Mikrokontrollerek

Tihanyi Attila

2007. május 8

!!! ZH !!!

- Pótlási lehetőség külön egyeztetve
- Feladatok:
  - Első ZH is itt pótolható
  - Munkapont számítás
  - Mikrokontroller program írása

Bitek kezelése

- Bit (adat bitek; flag bitek)
  - Tetszőleges regiszter adatbitje
    - nulladik . dbit 0,R00
    - elso . dbit 1,R00
    - masodik . dbit 2,R00
    - …
    - Ezek állítását a programozó végzi
  - Carry, Zero, Negatív, Overflow, Interrupt Enable (1,2)
    - Ezek állítását a műveletek végzik

Bitműveletek

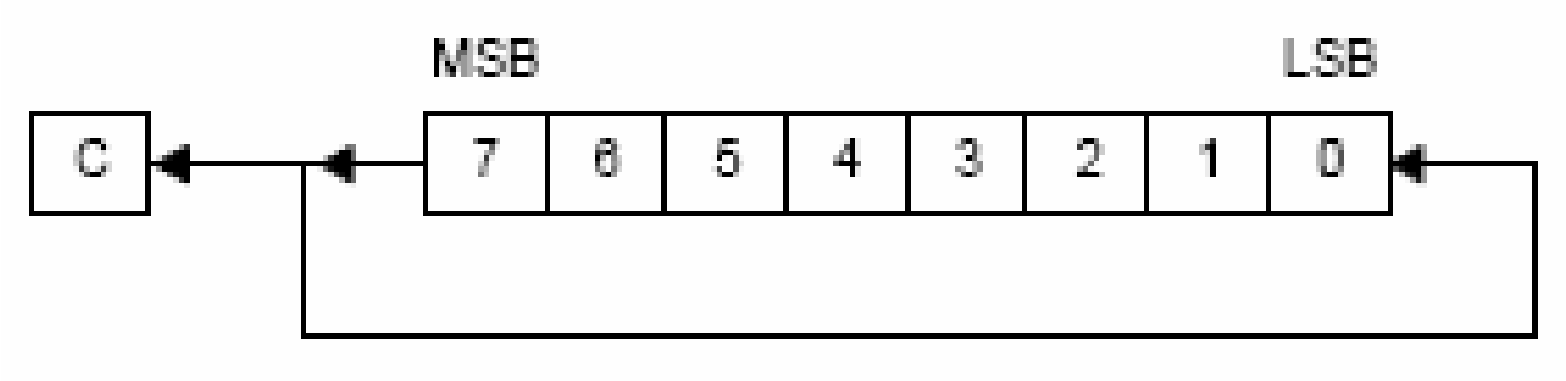
- SBIT0 name ; 0 → name
- SBIT1 name ; 1 → name

- JBIT0 name,címke8
- JBIT1 name,címke8
- BTJO s1,s2,címke8 ;s1 & s2
- BTJZ s1,s2,címke8 ;s1 & s2

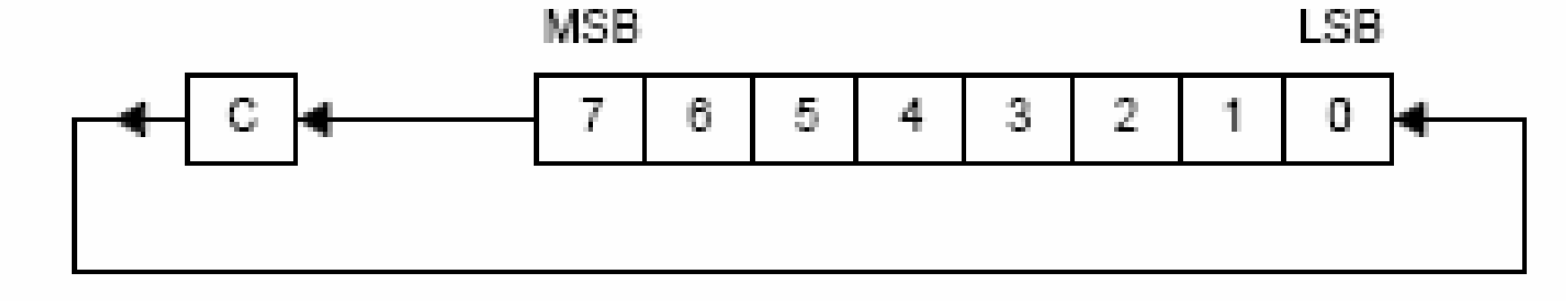
- SETC ; 1 → Cy
- CLRC ; 0 → Cy

Rotate

- RL Rd ;rotate left

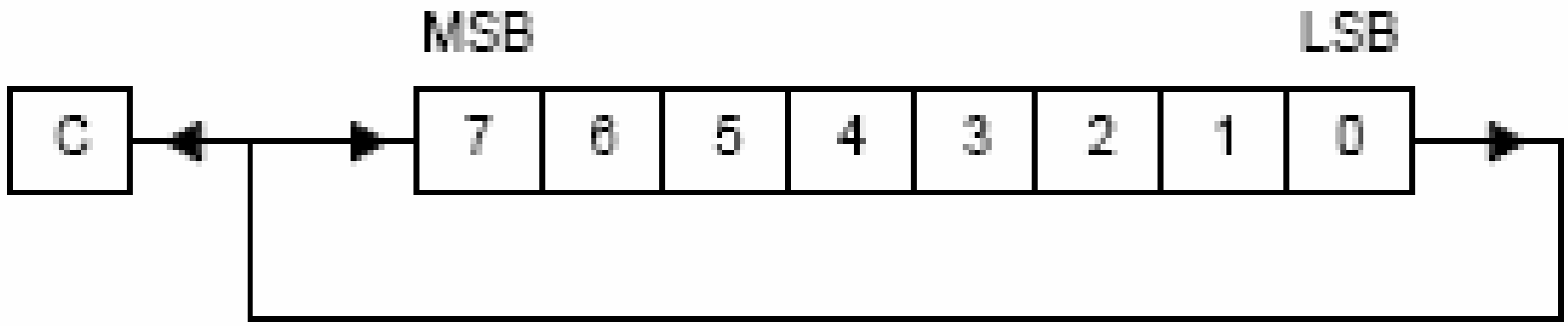


- RLC Rd ; Rotate Left Through Carry

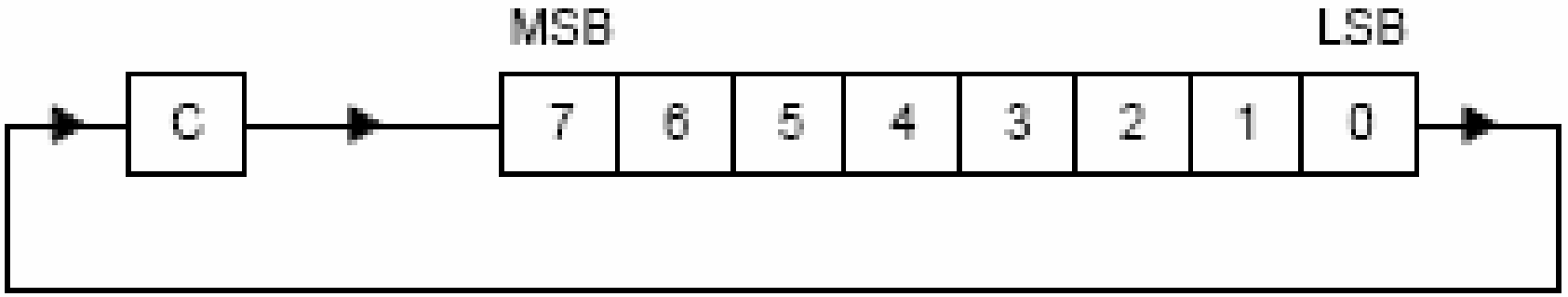


Rotate

- RR Rd ;rotate left



- RRC Rd ; Rotate Left Through Carry



Flag bitek

- **Bit 7 C**. Carry.
  - This status bit is set by arithmetic instructions as a carry bit or as a no-borrow bit.
  - It is also affected by the rotate instructions.
- **Bit 6 N**. Negative.
  - The CPU sets this bit to the value of the most significant bit (sign bit) of the result of the previous operation.
- **Bit 5 Z**. Zero.
  - This bit is set by the CPU if the result of the previous operation was 0; cleared otherwise.
- **Bit 4 V**. Overflow.
  - This bit is set by the CPU if a signed arithmetic overflow condition is detected during the previous instruction. The value of this flag is significant at the completion of the following instructions: ADC, ADD, INC, INCW, CMP, DEC,SUB, SBB, and DIV.

Összeadás

- signed int i;
- signed int j;

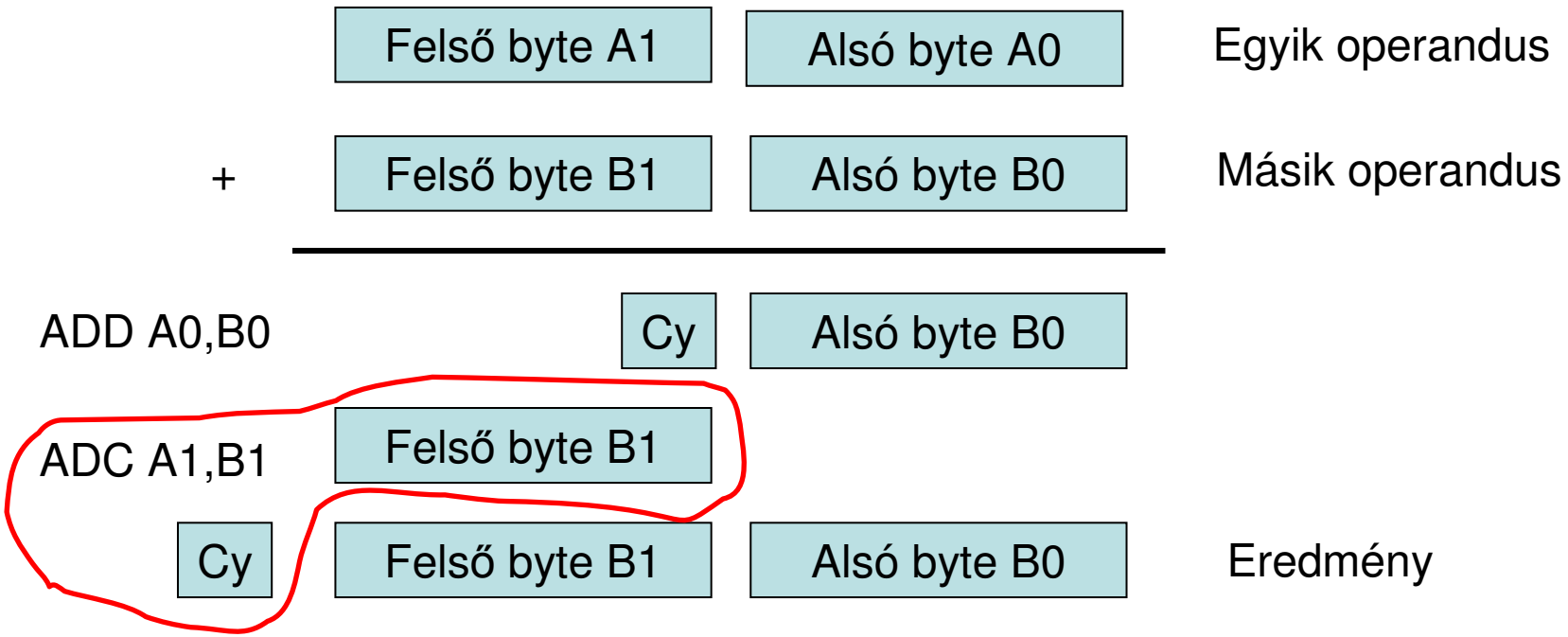
- j = j + i; vagy j += i;

- signed long i;
- signed long j;

- j = j + i; vagy j += i;

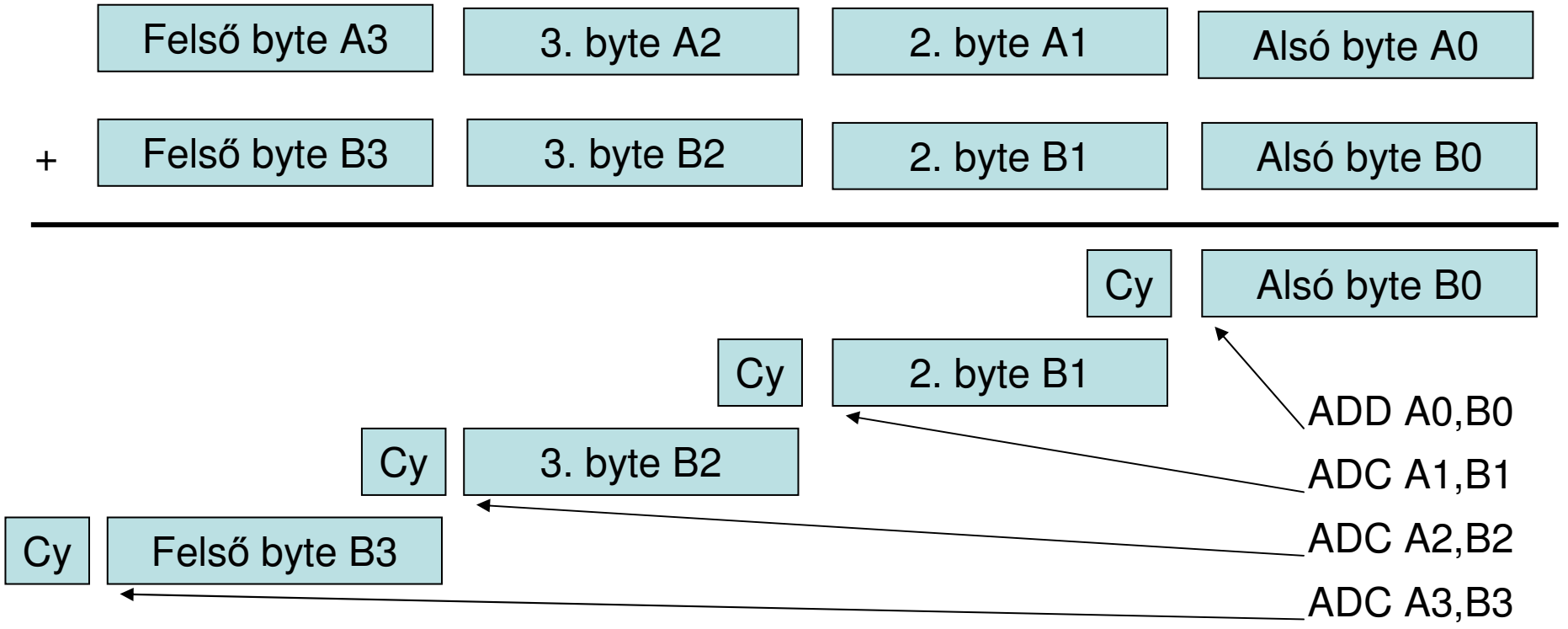
Példák

- 2 byte-os összeadás



Példák

- 4 byte-os összeadás



Kivonás

- signed int i;
- signed int j;

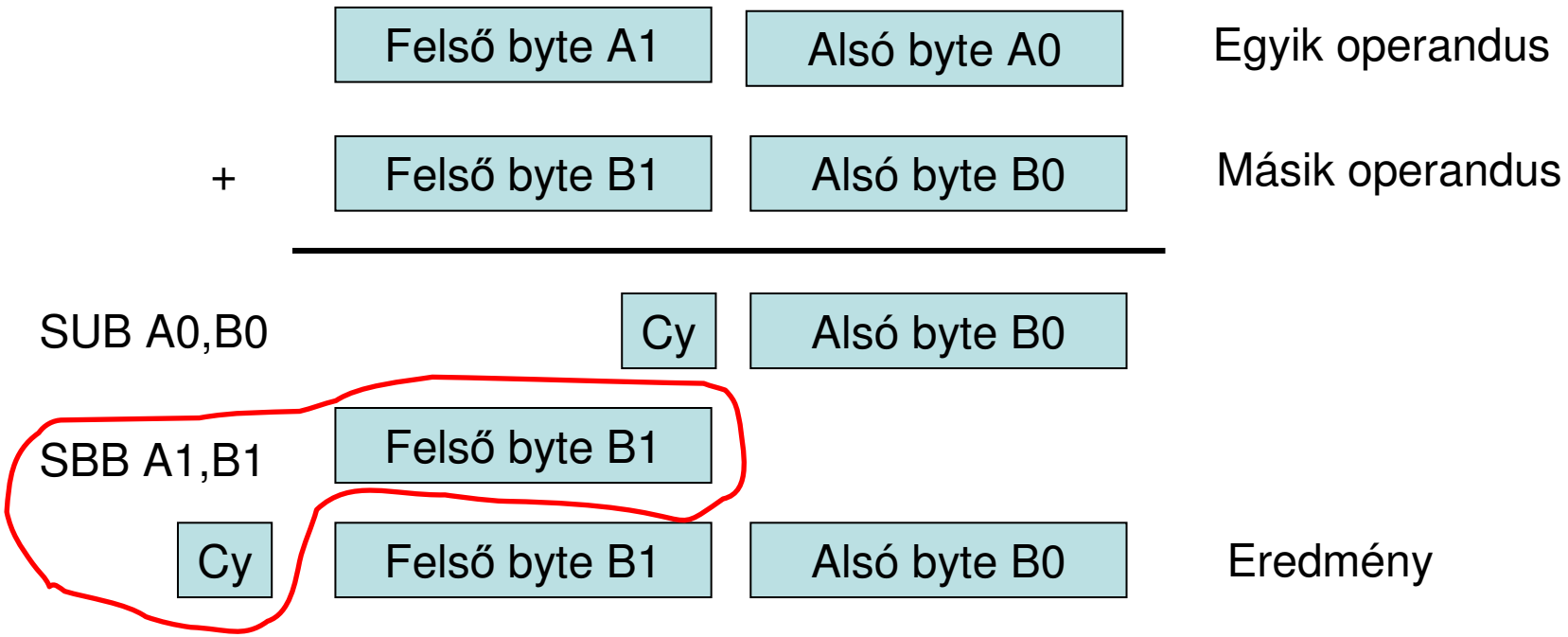
- j = j - i; vagy j -= i;

- signed long i;
- signed long j;

- j = j - i; vagy j -= i;

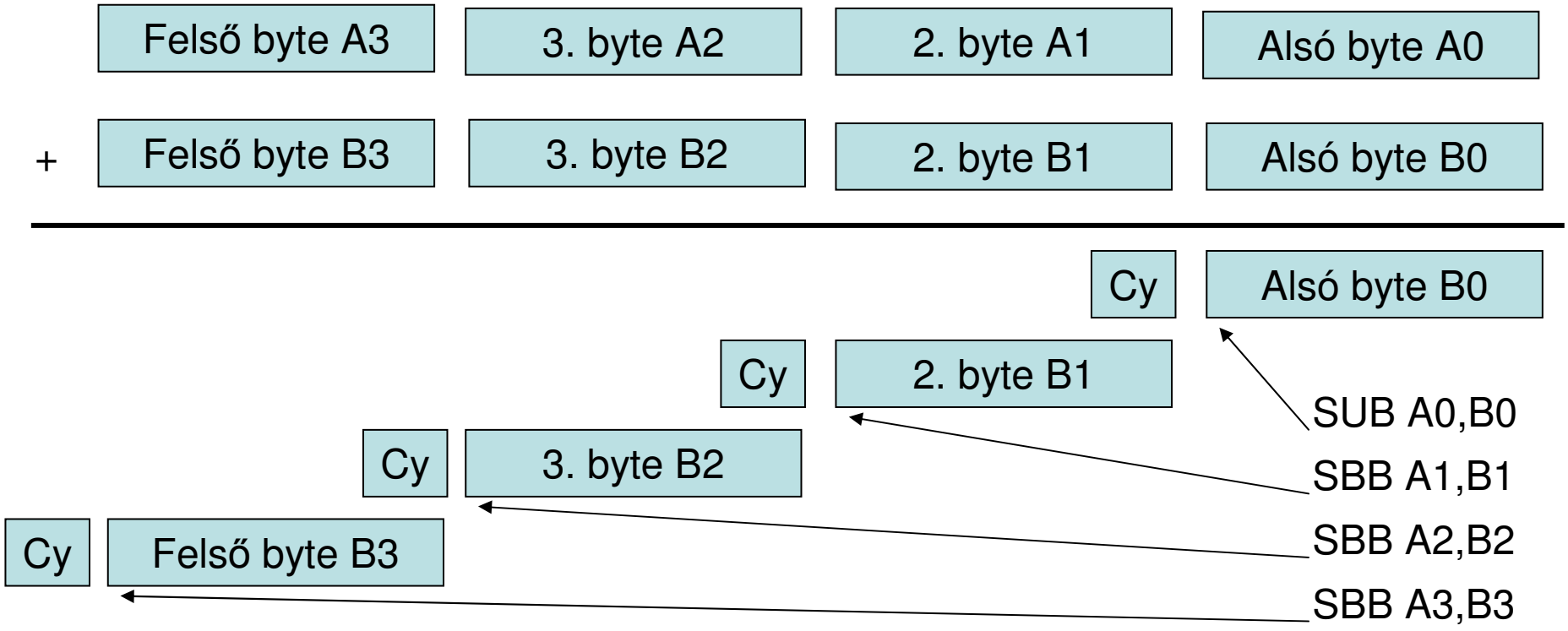
Példák

- 2 byte-os kivonás



Példák

- 4 byte-os kivonás



Összehasonlítás

- unsigned char l;
- // boolean l;

- if (l) …;

- if (!l) …;

- signed int i;
- signed int j;

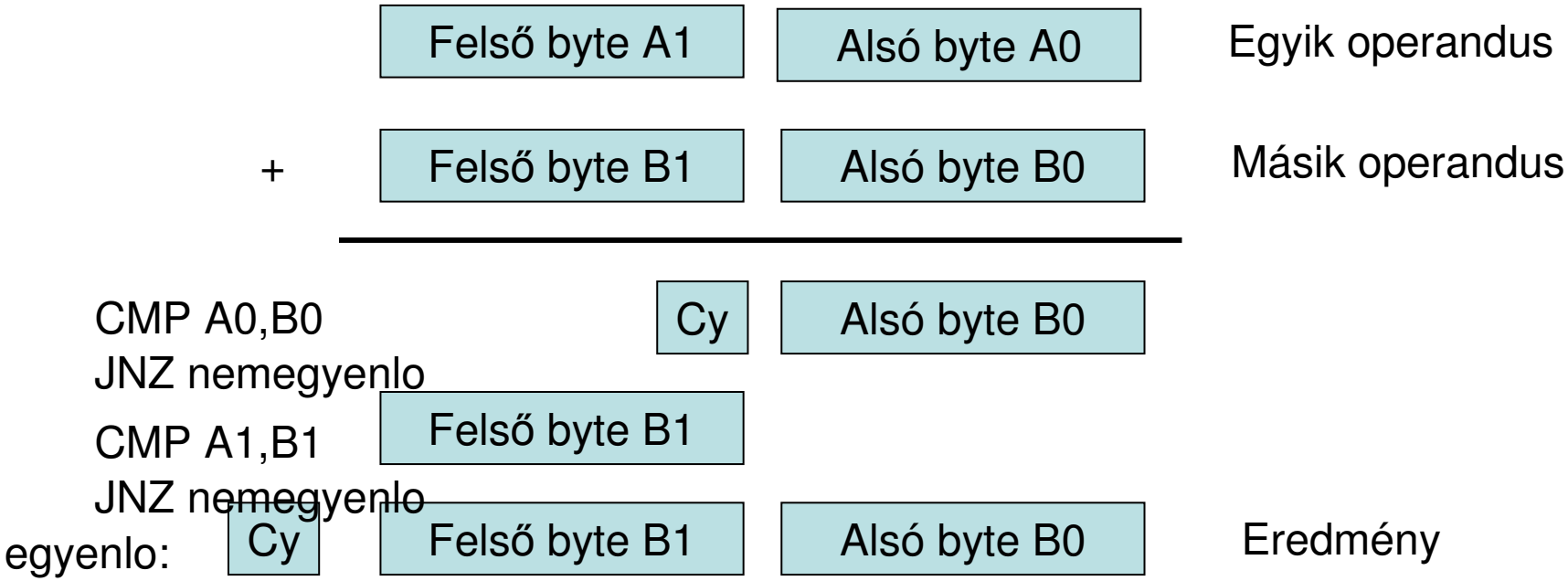
- if (j == i) …;

- signed long i;
- signed long j;

- if (j == i) …;

Példák

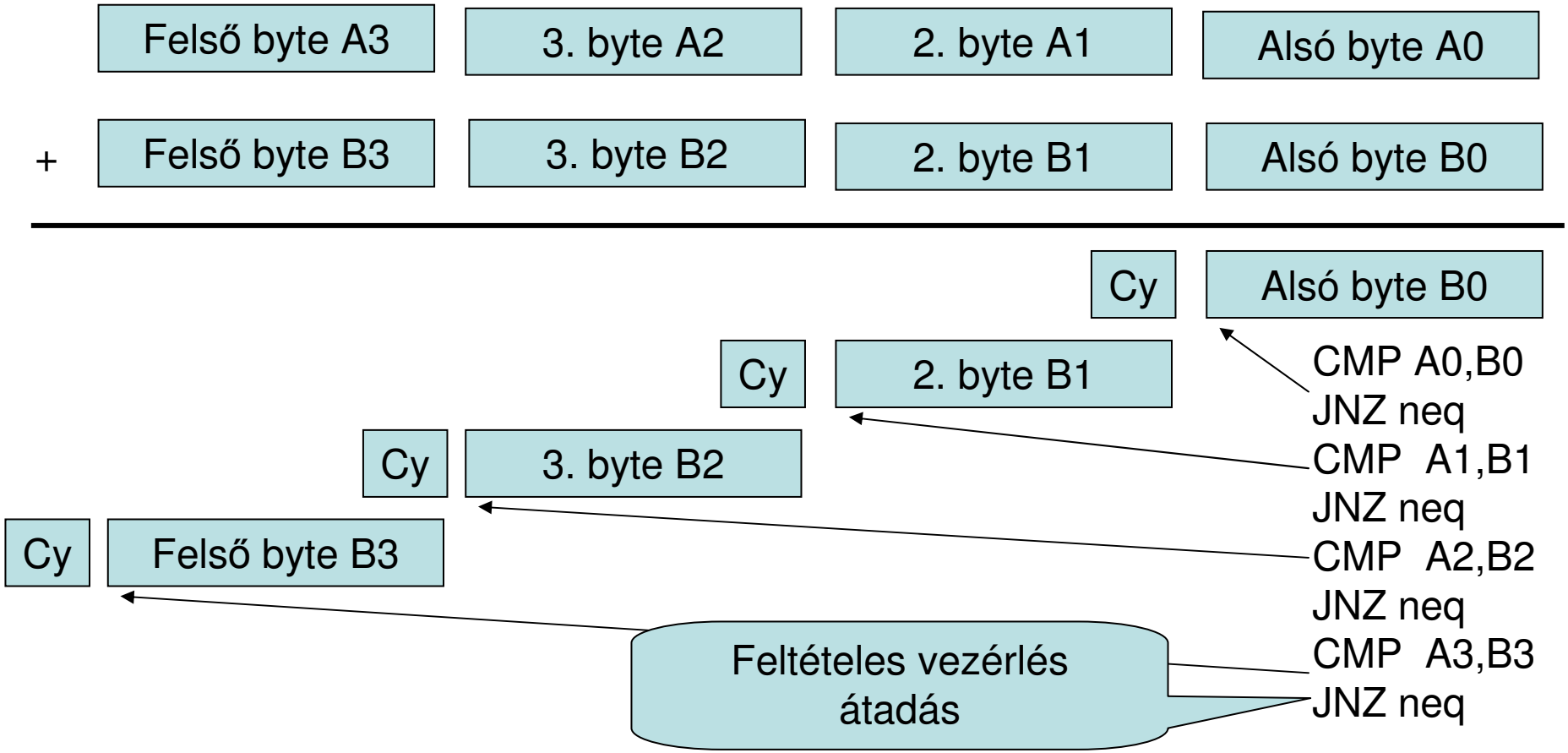
- 2 byte-os összehasonlítás



nemegyenlo:

Példák

- 4 byte-os összehasonlítás



Feltételek

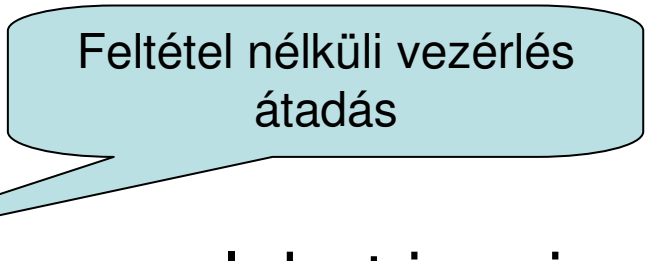
Instruction	Mnemonic	Opcode	C‡	N‡	Z‡	V‡	Operation
Jump if Carry	JC	03	1	x	x	x	
Jump if No Carry	JNC	07	0	x	x	x	
Jump if Equal	JEQ	02	x	x	1	x	
Jump if Not Equal	JNE	06	x	x	0	x	
Jump if Nonzero	JNZ	06	x	x	0	x	
Jump if Zero	JZ	02	x	x	1	x	
Jump if Lower	JLO	0F	0	x	0	x	
Jump if Higher or Same	JHS	0B	–	x	–	x	(C = 1) OR (Z = 1)
							—Signed Operation—
Jump if Greater	JG	0E	x	–	–	–	Z OR (N XOR V) = 0
Jump if Greater or Equal	JGE	0D	x	–	x	–	N XOR V = 0
Jump if Less	JL	09	x	–	x	–	N XOR V = 1
Jump if Less or Equal	JLE	0A	x	–	–	–	Z OR (N XOR V) = 1
Jump if Negative	JN	01	x	1	x	x	
Jump if Positive	JP	04	x	0	0	x	
Jump if Positive or Zero	JPZ	05	x	0	x	x	
Jump if No Overflow	JNV	0C	x	x	x	0	
Jump if Overflow	JV	08	x	x	x	1	

Program counter

- Mutató a következő programsorra
- TMS 370-nél 2 byte előjel nélküli szám
  - Maximális címezhető terület 64K
  - azaz 65536 db byte
- Összefüggés, különbség az elhelyezés számlálóval

Példa

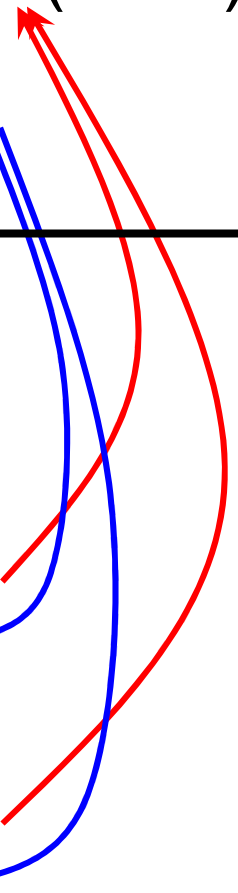
- PC = 2000H
  - 2000 52 60 mov #60h,A
- PC = 2002H
  - 2002 fd ldsp
- PC = 2003H
  - 2003 ‘8c 20 69 br init ; ez lehet jmp is
- PC = **2069H**
  - 2006 … ← ide nem adódik a vezérlés
  - ITT CIMKÉNEK kell állni !!!!!!!



függvény

- void fv(void)
- { … };

- {
- …
- fv();
- …
- fv();
- }



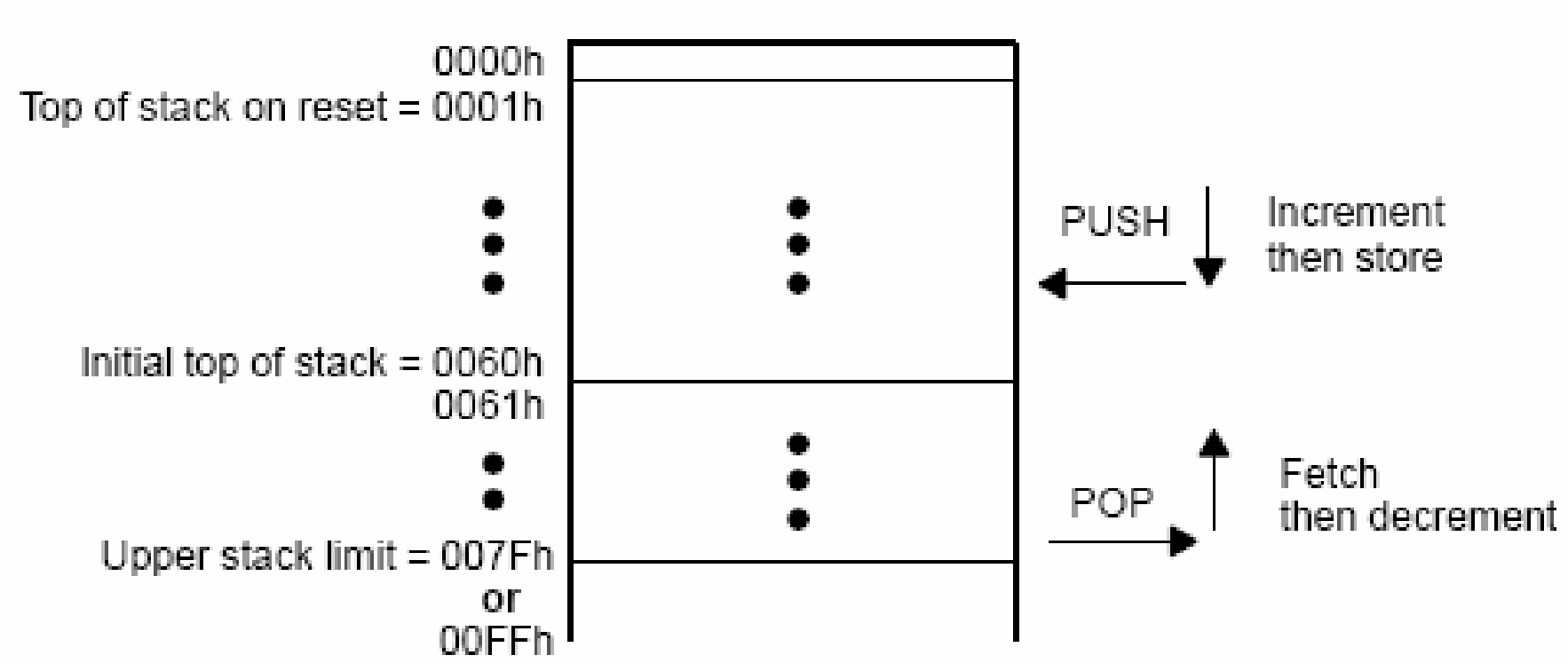
Függvények szubrutinok

- cimke:
- …
- rts

- Igénybevétele
- call cimke ; akár többször is
- …
- call cimke ; ez egy másik hely

STACK

- FILO
- megvalósítás



Példa

- PC = 2000H SP = ????
  - 2000 52 60 mov #60h,A
- PC = 2002H SP = ????
  - 2002 fd ldsp
- PC = 2003H SP = 0060H
  - 2003 ‘8e 20 69 call init
- Stack 0061 ← 20H 0062 ← 06H
- PC = **2069H** SP = 0062H
  - 2006 … ← ide visszatér a vezérlés

Példa

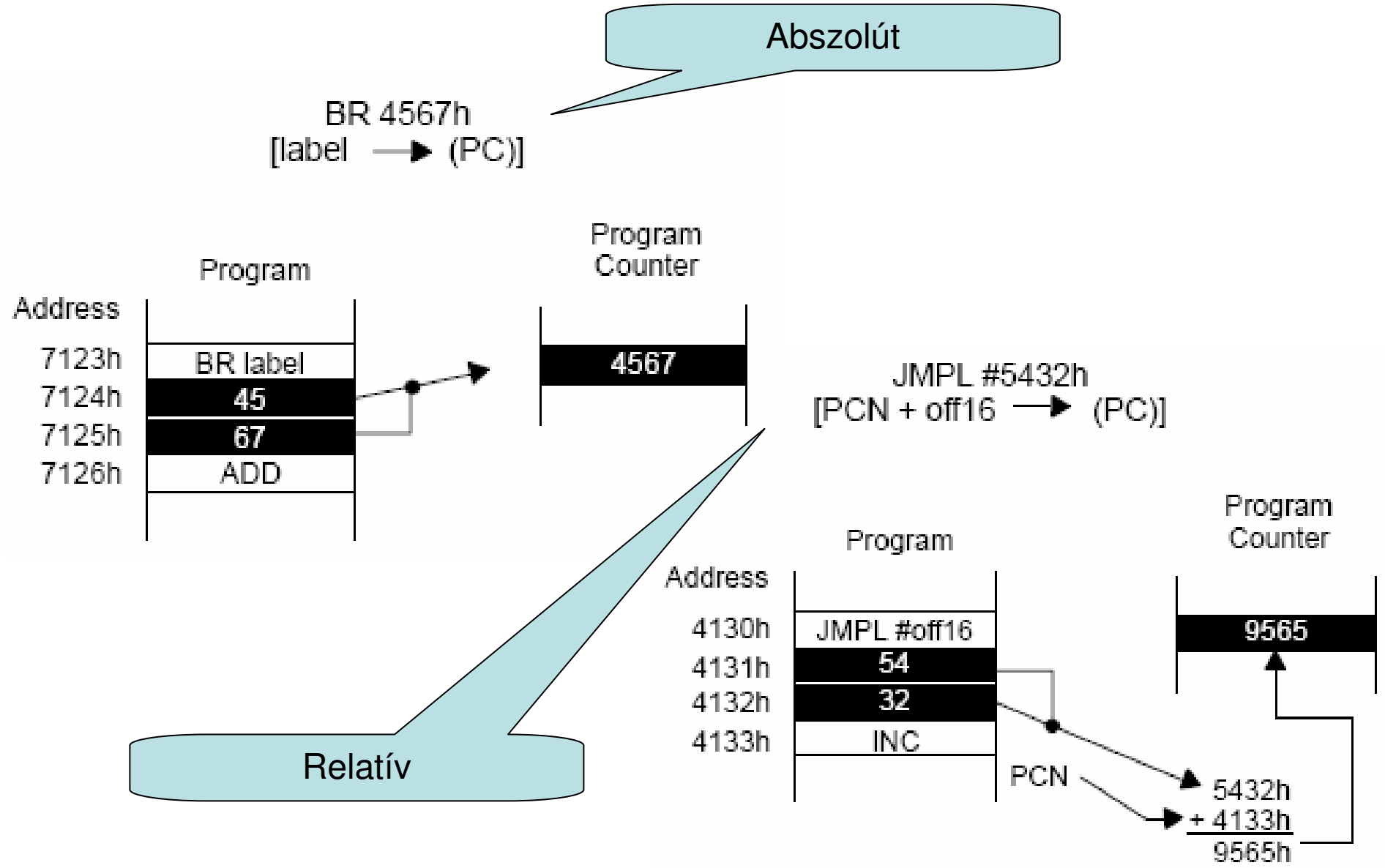
```
Wait2:
        call Wait
        call Wait
        rts
        jmp Wait        ; rts ott
Wait:
        mov #0ffh,C
Wait_l:
        djnz C,Wait_l
        rts
```

Címzési módok

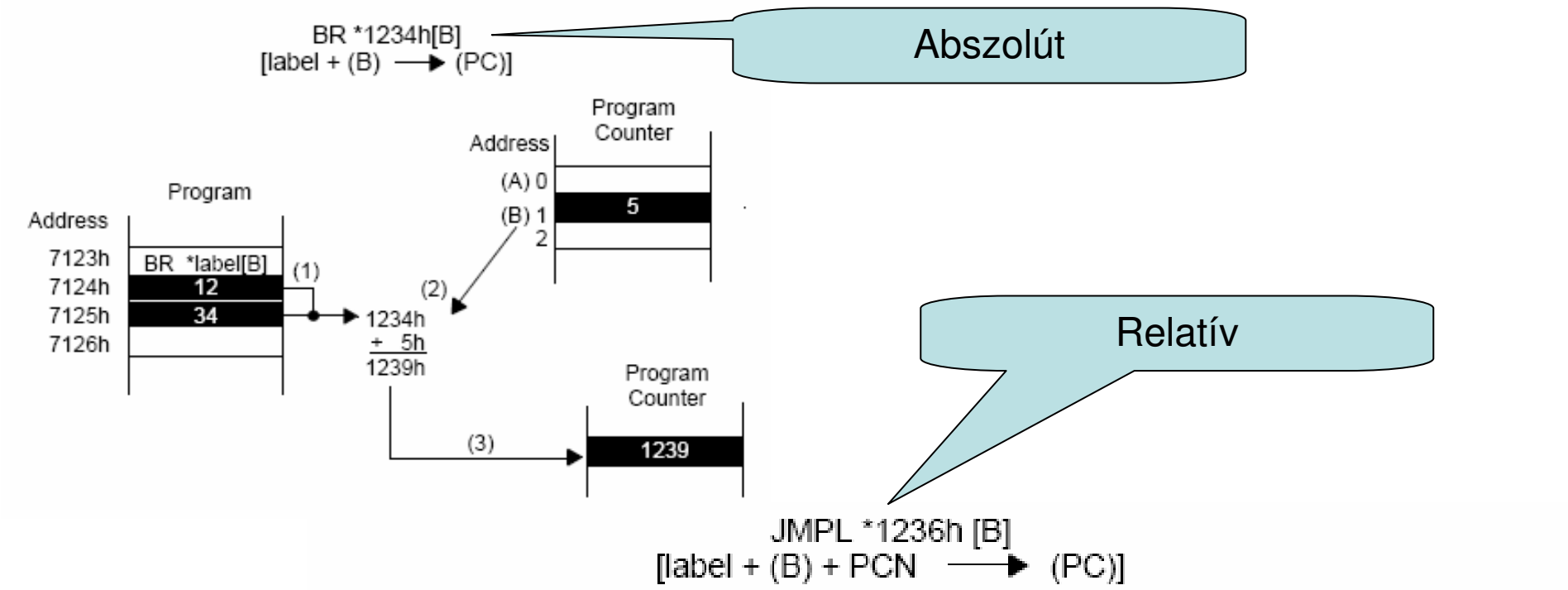
- Abszolút
- Indexelt abszolút
- Indirekt abszolút
- Offszet indirekt abszolút

- PC relatív
- Direkt relatív
- Indexelt relatív
- Indirekt relatív
- Offszet indirekt relatív

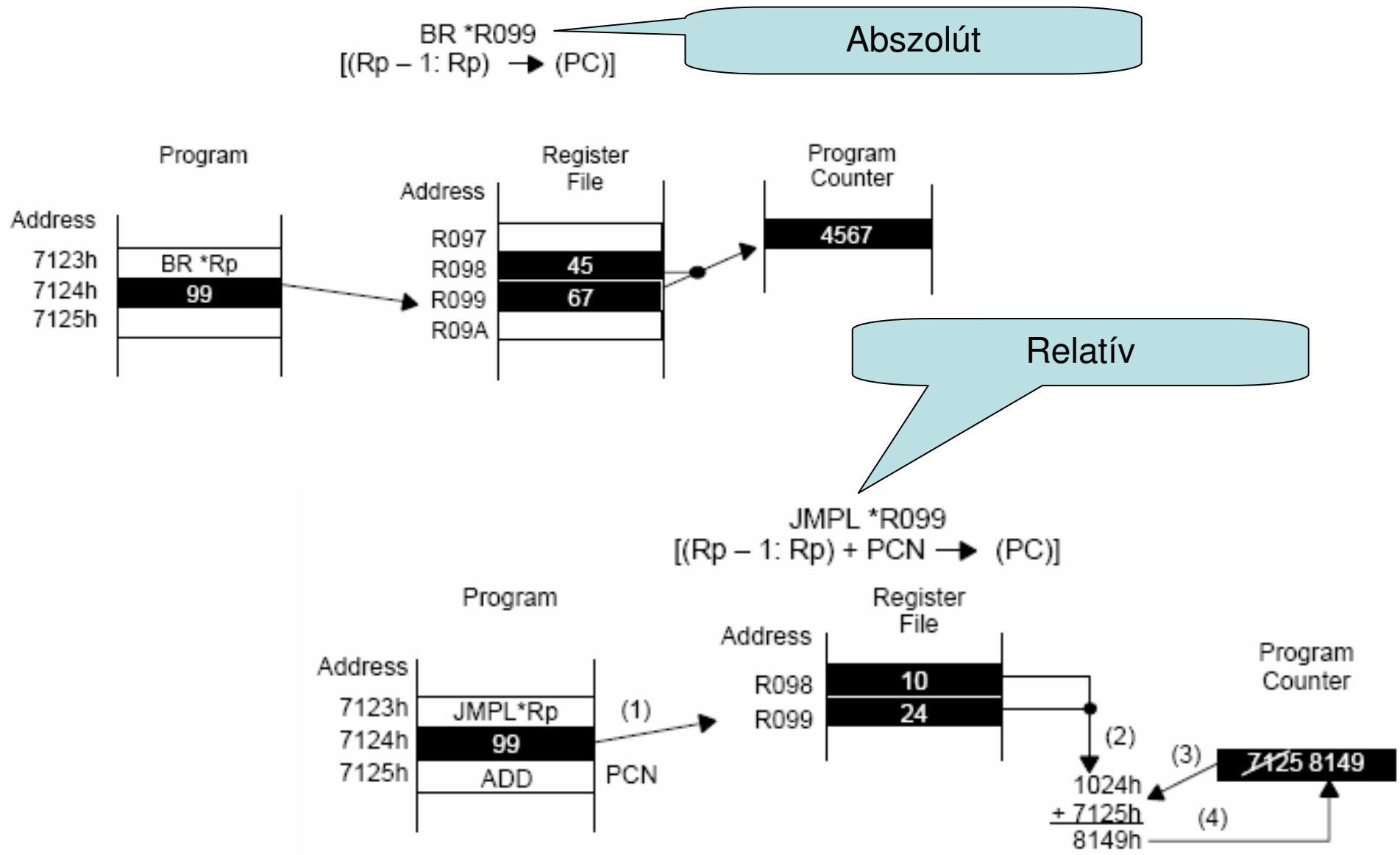
Közvetlen címzés



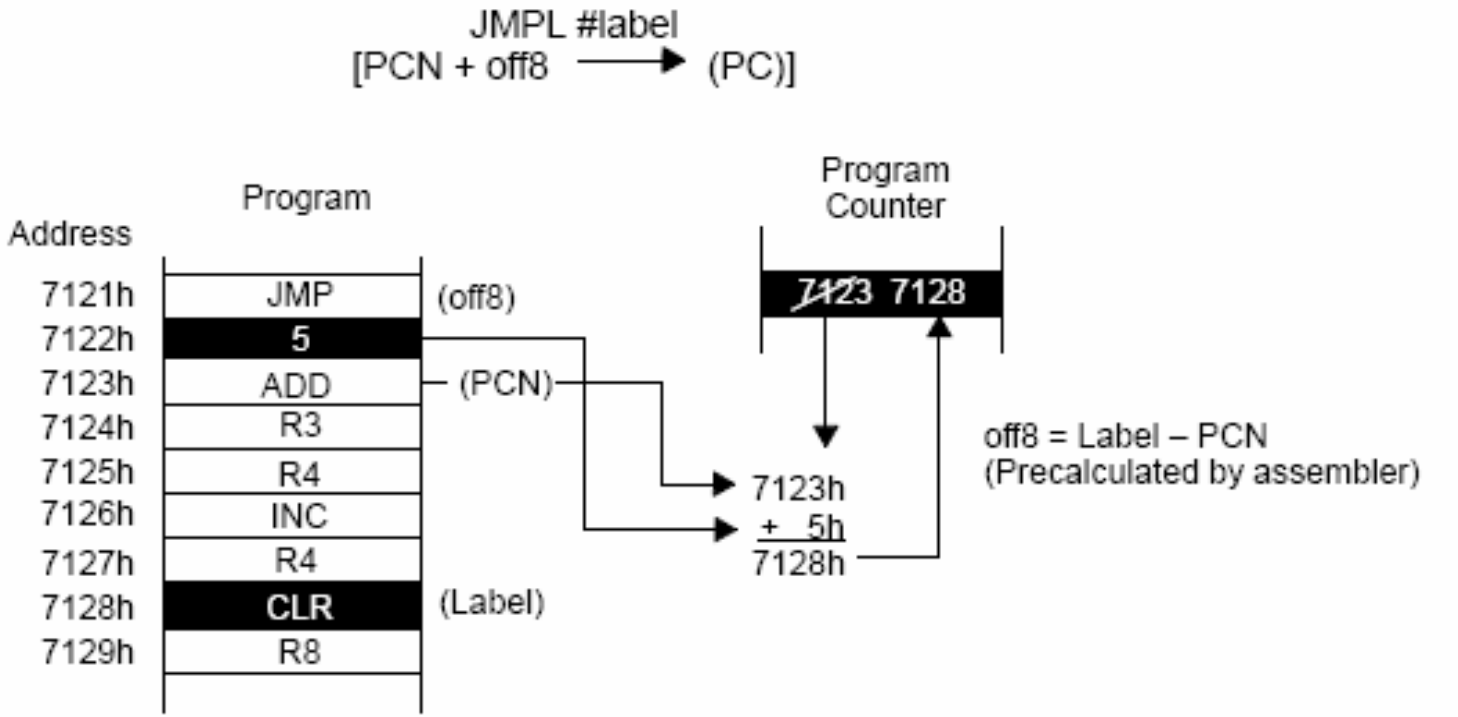
Indexelt címzés



Indirekt címzés



PC relatív


JMPL #label
[PCN + off8 → (PC)]
Program
Address
Program Counter
7121h
7122h
7123h
7124h
7125h
7126h
7127h
7128h
7129h
JMP
5
ADD
R3
R4
INC
R4
CLR
R8
(off8)
(PCN)
(Label)
7123 7128
7123h
+ 5h
7128h
off8 = Label – PCN
(Precalculated by assembler)

Adatok kezelése változók

- unsigned int i;
- … = i;
- Cím kiszámítása:
  - &i
  - *(&i)

Adatok kezelése tömbök

- unsigned int i[10];
- … = i[index];
- Cím kiszámítása:
  - sizeof(unsigned int) * index
  - sizeof(unsigned int) * index + &i
  - *(sizeof(unsigned int) * index + &i)

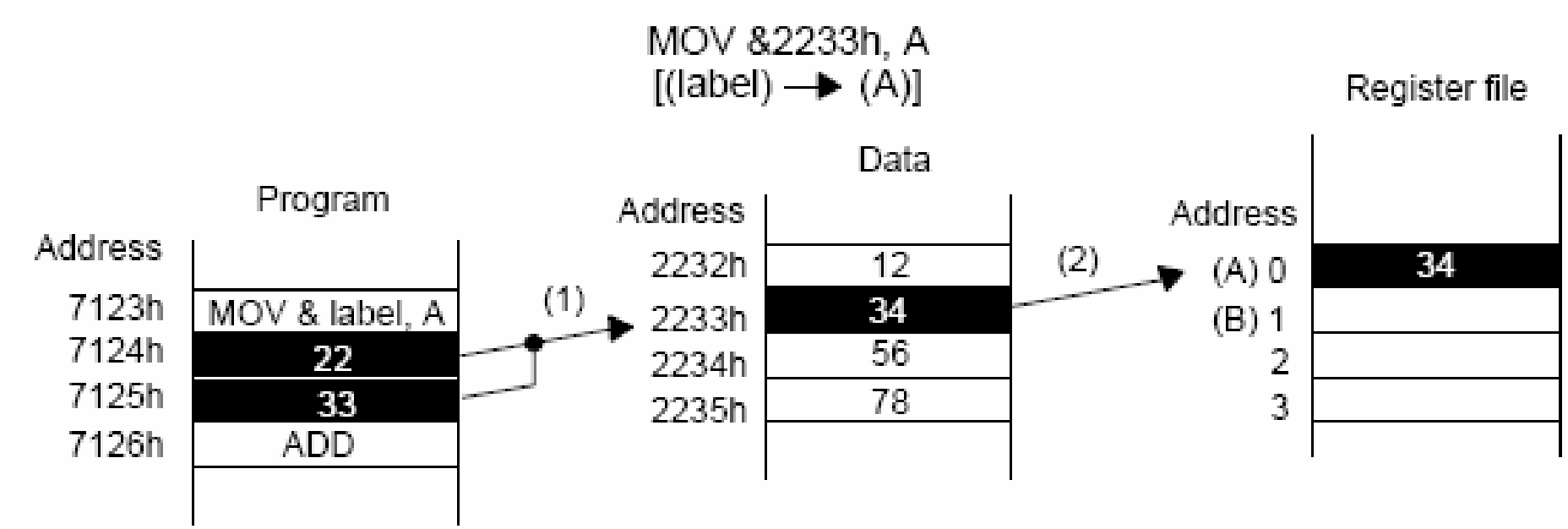
Adatok kezelése strutúrák

- Struct test {
- unsigned int i;
- signed char c;
- }t[10];

- … = t[index].c;
- Cím kiszámítása:
  - sizeof(struct test) * index
  - sizeof(struct test) * index + &t
  - *(sizeof(struct test) * index + &t + offset(c))

Közvetlen címzés

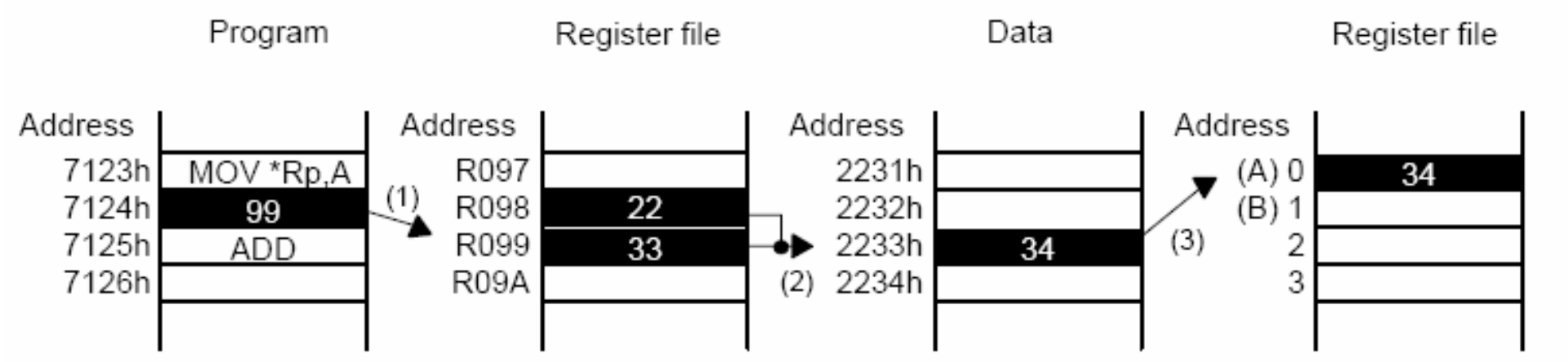
Közvetlenül kezelhető változók



Indirekt címzés

- Az adat címét egy regiszter tartalmazza
- Pl.: mov *R099,A

  [((Rp-1:Rp)) → (A)]



Offset indirekt címzés

- mov *2h[R099],A
- [(off8 + (Rp-1):Rp)) → (A)]