Universidade Federal Rural do Semiárido
Departamento de Computação
Ciência da Computação
Arquitetura e Organização de Computadores
Prof. Sílvio Fernandes

2.1 – Trabalho de Implementação

Trabalho em dupla ou individual
Data de entrega (via SIGAA): 04 de Setembro de 2023
Pontuação: Até 2,0 na II Unidade (1,5 implementando o *assembler* mais simples e 0,5 para reconhecer os registadores pelos nomes, ex: $s0 assim como $16)

Descrição:
Implemente, na linguagem de programação de sua preferência, um montador (*assembler*) simples para o MIPS. Sua implementação deve receber como entrada um arquivo de texto com código de montagem (*assembly*) do MIPS sem erros de sintaxe e devolver como saída um arquivo de texto com programa equivalente em linguagem de montagem. O trabalho precisa ser apresentado em sala de aula.

Seu montador deve ser capaz de reconhecer as instruções a seguir, montando o binário de cada uma delas de acordo com o tipo de instrução (R, I ou J):
Formato R:

op	rs	rt	rd	shamt	funct
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Nome	Formato	Exemplo	Codificação					
			opcode	rs	rt	rd	sa	function
sll	R	sll $8, $9, 3	0	0	9	8	3	0
srl	R	srl $8, $9, 3	0	0	9	8	3	2
jr	R	jr $8	0	8	0	0	0	8
mfhi	R	mfhi $8	0	0	0	8	0	16
mflo	R	mflo $8	0	0	0	8	0	18
mult	R	mult $9, $10	0	9	10	0	0	24
multu	R	multu $9, $10	0	9	10	0	0	25
div	R	div $9, $10	0	9	10	0	0	26
divu	R	divu $9, $10	0	9	10	0	0	27
add	R	add $8, $9, $10	0	9	10	8	0	32
addu	R	addu $8, $9, $10	0	9	10	8	0	33
sub	R	sub $8, $9, $10	0	9	10	8	0	34
subu	R	subu $8, $9, $10	0	9	10	8	0	35
and	R	and $8, $9, $10	0	9	10	8	0	36
or	R	or $8, $9, $10	0	9	10	8	0	37
slt	R	slt $8, $9, $10	0	9	10	8	0	42
sltu	R	sltu $8, $9, $10	0	9	10	8	0	43
mul	R	mul $8, $9, $10	28	9	10	8	0	2

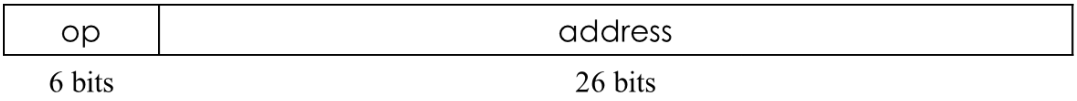
Figura 1 - Instruções da família R

Formato I:

Nome	Formato	Exemplo	Codificação			
			opcode	rs	rt	immediate
beq	I	beq $8, $9, 3	4	8	9	3
bne	I	bne $8, $9, 3	5	8	9	3
addi	I	addi $8, $9, 3	8	9	8	3
addiu	I	addiu $8, $9, 3	9	9	8	3
slti	I	slti $8, $9, 3	10	9	8	3
sltiu	I	sltiu $8, $9, 3	11	9	8	3
andi	I	andi $8, $9, 3	12	9	8	3
ori	I	ori $8, $9, 3	13	9	8	3
lui	I	lui $8, 3	15	0	8	3
lw	I	lw $8, 4($9)	35	9	8	4
sw	I	sw $8, 4($9)	43	9	8	4

Figura 2 - Instruções da família I

Formato J:



Nome	Formato	Exemplo	Codificação	
			opcode	endereço
j	J	j 1000	2	1000
jal	J	jal 1000	3	1000

Figura 3 - Instruções da família J

O montador deve gerar código de máquina apenas das instruções, não sendo necessário usar as diretivas de montador (.data, .text, etc). Seu montador deve inicialmente passar por todo o código procurando rótulos (*labels*), os quais são identificados por um nome seguido de dois-pontos e guardar a informação da linha onde foram encontrados (isso é sua tabela de símbolos). Em seguida, percorrendo todo o código novamente, e traduzir cada instrução para código de máquina. Nas instruções que fazem referência aos *labels,* os mesmos devem ser substituídos pelos respectivos valores relativos (instruções condicionais fazem referência ao valor de PC e incondicionais são pseudodiretas). Seu *assembler* ainda deve considerar que o endereço de memória inicial do programa é sempre 0x00400000. A saída gerada pode ser em em hexadecimal (ou seja, cada 4 bits equivalente a 1 caracter hexa). O montador mais simples deve implementar os registradores indicados diretamente pelo seu endereço no código assembly (ex: $16 ao invés de $s0).

A seguir um exemplo de entrada e saídas possíveis:

Código de entrada (arquivo salvo com extensão .asm):

```
L1: add $t0, $s1, $s2
L2:   addi $t1, $s3, 7
      beq   $t0, $t1, L1
      j L2
```

O binário equivalente ao código assembly seria:

```
00000010001100100100000000100000
00100010011010010000000000000111
00010001000010011111111111111101
00001000000100000000000000000001
```

Código de saída do seu montador em hexadecimal deve ser um arquivo de texto salvo com extensão .hex no seguinte formato:

```
v2.0 raw
12a4020 8d100000 ad100000 8100001
```

Esse formato de arquivo de saída deve começar com o cabeçalho “v2.0 raw”, e cada instrução deve ser escrita em hexadecimal separada por um espaço. Esse é o formato utilizado para inicializar memórias no Logisim, onde será testado em trabalhos futuros. É sugerido manter até 4 instruções por linha, no arquivo, de modo que se tivéssemos 8 instruções desse exemplo teríamos como saída isso:

```
v2.0 raw
12a4020 8d100000 ad100000 8100001
12a4020 8d100000 ad100000 8100001
```