Tema 2
Assemblador i tipus de dades bàsics

Estructura de Computadors (EC)

Vectors

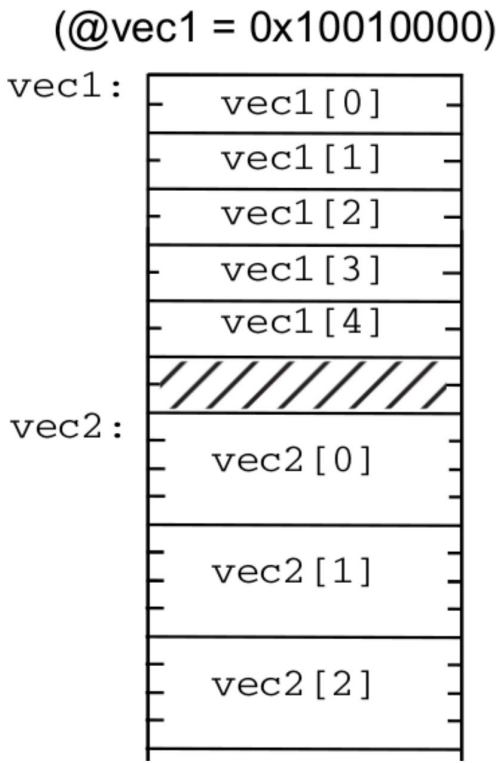
Vectors

- Col·lecció unidimensional d'elements del mateix tipus
  - Els elements s'identifiquen per un índex
    - El primer element té índex 0
    - Vector de `N` elements: `[0, N-1]`
  - Els elements s'emmagatzemen en posicions consecutives a partir de l'adreça inicial del vector
  - A MIPS els elements han de respectar les regles d'alineament

Declaració i emmagatzematge de vectors

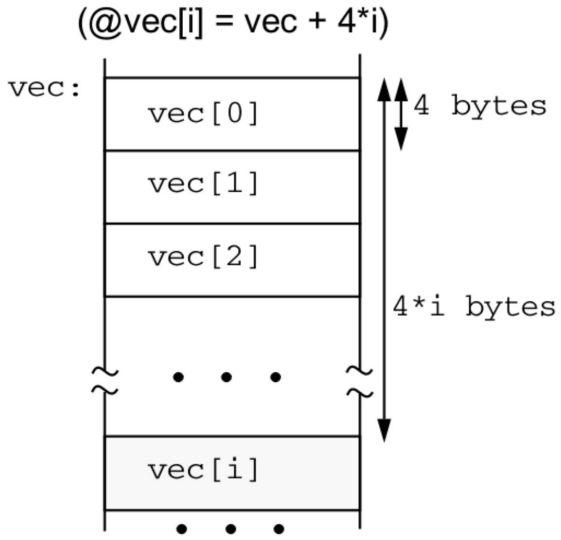
```
// En C
short vec1[5] = {0, -1, 2, -3, 4};
int vec2[100];
```

```
# En MIPS
          .data
vec1:     .half 0, -1, 2, -3, 4
          .align 2
vec2:     .space 400
```



Accés (aleatori) a un element d'un vector

- Per accedir a un element `i` de un vector s'ha de calcular la seva adreça
- Si els elements tenen tamany `T` bytes, l'adreça del element es calcula:
  - `@vec[i] = @vec[0] + (i * T)`



Exemple

- Donada la següent sentencia en C
  - `vec[i] = 10;`
- Indica el codi en llenguatge assemblador MIPS assumint que `i` està a `$t0` i que `vec` és un vector global d'enters de 32 bits

```
la $t2, vec             # $t2 = @vec[0]
sll $t1, $t0, 2         # $t1 = 4 * i
addu $t2, $t2, $t1      # $t2 = @vec[0] + 4 * i
li $t1, 10
sw $t1, 0($t2)
```

Exercici

- Donades les següents declaracions globals en C
  - `int val[100], vec[100];`
  - `int elem;`
- Tradueix la següent sentència a llenguatge assemblador MIPS

1. `elem = val[5] + vec[10];`
2. `elem = vec[10 + val[5]];`

Vectors i punters

- En C, els vectors tenen el mateix tipus que un punter
  - Un vector és un punter al primer element del vector

```
int vec[100];
int *p;
```

- Podem inicialitzar un punter assignant-li un vector - pero no al revés
  - `p = vec;`
- Podem utilitzar l'operador `[]` amb un punter
  - `p[8] = 1;`
- Podem utilitzar l'operador de desreferència `*` amb un vector
  - `*vec = 0;`

Vectors i punters

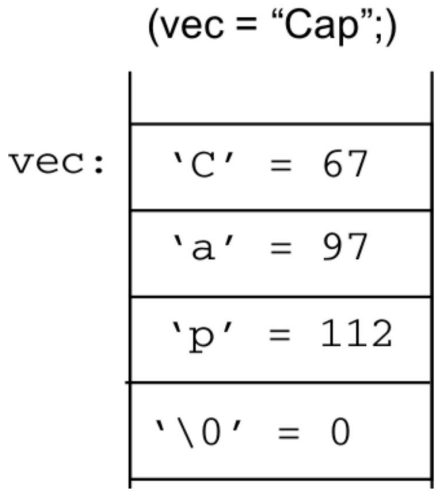
- Un punter `p` es pot considerar com un vector
  - El primer element del qual es l'apuntat per `p`

- La següent expressió és vàlida
  - `*(p + i) = 0;`
- I és equivalent a l'expressió
  - `p[i] = 0;`

Strings

Strings (cadenes de caràcters)

- Vectors amb un número variable de caràcters
- L'últim caràcter de la cadena (sentinella) sempre val 0 (‘\0’)
- El string “Cap” es representa amb els caràcters 67, 97, 112, 0



Declaració de strings

- En C
  - `char cadena[] = “Hello world!”;`
- El compilador reserva 13 bytes: 12 caràcters més el sentinella

Declaració de strings

- En C
  - `char cadena[] = “Hello world!”;`
- El compilador reserva 13 bytes: 12 caràcters més el sentinella

- En MIPS

```
        .data
cadena: .ascii “Hello world!”
        .byte 0                 # sentinella
```

- Alternativa en MIPS

```
        .data
cadena: .asciiz “Hello world!”
```

Accés als elements d'un string

- En C els caràcters es codifiquen en ASCII (1 byte)
- S'accedeixen utilitzant les instruccions `lb y sb`
- Mateix mètode que s'utilitza per accedir a vectors
  - `char cadena[] = "Hello world!;`
  - `@cadena[i] = @cadena[0] + i;`
  - El tamany dels elements és sempre 1 per els strings

Exemple

- Tradueix a assemblador MIPS la següent sentencia en C, assumint que les variables `i` i `j` estan als registres `$t0` i `$t1` respectivament
  - `cadena[i] = cadena[j] - 32`

Problemes

Problema 1 (Part I)

- Donades les següents declaracions de variables en llenguatge C

```
short a[] = {513, 17, -5};
long long b = 1030;
short c = 0;
char d[] = “2017”;
short *e = &c;
```

- Escriu el programa MIPS equivalent
- Indica el contingut de la memòria en hexadecimal, assumint que les variables globals s’emmagatzemen a partir de l’adreça `0x10010000`. El codi ASCII del `‘0’` és `0x30`

Problema 1 (Part II)

- Donat el següent codi en assemblador MIPS, indica el valor final en hexadecimal del registre `$t1`

```
la      $t0, d
lb      $t0, 1($t0)
addiu   $t1, $t0, -0x31
```

Problema 1 (Part III)

- Tradueix a llenguatge assemblador MIPS la següent sentencia en C

```
*e = a[0] - a[2];
```

Problema 2

- Donada la següent declaració de variables globals en un programa MIPS, emmagatzemades a partir de l'adreça 0x10010000

```
        .data
a:      .byte    19, 0x91
b:      .space   2
c:      .asciiz  “ABCD”          #ASCII ‘A’ = 0x41
d:      .half    -33
e:      .word    a+1
f:      .dword   0xFFFFFFFF
```

- Escriu la declaració equivalent en llenguatge C
- Indica el contingut de les primeres 24 posicions de memòria

Problema 3

- Tradueix a MIPS el següent programa en C

```
unsigned short v[8];
unsigned short *p = &v[2];

void main() {
    *(p + 5) = *p + 5;
}
```