FONAMENTS D'ORDINADORS

TEMA 6: Vectors

Manel Guerrero

[H7] Declaració de vectors

- Vectors inicialitzats:
  - nom_var: .tipus dades

```
.data
vl: .long 1,2,3
vw: .word 1,2,3
vb: .byte 1,2,3
s1: .ascii "abd"
s2: .asciz "abd"
(.asciz guarda una posició extra amb el char '\0')
```

- Vectors no inicialitzats:
  - .comm nom_var,mida_vector, alineació

```
.bss
.comm uvl,3*4,4
    # int uvl[3]
.comm uvw,3*2,2
    # short uvw[3]
.comm uvb,3,1
    # char uvb[3]
```

Declaració de vectors

- Adreçament indexat: D(Rb,Ri,S)
  L'operand és a l'@ D+Rb+Ri*S
- Amb long:
  - `v[i] = 0; # %ecx = i`
  - `movl $0, v(,%ecx,4)`
  - `v[i] = 1; # %ebx apunta a v i %ecx = i`
  - `movl $1, (%ebx,%ecx,4)`
- Amb char:
  - `c[i]='a' #ecx guarda i`
  - `movb $'a', c(%ecx)`

```
t6_vectors.s:
# short vw={3,4,5}, i=2;
# v[i]=i; (vw[i])++; (vw[i])++;
.data
vw: .word 3,4,5
.text
.global main
main:
  movw $2, %cx
  movswl %cx, %ecx        # Trampa!
# Rb & Ri tienen que ser de 32 bits!
  movw %cx, vw(,%ecx,2)
  incw vw(,%ecx,2)
  movl $vw, %ebx
  incw (%ebx,%ecx,2)
  movswl vw(,%ecx,2), %ebx
  movl $1, %eax
  int $0x80
```

Recorregut vector enters

```
t6_recorregut_i.s:
# Retorna la suma dels elements d'un vector acabat en 0.
# ./t6_recorregut_i ; echo $?
# 45
.data
v: .long 1,2,3,4,5,6,7,8,9,0
.text
.global main
main:
 movl $0, %eax
 movl $0, %ecx
loop:
  cmpl $0, v(,%ecx,4)
  je end_loop
  addl v(,%ecx,4), %eax
  incl %ecx
  jmp loop
end_loop:

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

Cerca vector enters

```
t6_cerca_i.s:
# Retorna 1er negatiu a v.
# ./t6_cerca_i ; echo $? # 253
# Si no hi ha cap neg retorna 0
.data
mida: .long 5
v: .long 1,2,-3,4,5
.text
.global main
main:
  movl $0, %ecx
loop:
  cmpl mida, %ecx
  je end_loop
  cmpl $0, v(,%ecx,4)
  jl end_loop
  incl %ecx
  jmp loop
end_loop:
  cmpl mida, %ecx
  je else
  movl v(,%ecx,4), %ebx
  jmp end_if
else:
  movl $0, %ebx
end_if:
  movl $1, %eax
  int $0x80
```

Cerca vector enters (2)

```
t6_cerca_i2.s:
# Versio optimitzada
.data
mida: .long 5
v: .long 1,2,-3,4,5
.text
.global main
main:
  movl $0, %ecx
loop:
  cmpl mida, %ecx
  je not_found
  cmpl $0, v(,%ecx,4)
  jl if_found
  incl %ecx
  jmp loop
if_found:
  movl v(,%ecx,4), %ebx
  jmp end_if
not_found:
  movl $0, %ebx
end_if:
  movl $1, %eax
  int $0x80
```

[H8] Recorregut vector bytes

```
t6_recorregut_a.s:
.data
v: .ascii "Cuento cuantas as."
.text
.global main
main:
  movl $0, %eax
  movl $0, %ecx
loop:
  cmpb $'.', v(%ecx)
  je end_loop
  cmpb $'a', v(%ecx)
  jne end_if
  incl %eax
end_if:
  incl %ecx
  jmp loop
end_loop:

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

Cerca vector bytes

```
t6_cerca_a.s:
# Retorna la pos de la 1era 'a'.
# ./t6_cerca_a ; echo $? # 2
# Si no hi ha 'a' ret 255 (-1)
.data
v: .asciz "Donde esta la 1era a."
.text
.global main
main:
  movl $0, %ecx
loop:
  cmpb $0, v(%ecx)
  je end_loop
  cmpb $'a', v(%ecx)
  je end_loop
  incl %ecx
  jmp loop
end_loop:
  cmpb $'a', v(%ecx)
  jne else
  movl %ecx, %ebx
  jmp end_if
else:
  movl $-1, %ebx
end_if:
  movl $1, %eax
  int $0x80
```

[H9] Més problemes

- De la col·lecció:
  - Problema 2: Escribir un programa en lenguaje ensamblador IA32 que convierta una cadena de caracteres numéricos acabados en espacio, a un número entero positivo.
  - Problema 3: Escribe un programa en ensamblador IA32 que encuentre el valor máximo y mínimo de un vector de N números enteros de 2 bytes (short) almacenados en memoria a partir de la dirección simbólica VECTOR. Las direcciones simbólicas MAX y MIN almacenarán el valor máximo y mínimo respectivamente.
  - Problema adicional 1