

## A számítógépek felépítése 1.: A Neumann-elvű számítógép

Markó Tamás  
PTE TTK, 2003

2003.07.11.

Markó Tamás, PTE TTK

1

## Vázlat

- Irodalom: J. L. Hennessy, D. A. Patterson: *Computer Organization and Design*
- Vázlat:
  - alapelvek
  - utasításslámláló
  - regiszter
  - regiszter-fájl
  - relatív címzés
  - blokkvázlat
  - utasításkészlet
  - végrehajtási ciklus
  - összeadás
  - betöltés
  - összehasonlítás és ugrás
  - példaprogram

2003.07.11.

Markó Tamás, PTE TTK

2

## Neumann János (1903-57)

- Külföldi irodalomban: John von Neumann
- Matematikus
  - a játékelmélet megalapítója
  - halmazelmélet
  - algebra
  - funkcionálanalízis
- Az USA-ban a hidrogénbomba előállításán dolgozik
- Vezető szerepe van a számítógépek elméletének kidolgozásában



2003.07.11.

Markó Tamás, PTE TTK

3

## A Neumann-elvű számítógép (1945)

- Alkatrészei: vezérmű, számológómű, tár, adatbeviteli és -kiviteli egység
- Általános célú, a működését vezérlő program kicserélhető
- Matematikai és logikai műveletek elvégzésére is képes
- Egyszerre mindig csak egy utasítást hajt végre
- A kiindulási adatok, a programok és az eredmények egy közös tárban vannak
- A tár azonos méretű rekeszekből áll, mindegyik rekesz egy sorszámmal (a címével) azonosítható
- A program egymás után következő utasításai a tár egymás után következő rekeszeiben vannak
- Van egy utasításslámláló, ami mindig a soron következő utasítás címét tartalmazza
- Van vezérlésátadó utasítás, amivel a program soros végrehajtásától el lehet térni

2003.07.11.

Markó Tamás, PTE TTK

4

## Kiegészítés az alapelvekhez 1.

- Szószervezésű memória (az eredeti elképzelés): bármelyik memóriarekesz tartalmazhat
  - egy adatot (pl. egy számot)
  - vagy egy utasítást
- Bájtszervezésű memória (a jelenlegi általános megoldás):
  - a rekeszek egy bájtosak
  - egy adat és egy utasítás több (nem mindig azonos számú) rekeszt is elfoglalhat
  - az egymás után végrehajtandó utasítások még mindig folytonosan vannak a memóriában

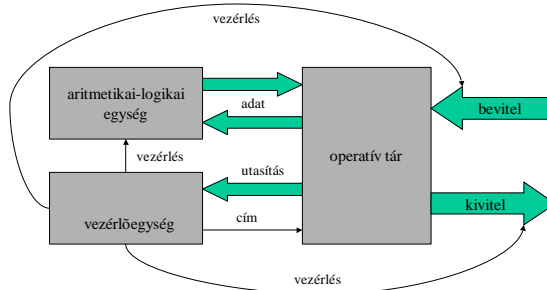
5

## Kiegészítés az alapelvekhez 2.

- Processzor (CPU, Central Processing Unit):
  - számológómű (aritmetikai-logikai egység, Arithmetic-Logic Unit, ALU)
  - vezérmű (vezérlő egység, Control Unit, CU)
  - regiszterek (gyors tárolóhelyek)
- Az utasításslámláló megvalósítása: egy regiszter a processzorban (PC, Program Counter),
- Egy utasítás végrehajtása három szakaszra bontható:
  - az operandusok beolvasása a memóriából a processzor regisztereibe
  - egy művelet elvégzése ezeken az operanduszokon
  - az eredmény kiírása a memóriába

6

## A számítógép blokkvázlata 1.

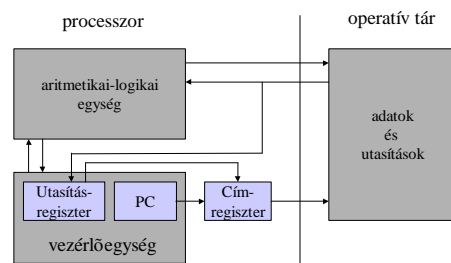


2003.07.11.

Markó Tamás, PTE TTK

7

## A számítógép blokkvázlata 2.



2003.07.11.

Markó Tamás, PTE TTK

8

## Az utasításszámláló

Hogyan lehet meghatározni a következő végrehajtandó utasítás címét?

**1. lehetőség:** benne van ez is az utasításban

Az utasítás alakja így:

{Művelet (Cím<sub>op1</sub>, Cím<sub>op2</sub>, Cím<sub>eredm</sub>), Cím<sub>köv.ut.</sub>}

**Probléma:** sok felesleges információ, mert nagyon gyakran a következő címen van a következő utasítás: Cím<sub>köv.ut.</sub> = Cím<sub>akt.ut.</sub> + 1

**Megoldás:** a 2. lehetőség

**2. lehetőség:** utasításszámláló, aminek a tartalma az utasítás beolvasása után automatikusan eggyel növekszik

**Probléma:** és ha mégsem a következő címen lévő utasítás kell?

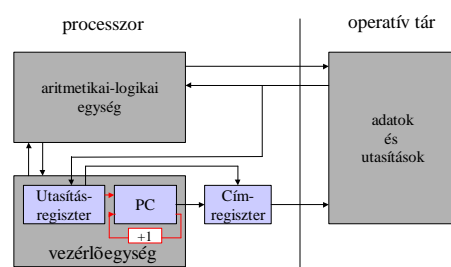
**Megoldás:** speciális utasítások a PC tartalmának átírására (ugró utasítás, JUMP)

2003.07.11.

Markó Tamás, PTE TTK

9

## Számítógép utasításszámlálójával



2003.07.11.

Markó Tamás, PTE TTK

10

## A regiszterek

**A tények:**

1. A memória kb. tízszer lassabb, mint a processzor
2. A processzor és a memória közötti interfész (adatátviteli lehetőség) drága, ezért „karcsú”: általában 1 szó / hozzáférés
3. A tipikus műveletekben három tényező vesz részt:  
Eredmény := Tényező<sub>1</sub> Műv Tényező<sub>2</sub>

**A megoldás:**

- a processzornak van belső puffertárolója: a regiszterek
- a processzor a műveleteket a regiszterek tartalmával végzi
- kivételek:
  - load (adott memóriarekesz tartalmának betöltése egy regiszterbe)
  - store (egy regiszter tartalmának tárolása adott memóriarekeszbe)

Az ilyenfajta számítógépeket **load-store** architektúrájának hívják.

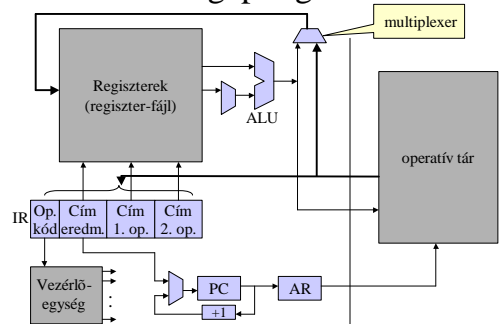
Nem minden számítógép **load-store** architektúrájú (pl. a Pentium sem), de minden modern processzornak vannak regiszterei.

2003.07.11.

Markó Tamás, PTE TTK

11

## Számítógép regiszterekkel



2003.07.11.

Markó Tamás, PTE TTK

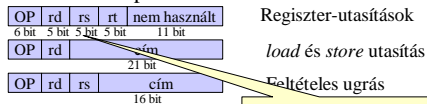
12

## Az abszolút címzés problémája

### Feltételezések:

- a szóhossz (a memóriarekeszek hossza) 32 bit
- 64 különböző utasítás? a művelet kódja 6 bites
- 32 regiszter? egy regiszter címe 5 bites

### Az utasítások felépítése:



Regiszter-utasítások

load és store utasítás

Feltételes ugrás

### Probléma:

A 21 bites címrészen csak  $2^{21}$  (2M), a 16 bites címrészen csak  $2^{16}$  (64k) memóriarekesz címezhető

2003.07.11.

Markó Tamás, PTE TTK

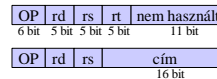
13

## A relatív címzés bevezetése

Az alapötlet: a memóriában lévő operandusz címét úgy számítjuk ki:

$$\langle \text{memóriacím} \rangle := \langle \text{egy regiszter tartalma} \rangle + \langle \text{az utasításban megadott cím} \rangle$$

Így - mivel a regiszterek 32 bitesek - mindig 32 bites címet kapunk, 4G db memóriarekesz címezhető közvetlenül



Regiszter-utasítások

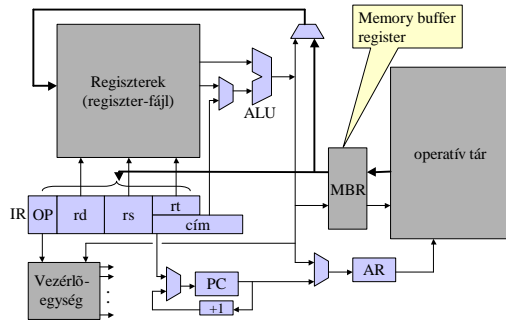
load / store utasítás és feltételes ugrás

2003.07.11.

Markó Tamás, PTE TTK

14

## Egy egyszerű számítógép felépítése



2003.07.11.

Markó Tamás, PTE TTK

15

## Egy egyszerű gép utasításkészlete

Kategória	Művelet	Utasítás	Jelentése
aritmetikai	add	add rd, rs, rt	$R[rd] := R[rs] + R[rt]$
	sub	sub rd, rs, rt	$R[rd] := R[rs] - R[rt]$
adatmozgató	load	load rd, rs, addr	$R[rd] := MM[R[rs] + \text{addr}]$
	store	store rd, rs, addr	$MM[R[rs] + \text{addr}] := R[rd]$
feltételes ugrás	beq	beq rd, rs, addr	ha $R[rd] = R[rs]$ , $PC := \text{addr}$
	bne	bne rd, rs, addr	ha $R[rd] \neq R[rs]$ , $PC := \text{addr}$

### Jelmagyarázat:

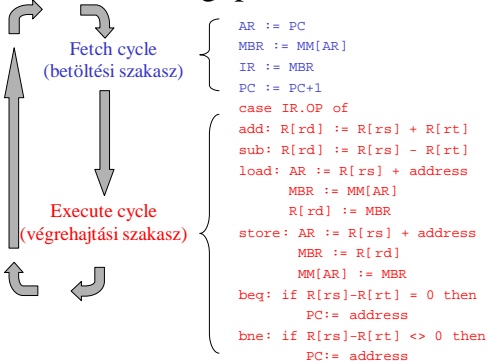
rd, rs, rt: egy-egy regiszter címe  
 rd: destination, célregiszter (eredmény)  
 rs: source, forrásregiszter (kiindulási adat)  
 rt: a másik forrásregiszter  
 R[n]: az n-edik regiszter  
 MM[n]: az n-edik memóriarekesz  
 PC: utasítászámáláló (Program Counter)

2003.07.11.

Markó Tamás, PTE TTK

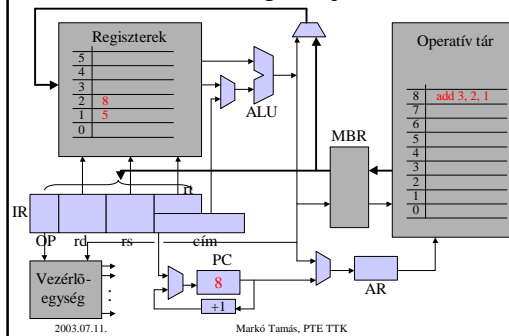
16

## A számítógép működési ciklusa



17

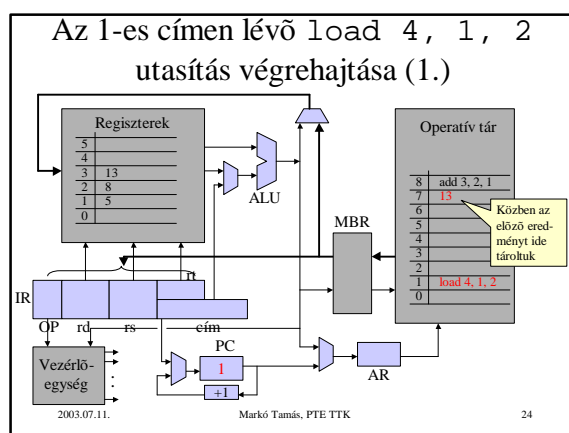
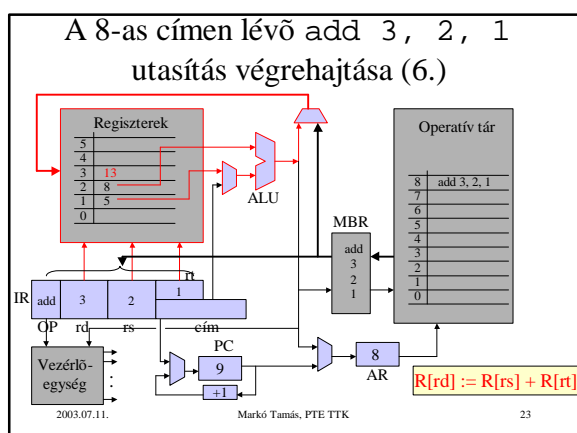
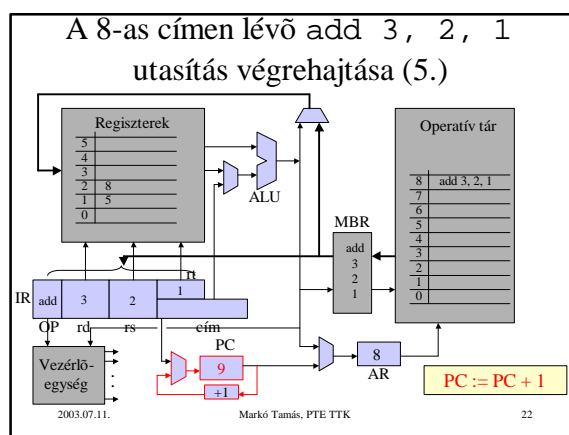
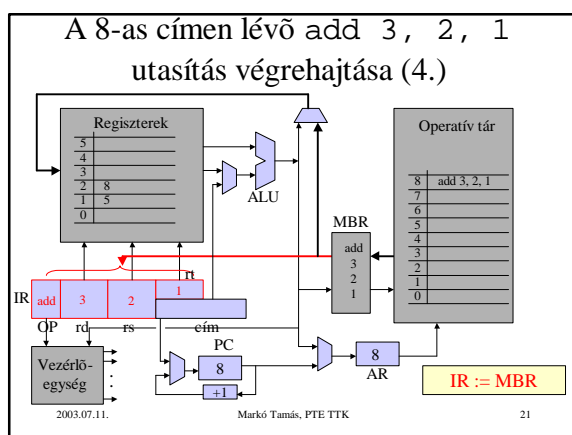
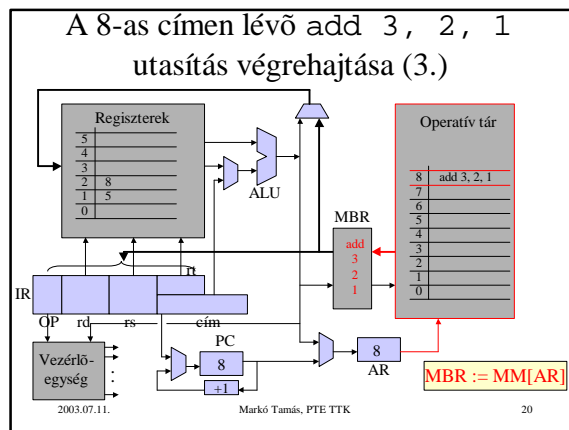
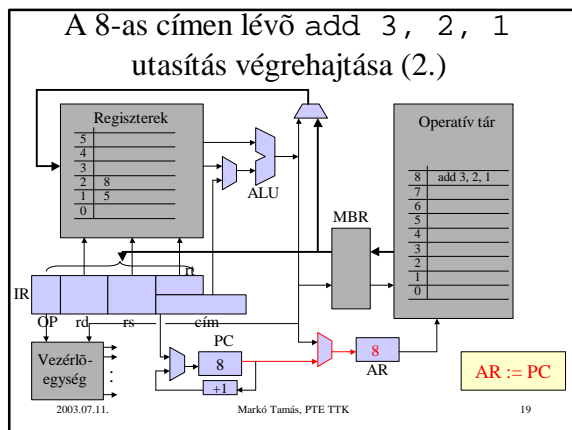
## A 8-as címen lévő add 3, 2, 1 utasítás végrehajtása (1.)



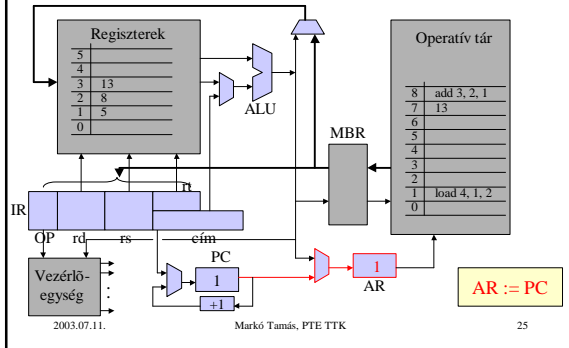
2003.07.11.

Markó Tamás, PTE TTK

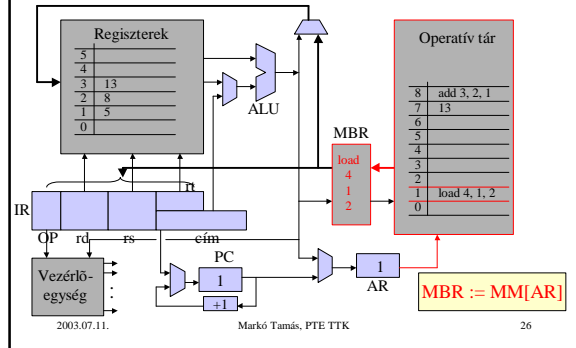
18



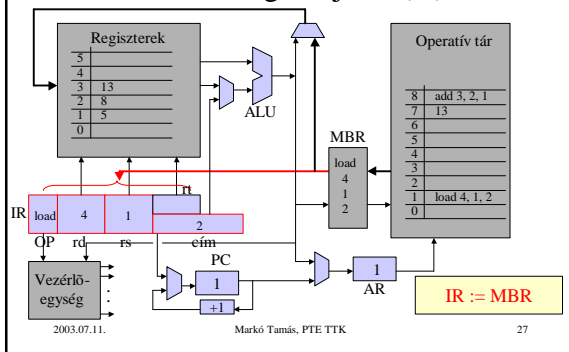
Az 1-es címen lévő load 4, 1, 2 utasítás végrehajtása (2.)



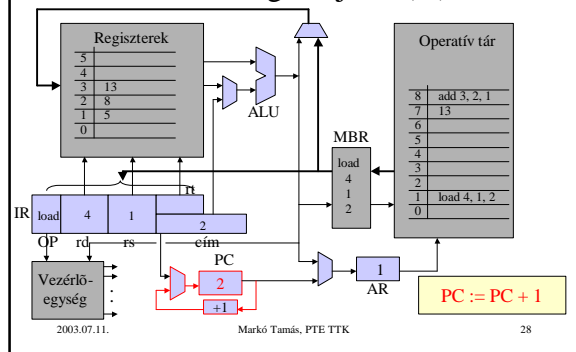
Az 1-es címen lévő load 4, 1, 2 utasítás végrehajtása (3.)



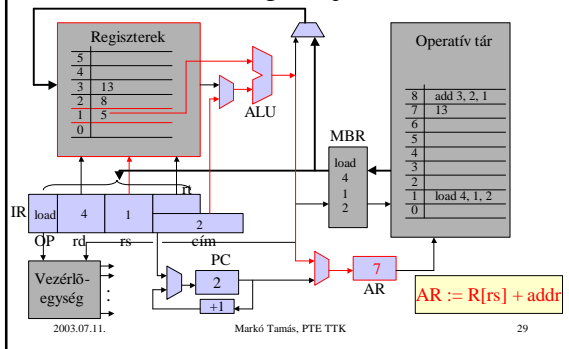
Az 1-es címen lévő load 4, 1, 2 utasítás végrehajtása (4.)



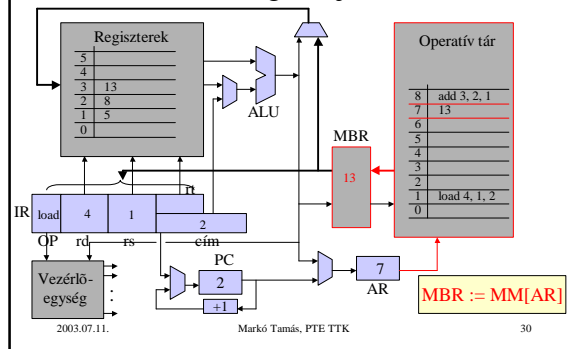
Az 1-es címen lévő load 4, 1, 2 utasítás végrehajtása (5.)

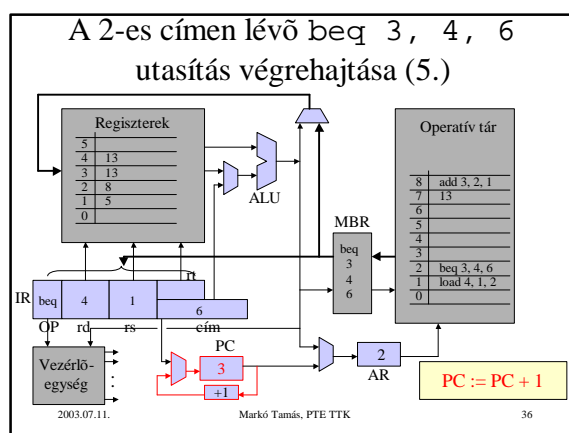
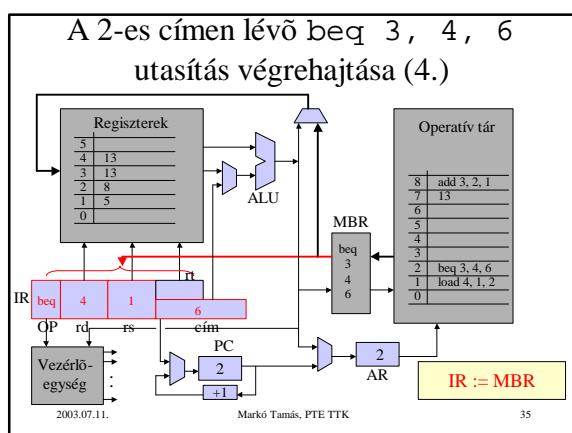
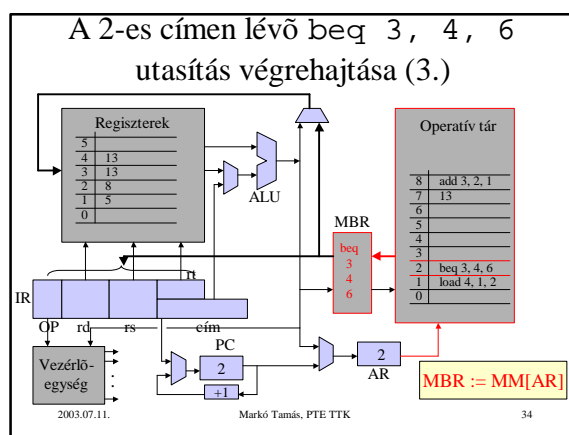
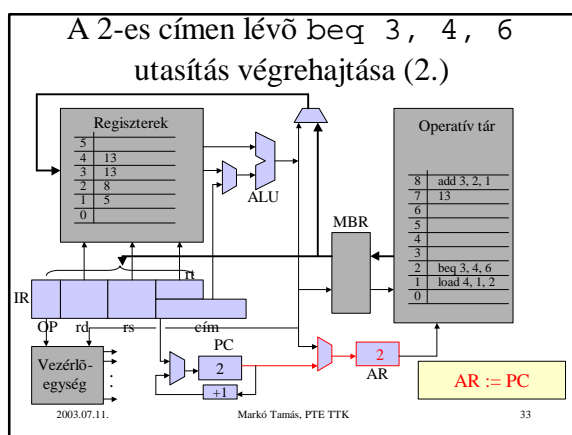
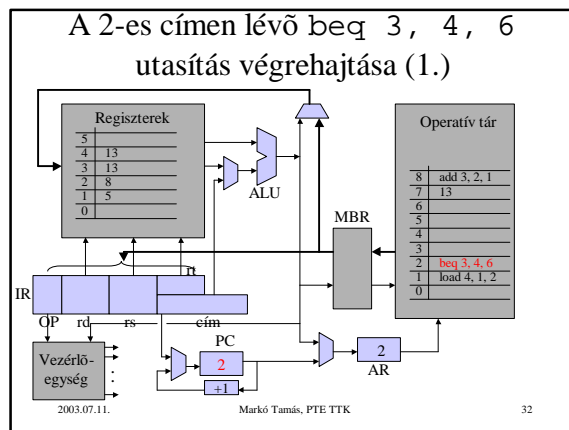
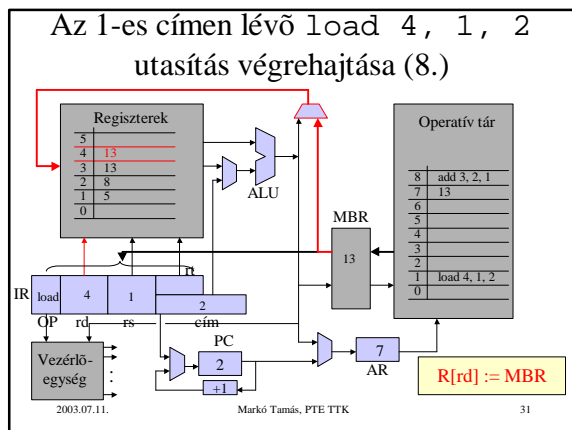


Az 1-es címen lévő load 4, 1, 2 utasítás végrehajtása (6.)

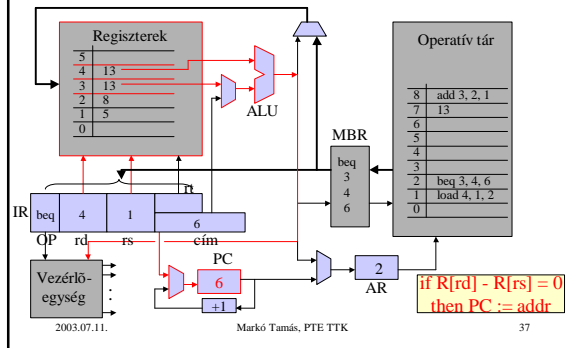


Az 1-es címen lévő load 4, 1, 2 utasítás végrehajtása (7.)





## A 2-es címen lévő beq 3, 4, 6 utasítás végrehajtása (6.)



## Példaprogram (1.)

Az elvégzendő számítás (Pascalban leírva):

```
for i := 1 to 10 do
  a[i] := a[i] * a[i+1]
```

Feltételezések:

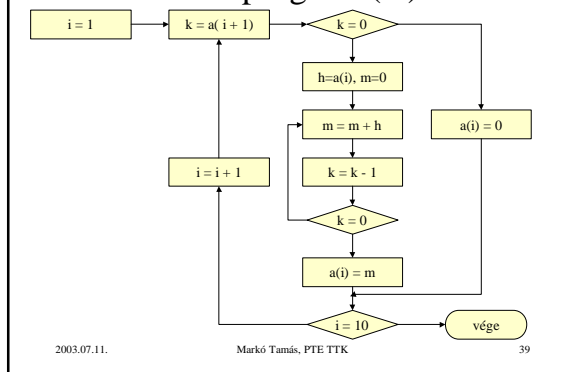
- a számítógép nem tud szorozni, ezért az  $a[i]$  tartalmát  $a[i+1]$ -szer összeadjuk
- szükségünk van a 0, 1 és 10 konstansokra, ezeket a 100, 101 és 102-es memóriarekeszekbe töltjük
- mivel gyakran használjuk őket, bemásoljuk őket a számolás megkezdése előtt a 0, 1, 2-es regiszterbe
- az  $a[1] \dots a[11]$  értékek a 200 .. 210-es memóriarekeszekben vannak
- kezdetben minden regiszter a 0 értéket tartalmazza

2003.07.11.

Markó Tamás, PTE TTK

38

## Példaprogram (2.)



## Példaprogram (3.)

for i:=1 to 10 do a[i] := a[i] \* a[i+1]

```
1. ld 1,0,101 R[1]:=1
2. ld 2,0,102 R[2]:=10
3. add 3,0,1 R[3]:=1 a ciklusváltozó inicializálása
4. ld 5,3,200 R[5]:=a(i+1) cím = (R[3])+200 = 201
5. beq 5,0,15 R[5]:=0? speciális eset figyelése
6. ld 6,3,199 R[6]:=a(i)
7. add 7,0,0 R[7]:=0 az összeadás gyorsabb, mint ld 7,0,100
8. add 7,7,6 R[7]:=R[7]+a(i) R[7]-et akkumulátorként használjuk
9. sub 5,5,1 R[5]:=a(i+1)-1
10. bne 5,0,8 ugrás a 8-as címre: a[i]-t a[i+1]-szer adjuk össze
11. sto 7,3,199 a[i]:=a[i]*a[i+1]
12. beq 3,2,17 a ciklus végére értünk-e
13. add 3,3,1 a ciklusváltozó növelése
14. bne 1,2,4 feltétlen ugrás a 4-es címre (R[1] ? R[2])
15. sto 0,3,199 a[i]:=0
16. bne 1,2,12 feltétlen ugrás a 12-es címre
```

2003.07.11.

Markó Tamás, PTE TTK

40