Politechnika Wrocławska

Architektura Systemów Komputerowych

Wykład 8

Dr inż. Radosław Michalski
Katedra Inteligencji Obliczeniowej, Wydział Informatyki i Zarządzania
Politechnika Wrocławska

Wersja 1.1, wiosna 2018

Źródła i licencja

Najbardziej aktualna wersja tego wykładu znajduje się tu:
https://github.com/rmhere/lecture-comp-arch-org

Zawartość tego wykładu

Organizacja pamięci

Typy danych

Gałęzie i skoki

Organizacja pamięci

Bajty i słowa w MIPS

- bajt - 8 bitów (wszędzie)
- słowo - 4 bajty - 32 bity

Jednak, zależnie od ISA, długość słowa bywa różna. Zobacz ten artykuł w Wikipedii.

Organizacja pamięci

Adresowanie - wprowadzenie

- każda adresowalna komórka pamięci przechowuje osiem bitów (jeden bajt)
- każdy rejestr w MIPS to 32 bity
- w jaki sposób połączyć jedno i drugie?

Organizacja pamięci

Adresowanie

- adresy pamięci są indeksowane od 0 do X
- w architekturze 32-bitowej indeksów jest 2^{32}
- górny limit pamięci w architekturze 32-bitowej wynosi 2^{32} bajtów (4 GB)
- górny limit pamięci w architekturze 64-bitowej wynosi 2^{64} bajtów (16 EB)

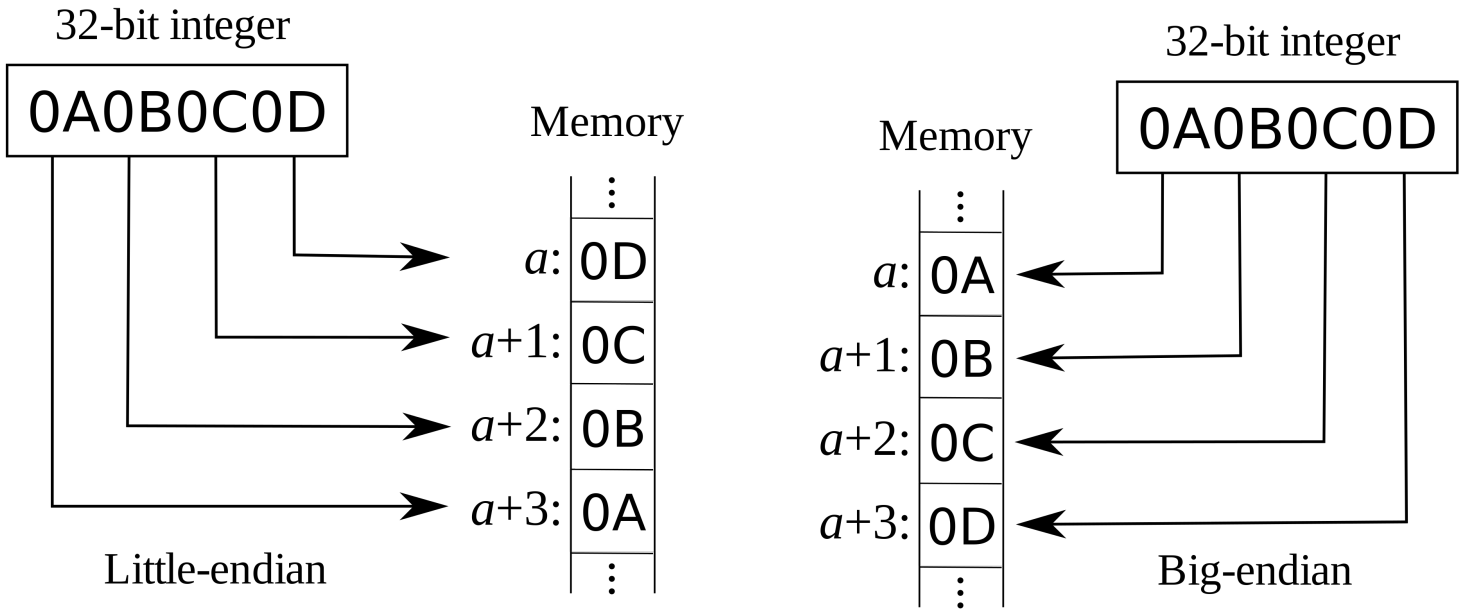
Organizacja pamięci

Endianness (kolejność bajtów) - wprowadzenie

- w archtekturze MIPS każdy rejestr przechowuje cztery komórki pamięci (1 słowo)
- w MIPS adresowanie jest dopasowane (ang. *word-aligned*), tj. adresujemy pełne słowa: 0, 4, ... X - 3
- ale w jaki sposób decyduje się o ułożeniu bajtów w słowie?

Organizacja pamięci

Big-endian vs. little-endian



R. S. Shaw, public domain

Organizacja pamięci

Problem nUxi

- załóżmy 16-bitowe słowo (np. w ISA Intel 8086)
- każdy znak kodowany jest na ośmiu bitach
- jeśli chcemy przechować ciąg znaków "Unix", potrzebne nam są dwa słowa
- kolejność bajtów w słowie determinuje jak zostaną one ułożone
- problemy z reguły pojawiają się tylko w przypadku interakcji z inną architekturą

Organizacja pamięci

Endianness - c.d.

- **Little-endian** - Intel x86 and x86-64
- **Big-endian** (network byte order) - IBM System/360, z/Architecture, IPv4, IPv6, TCP, UDP
- **Bi-endian** - ARM (v 3+), PowerPC, Alpha, SPARC V9, MIPS, PA-RISC, SuperH SH-4 and IA-64

Organizacja pamięci

Sposoby adresowania w MIPS

- adresowanie rejestrów
- podawanie stałej
- PC-relative (zostanie omówiony później)
- base displacement

Organizacja pamięci

Sposoby adresowania: rejestry, stała

Rejestry:

- argumenty w rejestrach
- add $rd, $rs, $rt

Stała:

- argument podawany wprost
- addi $rd, $rs, 5

Organizacja pamięci

Sposoby adresowania: base displacement

Base displacement:

- adres argumentu jest sumą stałej i wartości w rejestrze
- w tym przypadku rejestr traktowany jest jako baza a przesunięcie wskazywane jest jako stała
- znając rozmiary struktur danych, jest to dość wygodne podejście
- lw $rd, 100($rs)

Typy danych

Typy danych (pamięć)

.ascii str

- ciąg znaków bez terminacji

.asciiz str

- ciąg znaków z terminacją (”z” - zero), podobnie jak w C

.byte b_1**, ...,** b_n

- n bajtów w sposób ciągły

.halfword h_1**, ...,** h_n

- n półsłów w sposób ciągły

.word w_1**, ...,** w_n

- n słów w sposób ciągły

.space numBytes

- liczba bajtów w pamięci

Typy danych

Źródła i polecane materiały

- P.J. Jalics, T.S. Heines Transporting a portable operating system: UNIX to an IBM minicomputer, Communications of the ACM 26.12 (1983): 1066-1072 (artykuł naukowy)
- Summary of Addressing Modes in MIPS, University of Maryland, MD, United States (artykuł)

Gałęzie i skoki

W jaki sposób kod jest przechowywany i wykonywany

Ogólna perspektywa:

- W architekturze Princeton dane i instrukcje współdzielą pamięć
- jeśli nie jest powiedziane inaczej, procesor iteruje po pamięci w sposób sekwencyjny
- każda instrukcja znajduje się pod jakimś adresem w pamięci
- procesor ładuje słowo i stara się wykonać
- znając adres pierwszej instrukcji możesz określić pozostałe

Gałęzie i skoki

Etykiety

- etykiety `label:` wskazują na adres pamięci
- jest dla naszej wygody, nie procesora
- używane często do kontrolowania przebiegu wykonania programu

Gałęzie i skoki

Gałęzie

W jaki sposób kontrolować przebieg programu?

- sekwencyjnie (póki co)
- kolejny etap - gałęzie
- `beq $r1,$r2,Label` - jeśli równe, to rozgałęzienie
- `bne $r1,$r2,Label` - jeśli nierówne, to rozgałęzienie
- w przeciwnym razie czytaj kolejną instrukcję

Gałęzie i skoki

Skoki

- instrukcja `j` wykonuje skok do adresu
- gałąź bezwarunkowa

Gałęzie i skoki

Warunki

W jaki sposób zaimplementować instrukcję IF?

Pseudokod:

```
if t1 == t2 then t3=0
```

Język asemblera:

```
bne $t1, $t2, next
add $t3, $zero, $zero
next: (...)
```

Gałęzie i skoki

Warunki c.d.

W jaki sposób zaimplementować instrukcję IF ELSE?

Pseudokod:

```
if t1 == t2 then t3=0 else t3=2
```

Język asemblera:

```
beq $t1, $t2, nullify
addi $t3, $zero, 2
j skip
nullify: add $t3, $zero, $zero
skip: (...)
```

Gałęzie i skoki

Pętle

W jaki sposób zaimplementować pętlę FOR?

Pseudokod:

```
for i = 1 ... 3 {exec}
```

Język asemblera:

```
add $t0, $zero, $zero
addi $t1, $zero, 3
loop: beq $t0, $t1, exit
        addi $t0, $t0, 1
        exec: ...
        j loop
exit: (...)
```

Gałęzie i skoki

Mniejsze/większe niż

- póki co badaliśmy równość
- w jaki sposób porównuje się większość/mniejszość
- dwie pseudoinstrukcje: blt, bgt
- instrukcja: **SLT – ustaw gdy mniejsze**
- slt $rd, $rs, $rt
- jeśli `$rs` jest mniejsze niż `$rt`, `$rd` ustawiane jest na jeden, w przeciwnym razie $rd ustawiane jest na zero
- jak zaimplementować badanie większości?
- pseudoinstrukcje mogą mieć tę samą nazwę (tylko inne rodzaje argumentów)

Gałęzie i skoki

Program counter - PC

- specjalny rejestr przechowujący adres kolejnej instrukcji
- w przypadku sekwencyjnego wykonania programu zmienia się o słowo
- może być modyfikowany nie wprost (branching, jumping)

Gałęzie i skoki

Jump vs. jump and link

- `j` skacze
- `jal` skacze i linkuje
- `jal` kopiuje adres kolejnej instrukcji do rejestru `$ra` (rejestr 31) i wykonuje skok do adresu
- `jr $reg` skacze do wartości z danego rejestru (ustawia PC na wartość w rejestrze $reg)

Gałęzie i skoki

Branch delay slot

- potokowanie pozwala wykonywać wiele instrukcji jednocześnie
- skoki i gałęzie nie są lubiane przez potokowanie
- wynika to z faktu, że ponownie należy optymalizować wykonanie na procesorze
- branch delay slot w MIPS bezwarunkowo wykonuje kolejną instrukcję po skoku
- w jaki sposób to ominąć (jeśli trzeba): `nop`, zmiana kolejności instrukcji
- co zatem przechowuje `JAL` w rejestrze $ra? PC + 4 or PC + 8?

Gałęzie i skoki

Branch delay slot - przykład

Jak zachowa się poniższy kod?

```
j test
test: addi $t3, $t3, 2
```

Co się dzieje gdy mamy skok po skoku?

```
j test
j test1
test: addi $t3, $t3, 5
add $t3, $zero, $t3
test1: addi $t3, $t3, 2
```

Gałęzie i skoki

Elwro R-32

Wideo

Elwro - produkcja komputera R-32 (EC 1032)

Gałęzie i skoki

Źródła i polecane materiały

- MIPS32 Instruction Set Quick Reference, MIPS Technologies, Inc. (reference sheet)
- J.F. Frenzel, T.S. Heines, *MIPS Instruction Reference*, University of Idaho, ID, USA (materiały do kursu)
- M. Abrash, *"Michael Abrash's Graphics Programming Black Book"*, Redline GmbH, 1997 (książka)
- J. Pearson, *"Computer architecture"*, Uppsala University, Sweden (materiały do kursu)