Explicativo sobre o programa praticas_09b.s

Este programa modifica o código do exemplo anterior (que aloca o espaço do registro, lê os campos e depois mostra os campos lidos, usando alocação dinâmica com a função malloc da biblioteca libc) e o transforma num programa de manipulação de pilha. Uma variável ptreg foi criada para apontar para endereço do novo registro que está sendo criado e uma variável ptpilha foi criada para apontar para o topo da pilha (ultimo registro encadeado).

Um laco foi utilizado para repetir o processo de alocação e leitura de registros. Cada novo registro é ligado na pilha até então estabelecida (encadeamento dos registros anteriores). O ultimo registro inserido fica sendo o topo da pilha, o qual é controlado pela variável ptpilha. Para a remoção do topo da pilha (ultimo registro) usa-se a chamada free da biblioteca libc para liberar a memória, que somente requer que o endereço do registro esteja na pilha do sistema.

Para mostrar todos os elementos da pilha, o programa caminha pelo encadeamento de registros, iniciando por ptpilha e avança registro por registro até atingir o ultimo, o qual aponta para NULL. Remover um registro significa saltar o ptpilha para o próximo registro e dealocar o espaço de memória do registro saltado.

Não confunda a pilha que foi implementada com a pilha do sistema, a qual funciona automaticamento com as instruções pushl e popl e o tpo é controlado pelo registrador %esp.

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

```
as praticas_09b.s -o praticas_09b.o
```

e depois link dinamicamente com o seguinte comando:

```
ld praticas_09b.o -l c -dynamic-linker /lib/ld-linux.so.2 -o praticas_09b
```

O executavel se chamara praticas_09a, sem extensão, e para executá-lo digite:

```
./praticas_09b
```

```
.section .data

# definicao das mensagens do programa

titgeral:    .asciz    "\n*** APLICACAO DE PILHA ***\n\n"
titemp:      .asciz    "\nEMPILHAMENTO:\n"
titdesemp:   .asciz    "\nDESEMPILHAMENTO:\n"
titmostra:   .asciz    "\nELEMENTOS DA PILHA:\n"
titreg:      .asciz    "\nRegistro no %d:"

menu:        .asciz    "\nESCOLHA    A    OPCAO:\n1    -    EMPILHA\n2    -
DESEMPILHA\n3 - MOSTRA\n4 - FIM\n> "

msgerro:     .asciz    "\nOPCAO INCORRETA!\n"
msgvazia:    .asciz    "\nPILHA VAZIA!\n"
msgremov:    .asciz    "\nREGISTRO DESEMPILHADO!\n"
msginser:    .asciz    "\nREGISTRO EMPILHADO!\n"
```

```
pedenome:    .asciz    "\nDigite o nome: "
pedera:      .asciz    "Digite o ra: "
pedesexo:    .asciz    "Qual o sexo, <F>eminino ou <M>asculino?: "
pedecurso:   .asciz    "Digite o nome do curso: "

mostranome:  .asciz    "\nNome: %s"
mostrara:    .asciz    "\nRA: %d"
mostrasexo:  .asciz    "\nSexo: %c"
mostracurso: .asciz    "\nCurso: %s\n"

mostrapt:    .asciz    "\nptreg = %d\n"

formastr:    .asciz    "%s"
formach:     .asciz    "%c"
formanum:    .asciz    "%d"

pulalinha:   .asciz    "\n"

NULL:        .int 0

opcao:       .int 0

# identificacao de variaveis (campos) a serem utilizados nos registros.
# total de 84 bytes incluindo o campo de proximo

nome:        .space    44  # 1 espaco para fim de string: '\0' = 0
ra:          .space    8
sexo:        .space    4
curso:       .space    24   # usar 23. reservar 1 espaco para fim de
string
prox:        .int NULL

naloc:       .int 84
ptpilha:     .int NULL
ptreg:       .int NULL


.section .text

.globl       _start
_start:

             jmp main

A funcao abaixo espera que o endereco inicial da memoria alocada esteja
em %edi. Entao, ela le nome, RA, sexo e curso e coloca na respectiva
memoria

le_dados:
             pushl     %edi        # endereco inicial do registro

             pushl     $pedenome
             call      printf
             addl      $4, %esp
             call      gets

             popl      %edi        # recupera %edi
```

```
            addl      $44, %edi   # avanca para o proximo campo
            pushl     %edi        # armazena na pilha

            pushl     $pedera
            call      printf
            addl      $4, %esp
            pushl     $formanum
            call      scanf
            addl      $4, %esp

            popl      %edi     # recupera %edi
            addl      $8, %edi   # avanca para o proximo campo
            pushl     %edi       # armazena na pilha

            pushl     $formach   # para remover o enter
            call      scanf
            addl      $4, %esp

            pushl     $pedesexo
            call      printf
            addl      $4, %esp
            pushl     $formach
            call      scanf
            addl      $4, %esp

            popl      %edi     # recupera %edi
            addl      $4, %edi   # avanca para o proximo campo
            pushl     %edi       # armazena na pilha

            pushl     $formach   # para remover o enter
            call      scanf
            addl      $4, %esp

            pushl     $pedecurso
            call      printf
            addl      $4, %esp
            #pushl    $formastr
            call      gets
            #addl     $4, %esp

            popl      %edi      # recupera %edi
            addl      $24, %edi   # avanca para o proximo campo
            movl      $NULL, (%edi)

            subl      $80,%edi    # deixa %edi tal como estava no inicio


            RET

# a funcao abaixo mostra os campos da memoria apontada por %edi, a saber:
# nome, RA, sexo e curso e coloca na memoria apontada %edi

mostra_dados:
            pushl     %edi      # endereco incial do registro, contendo
todos os campos

            pushl     $mostranome
            call      printf
```

```
                addl      $4, %esp

                popl      %edi     # recupera %edi
                addl      $44, %edi  # avanca para o proximo campo
                pushl     %edi       # armazena na pilha

                pushl     (%edi)
                pushl     $mostrara
                call      printf
                addl      $8, %esp

                popl      %edi     # recupera %edi
                addl      $8, %edi   # avanca para o proximo campo
                pushl     %edi       # armazena na pilha

                pushl     (%edi)
                pushl     $mostrasexo
                call      printf
                addl      $8, %esp

                popl      %edi     # recupera %edi
                addl      $4, %edi   # avanca para o proximo campo
                pushl     %edi       # armazena na pilha

                pushl     $mostracurso
                call      printf
                addl      $4, %esp

                popl      %edi      # recupera %edi

                subl      $56,%edi    # deixa %edi tal como estava no inicio


                RET

    empilha:
                pushl     $titemp
                call      printf

                movl      naloc, %ecx
                pushl     %ecx
                call      malloc
                movl      %eax, ptreg

                pushl     ptreg
                pushl     $mostrapt
                call      printf

                addl      $16, %esp

                movl      ptreg, %edi
                call      le_dados

                movl      ptpilha, %eax
                movl      %eax, 80(%edi)
                movl      %edi, ptpilha

                pushl     $msginser
```

```
                call      printf
                addl      $4, %esp

                jmp       menuop

desempilha:
                pushl     $titdesemp
                call      printf
                addl      $4, %esp

                movl      ptpilha, %edi
                cmpl      $NULL, %edi
                jnz       continua

                pushl     $msgvazia
                call      printf
                addl      $4, %esp

                jmp       menuop

continua:

                movl      ptpilha, %edi
                pushl     %edi
                movl      80(%edi), %edi
                movl      %edi, ptpilha

                pushl     $msgremov
                call      printf
                addl      $4, %esp

                call      free
                addl      $4, %esp


                jmp       menuop

mostrapilha:

                pushl     $titmostra
                call      printf

                movl      ptpilha, %edi
                cmpl      $NULL, %edi
                jnz       continua2

                pushl     $msgvazia
                call      printf
                addl      $4, %esp

                jmp       menuop

continua2:
                movl      ptpilha, %edi
                movl      $1, %ecx
volta:
                cmpl      $NULL, %edi
                #cmpl     $3, %ecx
```

```
                jz          menuop

                pushl       %edi

                pushl       %ecx
                pushl       $titreg
                call        printf
                addl        $4, %esp

                movl        4(%esp), %edi   # recupera %edi sem desempilhar
                call        mostra_dados

                popl        %ecx
                incl        %ecx
                popl        %edi
                movl        80(%edi), %edi

                jmp         volta

                jmp         menuop

menuop:
                pushl       $menu
                call        printf

                pushl       $opcao
                pushl       $formanum
                call        scanf

                addl        $12, %esp

                pushl       $formach   # para remover o enter
                call        scanf
                addl        $4, %esp

                cmpl        $1, opcao
                jz          empilha
                cmpl        $2, opcao
                jz          desempilha
                cmpl        $3, opcao
                jz          mostrapilha
                cmpl        $4, opcao
                jz          fim

                pushl       $msgerro
                call        printf
                addl        $4, %esp

                jmp         menuop


main:

                pushl       $titgeral
                call        printf
                jmp         menuop
```

```
fim:

                pushl     $0
                call exit
```