AC1 2013-2014

universidade de aveiro theoria poiesis praxis
Departamento de Electrónica, Telecomunicações e Informática

Arquitectura de Computadores I

Pedro Miguel Cabral

Aula 04

Ponteiros em Linguagem C

1

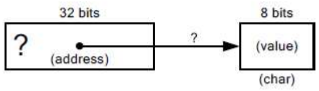
AC1 2013-2014

1. Ponteiro para caracter, não inicializado

a) Exemplo de declaração em linguagem C:

```
char *p;
```

b) Interpretação gráfica:



c) Acção desenvolvida na tradução para linguagem máquina:

- Definir o registo interno / reservar espaço na memória para alojar um endereço (32 bits)

d) Caso o ponteiro resida num registo interno, basta definir qual o registo a usar para esse efeito e incluí-lo nas instruções que manipulam o ponteiro.

e) Caso o ponteiro resida na memória, uma possível tradução para *Assembly* do MIPS da sua declaração é:

```
ptr_p:   .space   4    # Reserva 4 bytes de memória
                       # (32 bits) para alojar o
                       # ponteiro. Não há inicialização
```

2

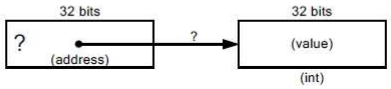
AC1 2013-2014

2. Ponteiro para inteiro, não inicializado

a) Exemplo de declaração em linguagem C:

```
int *p;
```

b) Interpretação gráfica:



c) Acção desenvolvida na tradução para linguagem máquina:

- Reservar espaço na memória/registo interno para um endereço (32 bits)

d) Possível tradução para *Assembly* do MIPS (caso o ponteiro resida na memória):

```
ptr_p:   .space   4    # Reserva 4 bytes de memória
                       # (32 bits) para alojar o
                       # ponteiro. Não há inicialização
```

3

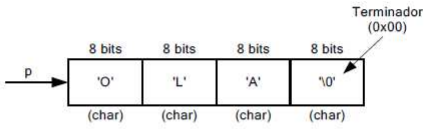
AC1 2013-2014

3. *Array* de caracteres

a) Exemplo de declaração em linguagem C:

```
char p[]="OLA";
```

b) Interpretação gráfica:



c) Acção desenvolvida na tradução para linguagem máquina:

- Reservar espaço na memória para um *array* de caracteres (incluindo para o terminador, o byte 0x00), e efectuar a respectiva inicialização

d) Possível tradução para *Assembly* do MIPS:

```
p:   .asciiz  "OLA"    # Reserva 4 bytes de memória e
                       # inicializa-os com os códigos
                       # ASCII dos 3 caracteres e com o
                       # código do terminador (0).
                       # O valor de "p" pode ser obtido
                       # com a instrução "load address"
```

Ou, alternativamente:

```
p:   .ascii   "OLA"    # Reserva 3 bytes de memória e
                       # inicializa-os com os códigos
                       # ASCII dos 3 caracteres
     .byte    0x00     # Reserva 1 byte e inicializa-o
                       # com o valor 0
```

4

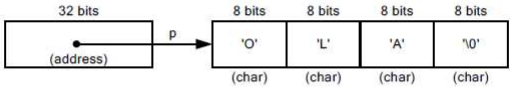
AC1 2013-2014

4. Ponteiro para *Array* de caracteres

a) Exemplo de declaração em linguagem C:

```
char *p = "OLA";
```

b) Interpretação gráfica:



c) Acções desenvolvidas na tradução para linguagem máquina:

- Reservar espaço para um *array* de caracteres e efectuar a respectiva inicialização
- Reservar espaço para um endereço e efectuar a respectiva inicialização

d) Tradução para *Assembly* do MIPS (caso o ponteiro resida na memória):

```
p:      .asciiz  "OLA"
ptr_p:  .word    p    # Reserva 4 bytes de memória
                      # e inicializa-os com o endereço
                      # da primeira posição do array
                      # de caracteres (i.e. &array[0]).
                      # O valor de "ptr_p" pode ser
                      # obtido com a instrução "load
                      # address"
```

5

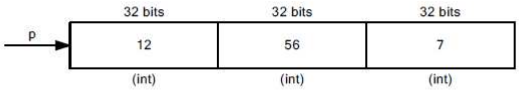
AC1 2013-2014

5. *Array* de inteiros

a) Exemplo de declaração em linguagem C:

```
int p[] = {12, 56, 7};
```

b) Interpretação gráfica:



c) Acção desenvolvida na tradução para linguagem máquina:

- Reservar espaço para um *array* de inteiros e efectuar a respectiva inicialização

d) Tradução para *Assembly* do MIPS:

```
p:  .word   12, 56, 7    #
                         # O valor de "p" pode ser obtido
                         # com a instrução "load address"
```

NOTA:
A linguagem C não permite a declaração de um ponteiro para um *array* de inteiros, cuja representação seria, por exemplo: "`int *p = {12, 56, 7};`". Contudo, esta declaração pode ser decomposta em duas, do seguinte modo:

```
int pp[] = {12, 56, 7};
int *p = pp;
```

6

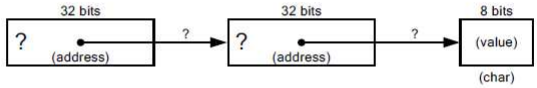
AC1 2013-2014

6. Ponteiro para ponteiro para caracter, não inicializado

a) Exemplo de declaração em linguagem C:

```
char **p;
```

b) Interpretação gráfica:



c) Acção desenvolvida na tradução para linguagem máquina:

- Reservar espaço para um endereço (32 bits)

d) Tradução para *Assembly* do MIPS (caso o ponteiro resida na memória):

```
ptr_p:   .space   4    # Reserva 4 bytes na memória para
                       # alojar o ponteiro
```

7

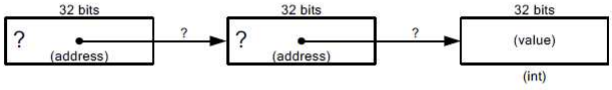
AC1 2013-2014

7. Ponteiro para ponteiro para inteiro, não inicializado

a) Exemplo de declaração em linguagem C:

```
int **p;
```

b) Interpretação gráfica:



c) Acção desenvolvida na tradução para linguagem máquina:

- Reservar espaço para um endereço (32 bits)

d) Tradução para *Assembly* do MIPS (caso o ponteiro resida na memória):

```
ptr_p:   .space   4    # Reserva 4 bytes na memória para
                       # alojar o ponteiro
```

8

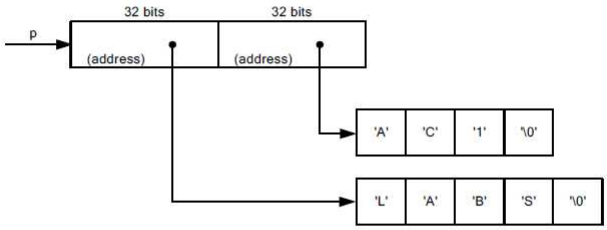
AC1 2013-2014

8. Array de ponteiros para caracter

a) Exemplo de declaração em linguagem C:

```
char *p[] = {"AC1", "LABS"};
```

b) Interpretação gráfica:



c) Acções desenvolvidas na tradução para linguagem máquina:

- Reservar espaço para os *arrays* de caracteres e efectuar a respectiva inicialização
- Reservar espaço para o *array* de ponteiros (*array* de inteiros) e efectuar a respectiva inicialização

d) Tradução para *Assembly* do MIPS (caso os ponteiros residam na memória):

```
array1:  .asciiz  "AC1"
array2:  .asciiz  "LABS"
p:       .word    array1, array2
```

9

AC1 2013-2014

```
char str[] = "Ola' Mundo";

void main (void)
{
    unsigned int len;
    char *p;

    p = str;
    len = 0;
    while( *p != 0x00 )
    {
        len++;
        p++;
    }
}
```

```
        .data
str:    .asciiz  "Ola' Mundo"

        .text
        .globl   main

main:   la       $t0, str
        li       $t1, 0
loop:   lb       $t2, 0($t0)

        beq      $t2, 0, lpx

        addiu    $t1, $t1, 1
        addiu    $t0, $t0, 1
        j        loop
lpx:    jr       $ra
```

10