Explicativo sobre o programa praticas_11.s

O objetivo do programa eh encontrar o produto escalar entre dois vetores de de numeros em ponto flutuante. O produto escalar eh obtido multiplicando os dois vetores, de mesmo tamanho, fazendo cada elemento do primeiro vetor multiplicar pelo elemento correspondente do segundo vetor, e somando todas as multiplicacoes. Os vetores devem ter o mesmo tamanho.

Para gerar o executavel, gere primeiro o objeto executando o seguinte comando:

```
as praticas_11.s -o praticas_11.o
```

e depois link dinamicamente com o seguinte comando:

```
ld praticas_11.o -l c -dynamic-linker /lib/ld-linux.so.2 -o praticas_11
```

O executavel se chamara praticas_11, sem extensão, e para executá-lo digite:

```
./praticas_11
```

```
.section .data

titulo:  .asciz  "\n*** Programa Multiplica Vetores Ponto Flutuante 1.0 ***\n\n"

pedeTam:      .asciz  "\nEntre com o tamanho dos vetores A e B => "
pedeVet:      .asciz  "\nDigite os elementos do vetor %c:\n"

formatoInt:   .asciz  "%d"
formatoReal:  .asciz "%lf"

pedeNum:     .asciz  "\n%c[%d] = "

mostraVet:    .asciz  "\n\nElementos Lidos do Vetor %c:\n"
mostraNum:  .asciz  "\n%.2lf"
mostraPE:     .asciz  "\n\nProduto Escalar = %.2lf\n"

pulaLin:        .asciz  "\n"

maxTam:       .int     50
tam:            .int     0
num:            .double        0
prodEsc:       .double        0

vetorA:         .int    0  # ponteiro para o vetor A
vetorB:        .int     0  # ponteiro para o vetor B

A:               .int     'A'
B:               .int     'B'
vet:             .int     ' '

limpaBuf:      .string   "%*c"    # ou .asciz "%*c"   # para limpar o buffer do teclado

.section .text

.globl _start
```

```
_start:

principal:

        pushl   $titulo
        call    printf
        finit
        call    leTam
        call    alocaVetores
        call    leVetores
        call    printVetores
        call    calcPE
        call    printPE

        jmp     fim

leTam: # rotina que le o tamanho dos vetores (numero de elementos)

        pushl   $pedeTam
        call    printf
        pushl   $tam
        pushl   $formatoInt
        call    scanf

        pushl   $limpaBuf
        call    scanf

        addl    $16, %esp               # descarta os 3 pushls anteriores

        movl    tam, %ecx
        cmpl    $0, %ecx
        jle     leTam
        cmpl    maxTam, %ecx
        jg      leTam

        ret

alocaVetores: # rotina que aloca espaco de memoria para armazenar os 2 vetores

        movl    $8, %eax        # 8 bytes para cada elemento double
        movl    tam, %ecx
        mull    %ecx
        pushl   %eax
        call    malloc          # retorna endereco em %eax
        movl    %eax, vetorA
        call    malloc
        movl    %eax, vetorB

        addl    $4, %esp
        ret

leVetores:   # rotina que realiza a leitura dos vetores

        movl    A, %eax
        movl    %eax, vet

        pushl   %eax
```

```
        pushl   $pedeVet
        call    printf
        movl    vetorA, %edi
        movl    tam, %ecx
        call    leNum

        movl    B, %eax
        movl    %eax, vet

        pushl   %eax
        pushl   $pedeVet
        call    printf
        movl    vetorB, %edi
        movl    tam, %ecx
        call    leNum

        addl    $16, %esp     # remove os 4 ultimos pushls
        ret

leNum:          # rotina que lê os números e os coloca na memória alocada

        pushl   %ecx
        movl    tam, %ebx
        addl    $1, %ebx
        subl    %ecx, %ebx
        pushl   %ebx
        movl    vet, %eax
        pushl   %eax
        pushl   $pedeNum
        call    printf
        pushl   $num
        pushl   $formatoReal
        call    scanf

        pushl   $limpaBuf
        call    scanf

        fldl    num
        fstpl   (%edi)
        addl    $8, %edi        # avanca posicao no vetor

        addl    $24, %esp     # remove os 6 ultimos pushls
        popl    %ecx
        loop    leNum         # decrementa em 1 o %ecx

        ret

printVetores:

        movl    A, %eax
        movl    %eax, vet
        pushl   %eax
        pushl   $mostraVet
        call    printf
        movl    tam, %ecx
        movl    vetorA, %edi
        call    printNros
```

```
        movl    B, %eax
        movl    %eax, vet
        pushl   %eax
        pushl   $mostraVet
        call    printf
        movl    tam, %ecx
        movl    vetorB, %edi
        call    printNros

        addl    $16, %esp     # remove os 4 ultimos pushls
        ret

printNros:

        fldl    (%edi)
        addl    $8, %edi
        pushl   %ecx
        subl    $8, %esp      # abre espaco para o double
        fstpl   (%esp)                # empilha double na pilha normal
        pushl   $mostraNum
        call    printf

        addl    $12, %esp     # remove o ultimo pushl e o double
        popl    %ecx
        loop    printNros

      ret

calcPE:

        movl    $0, prodEsc
        movl    tam, %ecx
        movl    vetorA, %edi
        movl    vetorB, %esi

voltaPE:

        fldl    (%edi)
        fmull   (%esi)
        faddl   prodEsc
        fstpl   prodEsc
        addl    $8, %edi
        addl    $8, %esi
        loop    voltaPE

        ret

printPE:

        fldl    prodEsc
        subl    $8, %esp      # abre espaco para o double
        fstpl   (%esp)                # empilha double na pilha normal

        pushl   $mostraPE
        call    printf
```

```
        addl    $12, %esp     # remove o ultimo pushl e o double

        ret

fim:
        pushl   $0
        call    exit
```

Desafio 1: Altere o programa de forma que ele faça um produto escalar triplo, multiplicando e somando os elementos de 3 vetores.

Desafio Extra Turbo: Altere o programa de forma que ele faça um produto escalar sobre uma quantidade qualquer de vetores.