

Linguagem de montagem

1. Introdução, simulador, operações aritméticas, registradores, estrutura de um programa e syscalls

Prof. John L. Gardenghi
Adaptado dos slides do livro

Conjunto de instruções

- Quais instruções o computador é capaz de executar (lógica digital)
- Cada computador tem seu próprio conjunto de instruções
 - Com vários aspectos em comum
- Os primeiros computadores tinham conjuntos de instruções bem simples
 - Implementação simplificada
- Atualmente, muitos computadores ainda tem conjunto de instruções simples

Conjunto de instruções MIPS

- Usaremos ao longo da disciplina
- Conjunto de instruções RISC
- Stanford MIPS comercializado pela MIPS Technologies (www.mips.com)
- Muito usado no mercado de dispositivos embarcados
 - Eletrônicos, equipamentos de rede e armazenamento, cameras, impressoras, ...

Simulador MIPS

■ SPIM

- <http://spimsimulator.sourceforge.net/>
- Programe no editor de sua preferência
- Rode o programa pelo terminal

Operações aritméticas

- Soma e subtração, dois operandos
 - Dois operandos e um resultado

add a, b, c # a gets b + c
- Todas as operações aritméticas possuem esse formato
- *Princípio de Design 1: Simplicidade favorece regularidade*
 - Torna a implementação simples
 - Favorece desempenho por baixo custo

Exemplo de aritmética

- Código C:

```
f = (g + h) - (i + j);
```

- Código MIPS compilado:

```
add t0, g, h    # temp t0 = g + h
add t1, i, j    # temp t1 = i + j
sub f, t0, t1   # f = t0 - t1
```

Registradores

- Instruções aritméticas lidam com registradores
- MIPS possui 32 registradores de 32 bits
 - Usados para dados frequentemente acessados
 - Numerados de 00 a 31
 - Um dado de 32 bits se chama **palavra**
- *Princípio de Design 2: Menor é mais rápido*
 - Ao contrário da memória principal, que possui milhões de posições

Registradores

Notação	Número	Descrição
\$zero	0	A constante zero
\$at	1	Reservado para o <i>assembler</i>
\$v0-\$v1	2-3	Valores para resultados e avaliação de expressões
\$a0-\$a3	4-7	Argumento
\$t0-\$t7	8-15	Temporários (não preservados entre chamadas)
\$s0-\$s7	16-23	Salvos (preservados entre chamadas)
\$t8-\$t9	24-25	Mais temporários
\$k0-\$k1	26-27	Reservado para o <i>kernel</i> do SO
\$gp	28	ponteiro global
\$sp	29	ponteiro para pilha
\$fp	30	ponteiro para frame
\$ra	31	endereço de retorno

Convenção de uso dos registradores MIPS.

Exemplo com registradores

- Código C:

```
f = (g + h) - (i + j);
```

- f, ..., j = \$s0, ..., \$s4

- Código MIPS compilado:

```
add $t0, $s1, $s2
```

```
add $t1, $s3, $s4
```

```
sub $s0, $t0, $t1
```

Escrevendo em MIPS

- A estrutura de um programa é a seguinte:

```
.data
    # declarações de variáveis e constantes
.text
    # código MIPS
main:
    # programa principal
```

Escrevendo em MIPS

- Tipos de dados:
 - `.word` w_1, \dots, w_n : dado de 32 bits
 - `.half` h_1, \dots, h_n : dado de 16 bits
 - `.byte` b_1, \dots, b_n : dado de 8 bits
 - `.ascii str`: cadeia de caracteres
 - `.asciiz str`: terminando com o caracter nulo

Pseudoinstruções

- Pseudoinstruções são instruções que não fazem parte do conjunto de instruções
 - São combinações de instruções que fazem parte
- Algumas pseudoinstruções úteis:
 - `li reg, const`
 - Carrega o valor de uma constante num registrador
 - `la reg, label`
 - Carrega o endereço de memória no registrador
 - `move reg1, reg2`
 - Copia o conteúdo de `reg2` em `reg1`

System calls

- Executar tarefas que dependam do sistema operacional
 - Alocar memória
 - Lidar com entrada e saída
- Para fazer uma syscall, é necessário
 1. Carregar o código da syscall no reg. \$v0
 2. Carregar os argumentos necessários em \$a0-\$a3
 3. Fazer a chamada usando a instrução `syscall`

System calls

Serviço	Cod.	Argumentos	Resultado
imprimir inteiro	1	\$a0 = inteiro	n. a.
imprimir uma string	4	\$a0 = endereço da string	n. a.
ler um inteiro	5	n. a.	\$v0 = valor lido
ler uma string	8	\$a0 = endereço da string \$a1 = qtde. de caracteres + 1	n. a.
alocar memória	9	\$a0 = número de bytes	\$v0 = endereço do bloco
encerrar o programa	10	n. a.	n. a.
imprimir um caracter	11	\$a0 = inteiro (ASCII)	n. a.
ler um caracter	12	n. a.	\$v0 = caracter lido

Observação: as operações 2 e 3, 6 e 7 são operações com números de ponto flutuante, que veremos adiante.