

Trabalho Neander

Data de entrega: até 28/11/2024 às 23:59

Escreva um programa para o simulador Neander que calcule a distância de Hamming entre dois números a e b de 8 bits. A distância de Hamming é definida como a quantidade de bits que são alterados entre dois valores que possuem o mesmo número de bits. Por exemplo, a distância de Hamming entre os números binários 01001001 e 10001000 é igual a 3. A distância de Hamming pode ser calculada pela operação OU-exclusivo (XOR) entre os números a e b , seguido pela soma do número de bits em '1' do resultado da operação $a \text{ XOR } b$. Não existe uma instrução no Neander para a operação XOR, mas ela pode ser realizada com as instruções NOT, AND e OR através da seguinte forma:

$$a \text{ XOR } b = (\text{NOT}(a) \text{ AND } b) \text{ OR } (a \text{ AND } \text{NOT}(b))$$

Para contar os bits em '1', sugere-se calcular a máscara AND de cada um dos bits do resultado de $a \text{ XOR } b$, do menos significativo ao mais significativo, com os valores 1, 2, 4, 8, 16, 32, 64 e 128, respectivamente.

O programa Neander a ser desenvolvido receberá os valores de a e b e calculará a distância de Hamming desses valores. Devem ser **obrigatoriamente** utilizadas as seguintes posições de memória (mostradas em decimal):

- Posição 128 – entrada a
- Posição 129 – entrada b
- Posição 130 – saída $a \text{ XOR } b$
- Posição 131 – saída **Distância de Hamming**

Os trabalhos serão corrigidos de forma automática, com **20** pares de valores de entrada diferentes. Portanto, devem ser observadas rigorosamente as seguintes especificações:

- o código do programa deve iniciar no endereço 0 da memória;
- a primeira instrução executável deve estar no endereço 0;
- os endereços para os operandos e para o resultado devem ser exatamente os especificados acima;
- usar para variáveis adicionais, constantes, ou para código extra os endereços de memória de 132 em diante;
- no cálculo, os valores de a e b (endereços 128 e 129) não devem ser modificados. Ou seja, se necessário, deve-se copiar os valores de a e b para variáveis de trabalho (no endereço 132 ou superior) e codificar o algoritmo usando estas variáveis de trabalho;
- variáveis alteradas durante o programa devem ser inicializadas pelo próprio programa. Sempre que necessário, utilizar posições de memória não alteradas (constantes) para realizar a inicialização.

O trabalho deverá ser entregue através do sistema Moodle, na área de “Trabalho Neander”, na forma de um arquivo compactado (formato Zip) contendo:

- um arquivo de memória do Neander (.mem), com o código de máquina do programa.
- um arquivo texto, contendo o código do programa em formato simbólico, **comentado** (dica: usar a função “Arquivo ... Salvar texto ...” do simulador para gerar o texto inicial). Os comentários devem descrever as principais operações realizadas pelo programa. Lembre-se de incluir seu nome completo e seu número de cartão nas primeiras linhas deste arquivo.

- Para dar nomes aos arquivos, utilize o seu nome completo, sem espaços e sem acentos, seguido do seu número de cartão, sem zeros à esquerda. Por exemplo: João da Silva, cartão 00123456 utilizará JoaodaSilva123456.mem, JoaodaSilva123456.txt e JoaodaSilva123456.zip.
A entrega de arquivos cujos nomes não obedecem a esta regra implicará em um desconto de 5% na nota do trabalho.

IMPORTANTE: Este é um trabalho **individual**.

Data de Entrega: até 28/11/2024 às 23h59, via Moodle. Não haverá prorrogação deste prazo.

Alguns casos de teste

Teste	a (end. 128)	b (end.129)	$aXORb$ (end. 130)	$DistHamming$ (end. 131)
1	73	136	193	3
2	128	0	128	1
3	0	3	3	2
4	7	8	15	4
5	85	170	255	8