshoz való kapcsolódásuk, sorrendiségük a **hardverben fixen „behuzalozva” jelennek meg**
  - 2. megoldás
    - Az algoritmusnak megfelelő sorrendben, előre letárolt program szerint, egy vezérlő berendezés segítségével aktivizáljuk az egyes műveletvégző egységeket

Mikroprocesszoros rendszerek

- Algoritmusok megvalósítása
  1. megoldás: Huzalozott program
    - A rendszer összetevői és egymáshoz való kapcsolódásuk, sorrendiségük a hardverben fixen „behuzalozva” jelennek meg
      - Előny
        - Egyes részfeladatok párhuzamosan is végrehajthatók (gyors működés)
      - Hátrány
        - A hardver a rögzített struktúra miatt csak az adott feladat megoldására alkalmas



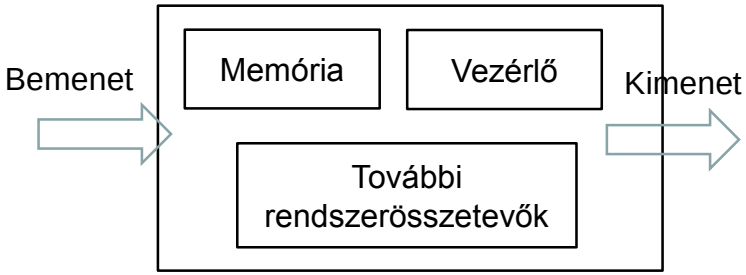
  2. megoldás: Tárolt program
    - Az algoritmusnak megfelelő sorrendben, előre letárolt program szerint, egy vezérlő berendezés segítségével aktivizáljuk az egyes műveletvégző egységeket
      - Előny
        - Ha megváltoztatjuk a memória tartalmát (a programot) más-más feladatra használhatjuk
      - Hátrány
        - A részfeladatok nehezen párhuzamosíthatók (lassúbb működés)
        - Szekvenciális utasítás végrehajtás



CPU

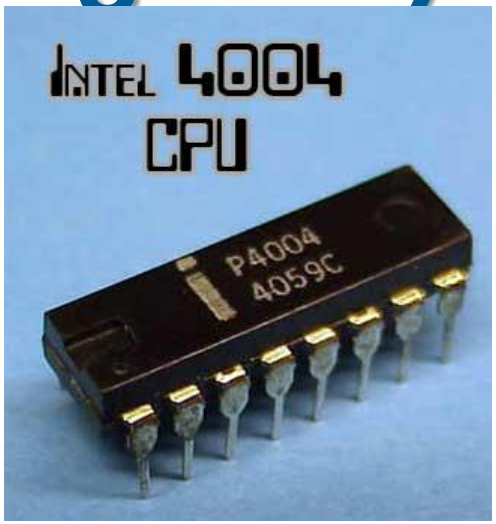
2. megoldás: Tárolt program

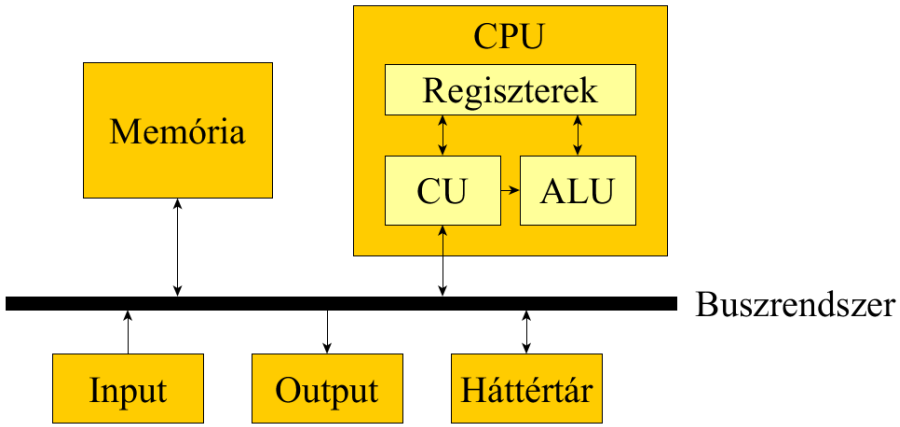
- Az algoritmusnak megfelelő sorrendben, előre letárolt program szerint, egy vezérlő berendezés segítségével aktivizáljuk az egyes műveletvégző egységeket
  - Előny
    - Ha megváltoztatjuk a memória tartalmát (a programot) más-más feladatra használhatjuk
  - Hátrány
    - A részfeladatok nehezen párhuzamosíthatók (lassúbb működés)
    - Szekvenciális utasítás végrehajtás



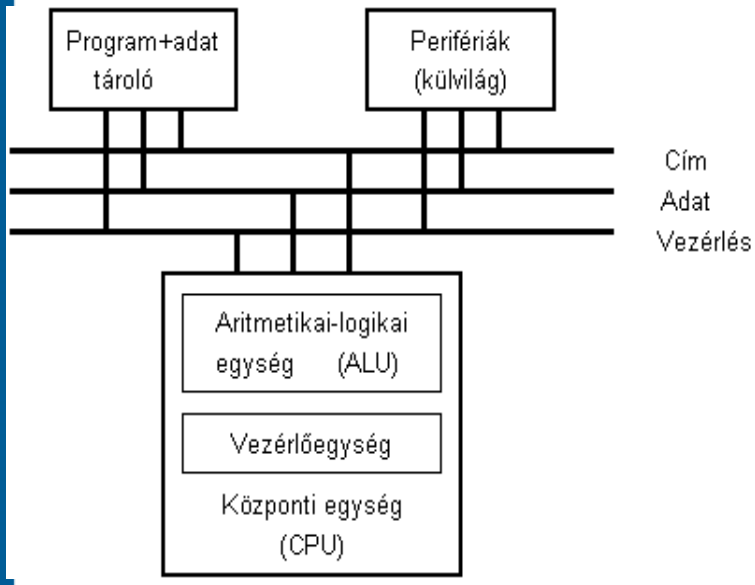
CPU (Central Processing Unit)

1. Az utasítás és az adatok beolvasása a memóriából a processzorba
2. A beolvasott utasítás dekódolása, elemzése
3. A művelet végrehajtása (ALU)
4. Eredmény tárolása
5. A következő utasítás címének meghatározása

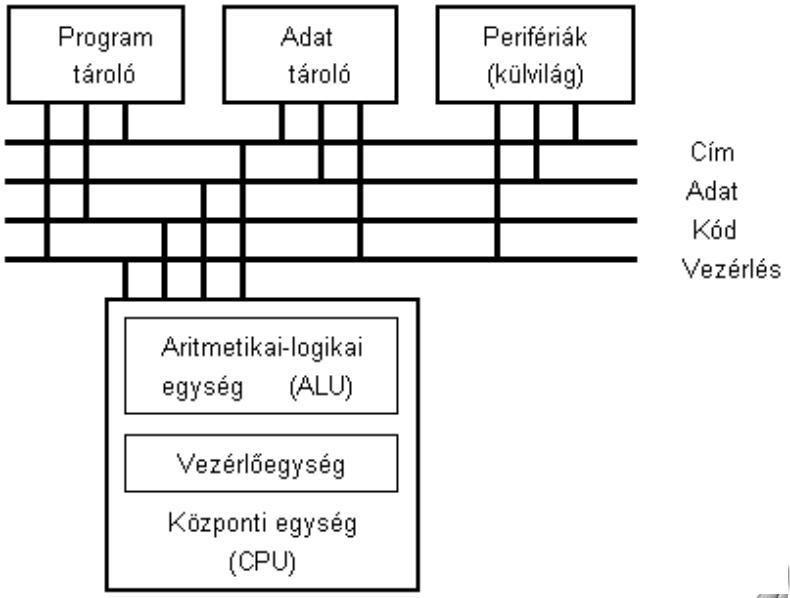




Neumann és Harvard architektúra



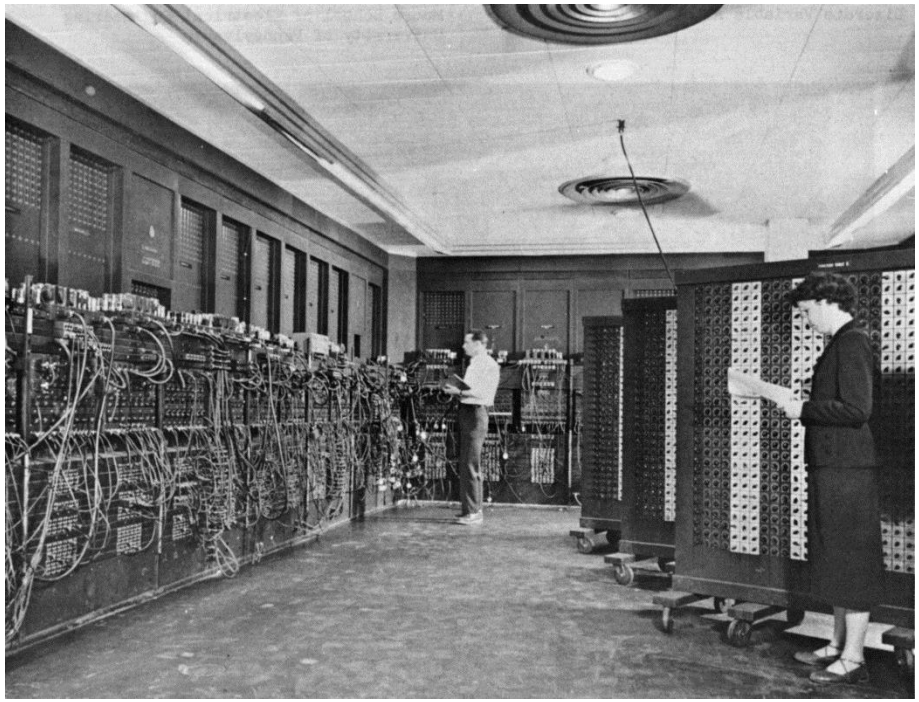
Neumann architektúra



Harvard architektúra

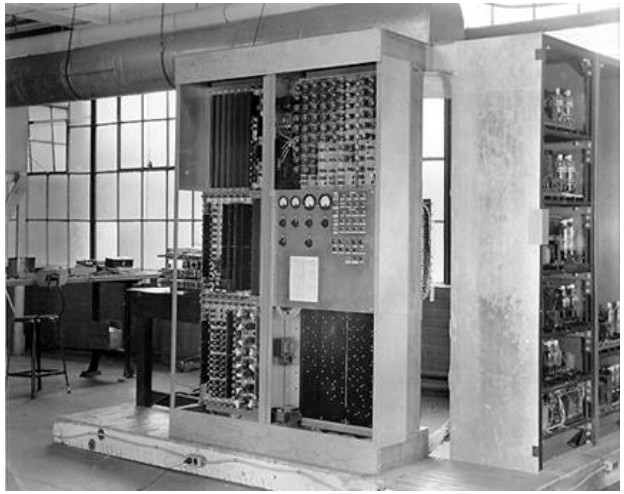
Neumann architektúra

ENIAC (Electronic Numerical Integrator and Calculator) 1943-46-ban:
18 ezer elektroncső, 10 ezer kondenzátor, 6 ezer kapcsolóelem és 1500 jelfogó (rel



Neumann architektúra

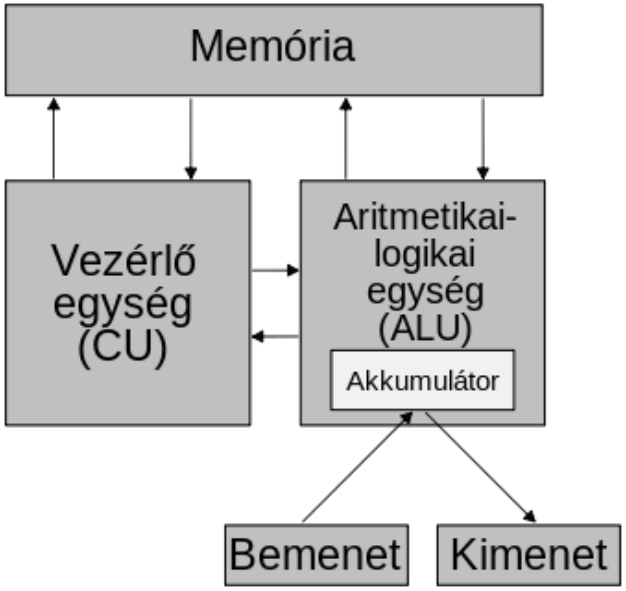
EDVAC (Electronic Discrete Variable Computer) 1949-ben:

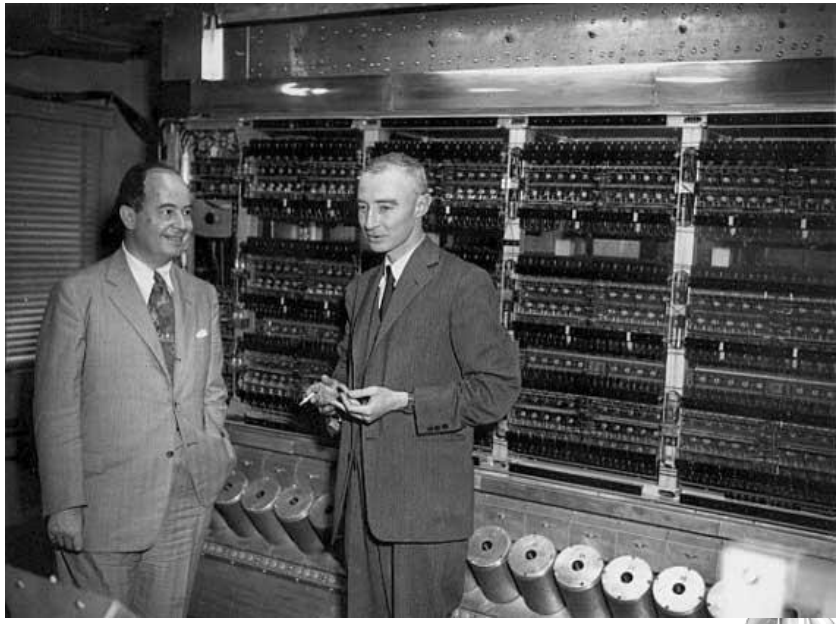


Neumann elvek:
- a kettes számrendszer alkalmazása
- teljes mértékben elektronikus elven működő számítógép
- központi vezérlő egység
- aritmetikai egység alkalmazása
- belső program- és adattárolás, programvezérlés

Neumann architektúra

IAS (Institute for Advanced Study) : JONIAC



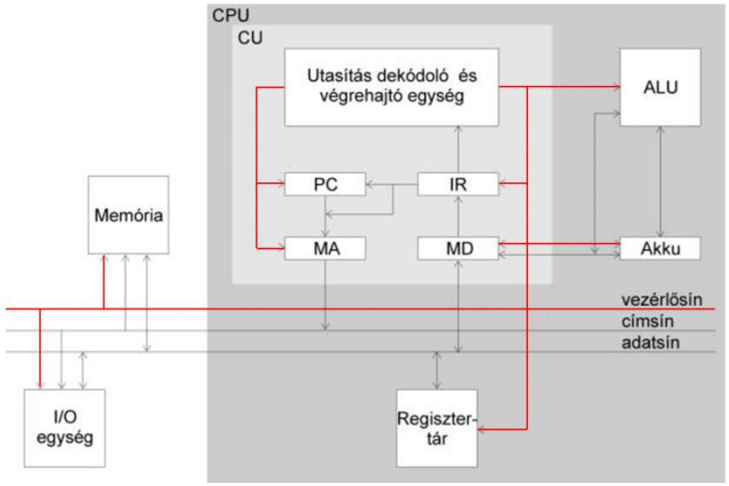


A CPU részei

- ALU: (Arithmetic and Logical Unit – Aritmetikai és Logikai Egység
- AGU: (Address Generation Unit) - a címszámító egység,
- CU: (Control Unit a.m. vezérlőegység vagy vezérlőáramkör).
- Regiszter (Register
- Kijelzők:
  - utasításszámláló, (PC=program counter, IP=instruction pointer)
  - utasításregiszter (IR=instruction register),
  - flagregiszter, veremmutató (SP = Stack Pointer) ,
  - akkumulátor, (AC)
- Buszvezérlő:
- Cache: A modern processzorok fontos része a cache (gyorsítótár).

CPU részei

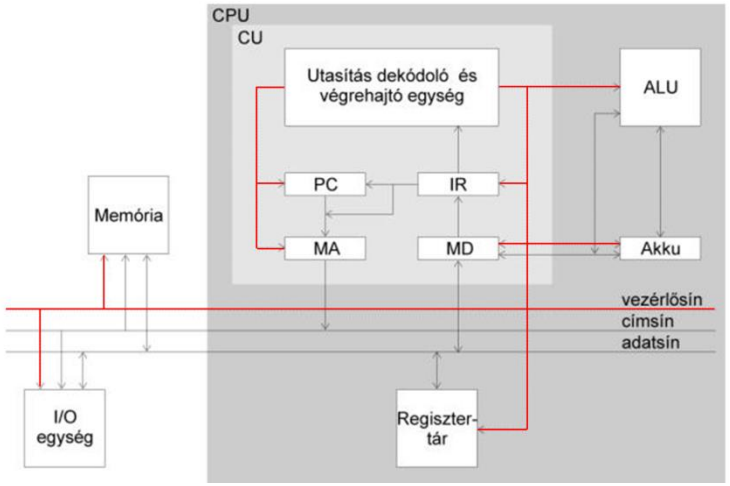
- Utasítás dekódoló és végrehajtó egység
  - Irányítja és ütemezi az összes többi egység működését
  - Az adatokat vagy címeket a megfelelő útvonalon vezeti végig, hogy a kívánt helyre eljussanak
  - Ha szükséges, beindítja az ALU valamelyik műveletvégz áramkörét
- PC (Program Counter Register) programszámláló regiszter
  - A következő utasítás memóriacímét tartalmazza
  - Minden utasítás végrehajtása során egyel nő az értéke



- IR (Instruction Register) utasítás regiszter
  - A memóriából kiolvasott utasítást tartalmazza
  - 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)
- 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)
- MD (Memory Data Register) memória adatregiszter
  - A memóriából kiolvasott adat közvetlenül ide kerül, illetve a memóriába innen töltjük az adatokat

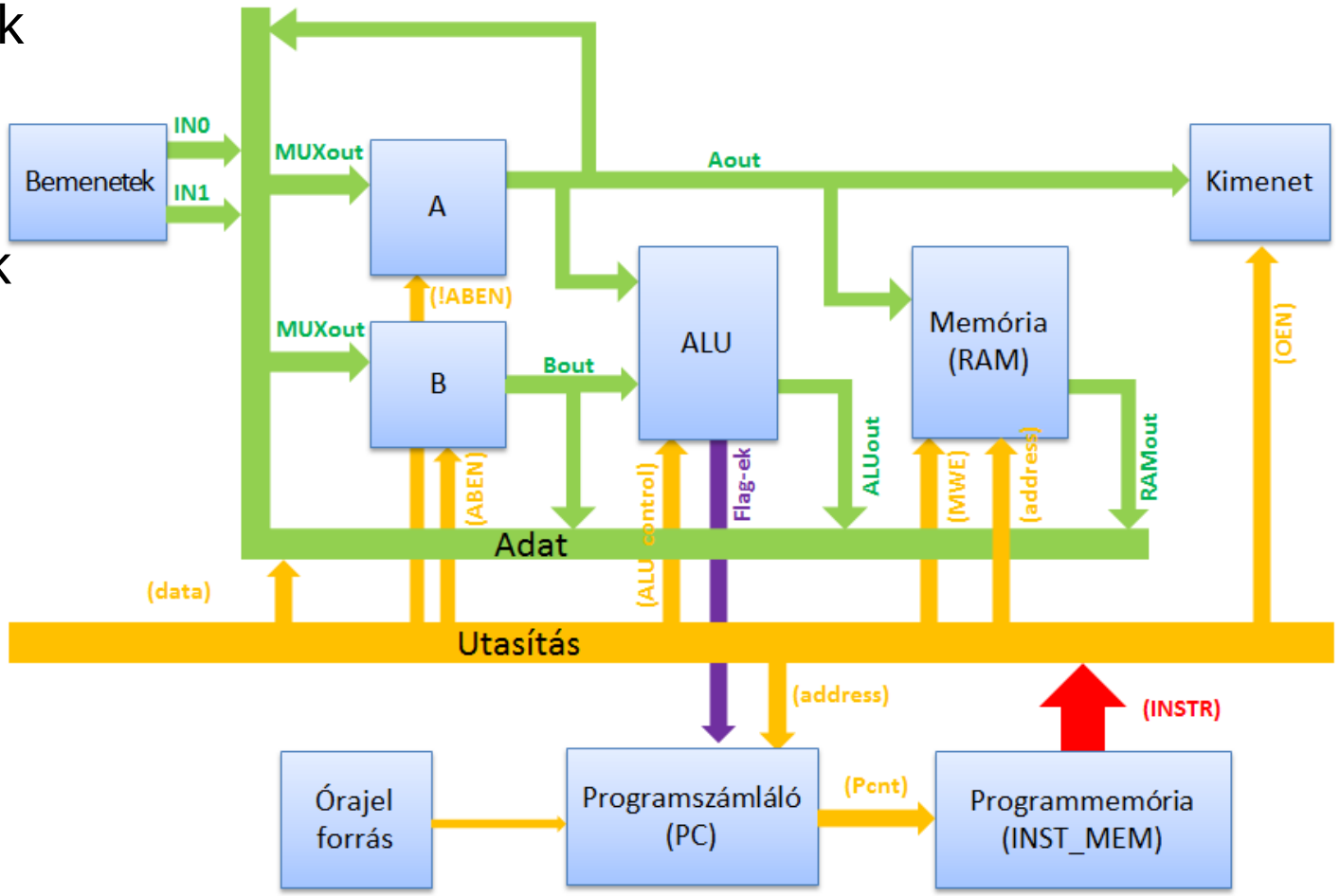
CPU részei

- ALU (Arithmetic Logic Unit)
  - CPU "kalkulátora”
  - Néhány alapvető műveletet képes végrehajtani
    - Összeadás, kivonás, átvitel bitek kezelése
    - Fixpontos szorzás osztás
    - Logikai műveletek
    - Léptetések, bitek mozgatása jobbra/balra
    - Lebegőpontos aritmetikai műveletek
- Akkumulátor
  - Ideiglenes tárolást (munkamemóriát) biztosít(anak) az ALU számára
- Egyéb regiszterek
  - A CPU belső tárolóelemei
  - Írásuk és olvasásuk sokkal gyorsabb a memóriákénál
  - Segítik a címképzést, tárolnak állapotjellemzőket, státusokat (ezzel a vezérlést segítik)
  - Tartalmuk gyorsan és egyszerűen elérhető a CPU elemei számára

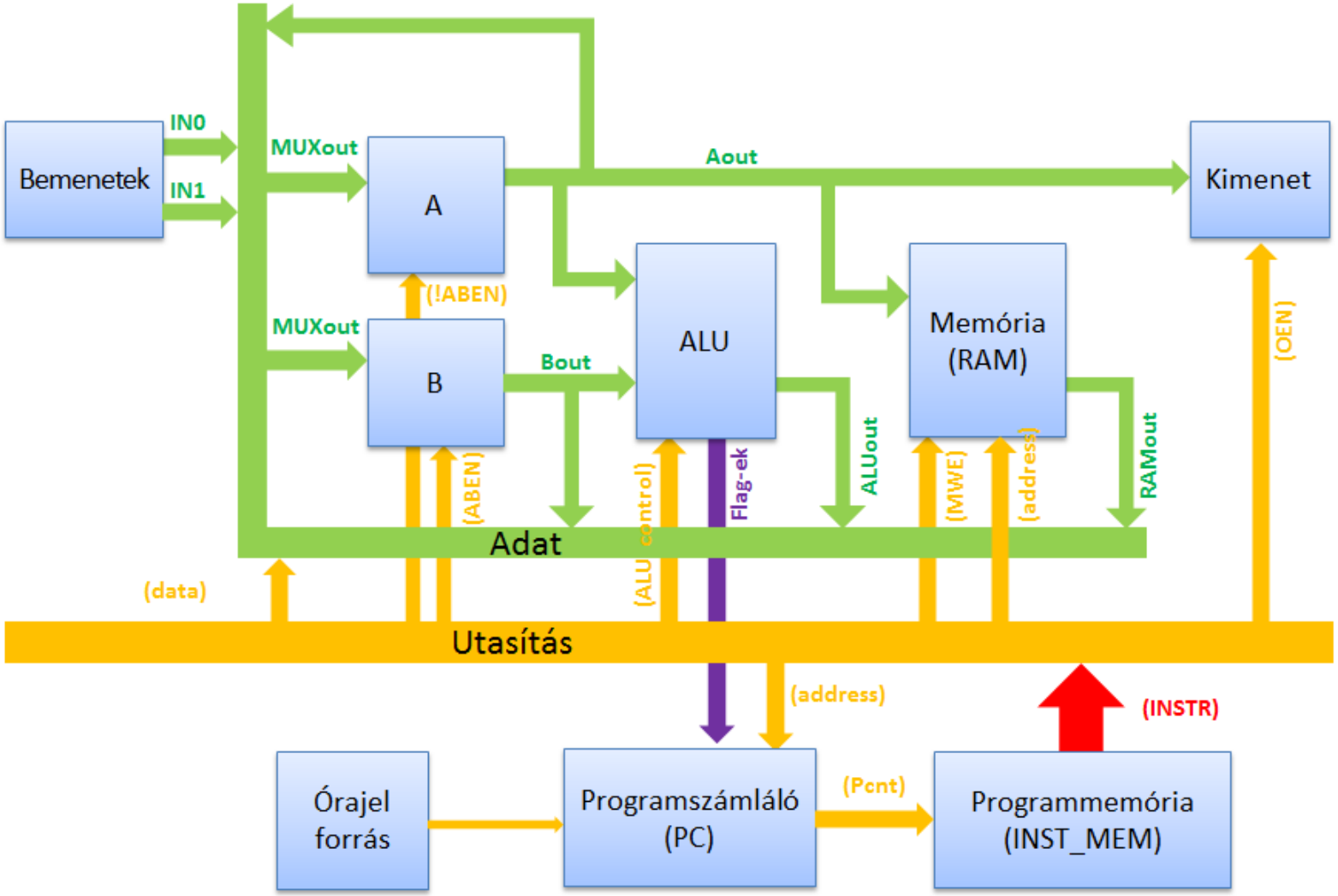


CPU felépítése

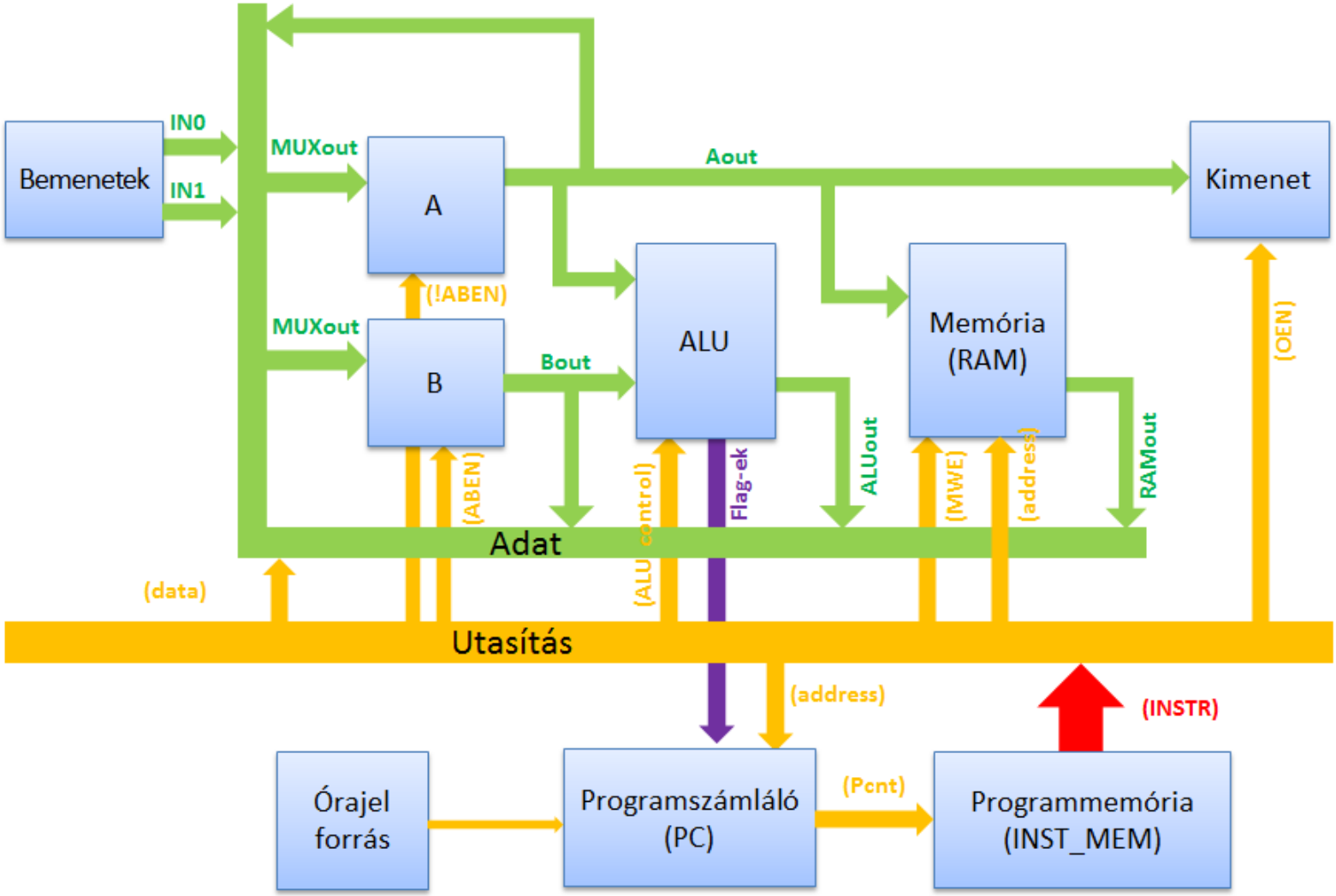
- Programszámláló
- Programmemória
- Regiszterek
- ALU
- RAM
- Bemenetek
- Kimenetek



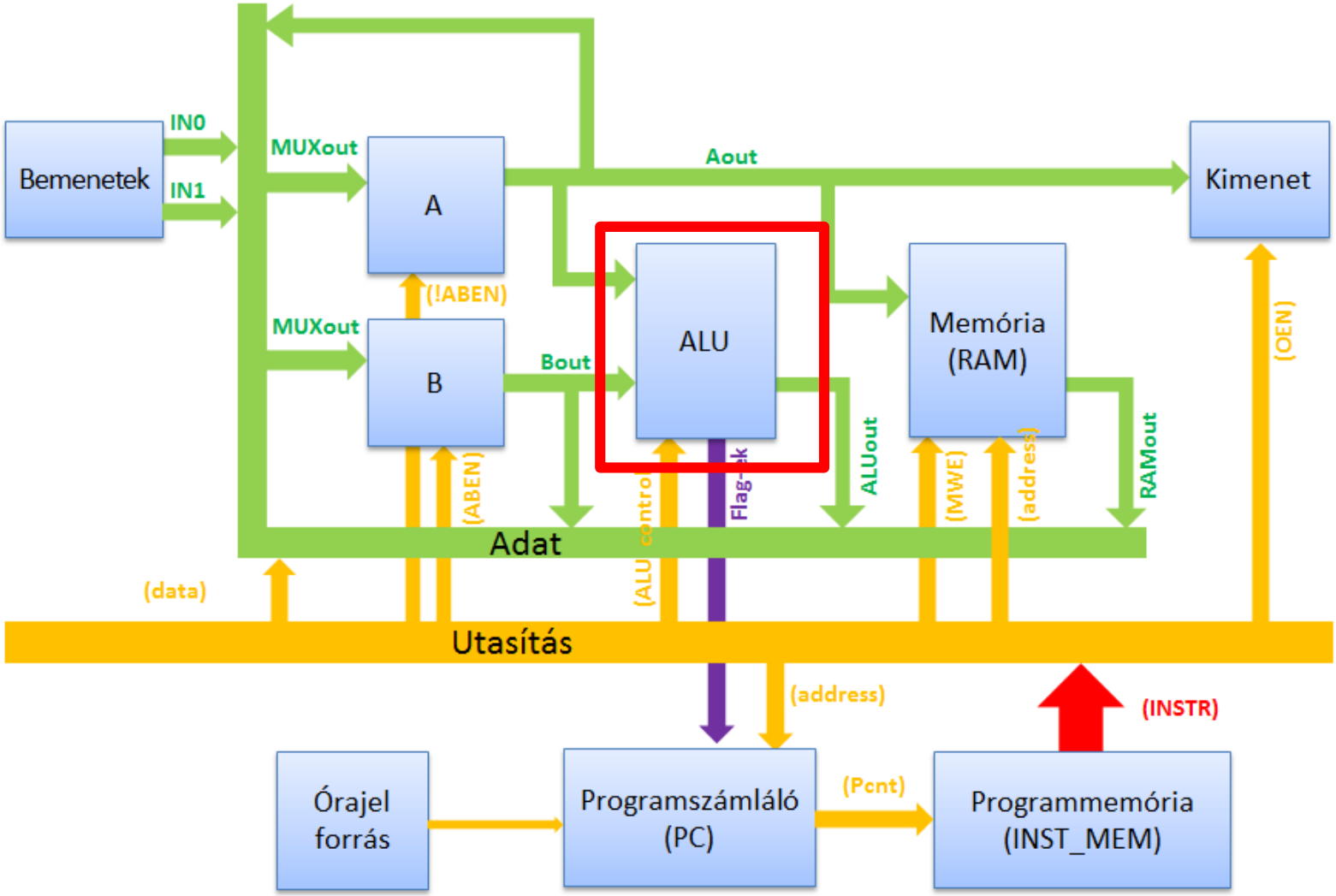
Milyen architektúra?



CPU


Bemenetek
IN0
IN1
MUXout
A
Aout
Kimenet
(!ABEN)
MUXout
B
Bout
ALU
Memória
(RAM)
(OEN)
(ABEN)
(ALU control)
Flag-ek
ALUout
(MWE)
(address)
RAMout
Adat
(data)
Utasítás
(address)
(INSTR)
Órajel
forrás
Programszámláló
(PC)
(Pcnt)
Programmemória
(INST_MEM)

CPU


Bemenetek
IN0
IN1
MUXout
A
Aout
Kimenet
(!ABEN)
MUXout
B
Bout
ALU
Memória
(RAM)
(OEN)
(ABEN)
ALUout
(MWE)
(address)
RAMout
Adat
(data)
Utasítás
(address)
(INSTR)
Órajel
forrás
Programszámláló
(PC)
(Pcnt)
Programmemória
(INST_MEM)

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.

Aritmetikai egységek

- Digitális komparátor
- Összeadó
- Kivonó
- szorzó

Digitális komparátor

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

Digitális komparátor

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

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

Digitális komparátor

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

Digitális komparátor

Y_0 = 1 ha A > B egyébként 0

Digitális komparátor

Y_0 = 1 ha A > B egyébként 0

	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

Digitális komparátor

Y_0 = 1 ha A > B egyébként 0

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

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

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

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

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

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

	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

Digitális komparátor

Y_0 = 1 ha A > B egyébként 0

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

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

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

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

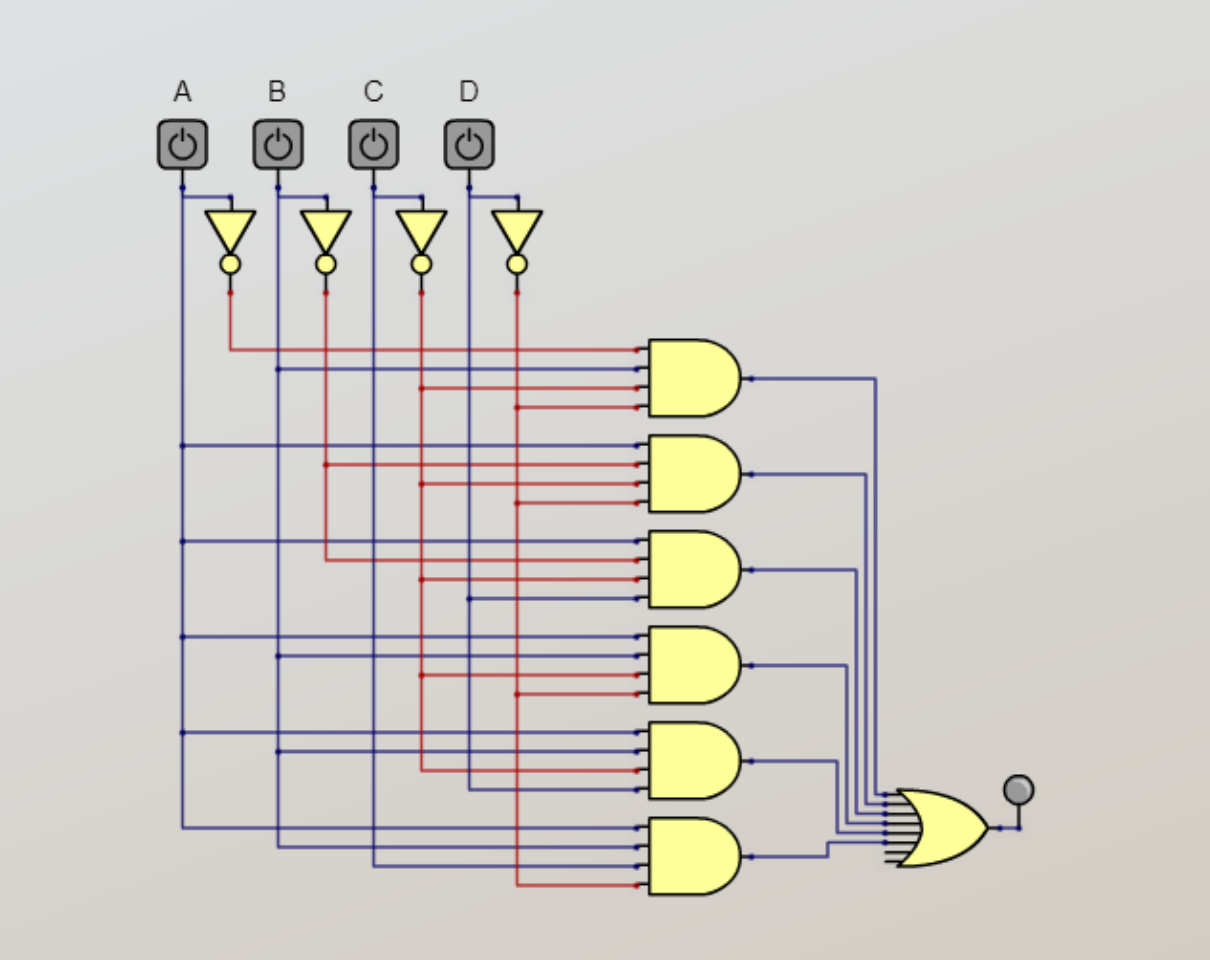
$$Y5 = AB\overline{C}D$$

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

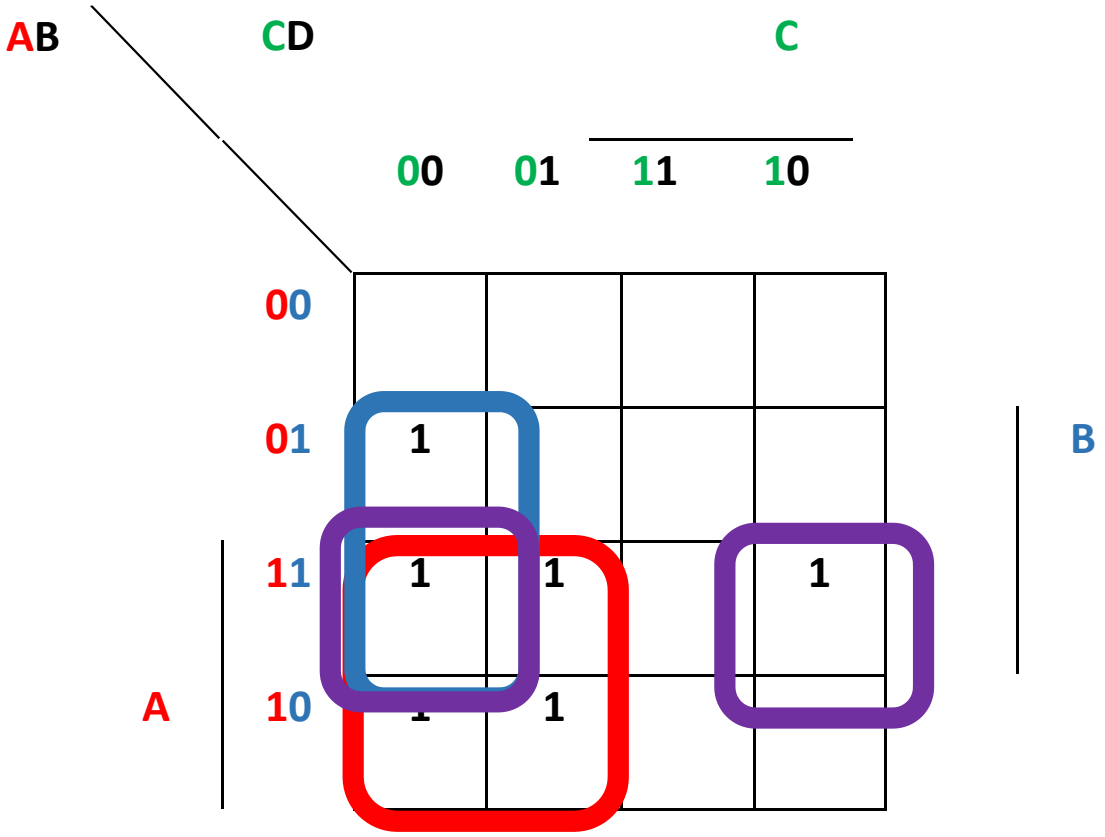
$$Y = \overline{A}B\overline{C}\overline{D} + A\overline{B}\overline{C}\overline{D} + A\overline{B}C\overline{D} + AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D}$$

	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
	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

Digitális komparátor



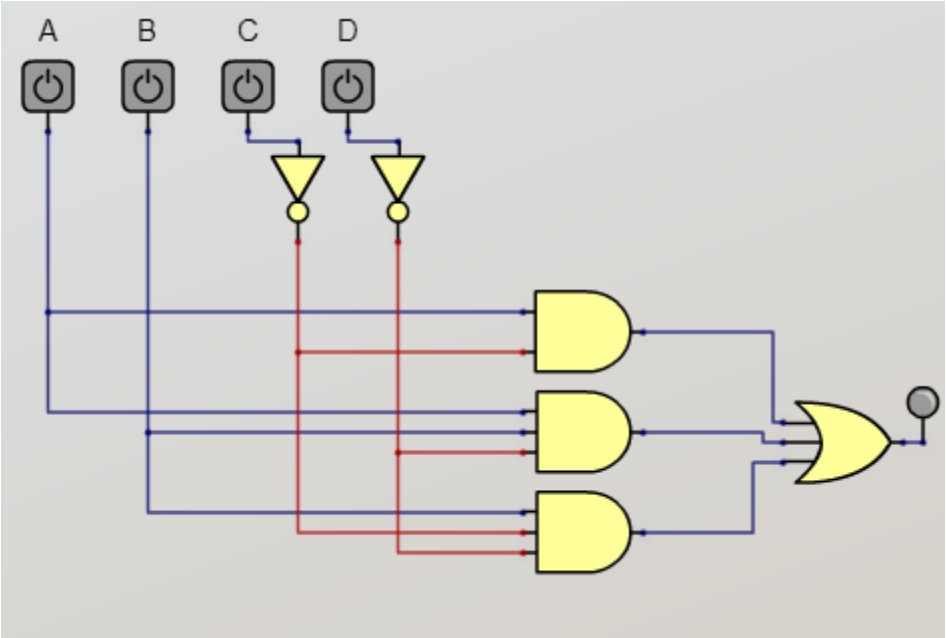
Digitális komparátor



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

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

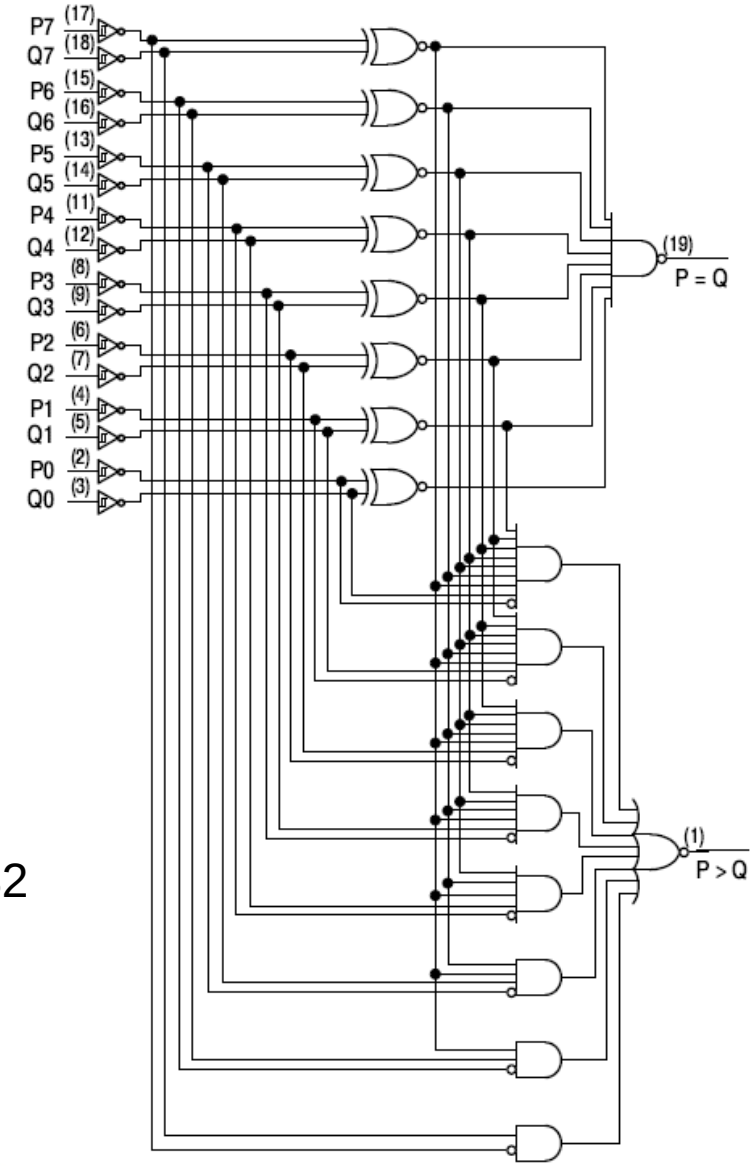
Digitális komparátor



Digitális komparátor

- Bináris számok összehasonlítása

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



74LS682

Számlálás

Decimálisan:

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

Binárisan:

0000	
0001	
0010	←átvitel / túlcsordulás
0011	
0100	←átvitel / túlcsordulás
0101	
0110	
0111	
1000	←átvitel / túlcsordulás
1001	
1010	
1011	
1100	
1101	
1110	
1111	

Összeadó

Decimális számok összeadása

- Papíron végzett összeadásnál, jobbról balra haladva sorra összeadjuk az egyes számjegyeket
- Ahol kilencnél nagyobb eredményt kapunk, ott hozzáadjuk a maradék egyest az egyel nagyobb helyiértékű számjegyekhez

```
   2   5     6
+  3   1     7
-------------
   5   7 (1) 3
```

Összeadó

Decimális számok összeadása

- Papíron végzett összeadásnál, jobbról balra haladva sorra összeadjuk az egyes számjegyeket
- Ahol kilencnél nagyobb eredményt kapunk, ott hozzáadjuk a maradék egyest az egyel nagyobb helyiértékű számjegyekhez

```
   2   5     6
+  3   1     7
-----------------
   5   7 (1) 3
```

Bináris számok összeadása

- Kettes számrendszerben ugyan így járunk el, csak a felhasználható számjegyek 0 és 1
- Ahol 1-nél nagyobb eredményt kapunk, ott hozzáadjuk a maradék egyest az egyel 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)
```

Műveletek

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

Kivonás	Példa
$0 - 0 = 0$ $0 - 1 = -1$ $1 - 0 = 1$ $1 - 1 = 0$	$1011_2 - 111_2 = 100_2$

Műveletek

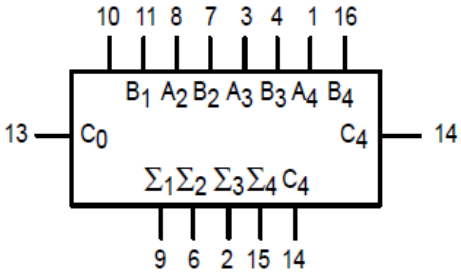
Szorzás	Példa
0 . 0 = 0 0 . 1 = 0 1 . 0 = 0 1 . 1 = 1	1010_2 $\cdot\ 11_2$ 11110_2

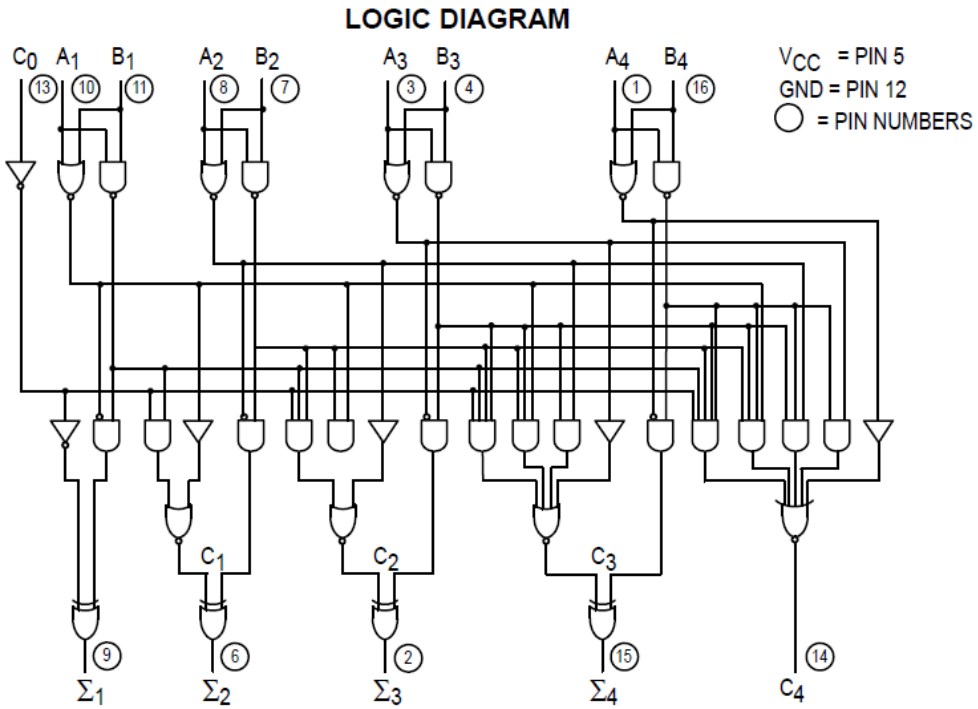
Osztás	Példa
0 / 0 = n.def. 0 / 1 = 0 1 / 0 = n.def. 1 / 1 = 1	1010_2 $/\ 10_2$ 101_2

Összeadó példa

Pl.: 4-bites teljes összeadó

74LS83A





	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)

Összeadó

Egybites teljes összeadó

- Két bináris szám egy-egy bitjét adja össze
- Figyelembe veszi az egyel 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

Összeadó

Egybites teljes összeadó

- Két bináris szám egy-egy bitjét adja össze
- Figyelembe veszi az egyel 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

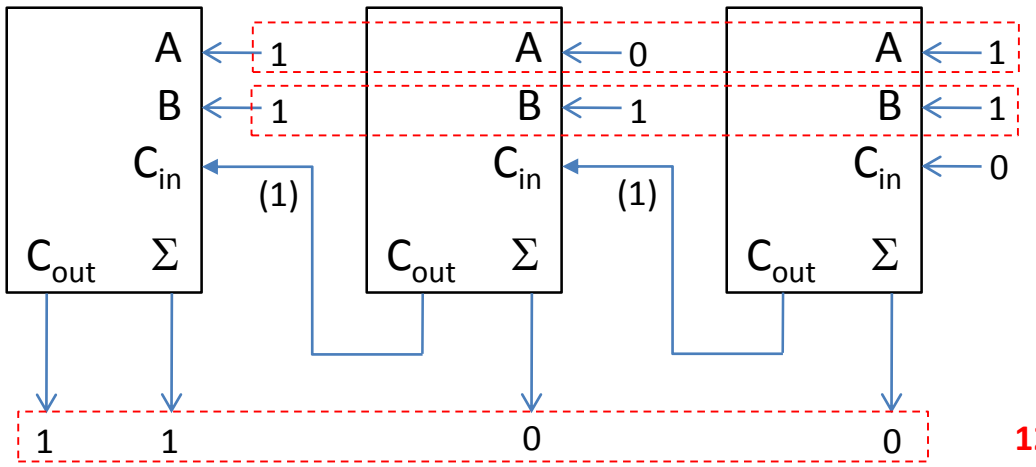
C_{in}	A	B	Σ	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Összeadó

Egybites teljes összeadó

- Két bináris szám egy-egy bitjét adja össze
- Figyelembe veszi az egyel 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

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



Kivonó

- Kettes komplemens

$$8 - 5 = 3$$

Kivonó

- Kettes komplemens

$$8 - 5 = 3$$

8:1000
5:0101

Kivonó

- Kettes komplemens

$$8 - 5 = 3$$

1) Negálás

8:1000
5:0101

5:1010

Kivonó

- Kettes komplemens

$$8 - 5 = 3$$

1) Negálás 2) +1

8:1000
5:0101

5:1010 **5:1011**

Kivonó

- Kettes komplemens

$$8 - 5 = 3$$

8:1000
5:0101

1) Negálás **5:1010**

2) +1 **5:1011**

3) összeadás

5:1011
8:1000

10011

Kivonó

- Kettes komplemens

$$8 - 5 = 3$$

8:1000
5:0101

1) Negálás **5:1010**

2) +1 **5:1011**

3) összeadás

5:1011
8:1000

10011

4) Egyes nem kell

10011

Kivonó

- Kettes komplemens

$$8 - 5 = 3$$

8:1000
5:0101

1) Negálás **5:1010**

2) +1 **5:1011**

3) összeadás

5:1011
8:1000

10011

4) Egyes nem kell

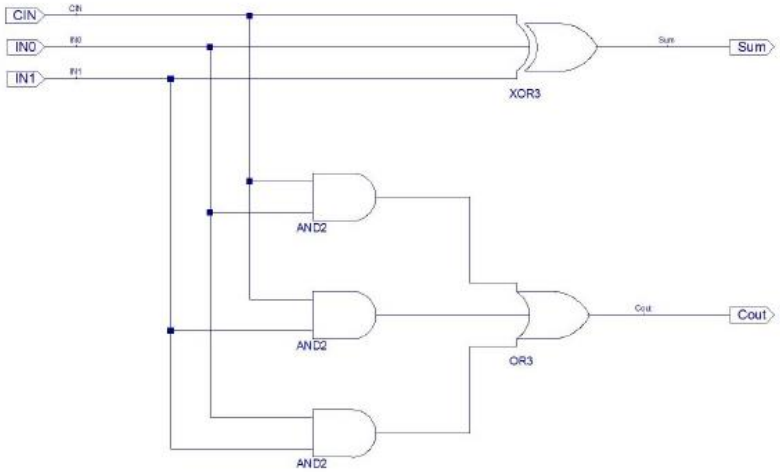
10011

Eredmény

0011 = 3

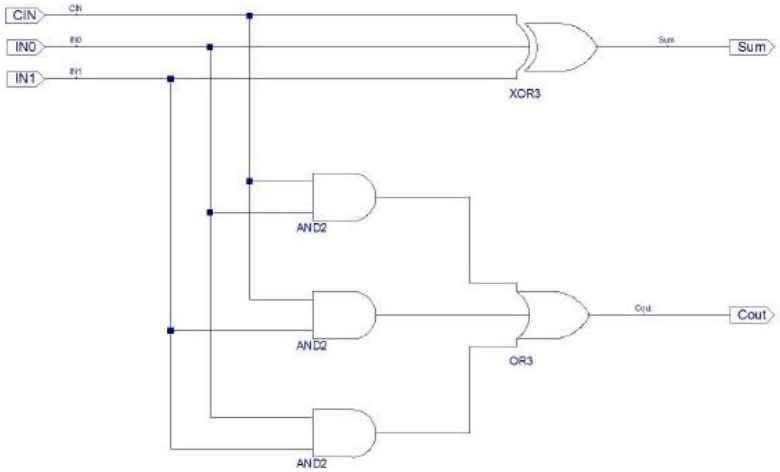
Emlékeztető: A korábbi laborgyakorlatokon megoldott aritmetikai és logikai műveletek

1 bites teljes összeadó

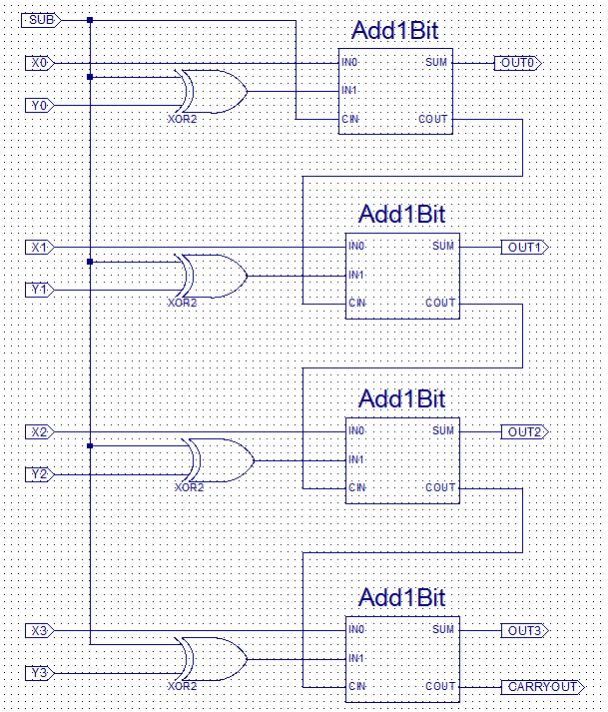


Emlékeztető: A korábbi laborgyakorlatokon megoldott aritmetikai és logikai műveletek

1 bites teljes összeadó

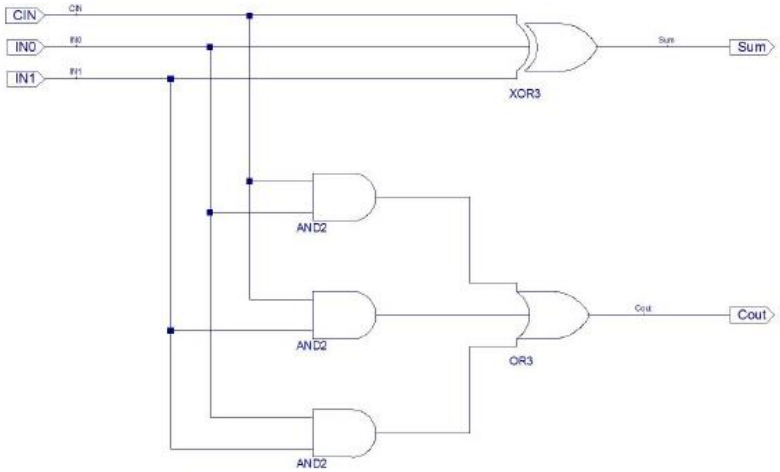


4 bites összeadó/kivonó áramkör

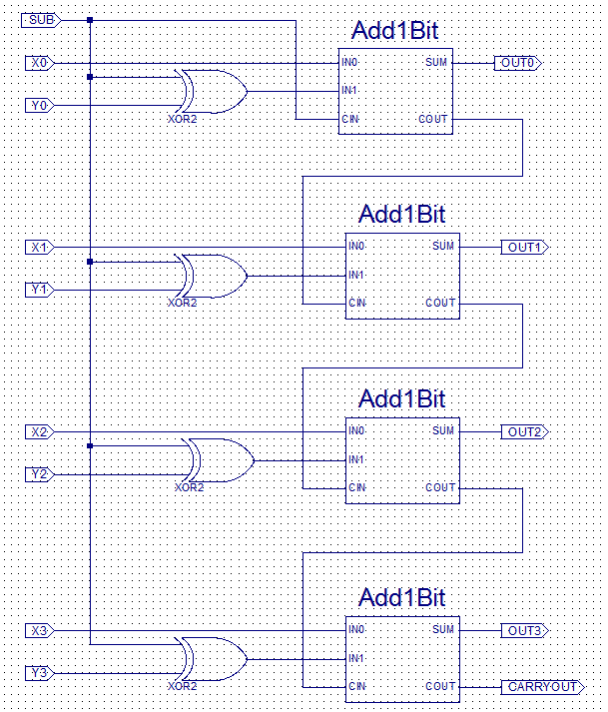


Emlékeztető: A korábbi laborgyakorlatokon megoldott aritmetikai és logikai műveletek

1 bites teljes összeadó

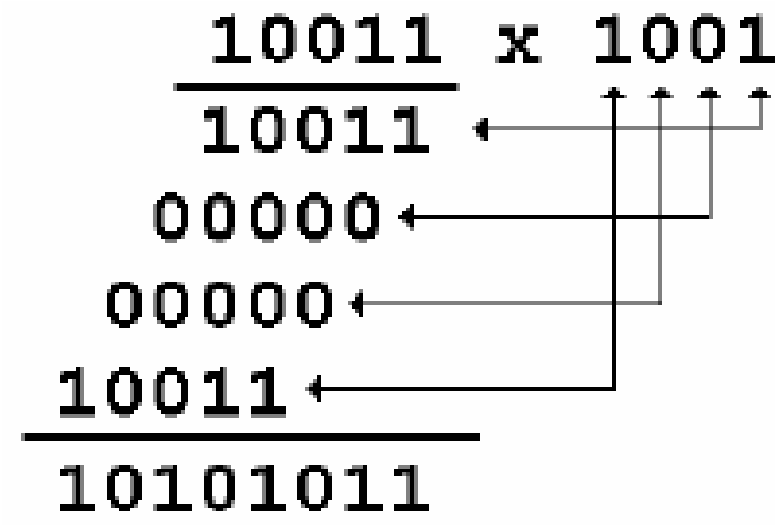


4 bites összeadó/kivonó áramkör

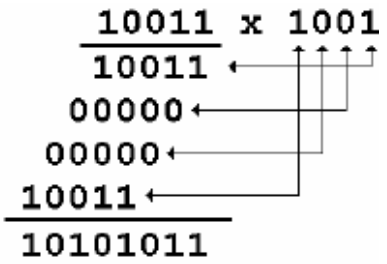


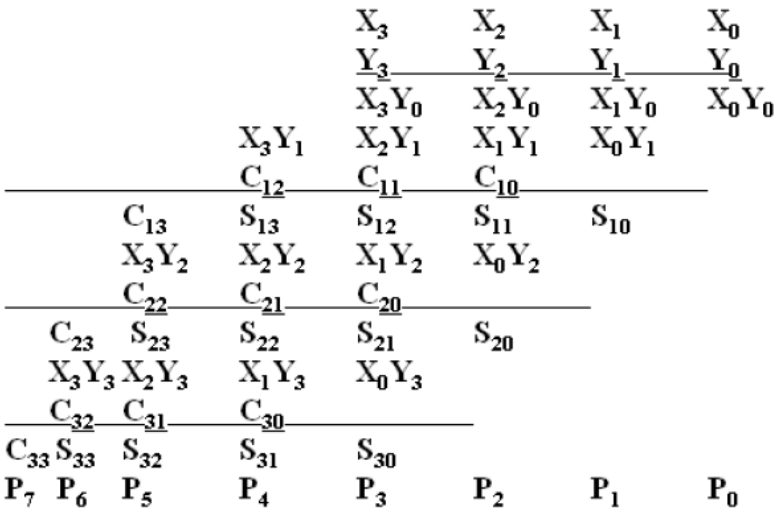
Két N bites szám összege egy N+1 bites számot eredményez!

Szorzás



Szorzás

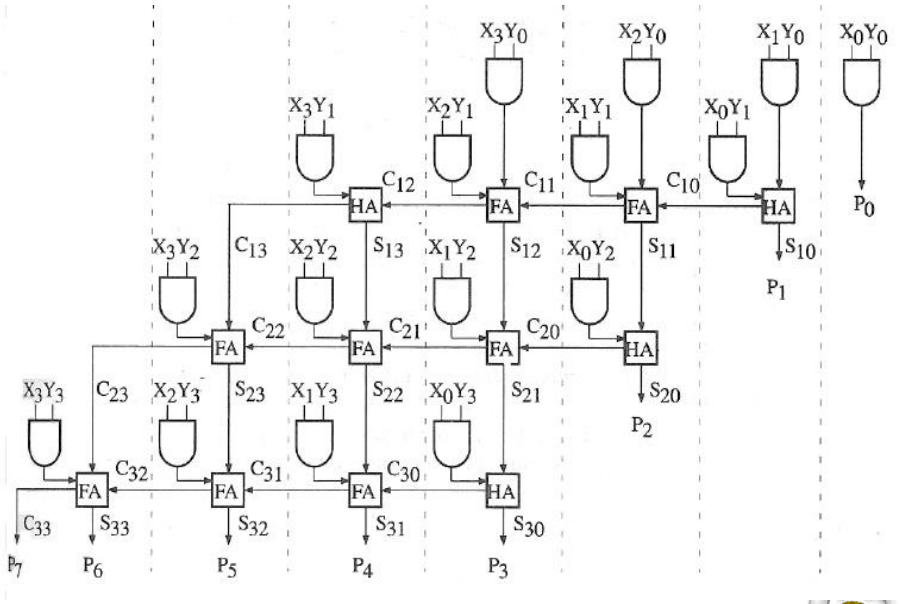




Szorzás

```
                                  X3      X2      X1      X0
                                  Y3      Y2      Y1      Y0
                                 X3Y0    X2Y0    X1Y0    X0Y0
                          X3Y1   X2Y1    X1Y1    X0Y1
                          C12    C11     C10
                  C13     S13    S12     S11     S10
                  X3Y2    X2Y2   X1Y2    X0Y2
                  C22     C21    C20
          C23     S23     S22    S21     S20
          X3Y3    X2Y3    X1Y3   X0Y3
          C32     C31     C30
C33       S33     S32     S31    S30
P7  P6    P5      P4      P3     P2      P1      P0
```

```
10011 x 1001
10011
00000
00000
10011
10101011
```

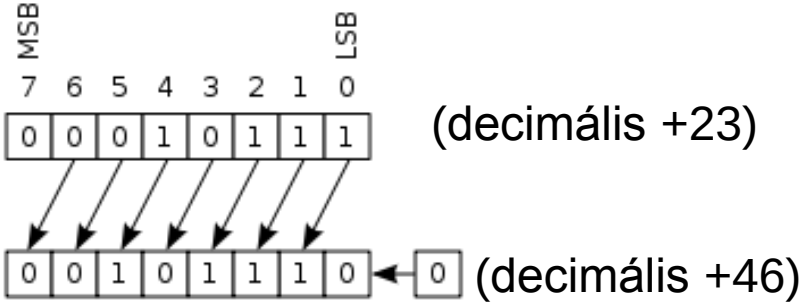


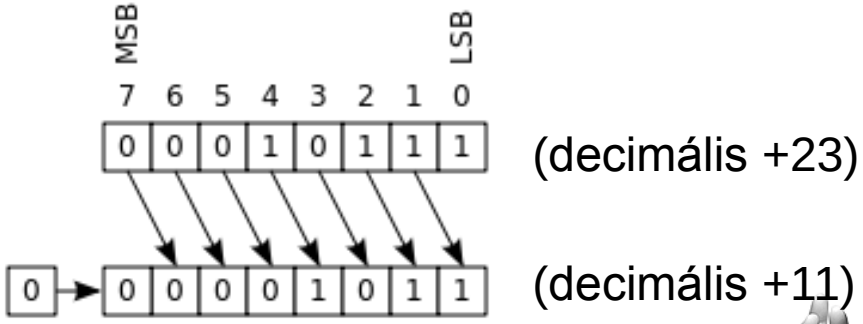
Két N bites szám szorzata egy 2N bites számot eredményez!

Léptetés/Eltolás

Alkalmazásuk:

- adatbeolvasás, pufferelés
- soros-párhuzamos átalakítás
- 2-vel való szorzás/osztás

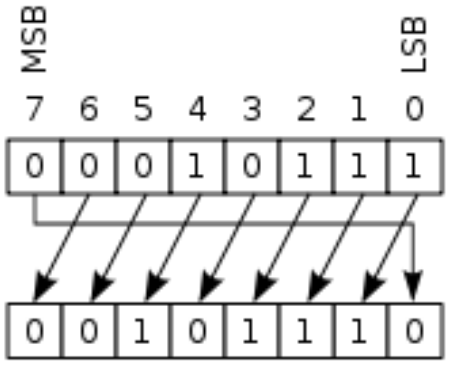


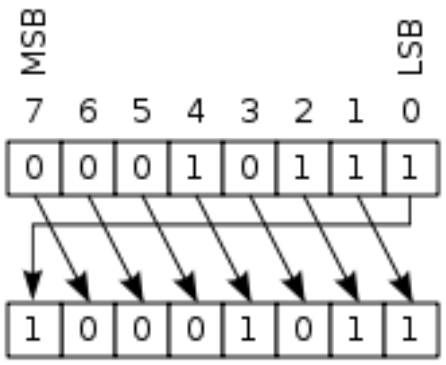


Körkörös eltolás

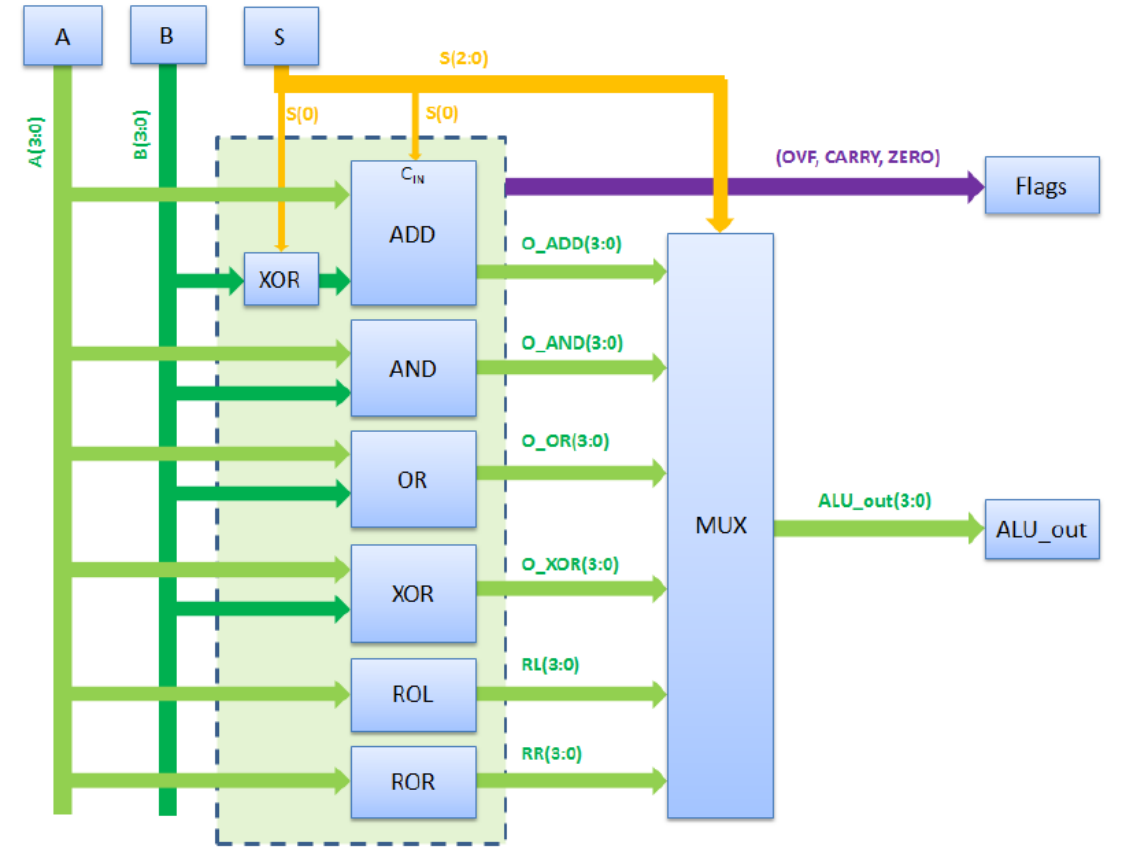
Alkalmazásuk:

- Speciális számlálók pl. Johnson számláló





Egy egyszerűsített ALU



Az ALU részei

- **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 ALU kimeneti jelei a vezérlőjeleinek függvényében

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	-	-

Az ALU flag bitjei

- **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.

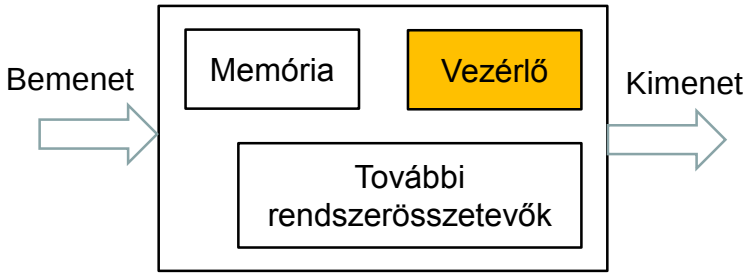
AGU

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

CU

- 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.

CU



- Vezérlő egység
  - A vezérelt rendszer kívánt algoritmus szerinti működését
    - Felügyelik
    - Szervezik
  - A vezérlő működési programja lehet
    - Fázisregiszteres
      - Kötött struktúrájú
      - A fázisregiszter (ez lehet egy számláló is) tárolja a vezérlő belső állapotát
      - A belső állapotnak megfelelően kell előállítani a kimeneti vezérlő jeleket
      - A módosítás 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

Regiszter

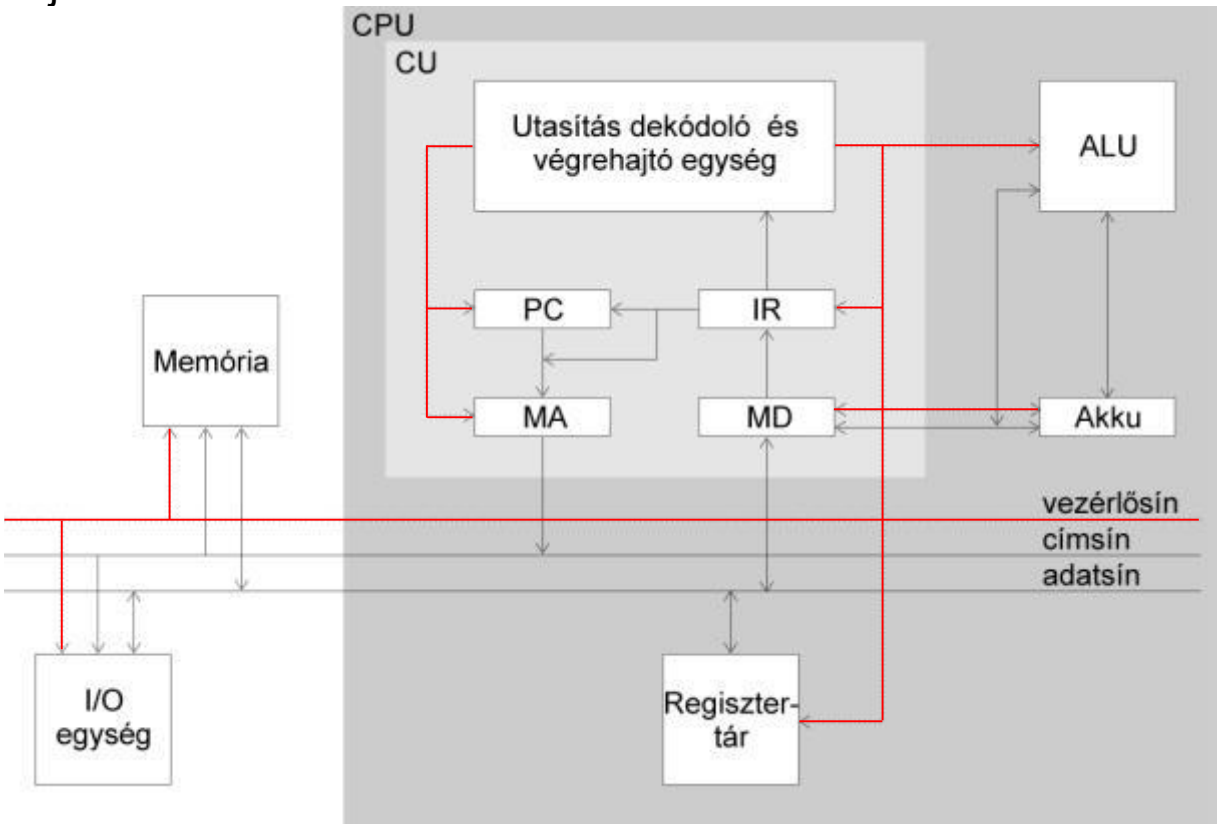
- Regiszter (Register): A regiszter a processzorba beépített nagyon gyors elérésű, kis méretű memória. A regiszterek addig (ideiglenesen) tárolják az információkat, utasításokat, amíg a processzor dolgozik velük. A mai gépekben 32/64 bit méretű regiszterek vannak. A processzor adatbuszai mindig akkorák, 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 :
  - utasításszámláló, (PC=program counter, IP=instruction pointer) ami mindig a következő végrehajtandó utasítás címét,
  - utasításregiszter (IR=instruction register), mely a memóriából kiolvasott utasítást tárolja. E kód alapján határozza meg a vezérlőegység az elvégzendő műveletet
  - flagregiszter, amely a processzor működése közben létrejött állapotok jelzőit (igaz, vagy hamis),
  - veremmutató (SP = Stack Pointer) ,
  - és az akkumulátor, (AC) amely pedig a logikai és aritmetikai műveletek egyik operandusát, majd az utasítás végrehajtása után az eredményt tartalmazza.

Buszvezérlő

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

Cím- adat- és vezérlősínek (buszrendszer)

- 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



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.

Mikroprocesszoros rendszerek

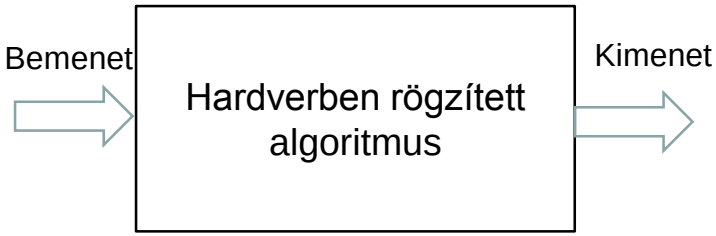
- Algoritmusok megvalósítása
  - Műszaki rendszereket mindig valamilyen feladat megoldása érdekében építünk
  - A feladatmegoldás általában valamilyen algoritmus szerint történik
    - Mérésadatgyűjtés
    - Adatok elemzése (pl. összehasonlítás)
    - Aritmetikai, logikai műveletek végzése az adatokon
    - Döntéshozatal stb.

Mikroprocesszoros rendszerek

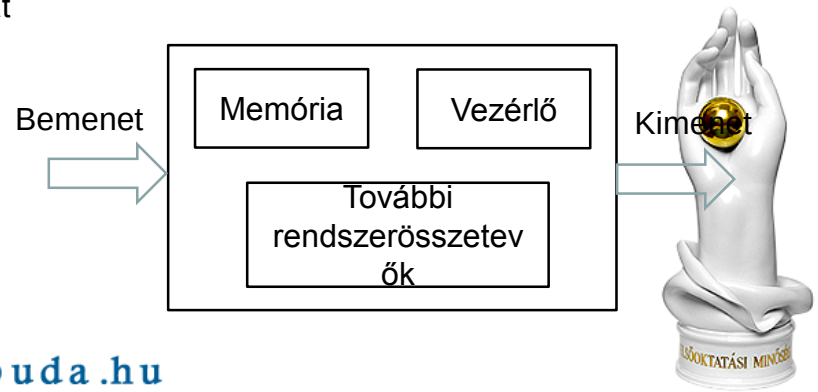
- Az információfeldolgozás menetét (programját) építjük be a rendszerbe
  - 1. megoldás:
    - A rendszer összetevői és egymáshoz való kapcsolódásuk, sorrendiségük a **hardverben fixen „behuzalozva” jelennek meg**
  - 2. megoldás
    - Az algoritmusnak megfelelő sorrendben, előre letárolt program szerint, egy vezérlő berendezés segítségével aktivizáljuk az egyes műveletvégző egységeket

Mikroprocesszoros rendszerek

- Algoritmusok megvalósítása
  1. megoldás: Huzalozott program
    - A rendszer összetevői és egymáshoz való kapcsolódásuk, sorrendiségük a hardverben fixen „behuzalozva” jelennek meg
      - Előny
        - Egyes részfeladatok párhuzamosan is végrehajthatók (gyors működés)
      - Hátrány
        - A hardver a rögzített struktúra miatt csak az adott feladat megoldására alkalmas

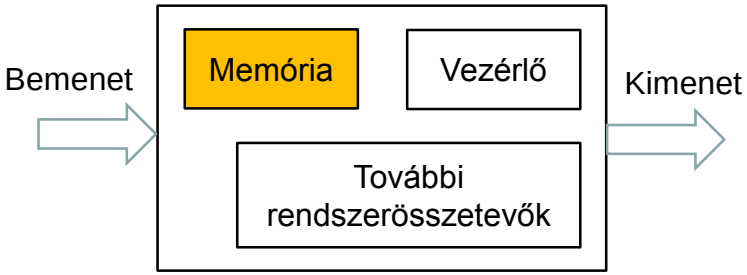


  2. megoldás: Tárolt program
    - Az algoritmusnak megfelelő sorrendben, előre letárolt program szerint, egy vezérlő berendezés segítségével aktivizáljuk az egyes műveletvégző egységeket
      - Előny
        - Ha megváltoztatjuk a memória tartalmát (a programot) más-más feladatra használhatjuk
      - Hátrány
        - A részfeladatok nehezen párhuzamosíthatók (lassúbb működés)
        - Szekvenciális utasítás végrehajtás



Memória

- Algoritmusok megvalósítása tárolt program alapján

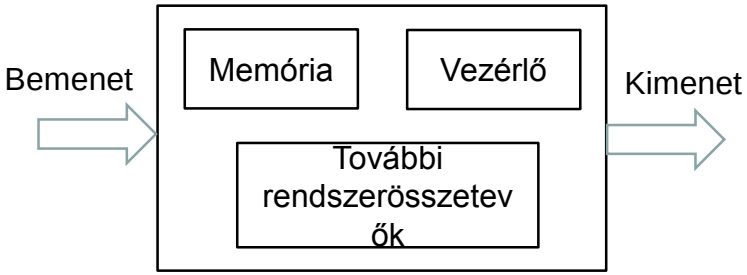


- Memória
  - A rendszer működése hatékonyabb, ha a memóriában nem csak az utasítások hanem adatok is tárolhatók későbbi felhasználás céljából
    - Programmemória
    - Adatmemória
  - Lehet közös memória (Neumann architektúra)
  - Vagy fizikailag külön adat és programmemória (Harvard architektúra)

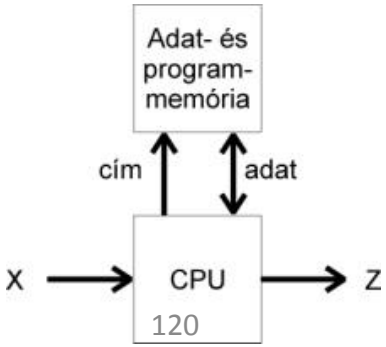
- Bemenetek- Kimenetek
  - A bementi adatok fogadására
  - A kimeneti adatok megjelenítésére
  - A külvilággal való kommunikációra, illesztésre

Mikroprocesszoros rendszerek

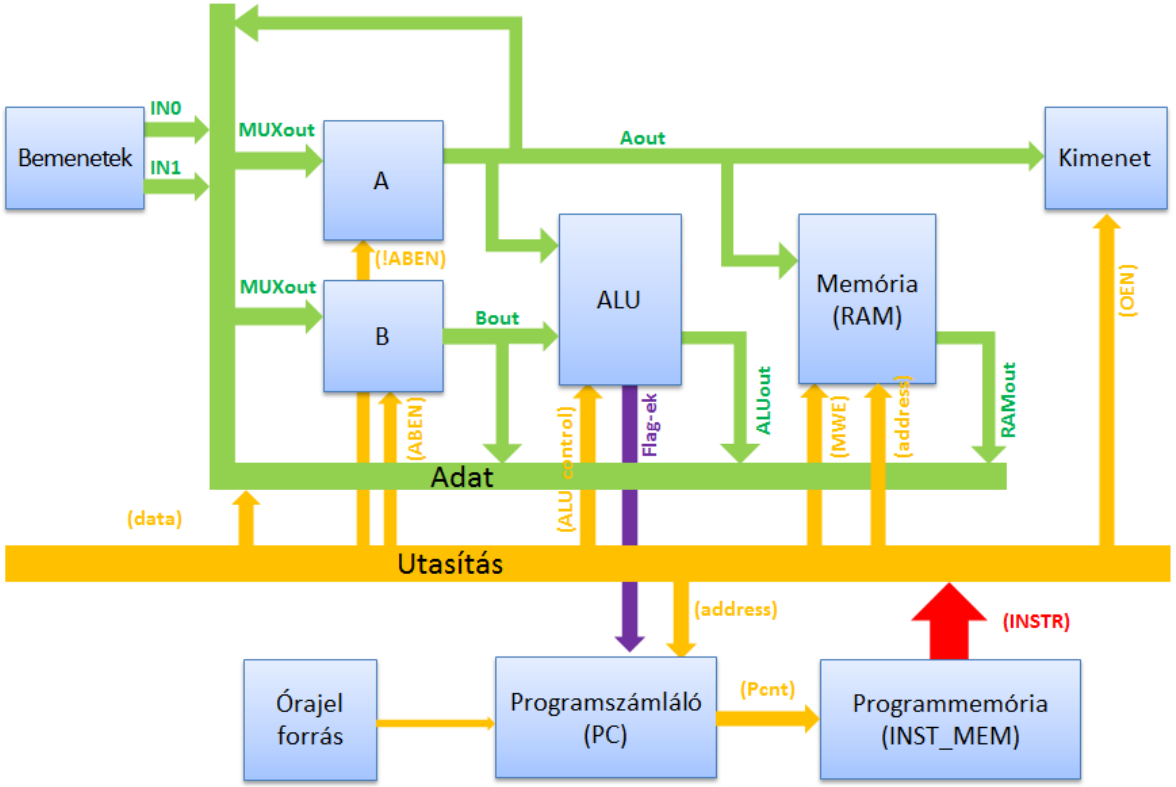
- Algoritmusok megvalósítása tárolt program alapján



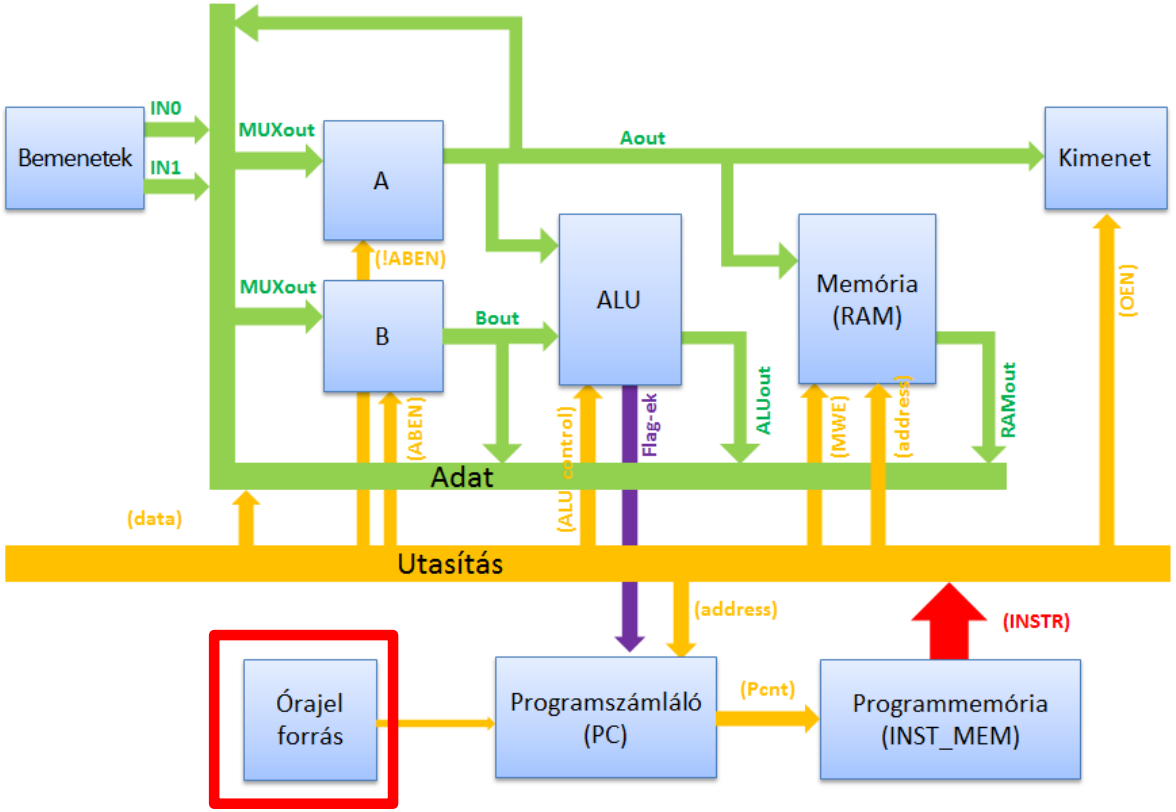
- Műveletvégző egységek
  - A gyakran használt aritmetikai és logikai műveletek végrehajtását célszerű külön erre a célra tervezett egységre bízni
    - ALU (Arithmetic Logic Unit)
  - A vezérlési, aritmetikai és logikai, illetve egyéb feldolgozási műveleteket elvégző központi egységet *CPU*-nak (*Central Processing Unit, központi feldolgozó egység*) hívjuk



Laborfeladat: 4 bites CPU tervezése és implementálása



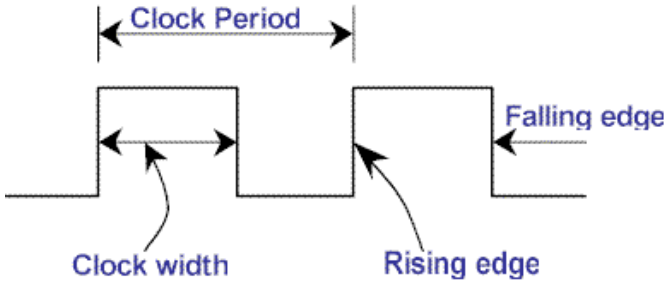
Órajel forrás



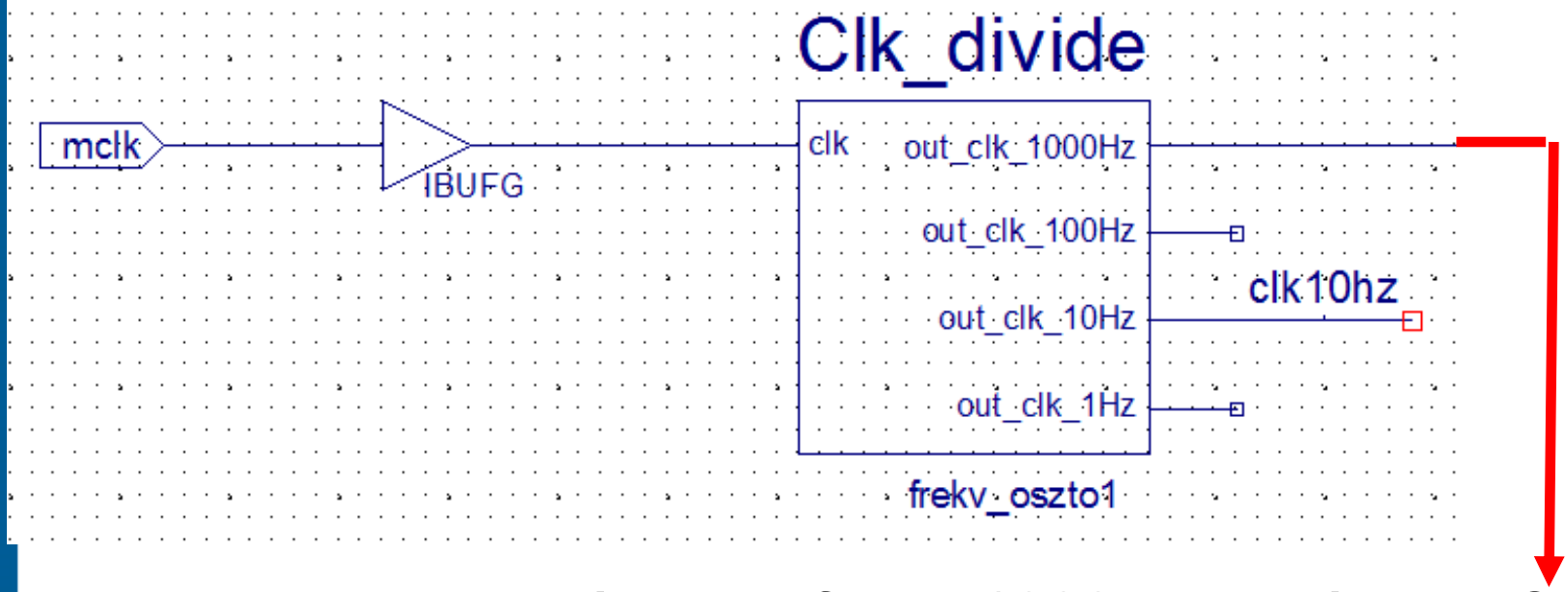
CPU órajelének előállítása

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

1. A CPU rendes működéséhez: Az FPGA 50 MHz-es rendszer-órajeléből a Clk_divide makró segítségével 10 Hz-es órajelet állítunk elő.
2. A CPU teszteléséhez: Az órajel fel- és lefutó éleit nyomógombok (UP és DOWN gombok) segítségével állítjuk elő.



A CPU 10 Hz-es órajelének előállítása a Clk_divide makró segítségével



Hol használhatjuk fel az 1000 Hz-es órajelet?

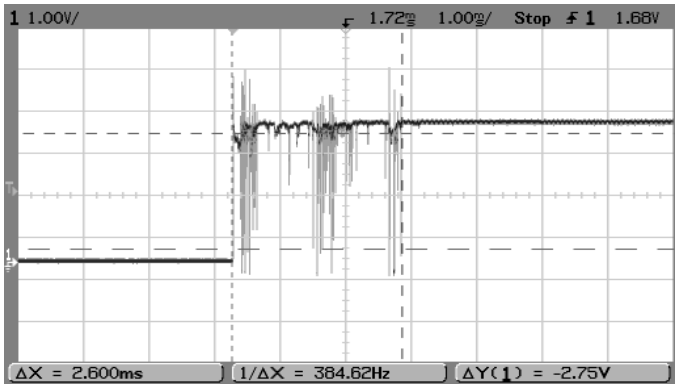
(Hétszegmenses kijelző vezérlésénél! Ld. előző laborgyakorlat!)

A CPU órajelének előállítása nyomógombok segítségével

Nyomógomb pergés/prellezés/pattogás jelensége:

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.



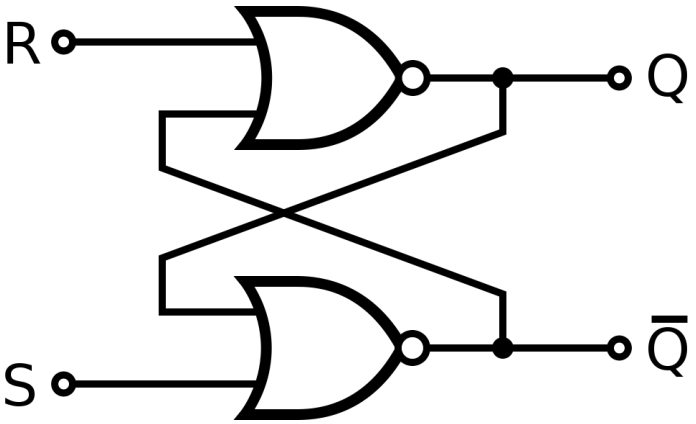
Ezt a jelenséget láthattuk az 1. laborgyakorlaton, a számláló áramkör kézi vezérlésekor.

Prellmentesítés (debouncing)

Megoldások:

- Speciális Hall generátoros vagy vezetőképes gumi nyomógombok alkalmazása
- Schmitt-trigger alkalmazása
- Lenyomás/kapcsolás idejének mérése, pergés idejéig változás letiltása

SR flip-flop



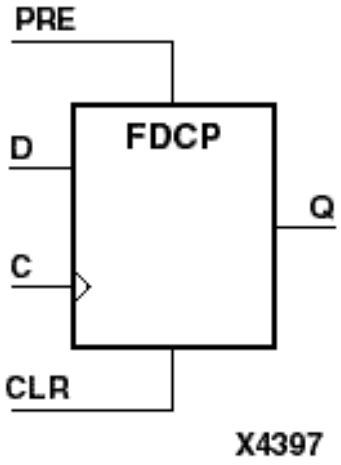
S	R	Q	$\bar{Q}$	
1	0	1	0	
0	0	1	0	S=1 és R=0 után
0	1	0	1	
0	0	0	1	S=0 és R=1 után
1	1	0	0	nem megengedett

A flip-flop tartja az értékét!

SR flip-flop (pergésmentesítés) megvalósítása D tárolóval

D tároló aszinkron preset és clear bemenetekkel

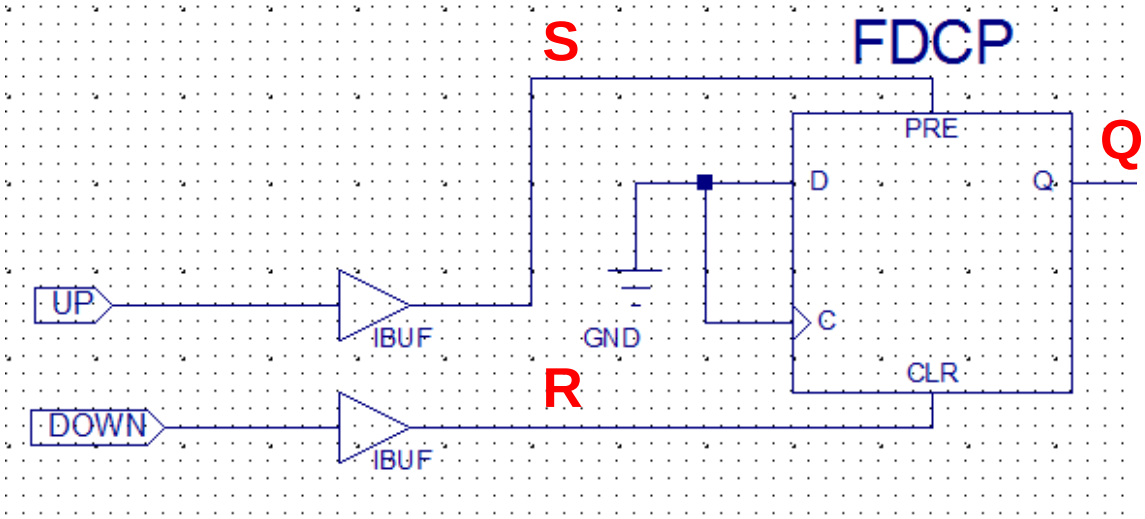
Inputs				Outputs
CLR	PRE	D	C	Q
1	X	X	X	0
0	1	X	X	1
0	0	0	↑	0
0	0	1	↑	1



- 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

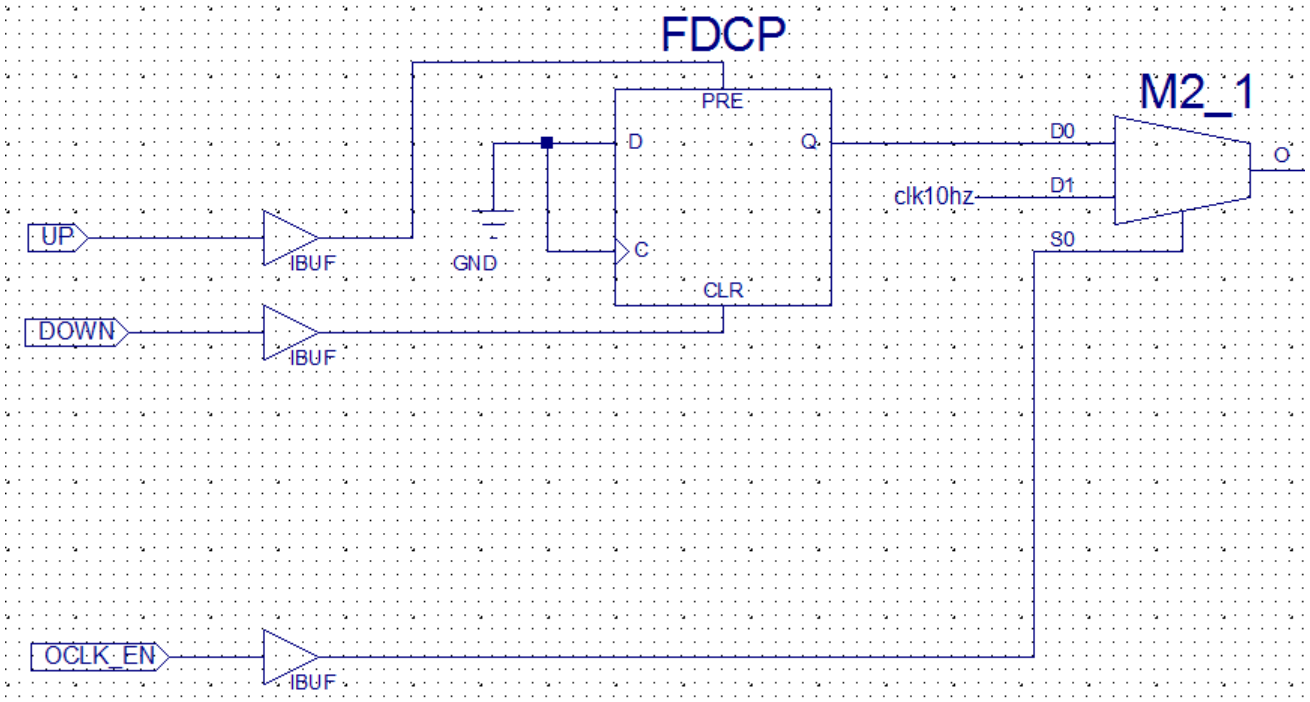
A CPU órajelének előállítása prellmentesített nyomógombok segítségével

S	R	Q	$\bar{Q}$	
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



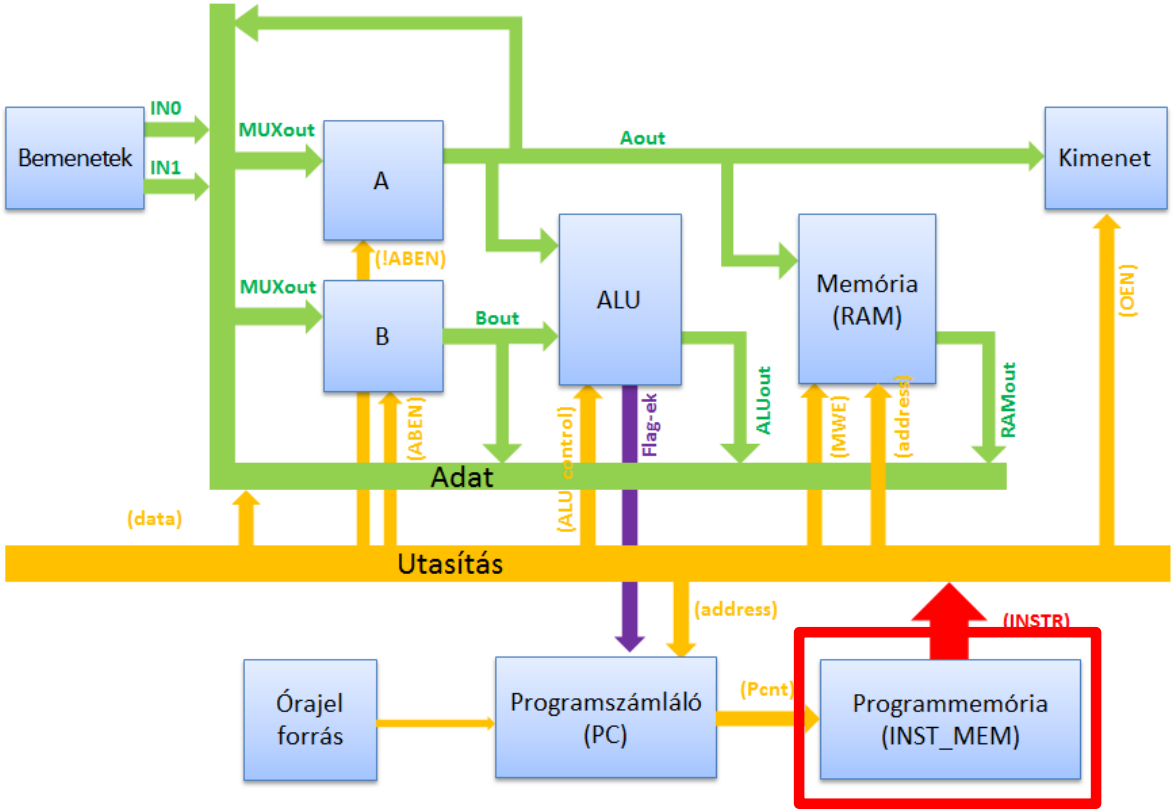
Megjegyzés: Az SR flip-flop-pal ellentétben itt megengedett az 11 bemenet (UP és DOWN gomb egyidejű megnyomása), mivel a CLR láb a többi bemenettől függetlenül 0 értékre állítja a Q kimenetet.

Választás a kétféle órajel között



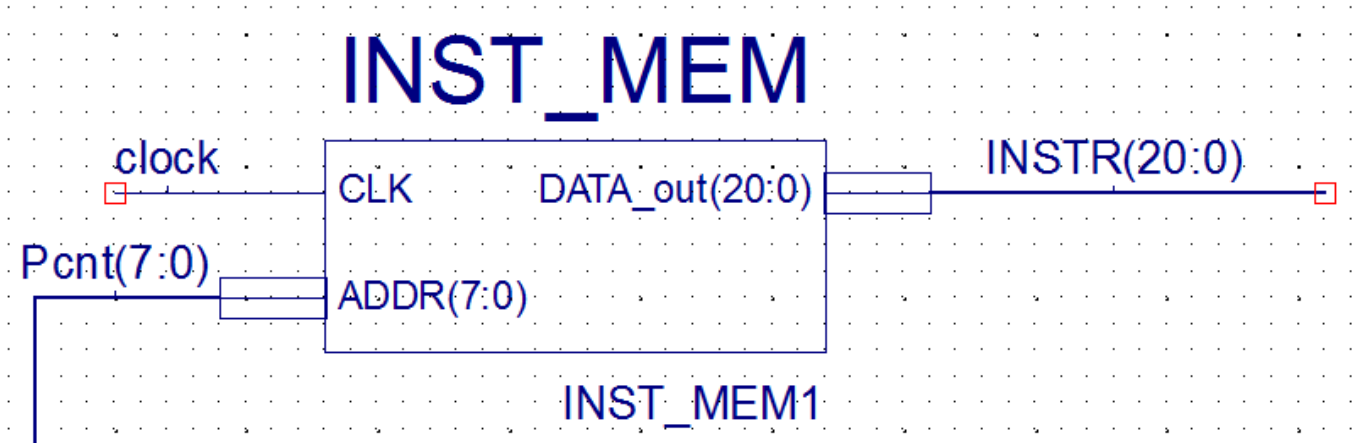
Az OCLK_EN nyomógomb dönti el (egy multiplexeren keresztül), hogy a 10 Hz-es vagy a kézi vezérlésű órajelet használjuk!

A programmemória (INST_MEM)



A programmemória bekötése

- 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.



A programmemóriában található kódrészlet

az utasítás helye a programmemóriában

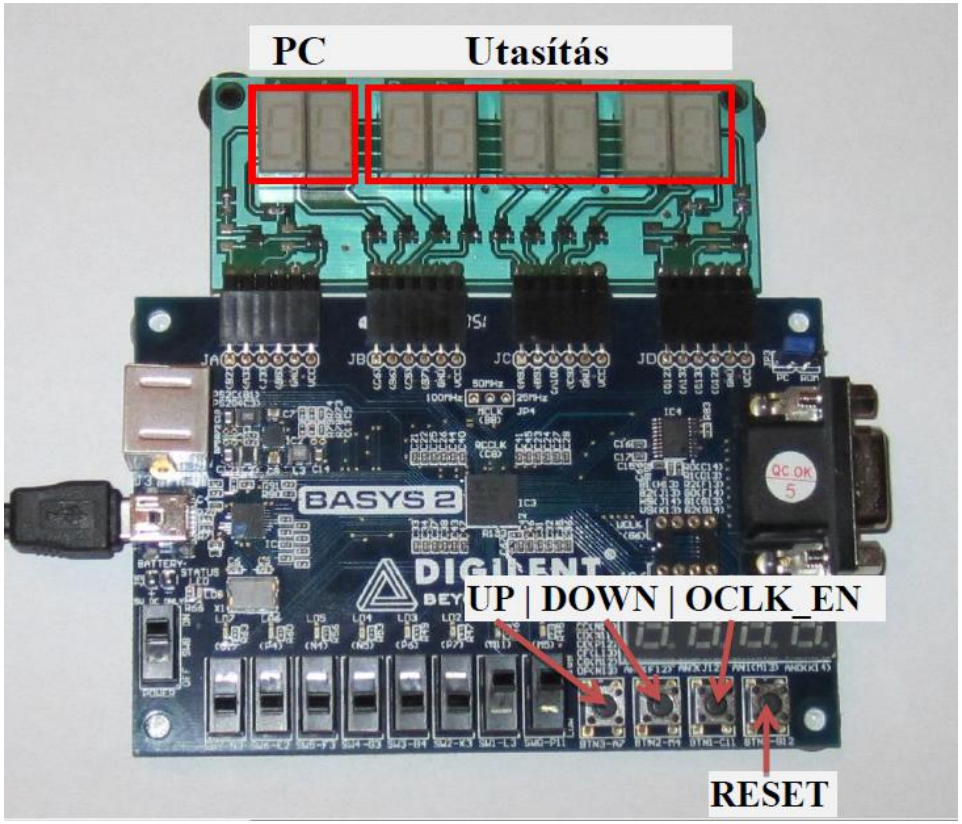
az utasítás mnemonikja

argumentum

az utasítás hexadecimális kódja

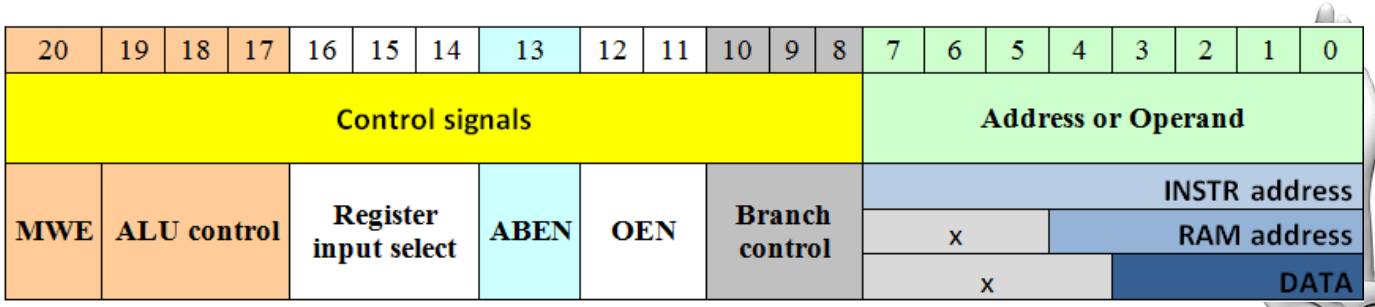
```
00      nop             opCode: 00000000
01      ldi     4       opCode: 0000c004
02      mvb             opCode: 00002000
03      ldi     5       opCode: 0000c005
04      add             opCode: 00018000
05      xor             opCode: 000d8000
06      sub             opCode: 00038000
07      ldi     7       opCode: 0000c007
08      and             opCode: 00058000
09      ldi     5       opCode: 0000c005
0a      or              opCode: 00098000
0b      ldi     2       opCode: 0000c002
0c      ror             opCode: 00078000
0d      ldi     2       opCode: 0000c002
0e      rol             opCode: 000b8000
0f      rol             opCode: 000b8000
```

Laborfeladat: 4 bites CPU tervezése és implementálása



Utasítás felépítése

- A CPU utasításokat hajt végre
- Vezérlőjelek
  - Vezérlik a processzor belső egységeit
- Argumentum
  - Az utasítás típusától függően lehet adat, vagy cím



Mnemonikok

- A vezérlőjelek egyszerű megadása
- 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)

ÓBUDAI EGYETEM

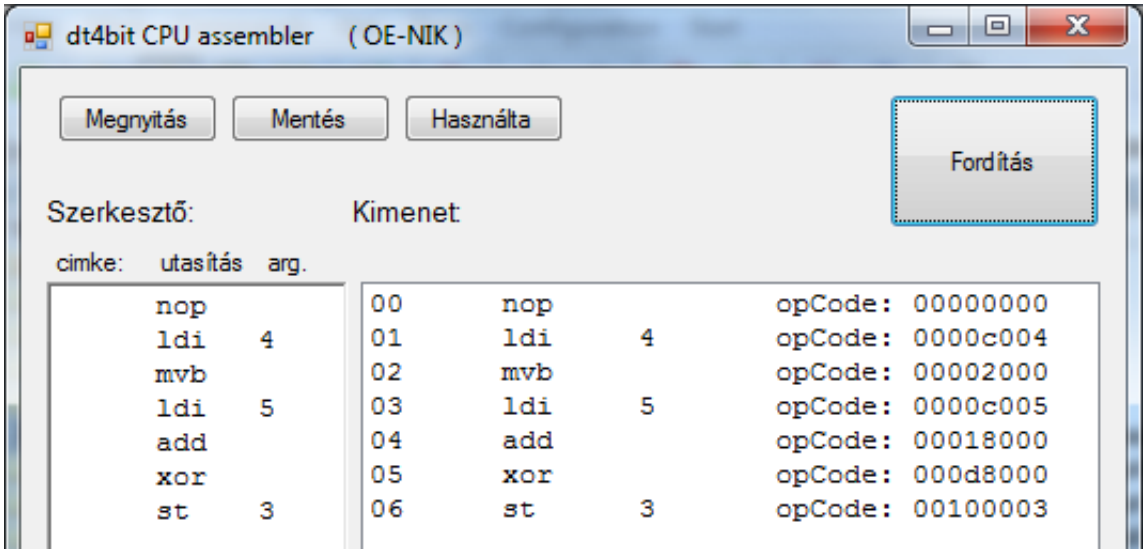
Utasításkészlet

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

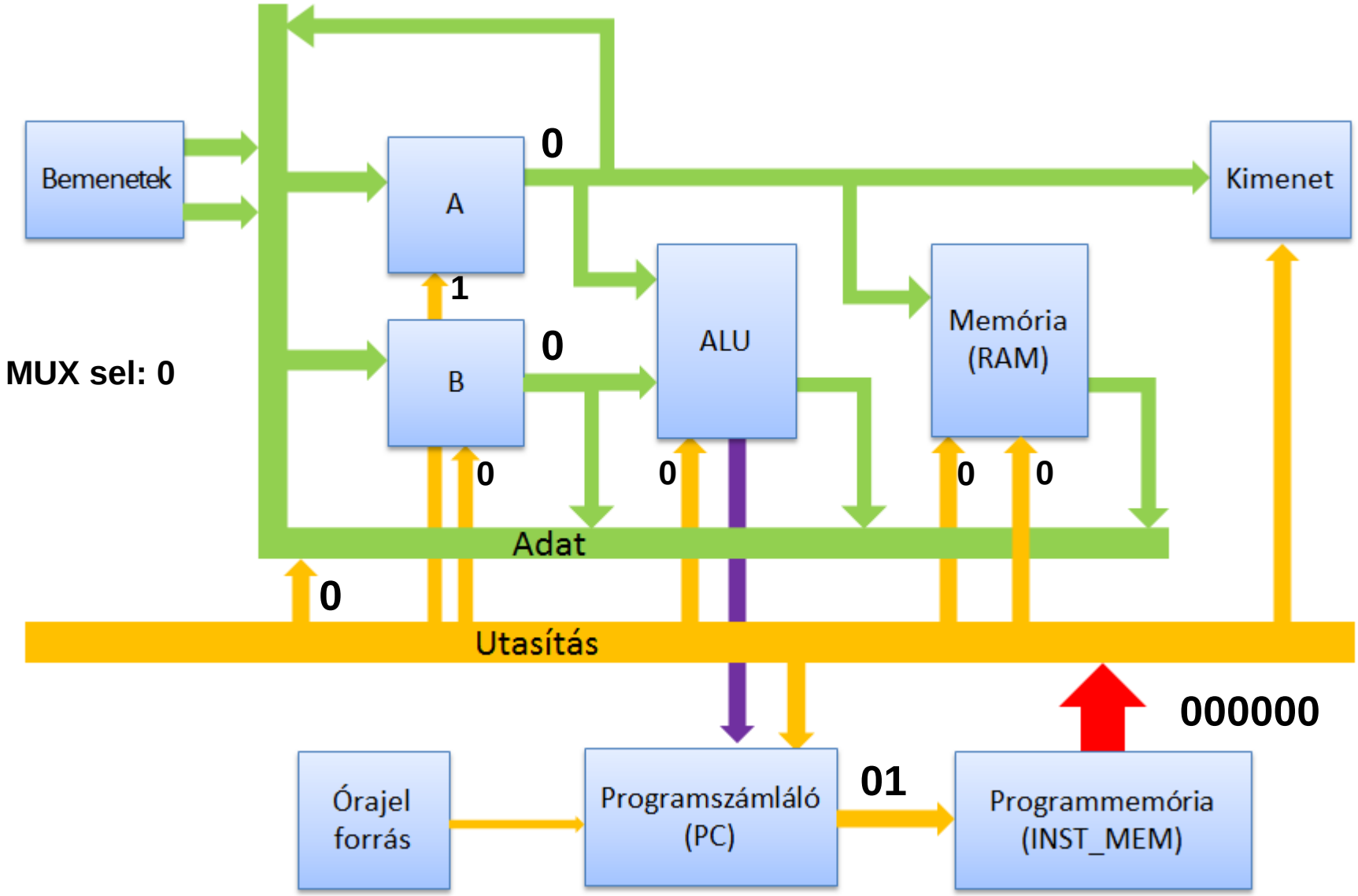
OR	A|B	98000
XOR	A^B	D8000
BR a	a->PC	700
BRZ a	a->PC	100
BRNZ a	a->PC	200
BRC a	a->PC	300
BRNC a	a->PC	400
BRO a	a->PC	500
BRNO a	a->PC	600
IN d	in(d)->A	10000
OUT k	A->out(d)	1000

Program készítése

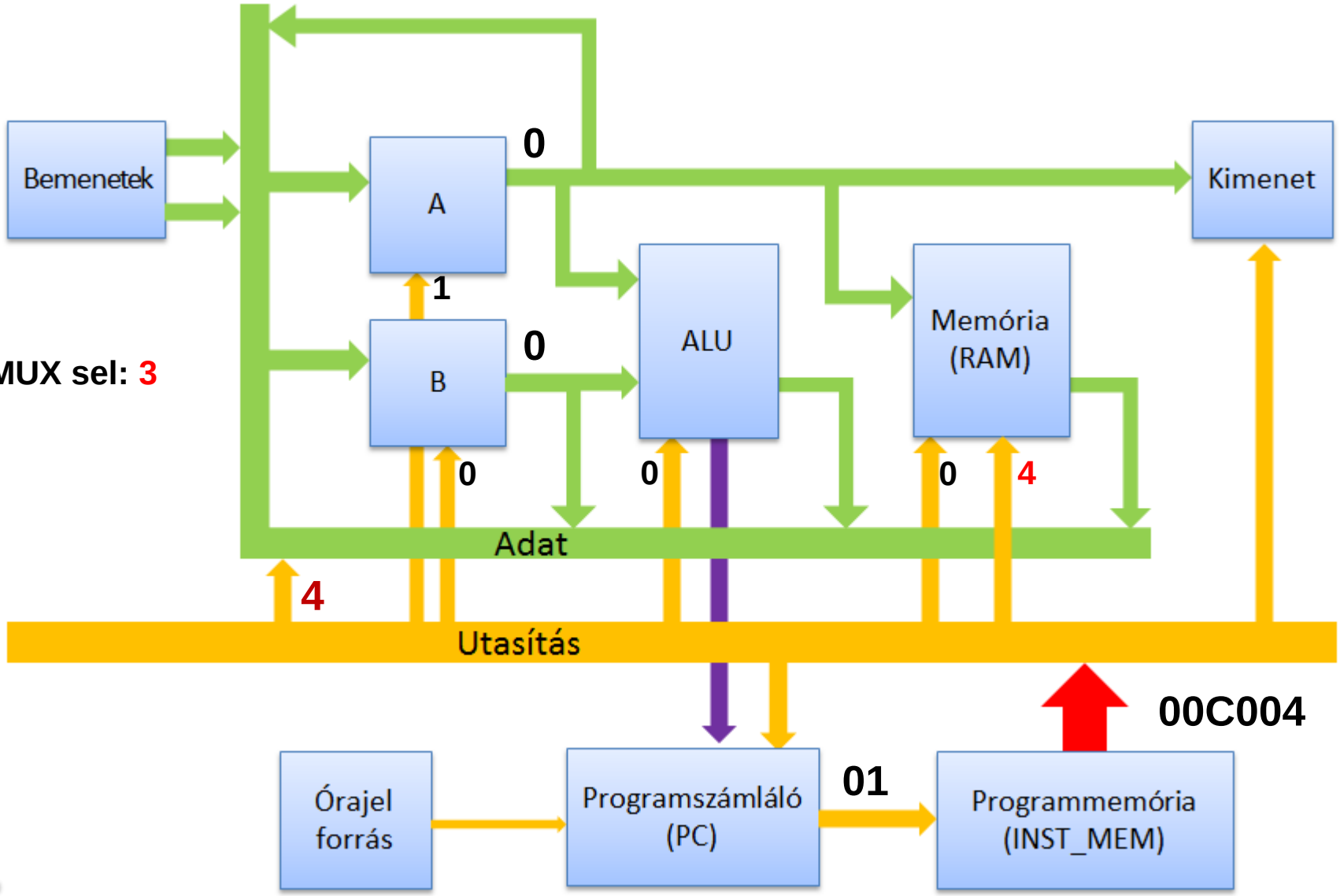
- A következő példaprogramot



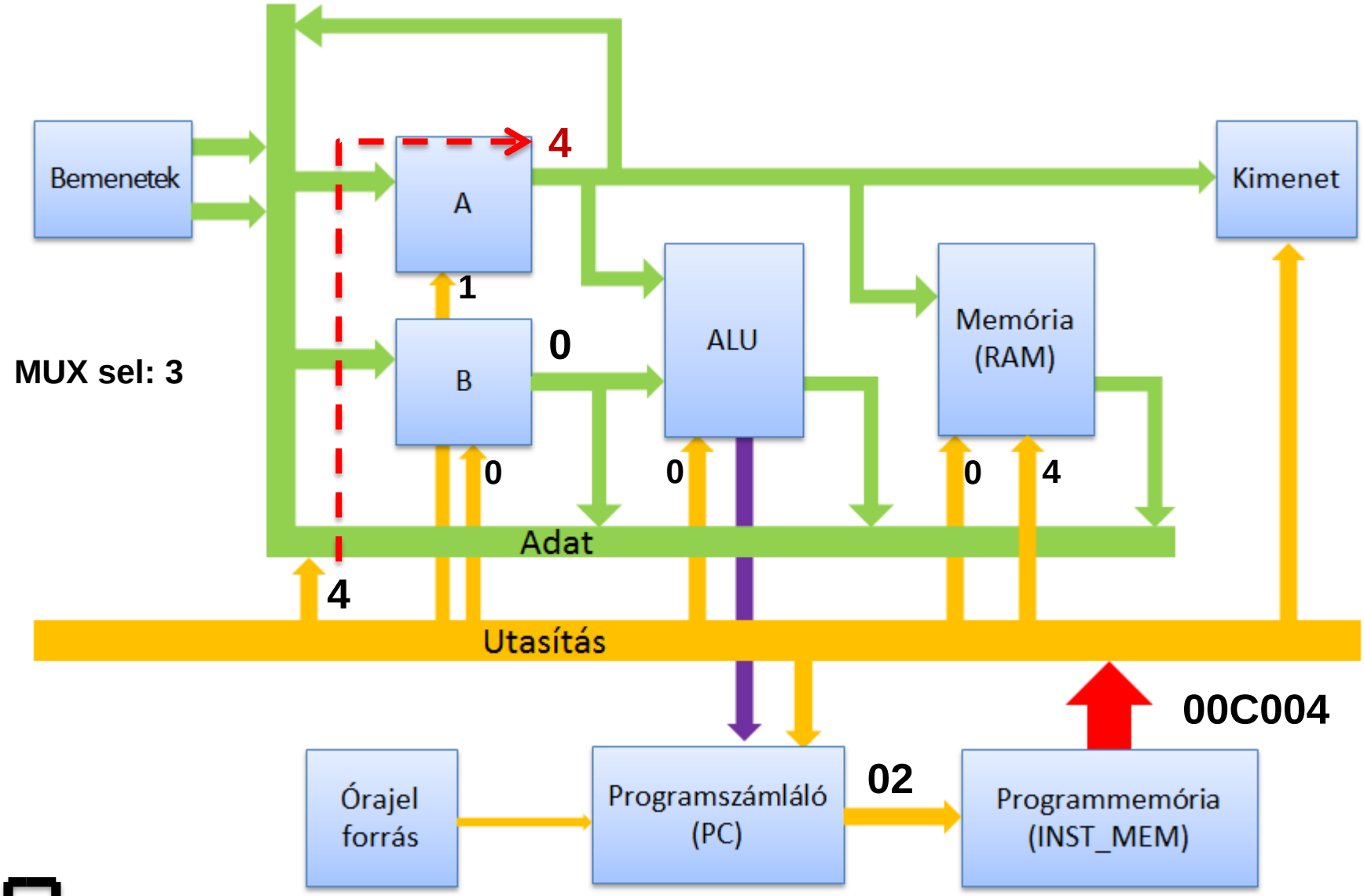
nop


Bemenetek
A
0
Kimenet
1
B
0
ALU
Memória
(RAM)
MUX sel: 0
0
0
0
0
Adat
0
Utasítás
000000
Órajel
forrás
Programszámláló
(PC)
01
Programmemória
(INST_MEM)

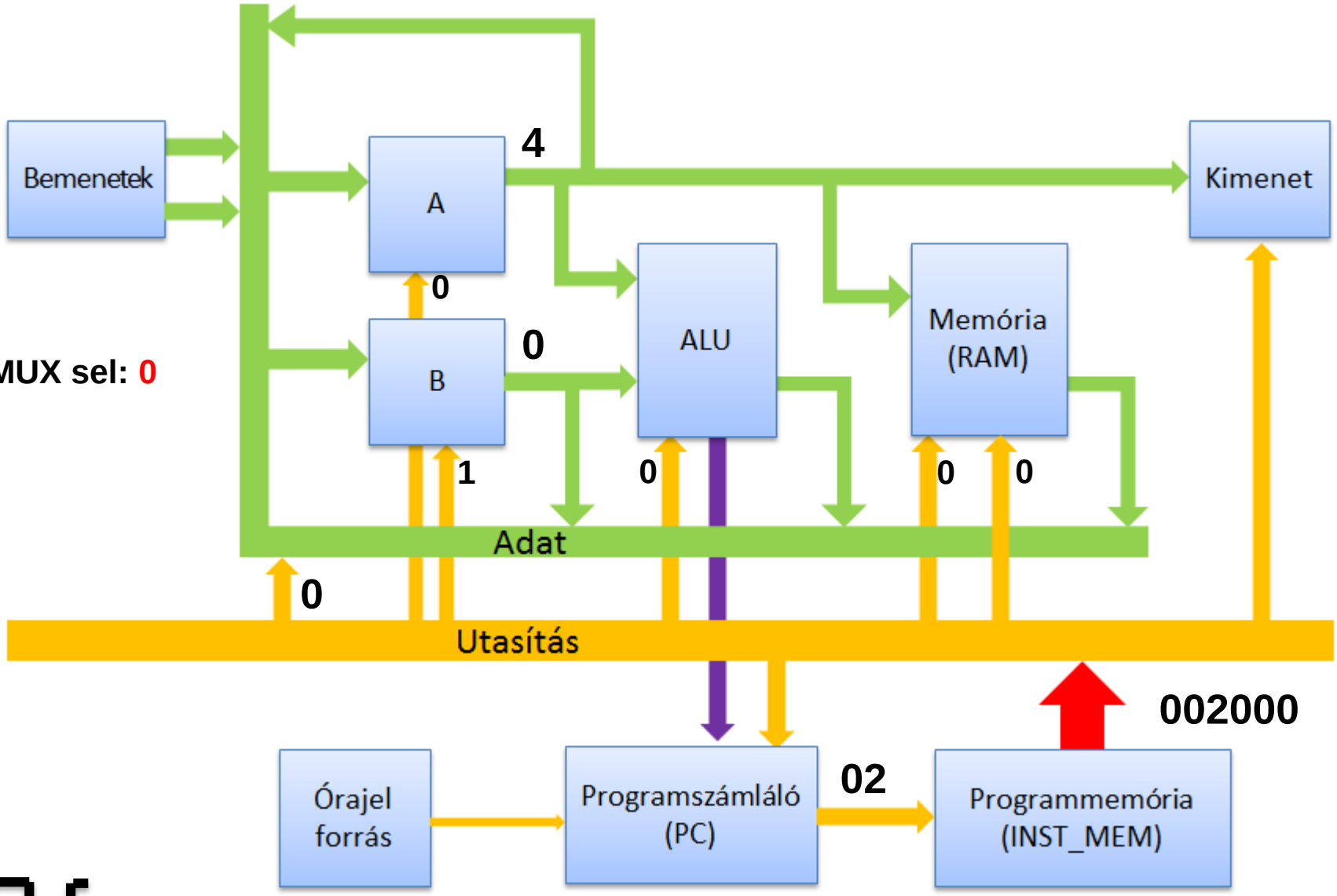
ldi 4


Bemenetek
A
0
Kimenet
1
B
0
ALU
Memória
(RAM)
MUX sel: 3
0
0
0
4
Adat
4
Utasítás
00C004
Órajel
forrás
Programszámláló
(PC)
01
Programmemória
(INST_MEM)

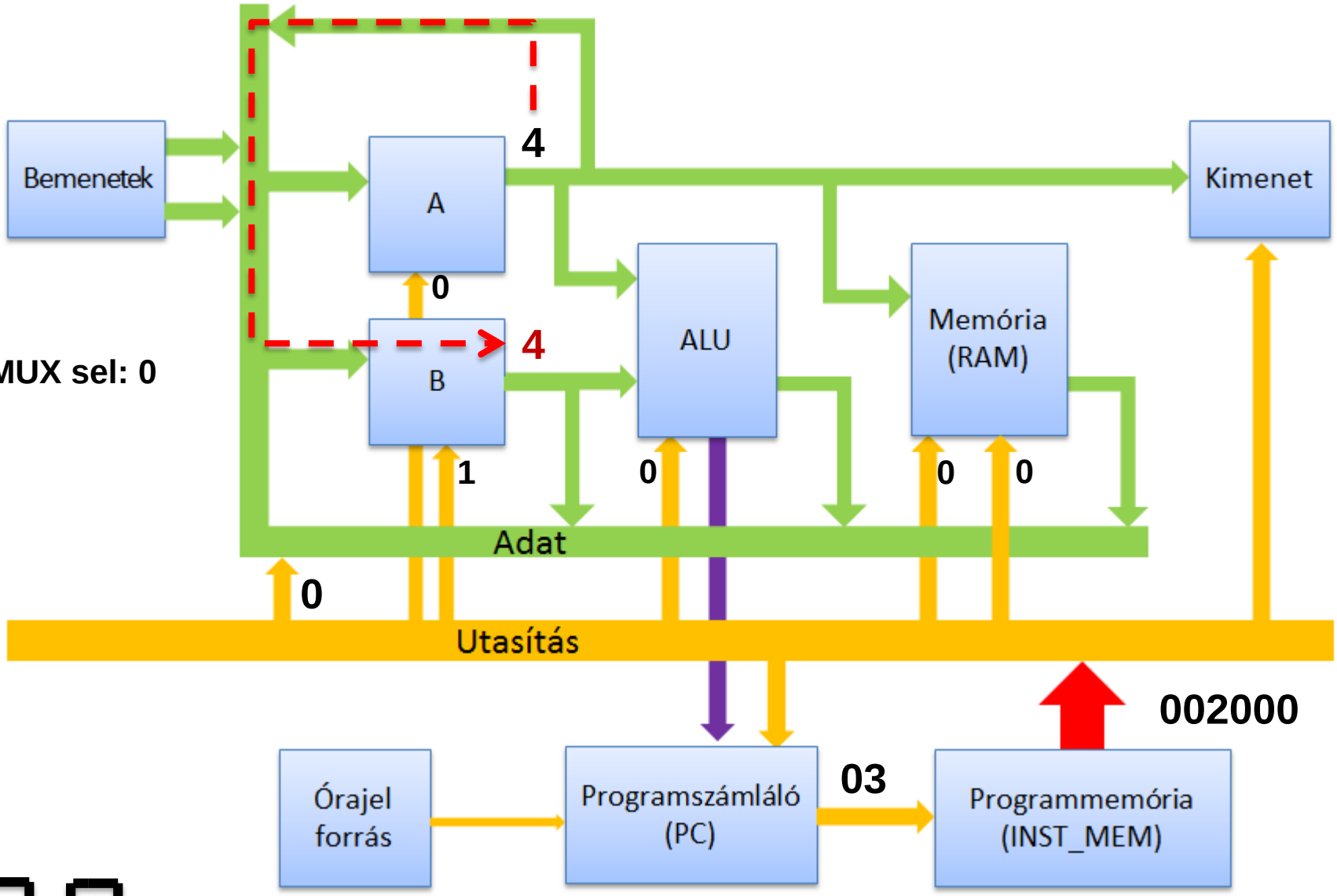
ldi 4


Bemenetek
A
4
Kimenet
1
MUX sel: 3
B
0
ALU
Memória
(RAM)
0
0
0
4
Adat
4
Utasítás
00C004
Órajel
forrás
Programszámláló
(PC)
02
Programmemória
(INST_MEM)

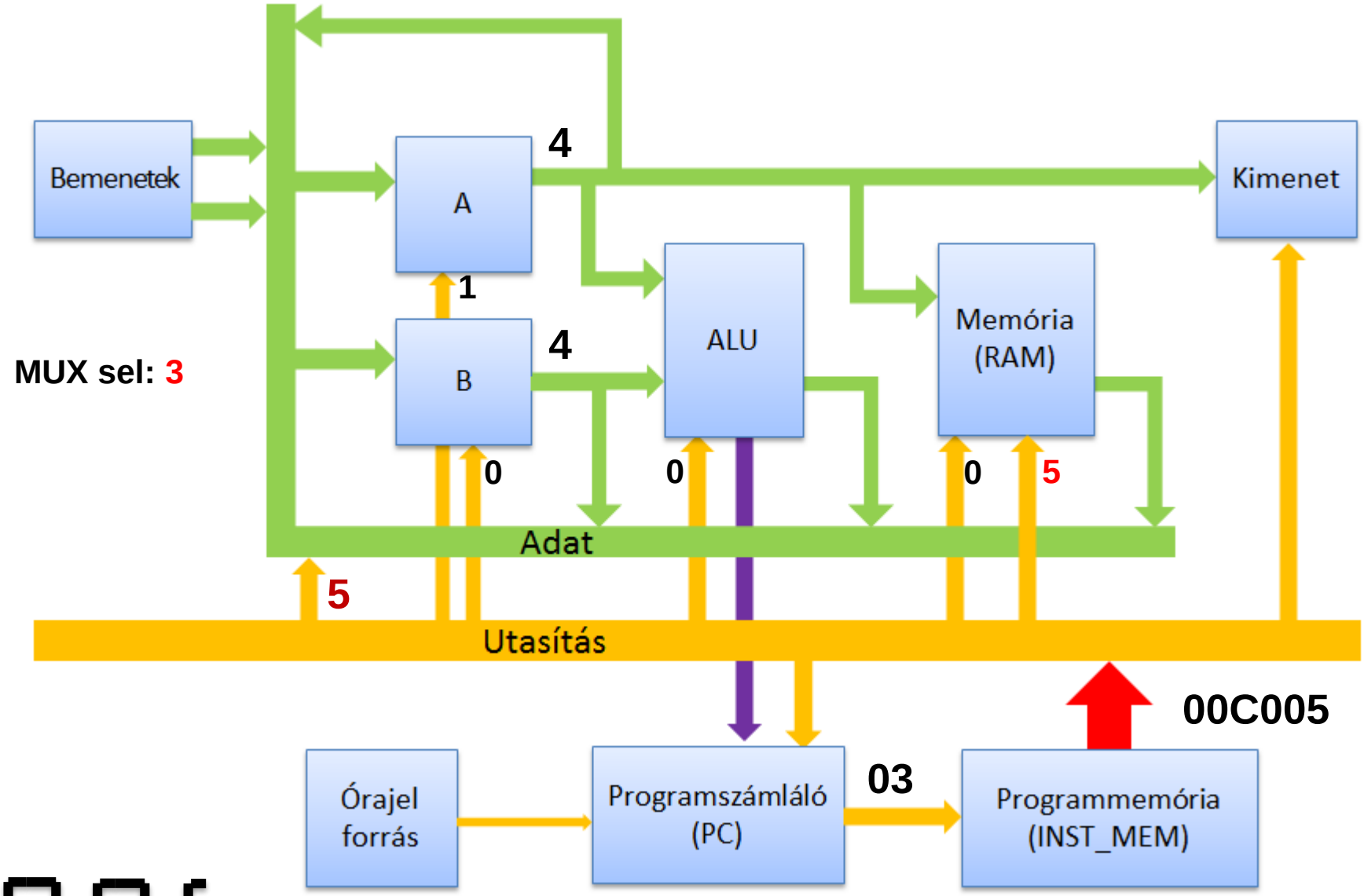
mvb


Bemenetek
A
4
Kimenet
0
B
0
ALU
Memória
(RAM)
MUX sel: 0
1
0
0
0
Adat
0
Utasítás
002000
Órajel
forrás
Programszámláló
(PC)
02
Programmemória
(INST_MEM)

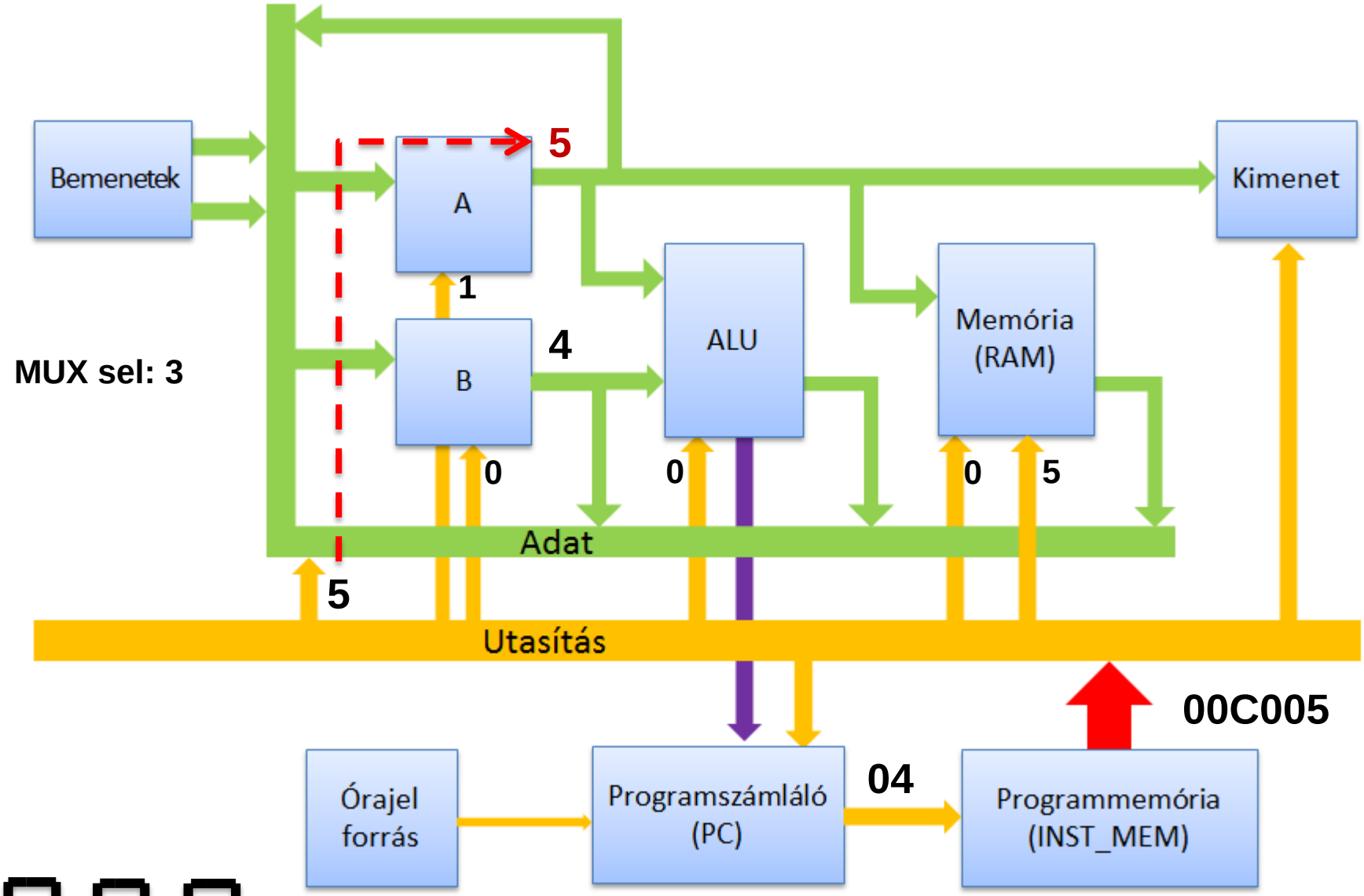
mvb


Bemenetek
A
B
4
4
0
1
ALU
Memória
(RAM)
Kimenet
0
0
0
MUX sel: 0
Adat
0
Utasítás
002000
Órajel
forrás
Programszámláló
(PC)
03
Programmemória
(INST_MEM)

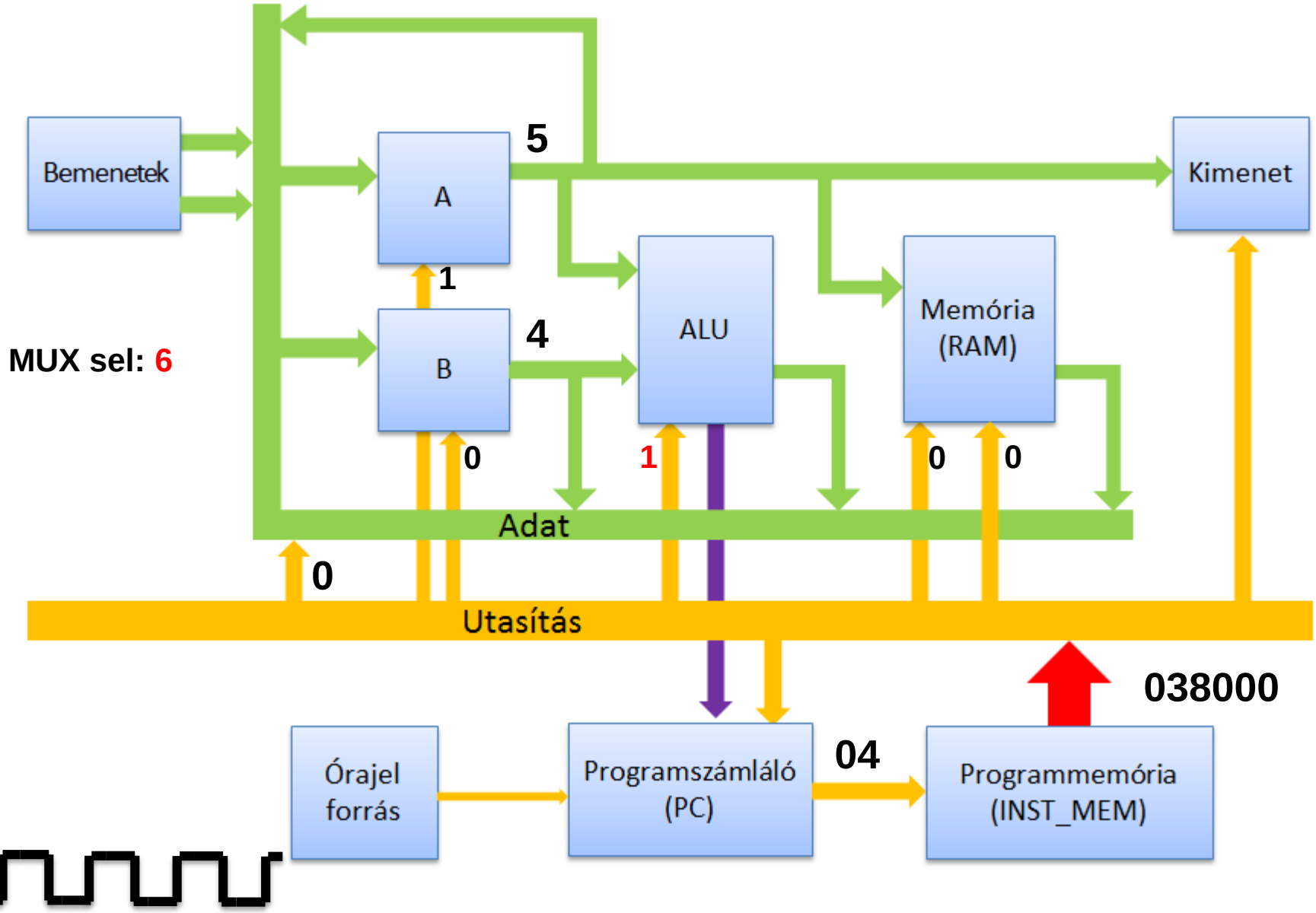
ldi 5


Bemenetek
A
4
Kimenet
1
B
4
ALU
Memória
(RAM)
MUX sel: 3
0
0
0
5
Adat
5
Utasítás
00C005
Órajel
forrás
Programszámláló
(PC)
03
Programmemória
(INST_MEM)

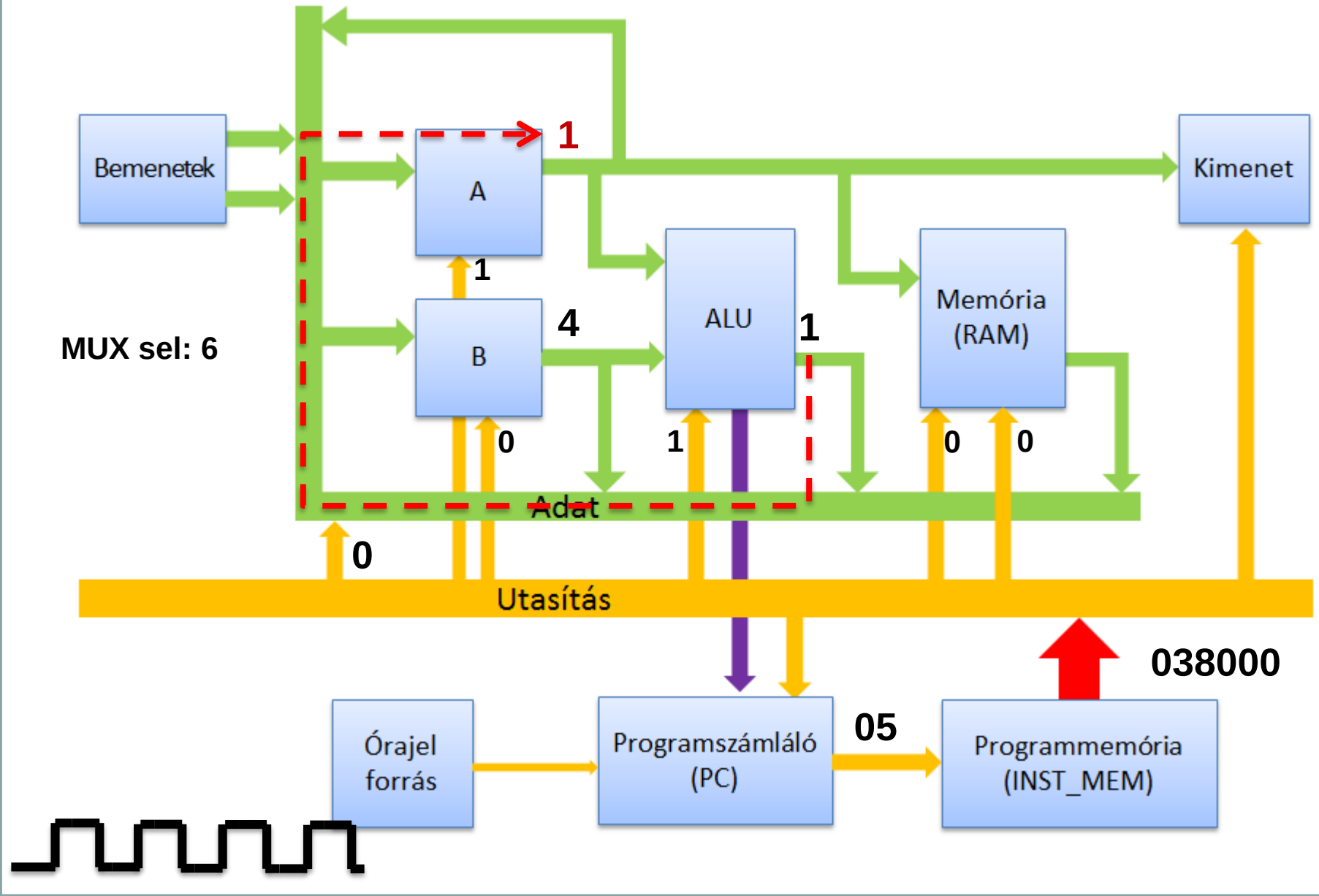
ldi 5


Bemenetek
A
5
Kimenet
1
B
4
ALU
Memória
(RAM)
MUX sel: 3
0
0
0
5
Adat
5
Utasítás
00C005
Órajel
forrás
Programszámláló
(PC)
04
Programmemória
(INST_MEM)

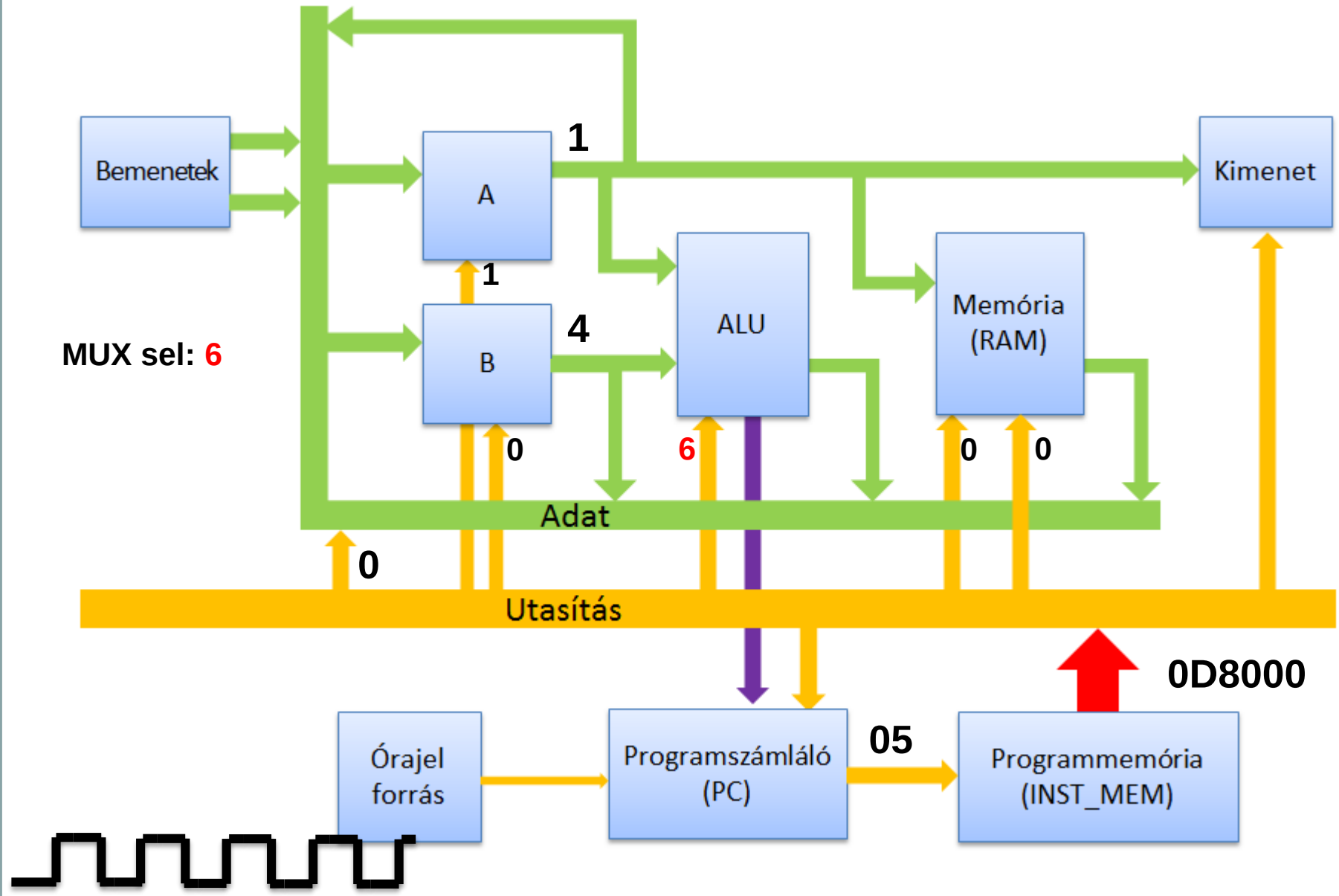
sub


Bemenetek
A
5
Kimenet
1
B
4
ALU
Memória
(RAM)
MUX sel: 6
0
1
0
0
Adat
0
Utasítás
038000
Órajel
forrás
Programszámláló
(PC)
04
Programmemória
(INST_MEM)

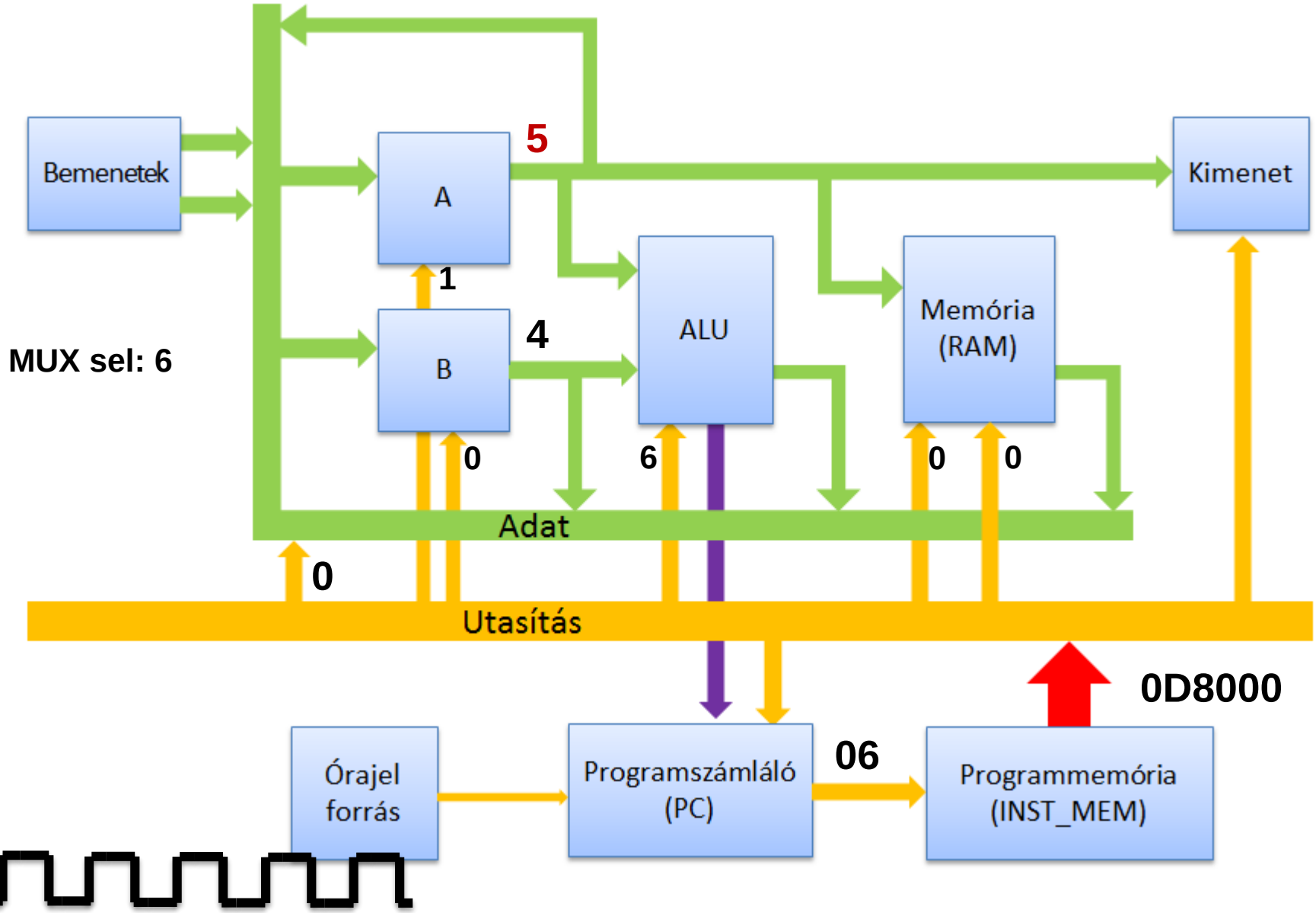
sub

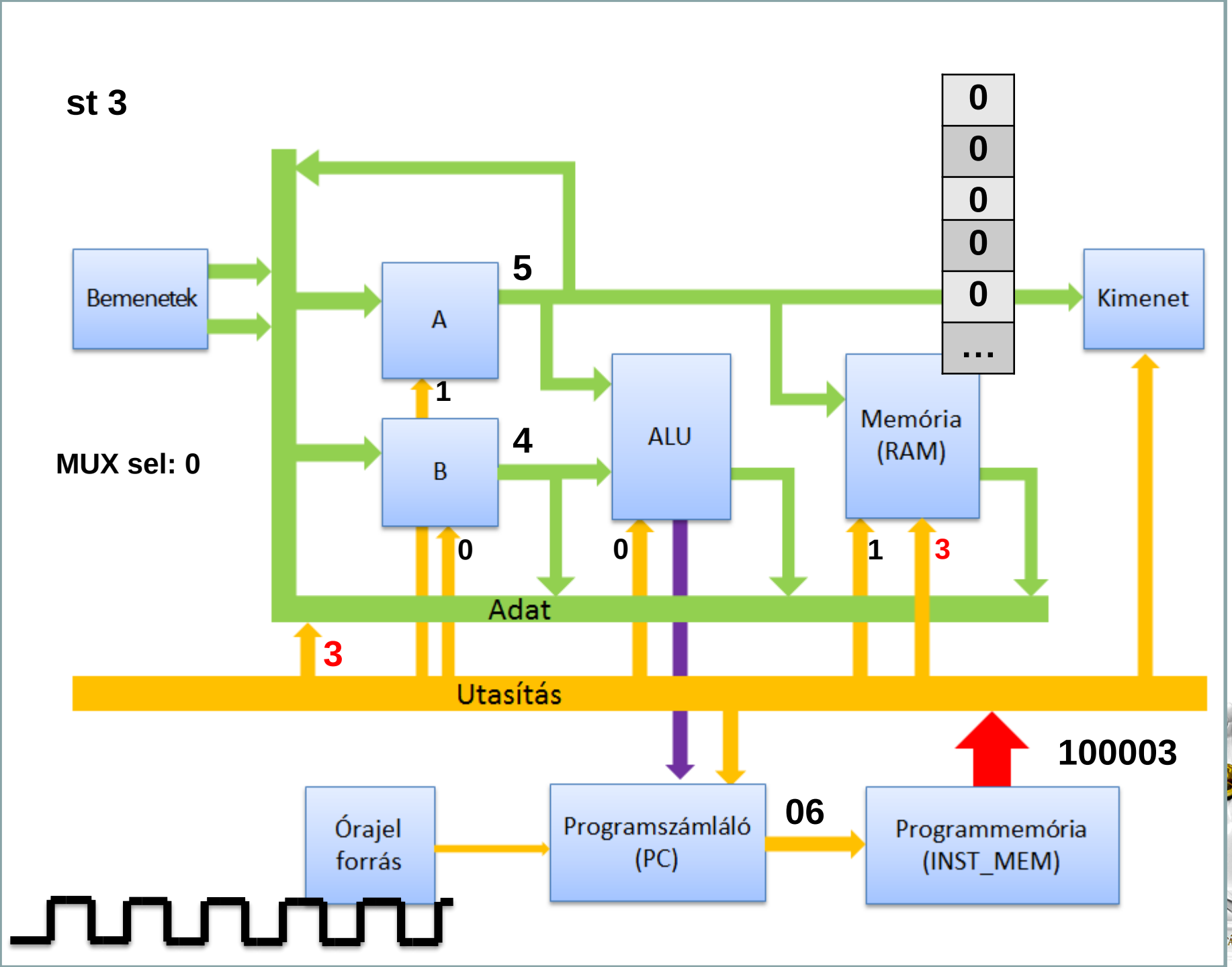

Bemenetek
A
1
Kimenet
1
MUX sel: 6
B
4
ALU
1
Memória
(RAM)
0
1
0
0
Adat
0
Utasítás
038000
Órajel
forrás
Programszámláló
(PC)
05
Programmemória
(INST_MEM)

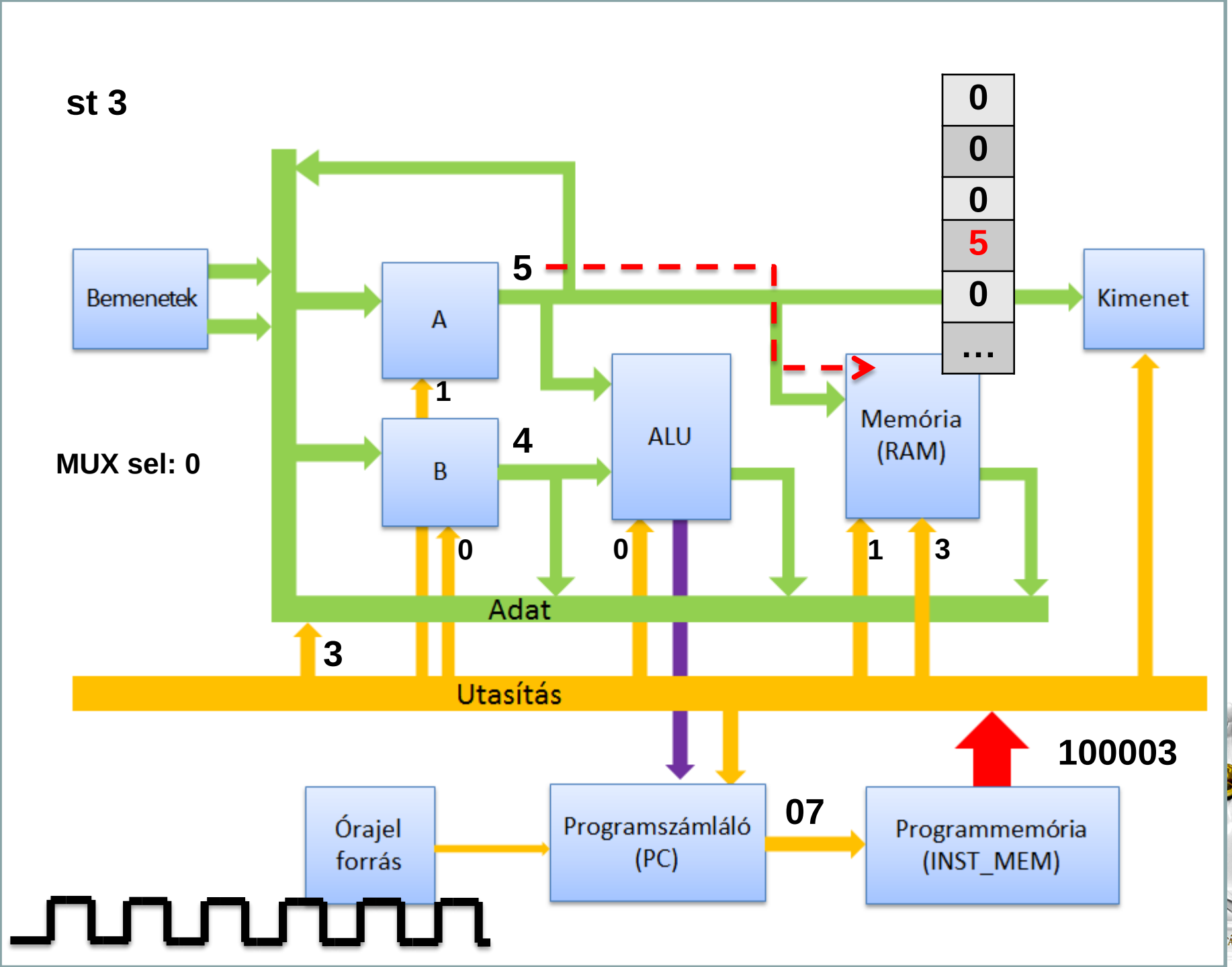
xor


Bemenetek
A
1
Kimenet
1
B
4
ALU
Memória
(RAM)
MUX sel: 6
0
6
0
0
Adat
0
Utasítás
0D8000
Órajel
forrás
Programszámláló
(PC)
05
Programmemória
(INST_MEM)

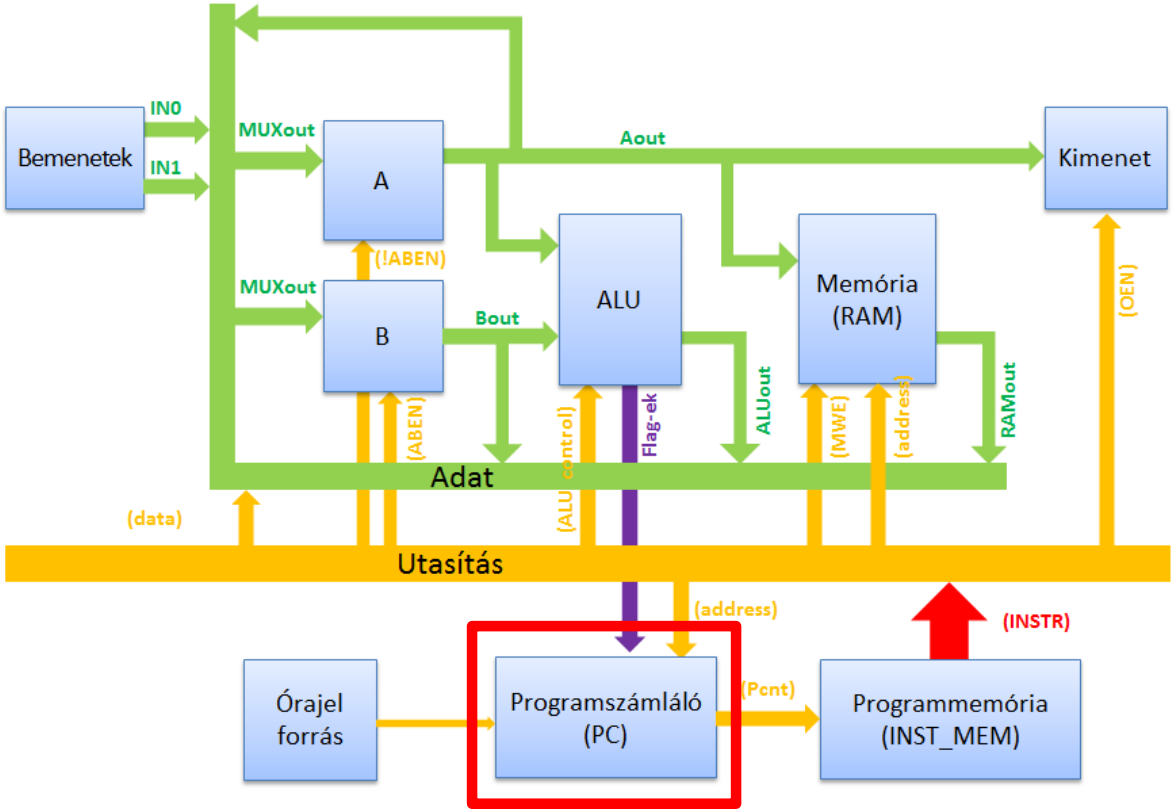
xor


Bemenetek
A
5
Kimenet
1
B
4
ALU
Memória
(RAM)
MUX sel: 6
0
6
0
0
Adat
0
Utasítás
0D8000
Órajel
forrás
Programszámláló
(PC)
06
Programmemória
(INST_MEM)


st 3
0
0
0
0
0
...
Bemenetek
A
5
Kimenet
1
B
4
ALU
Memória
(RAM)
MUX sel: 0
0
0
1
3
Adat
3
Utasítás
100003
Órajel
forrás
Programszámláló
(PC)
06
Programmemória
(INST_MEM)

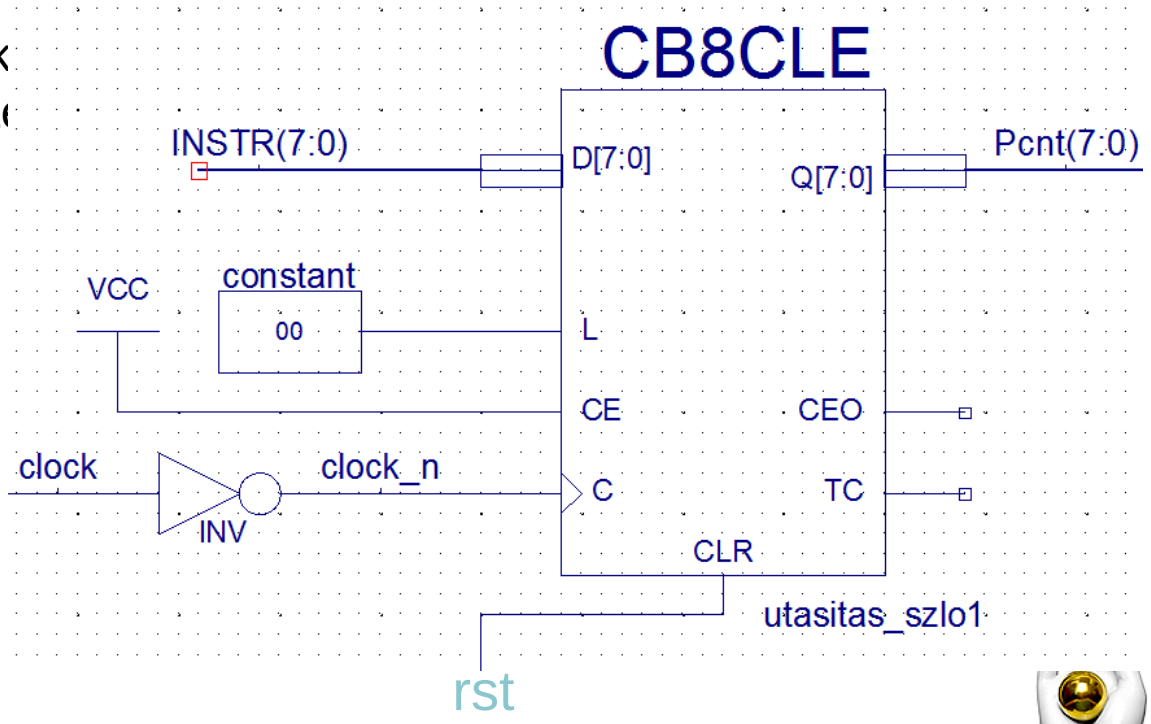

st 3
0
0
0
5
0
...
Bemenetek
A
5
Kimenet
1
B
4
ALU
Memória
(RAM)
MUX sel: 0
0
0
1
3
Adat
3
Utasítás
100003
Órajel
forrás
Programszámláló
(PC)
07
Programmemória
(INST_MEM)

A programszámláló (PC)



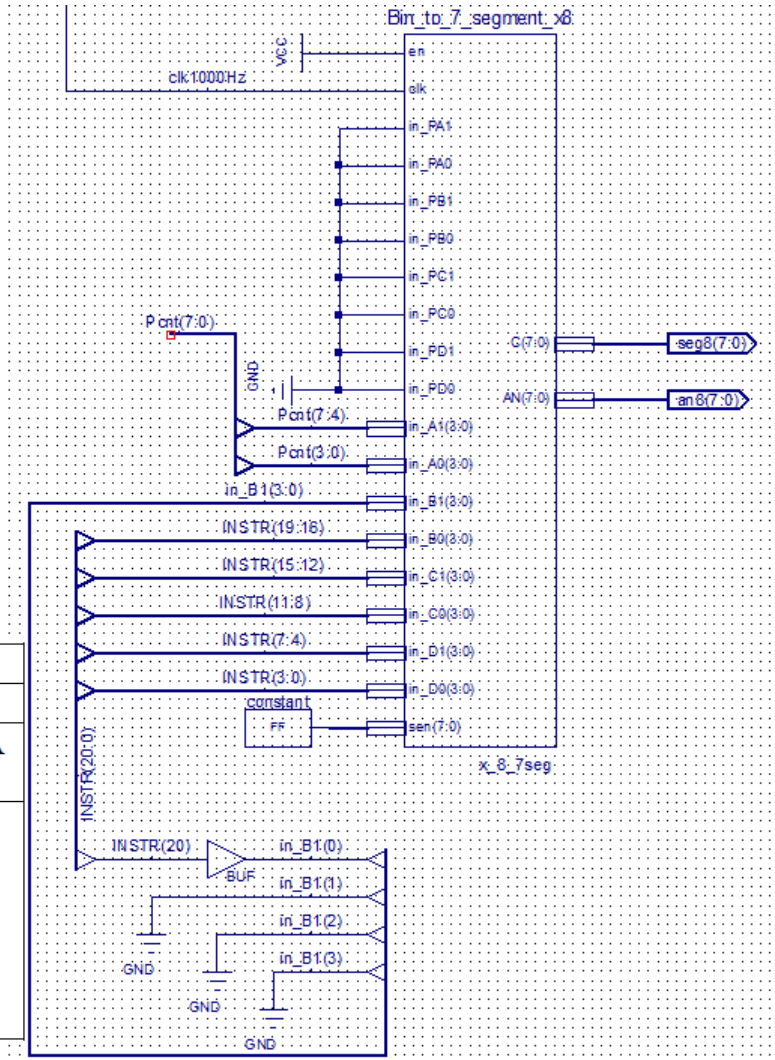
A programszámláló megvalósítása

- 8 bites, tehát max. 256 sora lehet a programnak
- Az **órajel lefutó éle** vez
- 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!)



Kijelzés

A kiegészítő kijelzőkre rákötjük a programmemória és a programszámláló aktuális értékeit



Bin_to_7_segment_x8 bekötése		
Bemenet	Jel	Leírás
in_A1	Pcnt(7:4)	Programszámláló (PC) megjelenítése az A kijelzőn
in_A0	Pcnt(3:0)	
in_B1	INSTR(20)	Utasítás megjelenítése a B, C és D kijelző modulokon
in_B0	INSTR(19:16)	
in_C1	INSTR(15:12)	
in_C0	INSTR(11:8)	
in_D1	INSTR(7:4)	
in_D0	INSTR(3:0)	

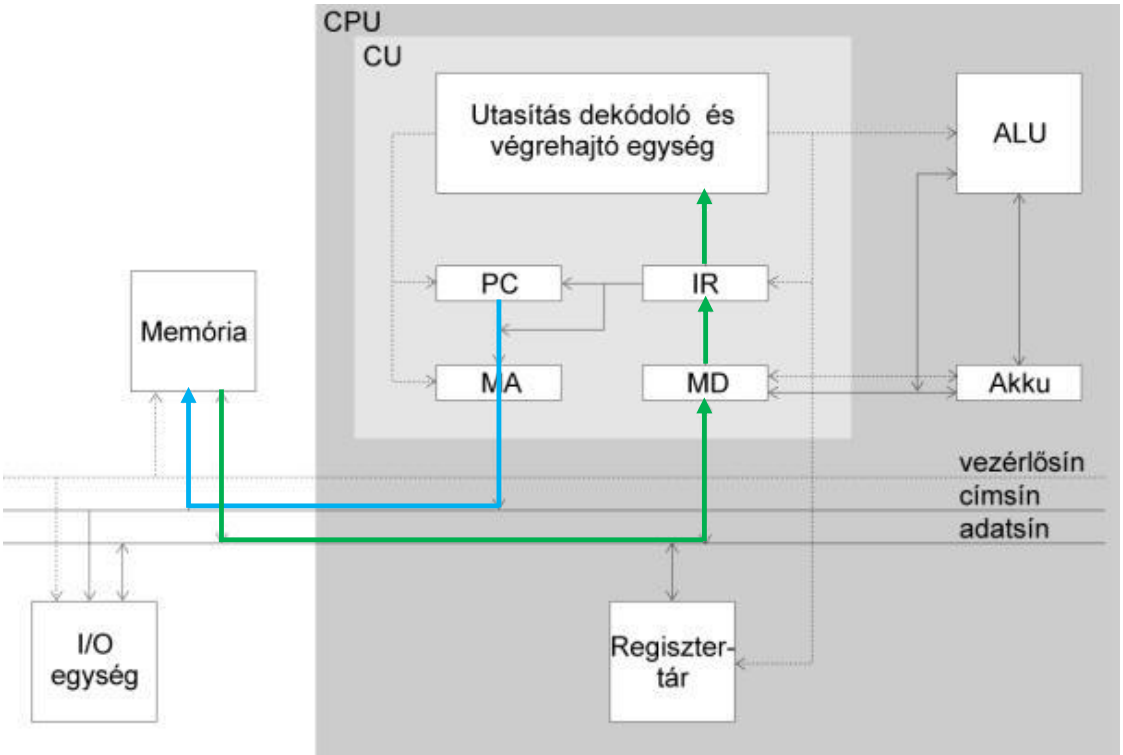
Mikroprocesszoros rendszerek

- CPU
  - Az utasítások alapján előállítja a processzoron belüli és a processzorhoz kapcsolt külső egységek működéséhez szükséges vezérlőjeleket
    - Belső vezérlő jelek
      - ALU működtetése
      - Regiszterek és a belső sínrendszer közötti adatátvitel
    - Külső vezérlőjelek
      - A memória – CPU közötti adatátvitel
      - A perifériák – CPU közötti adatátvitel
  - A művelet-végrehajtás elemi lépéseit mikroprogram írja le, amely a programtárban (memória) helyezkedik el
    - A mikroprogram mikroutasításokból áll
      - Műveleti kód
        - Az adott fázisban elvégzendő feladathoz rendelt vezérlővonal állapotokat adja meg
      - Címzésmód
        - A mikroprogram végrehajtása hogyan folytatódik a következő utasítással a tárban , feltétel nélküli vezérlésátadással (INC) vagy egy külső feltételtől függő vezérlésátadással (JMP)
      - Következő mikroutasítás címe
        - Automatikus inkrementálásnál lényegtelen
      - Operandus(ok) cime(i)
        - A memória mely elemeivel akarunk műveletet végezni
        - Pl: A = A+B (A és B a memóriában tárolt adatok címei)

Működés

- Pl. Adat beírása az akkumulátor regiszterbe

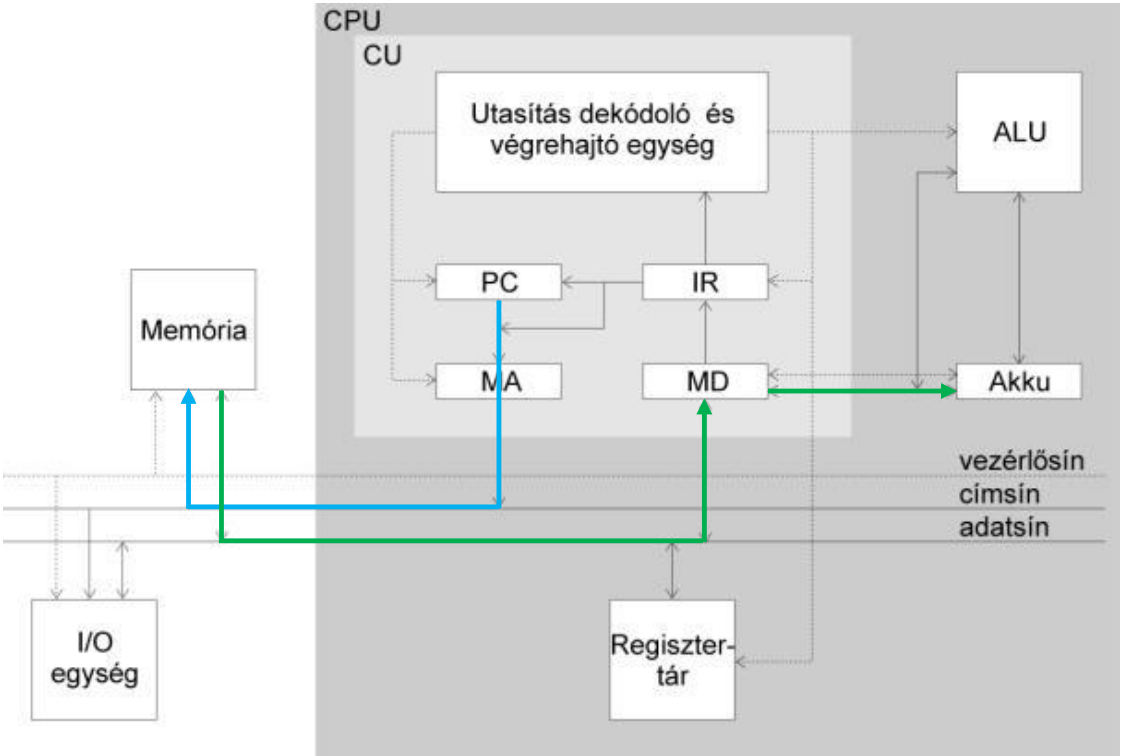
1. **lépés:** A PC-ből az MA-n keresztül a memória bemeneteire jut az utasítás címe. A memória adatvezetékein megjelenik a rekesz tartalma (vagyis a műveleti kód), az MD-n keresztül az IR-be kerül
2. **lépés:** Az utasítás dekódoló és végrehajtó egység beolvassa a műveleti kódot, és értelmezi azt
3. **lépés:** A PC értéke eggyel nő (így az operandus címére mutat)



Működés

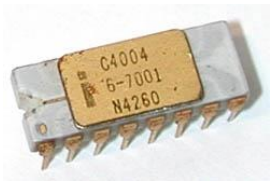
- Pl. Adat beírása az akkumulátor regiszterbe

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.



Mikroprocesszoros rendszerek

- Mikroprocesszorok (μP), mikroszámítógépek (μC)



Mikroprocesszoros rendszerek

- Mikroprocesszorok (μP), mikroszámítógépek (μC)

[ATmega168]
[TMS320C28x]
[TMS320C6416T]