Hardver alapok

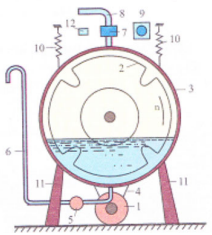
(VIIIBA01)

Mikrokontroller alapok

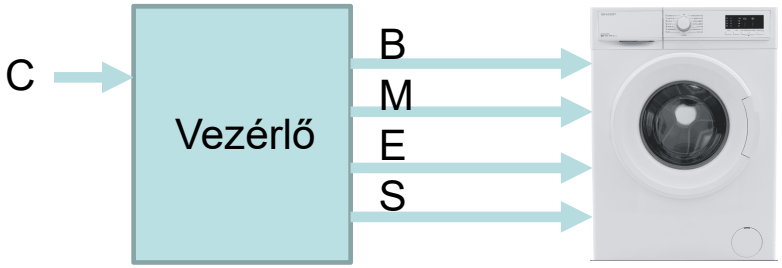
Rácz György gyuriracz@iit.bme.hu

2022/23. tanév őszi félév

„Bemelegítő” 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

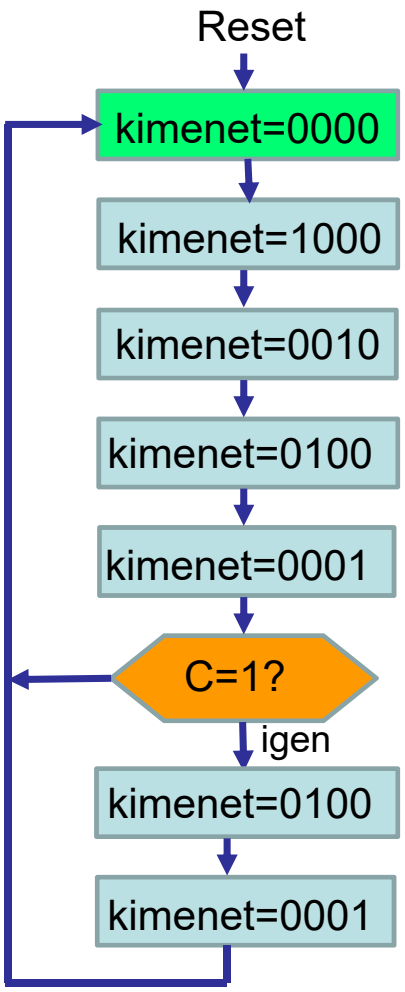
Folyamatábra

Mosási program (C=0) centrifuga nélkül

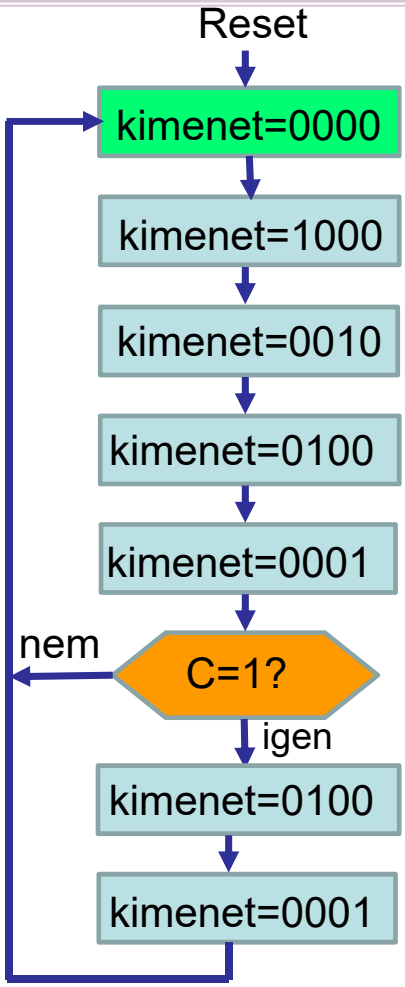
Lépés	Művelet	B	M	E	S
1.	Víz beengedés	1	0	0	0
2.	Melegítés	0	0	1	0
3.	Forgatás	0	1	0	0
4.	Ürítés	0	0	0	1

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

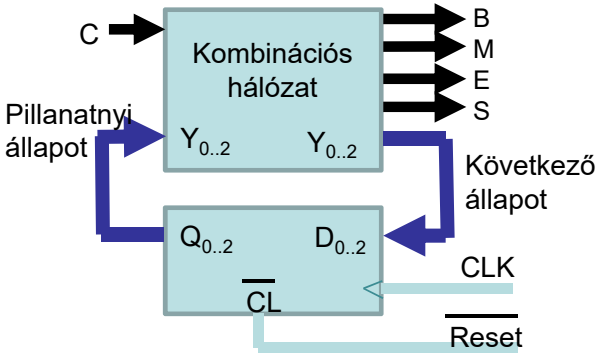


Folyamatábra



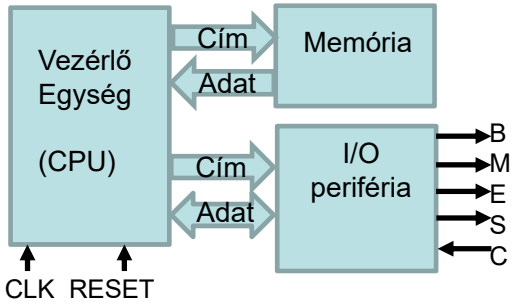
Megvalósítás #1

Szinkron Moore sorrendi hálózattal

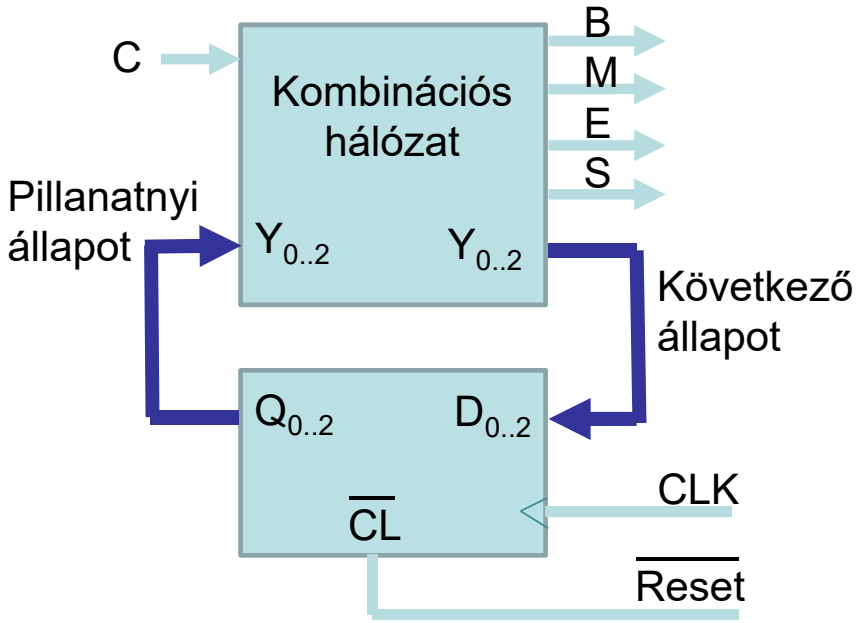


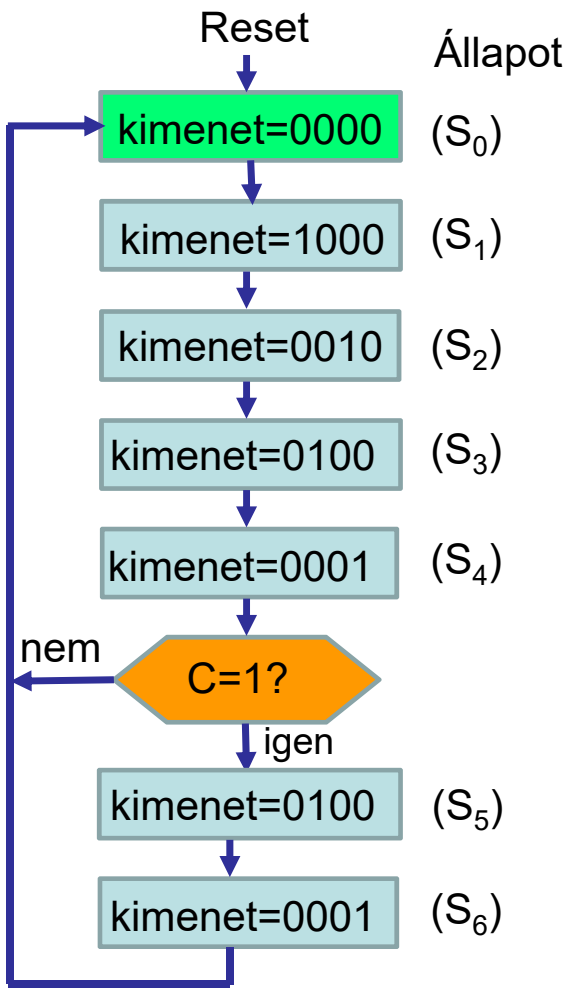
Megvalósítás #2

Mikroszámítógéppel



Sorrendi hálózat megvalósítás



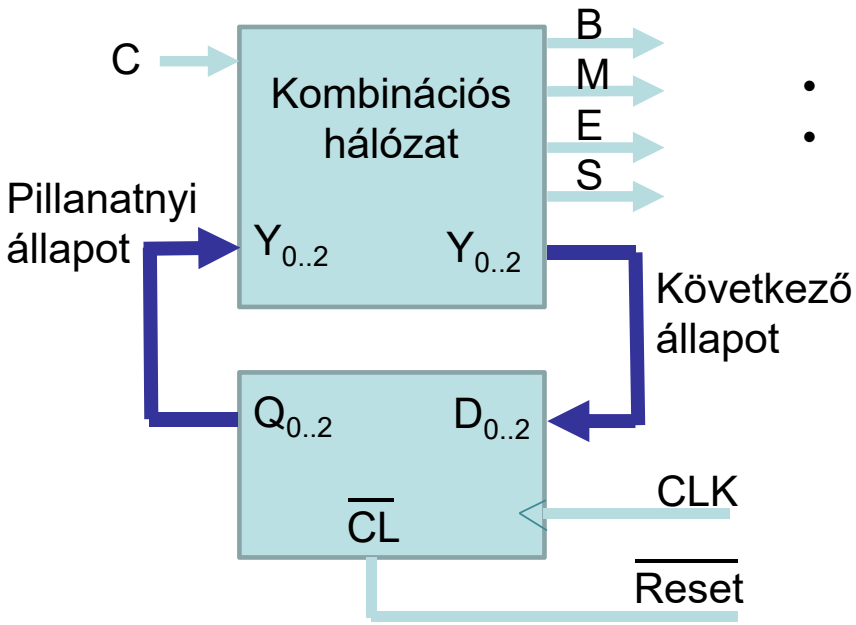


Megvalósítás #1

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

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

Sorrendi hálózat (FSM) megvalósítás



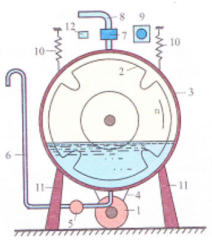
Megvalósítás #1

- Szinkron sorrendi hálózattal
- Moore modell

Kódolt állapottábla

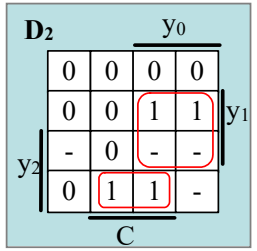
	$y_2y_1y_0$\C	0	1	BMES
(S_0)	000	001	001	0000
(S_1)	001	010	010	1000
(S_2)	010	011	011	0010
(S_3)	011	100	100	0100
(S_4)	100	000	101	0001
(S_5)	101	---	110	0100
(S_6)	110	---	000	0001
	111	---	---	----

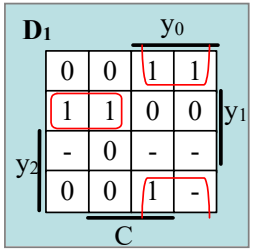
Sorrendi hálózat megvalósítás

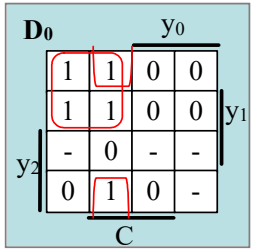


Kódolt állapottábla

$y_2y_1y_0$\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	---	---	----



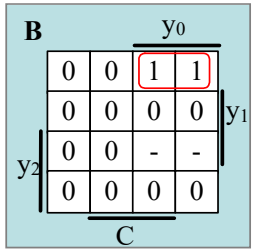


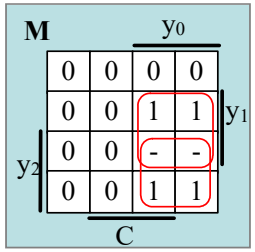


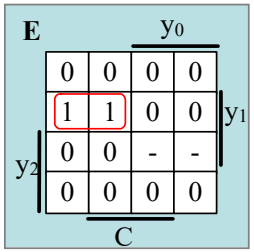
$$D_2 = y_1 \cdot y_0 + y_2 \cdot \overline{y_1} \cdot C$$

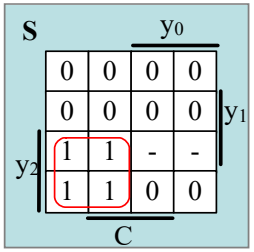
$$D_1 = y_0 \cdot \overline{y_1} + \overline{y_2} \cdot y_1 \cdot \overline{y_0}$$

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









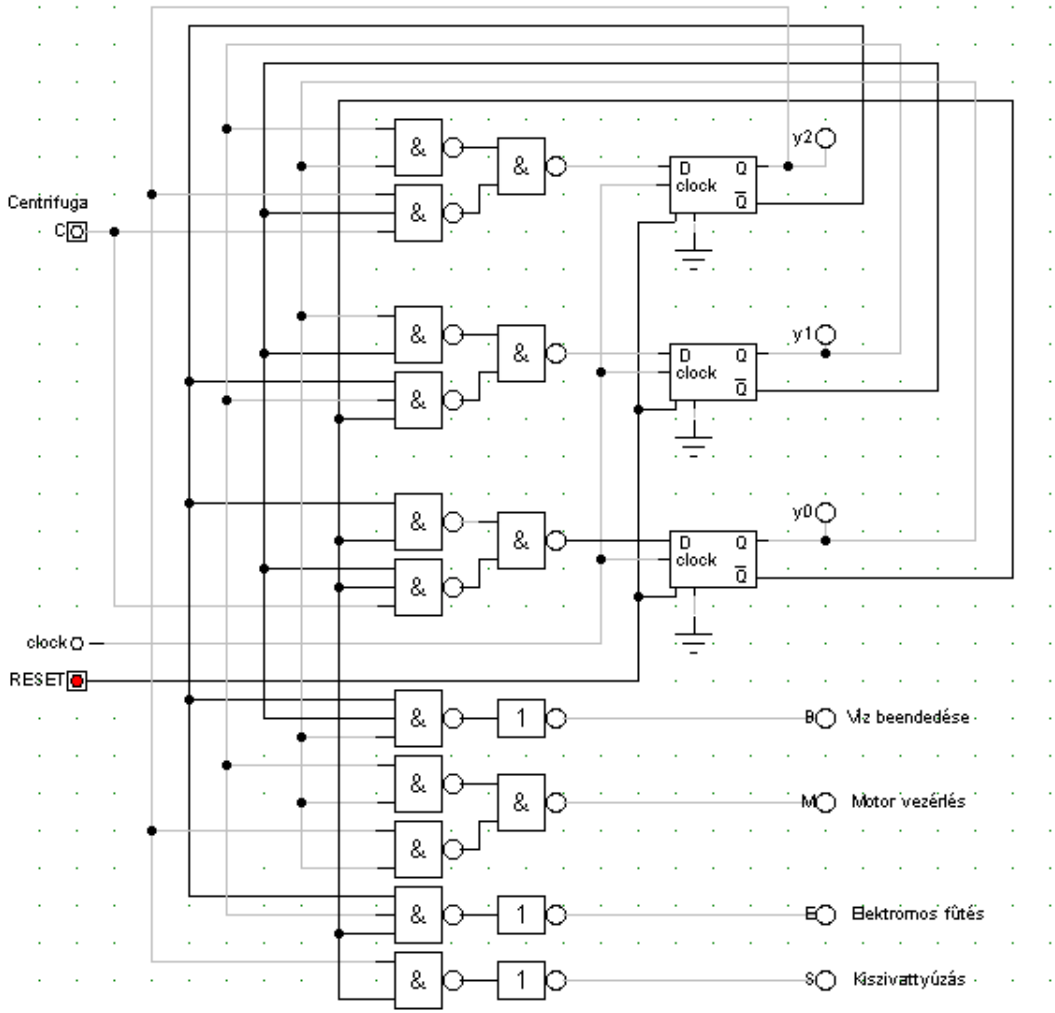
$$B = \overline{y_2} \cdot \overline{y_1} \cdot y_0$$

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

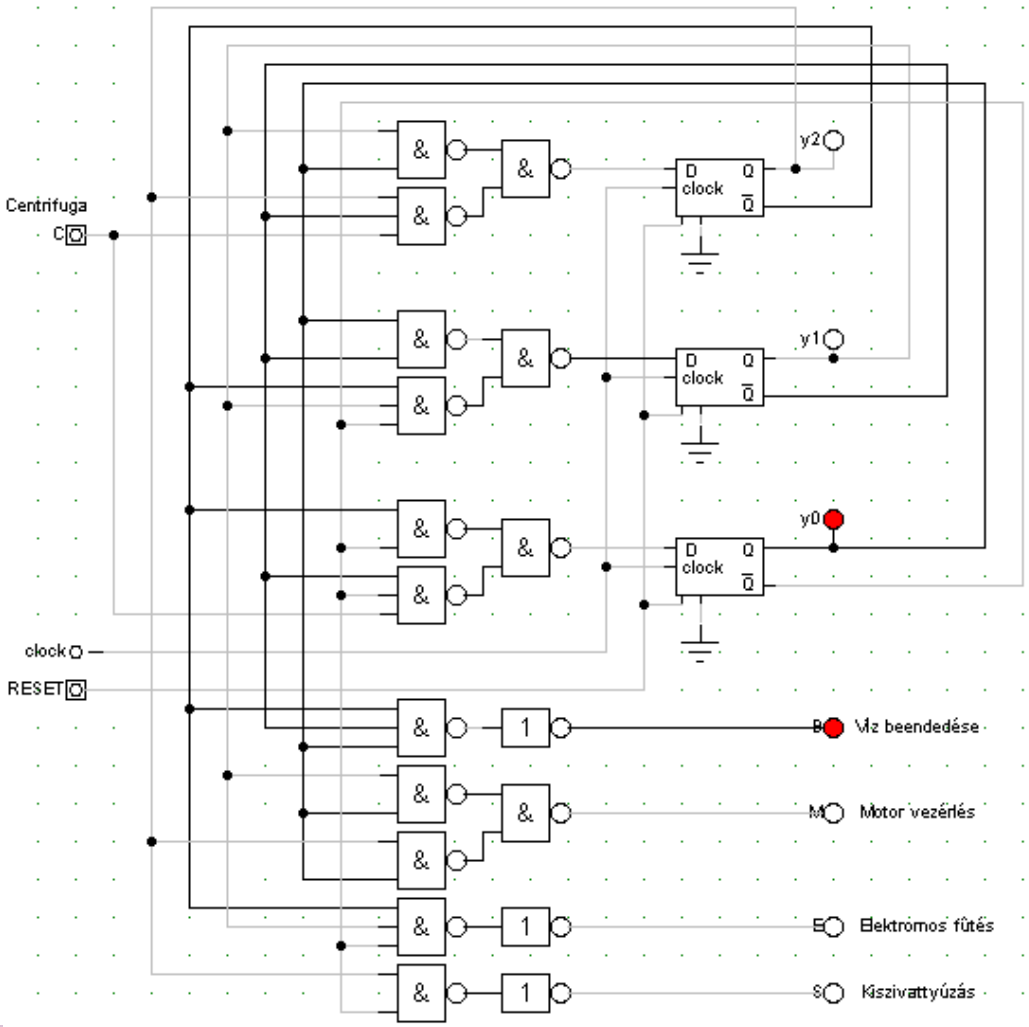
$$E = \overline{y_2} \cdot y_1 \cdot \overline{y_0}$$

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

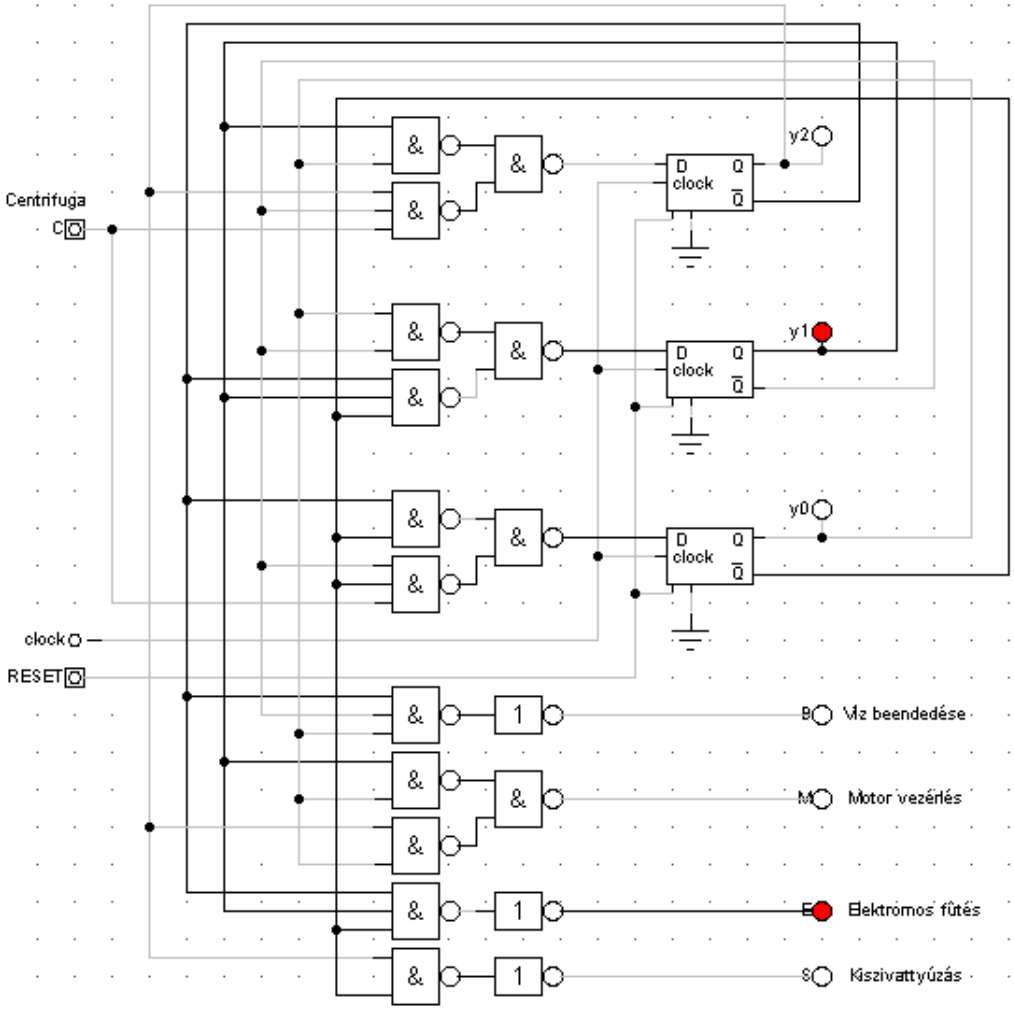
Sorrendi hálózat megvalósítás



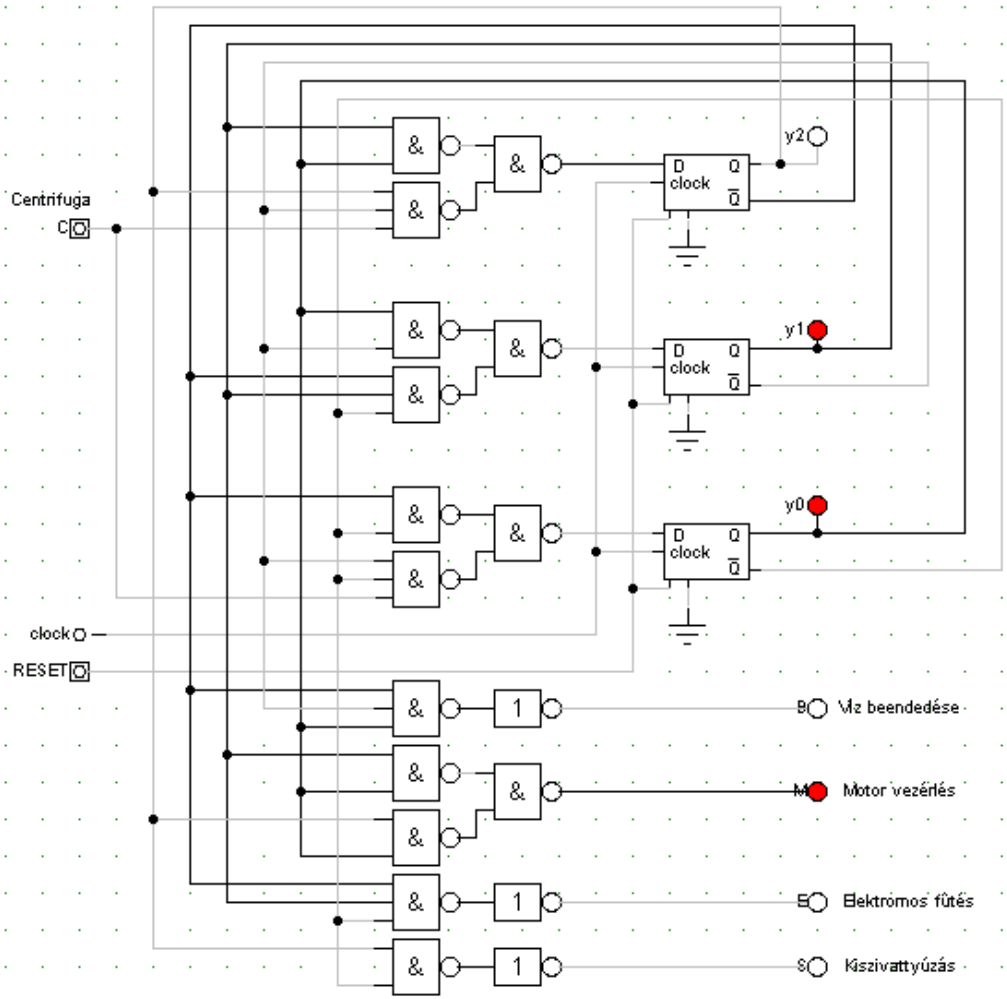
Sorrendi hálózat megvalósítás



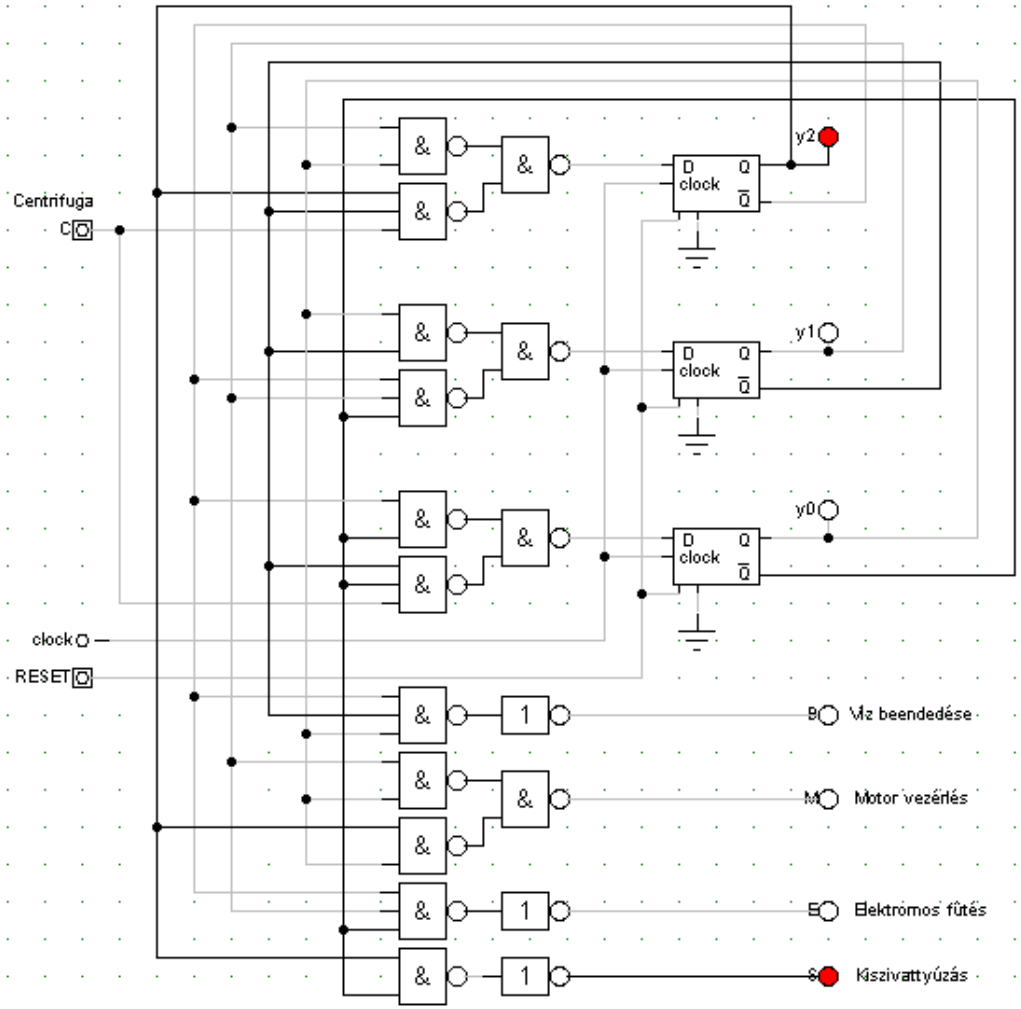
Sorrendi hálózat megvalósítás



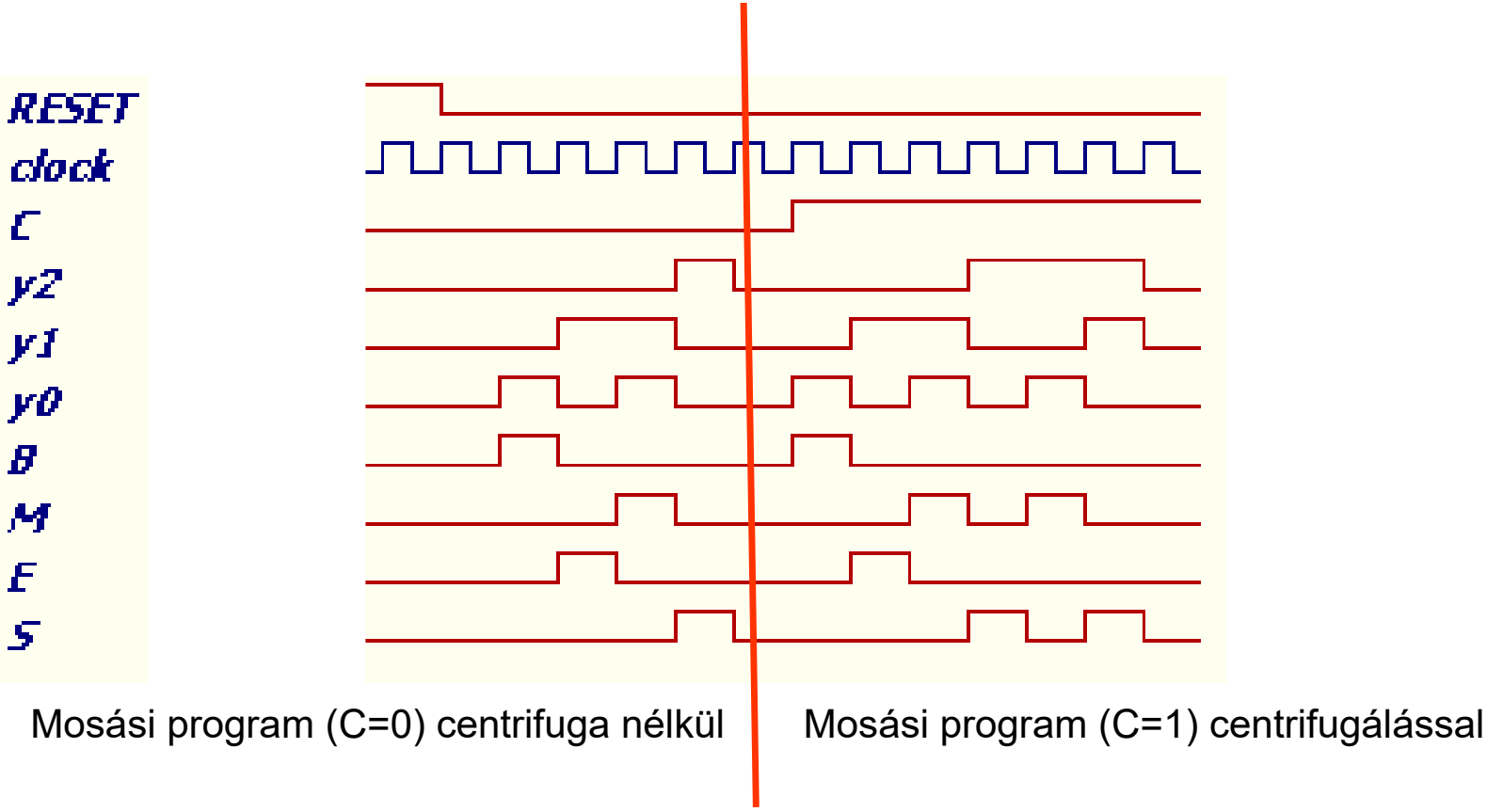
Sorrendi hálózat megvalósítás



Sorrendi hálózat megvalósítás



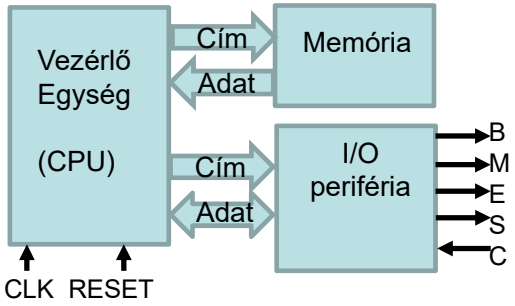
Sorrendi hálózat megvalósítás



Mosási program (C=0) centrifuga nélkül

Mosási program (C=1) centrifugálással

Számítógép alapú megvalósítás

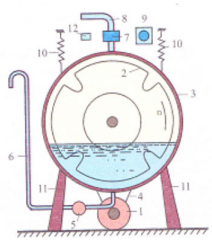


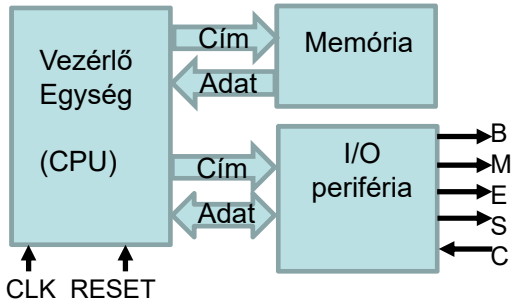
Általános „processzor”
Memóriában tárolt program szerinti vezérlés
Perifériák:
- Négy bites kimeneti regiszter (REG)
- C bemenet centrifuga programhoz

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

A processzort is meg tudjuk tervezni!

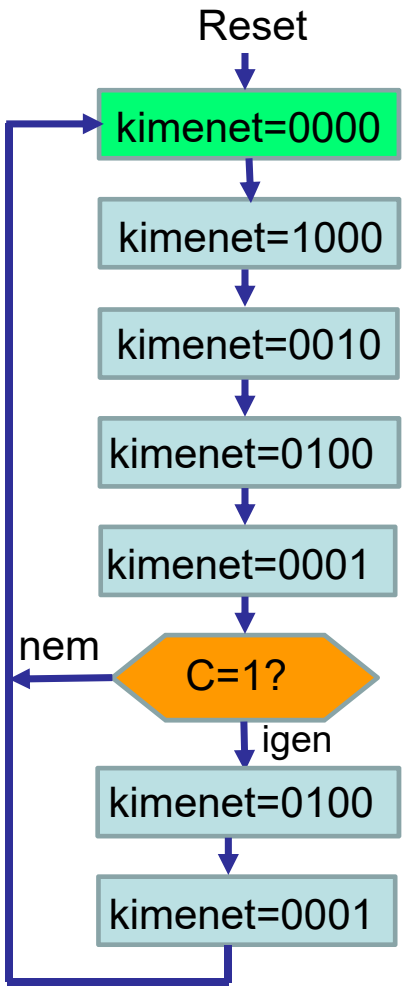
Számítógép alapú megvalósítás





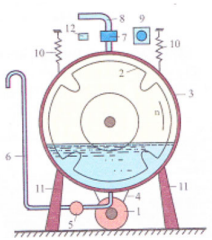
Általános processzor
Memóriában tárolt program szerinti vezérlés
Perifériák:

- 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 *memóriá*ban
Bináris formában
Az egyes utasítások formátuma:

Művelet	Adat

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

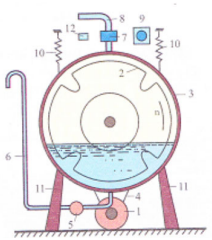
Három műveletre van szükségünk → 2 bit elegendő a művelet kódolásához
Négy kimenetünk van → válasszuk 4bitesre az adat hosszát
Egy utasítás tárolásához összesen 2+4 = 6 bit szükséges

D_5	D_4	D_3	D_2	D_1	D_0

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

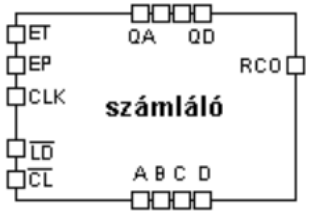
Gépikód (memória tartalom)

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

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

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

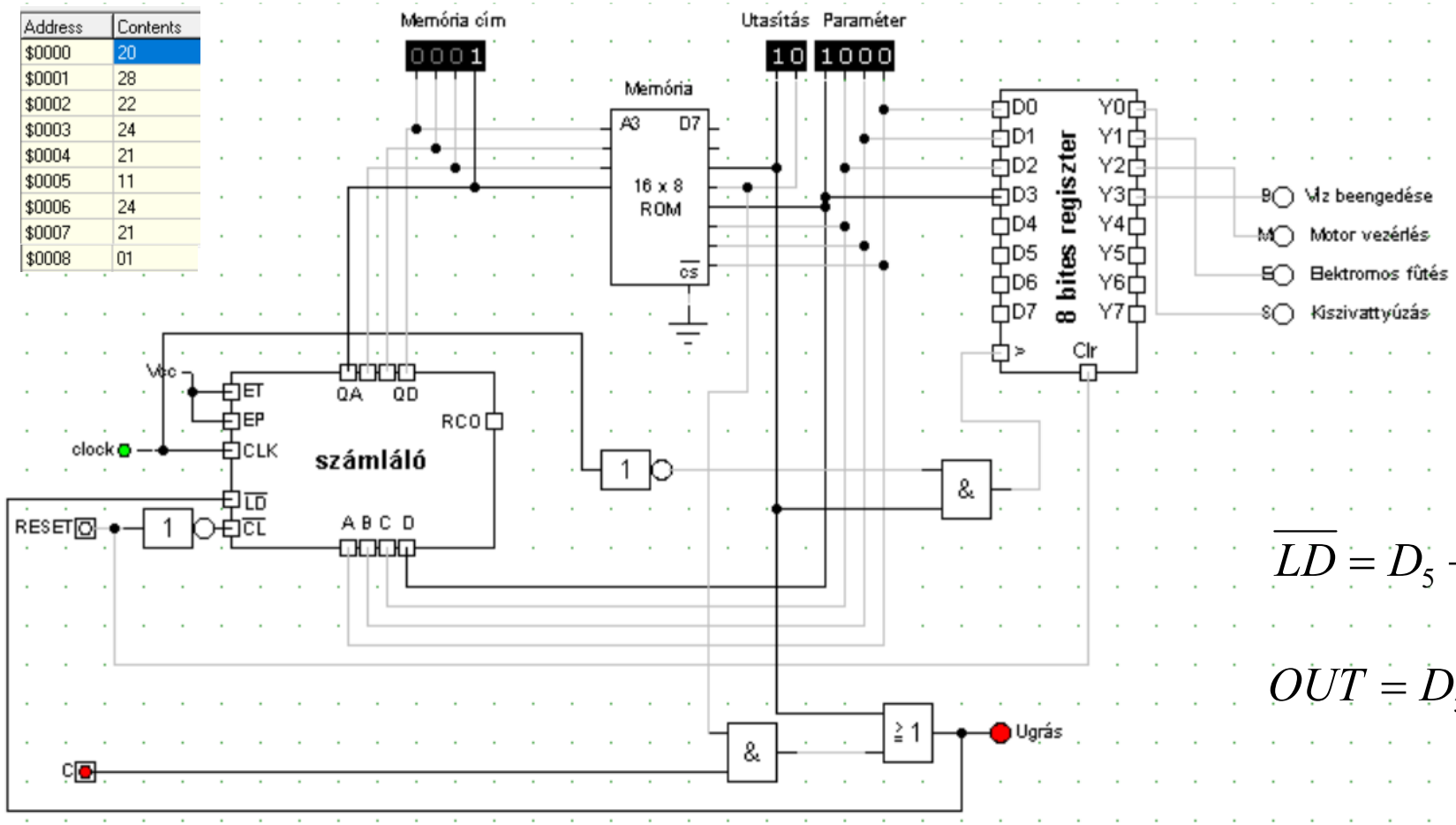
OUT művelet, ha D_5=1
(lefutó élre írjuk be a regisztert)

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

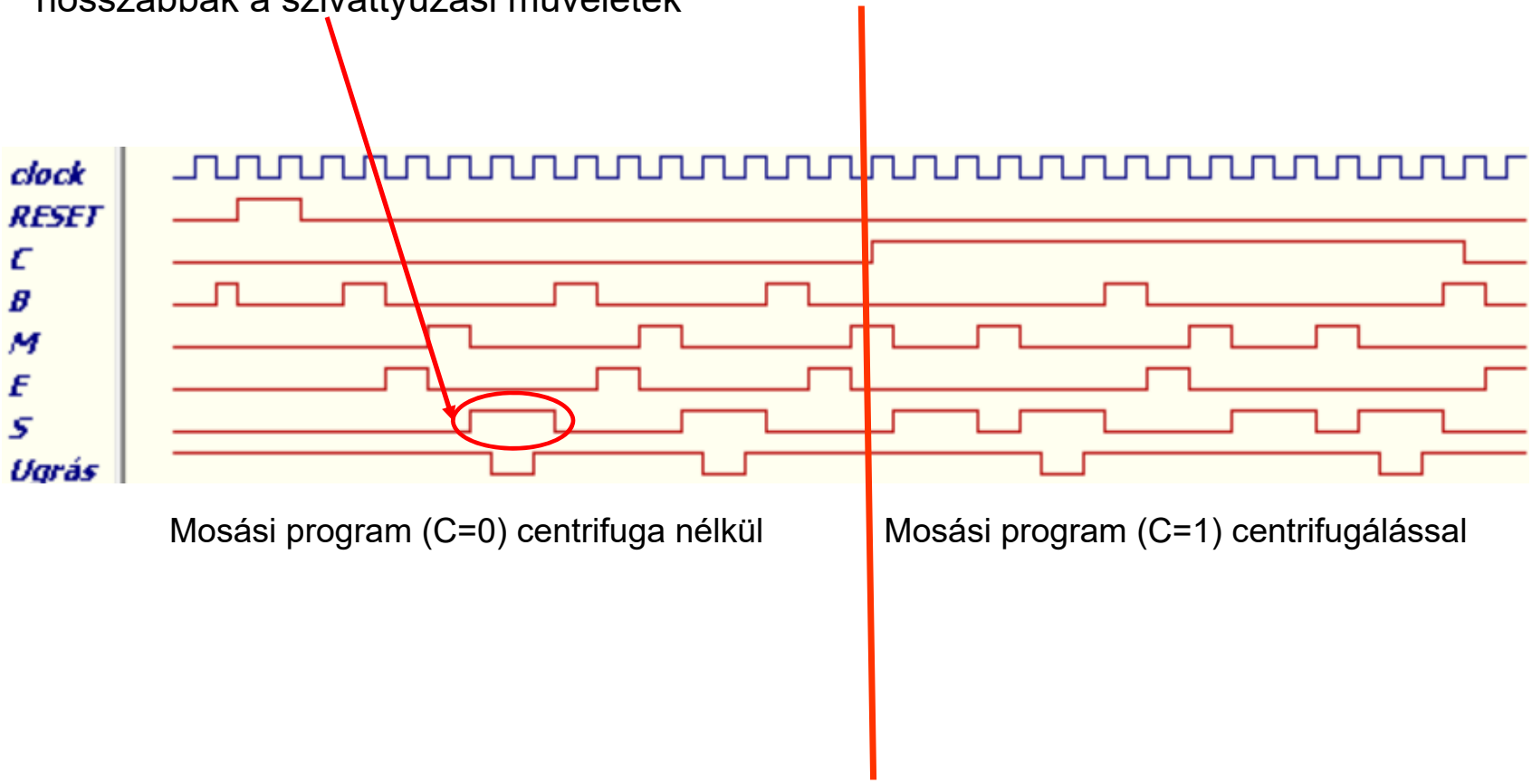


$$\overline{LD} = D_5 + D_4 \cdot C$$

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

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

	$y_3y_2y_1y_0$\C	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 függvények! 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
```

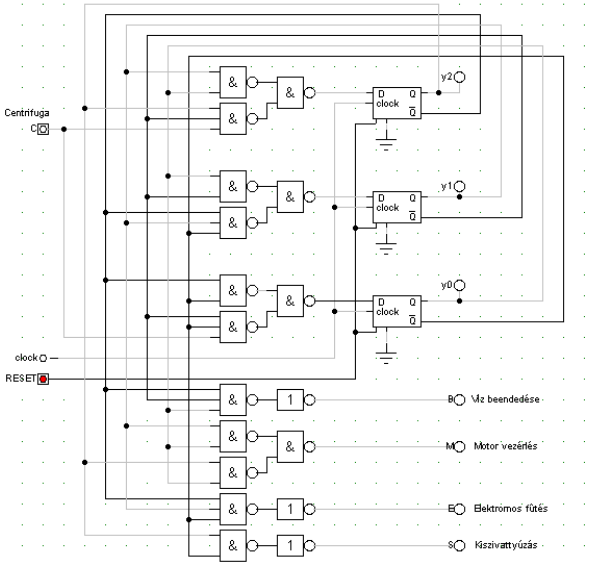
→ Itt csak program módosítás!

→ Minden hardver marad!

Összehasonlítás

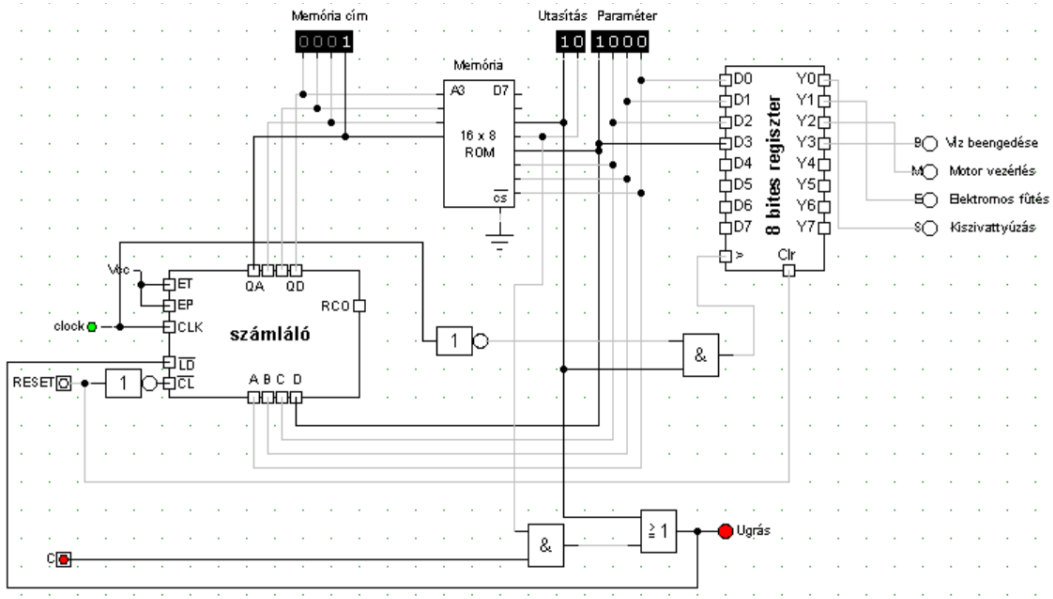
Sorendi hálózat alapú megvalósítás

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



Processzor alapú megvalósítás

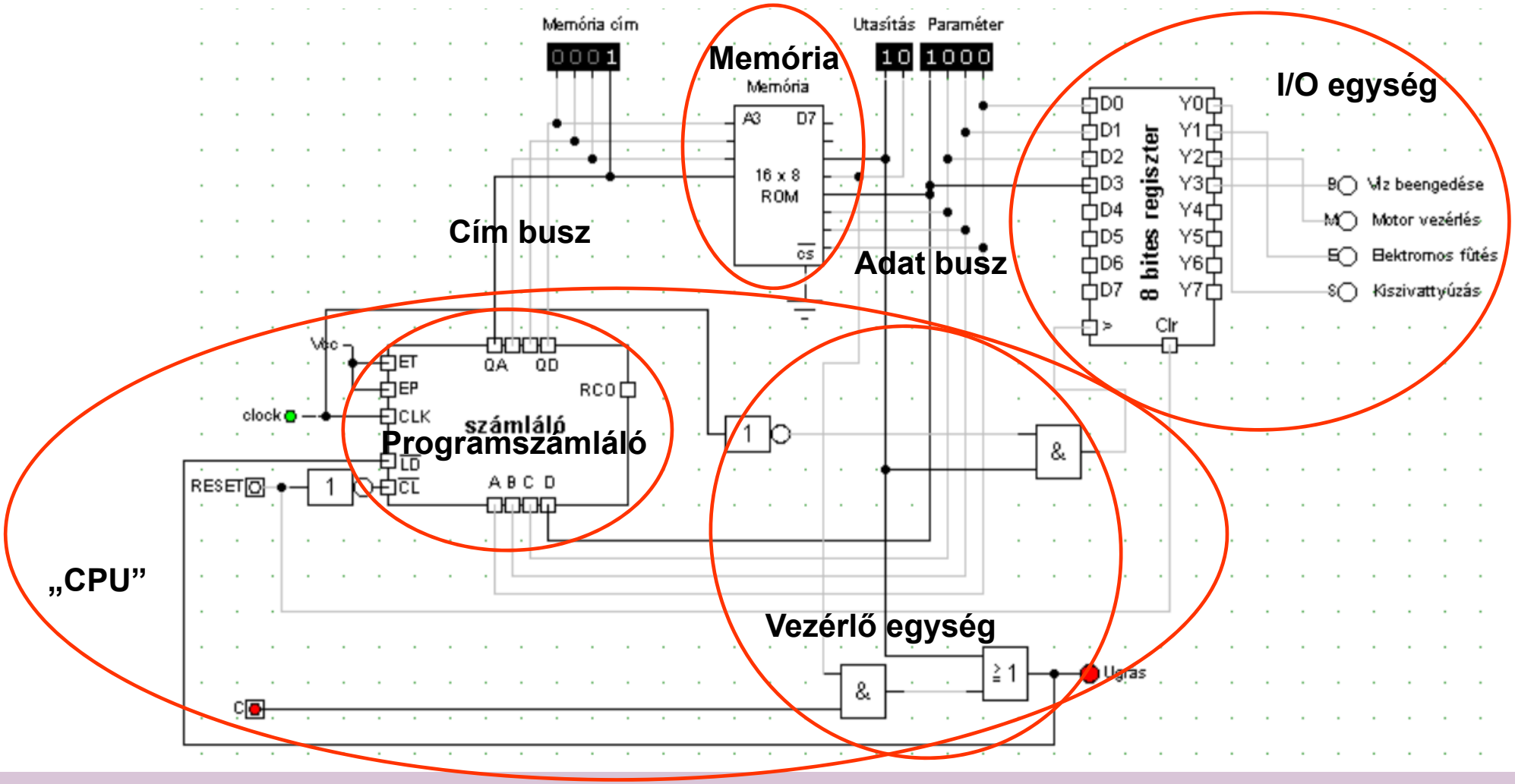
- Lassabb működés
- Bonyolult általános vezérlő
- Program módosítása könnyű
- Rövid tervezési idő (programozás)
- Időzítések programból változtathatók



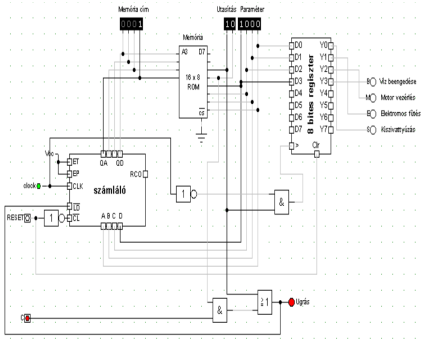
Általános célú processzor vagy célhardver

- 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

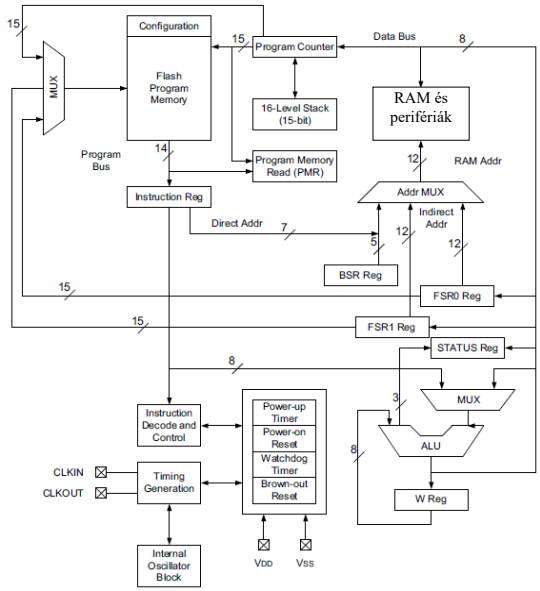


A mikrokontroller amit tanulni fogunk…



- 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

- 49 db különböző utasítás
- 14 bites utasítások
- 15 bites címzés utasításokra
  - → 32768 db utasítás lehet maximum a program
- Többszáz ki/bemeneti regiszter
- 8 bites adatok



Fontosabb fogalmak

- **ALU**
- **Vezérlőegység**
- **Memória**
- **Periféria**

} **CPU**

} **Mikrokontroller**

- **Neumann architektúra**

  azonos memóriatartományban a kód és adat

- **Harvard architektúra**

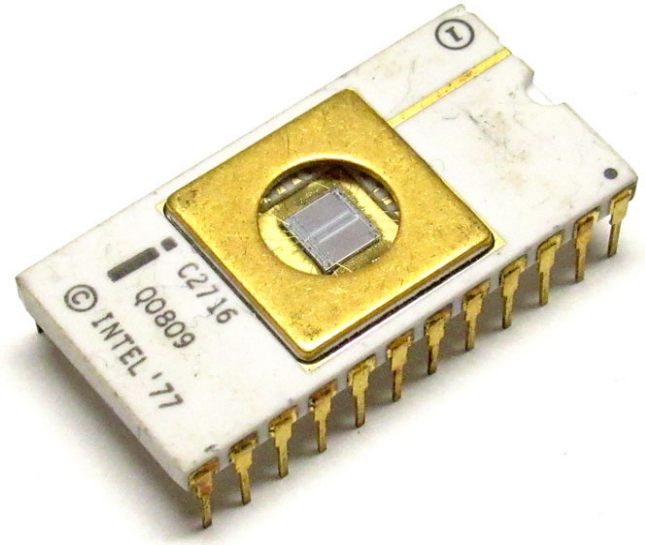
  külön memóriatartományban a kód és adat

(eltérő kezelés, lehet különböző formátum is)

→Módosított Harvard architektúra

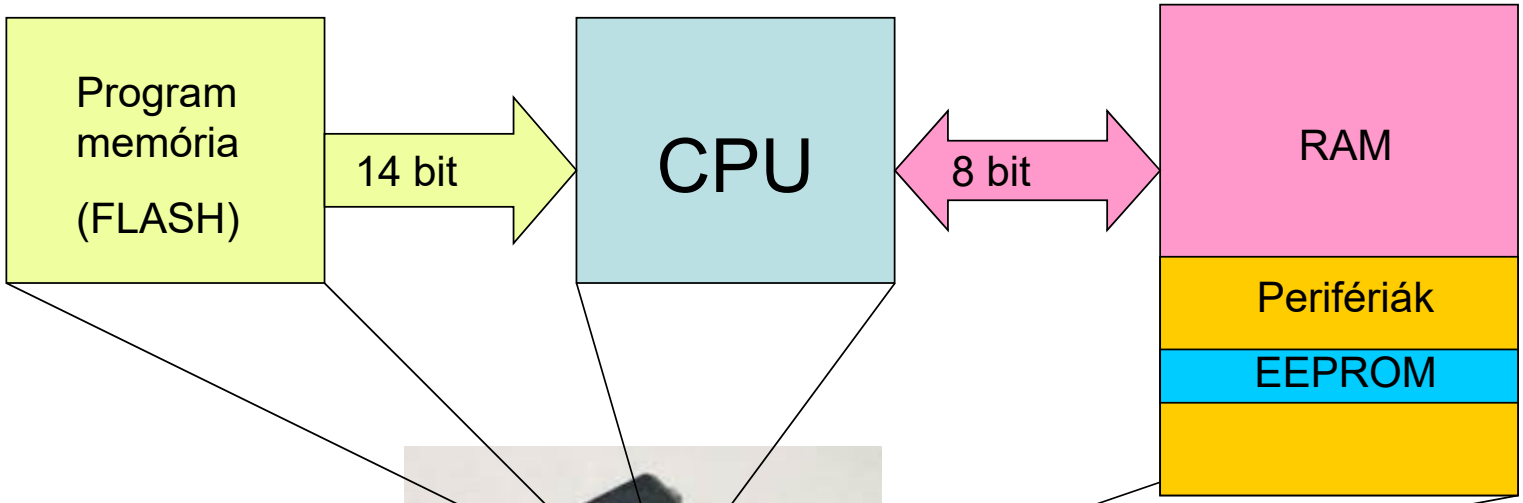
Memória

- Bináris információ tárolására szolgál
- Többféle áramköri megvalósítás és felhasználás (RAM, ROM, EPROM, EEPROM, FLASH, stb.)→később még lesz róluk szó…
- Lineáris címzés
- Belső dekódoló hálózat és tároló rekeszek
- Program tárolása
  →olvasás
- Változók tárolása
  →írás/olvasás



PIC mikrovezérlő felépítése

- Laboron használt típus: PIC16F18875



- Fizikailag:

32kW program

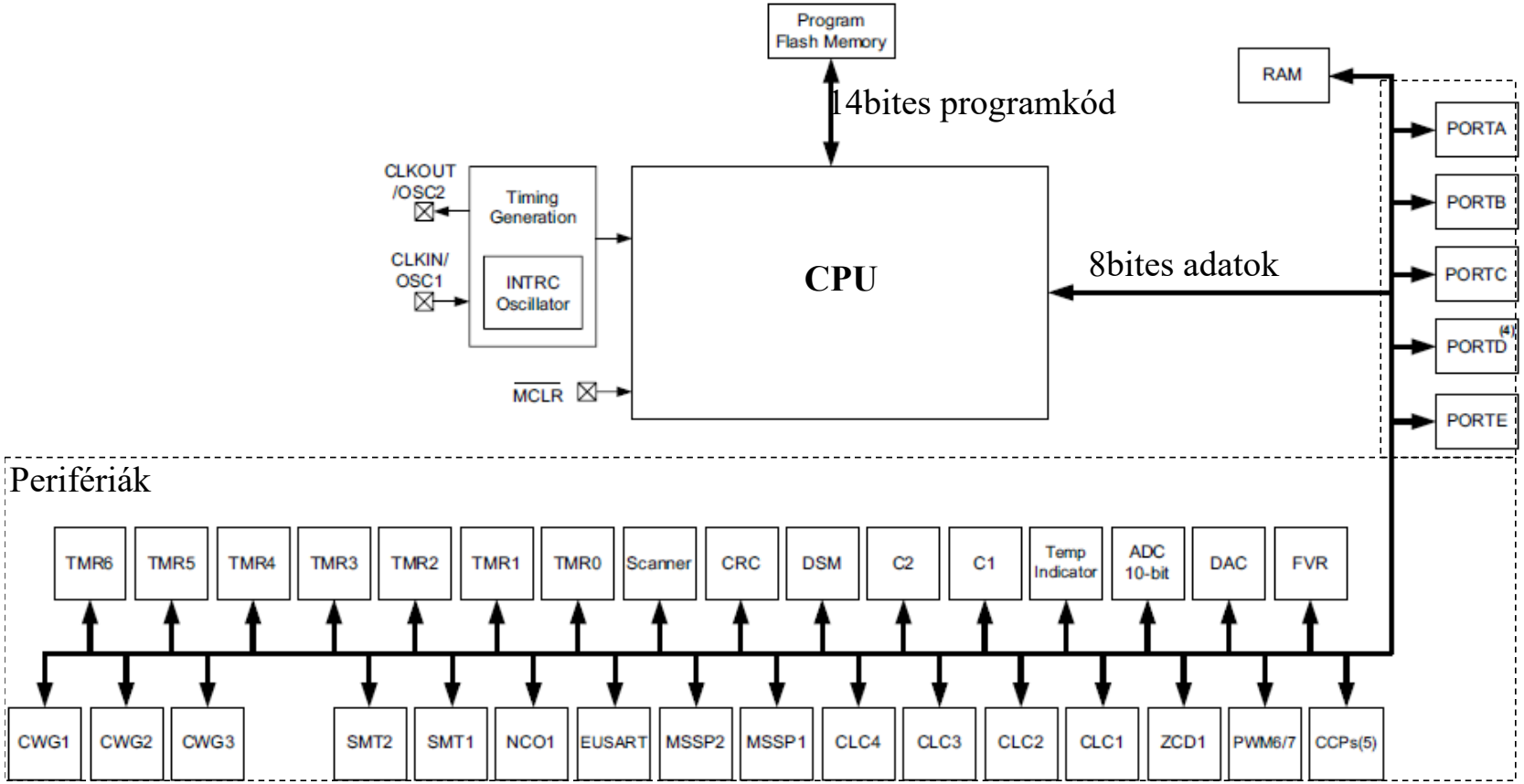
4kB RAM címezhető

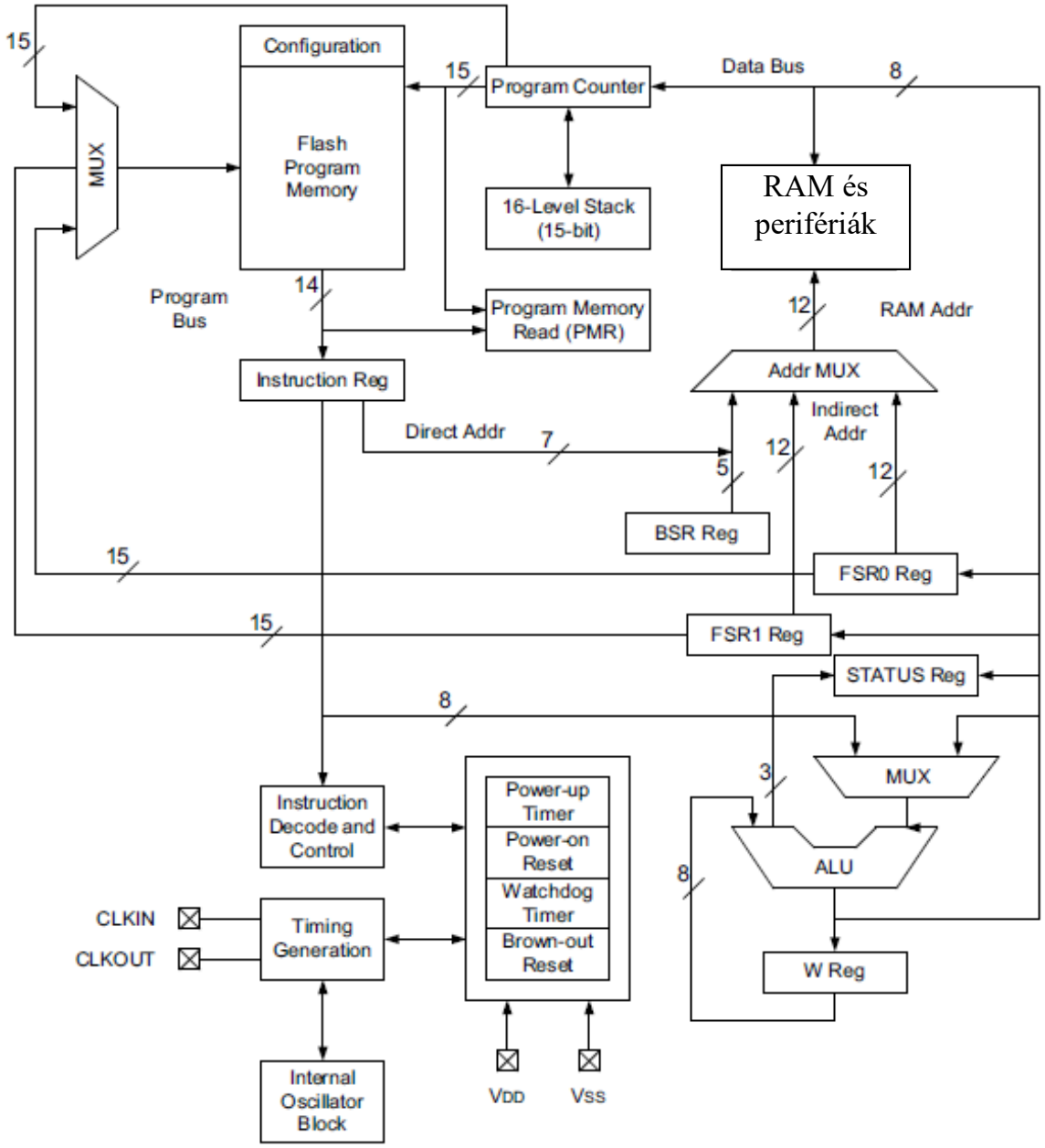
- Gyakorlatilag:

8kW program

1kB RAM van

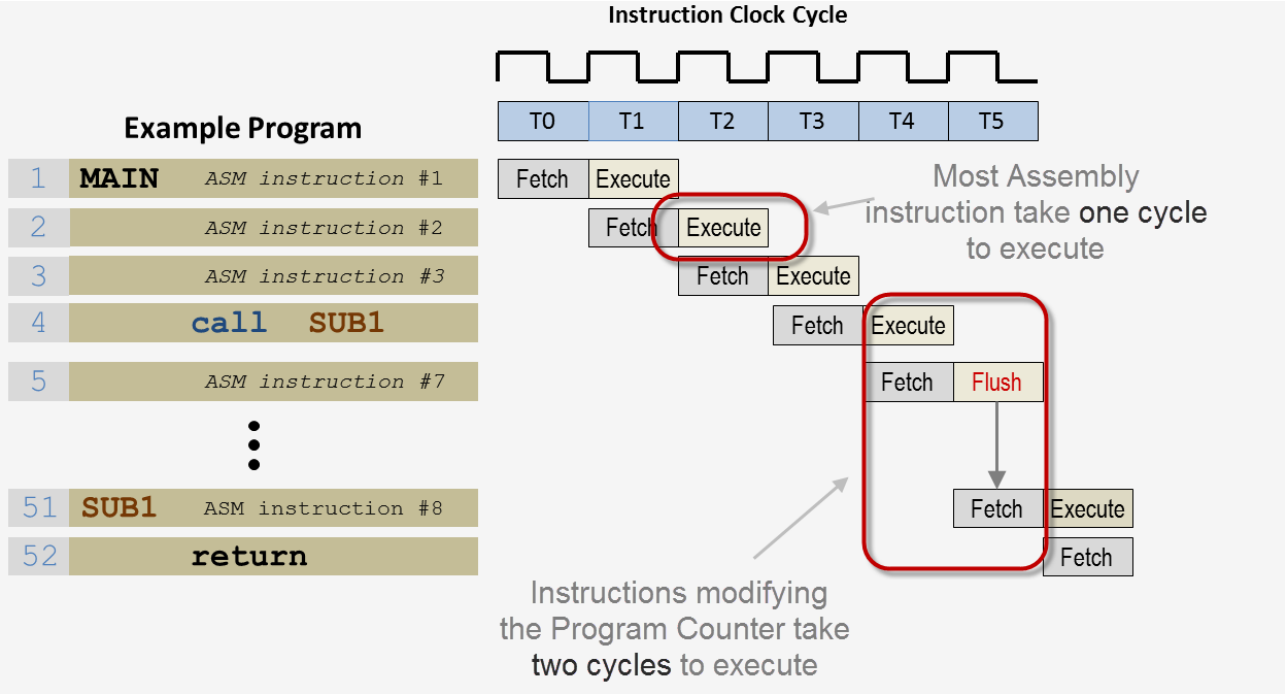
Felépítés részletesebben




Configuration
Flash Program Memory
MUX
15
Program Counter
Data Bus
8
16-Level Stack (15-bit)
RAM és perifériák
Program Bus
14
Program Memory Read (PMR)
12
RAM Addr
Instruction Reg
Addr MUX
Indirect Addr
Direct Addr
7
5
BSR Reg
FSR0 Reg
FSR1 Reg
STATUS Reg
3
MUX
Instruction Decode and Control
Power-up Timer
Power-on Reset
Watchdog Timer
Brown-out Reset
ALU
CLKIN
CLKOUT
Timing Generation
W Reg
Internal Oscillator Block
VDD
VSS

Működés

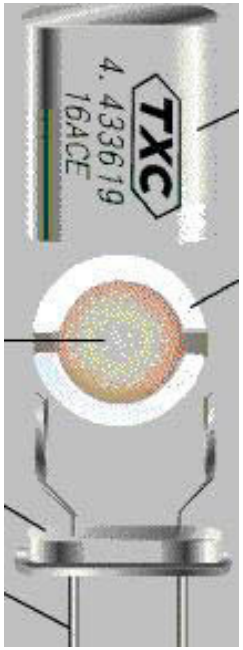
- Szinkron sorrendi hálózat
- Utasítások feldolgozása órajellel ütemezett elemi lépésekből áll



Forrás:http://microchip.wikidot.com/local--files/8bit:emr-architecture/pipeline.png

Fontos kiegészítők

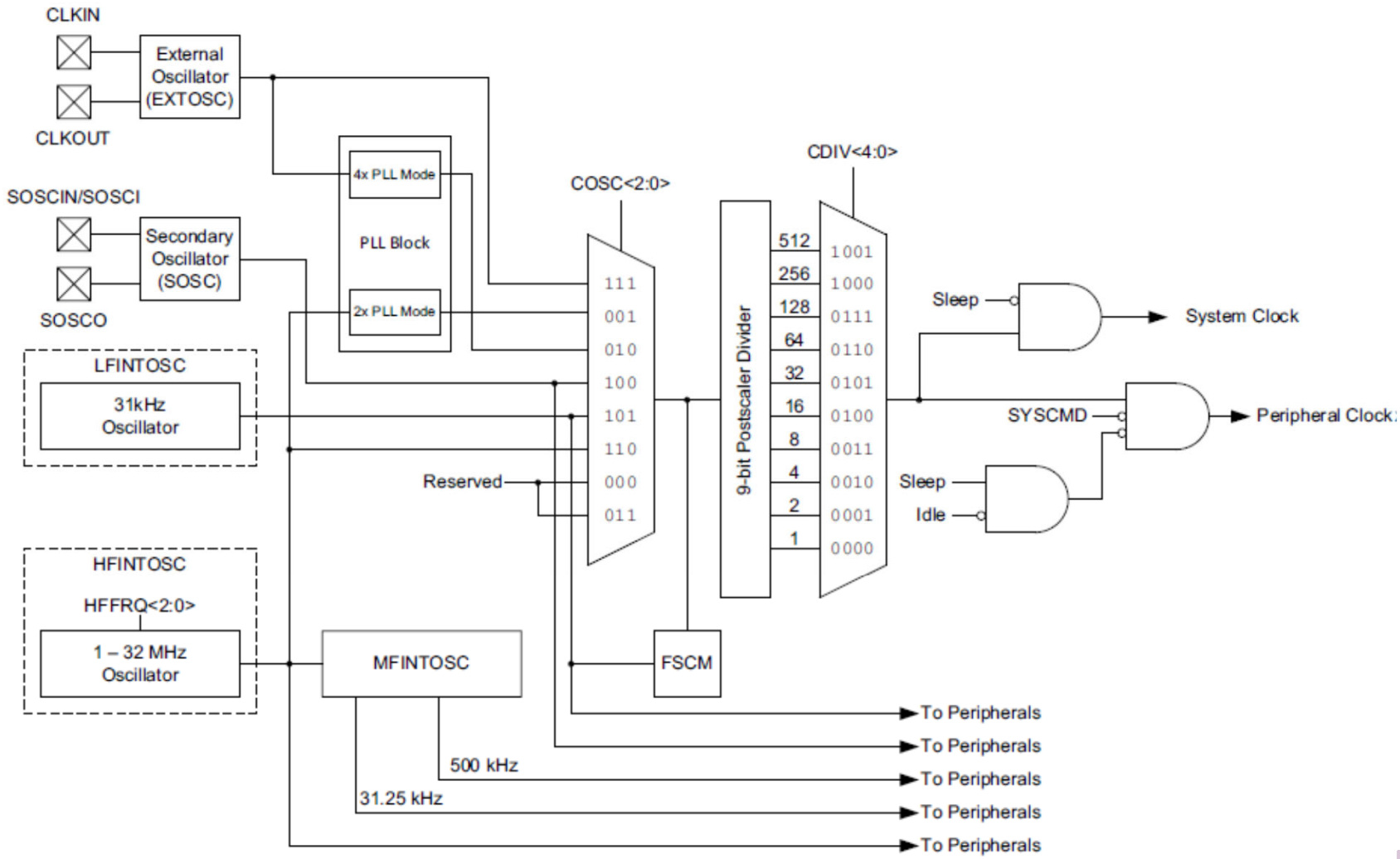
- Órajel előállítás (oszcillátor)
- Alaphelyzet (Reset áramkörök)
  - Külső (MCLR)
  - Belső (POR, PWRT)
  - Tápfeszültség hibára (Brown out)
  - Programhibára (Stack over/underflow)
- Watchdog
- Program betöltés és Debug támogatás
- I/O portok
- Perifériák



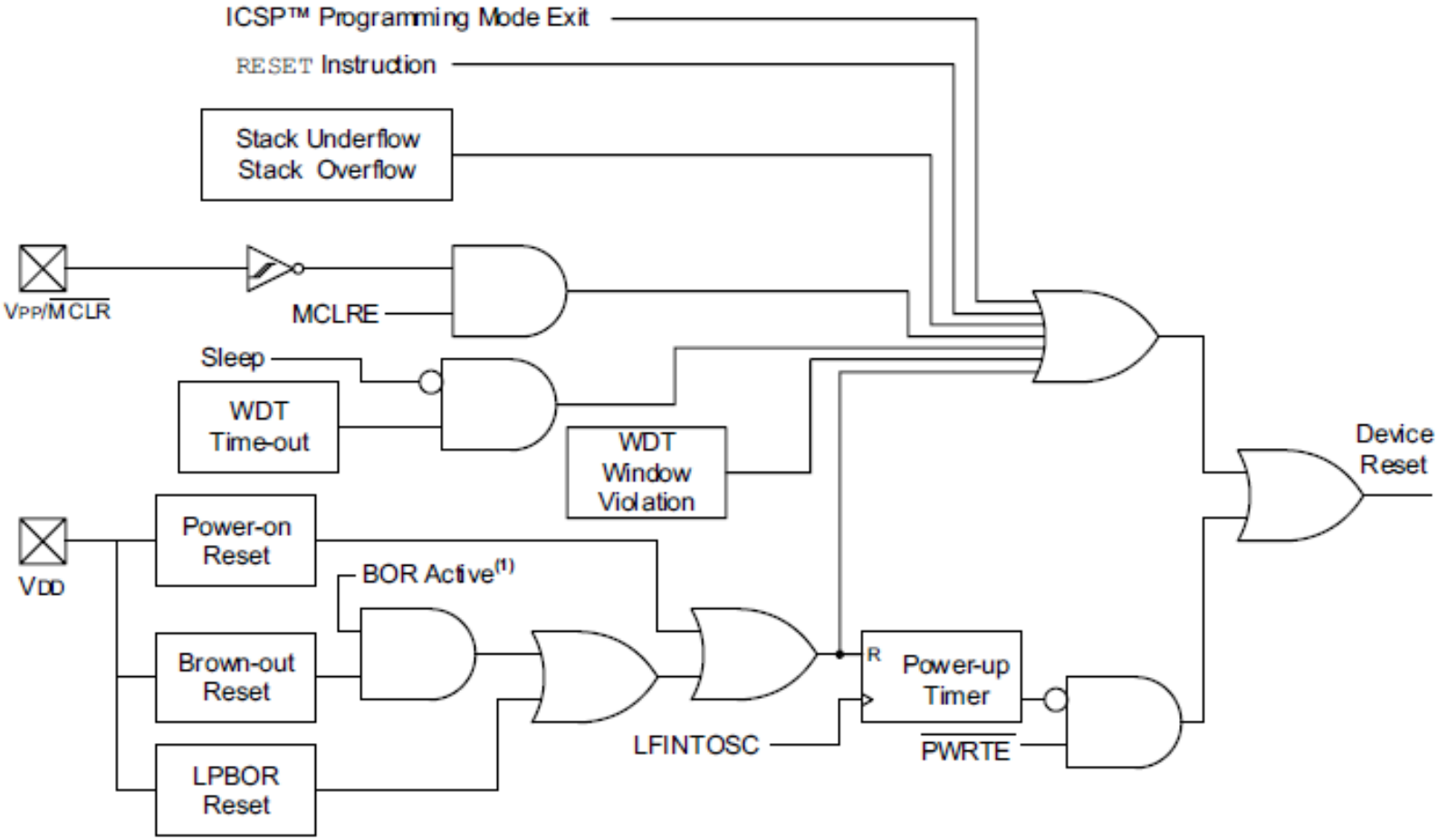
Oszcillátor

- Órajel előállítása adott frekvenciával
- Frekvencia meghatározó tag:
  piezo kristály, vagy RC tag
- Több választási lehetőség (konfiguráció)
- Belső szorzási lehetőség (x4PLL)
- Frekvencia átkapcsolási lehetőséggel
  →energiatakarékosság!

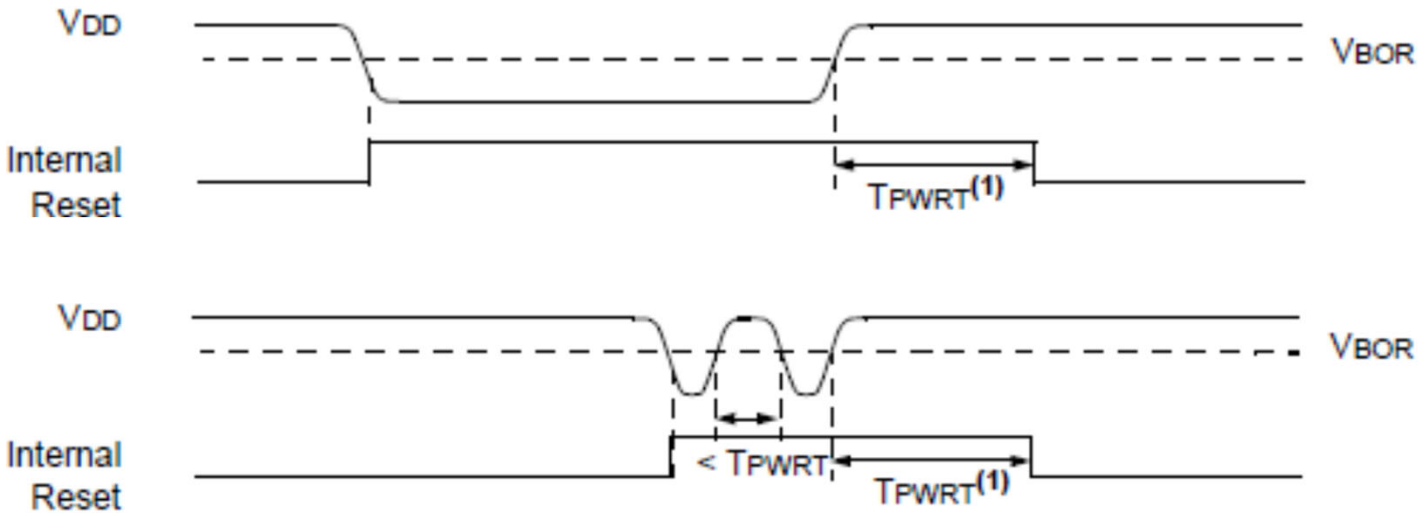
Belső órajelkezelő hálózat



Reset

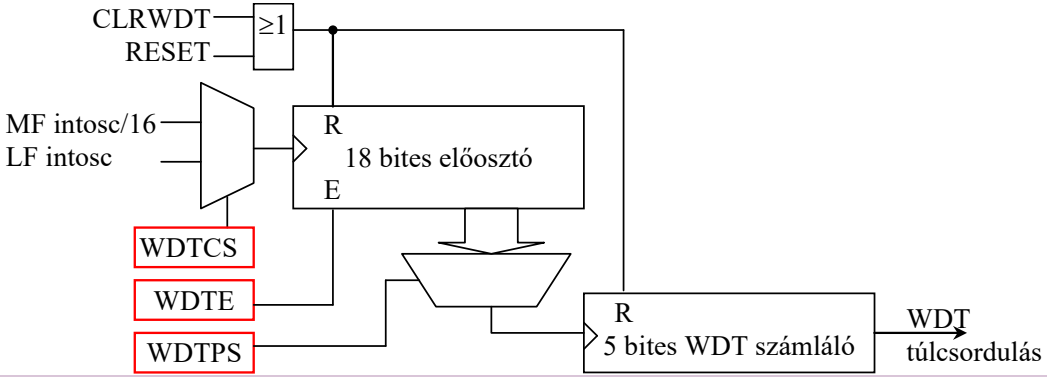

ICSP™ Programming Mode Exit
RESET Instruction
Stack Underflow
Stack Overflow
VPP/MCLR
MCLRE
Sleep
WDT
Time-out
WDT
Window
Violation
Device
Reset
VDD
Power-on
Reset
BOR Active(1)
Brown-out
Reset
LPBOR
Reset
LFINTOSC
R
Power-up
Timer
PWRTE

Belső reset működése



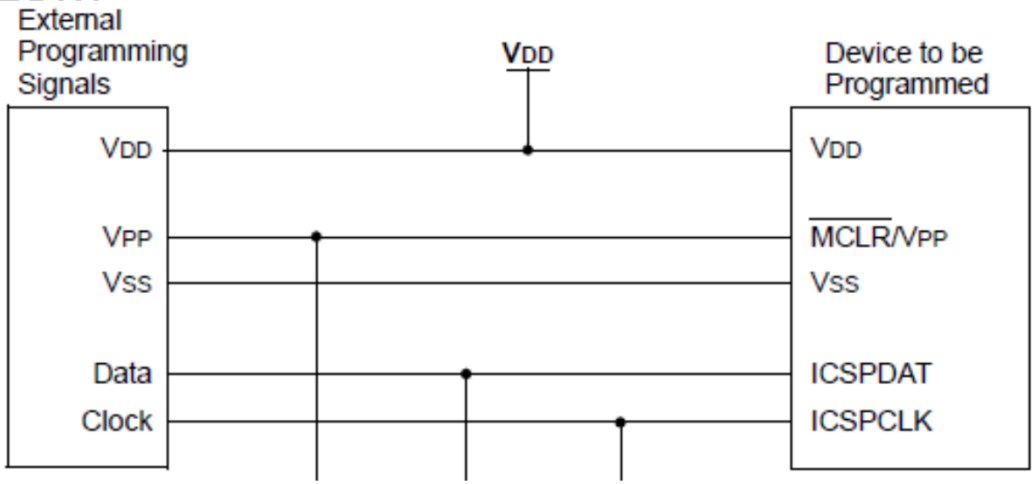
Watchdog

- Folyamatosan működő felfele számláló
- Túlcsorduláskor resetet okoz
  (1ms..256s között állítható)
- Programból ciklikusan törölni kell (CLRWDT)
- Növeli a berendezés megbízhatóságát („lefagyáskor” újraindít)
- Ablak lehetőség a törlési időpont „szigorításához”



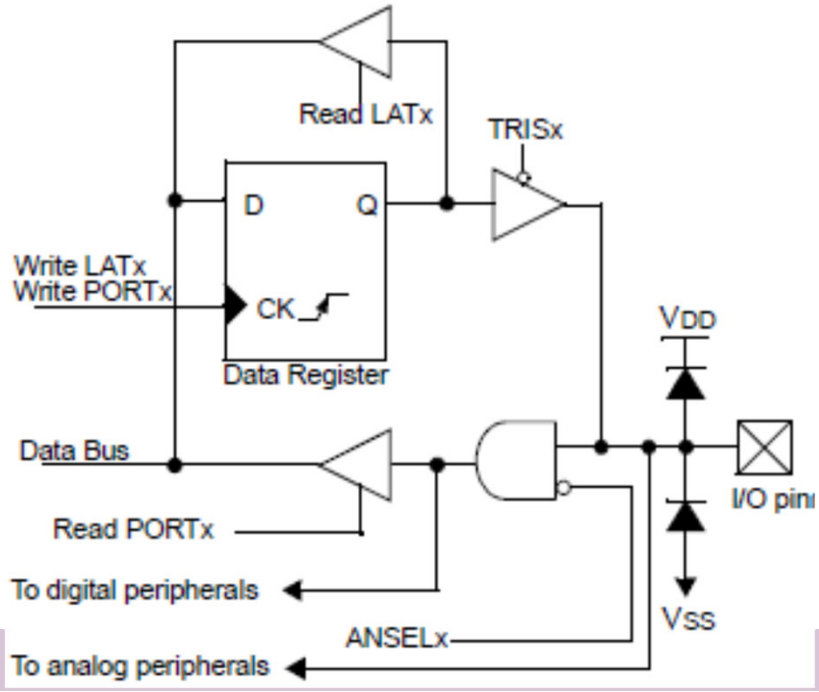
Programozás-Debug

- Áramkörben néhány IC-lábon keresztül lehet a mikrokontrollert programozni
- Ugyanezeken a lábakon lehet megfelelő eszközzel hibakeresést is végezni (debug)
  → később még lesz róla szó…

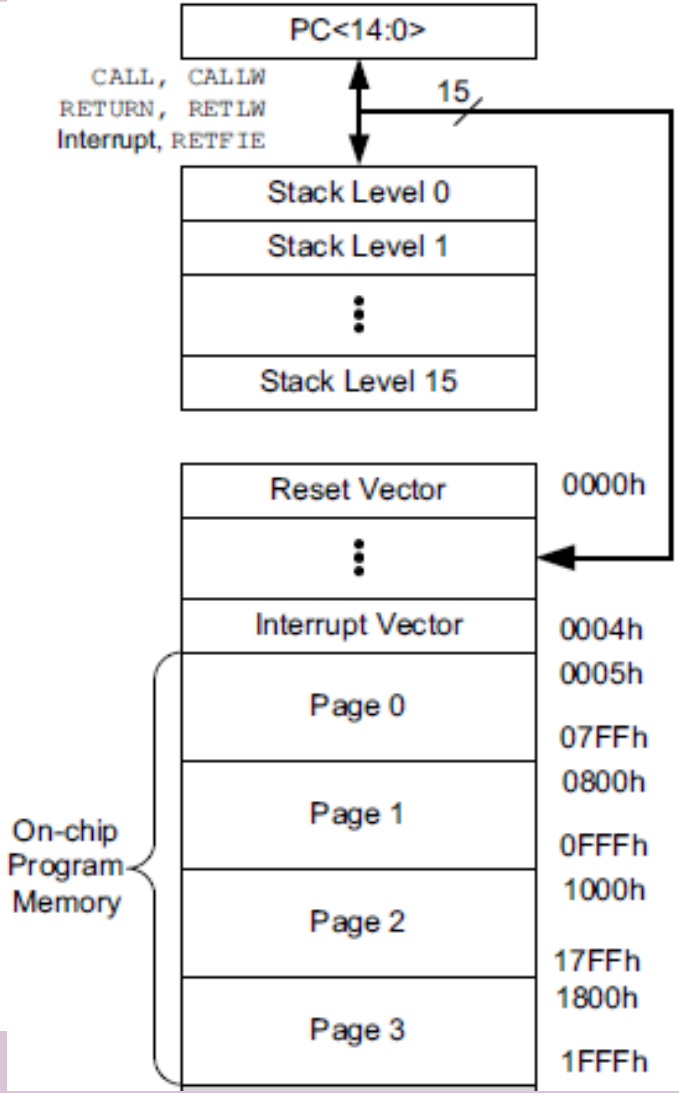


I/O portok

- Kapcsolódás a külvilághoz
- Lehet bemenet, kimenet, analóg üzemmód
- IC lábanként állítható a működési mód (TRISx=0→ kimenet)

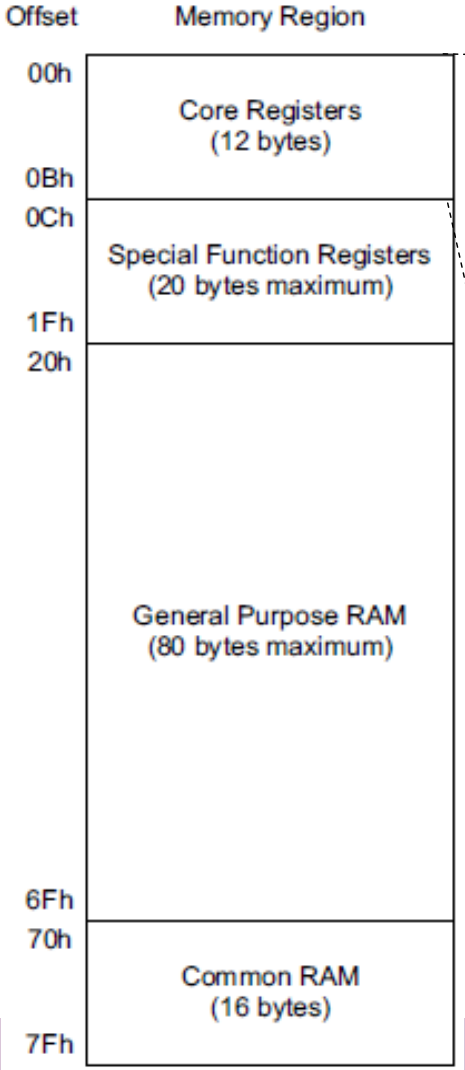


Program memória szervezés



- 8192db 14 bites szó
- FLASH memória
- Programkód tárolása
- Adatok „nehézkesen” tárolhatók benne→módosított Harvard arch.
- 16 mélységű verem a visszatérési címek tárolására→lásd később…

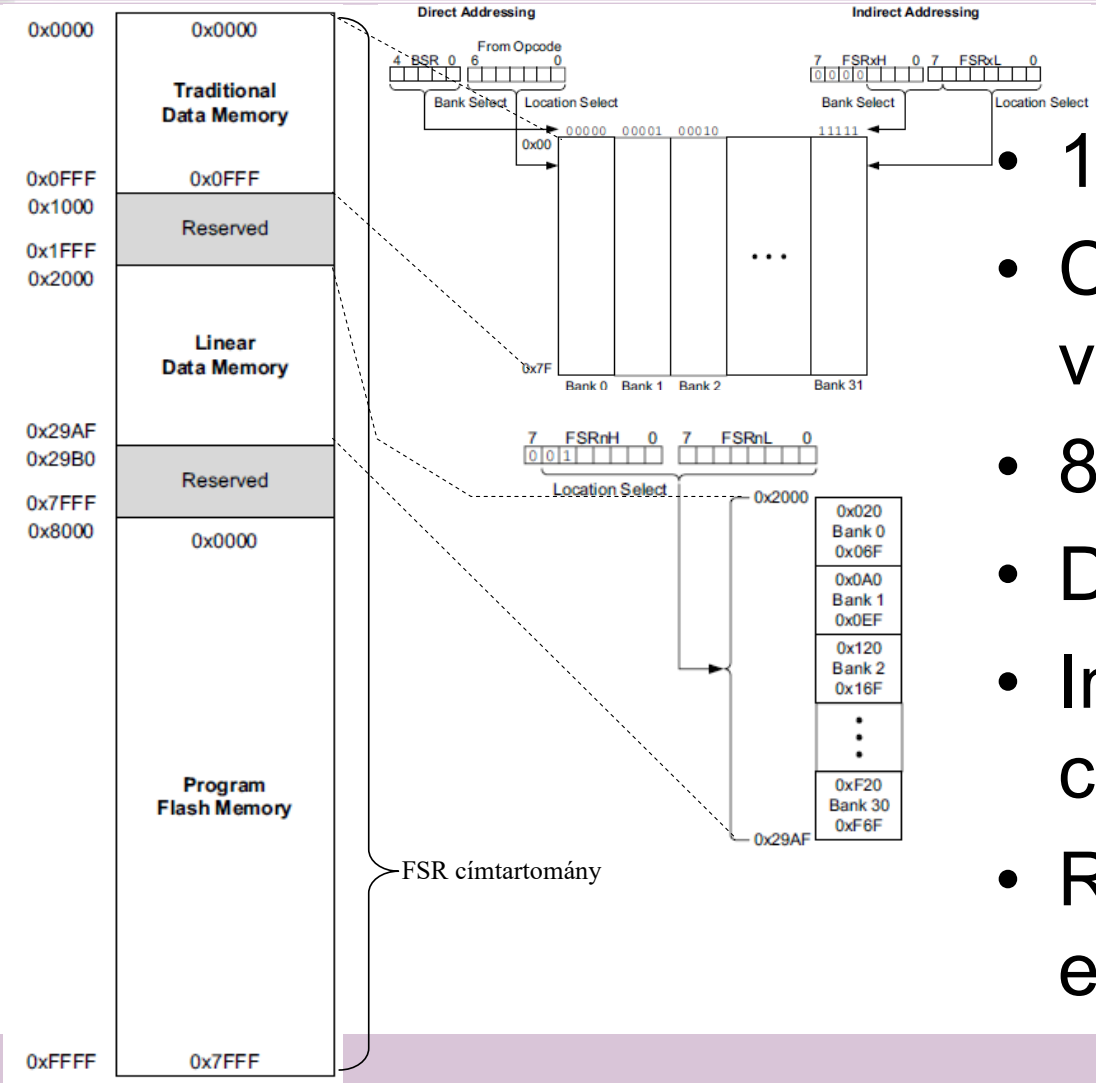
RAM/adat memóriatartomány



Addresses	BANKx
x00h or x80h	INDF0
x01h or x81h	INDF1
x02h or x82h	PCL
x03h or x83h	STATUS
x04h or x84h	FSR0L
x05h or x85h	FSR0H
x06h or x86h	FSR1L
x07h or x87h	FSR1H
x08h or x88h	BSR
x09h or x89h	WREG
x0Ah or x8Ah	PCLATH
x0Bh or x8Bh	INTCON

- Vegyesen található benne RAM és periféria vezérlő regiszter (SFR)
- BANK szervezés, összesen 32
- Vannak közös területek (CORE és COMMON)

Adat memória címzése



- 12bites cím
- Címezhetünk 7, vagy 12 bittel.
- 8 bites adatelérés
- Direkt (7bites) és
- Indirekt (12 bites) címzés
- RAM és FLASH is elérhető

Fontosabb regiszterek

	BANK 0		BANK 1		BANK 2		BANK 3
000h	Core Registers (Table 3-2)	080h	Core Registers (Table 3-2)	100h	Core Registers (Table 3-2)	180h	Core Registers (Table 3-2)
00Bh		08Bh		10Bh		18Bh	
00Ch	PORTA	08Ch	ADRESL	10Ch	ADCNT	18Ch	SSP1BUF
00Dh	PORTB	08Dh	ADRESH	10Dh	ADRPT	18Dh	SSP1ADD
00Eh	PORTC	08Eh	ADPREVL	10Eh	ADLTHL	18Eh	SSP1MSK
00Fh	PORTD	08Fh	ADPREVH	10Fh	ADLTHH	18Fh	SSP1STAT
010h	PORTE	090h	ADACCL	110h	ADUTHL	190h	SSP1CON1
011h	TRISA	091h	ADACCH	111h	ADUTHH	191h	SSP1CON2
012h	TRISB	092h	—	112h	ADSTPTL	192h	SSP1CON3
013h	TRISC	093h	ADCON0	113h	ADSTPTH	193h	—
014h	TRISD	094h	ADCON1	114h	ADFLTRL	194h	—
015h	TRISE	095h	ADCON2	115h	ADFLTRH	195h	—
016h	LATA	096h	ADCON3	116h	ADERRL	196h	SSP2BUF
017h	LATB	097h	ADSTAT	117h	ADERRH	197h	SSP2ADD
018h	LATC	098h	ADCLK	118h	—	198h	SSP2MSK
019h	LATD	099h	ADACT	119h	RC1REG	199h	SSP2STAT
01Ah	LATE	09Ah	ADREF	11Ah	TX1REG	19Ah	SSP2CON1
01Bh	—	09Bh	ADCAP	11Bh	SP1BRGL	19Bh	SSP2CON2
01Ch	TMR0L	09Ch	ADPRE	11Ch	SP1BRGH	19Ch	SSP2CON3
01Dh	TMR0H	09Dh	ADACQ	11Dh	RC1STA	19Dh	—
01Eh	T0CON0	09Eh	ADPCH	11Eh	TX1STA	19Eh	—
01Fh	T0CON1	09Fh	—	11Fh	BAUD1CON	19Fh	—
020h	General Purpose Register 96 Bytes	0A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h	General Purpose Register 80 Bytes
		0EFh		16Fh		1EFh	
		0F0h	Common RAM (Accesses 70h – 7Fh)	170h	Common RAM (Accesses 70h – 7Fh)	1F0h	Common RAM (Accesses 70h – 7Fh)
07Fh		0FFh		17Fh		1FFh	

INDF0
INDF1
PCL
STATUS
FSR0L
FSR0H
FSR1L
FSR1H
BSR
WREG
PCLATH
INTCON

Gépi utasítások

- A mikrokontroller által végrehajtható műveletek
- Bináris formában vannak tárolva 14 biten

  →gépi kód
- A gépi kód egy része a műveletet határozza meg, másik része tartalmazza a paramétereket (operandus)
- Az egyes bináris kódokhoz megjegyezhető neveket (mnemonik) rendelünk→assembly programozási nyelv

Gépi utasítások

- Akkumulátor (WREG) központú operandus kezelés

  Két regiszter majdnem minden utasítás végrehajtásában részt vesz:

  W (vagy WREG): akkumulátor (egyik operandus)

  STATUS: jelzőbitek (Z(ero), C(arry), AC)
- Utasítás csoportok:
  - Adatmozgató utasítások
  - Aritmetikai műveletek
  - Logikai műveletek
  - Bitműveletek
  - Vezérlés átadás
  - Vezérlő műveletek

Gépi kód felépítése

Byte-oriented file register operations

13 … 8	7	6 … 0
OPCODE	d	f (FILE #)

d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

Bit-oriented file register operations

13 … 10	9 … 7	6 … 0
OPCODE	b (BIT #)	f (FILE #)

b = 3-bit bit address
f = 7-bit file register address

Literal and control operations

General

13 … 8	7 … 0
OPCODE	k (literal)

k = 8-bit immediate value

CALL and GOTO instructions only

13 … 11	10 … 0
OPCODE	k (literal)

k = 11-bit immediate value

MOVLP instruction only

13 … 7	6 … 0
OPCODE	k (literal)

k = 7-bit immediate value

MOVLB instruction only

13 … 5	4 … 0
OPCODE	k (literal)

k = 5-bit immediate value

BRA instruction only

13 … 9	8 … 0
OPCODE	k (literal)

k = 9-bit immediate value

FSR Offset instructions

13 … 7	6	5 … 0
OPCODE	n	k (literal)

n = appropriate FSR
k = 6-bit immediate value

FSR Increment instructions

13 … 3	2	1 … 0
OPCODE	n	m (mode)

n = appropriate FSR
m = 2-bit mode value

OPCODE only

13 … 0
OPCODE

Rövidítések

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f. Default is d = 1 .
n	FSR or INDF number. (0-1)
mm	Pre-post increment-decrement mode selection

Utasításkészlet - 1

TABLE 34-3: ENHANCED MID-RANGE INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C, DC, Z	2
ADDWFC	f, d	Add with Carry W and f	1	11	1101	dfff	ffff	C, DC, Z	2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	2
ASRF	f, d	Arithmetic Right Shift	1	11	0111	dfff	ffff	C, Z	2
LSLF	f, d	Logical Left Shift	1	11	0101	dfff	ffff	C, Z	2
LSRF	f, d	Logical Right Shift	1	11	0110	dfff	ffff	C, Z	2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRW	–	Clear W	1	00	0001	0000	00xx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	2
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	2
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		2
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C, DC, Z	2
SUBWFB	f, d	Subtract with Borrow W from f	1	11	1011	dfff	ffff	C, DC, Z	2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	2
BYTE ORIENTED SKIP OPERATIONS									
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1, 2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1, 2

Utasításkészlet - 2

TABLE 34-3: ENHANCED MID-RANGE INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C, DC, Z	2
ADDWFC	f, d	Add with Carry W and f	1	11	1101	dfff	ffff	C, DC, Z	2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	2
ASRF	f, d	Arithmetic Right Shift	1	11	0111	dfff	ffff	C, Z	2
LSLF	f, d	Logical Left Shift	1	11	0101	dfff	ffff	C, Z	2
LSRF	f, d	Logical Right Shift	1	11	0110	dfff	ffff	C, Z	2
CLRF	f	Clear f	1	00	0001	1fff	ffff	Z	2
CLRW	—	Clear W	1	00	0001	0000	00xx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	2
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	2
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	2
MOVWF	f	Move W to f	1	00	0000	1fff	ffff		2
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C, DC, Z	2
SUBWFB	f, d	Subtract with Borrow W from f	1	11	1011	dfff	ffff	C, DC, Z	2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	2
BYTE ORIENTED SKIP OPERATIONS									
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00	1011	dfff	ffff		1, 2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00	1111	dfff	ffff		1, 2
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b	Bit Clear f	1	01	00bb	bfff	ffff		2
BSF	f, b	Bit Set f	1	01	01bb	bfff	ffff		2
BIT-ORIENTED SKIP OPERATIONS									
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01	10bb	bfff	ffff		1, 2
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01	11bb	bfff	ffff		1, 2
LITERAL OPERATIONS									
ADDLW	k	Add literal and W	1	11	1110	kkkk	kkkk	C, DC, Z	
ANDLW	k	AND literal with W	1	11	1001	kkkk	kkkk	Z	
IORLW	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z	
MOVLB	k	Move literal to BSR	1	00	0000	001k	kkkk		
MOVLP	k	Move literal to PCLATH	1	11	0001	1kkk	kkkk		
MOVLW	k	Move literal to W	1	11	0000	kkkk	kkkk		
SUBLW	k	Subtract W from literal	1	11	1100	kkkk	kkkk	C, DC, Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z	

Note 1: If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

2: If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

Utasításkészlet - 3

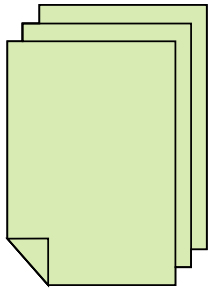
CONTROL OPERATIONS									
BRA	k	Relative Branch	2	11	001k	kkkk	kkkk		
BRW	–	Relative Branch with W	2	00	0000	0000	1011		
CALL	k	Call Subroutine	2	10	0kkk	kkkk	kkkk		
CALLW	–	Call Subroutine with W	2	00	0000	0000	1010		
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk		
RETFIE	k	Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	0100	kkkk	kkkk		
RETURN	–	Return from Subroutine	2	00	0000	0000	1000		
INHERENT OPERATIONS									
CLRWDT	–	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$, $\overline{PD}$	
NOP	–	No Operation	1	00	0000	0000	0000		
OPTION	–	Load OPTION_REG register with W	1	00	0000	0110	0010		
RESET	–	Software device Reset	1	00	0000	0000	0001		
SLEEP	–	Go into Standby mode	1	00	0000	0110	0011	$\overline{TO}$, $\overline{PD}$	
TRIS	f	Load TRIS register with W	1	00	0000	0110	0fff		
C-COMPILER OPTIMIZED									
ADDFSR	n, k	Add Literal k to FSRn	1	11	0001	0nkk	kkkk		
MOVIW	n mm	Move Indirect FSRn to W with pre/post inc/dec modifier, mm	1	00	0000	0001	0nmm kkkk	Z	2, 3
	k[n]	Move INDFn to W, Indexed Indirect.	1	11	1111	0nkk	1nmm	Z	2
MOVWI	n mm	Move W to Indirect FSRn with pre/post inc/dec modifier, mm	1	00	0000	0001	kkkk		2, 3
	k[n]	Move W to INDFn, Indexed Indirect.	1	11	1111	1nkk			2

Note 1: If the Program Counter (PC) is modified, or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

2: If this instruction addresses an INDF register and the MSb of the corresponding FSR is set, this instruction will require one additional instruction cycle.

3: See Table in the MOVIW and MOVWI instruction descriptions.

A témához kapcsolódó anyagok



- Microchip weboldala

  http://www.microchip.com
- MPLABX fejlesztői környezet és dokumentáció letöltése

  https://www.microchip.com/mplab/mplab-x-ide
- Online help a mikrokontrollerhez és a fejlesztői környezethez

  http://microchipdeveloper.com/mcu1102:start