Języki asemblerowe

WYKŁAD 5

Dr Krzysztof Balicki

Procesory RISC

- Architektura load/store:
  - tylko instrukcje load i store posiadają dostęp do pamięci
  - pozostałe instrukcje pobierają operandy z rejestrów i zapisują wyniki w rejestrach
- Proste instrukcje
- Kilka typów danych
- Proste tryby adresowania
- Duży zbiór rejestrów
- Operacje rejestr-rejestr

Procesory RISC

- Prosty format instrukcji
- Stała długość instrukcji
- Architektura Harvard

Procesory RISC

- Projektowanie procesorów RISC sięga początków lat 80-tych XX wieku
- Przykładowe procesory RISC:
  - SPARC, PowerPC, MIPS, Itanium
- Głównym celem wprowadzenia procesorów RISC było ograniczenia złożoności wytwarzanych procesorów

Architektura procesora MIPS

- MIPS R2000 - procesor 32 bitowy
- MIPS R4000 i późniejsze - procesory 64 bitowe
- MIPS R2000:
  - 32 rejestry ogólnego przeznaczenia
  - licznik rozkazów
  - 2 rejestry specjalnego przeznaczenia

Rejestry procesora MIPS R2000

- Rejestry ogólnego przeznaczenia:
  - $0, $1, ..., $31
  - pierwszy i ostatni rejestr zarezerwowane są dla funkcji specjalnych:
    - $0 - rejestr przechowujący wartość 0 - może być użyty w operacjach wymagających wartości 0
    - $31 - rejestr łącznikowy - używany przez instrukcje jump i link, przechowuje adres powrotu przy wywołaniu procedury

Rejestry procesora MIPS R2000

- Rejestry ogólnego przeznaczenia - przyjęta konwencja użycia:
  - $0 ($zero) - stała 0
  - $1 ($at) - zarezerwowany dla asemblera
  - $2, $3 ($v0, $v1) - wyniki zwracane przez procedury
  - $4 - $7 ($a0 - $a3) - argumenty dla procedur (pozostałe argumenty mogą być przekazywane przez stos)
  - $8 - $15 ($t0 - $t7) - rejestry tymczasowe nie zachowywane przy wywołaniu procedur

Rejestry procesora MIPS R2000

- Rejestry ogólnego przeznaczenia - przyjęta konwencja użycia (cd.):
  - $16 - $23 ($s0 - $s7) - rejestry tymczasowe zachowywane przy wywołaniu procedur
  - $24 - $25 ($t8 - $t9) - rejestry tymczasowe nie zachowywane przy wywołaniu procedur
  - $26 - $27 ($k0, $k1) - zarezerwowane dla jądra systemu operacyjnego
  - $28 ($gp) - wskaźnik przestrzeni globalnej w pamięci (gdzie przechowywane są stałe i zmienne globalne)

Rejestry procesora MIPS R2000

- Rejestry ogólnego przeznaczenia - przyjęta konwencja użycia (cd.):
  - $29 ($sp) - wskaźnik stosu
  - $30 ($fp) - wskaźnik ramki
  - $31 ($ra) - adres powrotu - używany przez procedury

Rejestry procesora MIPS R2000

- Przyjęta konwencja użycia rejestrów ogólnego przeznaczenia nie jest obowiązkowa

Rejestry procesora MIPS R2000

- Rejestry specjalnego przeznaczenia:
  - HI, LO - przechowują wyniki wykonania instrukcji mnożenia i dzielenia całkowitego:
    - przy mnożeniu 64 bitowy wynik: starsze 32 bity w rejestrze HI, młodsze 32 bity w rejestrze LO)
    - przy dzieleniu: część całkowita w rejestrze LO, reszta z dzielenia w rejestrze HI

Tryby adresowania procesora MIPS R2000

- Tryb adresowania udostępniany przez procesor:

Adres efektywny

=

Adres bazowy (w rejestrze Rx)

+

Przesunięcie (16 bitowe)

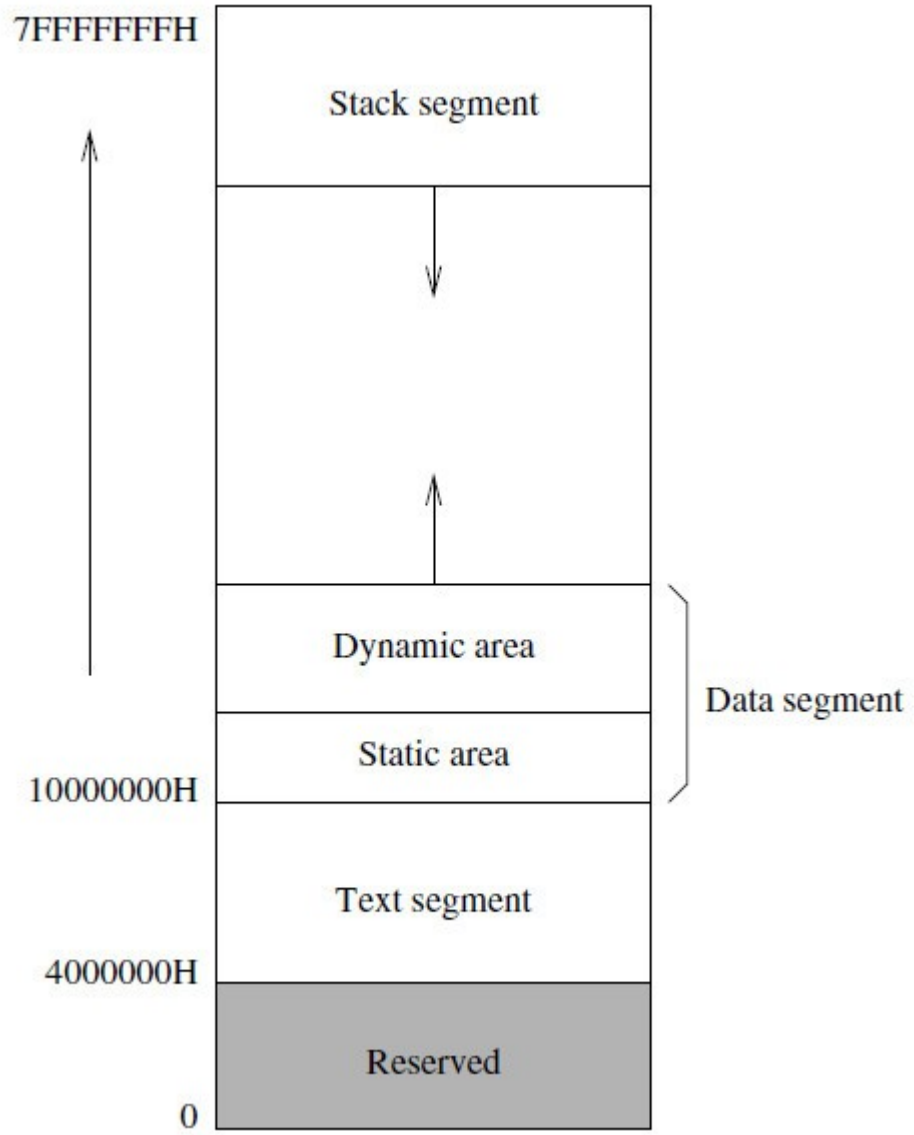
Tryby adresowania procesora MIPS R2000

- Maszyna wirtualna wspierająca asembler udostępnia kilka trybów adresowania:
  - **(Rx)** - zwartość rejestru Rx
  - **wart** - wartość bezpośrednia
  - **wart(Rx)** - wartość bezpośrednia plus zawartość rejestru Rx
  - **symbol** - adres symbolu
  - **symbol +/- wart** - adres symbolu plus wartość bezpośrednia
  - **symbol +/- wart(Rx)** - adres symbolu plus wartość bezpośrednia plus zawartość rejestru Rx

Wykorzystanie pamięci przez procesor MIPS R2000

- Przestrzeń adresowa programu podzielona jest na trzy części:
  - kod
  - dane
  - stos

Wykorzystanie pamięci przez procesor MIPS R2000



Źródło: Sivarama P. Dandamudi: Introduction to Assembly Language Programming. For Pentium and RISC Processors. Springer-Verlag, 2005.

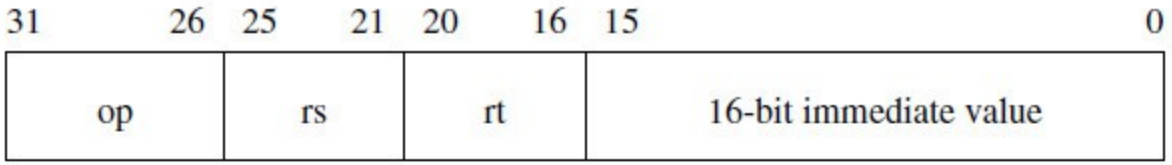
Instrukcje MIPS R2000

- Zbiór instrukcji można podzielić na dwie części:
  - *instrukcje* - instrukcje procesora
  - *pseudoinstrukcje* - dostarczane przez asembler w celu ułatwienia programowania, tłumaczone na sekwencje instrukcji procesora

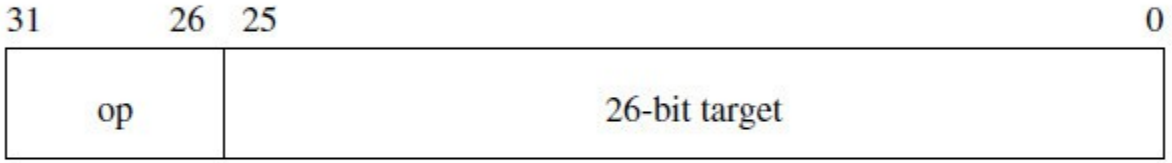
Instrukcje MIPS R2000

- Długość każdej instrukcji jest taka sama: 32 bity
- Formaty instrukcji:
  - format bezpośredni
  - format skoku
  - format rejestrowy

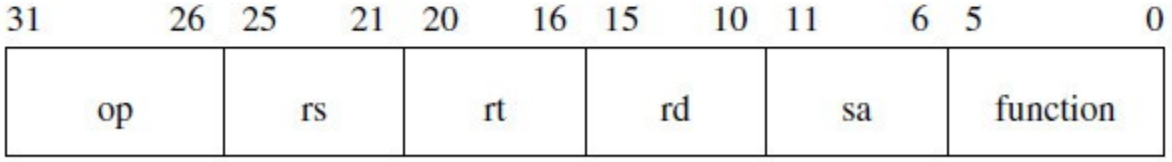
Instrukcje MIPS R2000



I-Type (Immediate)



J-Type (Jump)



R-Type (Register)

Źródło: Sivarama P. Dandamudi: Introduction to Assembly Language Programming. For Pentium and RISC Processors. Springer-Verlag, 2005.

Instrukcje MIPS R2000

- Wybrane instrukcje przesłania:
  - przesłanie z pamięci do rejestru
    - **`lb RPrzezn,Adres`** - załadowanie wartości z pamięci do najmłodszego bajtu rejestru, wartość traktowana jest jak liczba ze znakiem, w starszych bajtach rejestru powielany jest bit znaku
    - **`lbu RPrzezn,Adres`** - załadowanie wartości z pamięci do najmłodszego bajtu rejestru, wartość traktowana jest jak liczba bez znaku, starsze bajty rejestru są zerowane

Instrukcje MIPS R2000

- Wybrane instrukcje przesłania:
  - przesłanie z pamięci do rejestru (cd.)
    - **`lh RPrzezn,Adres`** - analogicznie jak poprzednio, przesłanie półsłowa (2 bajty) ze znakiem
    - **`lhu RPrzezn,Adres`** - analogicznie jak poprzednio, przesłanie półsłowa (2 bajty) bez znaku
    - **`lw RPrzezn,Adres`** - analogicznie jak poprzednio, przesłanie słowa (4 bajty)

Instrukcje MIPS R2000

- Wybrane instrukcje przesłania:
  - przesłanie z rejestru do pamięci
    - **`sb RZrodlowy,Adres`** - przesłanie najmłodszego bajtu rejestru do pamięci
    - **`sh RZrodlowy,Adres`** - przesłanie najmłodszych dwóch bajtów (półsłowa) rejestru do pamięci
    - **`sw RZrodlowy,Adres`** - przesłanie całego rejestru (słowa) do pamięci

Instrukcje MIPS R2000

- Wybrane instrukcje przesłania:
  - przesłanie z rejestru specjalnego do rejestru ogólnego przeznaczenia
    - **`mfhi RPrzezn`** - przesłanie z rejestru HI
    - **`mflo RPrzezn`** - przesłanie z rejestru LO
  - przesłanie z rejestru ogólnego przeznaczenia do rejestru specjalnego
    - **`mthi RPrzezn`** - przesłanie do rejestru HI
    - **`mtlo RPrzezn`** - przesłanie do rejestru LO

Instrukcje MIPS R2000

- Wybrane pseudoinstrukcje przesłania:
  - przesłanie pomiędzy rejestrami
    - **`mov RPrzezn, RZrodlowy`**
  - przesłanie adresu zmiennej do rejestru
    - **`la RPrzezn, Adres`**
  - przesłanie wartości bezpośredniej do rejestru
    - **`li RPrzezn, Wartosc`**

Instrukcje MIPS R2000

- Wybrane instrukcje arytmetyczne:
  - dodawanie (liczby ze znakiem)
    - **`add RPrzezn,ROperand1,ROperand2`**
    - **`addi RPrzezn,ROperand1,Wartosc`**
  - odejmowanie (liczby ze znakiem)
    - **`sub RPrzezn,ROperand1,ROperand2`**
  - mnożenie (liczby ze znakiem, wynik zwracany jest w rejestrach HI i LO)
    - **`mult ROperand1,ROperand2`**
  - dzielenie (liczby ze znakiem, wynik zwracany jest w rejestrach HI i LO)
    - **`div ROperand1,ROperand2`**

Instrukcje MIPS R2000

- Wybrane instrukcje logiczne:
  - iloczyn logiczny
    - **`and RPrzezn,ROperand1,ROperand2`**
  - suma logiczna
    - **`or RPrzezn,ROperand1,ROperand2`**
  - negacja symy logicznej
    - **`nor RPrzezn,ROperand1,ROperand2`**
  - różnica symetryczna
    - **`xor RPrzezn,ROperand1,ROperand2`**

Instrukcje MIPS R2000

- Wybrane instrukcje przesuwania:
  - przesunięcie w lewo o podaną liczbę bitów
    - **`sll RPrzezn,RZrodlowy,Liczba`**
  - przesunięcie w prawo o podaną liczbę bitów
    - **`srl RPrzezn,RZrodlowy,Liczba`**
  - rotacja w lewo o podaną liczbę bitów
    - **`rol RPrzezn,RZrodlowy,Liczba`**
  - rotacja w prawo o podaną liczbę bitów
    - **`ror RPrzezn,RZrodlowy,Liczba`**

Instrukcje MIPS R2000

- Wybrane pseudoinstrukcje porównania:
  - mniejszy lub równy (porównywane są liczby ze znakiem, wynik ustawiany jest w rejestrze przeznaczenia, jeden lub zero)
    - **`slt RPrzezn,ROperand1,ROperand2`**
    - **`slti RPrzezn,ROperand1,Wartosc`**
  - analogicznie jak poprzednio działają instrukcje porównań ze liczb ze znakiem
    - `sle, seq, sne, sgt, sge`

  oraz porównań liczb bez znaku
    - `sltu, sleu, sgtu, sgeu`

Instrukcje MIPS R2000

- Wybrane instrukcje skoków:
  - instrukcja skoku (adres przeznaczenia jest 26 bitowy)
    - **`j AdresPrzezn`**
  - instrukcja rozgałęzienia bezwarunkowego (adres przeznaczenia jest 16 bitowy)
    - **`b AdresPrzezn`**
  - instrukcja rozgałęzienia warunkowego (jeśli równe)
    - **`beq ROperand1,ROperand2,AdresPrzezn`**

Instrukcje MIPS R2000

- Wybrane instrukcje skoków (cd.):
  - instrukcja rozgałęzienia warunkowego (jeśli zero)
    - **`beqz ROperand1,AdresPrzezn`**
  - instrukcja rozgałęzienia warunkowego (jeśli różne)
    - **`bne ROperand1,ROperand2,AdresPrzezn`**

Instrukcje MIPS R2000

- Wybrane instrukcje skoków (cd.):
  - analogicznie jak poprzednio działają instrukcje rozgałęzienia warunkowego z porównywaniem liczb ze znakiem
    - `blt, bgt, ble, bge`

  z porównywaniem liczb bez znaku
    - `bltu, bgtu, bleu, bgeu`

  oraz z porównywaniem z zerem
    - `bnez, bltz, bgtz, blez, bgez`

Symulator SPIM

- Operacje wejścia/wyjścia
  - kod wywołania systemowego umieszczany jest w rejestrze $v0
  - argumenty umieszczane są w rejestrach $a0, $a1 (lub $f12 - dla liczb zmiennoprzecinkowych)
  - wywołanie systemowe odbywa się za pomocą

    `syscall`
  - zwracana wartość umieszczana jest w rejestrze $v0 (lub $f0 - dla liczb zmiennoprzecinkowych)

Symulator SPIM

- Operacje wejścia/wyjścia
  - wybrane operacje
    - **print_int** - kod: 1, argument w $a0
    - **print_float** - kod: 2, argument w $f12
    - **print_double** - kod: 3, argument w $f12
    - **print_string** - kod: 4, argument (adres łańcucha) w $a0
    - **read_int** - kod: 5, wartość zwracana w $v0
    - **read_float** - kod: 6, wartość zwracana w $f0
    - **read_double** - kod: 7, wartość zwracana w $f0
    - **read_string** - kod: 8, argumenty w $a0 (adres bufora) i $a1 (rozmiar bufora)
    - **exit** - kod: 10

Symulator SPIM

- Deklaracje segmentów
  - segment kodu

    .TEXT
  - segment danych

    .DATA

Symulator SPIM

- Alokacja pamięci
  - dla łańcuchów znaków

```
.ASCII ”teskt”
```

  - dla łańcuchów znaków (łańcuch zakończony jest wartością NULL)

```
.ASCIIZ ”teskt”
```

  - dla n bajtów

```
.SPACE n
```

  - dla ciągu n wartości dwubajtowych (półsłów)

```
.HALF h1,h2,...,hn
```

Symulator SPIM

- Alokacja pamięci (cd.)
  - dla ciągu n wartości czterobajtowych (słów)

```
.WRITE h1,h2,...,hn
```

  - dla n liczb zmiennoprzecinkowych (pojedynczej precyzji)

```
.FLOAT f1,f2,...,fn
```

  - dla n liczb zmiennoprzecinkowych (podwójnej precyzji)

```
.DOUBLE d1,d2,...,dn
```

Procedury

- Wywołanie procedury
  ```
  jal NazwaProcedury
  ```
- Powrót z procedury
  ```
  jr $ra
  ```
- Parametry mogą być przekazywane przez rejestry lub przez stos

Obsługa stosu

- Rezerwacja miejsca na stosie (n bajtów)
  ```
  sub $sp,$sp,n
  ```
- Wysłanie zawartości rejestru na stos od k-tego elementu
  ```
  sw RZrodlowy,k($sp)
  ```
- Pobranie wartości ze stosu od k-tego elementu
  ```
  lw RPrzezn,k($sp)
  ```