**Arquiteturas de Computador II**

**EPC2 – Arquitetura MIPS**

Rodrigo Rufino

Márcio Rotella

Pedro Moreira

Índice

Índice 2

1. Arquitetura de Conjunto de Instruções MIPS 3

2. Primeira Instrução MIPS 4

3. Compilação de Expressões no MIPS 5

4. Convertendo uma Instrução com Array no MIPS 7

5. Armazenanto um valor em Array no MIPS 8

# Arquitetura de Conjunto de Instruções MIPS

# Primeira Instrução MIPS

# Compilação de Expressões no MIPS

Um programa pode, eventualmente, apresentar expressões um pouco mais complexas, envolvendo mais de uma operação com certa ordem lógica de resolução dela. Um exemplo disso são expressões matemáticas de soma e subtração utilizando parênteses.

Como exemplo, a seguinte expressão pode ser tomada:

f = e -  (a - b ) + ( b - c )

Se fosse para ser resolvida normalmente na mão, respeitando os parênteses como em uma expressão matemática qualquer, a ordem seria a seguinte:

f = e -  (a - b ) + ( b - c )

f = e – T1 + T2

f = e – T1 + T2

f = T3 + T2

**f = T3 + T2**

O código MIPS da expressão ficaria o seguinte (considerando os registradores pra cada variável: a = $s0, b = $s1, c = $s2, d = $s3, e = $s4, f = $s5):

SUB $t1, $ s0, $ s1

SUB $t2, $ s1, $ s2

SUB $t3, $ s4, $t1

ADD $ s5, $t3, $t2

Representação das instruções:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Op** | **Rs** | **Rt** | **Rd** | **Shamt** | **Funct** |
| 000000 | 16 ou 010 000 | 17 ou 010 001 | 9 ou 001 001 | 00000 | 100 010 |
| 000000 | 17 ou 010 001 | 18 ou 010 010 | 10 ou 001 010 | 00000 | 100 010 |
| 000000 | 20 ou 010 100 | 9 ou 001 001 | 11 ou 001 011 | 00000 | 100 010 |
| 000000 | 11 ou 001 011 | 10 ou 001 010 | 21 ou 010 101 | 00000 | 100 000 |

Então, o código de máquina fica assim:

00000001000001000100100100000100010

00000001000101001000101000000100010

00000001010000100100101100000100010

00000000101100101001010100000100000

# Convertendo uma Instrução com Array no MIPS

Para iniciar uma explicação de como acontece a conversão de uma instrução com Array no MIPS, precisa-se conhecer o conceito de instruções de formato Tipo l e load word (LW).

A instruções de formato Tipo l são classificadas como instrução de transferência de dados, possuem 32 bits, porém têm menos campos que a Tipo R (aritméticas).



A LW tem como função transferir dados da memória para os registradores e sempre deve ser utilizada quando se trata de uma Array. Sua sintaxe é:

LW registrador\_destino, valor (registrador\_fonte)

Assim, supondo a operação a = b + c[10]

O primeiramente deve-se converter c[10] que ficará assim:

  LW $t0, 10 ($s2)    # $t0 = memória [ $s2 + 10 ]

Observe que **$s2** é o vetor **c**, 10 é a posição do Vetor e **$t0** é um registrador temporário que armazenará o valor que está em **c[10]**. O segundo precisa-se fazer b + c[10]:

 ADD $s0,$s1,$t0    #$s0=$s1+$t0

Em que **$t0** é o valor de **c[10]**, **$s0** é a variável **a** e **$s1** é a variável **b**. Assim, o código final para

a = b + c [10] fica da seguinte forma:

  LW $t0, 10 ( $s2 )

ADD $s0, $s1, $t0

# Armazenanto um valor em Array no MIPS

Para armazenar uma instrução numa Array, precisa-se conhecer a instrução Store Word(SW).

A SW tem como objetivo armazenar um valor, que está localizado no registrador, na memória. Ela é bem parecida com a LOAD WORD, o que inverte é a fonte com o destino:

**SW registrador\_fonte, valor (registrador\_destino)**

Assim, supondo a operação a[15] = b + c

Atribuindo $s0 para a, $s1 para b e $s2 para c. O primeiramente deve-se converter

b + c:

**ADD $t0, $s1, $s2         # $t0 = $s1 + $s2**

 A instrução acima realiza uma soma entre b e c, armazenando o resultado em $t0. Agora, vamos armazenar $t0 no endereço de memória, que é o nosso array a[15]:

**SW $t0, 15 ( $s0)      # memória [ 15 + $s0 ] =  $t0**

 a[15] é correspondente ao código 15 ( $s0 ) e $t0 é o valor da soma. Portanto, o código final é:

**ADD $t0, $s1, $2**

**SW $t0, 15 ( $s0 )**