Calculadora Básica - Netwide Assembler (NASM)

JESUS, W. P.[#1] e EGITO, H. S[#2]

[#]*Departamento de Computação e Eletrônica, UFES*
São Mateus , Espírito Santo - Brasil

[1]wellerson.prenholato@gmail.com

[2]hadamosieg@gmail.com

***Resumo*— Este trabalho foi realizado utilizando a linguagem de montagem (NASM), para executar as quatro operações básicas de uma calculadora para números reais.**

***Palavras-Chave*— Assembly, NASM, Linguagem de Montagem, Calculadora, GCC.**

I. Introdução

Netwide Assembler (NASM) é um montador e disassembler para a arquitetura Intel x86 e é comumente usado para criar programas de 16 bits, 32 bits (IA-32) e 64 bits (x86-64). Uma montagem transformará sua codificação de baixo nível, usando mnemônicos, em linguagem de máquina que pode ser entendida pelo processador.

Dessa forma utilizamos a linguagem de montagem (NASM), para executar as quatro operações básicas de uma calculadora para números reais. A entrada e saída de dados foram feitas através de chamadas das funções printf e scanf da linguagem C.

O resultado de uma operação passa a ser o primeiro operando da operação seguinte, assim como ocorre em uma calculadora comum.

II. Descrição do Programa

A calculadora criada em NASM utilizando arquitetura de 64 bits e sub rotinas em C para entrada e saída de dados, faz cálculos básicos utilizando números reais. Constam abaixo as características específicas do projeto.

A. Funções Implementadas

Na section .text do código foram implementadas as funções contidas na TABELA I.

TABELA I

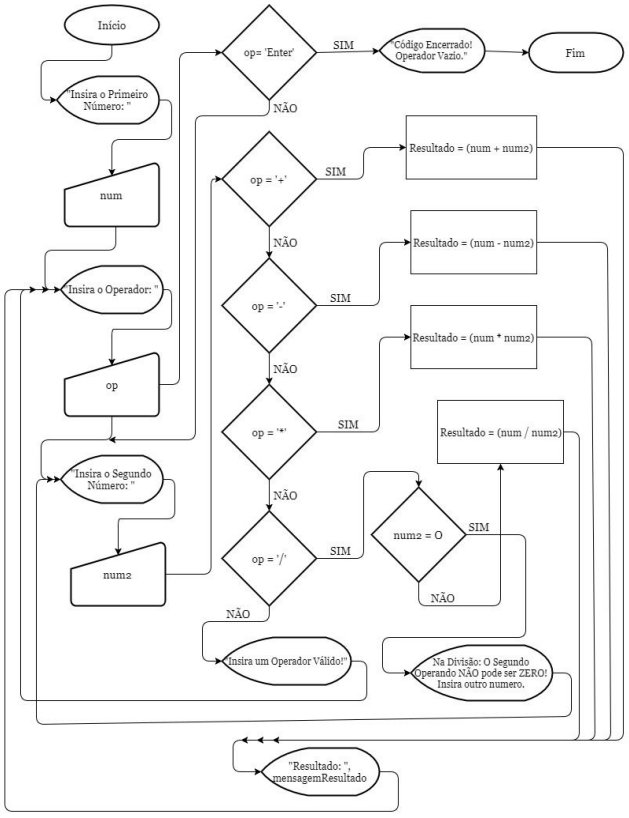
Funções implementadas no programa:

Função	O que ela faz
menu_mensagem	Apresenta a string "*** Calculadora Básica - Linguagem de Montagem (NASM) ***", ao iniciar o projeto.
menu_operacoes	Apresenta as operações que podem ser realizadas no projeto.
ler_op1	Faz a leitura do primeiro número.
ler_operador	Faz a leitura da operação.
ler_op2	Faz a leitura do segundo número.
def_operador	Identifica e define qual operação será feita.
subtracao	Operação de Subtração. Ex: 10-5 = 5.
adicao	Operação de Adição. Ex: 10+5 = 15.
multiplicacao	Operação de Multiplicação. Ex: 10 * 5 = 50.
divisao	Operação de Divisão. Ex: 10 / 5 = 2.
verifica_Operando2_Zero	Quando selecionado a operação de divisão, essa função verifica se o segundo número é zero.

	Caso seja zero, é apresentado uma mensagem de erro logo em seguida é solicitado a inserção de um novo número.
novo_operador	Caso o operador não seja identificado, é apresentado uma mensagem de erro, logo em seguida é solicitado a inserção de um novo operador.
exibir	Apresenta o resultado de cada operação.
fim	Encerramento do código.

IMAGEM I

Fluxograma do Programa.



B. Requisitos

É recomendado que o usuário esteja utilizando o Sistema Operacional Linux, e que tenha instalado Netwide Assembler (NASM), GNU Compiler Collection (chamado usualmente por GCC) e um editor de texto qualquer, dessa forma o usuário conseguirá montar, compilar e executar o projeto.

O projeto apresentado foi elaborado utilizando arquitetura de 64 bits.

C. Restrições de Uso

Para o projeto funcionar adequadamente o usuário deve obedecer algumas regras:

Inicialmente o usuário deve inserir o primeiro número (num), sendo ele inteiro ou real.

Sucessivamente o projeto irá pedir um operador (op) no qual é referente a operação que deseja realizar, em seguida o usuário deverá inserir o segundo número (num2) que será feita tal operação. Sendo esse segundo número também inteiro ou real. Após as inserções serem realizadas o código irá fazer o cálculo numérico, em seguida retornará o resultado referente a operação inserida.

O resultado de uma operação passa a ser o primeiro operando da operação seguinte.

Assim, caso o usuário queira fazer novas operações utilizando o resultado anterior, basta ele inserir uma nova operação e sucessivamente um novo valor para o segundo operando (Num2) como mostrado na IMAGEM II.

TABELA II

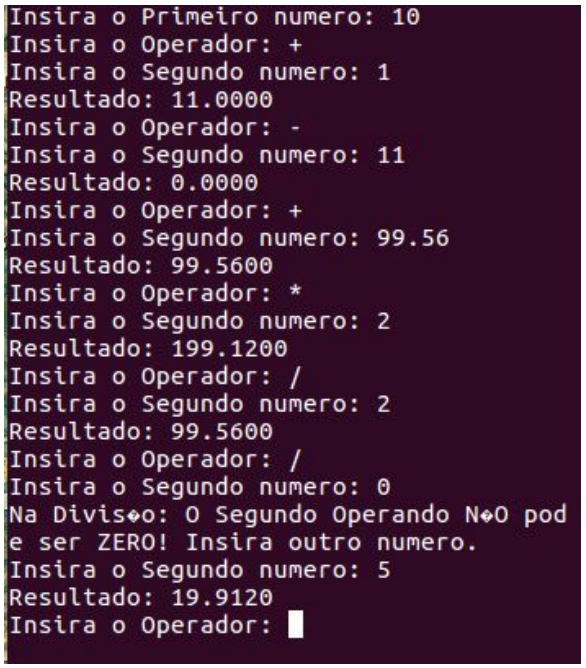
OPERAÇÕES E OPERADORES QUE PODEM SER USADOS:

Caracter	Operação
*	Multiplicação
+	Soma
-	Subtração
/	Divisão

Observação: Caso o usuário não obedeça às regras apresentadas o projeto poderá não funcionar da forma correta, e possivelmente apresentará erros.

IMAGEM II

Exemplo de Execução do Programa.



D. Capacidades

A calculadora pode realizar as quatro operações aritméticas básicas com valores reais, aceitando números em ponto flutuante com precisão de até quatro dígitos à direita do ponto e quinze dígitos à esquerda. Caso o número inserido ultrapasse estes limites, o valor lido poderá conter lixo de memória.

Os valores inseridos nos operandos devem ser números reais e os valores inseridos no operador devem ser os mesmos contidos na TABELA II.

E. Instruções Utilizadas

Na TABELA III constam as instruções pertencentes a NASM, utilizadas na construção do código, seus tipos e breve descrição de suas funcionalidades.

TABELA III

INSTRUÇÕES UTILIZADAS NO PROGRAMA:

Tipo	**Nome**	**Descrição**
Reserva de bytes para dados inicializados	label **db** conteúdo	Reserva 1 byte com valor inicializado
Reserva de byte para dados não inicializados	label **resq** qtd	Reserva 8 bytes para valor não inicializado
Reserva de byte para dados não inicializados	label **resb** qtd	Reserva 1 byte para valor não inicializado
Movimentação de dados	**mov** dest, ori	Move dados ou valor da origem ori para destino dest
Chamada de função externa	**call** função	Chama função de linguagem externa
Operação Aritmética	**sub** a,b	Realiza subtração de inteiros/registers a e b, guardando resultado em a
Operação Aritmética	**add** a,b	Realiza adição de inteiros/registers a e b, guardando resultado em a
Movimentação de dados/valores de ponto Flutuante	**movq** a,b	Move dados/valores em ponto flutuante ou valor da origem ori para destino dest
Operação Aritmética com Ponto Flutuante	**subsd** a,b	Realiza subtração de Ponto Flutuante /registers a e b, guardando resultado em a
Operação Aritmética com Ponto Flutuante	**addsd** a,b	Realiza soma de Ponto Flutuante /registers a e b, guardando resultado em a
Operação Aritmética com Ponto Flutuante	**mulsd** a,b	Realiza multiplicação de Ponto Flutuante /registers a e b, guardando resultado em a
Operação Aritmética com Ponto Flutuante	**divsd** a,b	Realiza divisão de Ponto Flutuante /registers a e b, guardando resultado em a

Operação de Comparação	**cmp** a,b	Compara dois valores inteiros e seta flags para resultado.
Deslocamento de fluxo de execução	**jmp** label de segmento	Realiza pulo para segmento do código
Deslocamento de fluxo de execução	**jne** label de segmento	Realiza pulo para segmento do código se receber flag de igualdade de um cmp
Deslocamento de fluxo de execução	**jz** label de segmento	Realiza pulo para segmento do código se receber flag de igualdade a 0 de um cmp
Interrupção	**int 80h**	Interrompe execução

F. Como executar o Programa.

Suponhamos que o usuário esteja usando o Sistema Operacional Linux.

Inicialmente iremos acessar pelo terminal a pasta onde encontra-se o código fonte .asm.

Sucessivamente iremos fazer a montagem do arquivo fonte, digitando no terminal:

"nasm -f elf64 -l ProjetoCalculadoraAssembly.lst ProjetoCalculadoraAssembly.asm"

Assim foi feito a montagem do arquivo fonte utilizando a linguagem de montagem (NASM) em arquitetura de 64 bits.

Em seguida, iremos compilar o mesmo arquivo utilizando GNU Compiler Collection (GCC), pois estamos utilizando sub rotinas em C (Linguagem de Programação C) para entrada e saída de dados, então iremos digitar no terminal:

"gcc -o ProjetoCalculadoraAssembly ProjetoCalculadoraAssembly.o".

Assim, compilamos o código fonte e criamos um executável. Para executar o código chamado "ProjetoCalculadoraAssembly", basta digitar esse comando no terminal:

"./ProjetoCalculadoraAssembly"

Dessa forma estaremos executando o código, e sucessivamente iremos fazer as devidas inserções de valores.

G. Como encerrar o Programa.

Quando solicitado o operador, basta o usuário apertar "Enter", assim o sistema irá identificar como operador vazio, será apresentado uma mensagem "Código Encerrado! Operador Vazio.", logo em seguida o código é encerrado.

Referências Bibliográficas

[1] "Assembly", Pt.wikipedia.org, 2017. [Online]. Available: https://pt.wikipedia.org/wiki/Assembly. [Accessed: 15- Dec- 2017].

[2] "NASM DOC - The Netwide Assembler", Nasm.us, 2017. [Online]. Available: http://www.nasm.us/doc/. [Accessed: 15- Dec- 2017].

[3] "Stack Overflow - Where Developers Learn, Share, & Build Careers", Stackoverflow.com, 2017. [Online]. Available: https://stackoverflow.com/. [Accessed: 15- Dec- 2017].