

Corso di Calcolatori Elettronici (IdT)

Facoltà di Ingegneria

Università di Firenze

Alcuni esercizi da compiti A.A. 2001/2002 (Parte 7)

1. In un'architettura 8086 si consideri il seguente spezzone di codice

```
S_SEG  SEGMENT STACK
        DW 200 DUP(?)
S_SEG  ENDS
C_SEG  SEGMENT
ASSUME CS:CS_SEG, SS:S_SEG
...
MOV AX, 5
PUSH AX
MOV BX, 4
PUSH BX
POP BX
PUSH AX
POP AX
POP BX
...
C_SEG ENDS
```

Disegnare lo stack, indicandone i punti importanti, e si descrivano le azioni causate dalle operazioni di push e di pop.

2. Dato il seguente programma

```
DATA_SEG  SEGMENT PARA
A          DW 10001b
DATA_SEG  ENDS
STACK_SEG  SEGMENT PARA  STACK
DW 10 dup (?)
STACK_SEG  ENDS
CODE_SEG   SEGMENT      PARA
MAIN       PROC          FAR
ASSUME CS:CODE_SEG,DS:DATA_SEG,SS:STACK_SEG
MOV        AX, DATA_SEG
MOV        DS, AX
PUSH       A
LABEL1:    CALL          SOTTOPROGRAMMA
LABEL2:    MOV           AH, 4Ch
INT        21h
MAIN       ENDP
SOTTOPROGRAMMA PROC      NEAR
MOV        BP, SP
MOV        AX, [BP+2]
```

```

MOV     BX, [BP]
ETICHETTA:      ADD AX, 2
RET
SOTTOPROGRAMMA   ENDP
CODE_SEG   ENDS

```

Indicare i valori di AX e BX quando l'esecuzione giunge a ETICHETTA.

- (a) AX = 10001b, BX= un indirizzo di offset
- (b) AX = 19, BX = indirizzo di LABEL1
- (c) AX= 10011b, BX = indirizzo di LABEL1
- (d) AX= 10011b, BX = indirizzo di LABEL2
- (e) Nessuna delle altre soluzioni (indicare quale)

3. Qual è il valore del registro BX al termine dell'esecuzione del seguente spezzone di codice?

```

MOV CX, 4
MOV AX, 7
UNO:   ADD AX, 2
DEC CX
CMP CX, 0
JNZ UNO
SUB AX, CX
JNZ DUE
ADD AX, 3
JMP FINE
DUE:   ADD AX, 5
FINE:  MOV BX, AX

```

- (a) 17
- (b) Nessuna delle altre risposte
- (c) 22
- (d) 16
- (e) Non è noto
- (f) 20
- (g) 18
- (h) 19

4. Quale valore contiene `val` al termine dell'esecuzione delle seguenti linee di codice assembler 8086?

```

DSEG SEGMENT
val    DB      ?
vett1  DB      11 DUP (4, 1, 3, 2)
DSEG ENDS
CSEG SEGMENT
...
MOV    AH, 0
MOV    CX, 11
MOV    val, 0
MOV    SI, 0
ciclo: MOV    AL, vett1[SI]
ADD    val, AL
INC    SI
LOOP   ciclo
SUB    val, 0
...
CSEG ENDS

```

- (a) 25
- (b) 10
- (c) 30
- (d) 110
- (e) 23
- (f) 28
- (g) Nessuna delle altre soluzioni (riportare la soluzione trovata)

5. Si indichi la lunghezza (numero di byte occupati in memoria) per ciascuna riga (dichiarazioni + istruzioni) del seguente programma assembly 8086:

	Codice	Lunghezza
Dati	SEGMENT	
varA	DW 0AB54H	
varB	DB 25H	
varC	DW 5678H	
Dati	ENDS	
Codice	SEGMENT	
...	...	—
LABEL:	ADD BX, varA	
	MOV AX, BX	
	ADD BX, 643	
...	...	—
Codice	ENDS	
Totale		

Si supponga, per semplicità che codice operativo e modi di indirizzamento degli operandi occupino i primi due byte di ciascuna istruzione. Se si ritengono necessarie altre assunzioni, si riportino nello spazio seguente:

6. Qual è il contenuto dei registri AX ed DX dopo l'esecuzione del seguente spezzone di codice

...

```

beta DW 3456H
var1 DW 7A8BH
. . .
code_sgm segment
mov AX, beta
mov DX, var1
push AX
SUB DX, 1
push DX
add AX, 4
pop AX
pop DX
. . .
code_sgm ends

```

- (a) AX =7A8AH DX = 3456H
- (b) AX =3456H DX = 7A8AH
- (c) Nessuna delle altre risposte (indicare la soluzione corretta)
- (d) AX =3456H DX = 7A8BH
- (e) AX =7A8BH DX = 3456H
- (f) AX =7A8BH DX = 7A8BH
- (g) AX =3456H DX = 3456H

7. In un'architettura 8086 si consideri il seguente spezzone di codice contenente una definizione di macro e il suo impiego all'interno di un programma.

```

prtmsg MACRO msg
    MOV     AH, 09h
    LEA    DX, msg
    INT    21h
ENDM
code SEGMENT PARA
main PROC FAR
    . . . . .
    prtmsg msg1
    prtmsg msg2
    . . . . .
main ENDP
code ENDS
datas SEGMENT PARA
msg1 DB "Hello$"
msg1 DB "Ciao$"
datas ENDS

```

Si scriva il segmento di codice ottenuto dopo l'espansione della macro. Si descrivano i principali vantaggi/svantaggi dell'uso di macro rispetto all'uso di procedure e chiamate ad interruzioni.

8. Nell'assembler 8086 si supponga che la codifica delle istruzioni e dei modi di indirizzamento occupi 2 byte. Si consideri il seguente spezzone di codice. Si indichi, per ciascuna linea di codice il numero di byte occupati, e si indichi infine lo spostamento presente nell'istruzione di salto. Si supponga (come indicato nel codice) che l'indirizzo della prima istruzione sia 0050H.

```
0050      ADD    AX,3
      QUI:  INC    CX
          ADD    AX,[BX]
          JNS   QUI
          MOV   RESULT CX
```

9. Scrivere una sequenza di istruzioni che somma 10 byte a partire dall'indirizzo IND1 e memorizza il risultato nell'indirizzo IND2. Se il valore ottenuto e' negativo, se ne calcoli il valore assoluto, memorizzandolo in IND3.
10. Nell'assembler 8086 l'istruzione ADC viene impiegata per eseguire la somma di due registri e di un eventuale riporto precedentemente generato e memorizzato nel flag CF (ADC DST, SRC è equivalente a: $(DST) \leftarrow (SRC) + (DST) + (CF)$). Impiegando le istruzioni MOV, ADD e ADC si scriva il codice necessario a sommare due numeri di 32 bit memorizzati nelle variabili DP1 e DP2. Si memorizzi il risultato nella variabile DPSUM.