Arquitetura de Computadores

Turma LI21N/LT21N

Exercícios

PROGRAMAS EM ASSEMBLY

-

Ano Lectivo 2019/2020

2º Semestre

Prof. Jorge Fonseca

Conjunto de instruções suportado pela arquitetura

Instrução	Descrição	
mov rx, imm4	Carrega o valor imediato **imm4** no registo **rx**.	rx = imm4
ld rx, [ry]	Transfere para o registo **rx** o conteúdo da posição de memória cujo endereço é definido pelo conteúdo de **ry**.	rx = mem[ry]
st rx, [ry]	Transfere o conteúdo do registo **rx** para a posição de memória cujo endereço está definido pelo conteúdo de **ry**.	mem[ry] = rx
sub rx, ry, rz	Subtrai **rz** a **ry** e coloca o resultado em **rx** e atualizando o registo **PSW**	rx = ry - rz atualiza PSW
add rx, ry, rz	Adiciona **rz** a **ry** e coloca o resultado em **rx** e atualizando o registo	rx = ry + rz atualiza PSW
bae offset5	Quando a flag **C** apresenta o valor 0, muda a execução para o endereço resultante da adição ao **PC** do deslocamento **offset5**.	PC = (C == 0) ? PC + offset5 : PC + 1
b offset5	Muda a execução para o endereço resultante da adição ao **PC** do deslocamento **offset5**.	PC = PC + offset5
cmp rx, ry	Subtrai **ry** de **rx** e atualiza a flag **C** em conformidade com o resultado, que é descartado.	rx - ry

ISA	OPCODE			AD		AA		AB	
	8	7	6	5	4	3	2	1	0
Instruções de Transferência									
mov rx, imm4	0	0	0	rx		imm4			
ld rx, [ry]	0	0	1	rx		-	-	ry	
st rx, [ry]	0	1	0	-	-	rx		ry	
Instruções de processamento de dados									
sub rx, ry, rz	0	1	1	rx		ry		rz	
add rx, ry, rz	1	0	0	rx		ry		rz	
cmp rx, ry	1	1	1	-	-	rx		ry	
Instruções de controlo de fluxo									
bae offset5	1	0	1	-	offset5				
b offset5	1	1	0	-	offset5				

OP_ ALU

Numa instrução é preciso codificar:
- OPCODE: código único que distingue uma instrução de outra
- Operandos: pode ser operação da ALU, constante, Offset e/ou registos
- a codificação da operação da ALU pode pertencer ao OPCODE

Características:
- A constante imm4 para expressar um literal do tipo inteiro fica limitada a 4 bits, portanto ao valor 15.
- O offset5 é codificado com 5 bits e representa um inteiro com sinal ficando limitado a um salto baseado na instrução corrente para +15 ou -16 instruções.

Exemplos de utilização:

- *Alterar uma variável em memória*

```
maior = 0
 ;r3 = endereço da variável maior
MOV r1, 0
ST r1, [r3]
```

- *Obter o valor de uma variável em memória*

```
i = menor
;r3 = endereço da variável menor
;r1 = i
LD r1, [r3]
```

Exemplos de utilização:

- *Implementar a comparação de dois valores e tomar decisão*

```
        If a < 6
           a = a + 1
        else
           a = a – 1
        ; r2=6
        ; r1=a
        CMP r1, r2
        BAE else
then:
        MOV r3,1
        ADD r1, r1, r3
        B end_if
else:
        MOV r3,1
        SUB r1, r1, r3
end_if:
```

Exemplos de utilização:

- *Implementar um ciclo for*

```
for (j=0; j < 4; j++) {
    …
}
```

```
 ;r1=j
MOV r1, 0
for:
MOV r2, 4
CMP r1, r2
BAE end_for
…
MOV r2, 1
ADD r1, r1, r2
B for
end_for:
```

Exemplos de utilização:

- Implementar um ciclo while

```
        while (j >= 5) {
            …
        }
        ;r1=j
while:
        MOV r2, 5
        CMP r1, r2
        BAE body_while
        B end_while
body_while:
        …
        B while
end_while:
```

PROGRAMAS

Exercício3 : Escrever em Código Máquina um programa para determinar o maior valor contido no array x.

				OPCODE			AD		AA		AB	
				8	7	6	5	4	3	2	1	0
Code Address	MNEMONICS			INSTRUCTION								
	LABEL	OPCODE	OPERANDS	BIN								HEX
00		MOV	R0,2	0	0	0	0	0	0	0	1	0	002
01	r3=x[0]	LD	R3, [R0]	0	0	1	1	1	-	-	0	0	070
02		MOV	R2, 1	0	0	0	1	0	0	0	0	1	021
03	maior=x[0]	ST	R3, [R2]	0	1	0	0	0	1	1	1	0	08E
04	r1=i	MOV	R1,1	0	0	0	0	1	0	0	0	1	011
05		MOV	R2, 0	0	0	0	1	0	0	0	0	0	020
06	i = 1	ST	R1, [R2]	0	1	0	0	0	0	1	1	0	086
07	for:	MOV	R2,6	0	0	0	1	0	0	1	1	0	026
08	i<6	CMP	R1,R2	1	1	1	0	0	0	1	1	0	1C6
09		BAE	+13	1	0	1	0	0	1	1	0	1	14D
0A		ADD	R2,R1,R0	1	0	0	1	0	0	1	0	0	124
0B	r2=x[i]	LD	R2,[R2]	0	0	1	1	0	0	0	1	0	062
0C	if:	CMP	R2,R3	1	1	1	0	0	1	0	1	1	1CB
0D		BAE	+2	1	0	1	0	0	0	0	1	0	142
0E		B +5	+5	1	1	0	0	0	0	1	0	1	185
0F	THEN	MOV	R3, 0	0	0	0	1	1	0	0	0	0	030
10		ADD	R3,R2,R3	1	0	0	1	1	1	0	1	1	13B
11		MOV	R2, 1	0	0	0	1	0	0	0	0	1	021
12	maior=r3	ST	R3, [R2]	0	1	0	0	0	1	1	1	0	08E
13	i++	MOV	R2,1	0	0	0	1	0	0	0	0	1	021
14		ADD	R1, R1, R2	1	0	0	0	1	0	1	1	0	116
15		MOV	R2,0	0	0	0	0	1	0	0	0	0	010
16		ST	R1, [R2]	0	1	0	0	0	0	1	1	0	086
17		B	-16	1	1	0	0	1	0	0	0	0	190
18	fim_for:	B 0	0	1	1	0	0	0	0	0	0	0	180

RAM	
ADDR	
0	i
1	maior
2	x[0]
3	x[1]
4	x[2]
5	x[3]
6	x[4]
7	x[5]

Algoritmo

```
nibble i, maior;
nibble x[6]; /* inteiros sem sinal */
(r3)maior=(r0)x[0];
for ((r1)i=1; i < 6; i++) {
   if (x[i] > maior) maior = x[i];
}
```

ISA	OPCODE			AD		AA		AB	
	8	7	6	5	4	3	2	1	0
mov rx, imm4	0	0	0	rx		imm4			
ld rx, [ry]	0	0	1	rx		-	-	ry	
st rx, [ry]	0	1	0	-	-	rx		ry	
sub rx, ry, rz	0	1	1	rx		ry		rz	
add rx, ry, rz	1	0	0	rx		ry		rz	
bae offset5	1	0	1	-	offset5				
b offset5	1	1	0	-	offset5				
cmp rx, ry	1	1	1	-	-	rx		ry	