;—Cálculo de MDC no NeanderWin—;

Centro Universitário Tiradentes - UNIT

Docente: Alexsandro Henrique Batista

Discente: Guilherme Santana Bernardo Ximenes

Data: 17/09/2025

Recife, PE

1. Qual era o Objetivo?

Esse projeto surgiu de uma proposto pela UFRGS e a tarefa era fazer um código que conseguisse calcular o Maior Divisor Comum (MDC) entre dois números inteiros em 8 bits que podiam ser positivos ou negativos. A entrada desses números seria pelos endereços 128 e 129 da memória, e a resposta final tinha que ser guardada no endereço 130.

2. Pensamento para resolver o problema:

A primeira coisa foi decidir como fazer o cálculo. No material foi dado como sugestão o Algoritmo de Euclides e explicou como é utilizado, ou seja, já induziu a gente a utilizar esse método. Isso foi perfeito, porque, como o método mostra que é possível utilizar subtrações sucessivas, seria possível realizar essa tarefa no Neander, já que o simulador não realiza divisões e nem multiplicações.

Então, meu plano foi basicamente este:

1. Dar um jeito de lidar com os números negativos;

2. Aplicar o algoritmo de Euclides com as subtrações;

3. Por fim, mostrar o resultado final.

3. O Código Passo a Passo:

Eu separei o código em algumas partes pra ficar mais organizado e fácil de entender o que está acontecendo.

Parte 1: Deixando os Números Prontos (Dando um jeito nos números negativos)

O enunciado dizia que o MDC de um número negativo é o mesmo que o do seu positivo (ex: mdc(-10, 5) = mdc(10, 5)). Então, a primeira coisa que meu programa faz é garantir que ele só vai trabalhar com números positivos.

Para o número 'a' (que entra no endereço 128), a lógica foi:

- Eu uso o comando “LDA” pra pegar o número que o usuário digitou e coloco ele no Acumulador, que seria tipo a “área de trabalho”;

- Depois, coloquei o comando JN (Jump if negative). Isso faz com que os Flags da ULA(Unidade Lógica e Aritmética) acendam dependendo do resultado das operações. Se o número no Acumulador for negativo, a luz "N" acende, e o JN vê isso e pula pra uma parte do código que eu chamei de “A_NEGATIVO;
- Se o número for positivo, o programa simplesmente ignora o pulo e guarda o número numa variável temporária que eu criei, a “NUM_A”;
- Na parte “A_NEGATIVO”, eu pensei da seguinte maneira pra inverter o sinal: coloco o número 0 no Acumulador com “LDI 0” e depois uso “SUB 128”. Basicamente, ele faz 0 - (o número negativo), o que, pela regra de sinais da matemática, resulta em um número positivo. Aí sim eu guardo esse valor novo positivado na “NUM_A”.

Depois de fazer tudo isso pro 'a', ele pula (JMP) e repete o processo inteirinho para o número 'b', guardando o resultado na variável “NUM_B”.

Parte 2: Utilizando o algoritmo de euclides - Loop principal

Aqui o programa fica repetindo os mesmos passos até encontrar a resposta. Eu chamei o início desse loop de” LOOP_AE”:

- O pulo do gato aqui é o comando “SUB”. Eu carrego o “NUM_A” e subtraio o “NUM_B” dele. O melhor é que essa única conta me dá três informações de uma vez, só olhando para as "luzinhas" (flags) do processador:
    1. Se o resultado for **zero**, a luz "Z" acende. Isso significa que a e b são iguais! Se eles são iguais, eu achei o MDC. Aí o comando JZ (Jump if zero) manda o programa direto pro final;
    2. Se o resultado for **negativo**, a luz "N" acende. Isso significa que “a” era menor que “b”. Aí o comando JN manda o programa para uma parte que eu chamei de “A_MENOR_QUE_B”;
    3. Se nenhuma luz acendeu, é porque o resultado foi positivo, o que significa que “a” era maior que “b”.
- **Caso em que a > b:** O resultado da conta “a - b” já está guardado no Acumulador. Então, eu só uso o “STA NUM_A” para atualizar o valor de “a” com esse novo resultado. Depois, mando ele de volta pro começo do loop com “JMP LOOP_AE”;
- **Caso em que a < b:** A conta “a - b” não serve mais. Então, eu tenho que recarregar o “NUM_B” no Acumulador e subtrair “NUM_A” dele, pra fazer a conta certa (b - a). Aí eu atualizo o “NUM_B” com esse resultado e mando de volta pro começo do loop.

Ele fica repetindo isso, diminuindo os números, até que eles finalmente fiquem iguais e o “JZ” tire-os do loop.

Parte 3: Mostrando a Resposta (O Fim)

Quando o programa finalmente pula para “FIM_FINALMENTE”, o trabalho tá perto do final na verdade.

- Eu uso “LDA NUM_A” para pegar o resultado final (tanto faz pegar “NUM_A” ou “NUM_B”, porque agora eles são iguais);
- Guardo esse resultado no endereço 130 com “STA 130”, como foi pedido no projeto;
- Uso o comando “OUT 0”, para aparecer o resultado do MDC no visor;

- E, finalmente, coloco “HLT” para terminar o programa.

Obs.: Para não misturar o código com as variáveis temporárias “NUM_A” e “NUM_B”, eu usei o ORG 131 para pular lá pra frente na memória e reservei um espacinho para cada uma delas com o “DS 1”.

LINK DO PDF DA PROPOSTA FEITA PELA UFRGS:
https://www.inf.ufrgs.br/arq/wiki/lib/exe/fetch.php?media=wiki:trab:neander:2015-maior_divisor_comum.pdf