Praticas 03a

O objetivo deste programa é o de obter o nome do fabricante do processador usando a instrução *cpuid* e escrever o resultado na tela usando a função *printf* da biblioteca *libc*

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

as praticas_03a.s -o praticas_03a.o

e depois link dinamicamente com o seguinte comando:

ld praticas_03a.o -l c -dynamic-linker /lib/ld-linux.so.2 -o praticas_03a

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

O parametro ***-l x*** serve para informar a biblioteca libx.so.v que deve existir no diretório local. Neste exemplo, x = c, pois a biblioteca é a libc.so.6, versão 6.

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

Caso tenha dúvidas, saiba mais digitando ***ld -help***

O executavel se chamará praticas_03a, sem extensão, e para executá-lo digite:

./praticas_03a

A seguir uma *string* é declarada. O rótulo *output* aponta para ela. Observe que a diretiva agora é a *.asciz*, ao inves de *.ascii*, que já coloca na *string* um caractere nulo (final de string) automaticamente, conforme obriga a função *printf*.

```
.section .data

     output:  .asciz    "O ID do fabricante eh 'xxxxxxxxxxxx'\n"

.section .text

.globl _start

_start:

     movl    $0, %eax
     cpuid
```

Coloca os dados gerados pela instrução *cpuid* na string apontada pelo rótulo output

```
     movl    $output,%edi
     movl    %ebx, 23(%edi) # posição 23 da area apontada
     movl    %edx, 27(%edi) # posição 27 da area apontada
     movl    %ecx, 31(%edi) # posição 31 da area apontada
```

O trecho a seguir imprime a string no vídeo, usando a função printf. Tal função, quando chamada, pega no topo da pilha o endereço inicial da memória onde se encontra a string e a imprime até encontrar o caractere de final de string, mas nao remove tal endereço da pilha. Como o topo da pilha é apontado pelo registrador %esp, basta adicionar $4 em seu valor para desconsiderar o elemento empilhado. Atente-se que a pilha cresce do endereço mais alto em direção ao endereço mais baixo, ou seja, o %esp diminui 4 bytes a cada inserção. Por isso que somar $4 no %esp causa o mesmo efeito que remover da pilha. Outra possibilidade seria usar a instrucao popl que remove o topo da pilha e ajusta automaticamente %esp.

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

A seguir, o código chama a função exit presente na mesma biblioteca, que também obtém da pilha o código de término bem sucedido.

```
pushl   $0
call    exit
```