# Esercizi per il corso di Architettura dei Calcolatori

## Anno accademico 2008/09

## Si prevede l'utilizzo dell'emulatore Eniac ( presentato a lezione )

**1)** Caricare nel registro accumulatore AX il contenuto della cella 7.
**2)** Caricare nel registro CX il numero nella cella indirizzata dalla cella 7.
**3)** Dato il seguente frammento di codice assembly (Eniac) :

```
0     LOAD 50
1     STORE BX
2     LOAD 0
3     STORE CX
4     LOAD BX
5     SUB 0
6     JZ 10
7     DEC BX
8     INC CX
9     JMP 4
10    HLT
```

Indicare il contenuto del registro CX al termine del codice. Indicare il contenuto di CX se si sostituisce il comando JZ con JS nella riga 6 e descrivere brevemente cosa è cambiato.

**4)** Dato il seguente programma che calcola i divisori di un numero (inizia a scrivere i divisori dalla cella di memoria il cui valore è contenuto nella cella 5) :

```
0     LOAD @4
1     STORE CX
2     LOAD @5
3     STORE DX
4     65
5     27
6     JMP 8
```

```
7     DEC CX
8     LOAD @4
9     INC BX
10    MOD BX
11    JNZ 7
12    LOAD BX
13    STORE @DX
14    INC DX
15    DEC CX
16    JNZ  8
17    HLT
```

Modificare il programma affinché controlli se il numero contenuto nella cella di memoria 4 è primo.
Per indicare se il numero è primo far sì che il registro AX contenga il valore 0, 1 altrimenti.

**5)** Scrivere un programma che prende 3 valori in input e calcola la somma dei primi 2, se questa è maggiore o uguale del 3° valore allora divide la somma per il terzo valore, altrimenti moltiplica la somma per un numero di volte pari al 3 valore. In entrambi i casi il risultato va memorizzato in una cella di memoria a scelta (indicare tale cella nel commento del codice prodotto).

**6)** Calcolare una potenza di un numero.

**7)** Scrivere un programma in BASE 10 che mi faccia ottenere in celle di memoria consecutive il triangolo mostrato di seguito :
1
11
111
1111
In input l'altezza del triangolo.
( si può iniziare a scriverlo da una qualunque cella di memoria)

**8)** scrivere un programma in base 10 che crei il seguente triangolo
1
22
333
4444
In input l'altezza del triangolo.

**9)** Calcolare il fattoriale di un numero

**10)** Calcolare i divisori di un numero.

**11)** Scrivere un programma assembly per ottenere, in celle di memoria consecutive a partire da una scelta arbitrariamente, il triangolo mostrato di seguito :
1111
111
11
1
In input l'altezza del triangolo.
Descrivere chiaramente TUTTE le assunzioni che vengono fatte per risolvere l'esercizio.

**12)** Scrivere un programma assembly per calcolare, dati due numeri A e B, i primi N termini della successione di Fibonacci. A, B, e N devono essere variabili del programma. P.es.:, se A=0, B=1 e N=6 il programma calcola 0 1 1 2 3 5. Ogni numero della successione va messo in una cella di memoria distinta. La serie
ottenuta va memorizzata in celle di memoria consecutive.
Descrivere chiaramente TUTTE le assunzioni fatte per chiarire aspetti non specificati o che si ritengono ambigui.

**13)** Scrivere un programma che calcoli il numero di occorrenze di un valore dato A all'interno di una stringa numerica di N cifre e sostituisca tutte queste occorrenze con un altro valore dato B. Si assuma di avere in celle di memoria sia i parametri del programma (A,B,N) sia la stringa, quest'ultima in celle di memoria consecutive. Il numero di occorrenze va restituito in una cella di memoria.

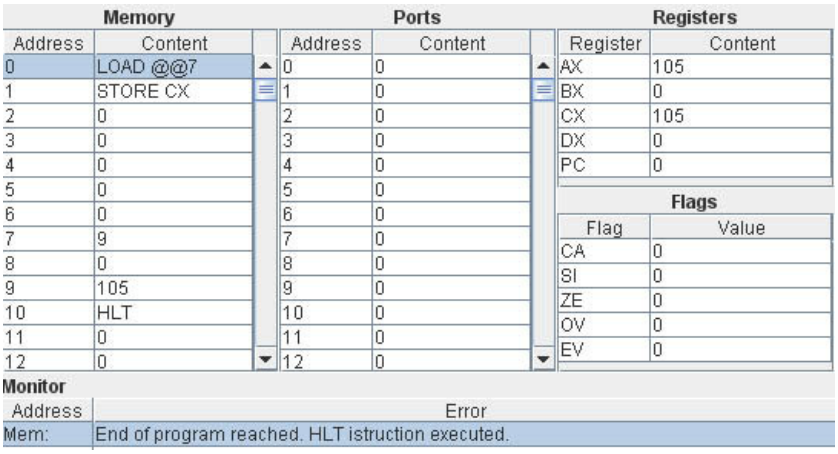
Esempio:

| Cella | Valore prima dell'esecuzione | Valore dopo l'esecuzione |
|---|---|---|
| 1 (A) | 5 | 5 |
| 2 (B) | 7 | 7 |
| 3 (N) | 5 | 5 |
| 4 | 3 | 3 |
| 5 | 5 | 7 |
| 6 | 4 | 4 |
| 7 | 5 | 7 |
| 8 | 7 | 7 |
| 9 (risultato) | | 2 |

Descrivere chiaramente TUTTE le assunzioni fatte per chiarire aspetti non specificati o che si ritengono ambigui.

Soluzioni

**1)**LOAD @7
**2)**LOAD @@7
STORE CX
Prima dell'esecuzione memorizzare :
nella cella 7 il valore di una cella di memoria
nella cella puntata dal contenuto della cella 7 il valore che verrà effettivamente copiato nel registro CX



**Memory**

| Address | Content |
|---|---|
| 0 | LOAD @@7 |
| 1 | STORE CX |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 9 |
| 8 | 0 |
| 9 | 105 |
| 10 | HLT |
| 11 | 0 |
| 12 | 0 |

**Ports**

| Address | Content |
|---|---|
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 0 |
| 4 | 0 |
| 5 | 0 |
| 6 | 0 |
| 7 | 0 |
| 8 | 0 |
| 9 | 0 |
| 10 | 0 |
| 11 | 0 |
| 12 | 0 |

**Registers**

| Register | Content |
|---|---|
| AX | 105 |
| BX | 0 |
| CX | 105 |
| DX | 0 |
| PC | 0 |

**Flags**

| Flag | Value |
|---|---|
| CA | 0 |
| SI | 0 |
| ZE | 0 |
| OV | 0 |
| EV | 0 |

**Monitor**

| Address | Error |
|---|---|
| Mem: | End of program reached. HLT istruction executed. |

**3)** Il ciclo da riga 4 a riga 9 termina quando CX=0. Ad ogni iterazione, il contenuto di CX è incrementato, mentre quello di BX viene decrementato. Quindi, al termine del codice CX vale 50.

Sostituendo in riga 6 l'istruzione JZ con JS, il ciclo termina quando CX<0, quindi CX vale 51.

**4)** Partendo dal presupposto che le celle di memoria dopo la cella 17 siano tutte inizialmente settate al valore 0 per verificare se un numero è primo è sufficiente controllare se la cella di memoria che segue le due in cui sono salvati l' 1 e il numero stesso è zero.
Ovvero non avendo il numero altri divisori oltre 1 e sé stesso è primo.

(Supponendo di iniziare a scrivere i divisori dalla cella 27 se il numero è primo la cella 27 conterrà l'1 e la cella 28 il numero stesso, mentre le celle successive saranno pari a zero perché mai toccate dal programma iniziale).

E' quindi sufficiente modificare il programma dalla riga 17 nel seguente modo :

```
0     LOAD @4
1     STORE CX
2     LOAD @5
3     STORE DX
4     65
5     27
6     JMP 8
7     DEC CX
8     LOAD @4
9     INC BX
10    MOD BX
11    JNZ 7
12    LOAD BX
13    STORE @DX
14    INC DX
15    DEC CX
16    JNZ  8
17    LOAD @5
18    ADD 2
19    LOAD @AX
20    SUB 0
21    JZ 24
22    LOAD 1
23    JMP 24
24    HLT
```

**5)** I primi due valori sono memorizzati nei registri BX e CX e la loro somma successivamente salvata nel registro DX . Il terzo valore è contenuto nella cella di memoria 4.

Considerata la somma dei due, in DX quindi, gli sottraggo il terzo numero. Se genera un valore negativo (valore del registro SI=1) salto a riga 14 ricarico in AX la somma dei due numeri e la moltiplico per il terzo valore. Se la sottrazione ha generato un numero non negativo vuol dire che la somma dei numeri è maggiore o uguale del terzo valore e quindi posso dividere tale somma per il terzo valore.

Memorizzo il risultato nella cella di memoria 21.

| | |
|---|---|
| 0 | LOAD 10 |
| 1 | STORE BX |
| 2 | LOAD 12 |
| 3 | STORE CX |
| 4 | 75 |
| 5 | LOAD BX |
| 6 | ADD CX |
| 7 | STORE DX |
| 8 | SUB @4 |
| 9 | JS 14 |
| 10 | LOAD DX |
| 11 | DIV @4 |
| 12 | STORE @21 |
| 13 | JMP 17 |
| 14 | LOAD DX |
| 15 | MUL @4 |
| 16 | STORE @21 |
| 17 | HLT |

**6)**

| Memory | |
|---|---|
| Address | Content |
| 0 | 0 |
| 1 | 3 |
| 2 | 2 |
| 3 | LOAD @2 |
| 4 | SUB 0 |
| 5 | JZ 21 |
| 6 | 0 |
| 7 | 0 |
| 8 | STORE CX |
| 9 | STORE BX |
| 10 | LOAD @1 |
| 11 | MUL @1 |
| 12 | STORE CX |
| 13 | LOAD BX |
| 14 | SUB 1 |
| 15 | STORE BX |
| 16 | JZ 24 |
| 17 | LOAD CX |
| 18 | 0 |
| 19 | JMP 11 |
| 20 | 0 |
| 21 | LOAD 1 |
| 22 | STORE CX |
| 23 | 0 |
| 24 | HLT |

Nell’esempio  nella cella di memoria numero  1 è inserito il numero da elevare a potenza, mentre nella cella numero 2 il valore della potenza.

Nel registro CX è memorizzato il risultato.

**7)**

| Address | Content |
|---|---|
| 0 | JMP 3 |
| 1 | 4 |
| 2 | 27 |
| 3 | LOAD @1 |
| 4 | SUB 0 |
| 5 | JZ 26 |
| 6 | LOAD CX |
| 7 | INC AX |
| 8 | STORE CX |
| 9 | LOAD @2 |
| 10 | ADD CX |
| 11 | DEC AX |
| 12 | STORE DX |
| 13 | LOAD CX |
| 14 | STORE BX |
| 15 | LOAD @DX |
| 16 | MUL 10 |
| 17 | INC AX |
| 18 | STORE @DX |
| 19 | LOAD BX |
| 20 | DEC AX |
| 21 | STORE BX |
| 22 | JNZ 15 |
| 23 | LOAD CX |
| 24 | SUB @1 |
| 25 | JNZ 6 |
| 26 | HLT |
| 27 | 0 |
| 28 | 0 |
| 29 | 0 |

Nella cella di memoria numero  1 è memorizzata l'altezza del triangolo.
Nella cella 2 metto il valore della cella dalla quale vorrò iniziare a scrivere il triangolo.
Il registro CX contiene il numero della cella corrispondente all’altezza del triangolo nella quale si è arrivati a scrivere.
Il registro DX contiene la nella quale sto scrivendo.

Il registro BX contiene la “profondità” di ogni singola cella, quando è raggiunta la profondità giusta relatina alla cella in cui sto scrivendo il triangolo si può passare a scrivere nella cella di memoria successiva. (Ad esempi o se ci si trova a scrivere la terza riga del triangolo  cioè “111” quando sarà stata raggiunta profondità 3 si potrà passare alla riga successiva).

**8)**

| Address | Content |
|---|---|
| 0 | JMP 3 |
| 1 | 7 |
| 2 | 34 |
| 3 | LOAD @1 |
| 4 | SUB 0 |
| 5 | JZ 26 |
| 6 | INC @33 |
| 7 | LOAD CX |
| 8 | INC AX |
| 9 | STORE CX |
| 10 | LOAD @2 |
| 11 | ADD CX |
| 12 | DEC AX |
| 13 | STORE DX |
| 14 | LOAD CX |
| 15 | STORE BX |
| 16 | LOAD @DX |
| 17 | MUL 10 |
| 18 | INC AX |
| 19 | STORE @DX |
| 20 | LOAD BX |
| 21 | DEC AX |
| 22 | STORE BX |
| 23 | JNZ 16 |
| 24 | LOAD @DX |
| 25 | 0 |
| 26 | MUL @33 |
| 27 | STORE @DX |
| 28 | 0 |
| 29 | LOAD CX |
| 30 | SUB @1 |
| 31 | JNZ 6 |

L'idea è la seguente : ogni volta che per una riga è stata raggiunta la profondità desiderata si moltiplica il numero ottenuto per un contatore e poi (se necessario) si passa a scrivere nella cella successiva.

Il contatore è memorizzato nella cella di memoria 33.

**9)**

| Memory | |
|---|---|
| Address | Content |
| 0 | JMP 3 |
| 1 | 6 |
| 2 | 1 |
| 3 | LOAD @1 |
| 4 | STORE BX |
| 5 | SUB @2 |
| 6 | JZ 12 |
| 7 | STORE CX |
| 8 | MUL BX |
| 9 | STORE BX |
| 10 | LOAD CX |
| 11 | JMP 5 |
| 12 | HLT |
| 13 | 0 |
| 14 | 0 |

Nella cella di memoria  1 c'è il valore di cui calcolare il fattoriale.

Nel registro  BX è contenuto il risultato.

**10)**

| Memory | |
|---|---|
| Address | Content |
| 0 | LOAD @4 |
| 1 | STORE CX |
| 2 | LOAD @5 |
| 3 | STORE DX |
| 4 | 65 |
| 5 | 20 |
| 6 | JMP 8 |
| 7 | DEC CX |
| 8 | LOAD @4 |
| 9 | INC BX |
| 10 | MOD BX |
| 11 | JNZ 7 |
| 12 | LOAD BX |
| 13 | STORE @DX |
| 14 | INC DX |
| 15 | DEC CX |
| 16 | JNZ 8 |
| 17 | HLT |
| 18 | 0 |

Nella cella di memoria numero 4 è memorizzato il numero di cui si devono trovare i divisori.

Nella cella numero 5 è specificata la cella di memoria dalla quale inizieremo a scrivere i divisori trovati.

**11)**

```
0  JMP 3
1  4
2  40
3  LOAD @1
4  SUB 0
5  STORE BX
6  JZ 27
7  LOAD CX
8  INC AX
9  STORE CX
10 LOAD @2
11 ADD CX
12 DEC AX
13 STORE DX
14 LOAD CX
15 0
16 LOAD @DX
17 MUL 10
18 INC AX
19 STORE @DX
20 0
21 0
22 DEC BX
23 JNZ 16
24 DEC @1
25 LOAD @1
26 STORE BX
27 LOAD @DX
28 SUB 1
29 DIV 10
30 INC DX
31 STORE @DX
32 DEC BX
33 SUB 0
34 JNZ 27
35 HLT
```

**12)**

```
0  JMP 6
1  8
2  0
3  1
4  35
5  0
6  LOAD @2
7  STORE BX
8  STORE @@4
9  INC @4
10 DEC @1
11 LOAD @3
12 STORE CX
13 STORE @@4
14 INC @4
15 DEC @1
16 ADD BX
17 STORE DX
18 STORE @@4
19 INC @4
20 DEC @1
21 JS 34
22 0
23 LOAD CX
24 STORE BX
25 LOAD DX
26 STORE CX
27 ADD BX
28 STORE DX
29 STORE @@4
30 INC @4
31 DEC @1
32 JS 34
33 JNZ 23
34 HLT
```