LABORATÓRIO DE ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES I

AULA 7: Projeto de um Processador: implementação do programa-teste no MARS

Professor: Mateus Felipe Tymburibá Ferreira
Data: 19/07/2021

Aluno: Darmes Araújo Dias

Mudanças no projeto:

Instrução	opcode	rd	rs	immediate	description
load	000	X	X	XXX	rd = MEM[rs*immediate]
store	001	X	X	XXX	MEM[rs*immediate]= rd
addi	010	X	-	LABEL(XXXX)	rd = rd + immediate
add	011	X	X	rs2(X)	rd = rs+rs2
beq	100	-	-	LABEL(XXXXX)	rd==rs
la	101	X	-	LABEL(XXXX)	rd = LABEL
j	110	-	-	LABEL(XXXXX)	j LABEL
halt	111	-	-	-	PC = PC

Trocamos as instruções AND e OR por LA (Load Address) e J (jump).
E na instrução add adicionamos mais um argumento.

1) Apresente o código do programa-teste em assembly do MIPS (utilize comentários para facilitar a revisão do seu código).

```
#AULA 7:Projeto de um Processador: implementação do programa-teste no MARS
#Aluno: Darmes Araujo Dias
.data # definindo onde começam os dados do programa
        array:
        .word 1,2,3,4,5,6,7,8 # inicializando valores do array

.text # a partir daqui começam as instruções
.globl main
        main:
                la $s1, array # carregando t1 com o endereço de array
                li $s2, 0 #garantir que a soma comece com 0
                li $t1, 2 # vai fazer a comparação
                li $t2,0 #iterator
                li $t3, 8 #size
                j loop
        loop:
                beq $t2,$t3,exit # quando o iterador for igual ao size sai do loop
                mul $t4,$t2,4  # calcula o index do array
                add $t5,$s1,$t4 #x[i]
                lw $t6,0($t5) #carregamos o valor de t5 em t6, só para podermos mexer em no conteúdo sem alterar o original
                j if
        is_even:        # Quando é par soma em s2 o valor
                lw $t7, 0($t5) # salva o valor de t5 em t7
                add $s2,$s2,$t7 # adiciona t7 em s2(vai ser a soma dos pares)
                j final_of_loop
        final_of_loop:
                addi $t2,$t2,1 # i+=1
                j loop
        if:
                sub $t6,$t6,$t1 # subtraimos 2 do número até ele ficar igual a -1 ou 0 ou 1, dessa maneira, conseguimos checar se ele é par ou impar
                beq $t6,0,is_even #par
                beq $t6,1,final_of_loop #impar
                beq $t6,-1,final_of_loop #impar
                j if

        exit:
                .data
```

(O código está na pasta, caso seja mais fácil ler por lá).

2) Apresente o código do programa-teste em assembly do seu nRisc (utilize comentários para facilitar a revisão do seu código).

```
#AULA 7:Projeto de um Processador: implementação do programa-teste no MARS
#Aluno: Darmes Araujo Dias
.data # definindo onde começam os dados do programa
        zero: # Este valor deve ser sempre o endereço 0
        .word 0
        array:
        .word 1,2,3,4,5,6,7,8 # inicializando valores do array
        size:
        .word 8
#Só possuo dois registradores na minha arquitetura, $s1 e $s2
# end do iterator 10
#end da soma 11
#end do valor de array[i] 12
#beq sempre compara os únicos dois registradores
.text # a partir daqui começam as instruções
.globl main
        main:
                la $s2, zero # dessa maneira zeramos o reg
                la $s1, zero # dessa maneira zeramos o reg
        loop:
                la $s1, zero
                addi $s1, 10 # endereço do iterator
                lw $s1, 0($s1) # colocamos o conteúdo do endereço do iterator
                la $s2, size
                lw $s2, 0($s2)
                beq exit # compara s1 com s2, se caso s1 for igual a 8 sai do loop
                la $s2, array
                add $s2,$s2,$s1 # array[i]
                lw $s2, 0($s2) # colocamos o conteúdo do endereço de array[i]
                la $s1, zero # dessa maneira zeramos o reg
                addi $s1, 12
                sw $s2, 0($s1) # guardamos o valor de array[i] no endereço 12
                # parte de acrescentar 1 no iterator e guarda-lo na memória
                la $s1, zero
                addi $s1, 10
                lw $s1, 0($s1)
                addi $s1, 1 # i+=1
                la $s2, zero
                addi $s2, 10
                sw $s1, 0($s2) #guardando o valor do iterator na memória
```

```
        sw $s1, 0($s2) #guardando o valor do iterator na memória
        la $s2, zero
        addi $s2, 12
        sw $s2, 0($s2)
if:
        addi $s2, -2 # subtraimos 2 do número até ele ficar igual a -1 ou 0 ou 1, dessa maneira, conseguimos checar se ele é par ou impar
        la $s1, zero
        beq is_even #par
        addi $s1, 1
        beq loop #impar
        addi $s1,-2
        beq loop #impar
        j if
is_even:
        la $s1, zero
        addi $s1, 12 # endereço do conteúdo de array[i] que foi salvo em loop
        lw $s1, 0($s1)
        la $s2, zero
        addi $s2, 11 # endereço de soma
        lw $s2, 0($s2) # conteúdo de soma
        add $s2, $s2,$s1
        la $s1, zero # salvando no endereço de soma
        addi $s1, 11
        sw $s2, 0($s1)# guardamos o resultado da soma no endereço da soma
        j loop
exit:
        .data
```

(O código está na pasta, caso seja mais fácil ler por lá).

3) Apresente o código do programa-teste em binário no formato de instruções definido para o seu nRisc (indique cada instrução em uma linha de texto separada).
Registradores: $s1 = 0 e $s2 = 1
Só possuo dois registradores na minha arquitetura, $s1 e $s2
end do iterator 10
end da soma 11
end do valor de array[i] 12
beq sempre compara os únicos dois registradores

```
main:
        la $s2, zero # 10110000
        la $s1, zero # 10100000
loop:
        la $s1, zero # 10100000
        addi $s1, 10 # 01001010
        lw $s1, 0($s1) # 00000000
        la $s2, size # 10111001
        lw $s2, 0($s2) # 00011000
        beq exit # 100endereço
        la $s2, array # 10110001
        add $s2,$s2,$s1 # 01111000 (estendi 0 s1)
        lw $s2, 0($s2) # 00011000
        la $s1, zero # 10100000
        addi $s1, 12 # 01001110
        sw $s2, 0($s1) # 00110000
        la $s1, zero # 10100000
        addi $s1, 10 # 01001010
        lw $s1, 0($s1) # 00000000
        addi $s1, 1 # 01000001
        la $s2, zero # 10110000
        addi $s2, 10 # 01011010
        sw $s1, 0($s2) # 00101000
        la $s2, zero # 10110000
```

```
        addi $s2, 12 # 01011110
        sw $s2, 0($s2) # 00111000
if:
        addi $s2, -2 # 0101
        la $s1, zero # 10100000
        beq is_even # 100endereço
        addi $s1, 1 # 01000001
        beq loop # 100endereço
        addi $s1,-2 01001110 (complemento de 2)
        beq loop # 100endereço
        j if #110endereço
is_even:
        la $s1, zero # 10100000
        addi $s1, 12 # 01001110
        lw $s1, 0($s1) # 00000000
        la $s2, zero # 10110000
        addi $s2, 11 # 01011011
        lw $s2, 0($s2) # 00011000
        add $s2, $s2,$s1 # 01111000(estendi rs2)
        la $s1, zero # 10100000
        addi $s1, 11 # 01001011
        sw $s2, 0($s1) # 00110000
        j loop # 110endereço
exit: # 111XXXXX (HALT → PC)
        .data
```