Explicativo sobre o programa praticas_10a.s

Este programa experimenta a Unidade de Ponto Flutuante (FPU) para realizar operacoes com numeros “reais”, os quais são compostos por uma parte inteira e parte fracionária. São aqueles números que contém uma vírgula seguida de casas decimais. Podem ser de dois tipos: do tipo single float (4 bytes) e double float (8 bytes). O primeiro tipo é para números de ponto flutuante de simples precisão e o segundo de dupla precisão.

A FPU é uma parte do processador, específica para realizar Operacões com/em Ponto Flutuante, em adição as operações normais, vistas até agora, as quais são chamadas de Operações com Inteiros. Ela possui registradores próprios, chamados de Registradores da FPU, que trabalham como uma pilha, a qual chamaremos aqui de Pilha PFU. A pilha tradicional que opera com instruções push e pop, vista até agora, podemos chamar de Pilha do Sistema. E qualquer outra pilha feita pelo usuário, tal como a prática passada, podemos chamar de Pilha do Usuário.

Para trabalhar sobre números reais, estes devem primeiramente ser levados da memória para os registradores da FPU (Operação de carga - load). Depois, aplica-se sobre esses valores instruões específicas da FPU. Após serem realizadas as operações desejadas, os dados devem ser retirados dos registradores da FPU e enviados novamente para a memória (Operação de armazenamento - store).

Existem 8 registradores da FPU, denominados R0, R1, .. R7. Esses registradores são operados de forma similar a uma pilha, denominada Pilha FPU. Números em ponto flutuante podem ser inseridos e retirados do topo da pilha. Entretanto, os números localizados no meio da pilha podem ser manipulados também. Estes registradores possuem 80 bits (dupla precisão estendida) e quando valores de 32 (single) ou 64 (double) bits sao ali carregados ou retirados, eles são convertidos em tamanho automaticamente pelo hardware.

Para implementar essa funcionalidade, eles são acessados por nomes lógicos. O topo da Pilha FPU é um registrador que logicamente é acessado por %st(0) e abaixo do topo estão os registradores logicamente denominados de %st(1), %st(2) .. até %st(7). %st(7) é a base da Pilha FPU. Todas as operações PFU sao executadas nesses registradores, usando instrucoes especificas executadas na PFU. As instrucoes normais nao funcionam sobre os registradores da FPU.

Para carregar um dado da memoria no topo da pilha PFU usa-se a instrucao fldx, sendo x = s ou l. Assim, temos flds (para single float) ou fldl (para double float). Para armazenar um dado do topo da pilha PFU na memória usa-se a instrucao fstx, sendo x = s ou l. Assim, temos fsts (para single float) ou fstl (para double float). A instrucao fstx nao retira o dado da pilha PFU, apenas o copia. Para retirar deve-se efetuar um pop da pilha FPU, usando fstpx.

Quando um número é carregado na Pilha FPU, ele é apontado por %st(0), e aquele que era apontado por %st(0) passa a ser apontado por %st(1), e aquele que era apontado por %st(1) passa a ser apontado por %st(2) e assim por diante. Da mesma forma, quando um número é removido na Pilha FPU, aquele que era apontado por %st(0) é removido, e aquele que era apontado por %st(1) passa a ser apontado por %st(0), e aquele que era apontado por %st(2) passa a ser apontado por %st(1) e assim por diante.

Durante a inserção ou remoção de números na Pilha FPU, os conteúdos não são deslocados entre os registradores R0 a R7, mas ao invés disso, os nomes lógicos %st(0) a %st(7) é que são reposicionados dentro do conjunto de registradores.

Se a pilha FPU estiver quase cheia (com 8 números), a inserção de um novo elemento sobrescreverá o mais antigo e causará uma exceção. Nesse sentido, a pilha FPU também pode funcionar como uma “Pilha Circular”. Entretanto, recomenda-se não estourar a pilha da FPU, pois seu tratamento é bastante complicado de aprender. Na prática, recomenda-se ir removendo os valores da pilha FPU na medida em que são usados, evitando passar de 7 registradores usados simultaneamente. A inserção do 8º número já causa problema no printf, sendo que a tentativa “printar” o 8º número lido levará a impressão no símbolo “-nan” (**not a number**). A questão é que a funçao printf também utiliza internamente registrador de ponto flutuante para printar números em ponto flutuante.

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

as praticas_10a.s -o praticas_10a.o

e depois link dinamicamente com o seguinte comando:

ld praticas_10a.o -l c -dynamic-linker /lib/ld-linux.so.2 -o praticas_10a

O executavel se chamara praticas_10a, sem extensão, e para executá-lo digite:

./praticas_10a

```
.section .data

pedido1:     .asciz    "\nEntrada de Dados:\n\nDigite A (single float) => 
"
pedido2:     .asciz    "Digite B (double float) => "

mostra1:     .asciz    "\nValor de A = %.4f\n"
mostra2:     .asciz    "Valor de B = %.4lf\n"

mostrasom:   .asciz    "\nOperacoes Realizadas:\n\B + A = %.4lf\n"
mostrasub:   .asciz    "B - A = %.4lf\n"
mostradiv:   .asciz    "B / A = %.4lf\n"
mostramul:   .asciz    "B * A = %.4lf\n"
mostratudo:  .asciz    "\nTudo Junto:\n\nmultiplicacao = %.2lf divisao = 
%.2lf subtracao = %.2lf soma = %.2lf\n\n"

pulalin:     .asciz    "\n"

formato1:    .asciz    "%f"       # para simples precisão
formato2:    .asciz    "%lf"      # para dupla precisão

float1:      .space    4     # aqui tbem pode ser .single      0
float2:      .space    8     # aqui tbem pode ser .double      0


.section     .text
```

```
.globl      _start
_start:
```

1) lê um número ponto flutuante single (4 bytes) e o mostra como um número double (8 bytes)

```
      pushl  $pedido1
      call   printf
      pushl  $float1
      pushl  $formato1
      call   scanf           # le um valor em simples precisao (4 bytes)
      addl   $12, %esp       # limpa a Pilha do Sistema de 3 pushls

      flds   float1          # carrega variavel single float no topo da
                             # Pilha PFU, convertendo 4 bytes em 80 bits

      subl   $8, %esp        # abre espaco de 8 bytes no topo da Pilha
                             # do Sistema

      fstl   (%esp)          # copia do topo da pilha PFU para o topo da
                             # Pilha do Sistema, convertendo 80 bits em 8
                             # bytes

      pushl  $mostra1
      call   printf
       addl  $4, %esp
```

Observacao 1: no trecho de código anterior, foi lido um float de 4 bytes, mas o mesmo foi mostrado como um float de 8 bytes. Isso foi feito porque printf sempre considera 8 bytes para floats, seja formatado com %f ou %lf

Observacao 2: note que para mostrar o número lido com scanf usamos a sequencia de instrucoes fldx+subl+fstx

2) lê um número ponto flutuante single (4 bytes) e tenta mostra-lo como um número single mesmo

```
      pushl  $pedido1
      call   printf
      pushl  $float1
      pushl  $formato1
      call   scanf           # le um valor em simples precisao (4 bytes)
      addl   $12, %esp       # limpa a Pilha do Sistema de 3 pushls

      flds   float1         # carrega variavel single float no topo da
                             # Pilha PFU, convertendo 4 bytes em 80 bits

      subl   $4, %esp        # abre espaco de 4 bytes no topo da Pilha
                             # do Sistema

      fsts   (%esp)          # copia do topo da pilha PFU para o topo da
                             # Pilha do Sistema, convertendo 80 bits em 4
                             # bytes

      pushl  $mostra1        # como printf sempre considera 8 bytes, aqui
      call   printf          # ocorrerá um erro na impressão
      addl   $4, %esp
```

Observacao: note que aqui tambem mostramos o número lido com scanf usando a sequencia de instrucoes fldx+subl+fstx

3) Le e Soma 2 numeros em ponto flutuante: um single e outro double

```
      pushl  $pedido1
      call   printf
      pushl  $float1
      pushl  $formato1
      call   scanf           # le um valor em simples precisao (4 bytes)
      addl   $12, %esp       # limpa a Pilha do Sistema de 3 pushls

      pushl  $pedido2
      call   printf
      pushl  $float2
      pushl  $formato2
      call   scanf           # le outro valor em dupla precisao (8 bytes)
      addl   $12, %esp       # limpa a Pilha do Sistema de 3 pushls

      flds   float1          # carrega variavel single float no topo da
                             # Pilha PFU, convertendo 4 bytes em 80 bits

      fldl   float2          # carrega variavel double float no topo da
                             # Pilha PFU, convertendo 8 bytes em 80 bits

      fadd   %st(1), %st(0)  # faz %st(0) + %st(1) e sobrescreve
                             # em %st(0)
      subl   $8, %esp
      fstpl  (%esp)          # remove (pop) da Pilha PFU para a Pilha
                             # do Sistema. O float1 fica no topo

      pushl  $mostrasom
      call   printf
      addl   $4, %esp
```

4) Subtrai o numero single do float. Observe que float1 ainda esta no topo

```
      fldl   float2          # recarrega float2 no topo da Pilha PFU
      fsub   %st(1), %st(0)  # faz %st(0) - %st(1) e sobrescreve
                             # em %st(0)
      subl   $8, %esp
      fstpl  (%esp)          # remove (pop) da Pilha PFU para a Pilha do
                              # Sistema. O float1 fica no topo

      pushl  $mostrasub
      call   printf
      addl   $4, %esp
```

5) Divide o numero double pelo single. Observe que float1 ainda esta no topo

```
      fldl   float2          # recoloca float2 no topo da pilha PFU
      fdiv   %st(1), %st(0)  # faz %st(0) / %st(1) e sobrescreve
                             # em %st(0)
      subl   $8, %esp
      fstpl  (%esp)          # remove (pop) da Pilha PFU para a Pilha
```

```
                        # do Sistema.  O float1 fica no topo

      pushl  $mostradiv
      call   printf
      addl   $4, %esp
```

6) Multiplica o numero double pelo single. Observe que float1 ainda esta no topo

```
      fldl   float2          # recoloca float2 no topo da Pilha PFU
      fmul   %st(1), %st(0)  # faz %st(0) * %st(1) e sobrescreve em
                             # %st(0)
      subl   $8, %esp
      fstpl  (%esp)          # remove (pop) da Pilha PFU para a Pilha
                             # do Sistema

      pushl  $mostramul
      call   printf
      addl   $4, %esp
```

7) Mostra tudo

```
      pushl  $mostratudo
      call   printf
      addl   $4, %esp
```

Finaliza o programa

```
      addl   $32, %esp       # remove os quatro numeros double empilhados
      pushl  $0
      call   exit
```

Extra: Instrucoes que colocam valores fixos na pilha de PFU

```
fld1        : coloca 1
fldz        : coloca 0
fldpi       : coloca pi (3.1415...)
fldlg2      : coloca log de 2 na base 10
fldln2      : coloca log de 2 na base e
fldl2t      : coloca log de 10 na base 2
fldl2e      : coloca log de e na base 2
```

DESAFIO 1: Nesse programa, para mostrar um numero lido com scanf, o carregamos na FPU e depois o armazenamos na pilha do sistema para o printf mostra-lo. Isso só é estritamente necessário quando lemos um single e queremos mostrar um double, pois nesse caso precisamos converte-lo de 4 para 8 bytes de precisao e somente a FPU pode fazer. Mas para ler e mostrar um double podemos usar a instrucao “movl”, movendo o dado lido diretamente para a pilha do sistema. Na etapa 3 desse teste, mostre o float2 lido usando a instrucao movl ao invés do sequencia fldl+subl+fstl. Observe que cada movl move apenas 4 bytes e o double possui 8 bytes. Assim, use dois movls seguidos, mas atente-se na ordem.

DESAFIO 2: Fazer um programa para calcular áreas de figuras geométricas: quadrática, triângulo e circunferência.