Linguagem de montagem

2. Instruções de acesso à memória e imediatas

Prof. John L. Gardenghi
Adaptado dos slides do livro

Acesso à memória

- A memória principal é usada para armazenar os dados que não cabem nos registradores
  - Arrays, structures, dados dinâmicos
- Para executar operações aritméticas:
  - Carrega o dado em um registrador
  - Armazena o resultado do registrador na memória
- A memória é endereçada por *bytes*
  - Cada endereço é um dado de 8-bits (1 byte)
- Palavras são alinhadas na memória
  - O endereço inicial deve ser múltiplo de 4
  - Restrição de alinhamento

Instruções de acesso à memória

- `lw reg, offset(base)`
  - `lw`: *load word,* `lh`*: load halfword,* `lb`*: load byte*
  - carrega o dado no registrador `reg` do endereço `base + offset`
- `sw reg, offset(base)`
  - `sw`: *store word,* `sh`*: store halfword,* `sb`*: store byte*
  - salva o dado do registrador `reg` no endereço `base + offset`
- *Syscall* código 9: alocação de memória
  - `$a0` recebe o tamanho em bytes
  - `$v0` retorna o endereço base
- **Atenção!**
  - `base` é um registrador
  - `offset` é um número

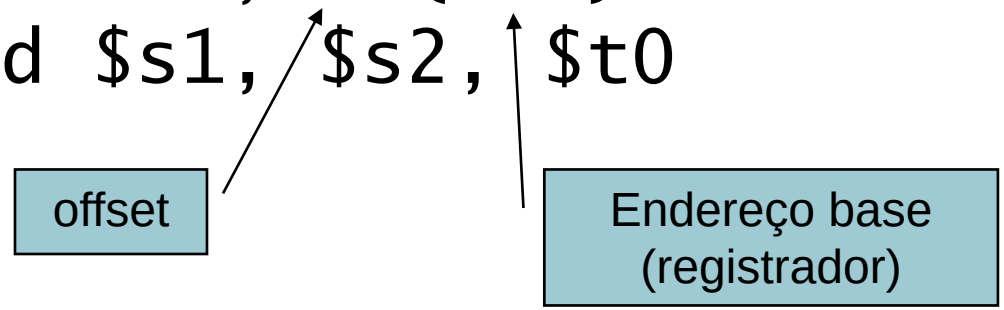
Acesso à memória – Exemplo 1

- Código C:

```
g = h + A[8];
```

  - g em $s1, h em $s2, endereço base de A em $s3
- Código MIPS compilado:
  - Índice 8 do vetor requer um *offset* de 32 *bytes*
    - 4 bytes por palavra

```
lw  $t0, 32($s3)
add $s1, $s2, $t0
```



Acesso à memória – Exemplo 2

- Código C:

```
A[12] = h + A[8];
```

  - h em $s2, endereço base de A em $s3

- Compiled MIPS code:
  - Índice 8 do vetor requer um *offset* de 32

```
lw  $t0, 32($s3)    # load word
add $t0, $s2, $t0
sw  $t0, 48($s3)    # store word
```

- Atenção com o uso de `lw` e `sw`!

Registradores vs. Memória

- Registradores possuem acesso mais rápido que memória
- Operar na memória requer carregar e salvar o dado
  - Mais instruções a serem executadas
- Um compilador deve usar os registradores o máximo possível
  - A memória deve ser acessada apenas para variáveis menos utilizadas
  - Otimização de registradores é importante!

Instruções imediatas

- Dado constante especificado na instrução

```
addi $s3, $s3, 4
```

- Não há instrução imediata de subtração
  - Basta usar uma constante negativa

```
addi $s2, $s1, -1
```

- *Princípio de Design 3:* Torne o caso comum mais rápido
  - O uso de constantes pequenas é muito comum
  - Instruções imediatas evitam uma instrução `lw` num registrador