Explicativo sobre o programa praticas_10b.s

O presente programa calcula áreas de figuras geométricas, a saber: quadrática, triangular e circunferencial, usando medidas em ponto flutuante. Um menu de opções é utilizado. Nesse programa aparece a instrução finit, que inicializa os 3 registradores de gerenciamento da pilha FPU: registrador de status, registrador de controle e registrador de flag.

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

as praticas_10b.s -o praticas_10b.o

e depois link dinamicamente com o seguinte comando:

ld praticas_10b.o -l c -dynamic-linker /lib/ld-linux.so.2 -o praticas_10b

O executavel se chamara praticas_10b, sem extensão, e para executá-lo digite:

./praticas_10b

```
.section .data

titulo:      .asciz    "\nPROGRAMA PARA CALCULAR AREAS\n\n"
menu:        .asciz    "\n1   -   QUADRATICO\n2   -   CIRCUNFERENCIA\n3   -
TRIANGULO\n4 - SAIR\n> "
msgerro:     .asciz    "\nOpcao Invalida!\n"
pedelado1:   .asciz    "\nDigite o lado 1: "
pedelado2:   .asciz    "\nDigite o lado 2: "
pederaio:    .asciz    "\nDigite o raio: "
pedebase:    .asciz    "\nDigite a base: "
pedealtura:  .asciz    "\nDigite a altura: "

mostraarea:  .asciz    "\nArea = %f\n"

resp:        .int      0

lado1:       .float    0
lado2:       .float    0

base:        .float    0
altura:      .float    0

raio:        .float    0

dois:        .float    2

area:        .double   0

formanum:    .asciz    "%d"
formanumf:   .asciz    "%f"

.section .text

.globl _start
```

```
_start:

      pushl  $titulo
      call   printf
      finit             # inicializa a FPU: reseta controle, status,
                        # flag e registradores

menuop:
      pushl  $menu
      call   printf
      pushl  $resp
      pushl  $formanum
      call   scanf

      addl   $16, %esp

      cmpl   $1, resp
      jz     calcquad
      cmpl   $2, resp
      jz     calccirc
      cmpl   $3, resp
      jz     calctri
      cmpl   $4, resp
      jz     fim

      pushl  $msgerro
      call   printf
      jmp    menuop


calcquad:

      pushl  $pedelado1
      call   printf
      pushl  $lado1
      pushl  $formanumf
      call   scanf

      pushl  $pedelado2
      call   printf
      pushl  $lado2
      pushl  $formanumf
      call   scanf

      flds   lado1            # flds lado1
      fmuls  lado2            # flds lado1
                              # fmul %st(0), %st(1)

      subl   $8, %esp         # abre espaco para acomodar o double
      fstpl  (%esp)           # envia o double da pilha F ara a pilha
                              # normal (no espaco aberto)
      pushl  $mostraarea
      call   printf

      addl   $36, %esp        # retira da pilha normal todos os pushls e
                              # o double
      jmp    menuop
```

```
calccirc:

      pushl  $pederaio
      call   printf
      pushl  $raio
      pushl  $formanumf
      call   scanf

      flds   raio
      fmuls  raio
      fldpi
      fmul   %st(1), %st(0)

      subl   $8, %esp
      fstpl  (%esp)
      pushl  $mostraarea
      call   printf

      addl   $24, %esp
      jmp    menuop

calctri:

      pushl  $pedebase
      call   printf
      pushl  $base
      pushl  $formanumf
      call   scanf

      pushl  $pedealtura
      call   printf
      pushl  $altura
      pushl  $formanumf
      call   scanf

      flds   base
      fmuls  altura
      fdivs  dois

      subl   $8, %esp
      fstpl  (%esp)
      pushl  $mostraarea
      call   printf

      addl   $36, %esp
      jmp    menuop

fim:
      pushl  $0
      call   exit

# IMPORTANTE: as instruções Scanf e Printf utilizam o registrador %st(7) internamente,
#             e pode sobreescrever valores neles, causando a perda de dados. Evite
#             usar tal registrador. Não empilhe mais doque 7 valores simultaneamente
#             na FPU para evitar problemas.
```

```
# Mais Instrucoes: Vejam o livro “Professional Assembly Language”
#
# FSTCW control  : grava o registrador de controle da FPU na
#                  variavel “control” que deve ter 2 bytes. Pode-se
#                  tbem copiar no registrador %ax (16 bits). A instrução
#                  FLDCW possui o efeito inverso da operação.
# FSTSW status   : copia o registrador de status da FPU para a variavel
#                  “status” de 2 bytes. Tbem pode-se usar %ax.
# FILDS val1     : carrega o valor inteiro da variável “val1” na FPU
# FISTS int1     : armazena o topo da FPU na variavel “int1” como inteiro
# FST %st(4)     : move %st(0) para outro registrador, no caso, %st(4)
# FXCH %st(4)    : troca os valores entre registradores %st(0) e outro,
#                  no caso, %st(4)
# FCHS       : troca o sinal em %st(0)
# FDIVR      : div reverso
# FSUBR      : sub reverso
# F2XM1      : calcula 2 elevado a potência de %st(0)
# FCOS       : calcula o cosseno de %st(0)
# FSIN       : calcula o seno de %st(0)
# FSQRT      : calcula a raiz quadrada de %st(0)
# FRNDINT    : arredonda %st(0) para o inteiro mais próximo
# FSCALE     : calcula a potencia de %st(0) a %st(1)
# FCOM       : compara %st(0) com %st(1)


# Outras: FIDIV, FIADD, FIMUL, FISUB, FDIBRP, FABS, ...
```

**OBS**: A Pilha FPU é inicializada (zerada) quando um programa é iniciado. A instrução finit pode ser usada para limpar a Pilha FPU quando ela estiver cheia, permitindo começar de novo a sua utilização sem precisa iniciar o programa novamente.

**DESAFIO**: Fazer um programa para encontrar as raizes de uma equação do segundo grau.