Implementação de uma calculadora básica em uma linguagem de baixo nível

Mariana Ferreira Rocha e Natália Andrade Calmon

Resumo — Esse documento é relacionado a implementação de uma calculadora de operações básicas - soma, subtração, multiplicação e divisão - em uma linguagem de baixo nível, sendo tratada em Netwide Assembler (NASM) que é um montador da linguagem assembly.

O artigo em questão contém todas as restrições e requisitos do sistema, assim como uma descrição do problema, abordando suas funções e capacidade de uso, limitando assim ao seu uso correto.

***Abstract* — This document is about a implementacion of a calculator of basic operations - sum, division, multiplication, subtraction - in a low-level programming language using Netwide Assembler (NASM) that it is a assembler of assembly language.**

The article in question contains all the restriction and system requirements, as well as a description of the problem, addressing your functions and capacity to use, limiting in this way the correct way to use.

***Palavras Chave* — NASM, Netwide Assembler, Calculadora, Assembly, Linguagem de Montagem.**

I. INTRODUÇÃO

De acordo com o dicionário da língua portuguesa Priberam, uma calculadora é uma "máquina que efetua operações aritméticas simples ou complexas e cujas pequenas dimensões são devidas ao emprego de semicondutores e de circuitos integrados"[1]. Com essa definição, podemos restringir o contexto e aplicá-lo ao escopo do trabalho desenvolvido, onde foi desenvolvido o algoritmo de uma calculadora de operações básicas, envolvendo soma, subtração, multiplicação e divisão de números de ponto flutuante.

Esse trabalho consiste em uma implementação de uma máquina de operações numéricas na linguagem de máquina assembly utilizando o montador - assembler - NASM.

O Netwide Assembler, NASM, é um montador 80x86 e x86-64, ou seja é compatível com arquiteturas de 32 e 64 bits, projetado para ter portabilidade e modularidade. Ele suporta uma variedade de formato de arquivo, podendo ser utilizado em máquinas linux e windows . Sua sintaxe é projetada para ser simples e de fácil de se compreender[2][7].

II. DESCRIÇÃO DO PROGRAMA

Esse algoritmo desenvolvido é dividido em funções que descrevem o funcionamento de uma calculadora, onde cada parte representa uma etapa do processo.

A função inicial, *main,* têm como objetivo fazer a leitura da primeira variável a ser calculada, e logo em seguida o operador. Dado o operador, é conferido a validade do mesmo e então é enviado a suas respectivas funções de operação - *soma, subtracao, divisao, multiplicacao* - por uma chamada de função, permanecendo em um loop até a tecla enter seja pressionada pelo usuário, no lugar do operando, então a função *fechar* é executada finalizando o programa.

As funções *soma, subtracao e multiplicacao* seguem o mesmo padrão, lendo o segundo numero digitado, efetuando a operação proposta pelo nome da função, imprimindo seu resultado na tela e retornando para função inicial. Ao contrário das funções descritas acima que não necessitam de nenhum tipo de verificação, a função *divisao*, após receber o segundo valor confere se o mesmo é igual a zero. Caso isso ocorra, é chamada a função *erroDivisao,* o qual emite uma mensagem de erro, descrevendo que o número digitado é inválido - na matemática um número dividido por zero tende a infinito, impossibilitando sua representação[8] - e então é permitido que seja digitado um novo valor que substituirá o zero inserido anteriormente.

A *Figura 1* ilustra essa distribuição de funções para o exercício do programa de maneira mais simples, para um bom entendimento, onde cada cor representa uma função implementada de acordo com a legenda.

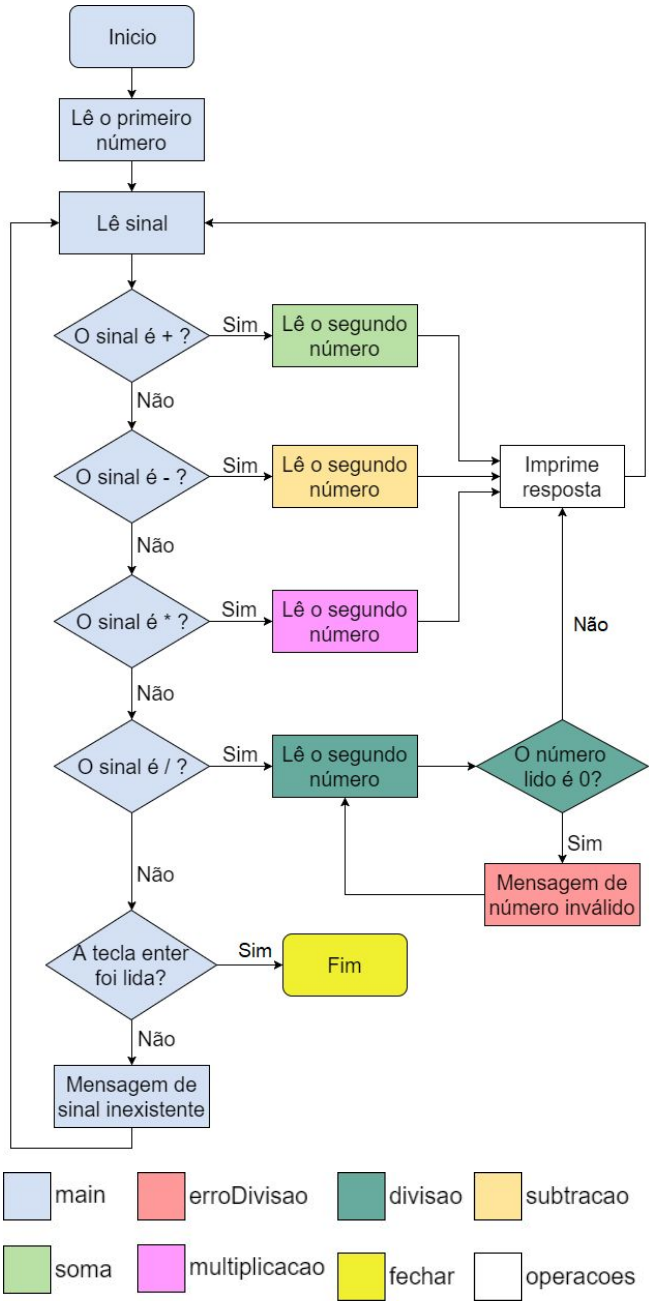

Figura 1- Fluxograma de funcionamento do programa.

Como qualquer função, essas descritas acima, são compostas por operações, que são as intruções do programa. São elas que regem o funcionamento do sistema a partir de uma entrada de dados do problema. A *Tabela 1* lista o nome de cada instrução, assim como, seu tipo e descrição.

Nome	Tipo	Descrição
push	Movimentação de dados	Adiciona um novo valor ao topo da pilha[2].
pushad	Movimentação de dados	Armazena o conteúdo dos registros de uso geral na pilha[3].
call	Chamada de função	Usado para denotar uma função a ser chamada[2].
ret	Retorno	Retorna a subrotina. O endereço de retorno é carregado da pilha.
mov	Movimentação de dados	Instrução que move dados entre registros e memória. Esta instrução tem dois operandos: o primeiro é o destino e o segundo especifica a origem[4].
add	Aritmética	Soma dois operandos, armazenando o resultado no primeiro operando. Ambos os operandos podem ser registradores, mas apenas um operando pode ser uma localização de memória[5].
sub	Aritmética	Armazena no primeiro operando o resultado de subtrair o valor do segundo operando com o valor do primeiro operando[4].
cmp	Comparação	Compara os valores dos dois operandos especificados[4].
je	Salto Condicional	Executa um salto para a uma determinada função se o resultado da comparação executada anteriormente for igual[4].
jmp	Salto	Transfere o fluxo de controle do programa para a instrução no local de memória indicado pelo operando[4].
fld	Movimentação de dados	Armazena um valor em ponto flutuante na pilha[5].
fst	Movimentação de dados	Copia o valor que está na parte superior da pilha para uma variável ponto flutuante[5].
fadd	Aritmética	Soma valores em pontos

		flutuantes[5].
fsub	Aritmética	Subtrai valores em pontos flutuantes[5].
fmul	Aritmética	Multiplica valores em ponto flutuante[5].
fdiv	Aritmética	Divide valores em ponto flutuante[5].
fcomip	Comparação	Compara os valores que estão no topo da pilha, atualiza as flags e da o um pop do topo da pilha[5].
fstp	Movimentação de dados	Remove o elemento da pilha podendo transferir ou não esse dado[5].
popad	Movimentação de dados	Remove o conteúdo dos registros de uso geral da pilha[6].
int	Interrupção	Chama uma rotina de serviço de interrupção[7].

Tabela 1- Tabela de instruções utilizadas.

III. REQUISITOS

Para que o software proposto seja executado, deve ser operado em computador com sistema operacional linux com uma arquitetura de 32 bits, o qual é o mesmo que foi usado para implementação do código.

Além disso, o código faz ligação com a linguagem de programação C, o qual utiliza funções printf e scanf da mesma, para a leitura e escrita do programa.

Para que o algoritmo seja executado, é sugerido o uso dos seguintes comandos no terminal:

nasm -f elf -l calculadora.lst calculadora.asm
gcc -m32 calculadora.o -o calculadora
./calculadora

São essas instruções que tornam possível fazer o exercício da calculadora básica em NASM proposta.

IV. RESTRIÇÕES DE USO

Para que o usuário possa manusear sistema de maneira funcional, é indicado que ele siga e se baseie no fluxograma da *Figura 2* que demonstra os passos realizados adequados para que uma pessoa faça uso dele.

Outra restrição se dá ao digitar os números para a realização das operações, sendo indicado a não colocação de caracteres. Do mesmo modo, quando for o momento de digitar um sinal só sera aceito os caracters +, -, /, * sendo informado erro pelo programa, dando a possibilidade de inserção de novo operador.

Como dito no início, a divisão por zero é inválida, sendo impossível de ser calculado e representado. Essa exceção também é tratada no algoritmo informando o erro e dando a possibilidade do usuário digitar um novo número.

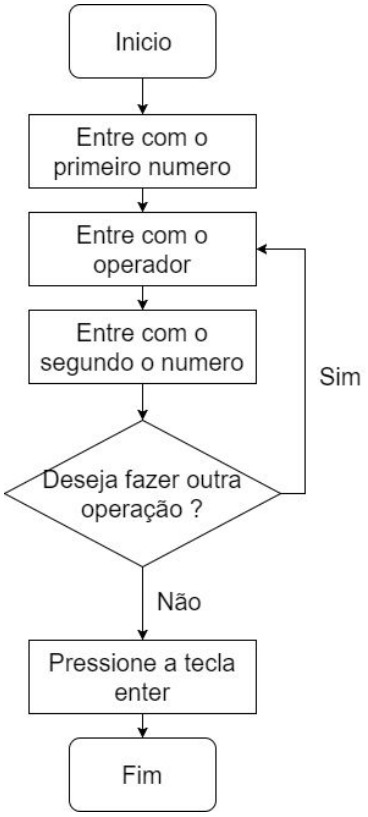

Figura 2- Fluxograma de funcionamento do programa.

V. CAPACIDADE

O programa é capaz de executar, sem falhas, operações entre números de no máximo 22 dígitos, sendo eles representados e armazenados em sistema de ponto flutuante, onde é permitido casas decimais de quantidade variada.

Dado os 22 dígitos, o range dos números de entrada varia entre limites interiores e superiores descritos na *Tabela 2*. Para qualquer número fora desses limites,

mesmo que o algoritmo aceite e o calcule, o resultado obtido será incorreto.

	Limite Inferior	Limite Superior
Negativo	-999999999999999 9999999	-0.00000000000000 0000001
Positivo	0.00000000000000 0000001	9999999999999999 999999

Tabela 2 - Limites de Capacidade.

REFERÊNCIAS BIBLIOGRÁFICAS

[1]Dicionário Priberam. "calculadora", in Dicionário Priberam da Língua Portuguesa [em linha], 2008-2013, Disponível em: <https://www.priberam.pt/dlpo/calculadora> [consultado em 11-12-2017].

[2]NASM.. "NASM — The Netwide Assembler", version 2.09.04; Disponível em:<http://www.nasm.us/xdoc/2.09.04/nasmdoc.pdf> Acessado em: 11 dez. 2017.

[3]NACAD/COPPE-UFRJ. PUSHA/PUSHAD--Push All General-Purpose Registers.<http://www.nacad.ufrj.br/online/intel/vtune/users_guide/mergedProjects/analyzer_ec/mergedProjects/reference_olh/mergedProjects/instructions/instruct32_hh/vc267.htm> Acessado em: 11 dez. 2017.

[4]A. Ferrari, A. Batson, M. Lack, A. Jones, D. Evans; "x86 Assembly Guide"; Program and Data Representation, Spring 2006, University of Virginia Computer Science; Disponível em: <http://www.cs.virginia.edu/~evans/cs216/guides/x86.html> Acessado em: 11 dez. 2017.

[5]R. Hyde, "The Art of Assembly Language", Assembly Language Programming, pp. 1418 , 1996.

[6]NACAD/COPPE-UFRJ. POPA/POPAD--Pop All General-Purpose Registers.<]http://www.nacad.ufrj.br/online/intel/vtune/users_guide/mergedProjects/analyzer_ec/mergedProjects/reference_olh/mergedProjects/instructions/instruct32_hh/vc246.htm > Acessado em: 11 dez. 2017.

[7]S. P. Dandamudi, "Guide to Assembly Language Programming in Linuxi", 545 pp. , 2005.

[8]Weisstein, Eric W. "Division by Zero." From MathWorld--A Wolfram Web Resource. http://mathworld.wolfram.com/DivisionbyZero.html.