Programmazione Assembly per 8088: Esercizi svolti

Marco Di Felice

13 dicembre 2006

1 Esercizio 1 (esercizio 1 del Tanenbaum, Appendice C)

TESTO. Dopo l'esecuzione dell'istruzione MOV AX, 702 qual è il valore (decimale) contenuto nei registri AH e AL?

SOLUZIONE. I registri a 8 bit AH ed AL contengono (rispettivamente) la parte più e meno significativa di AX. Dopo l'operazione di MOV:

$$AX = \underbrace{00000010}_{AH}\underbrace{10111110}_{AL} \quad (1)$$

quindi AH=2 e AL=190.

2 Esercizio 2 (esercizio 2 del Tanenbaum, Appendice C)

TESTO. Il registro CS contiene il valore 4. Qual è l'intervallo di indirizzi di memoria assoluti occupato dal segmento di codice?

SOLUZIONE. L'indirizzo iniziale di un segmento (a 20 bit) si ottiene estendendo i 16 bit del registro corrispondente concatenando 4 zeri a destra nelle posizioni meno significative. Quindi, se CS vale 4, l'indirizzo iniziale del segmento di codice vale $16 \times 4 = 64$. Ogni segmento ha lunghezza pari a 64 KB, quindi l'ultima locazione del segmento è: $64 + 64 \times 1024 - 1 = 65599$.

3 Esercizio 3 (esercizio 3 del Tanenbaum, Appendice C)

TESTO. Qual è il più grande indirizzo di memoria accedibile dall'8088?

SOLUZIONE. Il massimo valore che può essere memorizzato in un registro di segmento a 16 bit è pari a 65535 (configurazione con 16 bit di valore 1). In tal caso, l'indirizzo iniziale del segmento vale $65535 \times 16 = 1048560$, mentre l'ultima locazione del segmento si trova all'indirizzo $1048560 + 64 \times 1024 - 1 = 1114095$.[1]

[1] NOTA BENE: Tale indirizzo può essere maggiore di 2^{20}=1MB.

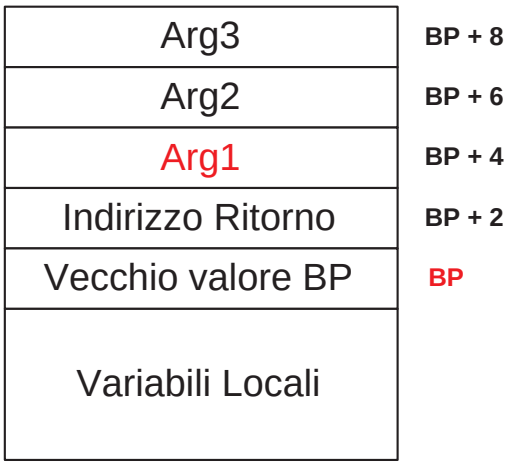

Figura 1: Il segmento di stack

4 Esercizio 4 (esercizio 3 del Tanenbaum, Appendice C)

TESTO. Sia CS=40, DS=8000 e PC=20.

1. Qual è l'indirizzo assoluto dell'istruzione successiva
2. Se viene eseguita MOV AX, (2), quale parola di memoria viene caricata in AX?

SOLUZIONE. L'indirizzo fisico dell'istruzione successiva di codice si ottiene sommando il valore dell'offset (specificato dal PC) a quello dell'indirizzo iniziale del segmento. L'indirizzo iniziale del segmento di codice è $CS \times 16 = 640$ (vedi esercizio 1). L' indirizzo della prossima istruzione è quindi $640 + PC = 660$.
Con un ragionamento analogo, si ottiene che l'indirizzo della parola di memoria caricata in AX è $DS \times 16 + offset = 8000 \times 16 + 2 = 128002$.

5 Esercizio 5 (esercizio 5 del Tanenbaum, Appendice C)

TESTO. Si immagini una subroutine con tre argomenti che viene chiamata con l'usuale sequenza: il chiamante impila i tre argomenti sullo stack in ordine inverso e quindi esegue l'istruzione CALL. Il chiamato salva il vecchio BP e inizializza il nuovo BP in modo che punti al suo vecchio valore. Quindi decrementa il puntatore allo stack in modo da allocare spazio sufficiente per le variabili locali. Date queste convenzioni, si indichi l'istruzione necessaria per copiare il primo argomento della subroutine nel registro AX.

SOLUZIONE. Lo stato dello stack è rappresentato dalla figura 1. Il primo argomento si trova all'indirizzo BP+4 quindi l'istruzione richiesta è: MOV AX, 4(BP).

6 Esercizio 6 (esercizio 7 del Tanenbaum, Appendice C)

TESTO. Si scriva il codice assemblativo per il calcolo dell'espressione: x= a + b + 2

SOLUZIONE. Assumendo che a e b siano costanti, il codice assembly che calcola l'espressione è il seguente:

```
MOV AX,a
ADD AX,b
ADD AX,2
MOV x,AX
```

7 Esercizio 7 (esercizio 8 del Tanenbaum, Appendice C)

TESTO. Una funzione C viene invocata con l'istruzione: foobar(x,y). Si scriva il codice assemblativo che realizza la chiamata.

SOLUZIONE. La funzione chiamante deve: (a) impilare sullo stack gli argomenti in ordine inverso (dall'ultimo al primo), (b) salvare sullo stack l'indirizzo di ritorno, (c) trasferire il controllo alla funzione chiamata. Tradotto in linguaggio assembly:

```
PUSH y
PUSH X
call foobar
```

8 Esercizio 8

TESTO. Dato il seguente frammento di codice assembly:

```
1    MOV CX, 100
2    MOV AX, 0
3  CICLO:
4      CMP CX,0
5      JL FINE_CICLO
6      INC AX
7      DEC CX
8      JMP CICLO
9  FINE:
```

Indicare il contenuto del registro AX al termine del codice. Indicare il contenuto di AX se si sostituisce il comando JL con JLE nella riga 5.

SOLUZIONE. Il ciclo da riga 3 a riga 8 termina quando CX<0. Ad ogni iterazione, il contenuto di AX è incrementato, mentre quello di CX viene decrementato. Quindi, al termine del codice AX vale 101. Sostituendo in riga 5 l'istruzione JL con JLE, il ciclo termina quando CX≤0, quindi AX vale 100.

9 Esercizio 9

TESTO. Dato il seguente frammento di codice assembly:

```
.SECT .DATA
    alpha: .BYTE 100
    beta:  .WORD 500
    gamma: .BYTE 4, 5, 6, 13
```

Indicare il contenuto di AL se vengono eseguite le seguenti istruzioni:

1. MOVB AL, (alpha+3)

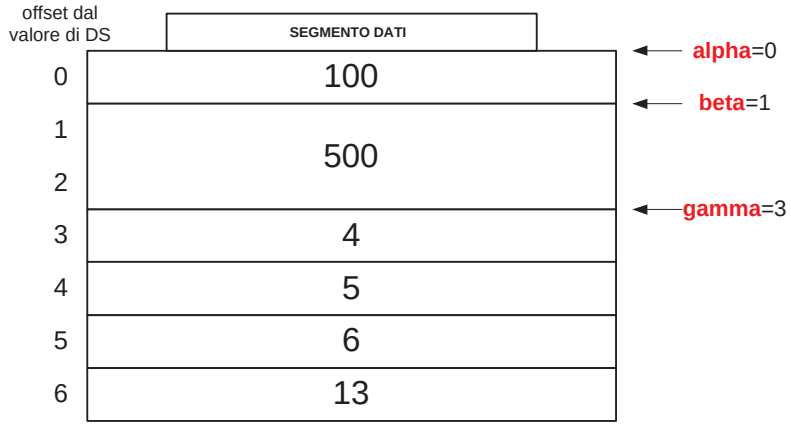

Figura 2: Contenuto del segmento dati

2. MOV BX, gamma; MOVB AL, 3(BX)
3. MOV SI, 2; MOV BX, beta; MOVB AL, 1(BX)(SI)

SOLUZIONE. La figura 2 rappresenta lo stato del segmento dati; il valore di AL è rispettivamente: 4 (quesito 1), 13 (quesito 2), 5 (quesito 3).

10 Esercizio 10

TESTO. Scrivere un programma assembly che verifica se un numero n>1 è primo.

SOLUZIONE. Una possibile implementazione in linguaggio assembly è riportata a seguire.

```
_EXIT = 1   ! Dichiarazione delle costanti
_WRITE  = 4
_STDOUT = 1

.SECT .TEXT start:
    MOV AX, (numero) ! Copia del numero in AX
    MOV BX, AX       ! Inizializza BX ad AX-1
    SUB BX, 1
1:
    DIV BX           ! Divide AX per BX
    CMP DX,0         ! Controlla il resto della divisione
    JE  2f           ! Se il resto è zero, salta a 2
    MOV DX,0
    MOV AX, (numero) ! Ripristina AX
    DEC BX           ! Decremento BX
    CMP BX,1         ! Confronto BX con 1
    JG  1b
    PUSH 6           ! Il numero è primo
    PUSH primo       ! Stampa la stringa "primo"
    PUSH _STDOUT
    PUSH _WRITE
```

```
    SYS
    JMP     3f        ! Salta alla fine

2:                    ! Il numero NON è primo
    PUSH 10           ! Stampa la stringa "non primo"
    PUSH nonprimo
    PUSH _STDOUT
    PUSH _WRITE
    SYS
    JMP 3f            ! Salta alla fine


3:                    ! Restituisce il controllo al Sistema Operativo
    PUSH _EXIT
    SYS

.SECT .DATA
numero: .WORD 4
primo: .ASCII   "Primo\n"
nonprimo:.ASCII "Non primo\n"
```