Tutorato di Informatica

Foglio 3+ - Assembly

Matematici I Anno

25 Marzo 2026

"Me: I'm so sorry, my dog ate my homework.
Computer Science Professor: your dog ate your coding assignment?
Me:
Prof:
Me: it took him a couple bytes."

Esercizi di Ripasso e Approfondimento su Assembly

Esercizio 01: Ricerca di un elemento in un Array

Completare il programma `ricerca_in_array.asm` che cerca il valore **5** all'interno di un array di 10 numeri interi. Caricare il file su QtSpim, eseguirlo passo passo e verificarne il funzionamento.
Test: Per verificare la robustezza del codice, provare a modificare il contenuto del vettore nel segmento `.data` affinché contenga (o non contenga affatto) il valore cercato.

Esercizio 02: Interazione Utente e Confronto

Scrivere un programma in assembly che esegua le seguenti operazioni:

1. Leggere due numeri interi (`num1` e `num2`) usando l'apposita **syscall**.
2. Confrontare i due valori inseriti.
3. Stampare a schermo il più **piccolo** (il minimo) dei due.
4. Caricare il codice su QtSpim e testarne la correttezza.

Extra: Aggiungere delle stampe a video (istruzioni per l'utente) prima di ogni lettura e prima del risultato finale (usando le direttive `.data` e `.asciiz`).

Esercizio 03: Passaggio di parametri (Per Valore vs Per Indirizzo)

Testo: L'obiettivo di questo esercizio è comprendere la differenza tra il passaggio di parametri per *valore* e per *riferimento (o indirizzo)*.

1. Scaricare il file `somma1.asm`. Leggere il codice, caricarlo su QtSpim ed eseguirlo passo passo (usando il tasto F10 o "Step"). Osservare cosa viene caricato nei registri `$a0` e `$a1` prima della chiamata a procedura (`jal`).
2. Scaricare il file `somma2.asm`. Fare la stessa cosa: caricarlo, eseguirlo passo passo e osservare il contenuto dei registri `$a0` e `$a1`.

3. Qual è la differenza fondamentale tra i due approcci dal punto di vista dell'accesso alla memoria all'interno della procedura?

Esercizio 04: Moduli Multipli e Visibilità delle Etichette

Testo: In progetti grandi, il codice assembly viene spesso diviso in più file. QtSpim permette di caricare più file consecutivamente nello stesso spazio di memoria.

1. Caricare su QtSpim **prima** il file `vector_mean.asm` e **poi** il file `mean_lib.asm`. Provare ad eseguire il programma.

2. Ora resettare completamente l'emulatore (File → Reinitialize and Load File). Caricare i file nell'ordine inverso: **prima** `mean_lib.asm` e **poi** `vector_mean.asm`. Eseguire il programma. Cosa succede e perché?

Esercizio 05: Costruzione Array e Media (Procedure Annidate)

Testo: Scrivere un programma completo in assembly MIPS che:

1. Dato un numero n definito in memoria, allochi uno spazio sufficiente per contenere un array di n interi (word).

2. Tramite un ciclo, popoli l'array con i numeri interi sequenziali da 1 a n (es. se $n = 5$, l'array conterrà `1, 2, 3, 4, 5`).

3. Chiami una procedura `mean` per calcolare la media dell'intero array, e la stampi a video.

Requisito: Si utilizzino le logiche delle procedure `sum` e `mean` viste precedentemente, **modificandole** affinché operino su dati di tipo `.word` (4 byte) invece che su `.byte` (1 byte). Caricare il file risolutivo su QtSpim e verificarne l'output.

Esercizio 06: Calcolo del Fattoriale (Procedura Ricorsiva)

Testo: Scrivere un programma in assembly MIPS che calcoli il fattoriale di un numero n (es. $n = 3$) utilizzando una procedura **ricorsiva**.

1. Definire il valore di n e lo spazio per il risultato nella sezione `.data`.

2. Chiamare una procedura `fact` passando n come argomento (nel registro `$a0`).

3. All'interno di `fact`, gestire il salvataggio dei registri sullo **Stack**. Poiché la procedura chiama sé stessa, è obbligatorio salvare ad ogni iterazione l'indirizzo di ritorno (`$ra`) e il valore corrente di n (`$a0`).

4. Implementare la logica matematica: se $n \leq 1$, il caso base ritorna 1. Altrimenti, la procedura richiama sé stessa calcolando $n \times \text{fact}(n-1)$.

5. Caricare il programma su QtSpim ed eseguirlo passo passo tenendo d'occhio il pannello dei dati (sezione Stack) per vedere come la memoria si espande e si contrae.

Esercizio 07: Inserimento in un Array (Codice da completare)

Testo: In questo esercizio vogliamo inserire un nuovo valore all'interno di un array preesistente, in una posizione specifica. Per fare spazio al nuovo elemento senza sovrascrivere i dati esistenti, è necessario "shiftare" (traslare) tutti gli elementi successivi di una posizione verso destra (ovvero di 4 byte in avanti).
Cosa fare: Scaricate il file `inserimento_array.asm` dalla pagina Moodle del corso. Il codice

fornito è parziale: mancano le istruzioni cruciali all'interno del ciclo di shift e per l'inserimento finale. Sostituite i commenti contrassegnati da `# -- COMPLETA QUI --` con le istruzioni MIPS corrette. Infine, caricate lo script su QtSpim ed eseguitelo passo passo per verificare che l'array finale diventi `10, 20, 99, 30, 40`.

Esercizio 08: Conto alla Rovescia e Rimozione per Shift

Testo: Scrivere un programma in assembly MIPS che implementi un conto alla rovescia interattivo seguito dalla rimozione di un elemento da un array.

Il programma deve soddisfare i seguenti requisiti:

- Inizializzare in memoria un array di 5 interi (ad esempio `8, 15, 23, 42, 108`) e una variabile contenente la sua dimensione.
- Richiedere all'utente un numero di partenza tramite console.
- Implementare un ciclo che stampi a video un conto alla rovescia dal numero inserito fino a 0.
- Al termine del conteggio, il programma deve rimuovere fisicamente l'elemento in **posizione 1** (ovvero il secondo elemento dell'array, nell'esempio il `15`).