Praticas 02

O objetivo do programa é obter o nome do fabricante do processador usando a instrução *cpuid* e escrever o resultado na tela usando uma Chamada ao Sistema Operacional (***system call***), via interrupção de software **0x80**.

Primeito escreva o programa fonte chamado de "pratica_02.s", com as partes de codificação distribuídas nesta explicação, depois, para gerar o executavel, gere primeiro o objeto usando o seguinte comando:

as praticas_02.s -o praticas_02.o

e depois *link* estaticamente com o seguinte comando:

ld praticas_02.o -o praticas_02

O executavel terá o nome *praticas_02*, sem extensão, e para executá-lo digite:

./praticas_02

Vamos então escrever o programa fonte. Novamente, abra um editor de texto e siga os passos a seguir.

Inicie a seção de dados e declare uma variável do tipo *string* chamada de *output*, conforme segue. Observe que o nome da *string* é um rótulo e ela já é inicializada.

```
.section .data

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

Saiba que cada caractere ocupa um *byte* de memória, assim, a *string* conterá tantos *bytes* for o número de caracteres. O rótulo que nomeia a *string*, na verdade, marca o primeiro caractere da *string*.

Agora, inicie a seção de código digitando o que segue. Lembre-se que o rótulo "*_start*" marca o ponto de início do programa.

```
.section .text

.globl _start

_start:

     movl  $0, %eax
     cpuid
```

O trecho a seguir coloca os dados gerados pela instrução *cpuid*, os quais são retornados em 3 registradores, na *string* apontada pelo rotulo *output*, nas posições adequadas. Note que a *string* será composta pela concatenacao dos 3 registradores %ebx, %edx e %ecx

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

O trecho a seguir imprime a string no video, usando Chamada ao Sistema Operacional, com os seguintes parâmetros:

```
# %eax contém o numero da system call
# %ebx contém o descritor de arquivo a escrever
# %ecx contém o início da string
# %edx contém o tamanho da string

      movl    $4, %eax
      movl    $1, %ebx
      movl    $output, %ecx
      movl    $37, %edx
      int     $0x80
```

Para finalizar o programa, execute a chamada ao sistema *exit*(código 1 no %eax), passando código 0 no %ebx para indicar término bem sucedido da aplicação.

```
      movl    $1, %eax
      movl    $0, %ebx
      int     $0x80
```

DESAFIO 1: Troque a inicialização da *string output* por: "*O fabricante 'xxxxxxxxxxxx' foi identificado*\n" e ajuste o restante do programa para que o mesmo funcione adequadamente. Monte-o novamente, gere o executável e verifique se funcionou. Se funcionou, passe para o próximo desafio, senão, tente detectar o erro e faça o programa funcionar.

DESAFIO 2: Insira 4 pontos de checagem espalhados no programa fonte, monte-o novamente com a opção *-gstabs*. Agora, execute o código no *gdb*. No *gdb* descubra como checar as posições de memória da *string output*. Teste os seguintes *comandos:*

```
(gdb) x /b &output <enter>
(gdb) x /c &output <enter>
(gdb) x /c &output+1 <enter>
(gdb) x /c &output+2 <enter>
(gdb) x /c &output+3 <enter>
(gdb) x /8c &output <enter>
(gdb) x /16c &output <enter>
(gdb) x /40c &output <enter>
```