Linguagem de Montagem - 13/06

Conjunto de instruções é o conjunto de operações que um computador é capaz de executar (ISA).

- Cada processador tem seu próprio conjunto (Assembly).
- Nessa disciplina, vamos trabalhar com o Assembly MIPS.

MIPS

- Conjunto RISC(reduced instruction set computer).
- Utilizado em sistemas embarcados.

Simulador

- SPIM.
- sudo apt install spim (instalação Ubuntu).
- Para rodar um código em Assembly MIPS:
  1. Programe no editor de sua preferência
  2. Salvar o código com a extensão .spim.
  3. Execute pelo terminal: spim -f codigo.spim

OBS: vamos adotar a 5ª edição do livro.

Operações aritméticas

Executam operações aritméticas e possuem um padrão de formação:

- [mnemônico] res, op1, op2

Realiza a operação entre op1 e op2 e salva o resultado em res.

Operação	Mnemônico
Adição	add
Subtração	sub
Multiplicação	mul
Divisão	div

Exemplo:

- add a, b, c # a = b + c

  Princípio de Design 1:

  Simplicidade favorece a regularidade.

Registrador

É uma unidade de memória inteira do processador.

- O MIPS possui 32 registradores de 32 bits cada, essa arquitetura é chamada de MIPS de 32 bits.
- Observação: o tamanho de uma variável inteira é de 4 bytes (32 bits), é o que cabe em um registrador.
- Todas as arquiteturas seguem o padrão de tamanho dos registradores: ou 32 ou 64 bits. Caso a arquitetura seja de 64 bits, um número inteiro ocuparia metade do registrador.
- Operações aritméticas operam **apenas** sobre registradores.
- São numerados de 0 a 31.
- Um dado de 32 bits se chama **palavra**.

Por que há apenas 32 registradores?

> Princípio de Design 2:
>
> Menor é mais rápido.

Tabela dos registradores

Registrador(es)	Nome	Descrição
0	$zero	A constante zero.
1	*at*|*Reservadoparaoassembler*||2–3|$*v*0 – $*v*1\|*Resultadosdefunção.*\|\|4–7\|$*a*0–a3	Argumentos.
8-15	$t0-*t*7\|*Temporários.*\|\|16–23\|$*s*0–s7	Salvos.
24-25	$t8-*t*9\|*Maistemporários.*\|\|26–27\|$*k*0–k1	Reservados pelo SO.
28	$gp	Ponteiro global.
29	$sp	Ponteiro para pilha.
30	$fp	Ponteiro do frame de ativação.
31	$ra	Endereço de retorno.

Exemplo: Código em C:

```
f = (g+h)-(i+j); // f, g, h, i, j -> $s0, $s1, $s2, $s3, $s4.
```

MIPS:

```
add $t0, $s1, $s2 # $t0 = $s1+$s2
add $t1, $s3, $s4 # $t1 = $s3+$s4
```

```
sub $s0, $t0, $t1 # $s0 = $t0-$t1
```

Tipos de dados

Declaração	Descrição
.word <nome1>, <nome2>, ..., <nomeN>	Dados de 32 bits.
.byte <nome1>, <nome2>, ..., <nomeN>	Dados de 8 bits (1 byte).
.ascii <nome>	Cadeia de caracteres.
.asciiz <nome>	Cadeia de caracteres terminada em \0.

Pseudoinstruções

São instruções que não fazem parte da ISA, o assembler faz a tradução.

Algumas pseudoinstruções úteis:

- li reg1, const: o registrador recebe o valor de uma constante (Ex: li $t0, 10)
  - Tradução: addi $t0, $zero, 10
- la reg1, label: o registrador recebe o endereço apontado pelo rótulo (label).
  - Tradução: ?
- move reg1, reg2: copia o conteúdo do registrador 2 para o registrador 1.
  - Tradução: add reg1, reg2, $zero

Estrutura de um programa em MIPS

O programa é dividido em 2 seções:

- **.data**: onde são feitas as declarações de variáveis, ou seja, espaços da memória que não são registradores (alocação de memória).
- **.text**: espaço dedicado para o código Assembly. O código deve possuir um rótulo *main*.

Exemplo:

```
.data:

.text:
    main:
```