Praticas 04a

O objetivo do programa é somar 3 valores da memória de duas diferentes formas. Primeiramente, os 3 valores são apontados por 3 rótulos diferentes. A segunda forma, os 3 valores são apontados por um mesmo rótulo. O resultado é colocado em um buffer e é mostrado com printf.

Note que os rótulos funcionam como variáveis. O uso do rótulo implica nouso do valor que ele aponta, mas não do seu endereco. Para acessar o endereço do rótulo deve-se usar o caractere $ antes do nome do rótulo.

Note tambem que o acesso a algum dado de memoria pode ser feito com o uso de um registrador entre parênteses, contendo o endereço que se deseja acessar, como por exemplo (%edi), mas não pode ser feito com rótulos, por exemplo ($res), pois o endereço de memória associado ao rótulo é uma constante.

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

as praticas_04a.s -o praticas_04a.o

e depois link dinamicamente com o seguinte comando:

ld praticas_04a.o -l c -dynamic-linker /lib/ld-linux.so.2 -o praticas_04a

Observe que a necessidade de linkagem dinâmica ocorre pelo fato do programa chamar a função "printf" que esta contida dentro de uma biblioteca ja compilada e que precisar ser ligada em tempo de execução.

O parametro -l x serve para informar a biblioteca libx.so.v que deve existir localmente. Neste exemplo, x = c, pois a biblioteca é a libc.so.v, onde v eh a versao, por exemplo, libc.so.6

O parametro -dynamic-linker informa quem será o software linkador, e no presente exemplo será o /lib/ld-linux.so.2

Caso tenha curiosidade, saiba mais digitando ld -help

O executável se chamará praticas_04, sem extensao, e para executá-lo digite:

./praticas_04a

```
.section .data

     saida:      .asciz      "Soma: %d + %d + %d = %d\n"

     n1:         .int        10
     n2:         .int        25
     n3:         .int        5

     v1:         .int        10, 25, 5

.section    .bss
```

```
.lcomm res, 4


.section   .text

.globl     _start
_start:
```

o trecho a seguir soma os valores dos 3 rótulos n1, n2, n3 e coloca o resultado no rótulo res

```
      movl    n3, %eax
      addl    n2, %eax
      addl    n1, %eax
      movl    %eax, res
```

O trecho a seguir coloca os valores, que serão inseridos nos caracteres de formatacao da string de saida, na pilha. O comando printf é então chamado.

```
      pushl   res
      pushl   n3
      pushl   n2
      push    n1
      pushl   $saida

      call    printf
      addl    $20, %esp
```

O trecho a seguir soma os 3 valores dos rótulo v1, que funciona como um vetor e coloca o resultado no rótulo res. O endereço inicial do vetor é colocado em %edi e as diferentes posições são acessadas como deslocamento neste vetor.

```
      movl    $v1, %edi
      movl    (%edi), %eax
      addl    $4, %edi
      addl    (%edi), %eax
      addl    $4, %edi
      addl    (%edi), %eax

      movl    %eax, res
```

Novamente, o trecho a seguir coloca os valores, que serão inseridos nos caracteres de formatação da string de saida, na pilha. O comando printf é então chamado.

```
      pushl   res
      pushl   n3
      pushl   n2
      pushl   n1
      pushl   $saida
      call    printf

      pushl   $0
      call    exit
```

DESAFIO 1: Altere o final do código de forma que os valores de impressão sejam obtidos do vetor v1 e não das variáveis n1, n2 e n3.

DESAFIO 2: Altere o programa de forma a acrescentar mais duas variáveis, a n4 e n5. Altere a somatória para "n1+n2-n3+n4-n5". Use a instrução "***subl op1, op2***" para fazer: op2 ← op2 – op1. Ajuste a cadeia impressa no vídeo.