Lo studente si colleghi al simulatore online[1] del linguaggio Assembly e segua le istruzioni riportate di seguito

Tutorial

1. Familiarizza con l'interfaccia del simulatore online[1] mostrato in Figura 1.

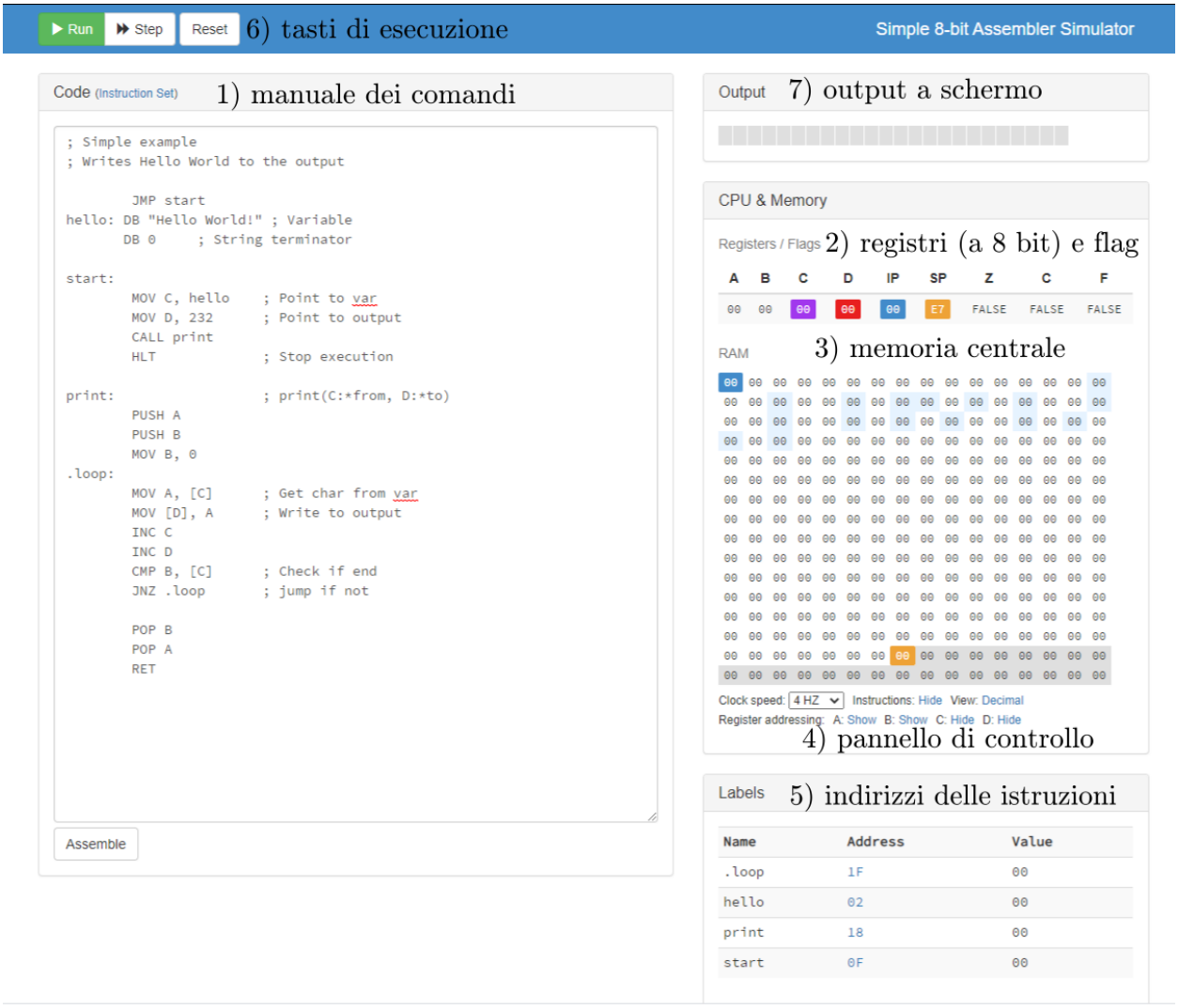

Figura 1: Simulatore del linguaggio Assembly a 8 bit disponibile online[1].

[1] schweigi.github.io/assembler-simulator

Sull'interfaccia di Fig. 1 ci sono:

1. Link al manuale dei comandi (Instruction Set[2]);
2. elenco dei registri (A, B, C, D) e dei flag (Z: *zero*, C: *carry*, F: *fault*[3]) del sumulatore. IP è l'*Istruction Pointer*, in alcuni casi chiamato anche *Program Counter* (PC), SP è lo *Stack Pointer* (pag. 112 del libro di testo). *Attenzione:* in questo simulatore, i registri sono da 8 bit (non 16).
3. memoria centrale: rappresenta i dati salvati sulla RAM. Da qui, l'Assembler estrae le istruzioni definite dal programmatore.
4. pannello di controllo: l'utente può mostrare o nascondere i registri e le istruzioni, cambiare la frequenza di clock e visualizzare i dati in decimale o in binario;
5. indirizzi delle istruzioni: elenca i blocchi di istruzioni del programma e la loro locazione in memoria centrale;
6. tasti di esecuzione: per avviare e interrompere l'esecuzione del programma, resettare o eseguire un'istruzione alla volta.

Gli Indirizzamenti (Intel x86)

In Assembly ci sono vari metodi di indirizzamento (Figura 18, pag. 132). Ecco alcuni esempi di indirizzamento:

- immediato (Figura 19, pag. 143):
  MOV AL, 4Dh
- diretto da registro a registro (Figura 20, pag. 143):
  MOV AX, BX
- diretto da registro a memoria e viceversa (Figura 21, pag. 144):
  MOV AX, [6Ah];
  MOV [6Ah], AX
- indiretto tramite registro (Figura 22, pag. 144):
  MOV BX, [AX]
- indiretto tramite registro e displacement (Figura 23, pag. 145):
  MOV BX, [AX+3];
  MOV [AX+3], 20
- indiretto tramite registri base e indice (Figura 24, pag. 145). Spesso usato per accedere agli array:
  MOV [BX][DI], AX
- indiretto tramite registro base, indice e displacement (Figura 25, pag. 146):
  MOV AX, 02[BX][SI]
- implicito che coinvolge lo *stack pointer*:
  PUSH AX
  POP BX

[2] schweigi.github.io/assembler-simulator/instruction-set
[3] Viene impostato a 1 (TRUE) se la CPU entra in uno stato di errore

Primo Livello: comprensione del codice

2. Aggiorna la pagina ed esegui il codice pre-caricato (**Run**). Cosa sta facendo il programma? Riconosci alcuni comandi visti in classe?

3. Usando il manuale dei comandi (link) individua il significato del conditional jump: JNZ.

4. Riscrivi nell'editor del simulatore il codice riportato di seguito. Nota la suddivisione in blocchi di istruzioni: **.confronta:** e **.finisci:**. Nota l'uso del "**;**" per inserire commenti.

```
start:
        MOV A, 8
        MOV B, 3
.confronta:
        CMP A, B
        JBE .finisci ;Jump if Below or Equal
        DEC A
        JMP .confronta
.finisci:
        HLT ;termina il programma
```

Riassumi a parole cosa fa il programma:

Secondo Livello: indirizzamento immediato, indirizzamento diretto da registro a registro, ADD e SUB

5. Scrivi un programma che carica nel registro A il numero decimale 12.

6. Avendo a disposizione solo registri a 8 bit, qual è il numero massimo (in decimale e in esadecimale) che può essere memorizzato in ciascun registro?

7. Immagina che al posto del registro A da 8 bit, ci sia un registro AX da 16 bit. Come scriveresti il programma *5* per salvare il numero decimale 12 nella parte alta di AX?

8. Scrivi un programma che carica nel registro A il numero decimale 5 e nel registro B il numero decimale 8. Esegui la somma e salva il risultato in C.

9. Modifica il programma precedente: esegui la differenza (**A-B**). Cosa succede ai flag?
   Z = ___
   C = ___

Terzo Livello: CMP e jump condizionali

10. Dati i registri A e B di 8 bit ed i Flag C (*carry*) e Z (*zero*), considera il seguente programma in Assembly:

```
start:
        MOV A, num1
        MOV B, num2
.confronta:
        CMP A, B
        Jx .salta ;jump condizionale
        HLT
.salta:
        MOV C, 1
```

In base ai valori di **num1** e **num2** ed ai tipi di jump riportati nelle tabella qui sotto, indica:
- quali sono i valori dei flag Z e C;
- se il jump condizionale avviene oppure no.

num1	num2		Z	C	Jx	Salta?
5	10	CMP A, B	0	1	JZ	NO
8	3	CMP A, B			JNZ	
2	2	CMP A, B			JA	
8	4	CMP A, B			JAE	
2	4	CMP A, B			JB	
8	7	CMP A, B			JBE	
2	2	CMP A, B			JE	
0x0C	0x0B	CMP A, B			JA	
0x0F	0x09	CMP A, B			JB	
0x10	0x09	CMP A, B			JBE	
0x1F	0x2B	CMP A, B			JB	

Tabella 1: Tabella dei Flag.

11. Verifica le tue risposte all'esercizio precedente riscrivendo il programma sul simulatore.

12. Aiutandoti col codice al punto *4* e con l'esempio di pag. 129 del libro di testo[4], scrivi il seguente programma: *dati due numeri decimali salvati in A e B, sommarli se sono diversi, sottrarli se sono uguali. In entrambi i casi, salvare il risultato in C.*

13. Aiutandoti col codice al punto 4. e con l'esempio di pag. 130 del libro di testo[4], scrivi il seguente programma: *dati due numeri decimali salvati in A e B: sommarli se il primo è minore o uguale del secondo altrimenti sottrarli. In entrambi i casi, salvare il risultato in C.*

[4]Attenzione: alcune istruzioni riportate nel libro di testo non sono supportate dal simulatore online. Consulta il manuale dei comandi per trovare il JUMP adatto al tuo scopo.

Quarto Livello: indirizzamento diretto da registro a memoria e viceversa; indirizzamento indiretto

14. Scrivi un programma che carichi un numero nella cella di memoria il cui indirizzo decimale è 100 (0x64) e successivamente lo copi nel registro D.

15. Scrivi un programma che carichi un numero nel registro A e successivamente lo copi nella memoria di indirizzo decimale 100 (0x64).

16. Dato il seguente programma in Assembly:

```
start:
        MOV B, 20 ;______
        MOV [15], B ;______
        MOV A, [15] ;______
        MOV [A+5], 200 ;______
        MOV [A-1], 100 ;_______
        HLT
```

Indica nei commenti al programma, i tipi di indirizzamento usati. Alla fine del programma, in quali celle della memoria centrale saranno salvati i numeri del programma?

MEMORIA CENTRALE

0							7							15
16							23							30

A =
B =

17. Verifica la tua risposta all'esercizio precedente riscrivendo il programma sul simulatore.

Quinto Livello: lo Stack

18. Dati i registri A, B, C, e D di 8 bit ed uno stack pointer (SP) che punta all'indirizzo 231, considera il seguente programma in Assembly:

```
start:
        istr1: MOV A, 3
        istr2: MOV D, 5
        istr3: ADD A, D
        istr4: PUSH A
        istr5: ADD A, D
        istr6: PUSH A
        istr7: ADD A, D
        istr8: PUSH A
        istr9: ADD D, A
        istr10: POP C
        istr11: POP B
        istr12: POP A
        istr13: HLT
```

Adesso scrivi nella tabella qui sotto i valori salvati nei registri, nello stack pointer (SP) e nello stack segment ad ogni istruzione. *Attenzione: i numeri in corsivo da 228 a 231 sono gli indirizzi dello stack segment.*

	A	**B**	**C**	**D**	**SP**	**Stack Segment**			
						228	*229*	*230*	*231*
istr1:	3	-	-	-	231	-	-	-	-
istr2:	3	-	-	5	231	-	-	-	-
istr3:									
istr4:									
istr5:									
istr6:									
istr7:									
istr8:									
istr9:									
istr10:									
istr11:									
istr12:									
istr13:									

19. Verifica le tue risposte all'esercizio precedente riscrivendo il programma Assembly sul simulatore. Osserva cosa succedere allo stack pointer e agli indirizzi in memoria centrale.

20. Scrivi un programma che memorizzi il numero 100 nel registro A e salvi nello stack segment i seguenti valori:
    a) la somma tra A e 50;
    b) il differenza tra il risultato ottenuto in a) e 20: (A-20).
    I risultati in a) e b) devono essere salvati nello stack segment.

21. Discuti con in tuoi compagni gli esercizi. Aiuta chi è in difficoltà. Approfondisci sul libro di testo.