Explicativo sobre o programa praticas_09a.s

Este programa testa o uso da função malloc, da biblioteca libc. Tal função pode ser usada para alocar memória dinamicamente. A quantidade de memoria (em bytes) deve ser passada no topo da pilha, antes de chamar a função com call. O endereço da região alocada é retornada no registrador %eax (endereço do primeiro byte).

No exemplo aqui apresentado, a função malloc é usada para alocar os bytes para um registro (conjunto de campos) acadêmico. Os campos são lidos, inseridos na região alocada, e depois lido e apresentado na tela.

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

as praticas_09a.s -o praticas_09a.o

e depois link dinamicamente com o seguinte comando:

ld praticas_09a.o -l c -dynamic-linker /lib/ld-linux.so.2 -o praticas_09ª

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

./praticas_09a

```
.section .data

definicao das mensagens do programa

titulo:      .asciz    "\nTeste de alocacao de memória para registro\n\n"
pedenome:    .asciz    "\nDigite o nome: "
pedera:      .asciz    "\nDigite o ra: "
pedesexo:    .asciz    "\nQual o sexo, <F>eminino ou <M>asculino?: "
pedecurso:   .asciz    "\nDigite o nome do curso: "

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

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

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

pulalinha:   .asciz    "\n"

NULL:        .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 extra para fim de string: '\0' = 0
ra:          .space    8
sexo:        .space    4
```

```
curso:      .space    24  # 1 espaco extra para fim de string
prox:       .int NULL

naloc:      .int 84
ptlista1:   .int 0
ptlista2:   .int 0
ptlista3:   .int 0
ptlista4:   .int 0
ptlista5:   .int 0

.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 incial 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

            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

main:

                pushl     $titulo
                call      printf

                movl      naloc, %ecx
                pushl     %ecx
                call      malloc
                movl      %eax, ptlista1

                pushl     ptlista1
                pushl     $mostrapt
                call      printf

                addl      $16, %esp

                movl      ptlista1, %edi
                call      le_dados
                call      mostra_dados

fim:


                pushl     $0
                call      exit
```