Tutorato di Informatica

Foglio 2 - QtSpim e Assembly

Matematici I Anno

11 Marzo 2026

"What do you call a group of computer scientists? An Assembly."

Codifica delle Istruzioni e Formati MIPS

Esercizio 01

Testo: Si consideri l'istruzione `j xx`. Occupa più spazio in memoria la sua rappresentazione ASCII o l'istruzione macchina corrispondente dopo che è stata assemblata?

Esercizio 02

Testo: Interpretare il codice esadecimale `0x0232502A` come istruzione MIPS, ed indicare:

- La sua rappresentazione binaria
- L'opcode ed il formato (R-type, I-type o J-type)
- La rappresentazione simbolica dell'istruzione, completa del valore dei suoi parametri

Nota: usare l'appendice del libro a pagina A50.

Esercizio 03

Testo: Indicare il formato dell'istruzione per effettuare la differenza tra due registri, ed esprimere esplicitamente l'operazione che effettua una sottrazione tra il valore nel registro $3 e quello nel registro $4, e deposita il risultato nel registro $2:

- in forma simbolica
- in forma binaria
- in forma esadecimale

Esercizio 04

Testo: Stiamo eseguendo il seguente programma:

Indirizzo	Istruzione (parziale)
`0x00400000`	`add $9, ...`
`0x00400004`	`sub $14, ...`
`0x00400008`	`lw $8, ...`
`0x0040000C`	`add $7, ...`

e ad un certo punto il Program Counter (PC) contiene il valore `0000 0000 0011 1111 1111 1111 1010 0100`. Inoltre sta per essere eseguita un'istruzione con opcode `000010`. Come devono essere i bit rimanenti di questa istruzione, affinché dopo venga eseguita l'istruzione `lw $8, ...`?

Esercizio 05

Testo: Qual è l'indirizzo della più lontana cella di memoria in cui è possibile saltare (in avanti) con una istruzione `bne` se PC = `0010 0010 1100 0011 0110 0010 0100 0000`?

Esercizio 06

Testo: Determinare a quale istruzione macchina MIPS corrisponde la sequenza binaria:
`00000000011010010010000000100000`

Esercizio 07

Testo: Determinare a quale istruzione macchina MIPS corrisponde la sequenza binaria:
`00100010111010110000000001100000`

Esercizio 08

Testo: A quale istruzione macchina MIPS corrisponde il codice esadecimale: `0x8fa40000`

Programmazione Assembly

Esercizio 09

Testo: Scrivere un programma che calcoli la somma del valore che è memorizzato nel registro $17, e del valore memorizzato nella locazione di memoria che si trova 14 word più avanti dell'indirizzo specificato al registro $16, e memorizzare il risultato 3 word di memoria più avanti rispetto alla locazione attuale del secondo operando. Nota: Vedi le istruzioni `lw` e `sw`, e ricorda che puoi usare la sintassi $n(x)$.

Esercizio 10

Testo: I valori relativi alle variabili della dimensione di una word a, b, c, d sono memorizzati di seguito in memoria a partire dall'indirizzo specificato dal contenuto del registro $10. Scrivere la sequenza di istruzioni assembly che aggiunge la costante 10 alle variabili a, b, c, d e salva i nuovi valori delle variabili in memoria.

Esercizio 11

Testo: In QtSpim, scrivere (manualmente) in memoria, nella sezione Data, la stringa seguente: `"Hello world!"`. La stringa deve apparire leggibile nel giusto ordine nell'interfaccia del pro-

gramma. NOTA: Usare la tabella ASCII (extended) per la codifica dei caratteri.

Esercizio 12

Testo: In QtSpim, inserire manualmente due valori nei registri $t0 e $t1. Poi scrivere un programma assembly che calcoli la somma dei due numeri presenti nei registri $t0 e $t1 e metta il risultato in $t2.

Esercizi su Emulatore QtSpim

Esercizio 13: La mia prima somma assembly

Scaricare il programma `la_mia_prima_somma_Assembly.asm` da Moodle, caricarlo ed eseguirlo passo passo in QtSpim. Annotare, ad ogni passaggio (istruzione), il valore del PC e dei registri usati, ed inoltre monitorare cosa succede nelle sezioni `.text` e `.data`.

Esercizio 14: Ricerca all'interno di un array

Completare il programma `ricerca_in_array.asm` che cerca il valore 5 all'interno di un array di 10 numeri interi. Farlo girare su QtSpim, passo passo e verificarne il funzionamento. Per verificare il funzionamento dell'intero programma, modificare il vettore perché contenga, o meno, il valore che si sta cercando.

Esercizio 15: Interazione utente e Salto Condizionato

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
3. Stampare a schermo il più piccolo dei due

dopodiché caricarlo su QtSpim e testarlo.
Consiglio: puoi partire dal file dell'esercizio precedente e prendere spunto.

Esercizio 16: Somma agli estremi di un array

Testo: Scrivere un programma in assembly MIPS che allochi in memoria un array chiamato `vet` contenente i seguenti 5 numeri interi: 2, 4, 6, 8, 10. Il programma deve poi:

1. Caricare l'indirizzo base dell'array in un registro temporaneo.
2. Leggere dalla memoria il primo elemento e l'ultimo elemento, salvandoli in due registri distinti.
3. Calcolare la somma di questi due valori e memorizzare il risultato finale in un terzo registro.

Dopodiché, salvare il codice in un file `.asm`, caricarlo su QtSpim ed eseguirlo passo passo per verificare che il valore finale nel registro di destinazione sia effettivamente 12.

Esercizi Extra Per i Più Coraggiosi (Lab 2)

01. L'Isomorfismo tra Dati e Istruzioni (Il Codice Automodificante)

Testo: Abbiamo visto che in MIPS ogni istruzione è codificata come una sequenza di 32 bit, esattamente come un numero intero (una `.word`). Dal punto di vista algebrico, c'è qualche differenza tra un'istruzione e un dato? Sarebbe teoricamente possibile per un programma MIPS calcolare un numero, salvarlo in memoria, e poi "saltare" su quel numero ed eseguirlo come se fosse codice? Prova a costruire un esempio.

02. La Sfida di Flavio Giuseppe e la Magia dello Shift Binario

Testo: Lo storico Flavio Giuseppe si salvò da un assedio disponendosi nel punto esatto di un cerchio di $n = 41$ persone, in cui, a turno, veniva eliminata una persona sì e una no. In quale posizione si mise per essere l'ultimo superstite?

Suggerimento: Prima di fare calcoli complessi, prova a disegnare cosa succede se nel cerchio ci sono 2, 4, 8 o 16 persone. Trova la regola matematica generale e poi spiega il sorprendente legame con il sistema binario: come si ottiene la soluzione in un millisecondo facendo un semplice *Left Circular Shift* sul numero n (ovvero prendendo il suo bit più a sinistra e spostandolo in fondo a destra)?

03. Il Minimalismo Estremo (Turing-Completezza con 1 sola istruzione)

Testo: Abbiamo visto molte istruzioni MIPS, ma qual è il numero *minimo* di comandi necessari a un processore per eseguire qualsiasi algoritmo possibile?

Dimostra che ne basta **uno solo**: usando l'istruzione teorica `subleq A, B, C` (che significa "sottrai A da B e salva in B; se il risultato è ≤ 0 salta all'indirizzo C"), fai vedere in che modo potresti azzerare una variabile (cioè fare $X = 0$) e come potresti simulare un salto incondizionato (l'equivalente di `j`).

04. La Sfida: Piegare il 2D in una RAM 1D

Testo: La memoria RAM di un computer è come un lunghissimo nastro monodimensionale (1D) di cellette numerate in fila. Ma in matematica e nei videogiochi, lavoriamo spesso con griglie o immagini bidimensionali (2D). Se volessimo salvare un'immagine nella RAM, avremmo bisogno di una funzione matematica che trasformi ogni coppia di coordinate (X, Y) in un singolo indirizzo Z (una biiezione da $\mathbb{N}^2$ a $\mathbb{N}$).

La sfida è questa: come possiamo mappare questa griglia in modo "intelligente"? Se usassimo un metodo banale (come contare riga per riga), due pixel vicinissimi nell'immagine 2D potrebbero finire salvati in indirizzi di memoria 1D lontanissimi tra loro, rallentando terribilmente il computer. Riesci a immaginare un modo per assegnare un numero Z a ogni cella in modo da preservare il più possibile la vicinanza spaziale?