Elementos de Sistema - **Simulado**
Prova “10” - Prática

Nome completo:

Pontos de:	**HW**	**SW**
	/ 30	**/ 55**

Instruções:

1. A avaliação tem duração total de 120 minutos.
2. **Você não pode consultar a internet, apenas seu repositório LOCAL**
3. Você deve editar esse documento.
4. Assim como nos projetos, os códigos fontes estão em: **/src/rtl/** ***src/nasm*** e o arquivo de configuração dos testes em **/test/config.txt**

1) (4 HW, 4 SW) Conceitos

a) (2 HW, 2 SW) Qual o papel do Program Counter na execução de um programa?

b) (2 HW, 2 SW) Explique como um programa executa no computador (RAM/ROM/CPU)

2) (0 HW, 15 SW) pseudocódigo

Arquivo:	Projetos/F-Assembly/src/nasm/p3Q2.vhd
Teste:	SIM, simulação
Output:	Código p3Q2.vhd

Transcreva o pseudocódigo a seguir para assembly do Z01.1:.

```
WHILE(TRUE):
	IF RAM[5] == 3:
		RAM[0] = -3
	ELSE:
		RAM[0] = 0
```

- Para testar, descomentar linha do `F-Assemblt/tests/config.txt`
  - `p3Q2.nasm 2 1000`

3) (0 HW, 35 SW) Pisca LED

Arquivo:	Projetos/F-Assembly/src/nasm/p3Q3.vhd
Teste:	Gravar no Hardware e ver LEDs piscarem
Output:	Vídeo dos LEDs piscando no repositório

Considerando que nosso hardware opera com um clock de 50.000.000 Hz, um código em assembly que faz com que os LEDs da placa a aproximadamente 1s.

(Piscar = tudo por aceso por um segundo e depois tudo apagado por um segundo)

- Para testar:
  - `./programFPGA`
  - `./programSoftware.py -n src/nasm/p3Q3.nasm`

4) (6 HW, 6 SW) Engenharia reversa

Arquivo:	
Teste:	
Output:	

Você está fazendo engenharia reversa em um equipamento para entender o que um programa realiza. Analisando o chip utilizado no projeto sua equipe percebeu que o produto faz uso do Z01.1.

Considere o estado das memórias como a seguir (extraído do chip via leitura do JTAG):

```
CONTENT BEGIN
0 : 0000000000000011
1 : 1001011000000010000
2 : 0001010010110000000
3 : 1000000110100100000
4 : 1000000110000010000
END;
```

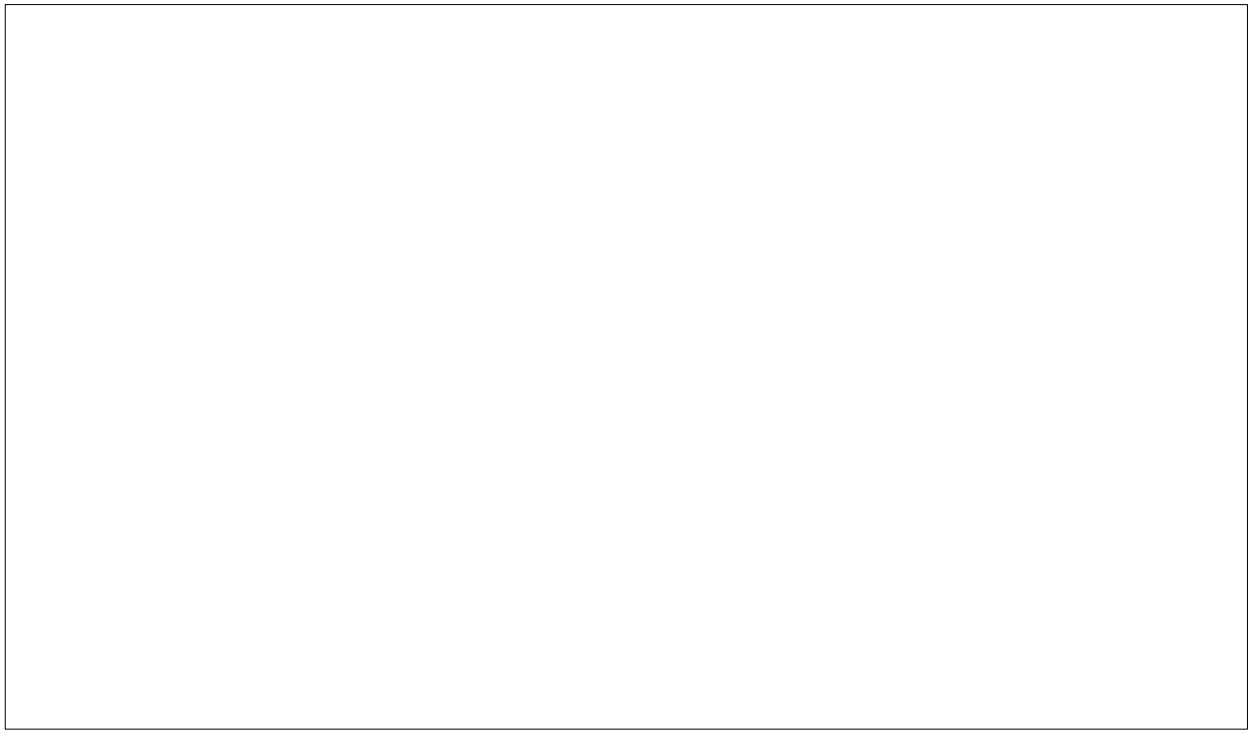
ROM

```
CONTENT BEGIN
0 : 0000000000000001
1 : 0101100000010000
2 : 0101001011000000
3 : 0000001000000000
4 : 0000011000001000
END;
```

RAM

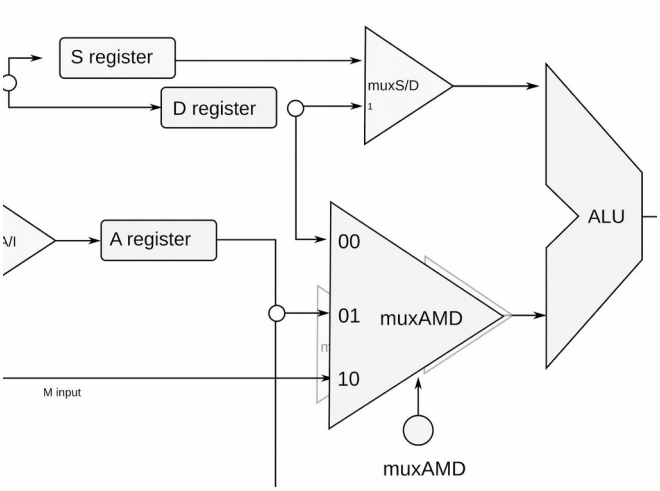
a) (6 HW, 3 SW) Faça o dissasembly do código (transcreva o programa para nasm)

b) (3 SW) explique o que devemos ver no hardware ao final de sua execução.



5) (20 HW, 5 SW) Mudando o HW

Um colega do seu grupo responsável por desenvolver a CPU achou muito mais interessante utilizar um único MUX (`muxAMD`) de quatro entradas (que já tinha sido desenvolvido na entrega C) no lugar dos dois mux: `muxA/M`, e `muxAM/D`



a) (5 HW, 0 SW) Faça a modificação no Hardware do Z01.1 para incorporar esse novo Mux.

Arquivo:	Projeto/G-CPU/
Teste:	Gerar RTL e analisar novo componente
Output:	RTL da nova CPU

- Ao realizar essa modificação, você deve realizar um **commit com a mensagem:** `“muxAMD na CPU”`

COLAR RTL AQUI

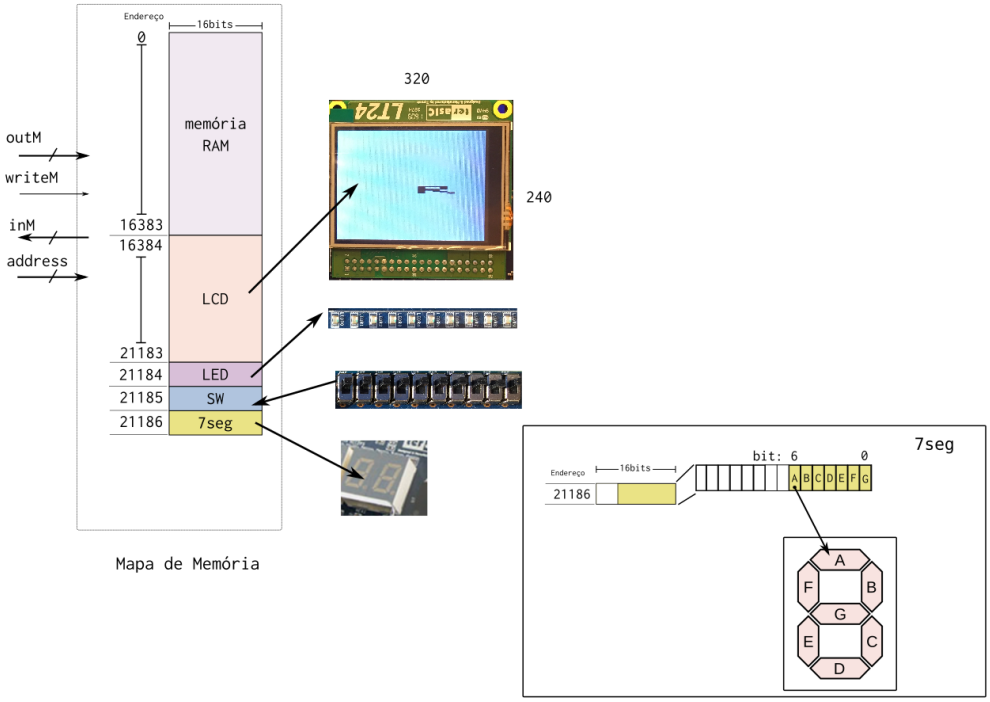
b) (5 HW, 0 SW) Com essa modificação, será necessário alterar o `ControlUnit` que era responsável por controlar os `muxA/M` e, `muxAM/D` para controlar o novo seletor do `muxAMD`.

Arquivo:	Projetos/G-CPU/
Teste:	testeHW.py e testeAssemblyMyCPU.py
Output:	Código modificado

- Para isso será necessário remover as duas saídas `muxAM` e `muxAMD` da entidade do `ControlUnit` e adicionar o novo sinal `muxAMD`

- Ao realizar essa modificação, você deve realizar um commit com a mensagem: “corrigido `ControlUnit` para controlar novo `muxAMD`”

Seu colega teve outra ideia! Inserir o display de sete segmentos como periférico do `MemoryIO`, como ilustrado a seguir:



c) (5 HW, 0 SW) Realize essa modificação no hardware do Z01.1

Arquivo:	Projetos/G-CPU/
Teste:	testeHW.py e testeAssemblyMyCPU.py

- Você deve adicionar a entidade do MemoryIO a seguinte porta:
  - **SSEG : `out std_logic_vector(6 downto 0);`**

- Você deve substituir o arquivo `/Projetos/G-Computador/src/rtl/Computador.vhd` com o arquivo `Computador_NEW.vhd` que está na mesma pasta.

d) (5 HW, 5 SW) Escreva um código em assembly que demonstre que escreva o digito 7 no display de sete segmentos e prove que o novo hardware está funcionando.

Arquivo:	Projetos/F-Assembly/p3Q5.nasm
Teste:	testeAssemblyMyCPU.py -g (waveform)
Output:	Waveform

COLAR AQUI O WAVEFORM AQUI QUE DEMONSTRA O SEVEN SEG FUNCIONANDO

- Para testar deixar só a linha a seguir descomentanda no `F-Assembly/tests/config.txt`
  - `p3Q5.nasm 1 1000`

- Na pasta do projeto G, executar teste: ./testeAssemblyMyCPU -g