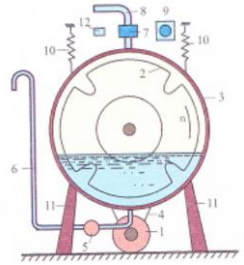
Digitális technika 2.

BMEVIIIAA06

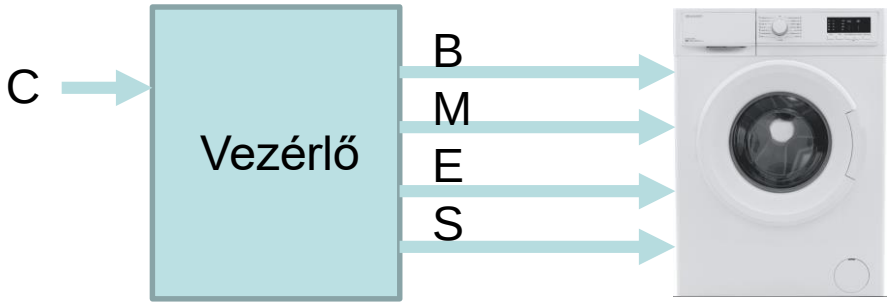
Előadás

Egyszerű processzor tervezése

Feladat: Mosógép vezérlés



Tervezzünk egy egyszerű vezérlő egységet az alábbi mosógéphez



Kimenetek:
- Beengedő szelep
- Motor
- Elektromos fűtés
- Szivattyúzás

Bemenet:
- Centrifuga választás

Mosási program (C=0) centrifuga nélkül

Lépés	Művelet	B	M	E	S
1.	Víz beengedés	1	0	0	0
2.	Melegítés	0	0	1	0
3.	Forgatás	0	1	0	0
4.	Ürítés	0	0	0	1

Mosási program (C=1) centrifugálással

Lépés	Művelet	B	M	E	S
1.	Víz beengedés	1	0	0	0
2.	Melegítés	0	0	1	0
3.	Forgatás	0	1	0	0
4.	Ürítés	0	0	0	1
5.	Forgatás	0	1	0	0
6.	Ürítés	0	0	0	1

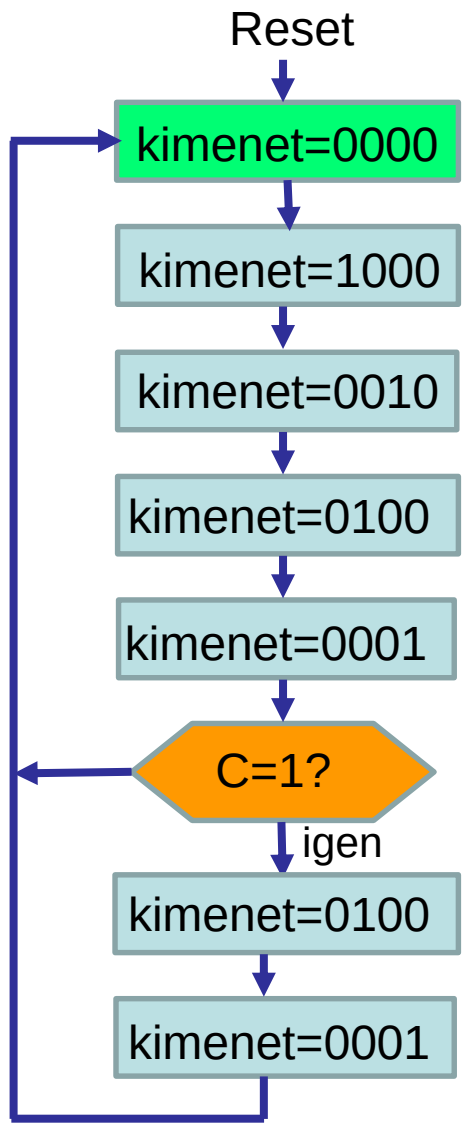
Folyamatábra

Mosási program (C=0) centrifuga nélkül

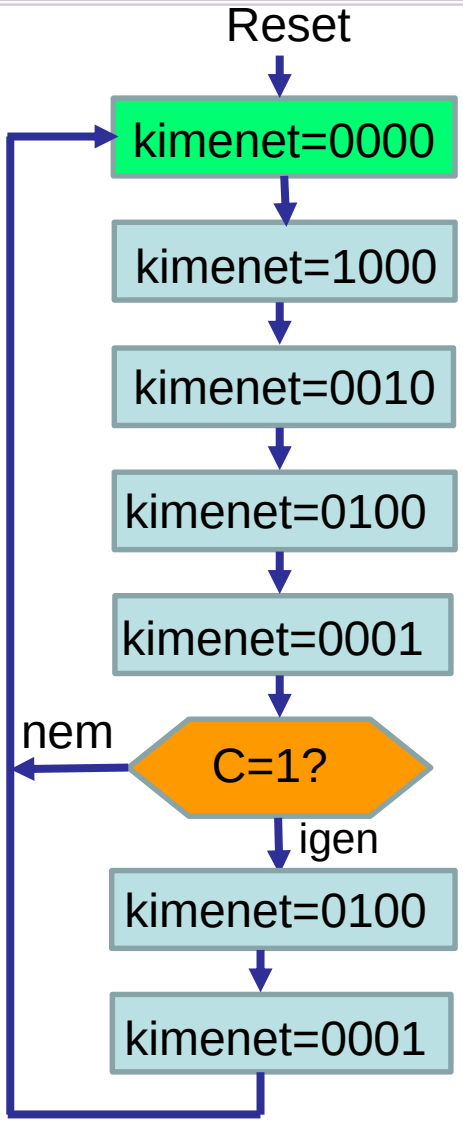
Lépés	Művelet	B	M	E	S
1.	Víz beengedés	1	0	0	0
2.	Melegítés	0	0	1	0
3.	Forgatás	0	1	0	0
4.	Ürítés	0	0	0	1

Mosási program (C=1) centrifugálással

Lépés	Művelet	B	M	E	S
1.	Víz beengedés	1	0	0	0
2.	Melegítés	0	0	1	0
3.	Forgatás	0	1	0	0
4.	Ürítés	0	0	0	1
5.	Forgatás	0	1	0	0
6.	Ürítés	0	0	0	1

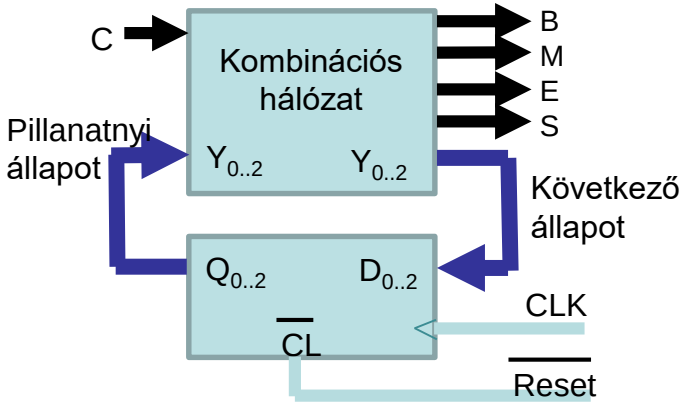


Folyamatábra



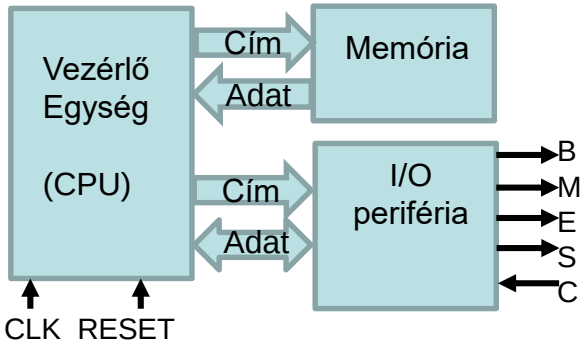
Megvalósítás #1

Szinkron Moore sorrendi hálózattal

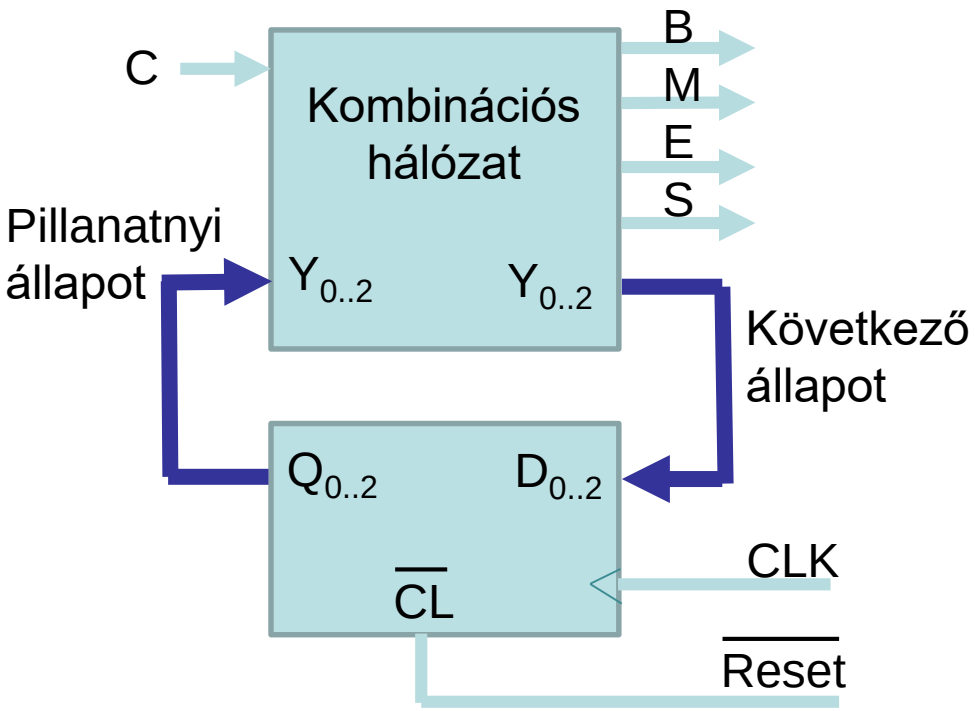


Megvalósítás #2

Mikroszámítógéppel



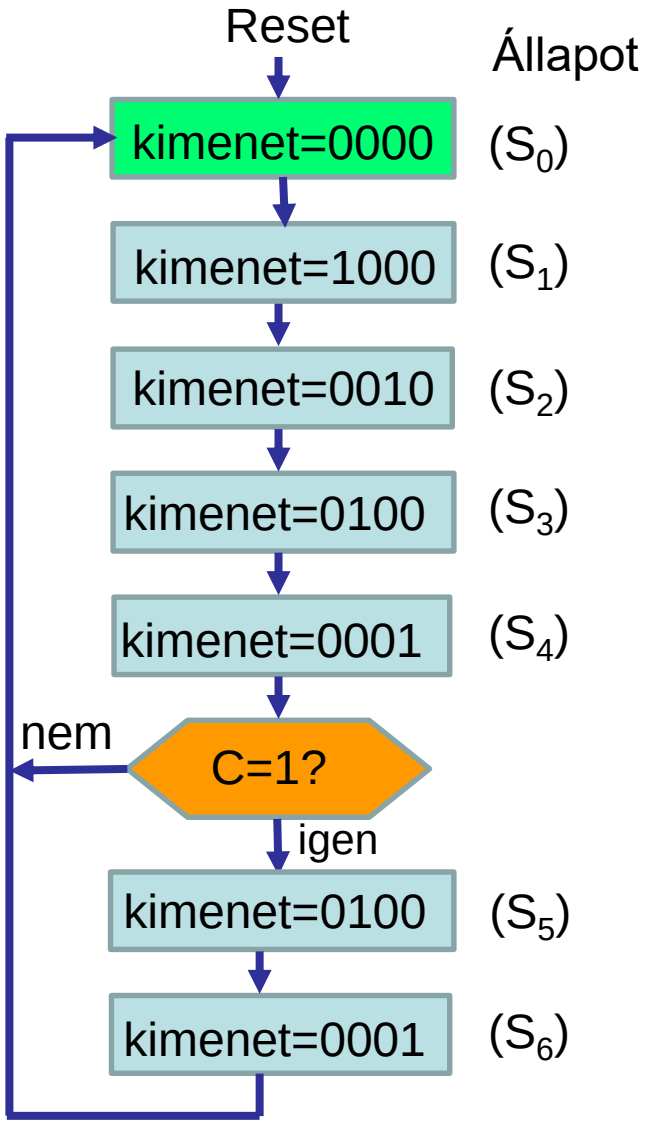
Sorrendi hálózat megvalósítás



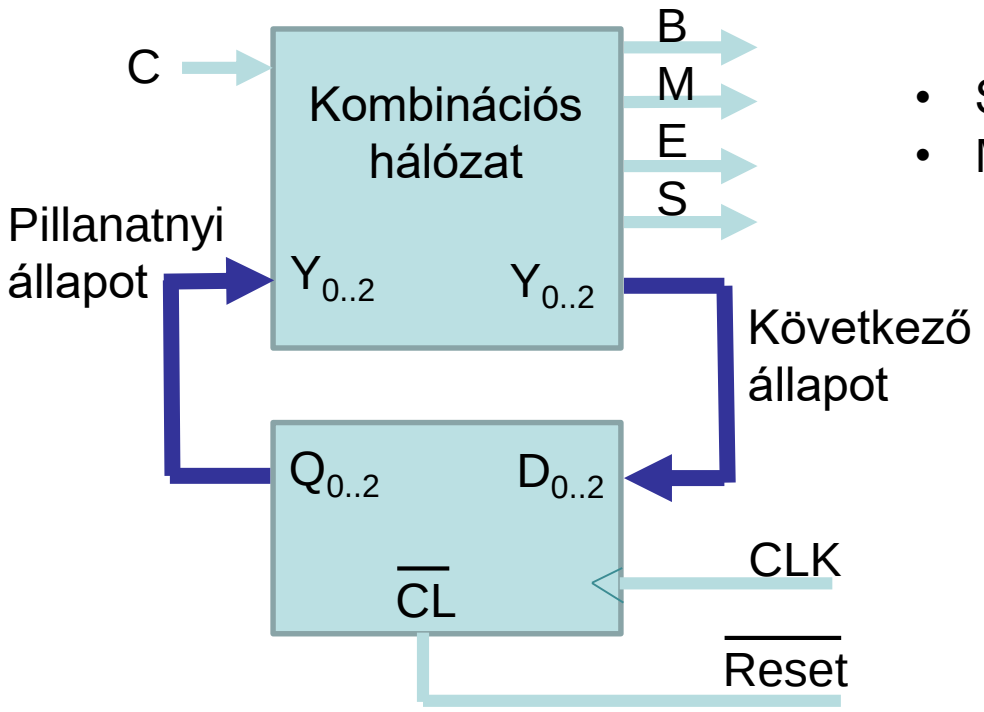
Megvalósítás #1

- Szinkron sorrendi hálózattal
- Moore modell
- Vegyük fel az állapottábláját

  →Állapotgép, vagy véges automata (Finite state machine, FSM)



Sorrendi hálózat (FSM) megvalósítás



Megvalósítás #1

- Szinkron sorrendi hálózattal
- Moore modell

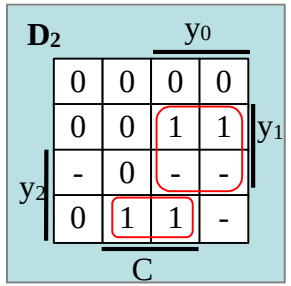
Kódolt állapottábla

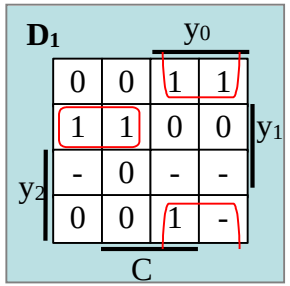
	$y_2y_1y_0$\C	0	1	BMES
(S_0)	000	001	001	0000
(S_1)	001	010	010	1000
(S_2)	010	011	011	0010
(S_3)	011	100	100	0100
(S_4)	100	000	101	0001
(S_5)	101	---	110	0100
(S_6)	110	---	000	0001
	111	---	---	----

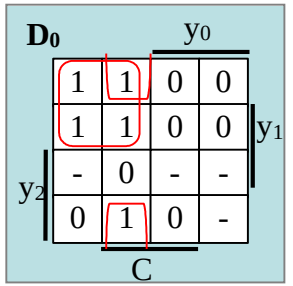
Sorrendi hálózat megvalósítás

Kódolt állapottábla

$y_2y_1y_0$\C	0	1	BMES
000	001	001	0000
001	010	010	1000
010	011	011	0010
011	100	100	0100
100	000	101	0001
101	---	110	0100
110	---	000	0001
111	---	---	----



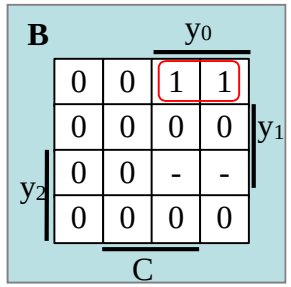


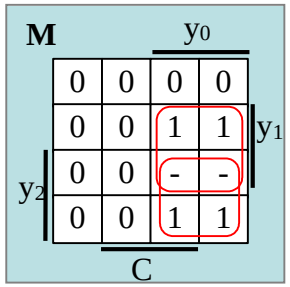


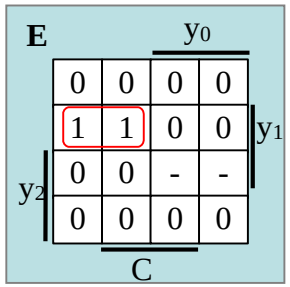
$$D_2 = y_1 \cdot y_0 + y_2 \cdot \overline{y_1} \cdot C$$

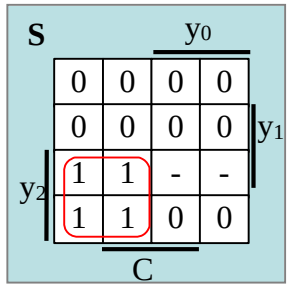
$$D_1 = y_0 \cdot \overline{y_1} + \overline{y_2} \cdot y_1 \cdot \overline{y_0}$$

$$D_0 = \overline{y_2} \cdot \overline{y_0} + \overline{y_1} \cdot \overline{y_0} \cdot C$$









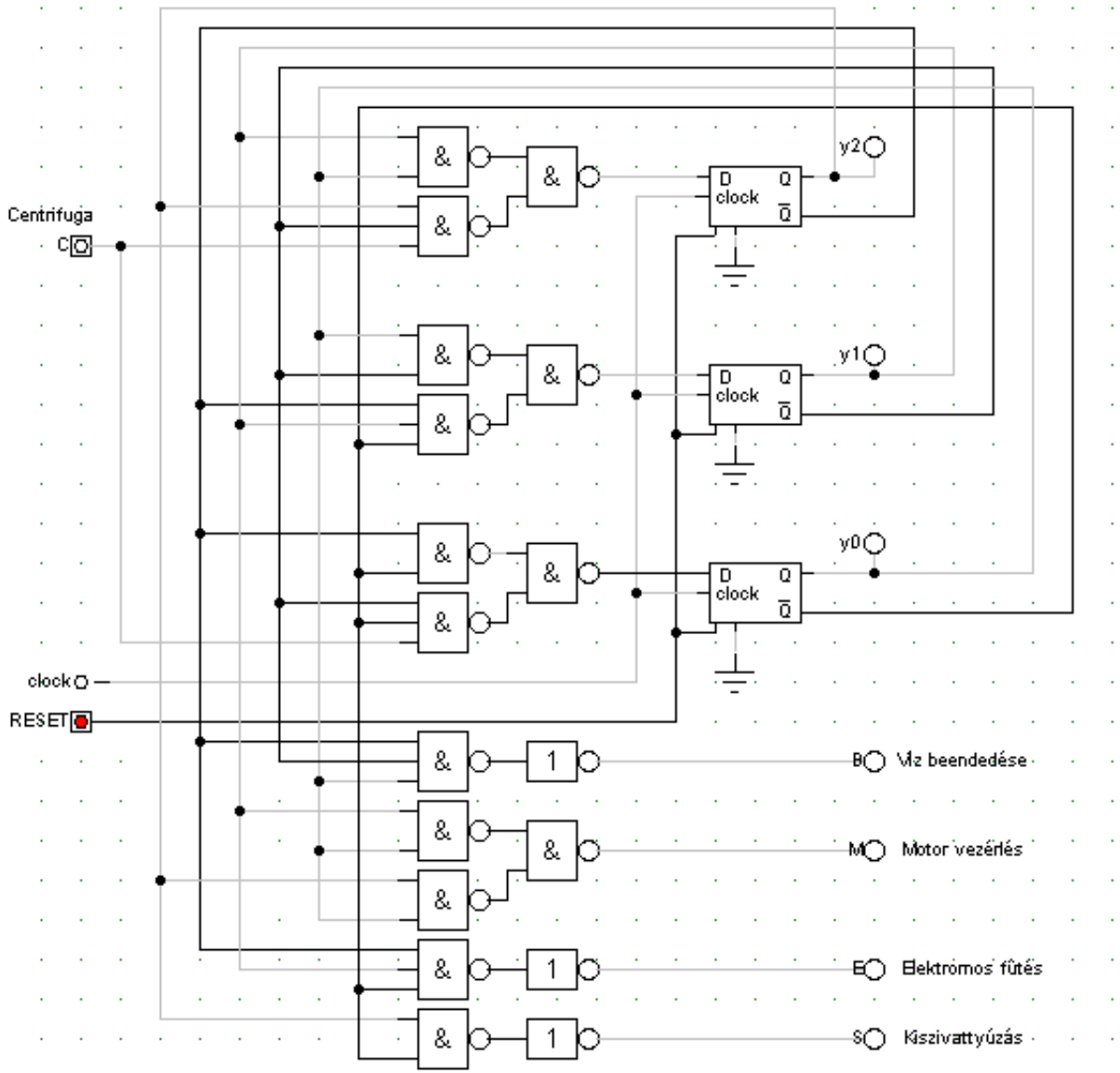
$$B = \overline{y_2} \cdot \overline{y_1} \cdot y_0$$

$$M = y_1 \cdot y_0 + y_2 \cdot y_0$$

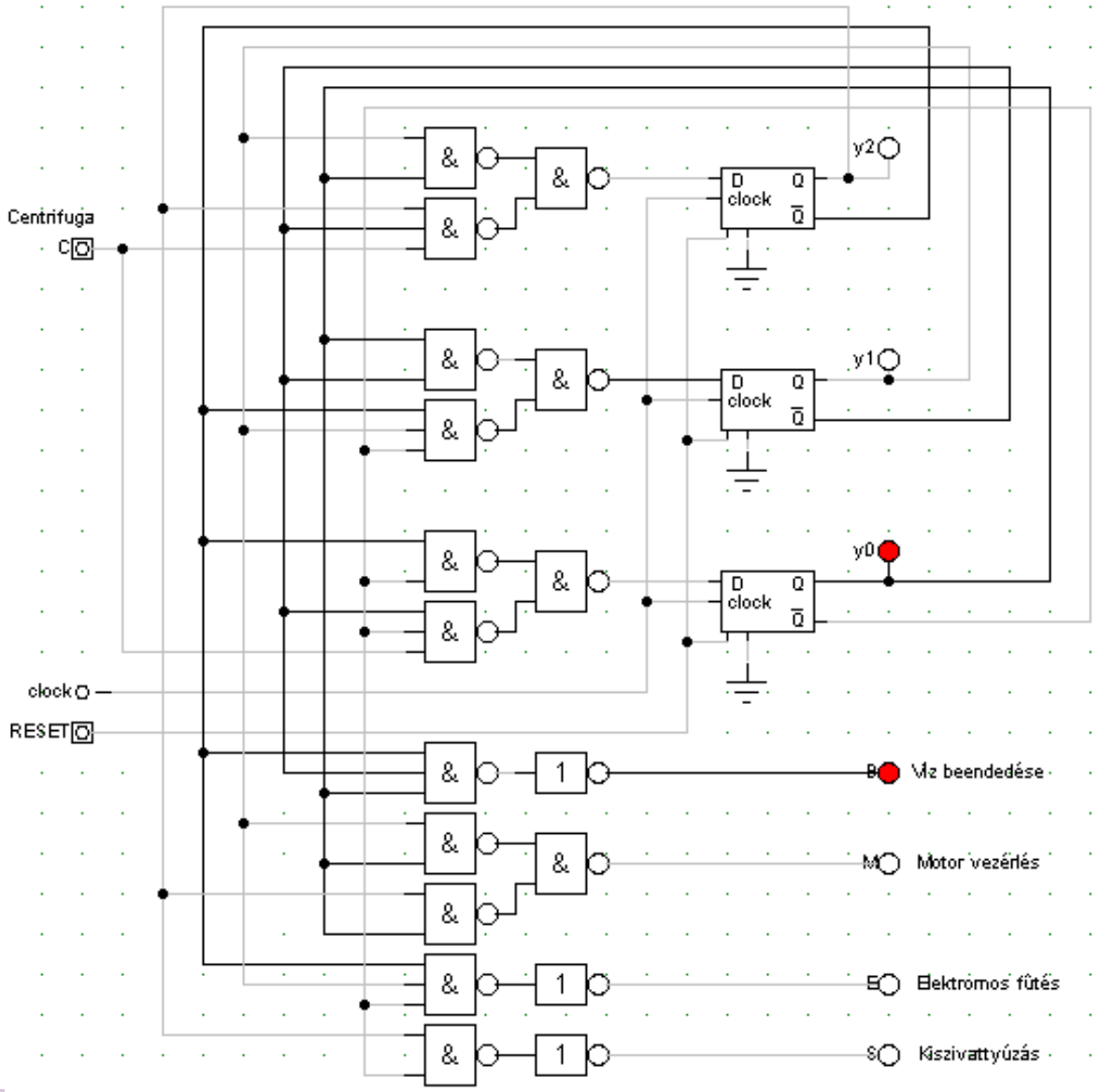
$$E = \overline{y_2} \cdot y_1 \cdot \overline{y_0}$$

$$S = y_2 \cdot \overline{y_0}$$

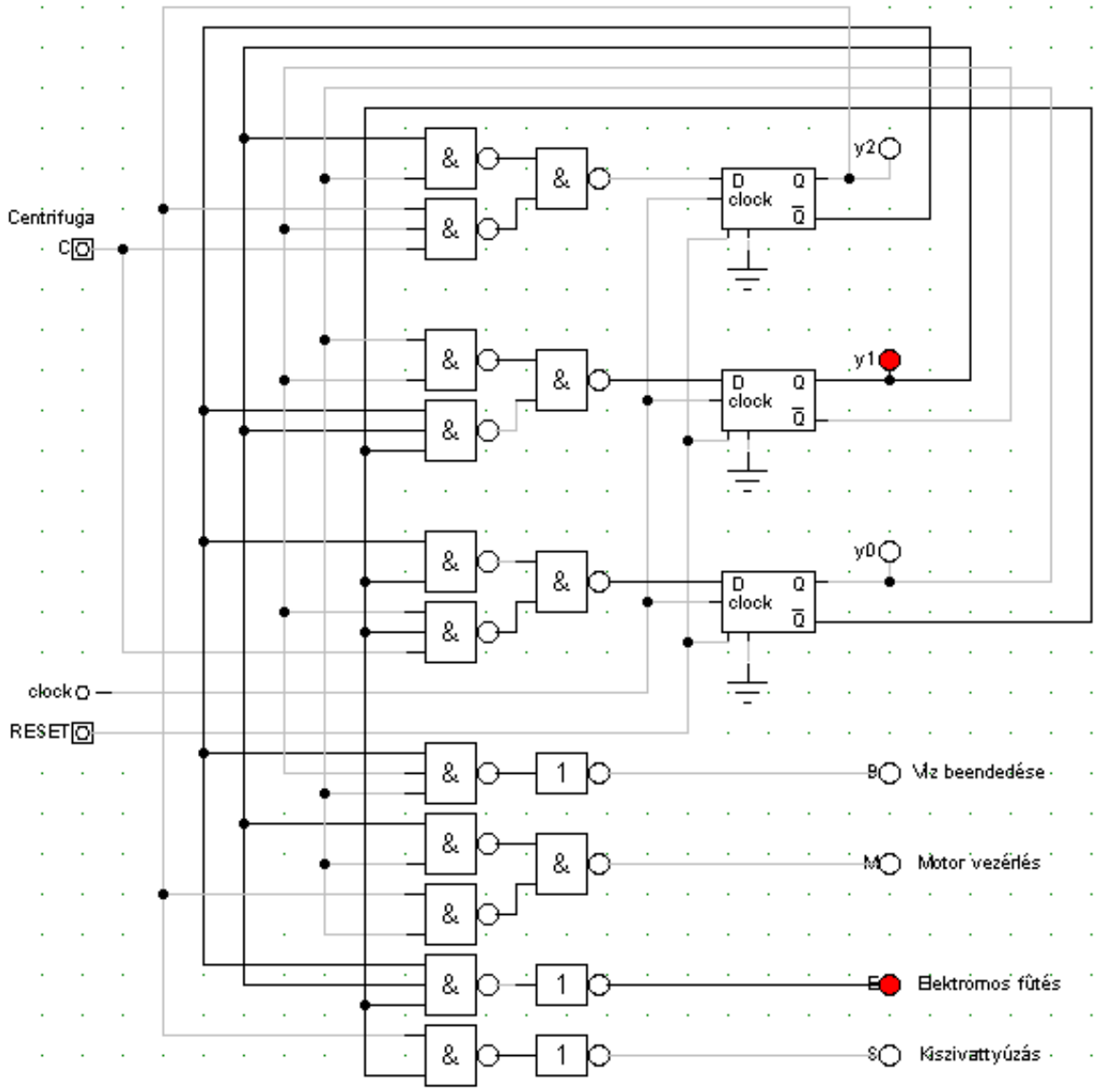
Sorrendi hálózat megvalósítás



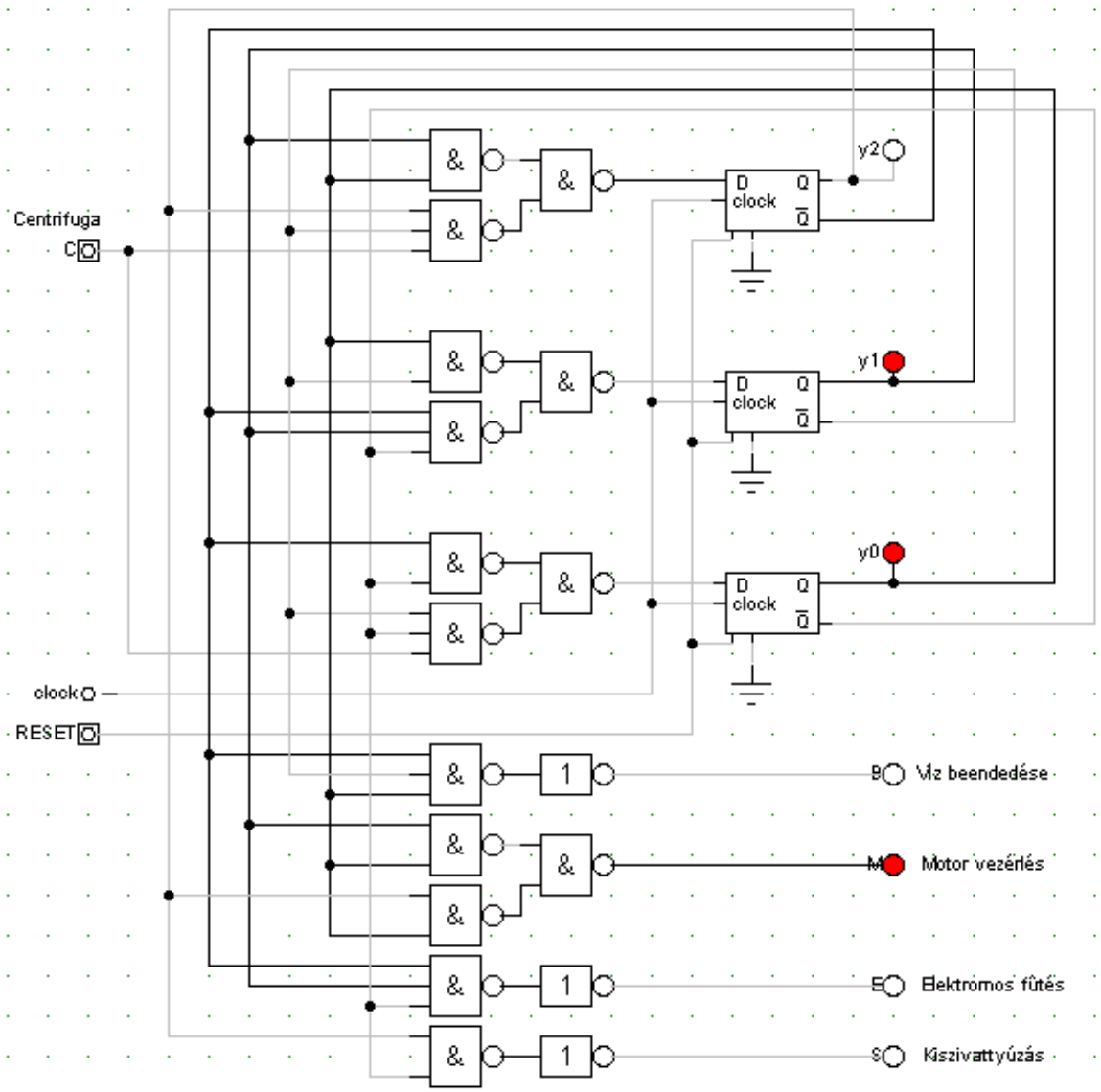
Sorrendi hálózat megvalósítás



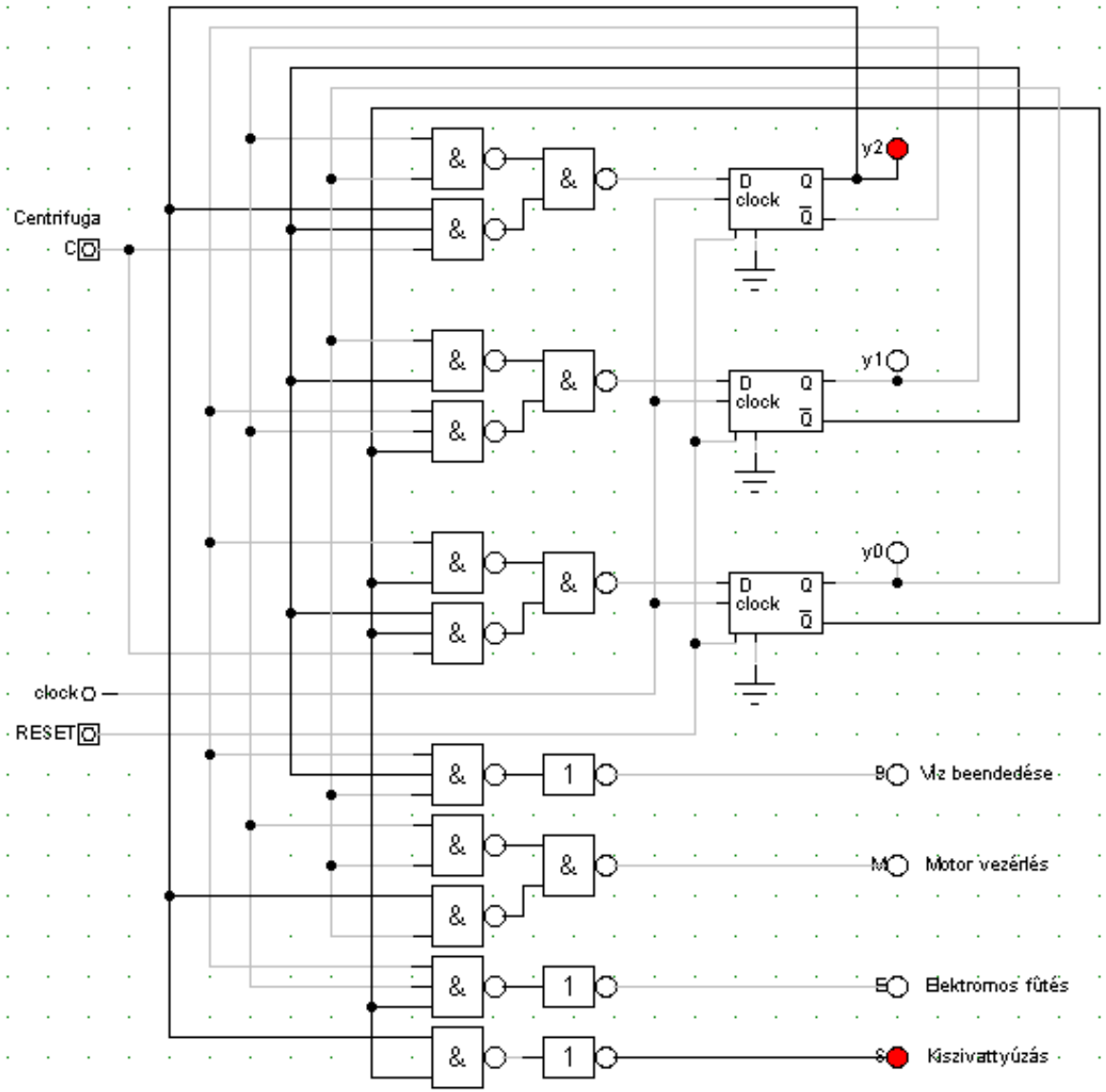
Sorrendi hálózat megvalósítás



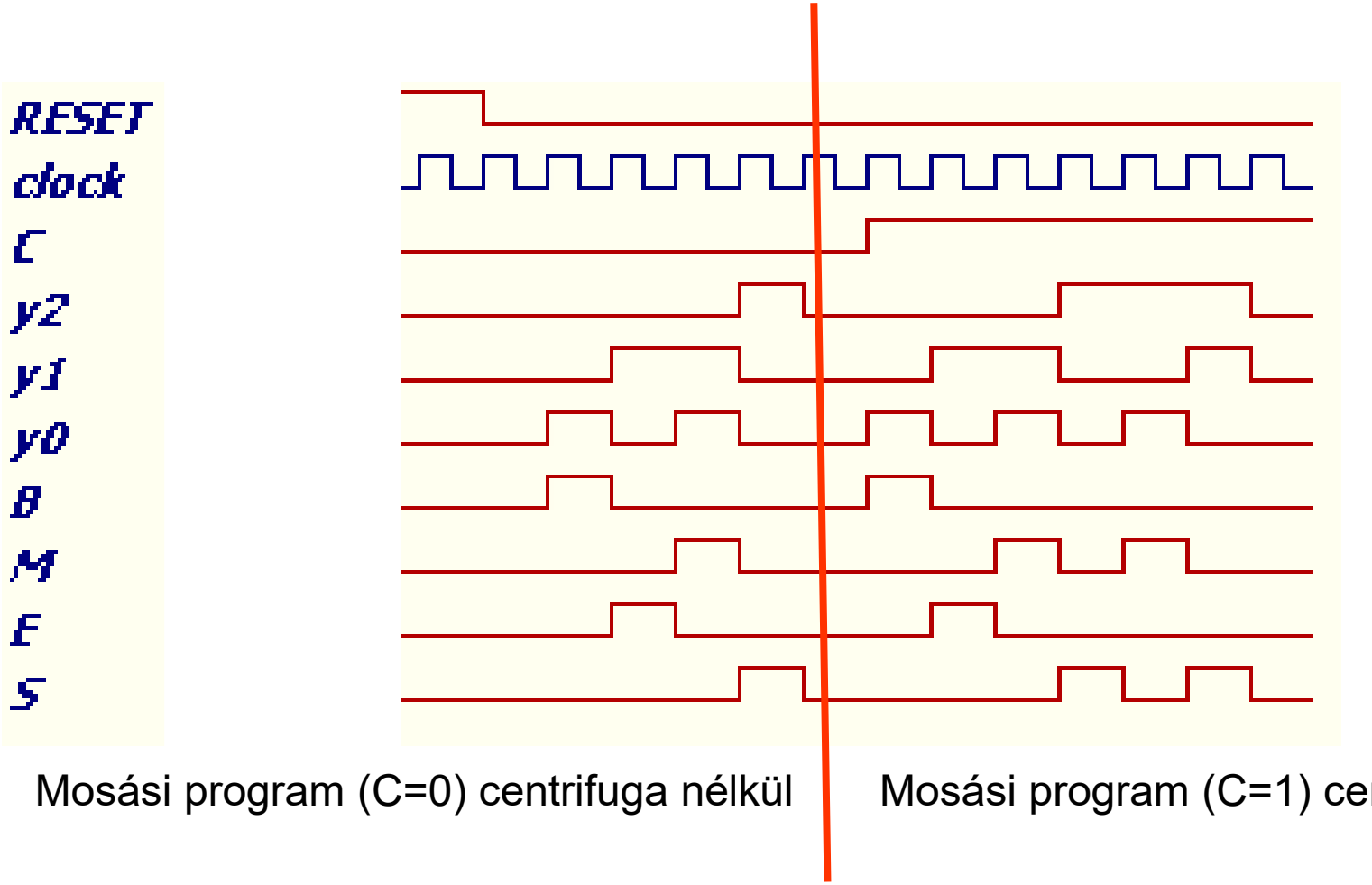
Sorrendi hálózat megvalósítás



Sorrendi hálózat megvalósítás



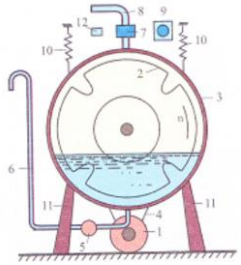
Sorrendi hálózat megvalósítás

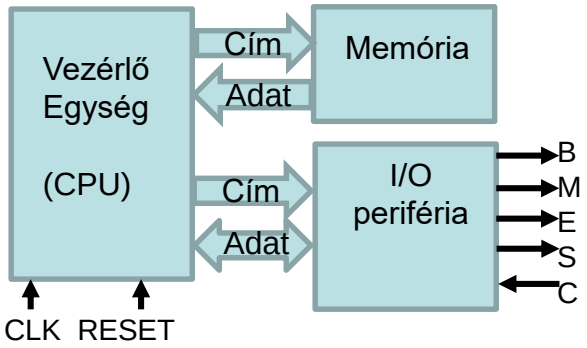


Mosási program (C=0) centrifuga nélkül

Mosási program (C=1) centrifugálással

Számítógép alapú megvalósítás





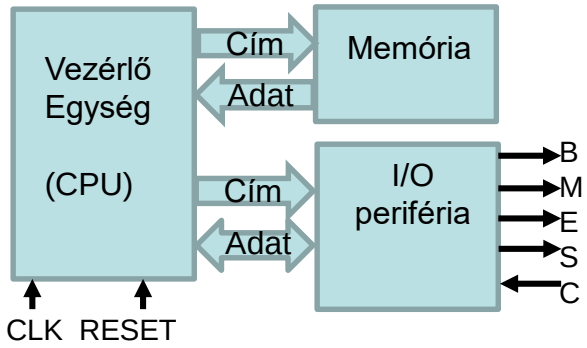
Általános „processzor”
Memóriában tárolt program szerinti vezérlés
Perifériák:

- Négy bites kimeneti regiszter (REG)
- C bemenet centrifuga programhoz

Előny: Csak programot kell írni
Hátrány: Valószínűleg bonyolultabb/drágább lesz

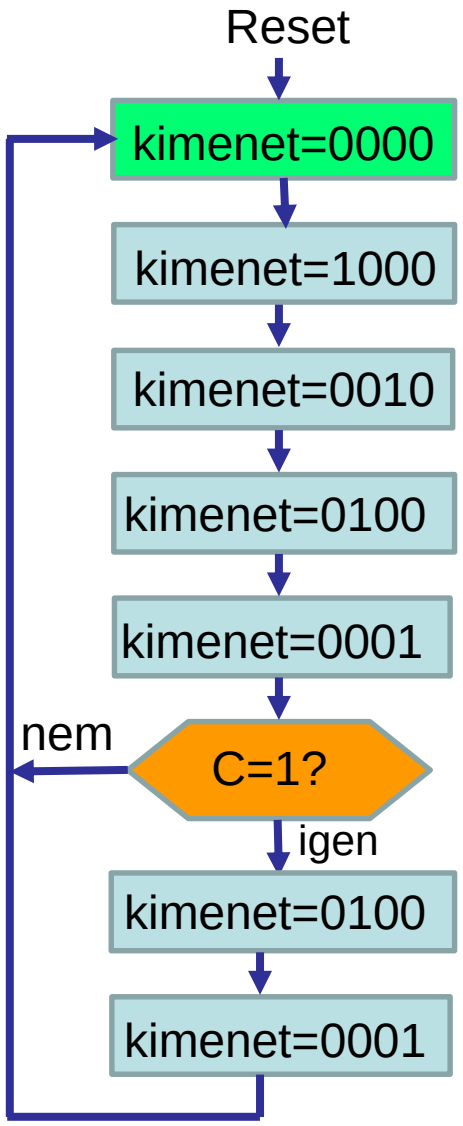
A processzort is meg tudjuk tervezni!

Számítógép alapú megvalósítás



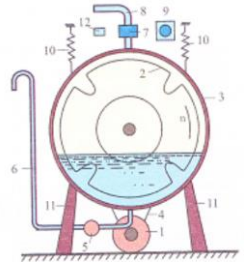
Általános processzor
Memóriában tárolt program szerinti vezérlés
Perifériák:

- Négy bites kimeneti regiszter (REG)
- C bemenet centrifuga programhoz




```
START:
        REG = '0000'  ;kezdőállapot
MAIN:         ; BMES kimenetek
        REG = '1000'  ;Vízbeengedés
        REG = '0010'  ;Melegítés
        REG = '0100'  ;Forgatás
        REG = '0001'  ;Szivattyúzás
        if (C==0) goto MAIN ;vizsgálat
        REG = '0100'  ;Forgatás
        REG = '0001'  ;Szivattyúzás
        goto MAIN
```

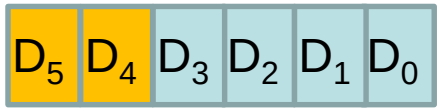
Számítógép alapú megvalósítás



Gépi utasítások tárolása:
Egydimenziós, lineáris címzésű *memória*ban
Bináris formában
Az egyes utasítások formátuma:


```
START:
        REG = '0000'
MAIN:
        REG = '1000'
        REG = '0010'
        REG = '0100'
        REG = '0001'
        if (C==0) goto MAIN
        REG = '0100'
        REG = '0001'
        goto MAIN
```

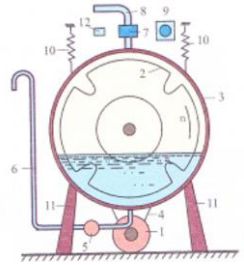
Három műveletre van szükségünk → 2 bit elegendő a művelet kódolásához
Négy kimenetünk van → válasszuk 4bitesre az adat hosszát
Egy utasítás tárolásához összesen 2+4 = 6 bit szükséges



Összesen 8 utasításból áll a programunk, így legalább 8db 6 bites utasítás tárolására alkalmas memóriára lesz szükségünk.
Ha figyelembe vesszük, hogy négy bites adataink vannak, akkor célszerű a címbitek számát is négyre választani, így későbbi fejlesztésre is marad tartalékunk…

Ha 4 bittel címzünk 16 különböző memóriarekesz címezhető…

Számítógép alapú megvalósítás



```
START:
        REG = '0000'
MAIN:
        REG = '1000'
        REG = '0010'
        REG = '0100'
        REG = '0001'
        if (C==0) goto MAIN
        REG = '0100'
        REG = '0001'
        goto MAIN
```

Pl: REG='1000'→101000

Utasítások kódolása

	D_5	D_4	D_3	D_2	D_1	D_0	
Ugrás:	0	0	4 bites cím				**JMP** cím
Feltételes ugrás:	0	1	4 bites cím				**JNC** cím
Kimenet állítása:	1	0	4 bites adat				**OUT** adat

Fogalmak:
- Mnemonik (utasítás rövid neve)
- Assembly program
- Assembler
- Gépikód

„Assembly" program:

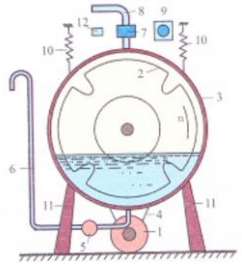
```
START:
        OUT  '0000'
MAIN:
        OUT  '1000'
        OUT  '0010'
        OUT  '0100'
        OUT  '0001'
        JNC   MAIN
        OUT  '0100'
        OUT  '0001'
        JMP   MAIN
```

Assembler

Gépikód (memória tartalom)

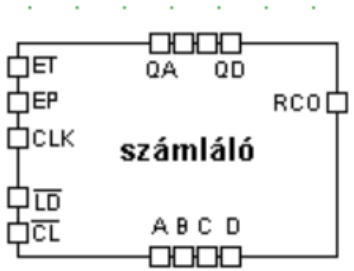
Cím	Adat (bináris)
0	100000
1	101000
2	100010
3	100100
4	100001
5	010001
6	100100
7	100001
8	000001

Számítógép alapú megvalósítás



Végrehajtó egység tervezése

- Memóriacímek generálása → 4 bites számláló
- Következő utasítás címének előállítása → inkrementálás
- Ugró utasítás → számláló LOAD művelet
- Feltételes ugrás → logikai hálózattal LOAD engedélyezése
- OUT utasítás → regiszter betöltés engedélyezése



D_5 D_4 D_3 D_2 D_1 D_0

JMP cím: 0 0 4 bites cím

JNC cím: 0 1 4 bites cím

Ugró utasítás, ha

D5	D4	C	-LD
0	0	0	0
0	0	1	0
0	1	0	0

Betölteni D0…D3 biteket kell az A,B,C,D bemeneteken keresztül

OUT adat: 1 0 4 bites adat

OUT művelet, ha D_5=1
(lefutó élre írjuk be a regisztert)

$$\overline{LD} = D_5 + D_4 \cdot C$$

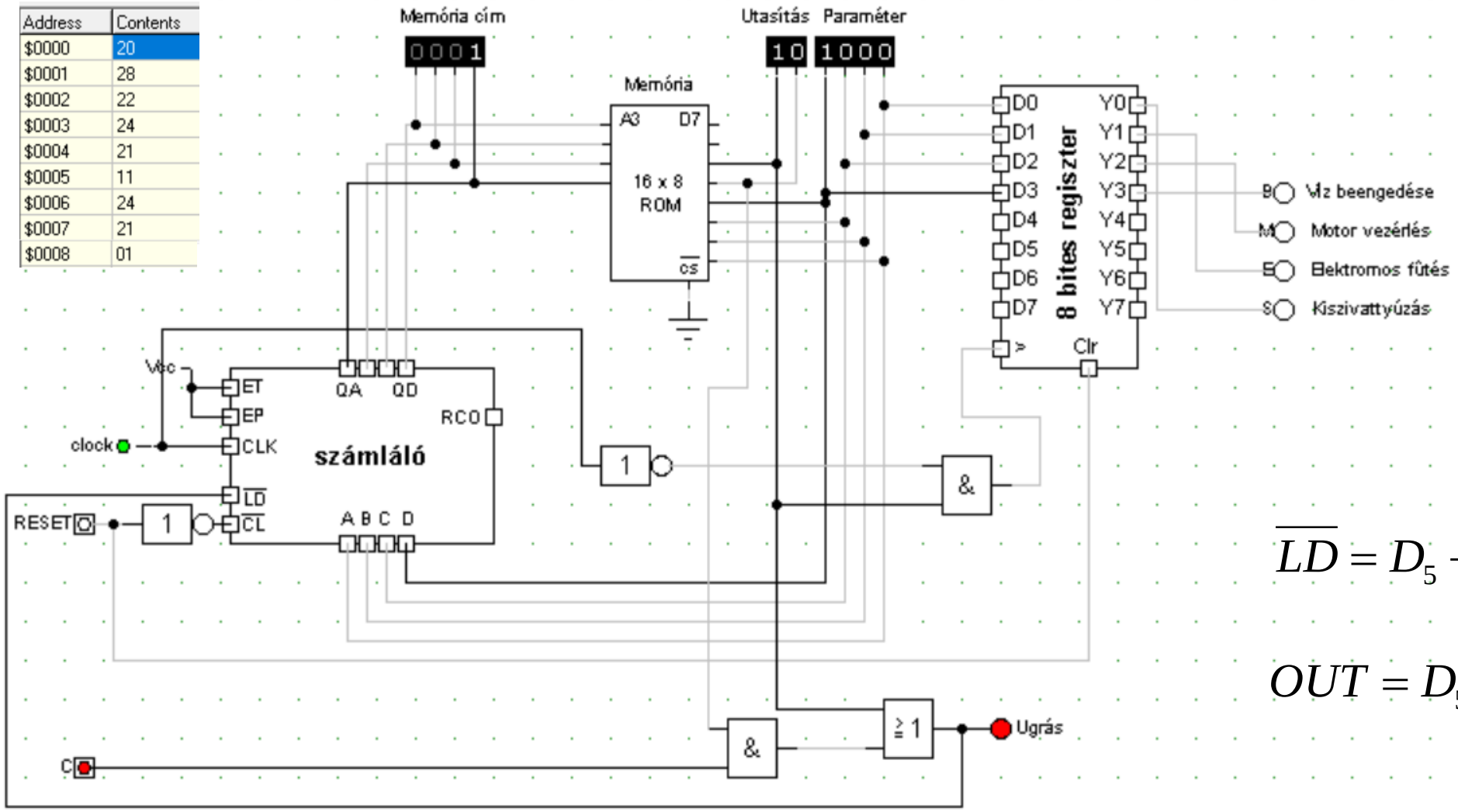
$$OUT = D_5 \cdot \overline{CLK}$$

MIPSh vezérlőegység terve

MIPSh → Mosógép Irányító Programozható Sorrendi hálózat ☺

Memória tartalma:

Address	Contents
$0000	20
$0001	28
$0002	22
$0003	24
$0004	21
$0005	11
$0006	24
$0007	21
$0008	01

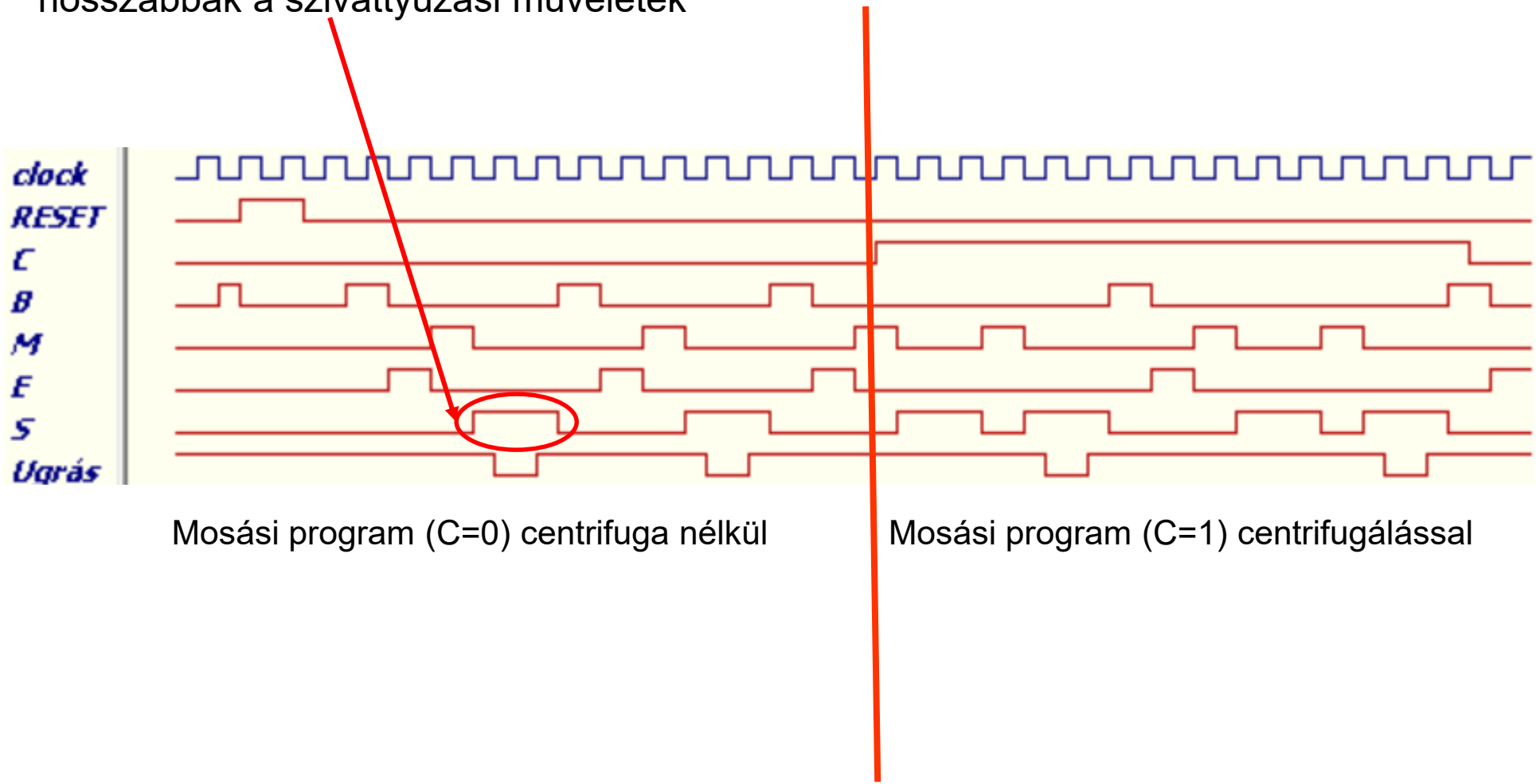


$$\overline{LD} = D_5 + D_4 \cdot C$$

$$OUT = D_5 \cdot \overline{CLK}$$

MIPSh vezérlőegység működése

Figyeljük meg, hogy az ugrás ütemében a kimenetek nem változnak, ezért hosszabbak a szivattyúzási műveletek



Mosási program (C=0) centrifuga nélkül

Mosási program (C=1) centrifugálással

Specifikáció változás…

A megrendelő kéri az alábbi módosításokat:

- Centrifugálással egyidőben induljon a szivattyúzás is
- A centrifugálás 2 időegységig tartson, (utána még 1 ütem szivattyúzás marad)
- A mosás utáni szivattyúzás is 2 időegységig tartson

Kódolt állapottábla

	$y_3y_2y_1y_0$IC	0	1	BMES
(S_0)	0000	0001	0001	0000
(S_1)	0001	0010	0010	1000
(S_2)	0010	0011	0011	0010
(S_3)	0011	0100	0100	0100
(S_4)	0100	0101	0101	0001
(S_5)	0101	0000	0110	0001
(S_6)	0110	---	0111	0101
(S_7)	0111	---	1000	0101
(S_8)	1000	---	0000	0001

→ Öt változós függvény! 4 flip-flop!

```
START:
        OUT  '0000'
MAIN:
        OUT  '1000'
        OUT  '0010'
        OUT  '0100'
        OUT  '0001'
        OUT  '0001'
        JNC   MAIN
        OUT  '0101'
        OUT  '0101'
        OUT  '0001'
        JMP   MAIN
```

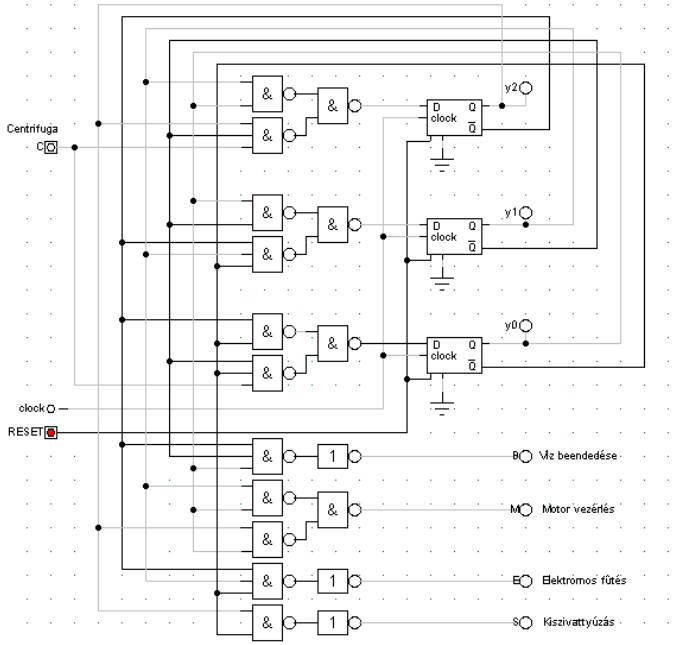
→ Program módosítás!

→ Hardver marad!

Összehasonlítás

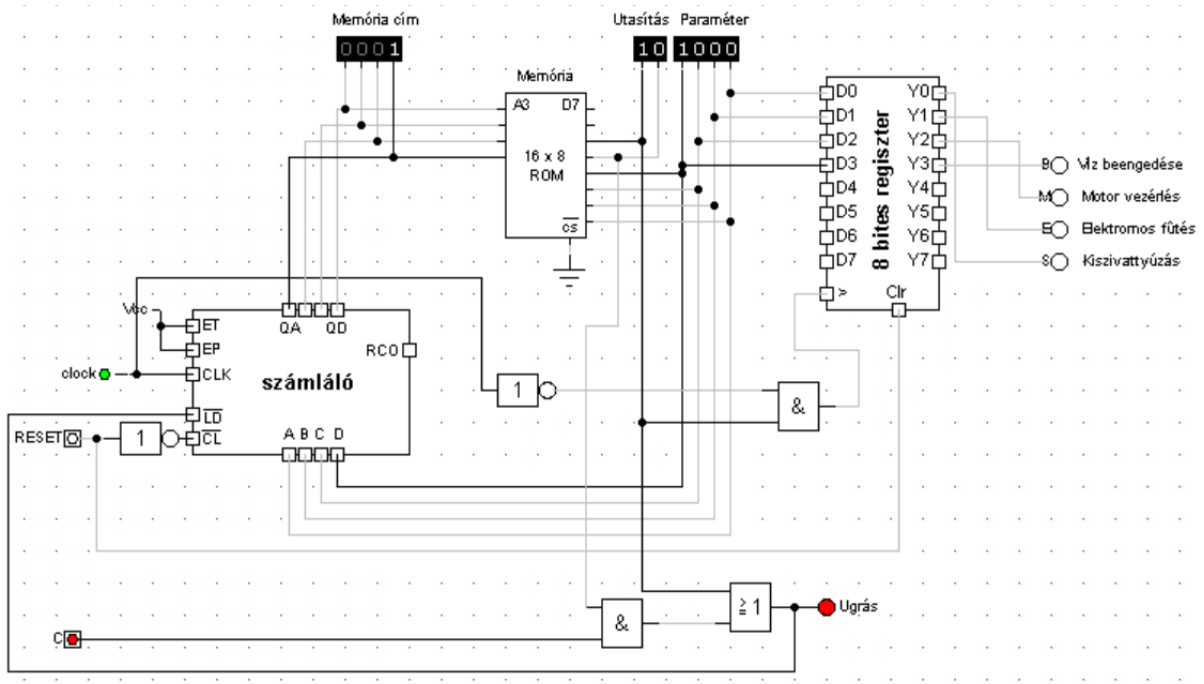
Sorendi hálózat alapú megvalósítás

- Szinkron sorrendi hálózat
- Leggyorsabb működés
- Néhány flip-flop elegendő (3db)
- Program módosítása teljes újratervezést igényel
- Hosszú tervezési idő
- Időzítések nehezen változtathatók



Processzor alapú megvalósítás

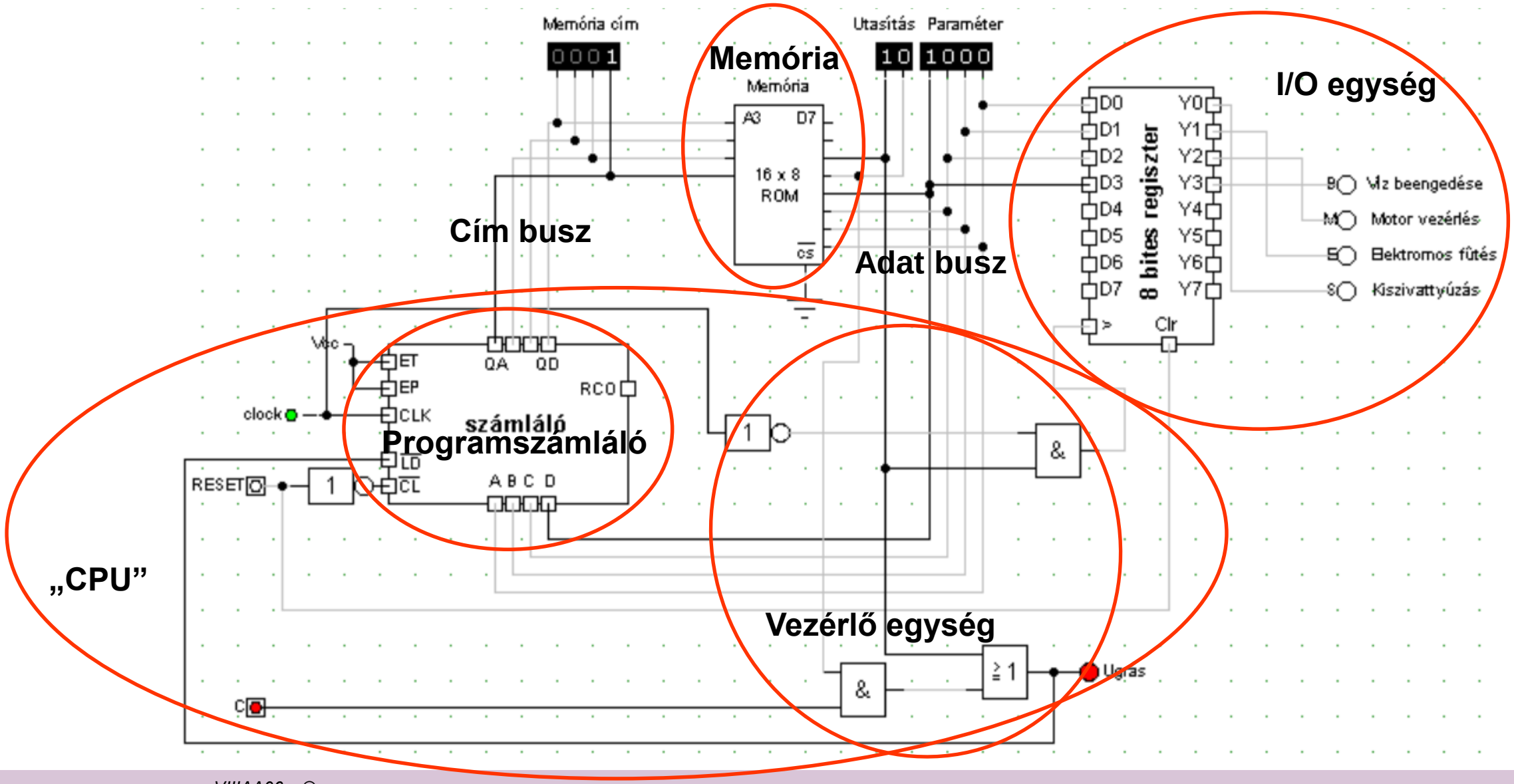
- Lassabb működés
- Bonyolult általános vezérlő
- Program módosítása könnyű
- Rövid tervezési idő (programozás)
- Időzítések programból változtathatók



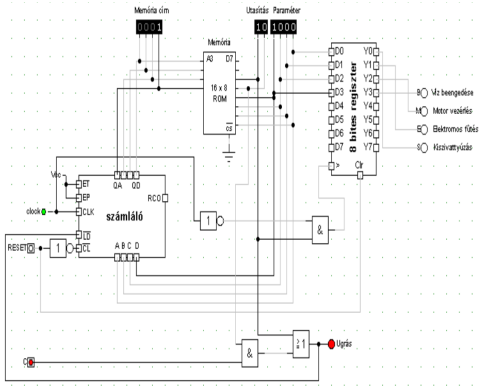
Általános célú processzor vagy célhardver

- Célhardver:
  - Speciális célú sorrendi hálózat egy bizonyos feladat **hatékony** végrehajtására
  - **Csak az adott feladat** elvégzésére képes
    - » Másik feladatra terveznünk kell egy másikat…
- Processzor:
  - Speciális célú sorrendi hálózat egy adott **utasításkészlet** utasításainak hatékony végrehajtására
  - Az utasításkészlet felhasználásával tetszőleges feladat megoldására **beprogramozható**
    - » Csak meg kell tanulni programozni

Általános célú számítógép felépítése

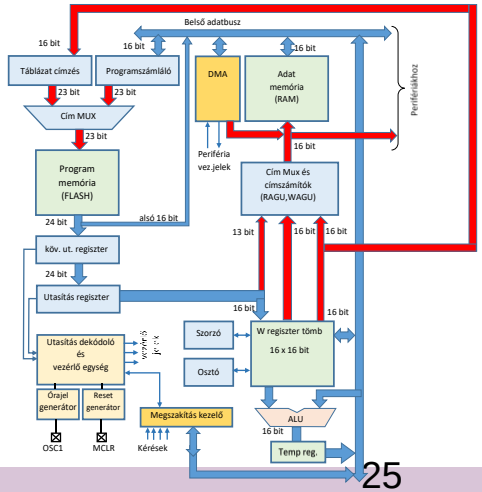


A mikrokontroller amit tanulni fogunk…



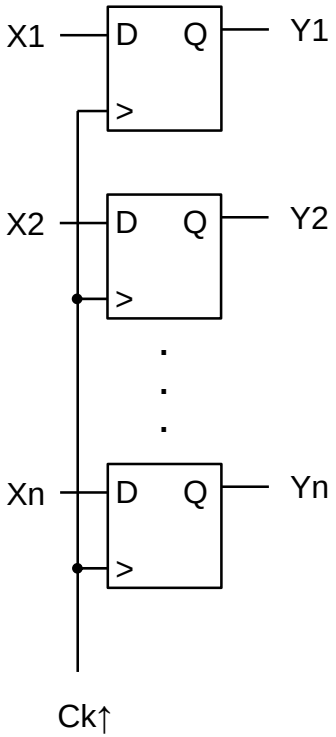
- 3 db különböző utasítás
- 6 bites utasítások
- 4 bites címzés
  →16 db utasítás lehet maximum a program
- 4 bites adatok
- 1 db kimenő regiszter, mint periféria

- 70 db különböző utasítás
- 24 bites utasítások
- 23 bites címzés utasításokra
  → 4 millió utasítás lehet maximum a program
- 16 bites adatok
- Többszáz ki/bemeneti regiszter



Regiszterek

Közös órajellel vezérelt D flip-flop csoport
Összetartozó adatbitek tárolása
A tárolás az órajel felfutó élére történik

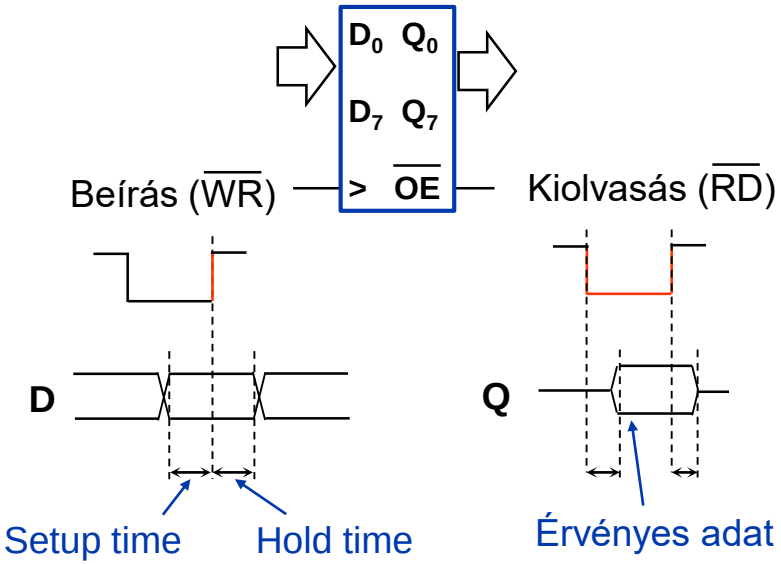


Műveletek:

Beírás ($\overline{WR}$)

Kiolvasás ($\overline{RD}$)

Regiszter three-state kimenettel

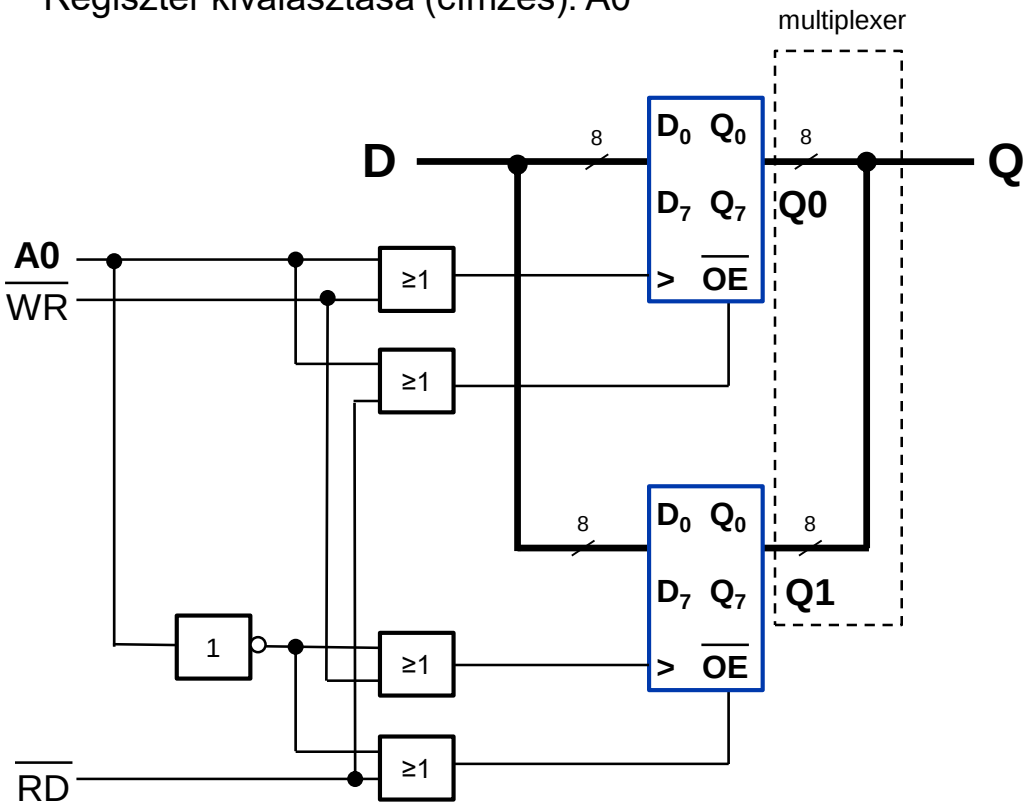


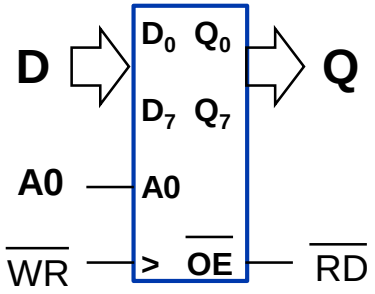
74LS374	20ns	1ns	28ns	20ns

Regiszter tömb

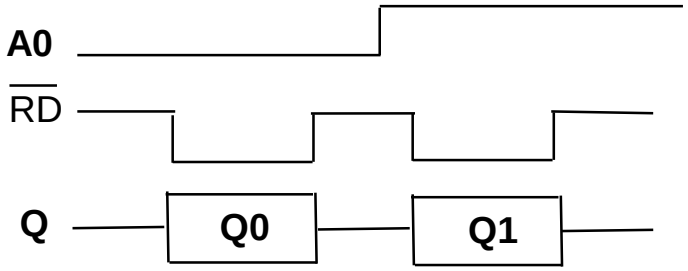
2 regiszter

Regiszter kiválasztása (címzés): A0





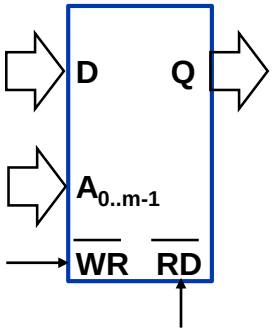
A0	$\overline{WR}$	$\overline{RD}$	Művelet
0	0	1	0. regiszter írása
0	1	0	0. regiszter olvasása
1	0	1	1. regiszter írása
1	1	0	1. regiszter olvasása
x	1	1	nincs művelet

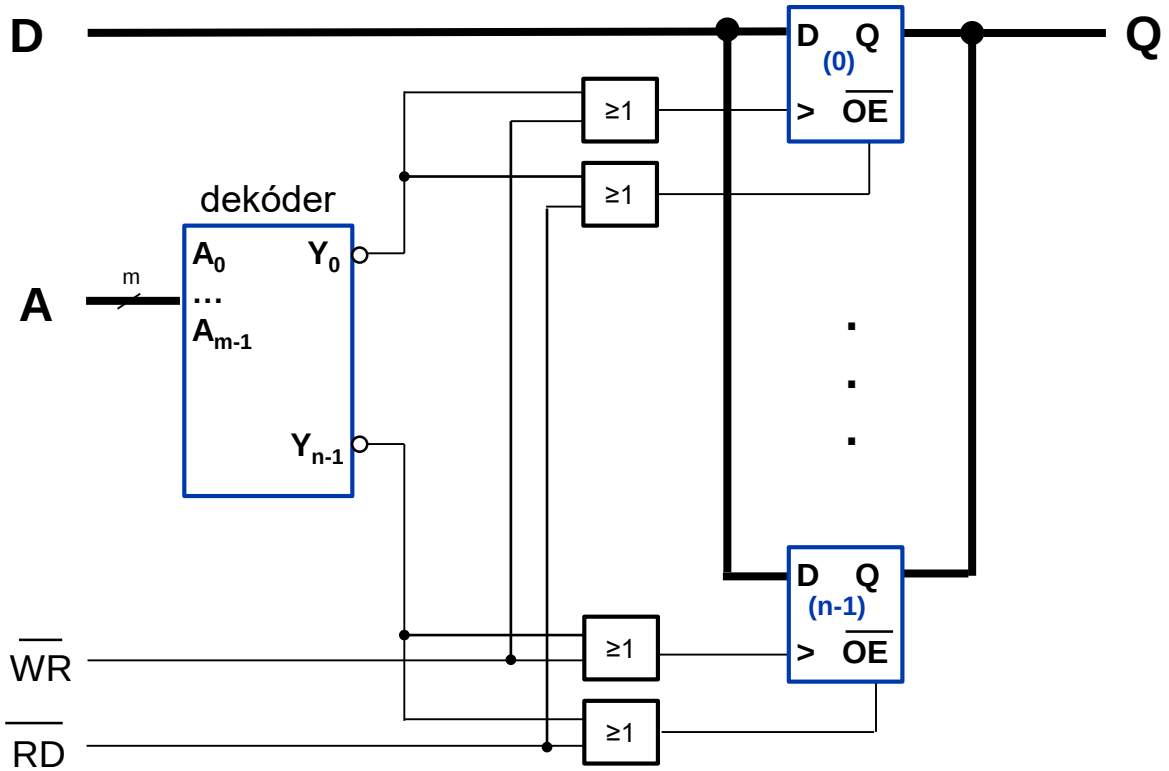


Regiszter tömb

n regiszter ($n = 2^m$)

Regiszter kiválasztása (címzés): $A_0 \ldots A_{m-1}$ (m db címvezeték)





Memória

Adatok tárolására alkalmas (nagy) regiszter tömb

ROM (read only memory)
- maszk programozott
- PROM (egyszer írható)
- EPROM (UV törölhető, újraírható)
- EEPROM (elektronikusan törölhető, újraírható)

Tárolt bitek száma 2^n

	kilo	Mega	Giga	Tera	Peta
bit →	kbit →	Mbit →	Gbit →	Tbit →	Pbit
2^0	2^{10}	2^{20}	2^{30}	2^{40}	2^{50}

RAM (random access memory)
- statikus (tárolás flip-flopban)
- dinamikus (tárolás kondenzátorban)
  → frissítés

Szervezés
egy művelettel elérhető bitek száma: 2^m

2^0	2^2	2^3	2^4	2^5	2^6	2^7	….
1	4	8	16	32	64	128	….
		byte	word	double word	quad word		

1 kbyte = 1024 byte = 8 kbit = 8192 bit