Praticas 03b

Este programa possui o mesmo objetivo que a Praticas 03a, só diferindo no uso de junção de strings. Nesse exemplo, criamos um buffer para armazenar o nome do fabricante, separado da string de saida. O buffer eh criado em uma secao especial definida pela diretiva .bss, usando a diretiva (pseudo operação) .lcomm (local common section) e o identificador/nome fabri, e informando a quantidade de bytes, no caso, 16.

Como efeito, o sistema aloca uma área de memória de 16 bytes apontada pela variável fabri. Mas porque 16 bytes?

O nome do fabricante requer 12 bytes (para 12 caracteres). Como é necessário inserir o caracter nulo (fim de string), o qual é representado pelo “\0”, cujo código asc é o próprio 0 (zero), torna-se necessário mais um byte, totalizando 13 bytes. Como os dados são manipulados de 32 em 32 bits (4 bytes), usamos 16 bytes, ou seja, o primeiro multiplo de 4 capaz de conter os 13 bytes.

```
.section .data

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

.section .bss

.lcomm fabri, 16

.section .text

.globl _start
_start:

     movl    $0, %eax
     cpuid
```

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

```
     movl    $fabri,%edi
     movl    %ebx, 0(%edi) # posicao 0 da area apontada
     movl    %edx, 4(%edi) # posicao 4 da area apontada
     movl    %ecx, 8(%edi) # posicao 8 da area apontada
     movl    $0, 12(%edi)  # caracter final de string na 13o posicao
                           # mas que ocupa os 4 ultimos bytes
```

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. Se a string tiver caracteres especiais de formatação (%d, %c, %s etc), quando encontrar algum, a função printf buscará os endereços dos dados associados nos proximos elementos da pilha, seguindo a ordem em que eles aparecem na string principal.

Conforme ja mencionado, o printf não remove os elementos da pilha. Como o topo da pilha é apontado pelo registrador %esp, basta usar a instrução popl para remover os elementos ou adicionar $8 no registrador que aponta pro topo da pilha (%esp). Nesse exemplo, usamos a instrução popl, mas um addl seria mais rápido

```
    pushl   $fabri
    pushl   $output
    call    printf
    popl    %eax
    popl    %eax
```

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

```
    pushl   $0
    call    exit
```