Praticas 07d

O objetivo do programa eh encontrar um subvetor (sequencia menor de elementos) dentro de um vetor (sequencia maior). O programa é construido a partir do exemplo da Praticas 07c e, portanto, as rotinas também sáo chamadas com call. A ideia é a seguinte:

Dados duas sequencias (vetores) informadas pelo usuário, o programa passa a procurar a ocorrência da menor dentro da maior, ou seja, tenta encontrar uma correspondência, elemento a elemento, do subvetor dentro do vetor, de forma que os elementos do subvetor estejam contidos dentro do vetor a partir de certa posição.

O algoritmo tenta encontrar o subvetor a partir da primeira posição do vetor. Se não encontrar, ou seja, se ocorrer alguma diferença na comparação elemento a elemento, o algoritmo tentara encontrar novamnte a partir da posição 2. E se não encontrar novamente, tentara encontrar a partir da posição 3 e assim sucessivamente, até encontrar ou acabar a possibilidade de encontrar, ou seja, o subvetor já não pode mais ser encontrado, pois os elementos restantes ainda não comparados constituem uma sequencia menor que o subvetor (o subvetor não cabe mais na parte restante do vetor).

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

```
as praticas_07d.s -o praticas_07d.o
```

e depois link dinamicamente com o seguinte comando:

```
ld praticas_07d.o -l c -dynamic-linker /lib/ld-linux.so.2 -o praticas_07d
```

O executavel se chamara praticas_07d, sem extensão, e para executá-lo digite:

```
./praticas_07d
```

```
.section    .data

apresenta:       .asciz     "\n*** Programa Localiza Subvetor dentro de Vetor 1.0 ***\n\n"
pedetamvet:      .asciz    "Digite o tamanho do vetor (0 < tam <= 50) => "
pedetamsubvet:   .asciz    "Digite o tamanho do subvetor (0 < tam <= %d) => "
pedenum:         .asciz    "Entre com o numero %d => "
infovet:         .asciz    "\nLEITURA DO VETOR:\n\n"
infosubvet:      .asciz    "\nLEITURA DO SUBVETOR:\n\n"
infovet2:        .asciz    "\nVetor Lido : "
infosubvet2:     .asciz    "\nSubvetor Lido : "
infoproc:        .asciz    "\nProcurando ...\n"
formain:         .asciz    "%d"
formaout:        .asciz    " %d"
resppositiva:    .asciz    "\nO Subvetor estah contido no vetor a partir da posicao %d!\n\n"
respnegativa:    .asciz    "\nO Subvetor não estah contido no vetor!\n\n"
pulalinha:       .asciz    "\n"
pergcont:        .asciz    "\nDeseja nova execucao <s>im ou <n>ao? => "
```

```
limpabuf:        .string   "%*c"


maxtam:          .int  50
maxaux:          .int  0
tamaux:          .int  0
tamsubvet:       .int  0
tamvet:          .int  0
posmax:          .int  0
num:             .int  0
n:               .int  0
resp:            .int  0
subvetor:        .space  204      #  4  bytes  para  cada  numero  a  ser
armazenado
vetor:           .space  204      #  4  bytes  para  cada  numero  a  ser
armazenado

.section .text
```

A seguir uma rotina para ler os tamanhos do subvetor e vetor e checar os limites permitidos. O numero deve ser maior que zero e menor que maxtam. O valor lido é retornado no registrador %ecx.

```
letam:

      pushl  $tamaux
      pushl  $formain
      call   scanf
      pushl  $pulalinha
      call   printf
      addl   $12, %esp # desfaz os ultimos 3 push's
      movl   tamaux, %ecx
      cmpl   $0, %ecx
      jle    letam
      cmpl   maxaux, %ecx
      jg     letam
      ret
```

A seguir uma rotina para ler os numeros do vetor. O endereco do vetor deve estar em %edi e o tamanho em %ecx.

```
levet:

      movl   $0, %ebx

volta1:

      incl   %ebx
      pushl  %edi
      pushl  %ecx
      pushl  %ebx
      pushl  $pedenum
      call   printf
      pushl  $num
      pushl  $formain
      call   scanf
      addl   $12, %esp # desfaz os ultimos 3 push's
      popl   %ebx
```

```
      popl   %ecx
      popl   %edi
      movl   num, %eax
      movl   %eax, (%edi)
      addl   $4, %edi
      loop   volta1
      ret
```

Segue uma rotina para mostrar os numeros do vetor. O endereco do vetor deve estar em %edi e o tamanho em %ecx.

```
mostravet:

      pushl  %edi
      pushl  %ecx
      movl   (%edi), %eax
      pushl  %eax
      pushl  $formaout
      call   printf
      addl   $8, %esp
      popl   %ecx
      popl   %edi
      addl   $4, %edi
      loop   mostravet
      pushl  $pulalinha
      call   printf
      addl   $4, %esp
      ret
```

Segue uma rotina que compara 2 strings, cujos enderecos devem estar nos registradores %edi e %esi e o tamanho em %ecx

```
comparasubvet:

      movl   (%edi), %eax
      movl   (%esi), %ebx
      cmpl   %eax, %ebx
      jnz    acabou
      addl   $4, %edi
      addl   $4, %esi
      loop   comparasubvet
      cmpl   %eax, %eax
acabou:
      ret

.globl _start
_start:

      pushl  $apresenta
      call   printf
      addl   $4, %esp
      movl   $0, n

le_tamanhos:

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

```
      movl   maxtam, %ecx
      movl   %ecx, maxaux
      call   letam
      movl   %ecx, tamvet

      pushl  %ecx
      pushl  $pedetamsubvet
      call   printf
      addl   $8, %esp

      movl   tamvet, %ecx
      movl   %ecx, maxaux
      call   letam
      movl   %ecx, tamsubvet

      movl   tamvet, %ecx
      subl   tamsubvet, %ecx
      incl   %ecx
      movl   %ecx, posmax

le_vetores:

      pushl  $infovet
      call   printf
      addl   $4, %esp

      movl   $vetor, %edi
      movl   tamvet, %ecx
      call   levet

      pushl  $infosubvet
      call   printf
      addl   $4, %esp

      movl   $subvetor, %edi
      movl   tamsubvet, %ecx
      call   levet

mostra_vetores:

      pushl  $infovet
      call   printf
      addl   $4, %esp

      movl   $vetor, %edi
      movl   tamvet, %ecx
      call   mostravet

      pushl  $infosubvet
      call   printf
      addl   $4, %esp

      movl   $subvetor, %edi
      movl   tamsubvet, %ecx
      call   mostravet

compara_vetores:
```

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

      movl   $vetor, %esi
      subl   $4, %esi

voltacomp:

      addl   $4, %esi
      pushl  %esi
      incl   n

      movl   n, %eax
      cmpl   posmax, %eax
      jg     naoachou

      movl   tamsubvet, %ecx
      movl   $subvetor, %edi
      call   comparasubvet
      jz     achou

      popl   %esi
      jmp    voltacomp

naoachou:

      pushl  $respnegativa
      call   printf
      addl   $4, %esp
      jmp    fim

achou:

      pushl  n
      pushl  $respppositiva
      call   printf
      addl   $8, %esp

fim:

      pushl  $pergcont
      call   printf
      pushl  $limpabuf
      call   scanf
      addl   $8, %esp
      call   getchar
      cmpl   $'s', %eax
      jz     _start
      pushl  $0
      call   exit
```

DESAFIO ULTRA: Percorrer o vetor de tamanho m e localizar algum subvetor de tamanho n que ocorra outras vezes (repete) dentro do vetor, a partir de outras posições a frente. O algoritmo deve testar todas as possibilidades de encontrar uma subsequência qualquer de tamanho n, localizada a partir de uma posição qualquer, em outras posições a frente.

O algoritmo pára quando encontrar qualquer subsequencia que ocorrer pelo menos 2 vezes. O algoritmo deve mostrar o vetor, o subvetor e o número de ocorrências/repetições. O usuário informa apenas o tamanho n do subvetor.

DICA: você pode usar o programa desta prática como uma função que dado um subvetor de tamanho n, procura este subvetor dentro do vetor, só que ao invés de ler o subvetor do teclado, o retira do próprio vetor. Na primeira tentativa, os n elementos do vetor a patir da 1ª posição são colocados no subvetor e a função é chamada para localizar este subvetor dentro do vetor a partir da 2ª posição. Se encontrar, bingo!, encontrou o subvetor. Se não encontrar, coloca os n elementos a partir da 2ª posição no subvetor e a função é novamente chamada para localizar este subvetor dentro do vetor a partir da 3ª posição. Se encontrar, bingo!, encontrou o subvetor. Se não encontrar, repete o processo a partir da 3ª posição e assim sucessivamente, até encontrar ou não ter mais possibilidade de encontrar.