



**Unioeste - Universidade Estadual do Oeste do Paraná**

**CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS**

**Colegiado de Ciência da Computação**

***Curso de Bacharelado em Ciência da Computação***

*Docente: Gabriela Stein*

## **Atividade Prática 02 - Calculadora**

*Pedro Henrique de Oliveira Berti*

**CASCADEL**

**2024**

## Descrição do Trabalho:

Escreva um código em assembly que permita a entrada de dois operandos sinalizados em ponto-flutuante de precisão simples e uma operação aritmética. O programa deve retornar a resposta do cálculo de todas as execuções em um arquivo acumulativo.

## Especificações:

1. Desenvolvimento do trabalho em linguagem de montagem para arquitetura AMD/Intel x86\_64 com sintaxe Intel.
2. Código montável, ligável e executável sem erros ou avisos (warnings).

```
;nasm -f elf64 calc.asm  
;gcc -m64 -no-pie calc.o -o calc.x  
;./calc.x
```

3. Número inferior a 400 linhas de código funcional.  
*obs: O Arquivo 'calc.asm' tem 600 linhas, porém grande parte são comentários. São menos de 400 linhas de código.*
4. Permite a entrada de dois operandos inteiros sinalizados em ponto-flutuante de precisão simples e uma operação aritmética utilizando as funções externas C (scanf e printf) para interagir com o usuário.
5. O programa retorna a resposta do cálculo de todas as execuções em um arquivo cumulativo. Todos os testes disponibilizados no arquivo da descrição do trabalho foram executados corretamente.
6. **Entrada de dados no formato:** scanf("%f %c %f") para operando1 operador operando2
7. **Saída nos formatos:** operando1 operador operando2 = resposta
  - a. **execução correta:** printf("%lf %c %lf = %lf\n")
  - b. **execução incorreta:** printf("%lf %c %lf = funcionalidade não disponível\n").

Operador de Entrada:	a	s	m	d	e
Operador de Saída:	+	-	*	/	^

### **Tabela de representação dos operadores implementados:**

Operação	Caractere utilizado para representação*	Descrição
Adição	a	Adição entre op1 e op2
Subtração	s	Subtração do op2 em op1
Multiplicação	m	Multiplicação entre op1 (multiplicando) e op2 (multiplicador)
Divisão	d	Divisão entre op1 (dividendo) e op2 (divisor)
Resto da Divisão	r	Resto da divisão inteira entre op1 (dividendo) e op2 (divisor)
Exponenciação	e	op1 (base) elevado à op2 (expoente)

### **Utilização das funções externas da linguagem C para Assembly:**

1. Printf
2. Scanf
3. Fopen
4. Fclose
5. Fprintf

### **Utilização de instruções sinalizadas para manipulação de valores positivos e negativos:**

1. Movss
2. Movzx
3. Movsx
4. Mulss
5. Divss
6. Cvtss2si

**Stack-frame** obrigatório para todas as operações com o registrador (push e pop) RBP e ret:

```
;O ponteiro de base do registro (rbp) é salvo na pilha com push rbp.  
push rbp  
;rbp é movido para o topo da pilha com mov rbp, rsp.  
;Isso estabelece um novo frame da pilha para a função main.  
mov rbp, rsp
```

```
;Restauram o ponteiro de base do registro (rbp) a partir da pilha  
;e retornam para o ponto de retorno da função com ret.  
;Isso encerra a execução da função e retorna ao ponto de chamada.  
pop rbp  
;Realiza a operação de soma de dois valores  
;em ponto flutuante de precisão simples e retorna o resultado.  
;Usa registradores xmm para manipular os valores de ponto flutuante  
;e armazena o resultado na memória antes de retornar.  
ret
```

## Organização do Código:

### A. Chamada das funções externas do C:

```
;Chamada das funções externas do 'C' para o assembly  
; 'extern' indica que as funções são definidas em outros módulos ou bibliotecas e  
; serão vinculadas durante o processo de ligação do programa final.  
4 references  
extern printf ;escrita stdout  
4 references  
extern scanf ;leitura stdin  
4 references  
extern fopen ;abertura e/ou criação de arquivo  
4 references  
extern fclose ;fechamento do arquivo  
4 references  
extern fprintf ;escrita no arquivo
```

## B. Definição das Sections .data, .bss e .text:

```
;Definições básicas do algoritmo  
;Dados estáticos utilizados no programa.  
;Estrutura necessária para a lógica de entrada  
;e saída do programa e para a interação com arquivos.  
2 references  
section .data ...  
;Variáveis não inicializadas  
2 references  
section .bss ...  
;Código principal do programa é definido.  
2 references  
section .text ...
```

## C. Definição das funções obrigatórias para cada operação:

```
;Função de soma de dois valores em float, dado pela entrada 'a'  
1 reference  
funcao_soma: ...  
;Função de subtração de dois valores em float, dado pela entrada 's'  
1 reference  
funcao_subtracao: ...  
;Função de multiplicação de dois valores em float, dado pela entrada 'm'  
1 reference  
funcao_multiplicacao: ...  
;Função de multiplicação de dois valores em float, dado pela entrada 'd'  
1 reference  
funcao_divisao: ...  
  
;Função de exponenciação de dois valores em float, dado pela entrada 'e'  
1 reference  
funcao_exponenciacao: ...  
;Função comparadora com o expoente sendo que qualquer número elevado a 0 é 1  
;chamando o auxiliar_0 e assim convertendo de inteiro para float  
1 reference  
igual_zero: ...  
1 reference  
menor_zero: ...  
;Função que multiplica um pelo outro na exponenciação, enquanto o valor  
;não for igual a 1, o laço não para  
2 references  
laco_expoente: ...  
;Usada para lidar com o caso em que o expoente é igual a 1.  
;Nesse caso, o código armazena o valor 1 em [aux_1] para indicar que o expoente é 1.  
1 reference  
igual_um: ...
```

**Função de exponenciação possui as subfunções:** igual\_zero, menor\_zero, laco\_expoente e igual\_um.

D. **Definição da main:** stackframe, abertura/criação do arquivo para escrita, syscall write para mensagem stdout, scanf para leitura dos valores stdin, chamadas das funções para operações e jump final para validação de execução do código.

```
;O ponteiro de base do registro (rbp) é salvo na pilha com push rbp.  
push rbp  
;rbp é movido para o topo da pilha com mov rbp, rsp.  
;Isso estabelece um novo frame da pilha para a função main.  
mov rbp, rsp
```

```
;Carregam os endereços das strings nome_arquivo e permissao_escrita  
;nos registradores rdi e rsi, respectivamente.  
;Em seguida, a função fopen é chamada usando call fopen,  
;o que abrirá o arquivo com o nome e a permissão especificados.  
;Salva o nome do arquivo e sua permissão.  
lea rdi, [nome_arquivo]  
lea rsi, [permissao_escrita]  
;Abre o arquivo  
call fopen
```

```
;Função write do sistema operacional para  
;exibir a string definida em entrada_print na saída padrão (stdout).  
;O valor 1 é colocado no registrador rax para indicar que a função write será chamada.  
;O valor 1 é movido para o registrador rdi para indicar que a saída padrão será usada.  
;O endereço da string entrada_print é carregado no registrador rsi,  
;e o tamanho da string entrada_print_l é movido para o registrador edx.  
;A syscall é feita para chamar a função write.  
mov rax, 1  
mov rdi, 1  
lea rsi, [entrada_print]  
mov edx, entrada_print_l  
syscall
```

```

;Carregam os endereços de memória onde os valores de entrada
;serão armazenados nos registradores rcx, rdx e rax.
lea rcx, [rbp-16]
lea rdx, [rbp-17]
lea rax, [rbp-12]

;Os registradores são movidos para os registradores apropriados
;para serem passados como argumentos para a função scanf.
mov rsi, rax
mov edi, entradas
mov eax, 0
;A função scanf é chamada para realizar a leitura dos valores de entrada fornecidos pelo
usuário.
;Chama a função para a entrada dos valores
call scanf

```

```

;Comparação para a adição -> a
cmp eax, 0x61
je adicao

;Comparação para a subtração -> s
cmp eax, 0x73
je subtracao

;Comparação para a multiplicação -> m
cmp eax, 0x6d
je multiplicacao

;Comparação para a divisão -> d
cmp eax, 0x64
je divisao

;Comparação para a exponenciação -> e
cmp eax, 0x65
je exponenciacao

```

```

;Se nenhuma das comparações anteriores for verdadeira,
;o salto incondicional jmp é tomado para a label operacao_final.
;Isso permite que o programa execute a operação final,
;independentemente do caractere de operação lido.
jmp operacao_final

```



E. **Assinatura de Funções:** A label `flagzero` sinaliza divisão por zero, o que deve causar a mensagem de erro.

```
;Função de adição, com a definição dos parametros, sendo acessada  
;pela comparação feita acima pela letra de entrada  
1 reference  
adicao: ...  
;Função de subtração, com a definição dos parametros, sendo acessada  
;pela comparação feita acima pela letra de entrada  
1 reference  
subtracao: ...  
;Função de multiplicação, com a definição dos parametros, sendo acessada  
;pela comparação feita acima pela letra de entrada  
1 reference  
multiplicacao: ...  
;Função de divisão, com a definição dos parametros, sendo acessada  
;pela comparação feita acima pela letra de entrada  
1 reference  
divisao: ...  
;Aciona flag que indica divisao por zero  
1 reference  
flagzero: ...  
;Função de exponenciação, com a definição dos parametros, sendo acessada  
;pela comparação feita acima pela letra de entrada  
1 reference  
exponenciacao: ...
```

F. **Definição das funções para finalização da execução:** `fclose`, `fprintf`, `syscall` para return 0 e operação final.

```
;Função dos dados, caso nenhuma comparação seja feita e os dados estejam  
;na pilha, por meio das instruções, ocorre uma conversão para ponto-flutuante  
7 references  
operacao_final: ...  
;Caso o arquivo esteja certo, sua escritura será feita a partir do  
;fprintf() com sua indentificação 'a+'  
0 references  
escrever: ...  
;Caso o arquivo não esteja certo, não será feito a escritura do  
;resultado mas sim de uma mensagem de 'ERRO' passado para dentro  
;do arquivo  
2 references  
erro_escrever: ...  
;Fecha o arquivo chamando o fclose() com base do exterr  
1 reference  
fecha: ...  
;Finalização classica do algoritmo  
0 references  
fim: ...
```



### **Funcionalidade:**

1. A partir da `.main` o `stackframe` é definido junto com a abertura/criação do arquivo resultado, `syscall write` para a mensagem de entrada (`entrada_print : db "Equação: ", 0`) e `scanf` para leitura `stdin`.
2. A partir disso as comparações são realizadas para disparar a respectiva operação de acordo com o operador “char” inserido no `scanf`.
3. Todas as assinaturas de funções após operações de suas funções definidas retornam para a label `operação_final` onde ocorre as conversões finais e é definido se a escrita no arquivo será feita pelas labels `.escrever(saída good)` ou `.erro_escrever(saída bad)`.
4. Feito isso o arquivo é fechado pela label `.fecha` e o programa encerrado pela label `.fim`.