Serviço Público Federal
Ministério da Educação
Fundação Universidade Federal de Mato Grosso do Sul

Nome: Raul Leão Chagas
RGA: 2021.1906.064-6
Professora: Daniela Luiza Catelan

Exponenciação

1. Introdução

Este relatório tem como objetivo detalhar o funcionamento e abordar sobre o processo de desenvolvimento do programa em MARIE que realiza a exponenciação de dois números inteiros, base e expoente, utilizando a técnica de multiplicação sucessiva. A exponenciação é realizada através da multiplicação repetida da base, o número de vezes indicado pelo expoente.

2. Explicação do funcionamento da exponenciação

O código implementa um algoritmo de exponenciação utilizando o método da multiplicação sucessiva (repetida). Ele recebe o endereço da base e o expoente como entrada, realiza o cálculo da exponenciação e exibe o resultado.

- **Entrada de Dados:** O programa recebe o endereço da base e o expoente como entrada.
- **Verificação do Expoente:** O programa verifica se o expoente é zero. Se for, a execução é encerrada.
- **Loop de Exponenciação:** O programa entra em um loop para calcular o valor da exponenciação. Ele decrementa o expoente a cada iteração e chama uma loop para realizar a multiplicação da base pelo resultado atual. Esse processo é repetido até que o expoente seja zero.
- **loop de Multiplicação:** A loop de multiplicação é responsável por multiplicar a base pelo resultado atual da exponenciação. Ela realiza repetidas subtrações e acumulações para realizar a multiplicação.
- **Saída do Resultado:** Após o loop de exponenciação, o resultado final é exibido.

3. Explicação e Organização do Código

O código foi organizado em seções, a qual incluem a entrada dos dados, verificação e execução do loop principal, e a saída do resultado.

Descrição das variáveis de entrada e saída

- **Entrada:**
  - base: Endereço da base da exponenciação.
  - expoente: Valor do expoente.
- **Saída:**
  - resultado: Resultado da exponenciação.

Clareza de comentário

Essa seção do código recebe os valores da base e do expoente e os armazena nas variáveis base e expoente, respectivamente. Em seguida, verifica se o expoente é zero. Se for, encerra a execução do programa.

```
// Essa parte do código inicializa o programa recebendo o endereço da base e o expoente.
// Em seguida, verifica se o expoente é zero. Se for, o programa é encerrado.
input_usuario, input                / Recebe o endereço da base
    StoreI base                     // Armazena o valor da base no endereço indicado
    store resultado                 // Inicializa resultado com o valor da base
    input                           / Recebe o expoente
    store expoente                  // Armazena o expoente
    Skipcond 800                    // Se o expoente for zero, pula para halt
    halt                            / Encerra a execução se o expoente for zero
```

Nesta seção é iniciado um loop para calcular a exponenciação. Ele verifica se o expoente é zero. Se não for, decrementa o expoente, chama loop de multiplicação e retorna para verificar novamente o valor do expoente.

```
// O programa verifica novamente se o expoente é zero.
// Se não for, continua para o cálculo da exponenciação. Caso contrário, pula para a saí
da.
verifica_expoente, load expoente            // Carrega o valor do expoente
    subt one                            // Decrementa o expoente
    store expoente                      // Armazena o novo valor do expoente
    Skipcond 800                        // Se o expoente for zero, pula para outp
    jump print                          // Pula para a saída

// O programa chama o loop de multiplicação e,
// em seguida, retorna para verificar novamente o valor do expoente.
    JnS loop_multiplicacao              // Chama o loop de multiplicação
    Jump verifica_expoente                      // Retorna para check_mult
```

Essa seção implementa a loop de multiplicação. Ela vai carregar a base e o resultado atual, realiza a multiplicação e armazena o novo resultado. Esse processo é repetido até que a base seja zero.

```
// O loop de multiplicação é definido e implementado. A base é carregada em 'a' e o
// resultado atual é carregado em 'b'. Em seguida, um loop é executado, subtraindo
// repetidamente a base do resultado atual e armazenando o novo resultado.
loop_multiplicacao, hex 0           // Inicializa o loop de multiplicação
    LoadI base                          // Carrega o valor da base
    store a                             // Armazena a base em 'a'
    load resultado                      // Carrega o valor atual de resultado
    store b                             // Armazena o valor atual de resultado em
'b'
```

```
logica_multiplicacao, load a                        // Carrega o valor de 'a'
    subt one                            // Decrementa 'a'
    Skipcond 800                        // Se 'a' for zero, pula para o endereço de
retorno
    JumpI loop_multiplicacao            // Retorna do loop de multiplicação

    store a                             // Armazena o novo valor de 'a'
    load resultado                      // Carrega o valor atual de resultado
    add b                               // Soma 'b' ao valor atual de resultado
    store resultado                     // Armazena o novo valor de resultado
    Jump logica_multiplicacao                       // Repete o loop de multipli
cação
```

Essa seção declara as variáveis e printa o resultado.

```
// Resultado final e encerra sua execução.
print, load resultado                   // Carrega o valor final de resultado
    Output                                  / Exibe o resultado
    Halt                                    / Termina a execução

// var
base, hex 0                             // Endereço da base
resultado, dec 0                        // Resultado da multiplicação
expoente, dec 0                         // Valor do expoente
one, dec 1                              // Constante 1
a, dec 0                                // Valor temporário 'a'
b, dec 0                                // Valor temporário 'b'
add, dec 1
```

Algoritmo:

- O algoritmo consiste em um loop de exponenciação que utiliza um loop de multiplicação para calcular a exponenciação(para realizar a multiplicação sucessiva).

Funções, Parâmetros, Variáveis:

- loop_multiplicacao: Loop responsável pela multiplicação da base pelo resultado atual da exponenciação.
- base, expoente, resultado: Variáveis para armazenar a base, o expoente e o resultado da exponenciação.
- a, b: Variáveis temporárias para armazenar valores intermediários durante a multiplicação.

4. Resultados e Discussão:

O programa calcula corretamente a exponenciação em quase todos os casos, utilizando o método da multiplicação repetida(sucessiva). Os resultados podem ser verificados com diferentes valores de entrada para a base e o expoente. Em continuidade, alguns casos de testes foram realizados e foi percebido alguns comportamentos do programa que não apresentaram o resultado esperado.

a) 2^2
   i) Valor esperado: 4
   ii) Valor retornado: 4

b) 2^0
   i) Valor esperado: 1
   ii) Valor retornado: valor não retornado pelo sistema

c) 0^2
   i) Valor esperado: 0
   ii) Valor retornado: 0

5. Dificuldades Encontradas:

Apesar de o código ter sido implementado parcialmente com sucesso, algumas dificuldades foram enfrentadas ao longo do processo. A falta de conhecimento em Assembly (MARIE) tornou a dificultou a implementação consideravelmente, pois a sintaxe e a lógica de controle em Assembly são bastante

diferentes das linguagens de programação de alto nível (Java, Python, JavaScript). Além disso, encontrar recursos e material de estudo sobre MARIE foi difícil, já que há menos documentação e tutoriais disponíveis para MARIE em comparação com linguagens de programação mais populares, como JavaScript, Java e Python. A complexidade da linguagem Assembly também contribuiu para as dificuldades enfrentadas, já que a programação em Assembly é notoriamente mais complexa do que em linguagens de alto nível. A manipulação direta de registradores e a necessidade de um gerenciamento das variáveis e do fluxo de controle aumentaram a complexidade do desenvolvimento.

6. Soluções de Implementação:

A implementação de loops para tarefas específicas, como a multiplicação, ajudou a organizar o código e tornar mais intuitivo. Também, foi adotado o uso de verificações condicionais e loops bem, a fim de garantir que o programa execute corretamente e termine conforme esperado. Além do mais, para calcular a exponenciação, o código utiliza o método de multiplicação repetida. Esse método envolve multiplicar a base pelo resultado acumulado repetidamente, até que o expoente se reduza a zero.

7. Conclusão:

Este relatório fornece uma análise detalhada do código de exponenciação usando multiplicação sucessiva. O código foi explicado em termos de seu funcionamento, estrutura, variáveis e algoritmo.