Calcolatori Elettronici
Prova di Laboratorio - assembly MIPS: array_1D
4 Aprile 2012

Introduzione

Lo scopo di questa prova di laboratorio è lo sviluppo di un semplice programma nel linguaggio assembly del processore MIPS. Non è richiesta una particolare base di conoscenze algoritmiche, ma semplicemente un minimo di dimestichezza con la programmazione assembly.

Istruzioni

Cominciate facendo il login sulla macchina del laboratorio che vi è stata assegnata. Per il login occorre usare matricola e password dello *student portal*. Sul desktop troverete una cartella contenente i simulatori QtSpim e Mars. Lanciate ed utilizzate quello che preferite. Tutto il vostro codice (sia esso costituito da un singolo file, o da file multipli) andrà salvato nella cartella "`mips`" da creare sul drive `H:` .

Create un file `student-info.txt` con incluso il vostro nome e cognome e numero di matricola nella cartella "`mips`". Per maggior sicurezza, includete anche nome, cognome e matricola come commento, in testa ad ogni file sorgente. Alla fine della prova, i file saranno prelevati automaticamente dalla directory . Tutto quello che lascerete nella cartella `mips` sarà utilizzato per la valutazione. Salvare i vostri file altrove, o non indicare nome e cognome, porterà inevitabilmente all'annullamento della vostra prova. *Tutti i file all'esterno della cartella verranno cancellati automaticamente!!!*

Informazioni generali

La prova ***non è*** a correzione automatica. Tutti gli studenti autori di un codice che viene assemblato senza errori e risponde alle specifiche indicate alla sezione seguente saranno ammessi all'orale. In quella sede, il codice prodotto sarà esaminato e discusso col docente.

Le specifiche

Dovete scrivere un programma assembly che inizializza un array monodimensionale, e successivamente accede all'elemento corrispondente all'indice fornito in input.

Si allochi staticamente in area dati globale un array di 20 interi. Il main richiama una funzione `array_init`, fornendole come unico parametro il puntatore all'array. La funzione inizializza ogni elemento A[i] dell'array al valore i*6+1. Al ritorno dalla funzione, il main legge da tastiera l'intero i, e richiama la funzione `array_element`, che ha due parametri (indirizzo base dell'array ed il valore di i) e ritorna l'intero A[i]. L'accesso viene ripetuto in ciclo finché l'utente non immette un indice -1.

Per maggior chiarezza, il seguente è un programma C dalle funzionalità analoghe al codice richiesto

```
int A[20];

void array_init(int *arr) {
        int i;

        for (i=0; i<20; i++)
           arr[i]=i*6+1;
    return;
}

int array_element(int *arr, int r) {
        return(arr[r]);
}


void main() {
        int i,element;

        printf("Inizializzo l'array...");
        array_init(A);
        printf(" fatto!\n\n");
loop:   printf("Numero dell'elemento (0-19, -1 per terminare): ");
        scanf("%d",&i);
        if (i== -1)  exit(0);
        element=array_element(A,i);
        printf("L'elemento di posto %d vale %d.\n\n",i,element);
        goto loop;
}
```

Facoltativo: chi è capace può provare a scrivere un programma che non faccia uso di codici di moltiplicazione (mult, multu, mul, . . .)

Suggerimenti

Potete leggere i interi in input da tastiera utilizzando la system call 5. Per la stampa di stringhe e di interi potete usare le syscall 4 e 1, rispettivamente.

Il seguente è un output di esempio:

```
Inizializzo l'array... fatto!

Numero dell'elemento (0-19, -1 per terminare): 3
L'elemento di posto 3 vale 19.

Numero dell'elemento (0-19, -1 per terminare): -1


-- program is finished running --
```

Valutazione

Scrivere un programma funzionante, che faccia uso di due funzioni come richiesto nelle specifiche e che segua le convenzioni di salvataggio dei registri è strettamente necessario per essere ammessi a sostenere l'orale. In quella sede, si entrerà nel dettaglio della struttura del codice, dando una valutazione migliore a soluzioni "pulite" e ben commentate.