Av3 - 2° Semestre de 2021

Avaliação 3 - Elementos de Sistemas

Pontos HW	Pontos SW
30	40

- Avaliação **individual**.
- **100 min** total.
- Ficar conectado no canal geral no Teams (para ouvir instruções).
- Ficar no blackboard durante a prova.
- Clonar o seu repositório (e trabalhar nele)
- Fazer **commit** ao final de cada questão.
- Lembre de dar **push** ao final.

As questões de hardware (`.vhd`) devem ser implementadas nos arquivos localizados na pasta `src/vhd` , as questões de software (`nasm`) devem ser implementadas nos arquivos localizados em `src/nasm` . Os scripts a seguir testam respectivamente a parte de hardware e software:

```
./testeHW.py
./testeAssembly.py
```

> Vocês devem editar o arquivo `config_testes.txt` selecionando o que desejam testar.

LEMBRE DE REALIZAR UM COMMIT (A CADA QUESTÃO) E DAR PUSH AO FINALIZAR

1. Separação de bits

Pontos HW	Pontos SW
0	15

Considere o nosso computador Z01.1 será usado para o processamento de uma imagem em escala de cinza, que estará armazenada na memória a partir da posição RAM[10]. A profundidade de cada pixel da imagem é representada por 8 bits, de forma que cada posição de memória possui o valor da profundidade de 2 pixels. Por exemplo, a RAM[10] armazena os valores dos 2 primeiros pixels da imagem.

- Os bits 15 a 8 devem ser copiados para a RAM[1].
- Os demais bits da RAM[0] e RAM[1] devem permanecer em '0'.

Exemplo:

Valor inicial da memória:

RAM[10] = 'xxxxxxxxyyyyyyyy'

Resultado:

RAM[0] = '00000000yyyyyyyy'

RAM[1] = 'xxxxxxxx00000000'

```
leaw $10, %A
movw (%A), %D
leaw $255, %A
andw %D, %A, %D
leaw $0, %A
movw %D, (%A)
leaw $10, %A
movw (%A), %D
leaw $65280, %A
andw %D, %A, %D
leaw $1, %A
movw %D, (%A)
```

Implementação

Implemente as funções lógicas para as saídas no arquivo `src/nasm/separacao.nasm`

Rubrica para avaliação:

Pontos HW	Descritivo
15	Função implementada e funcionando (implementação passa em todos os testes)
5	Cópia realizada sem selecionar os bits (passa nos dois primeiros testes)

2. Sequência de Fibonacci

Pontos HW	Pontos SW
0	15

A sequência de Fibonacci é uma sequência de números inteiros, onde cada termo subsequente corresponde à soma dos dois termos anteriores. A sequência normalmente começa com "0" e "1" e tem aplicações em diversas áreas como mercado financeiro, ciências da computação e fenômenos da natureza.

RAM[12] = '0000000000000001'

RAM[13] = '0000000000000010'

e assim suscessivamente.

Implementação

Implemente as funções lógicas para as saídas no arquivo `src/fibonacci.nasm`

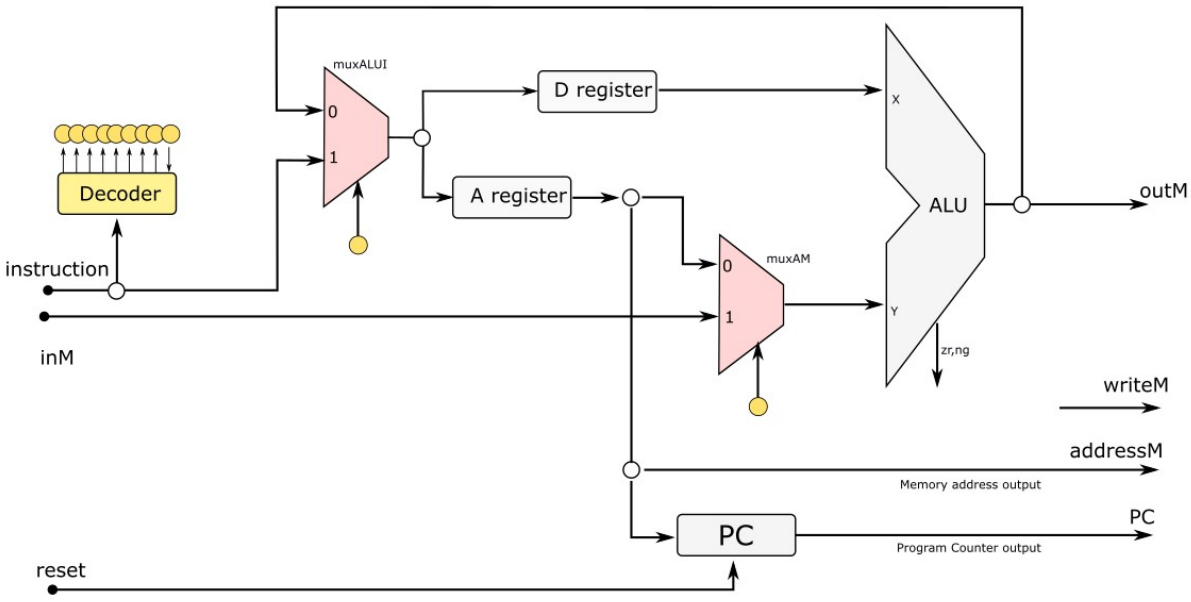
Rubrica para avaliação:

Pontos HW	Descritivo
15	Sequência criada com uso de saltos
5	Sequência criada sem uso de saltos

3. CPU modificada

Pontos HW	Pontos SW
20	0

Pretende-se fazer uma modificação na CPU de forma a alterar a posição do MuxALUI permitindo que um valor seja carregado tanto no registrador %A como no %D ou em ambos em um mesmo ciclo. A figura a seguir ilustra a modificação:




```
muxALUI <= '0' when instruction(17 downto 16) = "10" else '1';
loadA <= (not(instruction(17)) and not(instruction(16))) or
        (instruction(17) and not(instruction(16)) and instruction(3)) or
        (instruction(17) and (instruction(16)));
loadD <= (instruction(17) and not(instruction(16)) and instruction(4)) or
        (instruction(17) and (instruction(16))) or
        (not(instruction(17)) and instruction(16));
```

Implementação

Implemente no arquivo `vhd/controlUnit3.vhd` os sinais de controle do MuxALUI e dos sinais de load dos registradores %A e %D.

> Os sinais de load devem considerar tanto a operação de leaw como as demais instruções tipo C, assim como no CPU montado no projeto F.

Rubrica para avaliação:

Pontos HW	Descritivo
20	As três fuções implementadas e funcionando
10	Apenas o controle do MuxALUI implementado e funcionando
5	Apenas o load de um registrador implementado e funcionando

4. CPU - instrução

Pontos HW	Pontos SW
10	10

Considere que a instrução

"1000000110100100001"

seja aplicada a nossa CPU feita em aula e representada na figura a seguir.

- Reg_D: "0111111111111111"

a)

Determine o valor na saída da ULA e nos registradores %D e %A imediatamente após o processamento da instrução. ULA = %D = "1000000000000000"
%A = "0000000000000000"

Escreva sua resposta nos respectivos campos do arquivo `vhd/questao5.txt` .

b)

Quais são as linhas em Assembly que efetuariam a mesma operação dessa nova instrução?

```
notw %D
jg %D
```

Escreva sua resposta no respectivo campo do arquivo `vhd/questao5.txt` .

Rubrica para avaliação:

Pontos HW	Pontos SW	Descritivo
10	0	Item a
0	10	Item b

```
; valores iniciais
leaw $10, %A
movw $0, (%A)
leaw $11, %A
movw $1, (%A)
; inciando contador RAM na RAM[0]
leaw $12, %A
movw %A, %D
leaw $0, %A
movw %D, (%A)
LOOP:
leaw $0, %A
movw (%A), %D
leaw $20, %A
subw %D, %A, %D
leaw $END, %A
jg %D
nop
leaw $0, %A
movw (%A), %A
decw %A
movw (%A), %D
decw %A
movw (%A), %A
addw %A, %D, %D
leaw $0, %A
movw (%A), %A
movw %D, (%A)
incw %A
movw %A, %D
leaw $0, %A
movw %D, (%A)
leaw $LOOP, %A
jmp
nop
END:
```