Elementos de Sistema - Prova “11”

Nome completo:

Pontos de:	**HW**	**SW**
	/ 20 + 10	**/ 50 + 10**

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**

Revisão 1) (20 HW) Revisão HW

NO PAPEL

ENADE 2014
EXAME NACIONAL DE DESEMPENHO DOS ESTUDANTES

QUESTÃO DISCURSIVA 5

Um processo monitora três parâmetros para controle de qualidade: A, B, C. Cada parâmetro possui um valor na decisão final da qualidade. A existência do parâmetro A pesa 30% na decisão final, enquanto os parâmetros B e C pesam 30% e 40%, respectivamente. O grau de aprovação do processo é dado pela soma dos percentuais desses três parâmetros. O produto gerado pelo processo é considerado aprovado, caso o grau de qualidade seja superior ou igual a 60%, e reprovado, se o grau de qualidade for inferior ou igual a 30%. Caso o grau de qualidade esteja entre 30% e 60%, a decisão de aprovação ou reprovação é indiferente. Por exemplo, se um produto apresentar os parâmetros A e B, terá grau de qualidade de 30%+30% = 60%, levando à sua aprovação.

Com base na situação descrita, projete um circuito lógico com o menor número possível de portas lógicas, para determinar a aprovação ou não do produto de acordo com a presença de seus parâmetros. As entradas do circuito serão os sinais A, B, C, e a saída será um sinal Z. Para atingir esse objetivo, faça o que se pede nos itens a seguir.

a) Monte uma tabela verdade do sistema com a formação ABC. (valor: 4,0 pontos)

b) Desenhe o circuito final otimizado utilizando portas lógicas. (valor: 6,0 pontos)

Revisão 2) (10 HW) Simplificação

NO PAPEL

Simplifique os mapas a seguir:

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1	1	1
$\bar{A}B$	1	1	0	0
AB	0	0	0	1
$A\bar{B}$	0	0	1	1

(a)*

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	0	1	1
$\bar{A}B$	1	0	0	1
AB	0	0	0	0
$A\bar{B}$	1	0	1	1

(b)

	$\bar{C}$	C
$\bar{A}\bar{B}$	1	1
$\bar{A}B$	0	0
AB	1	0
$A\bar{B}$	1	X

(c)

Revisão 3) (10 SW) VectorMax

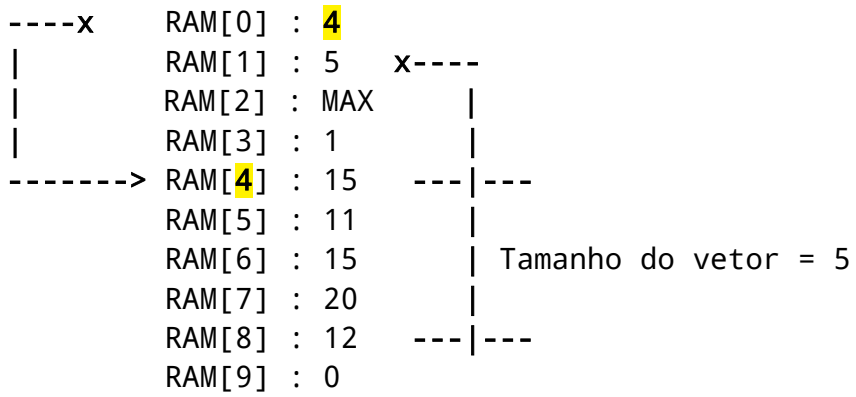
Arquivo	Projetos/F-Assembly/src/nasm/**p3VectorMax.nasm**		
Teste	Simulação	**./testeAssembly.py**	`p3VectorMax.nasm 2 1000`

Buscar pelo valor máximo de um vetor e atualizar a RAM 2 com o valor encontrado

Assuma que:
- O endereço **0** da RAM indica a **posição inicial de um vetor**
- O endereço **1** da RAM indica o **tamanho do vetor**
- O vetor é uma região contínua da RAM

Considere o exemplo na qual um vetor de tamanho 5 está armazenado na memória RAM começando no endereço 4.

Vector = [15, 11, 15, 20, 12]

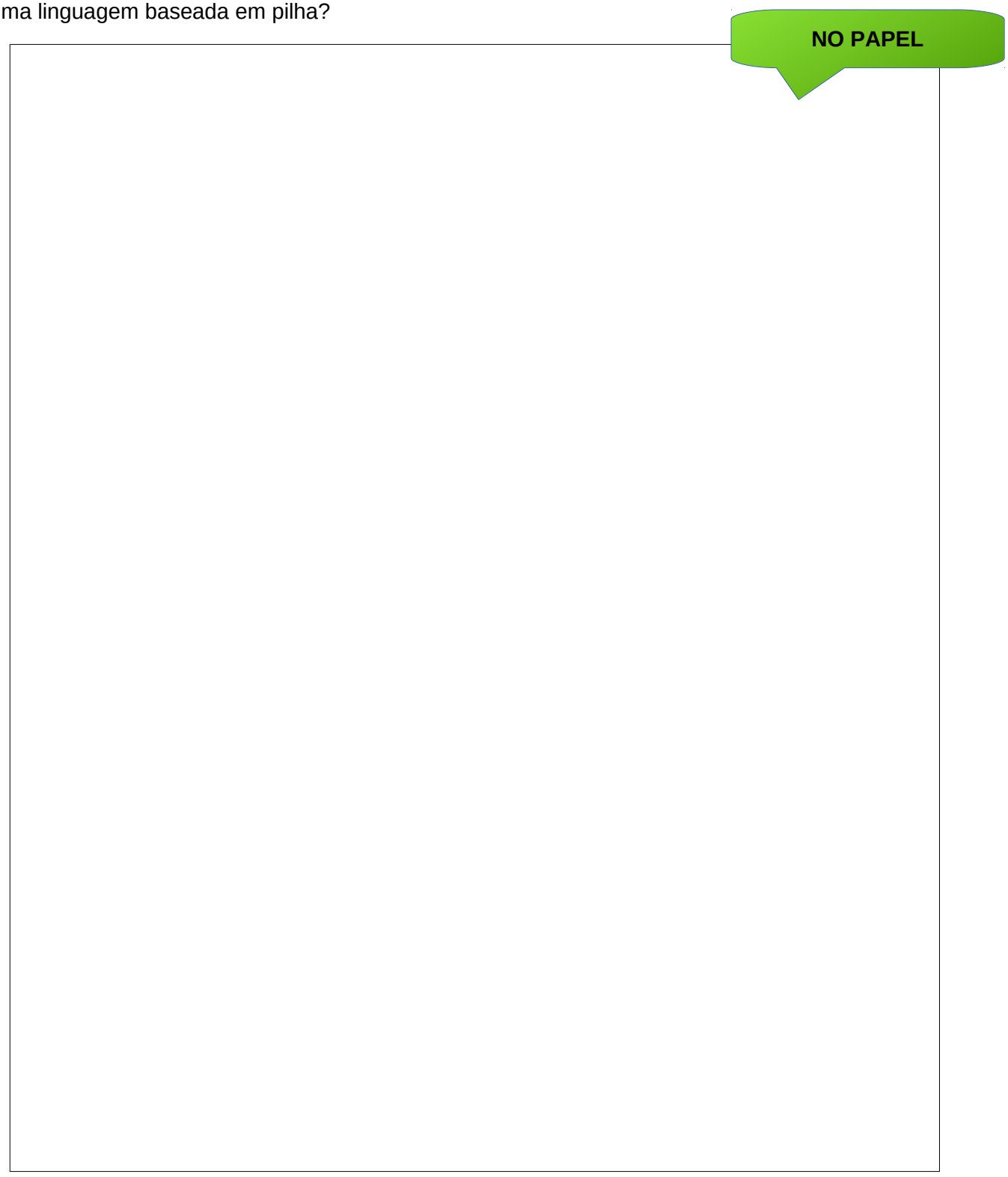


- Para testar: descomentar a linha do **`/F-Assembly/tests/config.txt`**
  - `p3VectorMax.nasm 2 1000`

Teste 0 (0 pts)	O teste 0 é o exemplo mostrado a seguir (vetor começa em 4 e tem 5 termos)
Teste 1 (10 pts)	O teste 1 é genérico, você deve ler os valores em RAM[0] e RAM[1]

1) (5 HW / 5 SW) Conceitos

Explique como um computador que possui arquitetura baseada em registradores, implementa uma linguagem baseada em pilha?



2) (0 HW, 15 SW) matemática

Arquivo	***/I-VM/src/vm/pseudo/***		
Teste	Simulação	**./I-VM/testeVM.py**	`pseudo 2 4000`

Transcreva a equação a seguir para ser executada em linguagem VM (precisa executar em VM).

```
IF (TEMP[0] * TEMP[1]) > 6:
       TEMP[3] = -3
ELSE:
       RAM[3] = - TEMP[0]
```

- Para testar descomentar linha a seguir do arquivo: *I-VM/tests/config.txt*
  - `pseudo 2 4000`
- E execute:
  - `./I-VM/testeVm.py`

Teste 0 (7.5 pts)	ELSE
Teste 1 (7.5 pts)	IF

3) (0 HW, 10 SW) Fibonacci

Arquivo	*Projetos/I-VM/src/vm/Q2-Simulado-factorial/factorial.vm*		
Teste	Simulação	**.**/I-VM/testeVM.py	`fibonacci 4 20000`

Implemente uma função em VM que retorna o valor do ultimo termo do Fibonacci, dado um valor N de interações.

	F(0)	F(1)	F(2)	F(3)	F(4)	F(5)	**F(6)**
F(n) =	0	1	1	2	3	5	**8**

Exemplo, se passado o valor 6 como argumento, a função deve retornar 8.

Você deve editar apenas a função *fibonacci.vm*, não precisa mexer no *main.vm*. A função fibonacci, recebe um argumento, e deve retornar o último valor da série, em N interações.

- Para testar descomente a linha a seguir do arquivo: *I-VM/tests/config.txt*
  - `fibonacci.vm 4 10000`
- E execute:
  - `./I-VM/testeVm.py`

Teste 0 (1 pts)	F(0)
Teste 1 (1.5 pts)	F(1)
Teste 2 (2.5 pts)	F(2)
Teste 3 (10 pts)	F(n), n > 2

4) (10 SW) Novo comando VM

Arquivo:	*J-VMTranslator/VMtranslator/src/main/java/vmtranslator/Code.java*
Teste:	SIM, simulação

Vamos implementar um novo comando em vm, chamado de **add3**, que adiciona três elementos da pilha, e retorna o valor no primeiro termo, como no exemplo a seguir

```
Stack antes     | Stack depois do comando de add3
-------------------------------------------------
        1       |       6
        2       |    ->
        3       |
    ->
```

A implementação deve ser feita dentro do método classe :

```
public void writeArithmetic(String command) {
          ...
    else if (command.equals("add3")) {
          …
          …
   }
```

- Para testar descomentar linha a seguir do arquivo: *I-VM/tests/config.txt*
  - add3 1 1000
- E execute:
  - ./J-VMTranslator/testeVMtranslator.py

Teste 0 (10 pts)	add3

5) (10 SW) Stack

NO PAPEL

Faça a evolução da Stack: