AULA Nº 07
ORGANIZAÇÃO DE COMPUTADORES

Conjunto de Instruções do MIPS
Parte 2

Compilando while (loop)

Código em C:

```
while (save[i] == k) i += 1;
```

Compiled MIPS code:

```
Loop: sll  $t1, $s3, 2
      add  $t1, $t1, $s6
      lw   $t0, 0($t1)
      bne  $t0, $s5, Exit
      addi $s3, $s3, 1
      j    Loop
Exit: …
```

i em $s3, k em $s5, endereço de save em $s6

Conjunto de instruções do MIPS

MIPS Reference Card

Repertório de operações
ULA: add, addi, and, andi, ...
Transferência de dados: load, store
Controle: bne, beq, j, jal, jr
Tipo e tamanho dos operandos (dados)
Byte, half word, word, unsigned
Formato de instrução: R, I e J

Operadores condicionais

Resultado igual a 1 se condição for verdade, e 0 caso falsa.

slt rd, rs, rt
if (rs < rt) rd = 1; else rd = 0;
slti rt, rs, constant
if (rs < constant) rt = 1; else rt = 0;
Pode ser usado combinado a beq, bne

```
slt $t0, $s1, $s2  # if ($s1 < $s2)
   bne $t0, $zero, L  #   branch to L
```

Compilando if

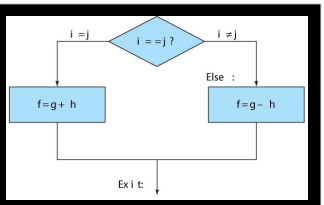
Código em C:

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

f, g, … in $s0, $s1, …

Código MIPS compilado:

```
      bne $s3, $s4, Else
        add $s0, $s1, $s2
        j   Exit
Else: sub $s0, $s1, $s2
Exit: …
```



Montador calcula endereços

Comparação com e sem sinal

Comparação com sinal (signed): slt, slti
Comparação sem sinal (unsigned): sltu, sltui
Exemplo:
$s0 = 1111 1111 1111 1111 1111 1111 1111 1111
$s1 = 0000 0000 0000 0000 0000 0000 0000 0001
slt $t0, $s0, $s1 # signed
–1 < +1 ⇒ $t0 = 1
sltu $t0, $s0, $s1 # unsigned
+4,294,967,295 > +1 ⇒ $t0 = 0

Chamadas de procedimento

Passos necessários:

1. Colocar parâmetros nos registradores ($a0 - $a3)
2. Transferir controle (jal)
3. Salvar registradores $s que usar na pilha
4. Executar operações do procedimento
5. Colocar resultados nos registradores de retorno ($v0, $v1)
6. Restaurar registradores $s e retornar (jr $ra)

Sincronização

Considere 2 processos **compartilhando área na memória**: P1 escreve, e P2 lê
Resultado final irá depender da ordem de acesso de escrita e leitura!
Requer suporte do hardware para operações atômicas de leitura/escrita
Nenhum outro acesso ao local permitido entre leitura e escrita

Exemplo

Código em C:

```
int folha (int g, h, i, j)
{ int f;
  f = (g + h) - (i + j);
  return f;
}
```

g, …, j em $a0, …, $a3
f em $s0 ⊟ salvar $s0
Resultado em $v0

Código MIPS:

```
folha:
  addi $sp, $sp, -4
  sw   $s0, 0($sp)
  add  $t0, $a0, $a1
  add  $t1, $a2, $a3
  sub  $s0, $t0, $t1
  add  $v0, $s0, $zero
  lw   $s0, 0($sp)
  addi $sp, $sp, 4
  jr   $ra
```

Sincronização no MIPS

Load linked: ll rt, offset(rs)
Store conditional: sc rt, offset(rs)
Sucesso se não houve mudança desde ll (rt = 1)
Falha se local alterado (rt = 0)
Exemplo: swap atômico

```
try:  add $t0,$zero,$s4    ;copia valor
      ll  $t1,0($s1)       ;load linked
      sc  $t0,0($s1)       ;store conditional
      beq $t0,$zero,try    ;desvia se store falhou
      add $s4,$zero,$t1    ;põe valor do load em $s4
```

Modos de Endereçamento

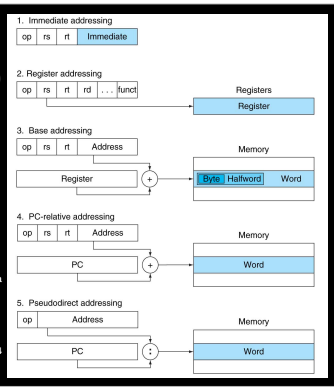


FIGURE 2.18 Illustration of the five MIPS addressing modes. The operands are shaded in color. The operand of mode 3 is in memory, whereas the operand for mode 2 is a register. Note that versions of load and store access bytes, halfwords, or words. For mode 1, the operand is 16 bits of the instruction itself. Modes 4 and 5 address instructions in memory, with mode 4 adding a 16-bit address shifted left 2 bits to the PC and mode 5 concatenating a 26-bit address shifted left 2 bits with the 4 upper bits of the PC.

Referências

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