

## Explicativo sobre o programa praticas\_10a.s

Este programa experimenta a Unidade de Ponto Flutuante (FPU) para realizar operações com números "reais", os quais são compostos por uma parte inteira e outra fracionária. São aqueles números contêm uma vírgula seguida de casas decimais. Podem ser de dois tipos: do tipo single (4 bytes) e double (8 bytes). O primeiro é chamado de simples precisão e o segundo de dupla precisão.

A FPU é uma parte do processador específica para realizar Operações 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 dados reais, estes devem ser levados da memória para os registradores da FPU. Após serem realizadas as operações desejadas, os dados devem ser retirados dos registradores da FPU para a memória. Operações denominadas de carga (load) e armazenamento (store), respectivamente.

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

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). Todas as operações PFU são executadas nesses registradores, usando instruções específicas executadas na PFU. As instruções normais não funcionam sobre os registradores da FPU.

Para carregar um dado da memória no topo da pilha PFU usa-se a instrução 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 instrução fstx, sendo x = s ou l. Assim, temos fsts (para single float) ou fstl (para double float). A instrução fstx não retira o dado da pilha PFU. Para retirar deve-se efetuar um pop da pilha FPU, usando fstpx.

Quando um dado 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 dado 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 dados 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 cheia (com 8 elementos), 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.

Para gerar o executável, 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 executável 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 = %.10lf\n"
```

```
mostra2:      .asciz      "Valor de B = %.10lf\n"
```

```
mostrasom:    .asciz      "\nOperacoes Realizadas:\n\nB + 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"
```

```
formato2:     .asciz      "%.1f"
```

```
float1:       .space      4
```

```
float2:       .space      8
```

```
.section      .text
```

```
.globl        _start
```

```
_start:
```

le e mostra dois numeros em ponto flutuante, um single (4 bytes) e um 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

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

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

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

pushl $mostra1
call  printf          # printf sempre considera 8 bytes, seja
                     # formatado com %f ou %lf

addl  $4, %esp

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

fstl  (%esp)          # copia da pilha PFU para a Pilha do Sistema
                     # convertendo 80 bits em 4 bytes

pushl $mostra1
call  printf          # como printf sempre considera 8 bytes, aqui
                     # ocorrerá um erro

addl  $4, %esp

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

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

fstl  (%esp)          # copia da Pilha PFU para a Pilha do Sistema
                     # convertendo 80 bits em 8 bytes

pushl $mostra2
call  printf          # printf sempre considera 8 bytes, seja
                     # formatado com %f ou %lf

addl  $4, %esp

```

Soma os 2 numeros lidos

```
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

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

Subtrai os 2 numeros lidos

```
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

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

Divide os 2 numeros lidos

```
fldl    float2           # recoloca float2 no topo da pilha PFU
fddiv   %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   $mostradiv
call    printf
addl    $4, %esp
```

Multiplica os 2 numeros lidos

```
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
```

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