O objetivo desta lista de exercícios é a de incorporar a Linguagem de Máquina do computador teórico NEANDER (LMN) à disciplina de Sistemas Digitais. Contribuindo com a compreensão do funcionamento deste computador teórico para facilitar o processo de implementação do simulador em VHDL.

Também será usado o arquivo .mem, gerado pelo wneander como arquivo a ser carregado no simulador implementado.

Esta lista contém 08 exercícios em ordem crescente de dificuldade.

EX01: Adição entre 3 números

Criar um programa em LMN que execute o seguinte comando:

```
d = a + b + c;
```

Considerações:

Posições de memória:

Variável:	a	b	c	d
Posição:	128	129	130	131

Demais variáveis e constantes são livre acima da posição 135.

Instruções sugeridas:

LDA end	ADD end	STA end	HLT

EX02: Operação em Complemento de Dois

Criar um programa em LMN que execute o seguinte comando:

```
b = -a;
```

Considerações:

Posições de memória:

Variável:	a	b
Posição:	128	129

Demais variáveis e constantes são livre acima da posição 130.

Instruções sugeridas:

LDA end	ADD end	STA end	NOT	HLT

Dica:

Operação de Complemento de Dois!

EX03: Subtração entre dois números

Criar um programa em LMN que execute o seguinte comando:

```
a = a-b;
```

Considerações:

Posições de memória:

Variável:	a	b
Posição:	128	129

Instruções sugeridas:

LDA end	ADD end	STA end	NOT	HLT

Demais variáveis e constantes são livre acima da posição 130.

Dica:

Lembre-se: subtração é adição de **a** com o Complemento de Dois de **b**.

EX04: Maior número

Criar um programa em LMN que execute o seguinte código:

```
if (a >= b){
        c = a;
}
else {
        c = b;
}
```

Considerações:

Posições de memória:

Variável:	a	b	c
Posição:	128	129	130

Demais variáveis e constantes são livre acima da posição 131.

Instruções sugeridas:

LDA end	ADD end	STA end	NOT	JZ end	HLT

Dicas:

(a – b) = 0 ←→ b = a

(a – b) < 0 ←→ b > a

(a – b) > 0 ←→ b < a

use SF

EX05: Contar

Criar um programa em LMN que execute o seguinte código:

```
contar = 1;
for(i=1; i<=n; i++){
        contar = contar + 1;
}
```

Considerações:

Posições de memória:

Variável/Constante:	contar	i	1
Posição:	128	129	130

Demais variáveis e constantes são livre acima da posição 131.

Instruções sugeridas:

LDA end	ADD end	STA end	NOT	JN end	JZ end	HLT

EX06: Multiplicação

Criar um programa em LMN que execute o seguinte código:

```
for(i=1; i<=multiplicador; i++){
        multiplicando = multiplicando + multiplicando;
}
```

Considerações:

Posições de memória:

Variável/Constante:	multiplicando	multiplicador	i
Posição:	128	129	130

Demais variáveis e constantes são livre acima da posição 131.

Instruções sugeridas:

LDA end	ADD end	STA end	NOT	JZ end	HLT

Dica:

use ZF

EX07: Progressão aritmética

Criar um programa em LMN que execute o seguinte código:

```
nessimo = primeiro;
for(i=(n-1); i>0; i--){
        nessimo = nessimo + razao;
}
```

Considerações:

Posições de memória:

Variável/Constante:	primeiro	i	nessimo	razao
Posição:	128	129	130	131

Demais variáveis e constantes são livre acima da posição 132.

Instruções sugeridas:

LDA end	ADD end	STA end	NOT	JZ end	HLT

Dica:

Faça todos os exercícios anteriores.

EX08: Divisão inteira

Criar um programa em LMN que execute a divisão de dois inteiros com possibilidade de resto.

Sem código para auxiliar raciocínio!

Considerações:

Posições de memória:

Posições de memória:

Variável/Constante:	a	b	Q	R
Posição:	128	129	130	131

Variáveis e constantes são livre acima da posição 132.

Instruções sugeridas:

Todas as que achar necessário!

Dica:

Subtração sucessiva até resultado ser menor do que zero.

5 / 2 → Q = 2 e R = 1