

# FONAMENTS D'ORDINADORS

## TEMA 6: Vectors

Manel Guerrero

# [H7] Declaració de vectors

- Vectors inicialitzats:

- nom\_var: .tipus dades

.data

v1: .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 uv1,3\*4,4

# int uv1[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:                cmpl $0, v(,%ecx,4)
# Retorna ler negatiu a v.    jl end_loop
# ./t6_cerca_i ; echo $? # 253  incl %ecx
# Si no hi ha cap neg retorna 0  jmp loop
.data                        end_loop:
mida: .long 5                cmpl mida, %ecx
v: .long 1,2,-3,4,5          je else
.text                        movl v(,%ecx,4), %ebx
.global main                 jmp end_if
main:                         else:
    movl $0, %ecx            movl $0, %ebx
loop:                         end_if:
    cmpl mida, %ecx          movl $1, %eax
    je end_loop              int $0x80
```

## Cerca vector enters (2)

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

## [H8] Recorregut vector bytes

```
t6_recorregut_a.s:          cmpb $'a', v(%ecx)
.data                        jne end_if
v: .ascii "Cuento cuantas as."  incl %eax
.text                        end_if:
.global main                 incl %ecx
main:                         jmp loop
    movl $0, %eax            end_loop:
    movl $0, %ecx
loop:                         movl %eax, %ebx
    cmpb $'.', v(%ecx)       movl $1, %eax
    je end_loop              int $0x80
```

## Cerca vector bytes

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