MIPS & SPIM

Modulo del Corso di Architettura degli Elaboratori

Nicola Paoletti

Università di Camerino
Scuola di Scienze e Tecnologie

15 Maggio 2013
AA 2012/2013

Riepilogo

Homeworks (1/3)

Adding machine

Scrivere e testare un programma MIPS che legge in continuazione un intero e lo somma ai precedenti. Non appena legge uno zero, il programma termina stampando la somma ottenuta fino a tal punto. In altre parole:

```
char c;
int somma=0;
do{
   c=getchar();
   somma+=atoi(c);
}
while (c != '0');
printf("Somma=%d\n", somma);
```

Homeworks (2/3)

Potenza iterativa

Scrivere una procedura MIPS che calcola la potenza a^b, con $a \in \mathbb{N}^+, b \in \mathbb{N}$ in **modo iterativo**, ovvero:

```
int potenza_iterativa(int a, int b)
{
        if(a < 1 || b < 0)
                return 0;
        else{
                int ret = 1;
                for(int i=0; i<b; i++)
                        ret = ret * a;
                return ret;
        }
}
```

Homeworks (3/3)

Potenza ricorsiva

Scrivere una procedura MIPS che calcola la potenza a^b, con $a \in \mathbb{N}^+, b \in \mathbb{N}$ in **modo ricorsivo**, ovvero:

```
int potenza_ricorsiva(int a, int b)
{

        if(a < 1 || b < 0)
                return 0;
        else if(b == 0)
                return 1;
        else
                return potenza_ricorsiva(a,b-1)*a;

}
```

Riepilogo

Adding machine (1/2)

Adding machine

Scrivere e testare un programma MIPS che legge in continuazione un intero e lo somma ai precedenti. Non appena legge uno zero, il programma termina stampando la somma ottenuta fino a tal punto. In altre parole:

```
char c;
int somma=0;
do{
   c=getchar();
   somma+=atoi(c);
}
while (c != '0');
printf("Somma=%d\n", somma);
```

Adding machine (2/2)

```
   .data
str: .asciiz "Somma = "

   .text
   .globl main
main:
   li $t0, 0
loop:
   li $v0, 5              #read_int
   syscall
   add $t0, $t0, $v0      #$t0+=$v0
   bne $v0, $zero, loop   #$v0 != 0 -> loop
   la $a0, str
   li $v0, 4
   syscall                #stampa str
   move $a0, $t0
   li $v0, 1
   syscall                #stampa la somma
   li $v0, 10
   syscall                #esce
```

Listing 1: Adding machine in MIPS

Potenza iterativa (1/4)

Potenza iterativa

Scrivere una procedura MIPS che calcola la potenza a^b, con $a \in \mathbb{N}^+, b \in \mathbb{N}$ in **modo iterativo**, ovvero:

```
int potenza_iterativa(int a, int b)
{
        if(a < 1 || b < 0)
                return 0;
        else{
                int ret = 1;
                for(int i=0; i<b; i++)
                        ret = ret * a;
                return ret;
        }
}
```

Potenza iterativa (2/4)

```
   .data
A:      .word 0xA   #a=10
B:      .word 0x3   #b=3
str1:   .asciiz " alla "
str2:   .asciiz " = "
str3:   .asciiz "\n"

   .text
   .globl main
main:   li $t0,0            #(ritorna 0)
        lw $a0,B            #Carico B in $a0
        bltz $a0, print     #B < 0 -> print
        lw $a1, A           #Carico A in $a1
        blez $a1, print     #A < 1 (A<=0)-> print
        li $t0, 1           #t0 = 1
for:    ...
```

Listing 2: Potenza iterativa in MIPS - Parte1

Potenza iterativa (3/4)

```
for:    beq $a0, $0, print   #B==0 -> end for
        mul $t0, $t0, $a1    #v0 = v0*A
        subu $a0, $a0, 1     #B=B-1
        j for
print:  lw $a0, A
        li $v0, 1            #system call code per print_int
        syscall              #"A"
        la $a0, str1
        li $v0, 4            #system call code per print_str
        syscall              #"A alla "
        lw $a0, B
        li $v0, 1
        syscall              #"A alla B"
        ...
```

Listing 3: Potenza iterativa in MIPS - Parte2

Potenza iterativa (4/4)

```
...
la $a0, str2
li $v0, 4
syscall          #"A alla B = "
move $a0, $t0    #v0=t0
li $v0, 1
syscall          #"A alla B = A^B"
la $a0, str3
li $v0, 4
syscall          #"A alla B = A^B\n"
li $v0,10        #codice di uscita
syscall
```

Listing 4: Potenza iterativa in MIPS - Parte3

Potenza ricorsiva (1/4)

Potenza ricorsiva

Scrivere una procedura MIPS che calcola la potenza a^b, con $a \in \mathbb{N}^+, b \in \mathbb{N}$ in **modo ricorsivo**, ovvero:

```
int potenza_ricorsiva(int a, int b)
{

        if(a < 1 || b < 0)
                return 0;
        else if(b == 0)
                return 1;
        else
                return potenza_ricorsiva(a,b-1)*a;

}
```

Potenza ricorsiva (2/4)

```
   .data
...

   .text
   .globl main
main:   li $v0, 0           #(ritorna 0)
        lw $a0, B           #Carico B in $a0
        bltz $a0, print     #B < 0 -> print
        lw $t0, A           #Carico A in $t0
        li $t1, 1
        blez $t0, print     #A < 1 (A<=0)-> print
        jal pow             #Chiama la funzione pow
print:  ...
```

Listing 5: Potenza ricorsiva in MIPS - Parte1

Potenza ricorsiva (3/4)

```
    .text
pow:    subu $sp,$sp,32         #Stack frame di 32 bytes
        sw $ra,20($sp)          #Salva l'indirizzo di ritorno
        sw $fp,16($sp)          #Salva il frame pointer
        addiu $fp,$sp,28        #Inizializza il frame pointer
        bgtz $a0,$L2            #B>0 -> L2
        li $v0,1                #altrimenti ritorna 1
        jr $L1
```

Listing 6: Potenza ricorsiva in MIPS - Parte2

Potenza ricorsiva (4/4)

```
$L2:    subu $a0,$a0,1          #Calcola B - 1
        jal pow                 #Chiama pow
        lw $v1,A                #Carica A
        mul $v0,$v0,$v1         #Calcola pow(A,B-1) * A
$L1:    lw $ra, 20($sp)         #Ripristina $ra
        lw $fp, 16($sp)         #Ripristina il frame pointer
        addiu $sp, $sp, 32      #Libera lo stack frame
        jr $ra                  #Rit. controllo al caller
```

Listing 7: Potenza ricorsiva in MIPS - Parte3