
SECONDA PARTE

Esercizio “struttura A”

Realizzare la seguente allocazione di dati statici nel segmento «.data» di un programma assembler

```
int main(){  
    // structure A  
    typedef struct  
    {  
        char    c;  
        short int s;  
    } structa_t;  
  
    structa_t istanza = {'c', 4};  
    return 0;  
}
```

1. **Comprendere l'allineamento dati realizzato da MARS**
2. **Calcolare la sizeof(istanza)**
3. **Disabilitare l'allineamento automatico**
4. **Ripristinare manualmente l'allineamento dei dati in memoria**

Direttiva «.align»

- Usata per saltare al prossimo indirizzo correttamente allineato per la dimensione del dato assegnato
 - Formato:
 - .align 0,1,2 oppure 3
 - 0=allineamento al byte
 - 1=allineamento alla mezza parola (2 byte)
 - 2=allineamento alla parola (4 byte)
 - 3=allineamento alla doppia parola (8 byte)
 - L'argomento di «align» non è il numero di byte di padding da inserire, ma la dimensione del prossimo dato, dunque la dimensione cui allinearne l'indirizzo
 - La dimensione del dato è espressa in byte e si ricava elevando il numero due alla potenza indicata dall'argomento
-

Soluzione 1-2

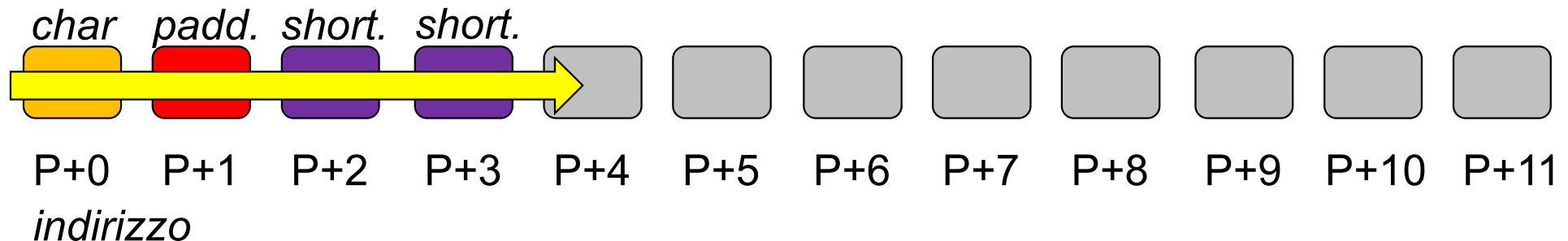
.data

c: .byte 'c' #codice ascii 0x63

s: .half 4 #allocazione in memoria di 0x0004

Data Segment		
Address	Value (+0)	Value (+4)
0x10010000	0x00040063	0x00000000

Little endian



Sizeof(istanza) = 4

(prova sul tuo ambiente di programmazione C)

Soluzione 3-4

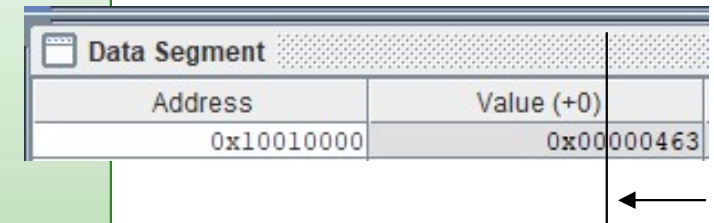
- ❑ Si può disabilitare l'allineamento con:

.data

.align 0

c: .byte 'c' #codice ascii 0x63

s: .half 4 #allocazione in memoria di 0x0004



Address	Value (+0)
0x10010000	0x00000463

- ❑ Si può manualmente ripristinare con:

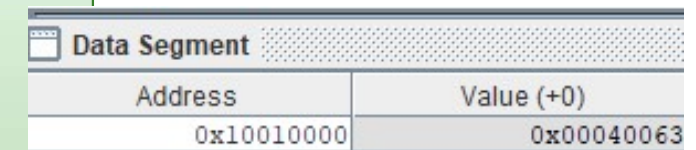
.data

.align 0

c: .byte 'c' #codice ascii 0x63

.align 1

s: .half 4 #allocazione in memoria di 0x0004



Address	Value (+0)
0x10010000	0x00040063

Esercizio “struttura B”

Realizzare la seguente allocazione di dati statici nel segmento «.data» di un programma assembler

```
int main(){
    // structure B
    typedef struct {
        short int s;
        char c;
        int i;
    } structb_t;

    structb_t istanza = {4, 'c', 8};
    return 0;
}
```

1. **Comprendere l'allineamento dati realizzato da MARS**
2. **Calcolare la sizeof(istanza)**
3. **Disabilitare l'allineamento automatico**
4. **Ripristinare manualmente l'allineamento dei dati in memoria**

Soluzione 1-2

.data

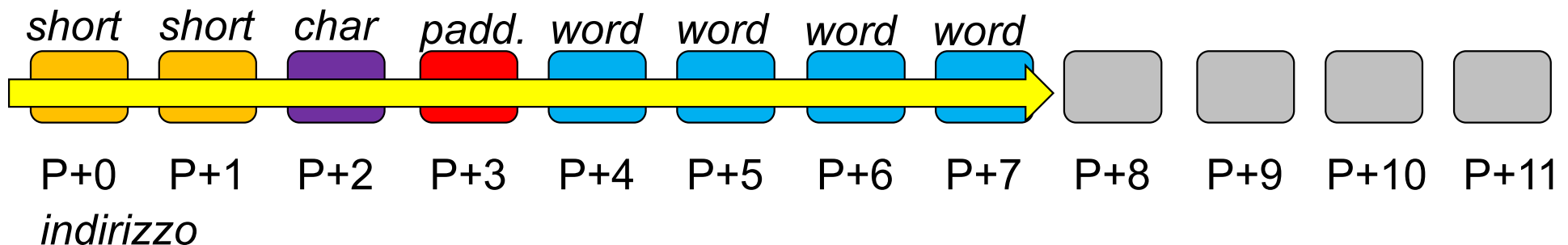
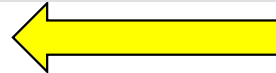
s: .half 4 #allocazione in memoria di 0x0004

c: .byte 'c' #codice ascii 0x63

i: .word 8 #allocazione in memoria di 0x00000008

Data Segment		
Address	Value (+0)	Value (+4)
0x10010000	0x00630004	0x00000008

Little endian



Sizeof(istanza) = 8

Soluzione 3-4

- Si può disabilitare l'allineamento con:

.align 0

da cui $\text{Sizeof}(\text{istanza})=7$

Come si fa a disattivare l'allineamento in C?

#pragma pack(push,1) (ripristina con #pragma pack(pop))

- Si può manualmente ripristinare con:

.data

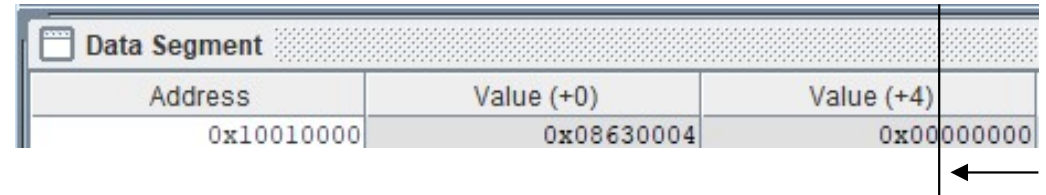
.align 0

s: .half 4

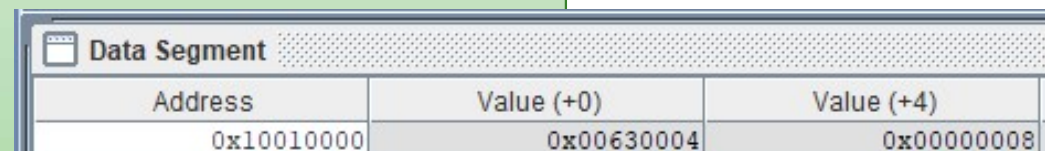
c: .byte 'c'

.align 2

i: .word 8



Address	Value (+0)	Value (+4)
0x10010000	0x08630004	0x00000000



Address	Value (+0)	Value (+4)
0x10010000	0x00630004	0x00000008

Esercizio “struttura C”

Realizzare la seguente allocazione di dati statici nel segmento «.data» di un programma assembler

```
int main(){
    // structure C
    typedef struct {
        char  c;
        double d;
        int  s;
    } structc_t;

    structc_t istanza = {'c', 0,-1};
    return 0;
}
```

1. **Comprendere l'allineamento dati realizzato da MARS**
2. **Calcolare la sizeof(istanza)**
3. **Disabilitare l'allineamento automatico**
4. **Ripristinare manualmente l'allineamento dei dati in memoria**

Soluzione 1-2

.data

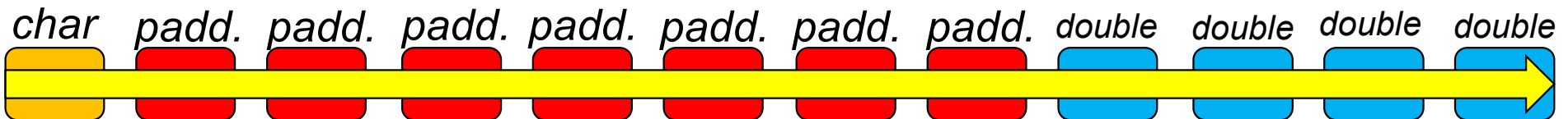
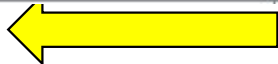
c: .byte 'c'

d: .double 0

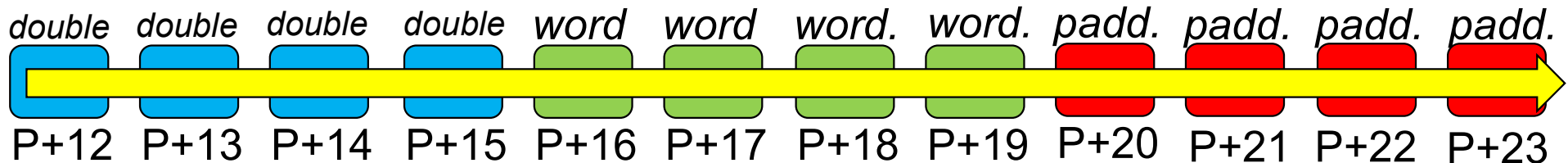
s: .word -1

Data Segment					
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	0x00000063	0x00000000	0x00000000	0x00000000	0xffffffff

Little endian



P+0 P+1 P+2 P+3 P+4 P+5 P+6 P+7 P+8 P+9 P+10 P+11
indirizzo



P+12 P+13 P+14 P+15 P+16 P+17 P+18 P+19 P+20 P+21 P+22 P+23

Sizeof(istanza) = 24

Soluzione 3-4

- Si può disabilitare l'allineamento con:

.align 0

Data Segment					
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	
0x10010000	0x00000063	0x00000000	0xffffffff00	0x000000ff	

da cui $\text{Sizeof}(\text{istanza})=13$

- Si può manualmente ripristinare con:

.data

.align 0

c: .byte 'c'

.align 3

d: .double 0

.align 2

s: .word -1

Data Segment					
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)
0x10010000	0x00000063	0x00000000	0x00000000	0x00000000	0xffffffff

Esercizio “struttura D”

Realizzare la seguente allocazione di dati statici nel segmento «.data» di un programma assembler

```
int main(){  
    // structure D  
    typedef struct {  
        double d;  
        int s;  
        char c;  
    } structd_t;  
  
    structd_t istanza = {0, -1, 'c'};  
    return 0;  
}
```

1. **Comprendere l'allineamento dati realizzato da MARS**
2. **Calcolare la sizeof(istanza)**
3. **Disabilitare l'allineamento automatico**
4. **Ripristinare manualmente l'allineamento dei dati in memoria**

Soluzione 1-2

.data

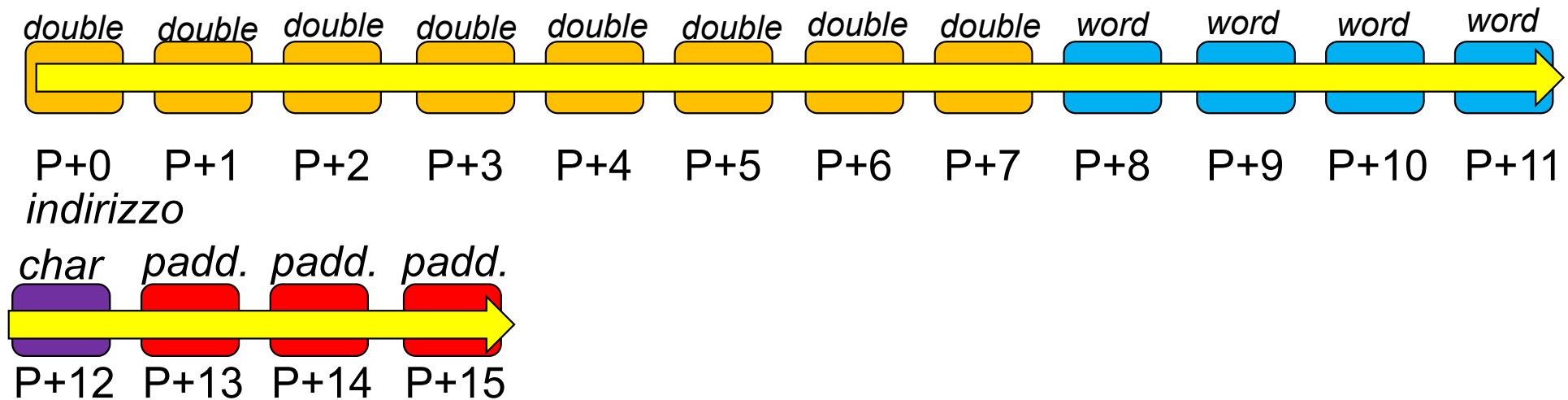
d: .double 0

s: .word -1

c: .byte 'c'

Data Segment				
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	0x00000000	0x00000000	0xffffffff	0x00000063

Little endian



Sizeof(istanza) = 16

Soluzione 3-4

- Si può disabilitare l'allineamento con:

.align 0

Data Segment				
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	0x00000000	0x00000000	0xffffffff	0x00000063

da cui $\text{Sizeof}(\text{istanza})=13$

- Si può manualmente ripristinare con:

.data

.align 0

d: .double 0

.align 2

s: .word -1

c: .byte 'c'

Data Segment				
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)
0x10010000	0x00000000	0x00000000	0xffffffff	0x00000063