Instruções MIPS

Arrays

Recapitulando da aula passada...

Linguagem de Montagem ou de Máquina

A linguagem de montagem ou de máquina corresponde a trocar os nomes dos registradores pelo seu número, então nosso código ficará assim:

Código MIPS correspondente à expressão a = b - (c - d) + e

```
1   SUB $9, $18, $19    # $t1 = ( $s2 - $s3 ) é o mesmo que $t1 = ( c - d )
2   SUB $9, $17, $9     # $t1 = $s1 - $t1 é o mesmo que $t1 = b - ( c - d )
3   ADD $16, $9, $20    # $s0 = $t1 + $s4 é o mesmo que $s0 = b - ( c - d ) + e
```

Representação

A representação corresponde a pegar as nossas instruções MIPS e colocá-las no formato correspondente, conforme mostra a Tabela 1:

Representação das instruções

op	rs	rt	rd	shamt	funct
0	18	19	9	0	34
0	17	9	9	0	34
0	9	20	16	0	32

opcode	rs	rt	endereço
6 bits	5 bits	5 bits	16 bits
código de operação	registrador destino	registrador fonte	endereço de memória

Vamos usar o registrador **$s0** para a variável **a**, **$s1** para variável **b** e **$s2** para a variável **c**.
c é um vetor e não sabemos quantas posições ele tem, só sabemos que iremos somar o valor da variável **b** com o valor que está armazenado na posição 10 do vetor **c**.

Instruções de Formato Tipo I

As instruções de formato Tipo I têm menos campos que as instruções do formato tipo R (aritméticas), mas ainda tem 32 bits (no MIPS todas as instruções tem exatamente o mesmo tamanho em bits).
A instrução de formato Tipo I é representada na Tabela 1 e normalmente são classificadas como instruções de transferência de dados.

opcode	rs	rt	endereço
6 bits	5 bits	5 bits	16 bits
código de operação	registrador destino	registrador fonte	endereço de memória

Representação de Instruções do Formato Tipo I

O primeiro campo, OPCODE, é o código da operação e tem 6 bits; o segundo campo, RS, é o registrador destino e tem tamanho 5 bits; o terceiro campo, RT, é o registrador fonte e tem tamanho 5 bits; e o quarto campo, ENDEREÇO, é o endereço de memória que tem 16 bits.

Load Word (LW) e Store Word (SW) são duas instruções de transferência de dados, do formato I.
Hoje vamos usar a LW.

LOAD WORD

Essa instrução transfere dados da memória para os registradores e, sempre que tivermos um Array, devemos utilizá-la pois, antes de manipularmos o valor de uma determinada posição do Array, devemos tê-lo disponível para isso. Sua sintaxe é a seguinte:

```
LW registrador_destino, valor (registrador_fonte)
```

Exemplo:

```
LW $t0, 30 ( $s0 )          # $t0 = memória [ $s0 + 30 ]
```

O registrador **$t0** receberá o valor que está no endereço de memória que é calculado pela própria instrução: **$s0 + 30**. Então, toda vez que você usar a instrução LW, você está transferindo para um registrador, um valor que está no endereço de memória calculado pela soma do registrador fonte com um valor. Neste exemplo é um valor dado (30), ou seja, é a posição 30 do Vetor que aqui é representado por $s0. Mais pra frente veremos como fazer isso sem usar um valor específico.

Compilação de uma atribuição com um operando na memória

Vamos ver como fica a conversão da nossa instrução. O primeiro passo é 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 passo é 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
```

Observe também que uma instrução da linguagem C foi convertida em duas instruções para linguagem MIPS.

Linguagem de Máquina

A Linguagem de Máquina para a = b + c [10] ficará da seguinte forma:

LW $8, 10 ($18)

ADD $16, $17, $8

Representação da Linguagem de Máquina

A representação da linguagem de máquina para a = b + c [10] ficará da seguinte forma:

opcode	rs	rt	rd	shamt	funct
35	8	18	10		
0	17	8	16	0	32

Código de Máquina

O código de máquina para a = b + c [10] ficará da seguinte forma:

opcode	rs	rt	rd	shamt	funct
100 011	01000	10010	0000 0000 0000 1010		
000 000	10001	01000	10000	00000	100 000

Formato Tipo R

opcode	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits
código da operação	registrador fonte	registrador fonte	registrador destino	deslocamento	sub código da operação

Formato Tipo I

opcode	rs	rt	endereço
6 bits	5 bits	5 bits	16 bits
código da operação	registrador destino	registrador fonte	endereço de memória

Instruções

Tipo R:

Instrução	Exemplo	
ADD registrador destino, registrador fonte, registrador fonte	ADD $t0, $s0, $s1	$t0 = $s0 + $s1
SUB registrador destino, registrador fonte, registrador fonte	SUB $t1, $s3, $s4	$t1 = $s3 - $s4

Tipo I:

Instrução	Exemplo	
LW registrador destino, valor (registrador fonte)	LW $t0, 20 ($s0)	$t0 = memória [20 + $s0]

Exercícios (para entrega via moodle)

1) Converta as instruções abaixo:

i. a = b[25] + c;
ii. b = a[15] - c[30];
iii. c = b - a[12];

Use $s0 para a, $s1 para b e $s2 para c.

2) Apresente uma execução do código assembly para os itens i, ii e iii da questão 1 no Procsim.

Opcodes

OPCODE	DECIMAL	BINÁRIO
ADD	32	100 000
SUB	34	100 010
OR	36	100 100
AND	37	100 101
SLT	42	101 010
BNE	4	000 100
BEQ	5	000 101
J	2	000 010
JR	8	001 000
LW	35	100 011
SW	43	101 011