Laboratorio di Architettura degli Elaboratori A - Esercitazione (31/3/03)

I. Editare il seguente programma, usando un qualsiasi editor, memorizzarlo in un semplice file di solo testo, ad esempio `prova.s`, e provare ad eseguirlo tramite `pcspim`. Questo esercizio ha lo scopo di

* verificare la conoscenza dei primi elementi di programmazione MIPS/SPIM (uso di direttive, etichette, e semplici syscall)
* usare il simulatore SPIM, e in particolare provare a caricare ed eseguire un programma Assembler, e ad usare le finestre Register, Text, Data, Messaggi e la console

```
        .data
str:   .asciiz "L'intero e' "
newline: .asciiz "\n"

       .text
main:
       li $v0, 4
       la $a0, str
       syscall

       li $v0, 1
       li $a0, 10
       syscall

       li $v0, 4
       la $a0, newline
       syscall

       jr $ra
```

1. Esegurire il programma (*Go*). Il programma parte da 0x00400000 con un preambolo che esegue una chiamata alla funzione `main`. Il programma caricato deve quindi contenere l'etichetta `main`, e viene caricato a partire dall'indirizzo 0x400020.
2. Verificare la traduzione del programma, e la trasformazione da pseudoistruzioni a istruzioni macchina (finestra Text).
3. Verificare l'allocazione dei dati globali in memoria (finestra Dati).
4. Eseguire il programma passo passo (*Single Step*), controllando la modifica dei registri (finestra Registri)
5. Reinizializzare memoria e registri, e ricaricare il programma.
6. Fissare un *breakpoint*, ed eseguire il programma (per fissare il *breakpoint* e' necessario dare l'indirizzo dell'istruzione su cui vogliamo bloccare l'esecuzione)

Modificare il programma, in modo che

7. **l'intero da stampare venga letto da tastiera**
8. **si faccia l'eco di una stringa inserita da tastiera**

Tabella delle chiamate di sistema:

Servizio	**Codice**	**Argomenti**	**Risultato**
Stampa intero	1	$a0 = intero	
Stampa float	2	$f12 = float	
Stampa double	3	$f12 = double	
Stampa stringa	4	$a0 = stringa	
Leggi intero	5		intero (in $v0)
Leggi float	6		float (in $f0)
Leggi double	7		double (in $f0)
Leggi stringa	8	$a0=buffer, $a1=lunghezza	
Sbrk	9	$a0=quantità	indirizzo (in $v0)
Exit	10		

II. Trascrivere il seguente programma che dato un array, calcola quanti sono gli elementi dispari. Completare la parte relativa alla procedura *pariodispari* e provarne il funzionamento, magari cambiando i valori dell'array e/o la sua lunghezza.

```
# Programma che dato un array calcola quanti sono gli elementi dispari

   .data
   array: .word 3,7,2,4,8  #array int a[5] (a[1]=3, a[2]=7 etc etc)
   annuncio: .asciiz "Il risultato e' "

   .text
```

```
main:
addi $sp,$sp,-4                 # PUSH(ra)
sw $ra,0($sp)                   #

add $s0,$zero,$zero             #pone s0=0
add $s2,$zero,$zero             #pone s2=0
addi $v0,$zero,1                #pone v0=1
la $a0,array                    #a0 e' l'inizio dell'array

ciclo:
lw $s0,0($a0)                   #s0= un elemento dell'array
jal pariodispari                #chiama la procedura pariodispari
add $s2,$s2,$s1                 #somma s1 a s2
slti $v1,$v0,5                  #v0 e' minore di 5?
beq $v1,$zero,esci              #se cosi' non e', esci
addi $v0,$v0,1                  #altrimenti, avanza nell'array
addi $a0,$a0,4
j ciclo                         #e continua il ciclo

esci:
addi $v0,$zero,4                #codice stampa stringa
la $a0,annuncio                 #stringa
syscall                         #stampa

addi $v0,$zero,1                #codice stampa intero
add $a0,$zero,$s2               #metti s2 in a0
syscall                         #stampa

lw $ra,0($sp)                   #POP (in ra)
addi $sp,$sp,4                  #

jr $ra                          #fine

# PROCEDURA pariodispari
# Calcola se un numero e' pari o dispari
# INPUT: $s0 (il numero da controllare)
# OUTPUT: $s1 (da' zero se il numero e' pari, uno se e' dispari)

pariodispari:

# COMPLETATELA VOI

jr $ra                          #fine della procedura pariodispari
```

III. Scrivere un programma che calcola i primi 40 termini della sequenza di Fibonacci e la memorizza in un array.

La sequenza di Fibonacci e' la sequenza dei numeri {1, 1, 2, 3, 5, 8, 13, etc} tali che:

$a_{n+2} = a_{n+1} + a_n$, con

$a_0 = 1$ e $a_1 = 1$

```
Versione in C
#include <stdio.h>
int Array[40];
int main() {
      int n = 2;
      Array[0] = 1;
      Array[1] = 1;
      for (int i = 2, i <40, i++)
             Array[n] = Array[n-1] + Array[n-2];
 return 0;
       }
```

IV provare a scrivernee la versione ricorsiva che richiede in input da tastiera il numero d'ordine dell'elemento della serie. Ad esempio se n=8 l'output sara' 21

```
Versione in C
#include <stdio.h>

int fib(int n){
       if(n == 0) return 1;
       if(n == 1) return 1;
       return fib(n-1) + fib (n-2);
       }

int main() {
     cin << n;  // passa in input il numero n
     cout << fib(n); // restituisce in output l'n-esimo elemento della serie.
         }
```