

A számítógépek felépítése 5.b: Információ-ábrázolás: utasítások és címzési módok

Markó Tamás
PTE TTK, 2003

2003.07.31.

Markó Tamás, PTE TTK

1

A rádiótelefonokat kérem KIKAPCSOLNI!

2003.07.31.

Markó Tamás, PTE TTK

2

Utasítások és címzés

- Az utasítások általános alakja:
$$x_e = f(x_1, \dots, x_n)$$
- Ez a végrehajtandó műveleten (f) kívül $n+1$ címet tartalmaz (x_e az eredmény címe, x_1, \dots, x_n az n db argumentum címe)
- **Definíció:**
Egy gépet m címesnek hívunk, ha az utasítások m db címet explicit módon tartalmaznak

2003.07.31.

Markó Tamás, PTE TTK

4

Utasításhossz és az operanduszok száma

- Osztályozás:
 - az utasítások hossza fix ? változó
 - az operanduszok száma fix ? változó
- Példák:
 - fix utasításhossz (? fix számú operandusz):
MIPS
 - változó utasításhossz, fix számú operandusz:
IBM 370, PDP 11, MC 68000
 - változó számú operandusz:
VAX és majdnem mindegyik modern processzor

2003.07.31.

Markó Tamás, PTE TTK

5

Utasításformátumok

3 címes:

Opcode	source1	source2	destination
--------	---------	---------	-------------

2 címes:

Opcode	source1	source2 destination
--------	---------	------------------------

1 címes:

Opcode	source destination
--------	-----------------------

- A forrás (source) és a cél (destination) általában egy-egy cím
- Keverék: 1,5 címes formátum (az egyik cím nagyon rövid, csak egy regiszter)

2003.07.31.

Markó Tamás, PTE TTK

6

Az adatok helyének azonosítása

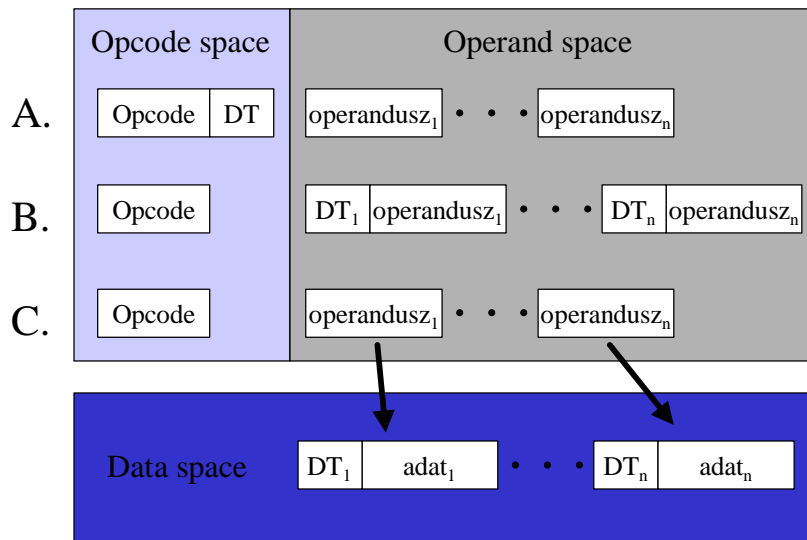
- Az előforduló helyek:
regiszter, operatív tár, I/O-tartomány, speciális regiszter, verem (stack)
- Egyik lehetőség: a hely megadása a műveleti kódban
 - példa: MC68020 PEA (Push Effective Address)
 - általában implicit módon a veremre hivatkozik
- Másik lehetőség: a hely megadása a címben
 - implicit (pl. az 1. operandusz mindig egy regiszter címe, a 2. cím az operatív tárban van)
 - explicit (a leggyakoribb megoldás)

2003.07.31.

Markó Tamás, PTE TTK

7

Az adatok típusának megadása 1.



2003.07.31.

Markó Tamás, PTE TTK

8

Az adatok típusának megadása 2.

- A műveleti kódban
 - IBM/370, Pentium, PowerPC, MIPS,...
 - Következmény: pl. 15 különféle ADD utasítás
- A címmel együtt
 - Csak kísérletek
 - Kereskedelmi forgalomban kapható gépnél nem fordul elő
- Magában az adatban
 - **Tagged architectures:** minden memória-rekesz két részből áll: a tárolandó adatból és egy címkéből (tag). A címke leírja, hogy az adatot hogyan kell értelmezni.

2003.07.31.

Markó Tamás, PTE TTK

9

Az adatok értékének megadása

- A műveleti kódban
 - ritkán fordul elő
- Az utasítás cím-résztében
 - úgynevezett *immediate* „címezés“
 - konstansok megadására
- Az adatoknál
 - szokásos megoldás

2003.07.31.

Markó Tamás, PTE TTK

10

A műveleti kód

- A szokásos megoldás:
 - fix hosszúságú műveleti kód:
 k bit ? 2^k különböző utasítás
- Probléma:
 - rövid szóhossz és sok utasítás
- Megoldás:
 - címrészt tartalmazó utasítások: rövid opcode
 - címrész nélküli utasítások: hosszú opcode
 - Figyelem! Egy rövid műveleti kódot fenn kell tartani prefixnek a hosszú opcode jelzésére!

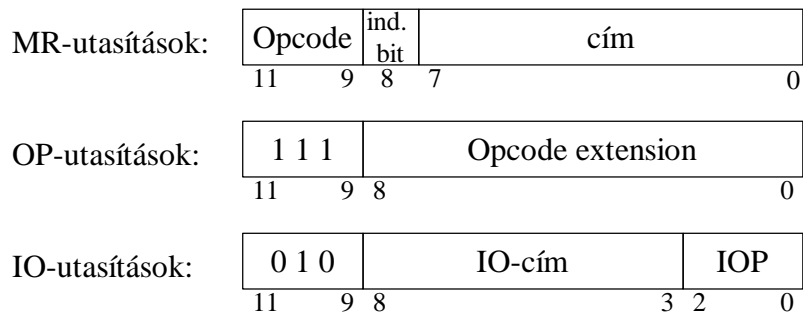
2003.07.31.

Markó Tamás, PTE TTK

11

A műveleti kód kiterjesztése

- Példa: PDP-8
 - az első tömeggyártású miniszámítógép (1965)
 - 12 bites szóhossz)



2003.07.31.

Markó Tamás, PTE TTK

12

Változó hosszúságú műveleti kód 1.

Rugalmas megoldás:

- A műveleti kód hossza a gyakoriságától függ (mint pl. a Morse-ABC)
- Optimalizálás: $L_{atl} = \sum length(I_j) P(I_j)$ minimális legyen
- Példa: öt parancsból álló utasításkészlet az alábbi gyakoriságokkal:

I_1	0,5	? 1
I_2	0,3	? 00
I_3	0,1	? 011
I_4	0,05	? 0100
I_5	0,05	? 0101

- Vegyük észre, hogy egyik műveleti kód sem fordul elő egy másik elején (prefix-tulajdonság)

2003.07.31.

Markó Tamás, PTE TTK

13

Változó hosszúságú műveleti kód 2.

- Nyereség:
 - fix hosszúságú opcode: $L_{\text{átl}} = L = 3$ bit
 - változó hosszúságú opcode:
$$L_{\text{átl}} = 0,5 \cdot 1 + 0,3 \cdot 2 + 0,1 \cdot 3 + 0,05 \cdot 4 + 0,05 \cdot 4 = 1,8$$
- Probléma:
 - a dekódolás nehezebb
 - a gyakoriságok eloszlása elvileg változhat
 - csak bitenként címezhető memória esetén értelmes
- Kereskedelmi példa:
 - Burroughs B 1700

2003.07.31.

Markó Tamás, PTE TTK

14

Az utasítások címrésze

Osztályozási szempontok:

- hivatkozási fok
- a memória fajtája
- címképzés

2003.07.31.

Markó Tamás, PTE TTK

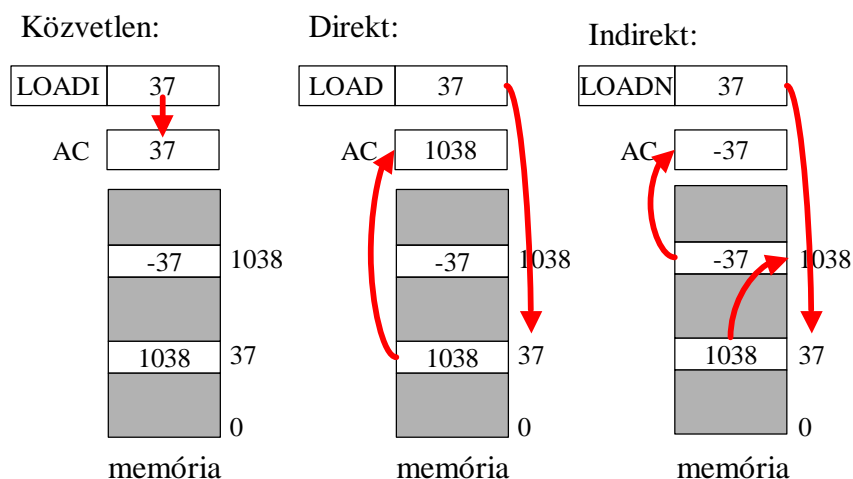
15

A hivatkozási fok

- 0-s hivatkozási fok:
 - közvetlen (immediate) „címrés”
 - maga az operandusz értéke áll az utasításban a címrészen:
 $value := instruction.adr$
- 1-es hivatkozási fok :
 - közvetlen (direkt) címrés
 - az operandusz címe van az utasításban
 $value := MM[instruction.adr]$
- 2-es hivatkozási fok :
 - indirekt címrés
 - az operandusz címének címe van az utasításban
 $value := MM[MM[instruction.adr]]$
- n -es ($n > 2$) hivatkozási fok:
 - • az előzőekhez hasonlóan képezhető (nemigen használható)

16

Címrési példák



2003.07.31.

Markó Tamás, PTE TTK

17

A memória fajtái

A legfontosabb típusok:

- regiszterek
- operatív tár
- I/O-tartomány

2003.07.31.

Markó Tamás, PTE TTK

18

Címképzés

- Abszolút címzés
 - a legegyszerűbb forma
 - az utasítás tartalmazza a teljes címet
- Relatív címzés
 - kényelmesebb: csak a cím egy része van az utasításban, a végleges cím ebből és a futási környezetből számítható
 - általános formája: $cím := f(D, R_1, \dots, R_K)$, ahol
 - D (displacement, eltolás) az utasítás címrésztében van
 - R_1, \dots, R_K kitüntetett memória-rekeszek (általában regiszterek)
 - fő alkalmazási módok:
 - bázisregiszter használata
 - indexregiszter használata

2003.07.31.

Markó Tamás, PTE TTK

19

A bázis- és az indexregiszter

- Bázisregiszter:
 - rögzített vagy az utasításban megcímzett R regiszter, amelyre $cím := R + D$
 - előnye:
 - a kód a memóriában bárhova betölthető
 - rövidítheti az utasítás hosszát
- Indexregiszter:
 - az indexelendő V objektum elemeit egymás után következő memóriarekeszekben tároljuk
 - az indexregiszter tartalmazza a megcímzendő elem sorszámát
 - $D = adr(V[o])$
 - $D + x = adr(V[x])$

2003.07.31.

Markó Tamás, PTE TTK

20

Pre- és poszt-indexelés

Indirekt címzésnél két lehetőség van az indexregiszter használatára:

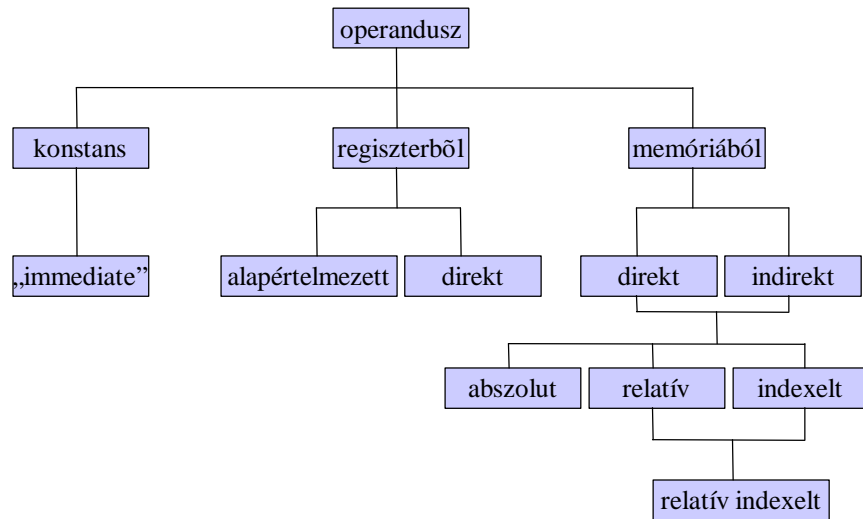
- Pre-indexelés (előindexelés):
 - $adr := MM[D + X_e]$
 - alkalmazás: ugrótáblázat
- Poszt-indexelés (utóindexelés):
 - $adr := MM[D] + X_e$
 - alkalmazás: áthelyezhető tömbök
- X_e általában egy regiszter

2003.07.31.

Markó Tamás, PTE TTK

21

A címzési módok áttekintése



2003.07.31.

Markó Tamás, PTE TTK

22

Népszerű processzorok utasításformátumai és címzési módjai

- Két példa:
 - Intel: Pentium
 - IBM/Motorola/Apple: PowerPC
- A címzésen és az utasításformátumon kívül
 - besorolás
 - blokkvázlat

2003.07.31.

Markó Tamás, PTE TTK

23

Egy korszerű CISC mikroprocesszor: a Pentium (1994)

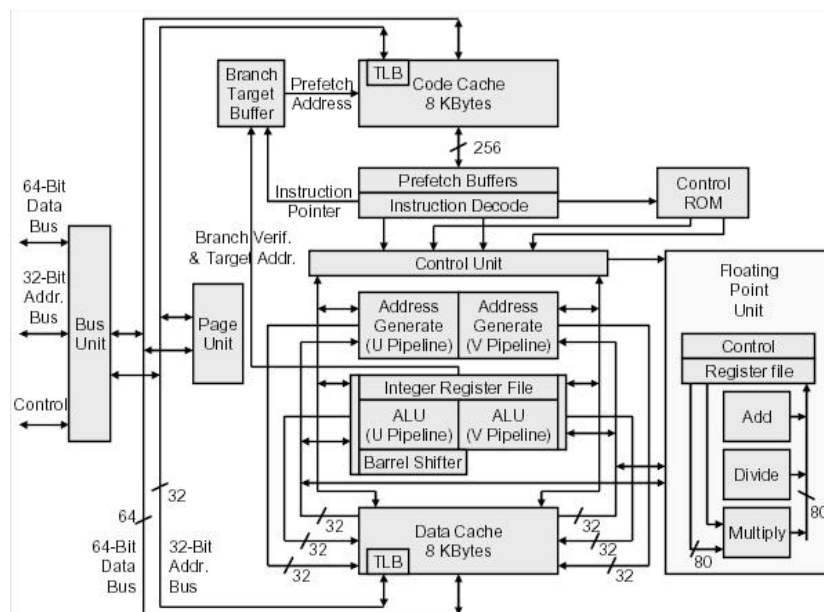
- 32/64 bites mikroprocesszor
 - 32 bites címzés
 - 64 bites adatbusz
- szuperskalár architektúra
 - két pipeline egészek feldolgozására
 - egy pipeline lebegőpontos számításokra
- elágazásjövendölés (branch prediction)
- multiprocesszoros üzemmód támogatása

2003.07.31.

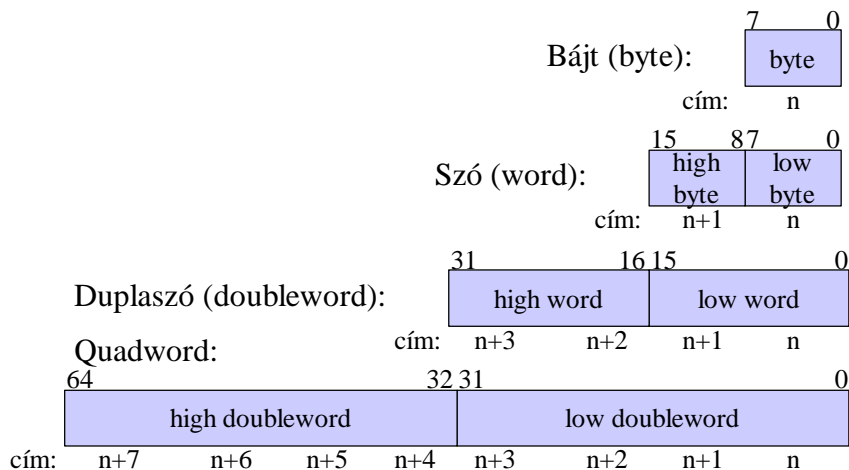
Markó Tamás, PTE TTK

24

A Pentium blokkvázlata



Adatméretek a Pentiumnál



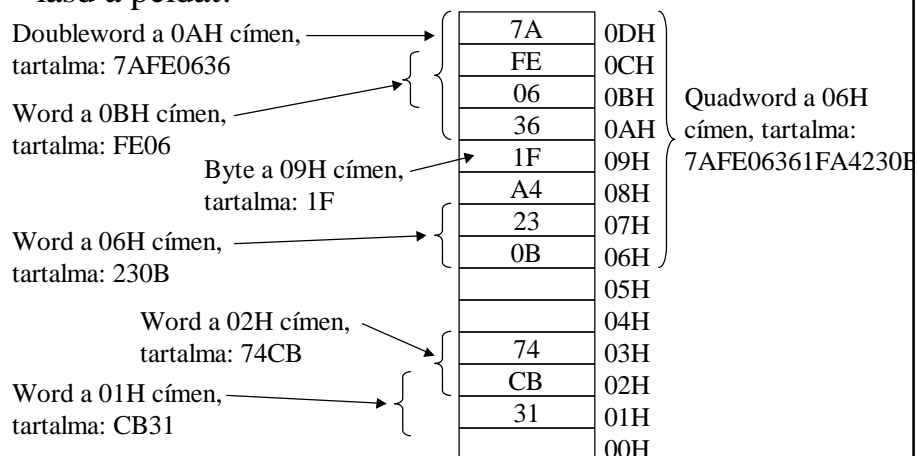
2003.07.31.

Markó Tamás, PTE TTK

26

Nincs alignment!

Nincs szükség semmiféle címigazításra (alignment),
lásd a példát:



2003.07.31.

27

A Pentium adattípusai

- Integer: 8, 16, 32 bit, kettes komplement
- Ordinal: 8, 16, 32 bit, előjel nélküli egész
- BCD integer, zónázott
- BCD integer, pakolt
- Near pointer: 32 bites cím
- Far pointer: 48 bites cím
- 16 bites szegmensválasztó (segment selector)
- 32 bites offset
- Bit field: max. 32 bit, a bájton belül akárhol kezdődhet
- Bit string: max. $2^{32}-1$ bit, tetszőleges kezdőpozíció
- Byte string: max. $2^{32}-1$ bájt
- Floating point: 32, 64, 80 bit

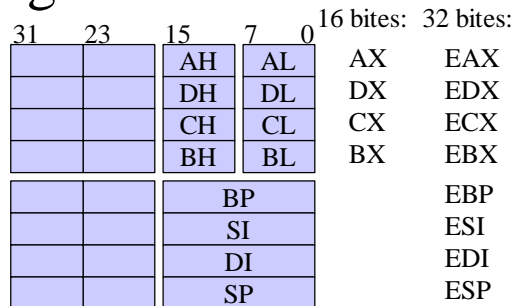
2003.07.31.

Markó Tamás, PTE TTK

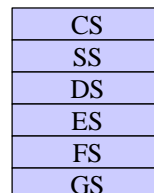
28

A Pentium regiszterkészlete

- 8 db 32 bites
általános célú
regiszter (Intel-
történelem!)



- 6 db 16 bites
szegmensregiszter



- 2 db állapot- ill.
vezérlőregiszter



2003.07.31.

29

A szegmensregiszterek

- CS (code segment): bizonyos utasítások (pl. CALL, RET, JMP) implicit módon beállítják
- SS (stack segment): a felhasználói program kezeli
- DS (data segment), ES, FS, GS: 4 adatszegmens számára
- A logikai szegmensek mutathatnak egy közös fizikai címtartományra, vagy akár különbözőre is

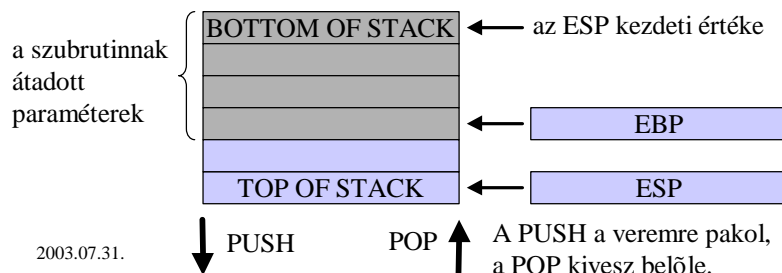
2003.07.31.

Markó Tamás, PTE TTK

30

A verem megvalósítása

- A verem az operatív tárban, max. 4 GB
- Minden időpillanatban 1 verem használható (amelyikre az SS mutat)
- A tetejére (TOS) az ESP regiszter mutat, ezt a POP és a PUSH utasítás implicit módon beállítja
- A verem védett területére az EBP regiszter mutat, ezt az ENTER és a LEAVE implicit módon beállítja



2003.07.31.

31

A Pentium utasításainak formátuma

Változó hosszúságú utasítás-formátum:

- prefix (opcionális)
- opcode (kötelező)
- regiszter-specifikáció (opcionális)
- bázis-/indexregiszter-specifikáció (opcionális)
- displacement (opcionális)
- immediate-rész (opcionális)

instruction prefix	address-size prefix	operand size prefix	segment override
0/1 bájt	0/1 bájt	0/1 bájt	0/1 bájt

opcode	MODR/M	SIB	displacement	immediate
1/2 bájt	0/1 bájt	0/1 bájt	0/1/2/4 bájt	0/1/2/4 bájt

2003.07.31.

Markó Tamás, PTE TTK

32

A prefixek

- Instruction prefix
 - repeat (string-műveleteknél: a parancs az operandusok minden bájtjára elvégzendő)
 - lock (többprocesszoros rendszereknél)
- Address size (16 bites vagy 32 bites)
- Operand size (16 bites vagy 32 bites)
- Segment override (ezzel lehet kioroszakolni másik szegmens használatát)

2003.07.31.

Markó Tamás, PTE TTK

33

Címzés

Az operandusz lehet:

- magában az utasításban (immediate)
- regiszterben:
 - 32 bit: EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP
 - 16 bit: AX, BX, CX, DX, SI, DI, SP, BP
 - 8 bit: AH, AL, BH, BL, CH, CL, DH, DL
 - Segment register
 - EFLAGS regiszter
- az operatív tárban
- egy I/O-porton

2003.07.31.

Markó Tamás, PTE TTK

34

A címtartomány kiválasztása

- modR/M bájt (közvetlenül az opcode után)
 - MOD mező: 8 regisztert és 24 indexelési módot azonosít
 - REG mező: 8 regisztert azonosít
 - R/M mező: a regisztert vagy a címezési módot azonosítja
- SIB (Scale Index Base) bájt
 - SS mező: megadja a skála-faktort (címszámítás!)
 - INDEX mező: megadja az indexregisztert
 - BASE mező: megadja a bázisregisztert

2003.07.31.

Markó Tamás, PTE TTK

35

A cím kiszámítása

- A szegmens kiválasztása (ált. a lenti táblázat szerint)
- A bázisregiszter kiválasztása
- Az indexregiszter kiválasztása
- Az indexregiszter tartalmának megszorozása a skála-faktorral
- A displacement hozzáadása

A hivatkozás típusa	A használt szegmens és regiszter	Default kiválasztási szabály
Utasítás	Code Segment, CS regiszter	Az utasítás beolvasásakor automatikusan.
Verem	Stack Segment, SS regiszter	Minden PUSH és POP. Minden memória-hivatkozás, ha ESP vagy EBP a bázisregiszter.
Helyi adat	Data Segment, DS regiszter	Minden adat-hivatkozás (kivéve a veremre hivatkozókat és a stringek tárolását).
String tárolása	E-Space Segment, ES regiszter	String-utasításoknál az eredmény tárolása.

A címkiszámítás képlete

SEGMENT + BASE + (INDEX * SCALE) + DISPLACEMENT

	<i>? EAX ?</i>	<i>? EAX ?</i>	
<i>? CS ?</i>	<i>? ECX ?</i>	<i>? ECX ?</i>	
<i>? SS ?</i>	<i>? EDX ?</i>	<i>? EDX ?</i>	<i>? 1 ?</i>
<i>? DS ?</i>	<i>? EBX ?</i>	<i>? EBX ?</i>	<i>? 2 ?</i>
<i>? ES ?</i>	<i>? ESP ?</i>	<i>? EBP ?</i>	<i>? 4 ?</i>
<i>? FS ?</i>	<i>? EBP ?</i>	<i>? ESI ?</i>	<i>? 8 ?</i>
<i>? GS ?</i>	<i>? ESI ?</i>	<i>? EDI ?</i>	<i>? 16 ?</i>
	<i>? EDI ?</i>		

semmi

8_bit_displacement

32_bit_displacement

Egy korszerű RISC mikroprocesszor: a PowerPC (1992)

- A Motorola, az IBM és az Apple közös fejlesztése
- Szuperskalár, nagyteljesítményű architektúra
 - akár 3 utasítás periódusonként
 - majdnem mindegyik utasítás egy periódus alatt
 - 3 független végrehajtó egység:
 - Branch Processing Unit (BPU), elágazásjövendölés (branch prediction)
 - 32 bites Integer Unit (IU)

38

A PowerPC

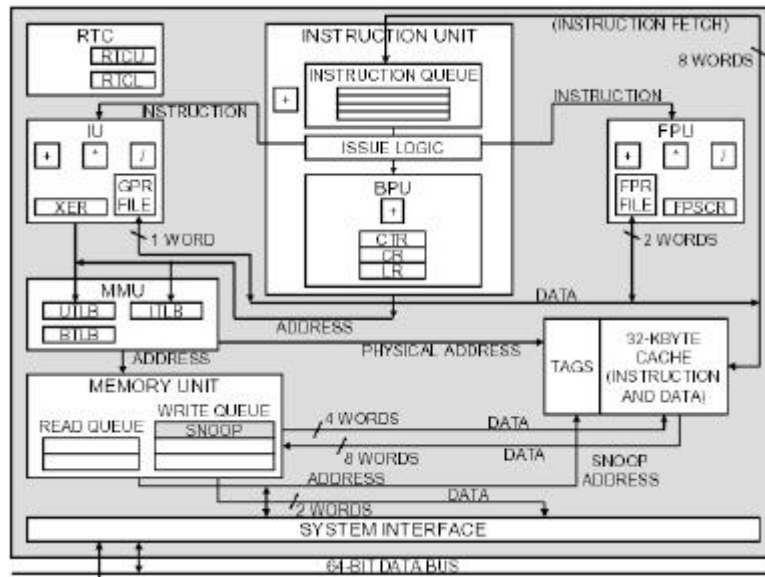
- A BPU előre figyeli a feltételregisztert (condition register, CR)
- Az IU 8 utasítást előolvas a cache-ből
- Pipeline
- 32 Kbájt cache, 8-as asszociativitás, LRU
- Memory Management Unit (MMU) read/write pufferrel
- Változatos memória-modell
 - címszámítás 4 KB-os lapokra, változó blokkméretek, 256 MB-os szegmensek
 - 256 elemű TLB
 - Inverted Page Table
 - 52 bites virtuális címek, 32 bites fizikai címek
- Rendszertámogatás
 - 64 bites külső busz
 - többprocesszoros rendszerek támogatása (snooping cache protocol (MESI))

2003.07.31.

Markó Tamás, PTE TTK

39

A PowerPC 601 blokkvázlata



A PowerPC 601 regiszterei

32 db általános

GPR0
GPR1
...
GPR31

0 31

32 db lebegőpontos

FPR0
FPR1
...
FPR31

0 63

Speciális regiszterek

SPR0	MQ Register
SPR1	XER: Integer Exception Register
SPR4	RTCU: RTC Upper Register
SPR5	RTCL: RTC Lower Register
SPR8	LR: Link Register
SPR9	CTR: Count Register

0 31

CR

0 31

Condition Register

FPSCR

0 31

Floating Point Status and Control Register

16 db szegmens

SR0
SR1
...
SR15

0 31

Supervisor szint

Machine State Register

MSR

0 31

Speciális regiszterek

SPR18
...
SPR1023

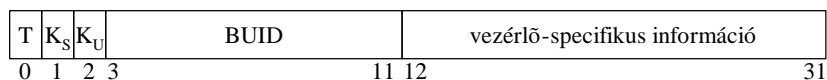
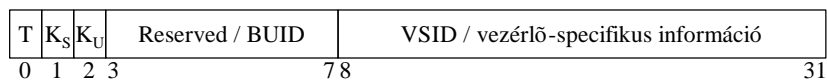
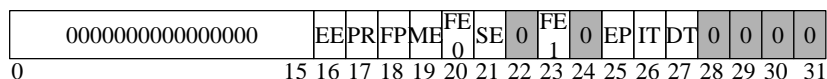
0 31

A felhasználói szintű regiszterek

- 32 db 32 bites általános célú regiszter (GPR)
- 32 db 64 bites lebegőpontos regiszter (FPR)
- Floating-point Status and Control Register (FPSCR): tartalmazza a vezérlő biteket a lebegőpontos műveletekhez
- Condition Register (CR): 32 bites, 8 db 4 bites mezőre osztva (PSW-nek felel meg a PC nélkül)
- Multiplier-Quotient (MQ): szokásos MQ aritmetikai műveletekhez, implicit módon használják
- Integer Exception Register (XER): speciális állapotbitek egész műveleteknél
- Real-time Clock (RTC):
 - RTCU: a beállítás óta eltelt másodpercek
 - RTCL: nanoszekundumok a másodpercen belül
- Link Register (LR): szubrutinhívásnál a visszatérési cím
- Count Register (CTR): ciklusok támogatása

A supervisor szintű regiszterek 1.

- Machine State Register (MSR): a processzor aktuális állapotát tárolja
- 16 db szegmensregiszter: 24 bites virtuális szegmenscímetek vagy 9 bites *bus unit ID*-t, valamint memóriavédelmi információt tartalmaz



2003.07.31.

Markó Tamás, PTE TTK

43

A supervisor szintű regiszterek 2.

- DSISR és DAR: adathozzáférési kivételek utáni teendők vezérlése
- DEC: beállítható idő múlva megszakítást okoz
- SDR1: a lapozótábla (page table) vezérlése
- SRR0, SRR1: a processzor állapotának elmentése megszakításnál
- SPRG0- SPRG3: az op.r. számára fenntartott regiszterek
- EAR, PVR: külső hozzáférések vezérlése és a processzor verziójának megadása
- BATOU-BAT3L (Block Address Translation): a szegmenstábla címzésére
- HIDO-HID15: hibakeresésre (debugging) szolgáló regiszterek

2003.07.31.

Markó Tamás, PTE TTK

44

A PowerPC adattípusai

- Egész típusok:
 - bájt (8 bit), félszó (16 bit), szó (32 bit), duplaszó (64 bit)
 - LOAD/STORE esetén ezek sorozatai is lehetnek
 - nagy endián (big endian) (alaphelyzet) vagy kis endián (little endian)
- Lebegőpontos típusok:
 - egyszeres pontosság (32 bit), dupla pontosság (64 bit)
 - IEEE 754-es szabvány vagy (beállítható) változatok

2003.07.31.

Markó Tamás, PTE TTK

45

A PowerPC címzési módjai

- Következétesen load/store architektúrájú
- Tudatosan nagyon egyszerű címzési módok
- Címzésnél egy 32 bites effective address-t (EA) számítanak ki. Ezt a Memory Management Unit 32 bites fizikai címmé transzformálja
- Két címzési mód:
 - register indirect with immediate index
 $EA = (r_A) + \text{offset}$ (az offset 0 is lehet).
 $EA = 0 + \text{offset}$ ($r_A=0$, az offset 0 is lehet)
 - register indirect with index
 $EA = (r_A) + (r_B)$
 $EA = 0 + (r_B)$

offset: eltolás

2003.07.31.

Markó Tamás, PTE TTK

46

Utasítás-formátumok

- A PowerPC fix hosszúságú, egyszerű utasításokat használ
- Alapformátumok:
 - háromcímes regiszter-parancsok (mint a MIPS-nél)
 - egy- vagy kétcímes load/store parancsok
- Speciális formátumok:

X-formátum:

OP	D	A	B	fenntartott	
0	5	10	15	20	31

D-formátum:

OP	D	A	közvetlen operandusz
0	5	10	15
			31

2003.07.31.

Markó Tamás, PTE TTK

47