AULA Nº 06
ORGANIZAÇÃO DE COMPUTADORES

Conjunto de Instruções do MIPS

Registradores como operandos

Princípio de Projeto 2: menor é mais rápido

Instruções aritméticas usam registradores como operandos
MIPS tem um conjunto de 32 registradores de 32 bits
Numerados de 0 a 31
Dados de 32 bits são chamados *word*

MIPS

Exemplo didático em livros
Stanford MIPS comercializado por MIPS Technologies (www.mips.com).
Ampla fatia de mercado de núcleos embarcados.
Aplicações em eletrônica de consumo, equipamentos de rede, armazenamento, impressoras, …
Típico de conjunto de instruções modernos.

Registradores

$t0, $t1, …, $t9 → valores temporários
$s0, $s1, …, $s7 → variáveis a serem salvas

Registrador 0 ($zero) do MIPS é a constante 0
Não pode ser sobrescrita
Ex., mover conteúdo de registradores

```
add $t2, $s1, $zero
```

Operações Aritméticas

Princípio de Projeto 1: simplicidade favorece a regularidade
Regularidade torna implementação mais simples
Simplicidade permite desempenho melhor a custo menor
Todas operações aritméticas tem a mesma forma
Três operandos: duas origens e um destino

```
add a, b, c  → a = b + c
```

Memória

Endereço por byte
Palavras (*words*) alinhadas na memória
Endereços precisam ser múltiplos de 4
É *Big Endian*: byte mais significativo no endereço menos significativo da palavra

Big Endian		Little Endian	
Base Address+0	Byte 3	Base Address+0	Byte 0
Base Address+1	Byte 2	Base Address+1	Byte 1
Base Address+2	Byte 1	Base Address+2	Byte 2
Base Address+3	Byte 0	Base Address+3	Byte 3

Operandos em memória

Código em C:
g = h + A[8];
g em $s1, h em $s2, endereço base de A em $s3

Código compilado MIPS:
Index 8 requer offset de 32 (4 bytes por palavra)
lw $t0, 32($s3) # load word
add $s1, $s2, $t0

Instruções tipo R

Instruções com registradores como operandos

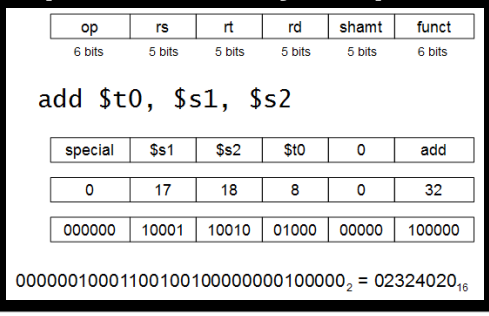
op: código da operação (opcode)
rs: registrador com primeiro operando origem
rt: registrador com segundo operando origem
rd: número do registrador destino
shamt: quantia de deslocamento
funct: código de função (estende opcode)

Operandos imediatos

Princípio de Projeto 3: Faça o caso comum rápido
Constantes pequenas são comuns
Operandos imediatos evitam instrução de load

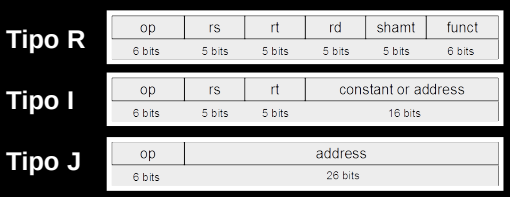
Constante especificada na própria instrução
addi $s3, $s3, 4
Não existe instrução de subtração com imediato
Use constante negativa
addi $s2, $s1, -1

Exemplo de instrução tipo R

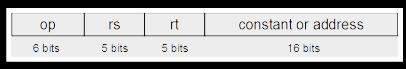


MIPS: Formato de instruções

32 bits cada instrução (palavra = *word*)
3 tipos de instrução:



Instruções tipo I



Instruções aritméticas com imediato e load/store
Constante: –2^15 to +2^15 – 1
Endereço: somado ao endereço base em rs
Princípio de Projeto 4: Um bom projeto implica em bons compromissos
Formatos diferentes complicam a decodificação, mas permitem instruções uniformes de 32 bits

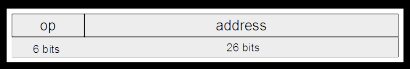
Desvios condicionais (tipo I)

Desvia para instrução com rótulo se condição é verdade;
Caso contrário, continue sequencialmente.

beq rs, rt, L1
if (rs == rt) desvie para instrução L1;

bne rs, rt, L1
if (rs != rt) desvie para instrução L1;

Instruções tipo J



Instruções de *jump* (salto)
j, jal
Codifica endereço completo na Instrução

Exemplo de endereçamento no jump:
Endereço alvo = PC31…28 : (endereço × 4)

Referências

Capítulo 2 - "Organização e Projeto de Computadores – A Interface Hardware/Software", David A. Patterson & John L. Hennessy, Campus, 4 edição, 2013.

Capítulo 10 - *Arquitetura e organização de computadores*, William Stallings, Editora Pearson, 8ª edição, 2010.