

Università degli studi di Bologna
Corso di laurea in Informatica

Documentazione del progetto di architettura degli
elaboratori
AA. 2005-2006

“MASTER MIND”

di

Fabio Quinzi

Alfredo Saglimbeni

Gabriele Verardi

Docenti : Prof. Ruben Scardovelli
Prof. Roberto Confalonieri

Introduzione:

La presente relazione ha lo scopo di analizzare e spiegare il codice sviluppato per il progetto in esame.

Il progetto è stato sviluppato in osservanza delle specifiche indicate dal prof. Roberto Confalonieri.

Mastermind:

Mastermind è un gioco di logica, che consiste nell'indovinare una sequenza di colori, scegliendo fra quelli disponibili.

La sequenza viene decisa prima dell'inizio del gioco da un altro giocatore (giocatore fittizio), che sceglie quattro colori diversi, e li inserisce in un ordine a piacere.

Lo scopo del gioco è indovinare la giusta sequenza, procedendo a tentativi (10 al massimo).

Ad ogni tentativo il gioco indicherà se sono stati indovinati dei colori, e se sono nella posizione giusta. La sequenza, infatti, deve rispettare sia la scelta di colori, sia la loro posizione all'interno della sequenza.

Se la sequenza inserita conterrà un giusto colore, il programma lo indicherà con |?|.

Se, inoltre, il colore sarà nella posizione corretta, il programma lo indicherà con |!|.

Il caso di errore (colore sbagliato), verrà indicato con |x|.

Sviluppo del codice:

A seguire, verranno esposte e descritte le parti principali del codice, prestando attenzione alle particolari scelte effettuate dal nostro gruppo.

- Si definiscono prima di tutto una serie di funzioni generali predefinite, create allo scopo di diminuire la quantità di codice all'interno dell'applicazione. Queste funzioni sono definite di seguito:

```
" #stampa stringhe
stampas: li $v0,4
        syscall
        jr $ra
#stampa interi
stampai: li $v0,1
        syscall
        jr $ra
#legge stringa
leggis: li $v0,8
        syscall
        jr $ra
#leggi integer
leggii: li $v0,5
        syscall
        jr $ra"
```

stampas: \$a0 viene definito l'indirizzo della stringa da stampare, prima di chiamare questa funzione. A questo punto la funzione setta il registro \$v0 a 4, che, con la chiamata di sistema, stampa una stringa a video.

stampai: Stampa un numero intero. Il numero da stampare viene inserito nel registro \$a0, in seguito chiamando la funzione, setta \$v0 a 1 così stampando il numero scelto con la chiamata di sistema.

leggis: legge una stringa. Prima della chiamata viene definito in ingresso nel registro \$a0 l'indirizzo di memoria dove salvare la stringa, e in \$a1 la lunghezza della stringa. La funzione così setta a 8 \$v0 ed effettua la syscall.

leggii: legge un numero intero. Prima della chiamata non viene definito nessun registro di ingresso, infatti settando \$v0 a 5, ed effettuando la chiamata di sistema, il numero inserito viene automaticamente salvato in \$v0.

- *Menù iniziale di scelta*: il gioco si apre con un menù, con quattro possibilità di scelta, descritto dal seguente codice:

```
"main:   la $a0,menu
        jal stampas

switch:  la $a0,menuPrompt
        jal stampas
        jal leggii
        beq $v0,1,inizio
        beq $v0,2,istruz
        beq $v0,3,credit
        beq $v0,4,esci
        j switch"
```

```
__ --^*@ MASTER MIND ||| MENU DI INIZIO @*^-- __
```

-Menu:

- 1) Inizi a Giocare
- 2) Istruzioni gioco
- 3) Credit
- 4) Esci

Scegli (1, 2, 3, 4):

La descrizione è molto semplice, sorvolando gli elementi grafici, nel codice in esame prendiamo in considerazione lo switch, che attende un numero in ingresso compreso tra 1 e 4, che serve per indirizzare l'utente alle varie sezioni del gioco.

In caso vengano inseriti caratteri sbagliati o fuori dal range, l'applicazione continua a ciclare in switch, presentando il menù finché non viene inserito un numero corretto.

- *Il giocatore fittizio*: prima dell'inizio del gioco, secondo specifiche del progetto, è stato implementato un giocatore fittizio, il cui compito è inserire la sequenza di colori che in seguito il giocatore reale dovrà indovinare.

```

"inizio: la $a0, capolungo
jal stampas
la $a0,fittiz
jal stampas
la $a0,soluz #giocatore fittizio
inseririsce la sequenza da indovinare
li $a1,5
jal leggis
jal fcontrolseq"

```

```

_..--^*@ MASTER MIND ||| GIOCATORE FITTIZIO @*^--._

```

I colori consentiti sono:
(r)-rosso (g)-giallo (v)-verde (b)-blu (n)-nero (a)-arancione
Inserisci la sequenza di colori da indovinare: □

Nella sequenza di codice rappresentato, il giocatore fittizio inserisce attraverso la tastiera la sequenza da indovinare. Viene prima di tutto impostato nel registro \$a0 l'indirizzo su cui verrà salvata la sequenza di colori *soluz* e la lunghezza della stringa in \$a1 pari a 5 (4 caratteri + null), viene così chiamata la funzione per leggere una stringa da tastiera. L'operazione successiva *fcontrolseq* è molto importante in quanto verifica che la sequenza inserita dal giocatore fittizio sia corretta o meno. Descriveremo questa parte di codice particolare di seguito.

Il codice, per il controllo che la sequenza inserita dal giocatore fittizio sia corretta, verifica che i caratteri inseriti siano corrispondenti a quelli possibili (*r* rosso *g* giallo *v* verde *n* nero *b* blu *a* arancio), in caso contrario chiede di inserire nuovamente la sequenza di caratteri. Il controllo verifica inoltre, che nella sequenza di caratteri non ci siano colori doppi (es. rggv oppure rgvr) anche in questo caso viene richiesto di inserire una nuova stringa di colori. Analizziamo quindi la parte di codice in esame:

```

"fcontrolseq: li $t0,0
fctrloop1: addu $t1,$t0,1
            lb $t2, soluz($t0)
fctrloop2: lb $t3, soluz($t1)
            addi $t1,1
            beq $t2,$t3,fseqerr
            blt $t1,4,fctrloop2
            li $t3,0
fctrloop3: lw $t4, colori($t3)
            lb $t5,($t4)
            addi $t3,4
            beq $t5,$t2,fseqend
            blt $t3,24,fctrloop3
fseqerr:   j inizio
fseqend:   addi $t0,1
            blt $t0,4,fctrloop1
            j $ra"
.....
"colori: .word rosso, giallo, verde, nero2, blu, arancio
rosso: .asciiz "rosso "
giallo: .asciiz "giallo "
verde: .asciiz "verde "
nero2: .asciiz "nero "
blu: .asciiz "blu "
arancio: .asciiz "arancione ""

```

Definiamo prima di tutto l'utilizzo dei registri:
\$t0 → è il nostro puntatore alla stringa *soluz*.
\$t1 → è il puntatore che scandisce la stringa *soluz* all'interno del ciclo *fctrloop2*, prima dell'entrata al ciclo viene settata sempre una posizione avanti rispetto a \$t0.
\$t2 → viene caricato il carattere a cui punta \$t0 che verrà esaminata nel ciclo *ctrloop2*, se ha doppi in nella stringa, e in *ctrloop3*, se è un carattere corretto.
\$t3 → viene caricato il carattere che dovrà essere confrontato nel ciclo *ctrloop2* con \$t2, in caso di uguaglianza siamo in errore, in quanto il carattere ha doppi e sarà necessario chiedere una nuova sequenza caratteri, altrimenti passa al controllo che il carattere sia corretto. In questo caso \$t3 viene riutilizzato come puntatore all'array di indirizzi di memoria definito come *colori*.
\$t4 → Carica l'indirizzo in cui si trova il colore.
\$t5 → carica la prima lettera della stringa del colore a cui punta \$t4, questa verrà confrontata con \$t2. Se durante il ciclo troverà un'uguaglianza, allora il carattere è corretto e l'applicazione continuerà la sua esecuzione dopo il controllo; in caso contrario la lettera inserita non è associata a nessun colore e andrà nuovamente in errore.

- *Giocatore Reale*: in questa parte di codice inizia il gioco vero e proprio del Master Mind. Il giocatore reale avrà a disposizione 10 tentativi per indovinare la sequenza di colori corretta. La costruzione della tabellina grafica, aiuterà il giocatore a capire quali colori sono stati indovinati, quali sono corretti, ma nelle posizioni sbagliate, e quali sono completamente errati.

```

__--^*@ MASTER MIND ||| INIZIA IL GIOCO @*^--__
----- | N TENTATIVO | INDOVINA |
|x| |x| |x| |x| | Rimanenti 9 | nabr |
|x| |x| |?| |?| | Rimanenti 8 | ngar |
|x| |!| |x| |?| | Rimanenti 7 | rgva |
!| |!| |!| |x| | Rimanenti 6 | rgwb |
!| |!| |!| |!|

```

Per questioni di ordine, si è deciso di utilizzare dei registri \$s# dove verranno impostate delle variabili che verranno utilizzate nell'arco dell'esecuzione di tutta l'applicazione. Al contrario, i registri \$t# verranno utilizzati localmente, all'interno dei vari cicli di controllo. Il loro utilizzo verrà descritto nelle pagine seguenti:

```

"li $s3,10
start:subu $s3,$s3,1
    la $a0,tenta
    jal stampas
    move $a0,$s3
    jal stampai
    la $a0,spazio
    jal stampas
    la $a0,indov
    li $a1,5
    jal leggis
    la $a0,fintab
    jal stampas
    jal rcontrolseq"

```

In questa parte di codice viene costruita la parte iniziale della grafica, per rendere intuitivo il gioco all'utente. Due sono gli elementi fondamentali da descrivere: in \$s3 viene impostato il contatore dei tentativi, che andrà da 9 a 0, e stampato a schermo nella tabellina grafica. Ad ogni tentativo il giocatore reale inserirà una sequenza di colori, che verrà salvata, come si vede nel codice, in *indov*. Come per la stringa *soluz* del giocatore fittizio anche in questo caso viene verificata la correttezza nella sequenza caratteri. A seguito di tale richiesta incomincerà il controllo sulla stringa inserita, per verificare se corrisponde a quella da indovinare.

Il codice per il controllo per la sequenza inserita dal giocatore reale, è identica a quella già descritta per il giocatore fittizio. Si distingue solo dal fatto che analizza la stringa *indov* e nell'uscita dal controllo dal controllo in caso di errore viene reimpostato il numero di tentativi e stampato a schermo un messaggio di errore. Si precisa che nel caso l'utente inserisca caratteri errati o inserisce caratteri doppi nella sequenza della stringa, si è deciso di penalizzarlo in ogni caso diminuendo lo stesso contatore dei tentativi.

```

"rcontrolseq: li $t0,0
rctrloop1: addu $t1,$t0,1
            lb $t2, indov($t0)
rctrloop2: lb $t3, indov($t1)
            addi $t1,1
            beq $t2,$t3,rseqerr
            blt $t1,4,rctrloop2
            li $t3,0
rctrloop3: lw $t4, colori($t3)
            lb $t5,($t4)
            addi $t3,4
            beq $t5,$t2,rseqend
            blt $t3,24,rctrloop3
rseqerr:   la $a0, frase3
            jal stampas
            j tentctr
rseqend:  addi $t0,1
            blt $t0,4,rctrloop1
            j $ra"

```

Il controllo sulla sequenza da indovinare si svilupperà su due verifiche: che i colori siano corretti, e che siano nella posizione giusta. Verranno stampate a schermo, rispettivamente la stringa "|!" se il colore è giusto nella posizione corretta, "|?" se il colore è giusto nella posizione sbagliata, "|x|" se il colore è completamente sbagliato.

