UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA - DEPARTAMENTO DE INFORMÁTICA APLICADA

INF01107 – Introdução à Arquitetura de Computadores –2021/2

Atividade de avaliação 2 - Simulador NEANDER

Escrever um programa para o simulador Neander que calcule a distância de Hamming entre dois dados *a* e *b* de 8 bits. A distância de Hamming é definida como a quantidade de bits que são alterados entre dois dados que possuem o mesmo número de bits no total. Pode-se pensar na distância de Hamming como o número de bits que devem ser alterados de um dado para se transformar em outro dado. Por exemplo, a distância de Hamming entre os dados binários 01001001 e 10001000 é igual a 3.

A distância de Hamming pode ser calculada por diversos algoritmos. Sugerimos o algoritmo que começa pelo cálculo da operação OU-exclusivo (XOR) entre os números *a* e *b*. Não existe uma instrução nativa no Neander para esta operação, mas ela pode ser realizada com as instruções NOT, AND e OR através da seguinte forma:

a XOR b = (NOT(a) AND b) OR (a AND NOT(b))

Após este cálculo, basta somar o número de bits em '1' no resultado da operação *a* XOR *b*. No exemplo anterior, o resultado de *a* XOR *b* é igual a 11000001. Para contar os bits em '1', sugere-se calcular a máscara AND de todos bits do resultado, 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 **Distância de Hamming**

Os trabalhos serão corrigidos de forma automática, com **20** pares de valores 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 131 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 131 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 "Entrega da Atividade 2", 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 obedeçam a esta regra implicará em um desconto de 5% na nota do trabalho**.

IMPORTANTE: Este é um trabalho **individual**. Trabalhos copiados terão a sua nota dividida entre os envolvidos.

Data de Entrega: até 04/04/2022 às 23h59, via Moodle. Não haverá prorrogação deste prazo.

Alguns casos de teste

Teste	*a* (end. 128)	*b* (end.129)	*DistHamming* (end. 130)
1	73	136	3
2	0	0	0
3	255	255	0
4	1	4	2
5	15	6	2
6	7	8	4
7	170	85	8
8	85	170	8
9	1	2	2
10	0	1	1