Calcolatori Elettronici
Esercitazione 0

M. Sonza Reorda – M. Monetti
M. Rebaudengo – R. Ferrero
L. Sterpone – E. Vacca

Politecnico di Torino
Dipartimento di Automatica e Informatica

Esercitazione 0 - Obiettivi

- Acquisire familiarità con il simulatore QtSpim
- Esempi mirati a :
  - Costruire un programma a partire dal template
  - Dichiarare Variabili
  - Manipolare dati da Memoria a Registro
  - Manipolare dati da Registro a Memoria
  - I/O basico da Console

Esercizio 1

- Scrittura di un valore in un registro e sua verifica su Ripes.

- Vengono memorizzati
  - `$t0 valore 10 decimale`
  - `$s0 valore DC esadecimale`

Soluzione-1

```
 .data

        .text
main:

        li  t0, 10
        li  s0, 0xdc

        li a7,10
        ecall
```

Esercizio 2

- Scrittura di un valore in una cella di memoria

- Dichiarazione della variabile inizializzata con il valore 3 decimale
  - `wVar:  .word    3`

Soluzione-2

```
 .data

wVar:   .word    3
        .text

main:

        li  t0, 10
        la  s11, wVar
        sw  t0, 0(s11)

        li a7,10
        ecall
```

Esercizio 3

- Somma di 2 valori contenuti in due variabili e memorizzazione risultato in una variabile Risultato
- I due operandi
  - `wOpd1:  .word  10`
  - `wOpd2:  .word  24`
- La variabile Risultato
  - `wResult: .space  4`

Soluzione-3

```
 .data
wOpd1:    .word   10
wOpd2:    .word   24
wResult:  .zero  4

        .text

main:
        lw  t0, wOpd1
        lw  t1, wOpd2

        add t2, t1, t0
        la s11, wResult
        sw t2, 0(s11)

        li  a7,10
        ecall
```

Esercizio 4

- Somma degli elementi di un Vettore (I)
  - `wVett:  .word  5, 7, 3, 4,`
- Risultato in una variabile Risultato
  - `wResult:  .space  4`
- Tecnica molto semplice fatta di somme successive con utilizzo di un Registro come accumulatore.
  - `t1  ACCUMULATORE`
  - `t0  INDIRIZZO Vettore`
  - `t2  Secondo OPERANDO`

Soluzione-4

```
 .data

wVett:      .word  5, 7, 3, 4, 3
wResult:    .zero  4

            .text

main:
            li    t1, 0
            la    t0, wVett
```

Soluzione-4 [cont.]

```
lw    t2, 0(t0)     # 0
add   t1, t1, t2
lw    t2, 4(t0)     # 1
add   t1, t1, t2
lw    t2, 8(t0)     # 2
add   t1, t1, t2
lw    t2, 12(t0)    # 3
add   t1, t1,  t2
lw    t2, 16(t0)     # 4
add   t1, t1, t2

la s11, wResult
sw    t1, 0(s11)

li a7,10
ecall
```

Esercizio 5

- Somma degli elementi di un Vettore (II) con un loop

- Registri
  - `t0.  Indirizzo Vettore`
  - `t1.  Accumulatore`
  - `t2.  Temporaneo`
  - `t3.  Contatore`

Soluzione-5

```
          .data

wVett:    .word  2, 5, 16, 12, 34, 7, 20, 11, 31, 44, 70, 69, 2, 4, 23
wResult:  .zero  4

          .text
main:
          li   t1, 0
          li   t3, 15
          la   t0, wVett
```

Soluzione-5 [cont.]

```
ciclo:    lw    t2, 0(t0)
          add   t1, t1, t2
          addi  t0, t0, 4     #  ++
          li    s11, 1
          sub   t3, t3, s11
          beqz   t3, fine
          j     ciclo

 fine:    la    s11, wResult
          sw    t1, 0(s11)

          li a7,10
          ecall
```

Esercizio 6

- Lettura da tastiera e visualizzazione a video di un vettore di 5 caratteri

- Registri
  - `t0.  Indirizzo Vettore`
  - `t1.  Contatore`
  - `ecall (a7=`**`1`**`).Print integer(a0 = integer)`
  - `ecall (a7=`**`4`**`).Print string (a0 = string)`
  - `ecall (a7=`**`63`**`).Get char(a0 = integer)`

Esercizio 6

- Tipo **string** - NULL terminated
  - `.string  "Inserire numeri\n"`
- Il NULL è un carattere ASCII non stampabile e viene utilizzato per contrassegnare la fine della stringa. La terminazione NULL è standard ed è richiesta dal servizio di sistema della stringa di stampa (per funzionare correttamente).

Esercizio 6

- Get char, si chiude con un **ENTER**

- Non viene eseguito nessun controllo su tipo di carattere inserito (ovvero se corrisponde ad una cifra numerica)

Soluzione-6

```
 .data

buffer:    .zero 255

wRes:         .zero   20
message_in:   .string   "\nInserire numeri: "
message_out: .string  "\nNumeri inseriti: "
space:        .string   " ; "


              .text

main:
              la a0, message_in
              addi   a7, x0, 4
              ecall

              la t0, wRes
              li t1, 0
```

Soluzione-6 [cont.]

```
uno:
# syscall 5 (read_int)
li a7, 63
li a0,0
la a1, buffer
li a2,255
ecall
lw a0, 0(a1)
andi a0,a0,255
li s11, 0x30
sub a0,a0,s11
#Risultato in a0
              sw a0, 0(t0)
              li s11,4      # DIM
              beq   t1, s11, print_num
              addi t1, t1, 1
              addi t0, t0, 4
              j uno
```

Soluzione-6 [cont.]

```
print_num:   la a0, message_out
             addi a7, x0, 4
             ecall
             la t0, wRes
             li t1, 0          # contatore
ciclo_print: addi a7, x0, 1    # call code, print int
             lw a0, 0(t0)      # value for int to print
             ecall             # system call
             li s11, 4
             beq   t1, s11, fine
             la a0, space
             addi a7, x0, 4
             ecall
             addi t1, t1, 1
             addi t0, t0, 4
             j ciclo_print

fine:        li a7,10
             ecall
```

Esercizio 7

- Ricerca del carattere minimo, vengono inseriti da tastiera DIM valori, si calcola il minimo e si visualizza

- Registri
  - `t0.  Indirizzo Vettore`
  - `t1.  Contatore`
  - `t2.  Valore Minimo`
  - `t3.  Temporaneo`

Soluzione-7

```
 .data
buffer:    .zero 255
wVet:          .zero    20
wRes:          .zero    4
message_in:    .string   "\nInserire numeri: "
message_out:   .string    "\nValore Minimo : "
acapo:         .byte     0x0D


               .text

main:
               la t0, wVet
               li t1, 0                  # contatore

               la     a0, message_in
               addi   a7, x0, 4
               ecall
```

Soluzione-7 [cont.]

```
uno:
# syscall 5 (read_int)
li a7, 63
li a0,0
la a1, buffer
li a2,255
ecall
lw a0, 0(a1)
andi a0,a0,255
li s11, 0x30
sub a0,a0,s11
#Risultato in a0
              sw     a0, 0(t0)
              addi   t2, x0,4
              beq    t1, t2, calc
              addi   t1, t1, 1
              addi   t0, t0, 4
              j   uno
```

Soluzione-7 [cont.]

```
calc:         la     t0, wVet
              li     t1, 0                # contatore
              lw     t2, 0(t0)            # $t2 memorizzo MIN
              addi   t0, t0, 4
loop_min:     addi   s11, x0, 4
              beq    t1, s11, print_num
              lw     t3, 0(t0)
              blt    t3, t2, change_min
cont:         addi   t1, t1, 1
              addi   t0, t0, 4
              j   loop_min

change_min:   mv     t2, t3
              j      cont
```

Soluzione-7 [cont.]

```
print_num:
            la     a0, message_out     # addr of NULL
            addi   a7, x0, 4           # call code, print
            ecall                      # system call

            addi    a7, x0, 1          # call code, print int
            mv      a0, t2             # value for int to print
            ecall                      # system call


fine:       li  a7,10
            ecall
```