Esercizi sul linguaggio Assembler

Corso di Laurea di Ing. Gestionale e di Ing. delle Telecomunicazioni

A.A. 2008-2009

1. A partire dalla locazione di nome simbolico CIFRA sono memorizzate le codifiche ASCII di quattro cifre decimali. Scrivere un programma che lasci nella parola di nome simbolico BINARY la traduzione in base due del numero intero assoluto rappresentato dalle quattro cifre decimali.
2. Scrivere un programma che prelevi 50 caratteri ASCII da un'interfaccia seriale e li immetta in memoria a partire dalla locazione di nome simbolico BETA. Dopo aver mascherato i caratteri sommando al contenuto della locazione di indirizzo (BETA + i) l'intero assoluto i, emette tramite interfaccia seriale di uscita ciascun carattere mascherato come coppia di caratteri ASCII corrispondenti ad una interpretazione esadecimale del carattere mascherato stesso.
3. Scrivere un programma ciclico che:
    a) prelevi la codifica ASCII di un carattere prelevato tramite interfaccia seriale;
    b) controlli se tale codifica è quella di una lettera miniscola o maiuscola dell'alfaveto; in caso affermativo viene emessa sull'interfaccia seriale di uscita la codifica della lettera successiva (nota: il successivo di "z" è "a");
    c) in caso negativo viene emessa sull'interfaccia seriale di uscita la codifica ASCII dell' "*"
    d) torna al punto a).
4. Due vettori di nome simbolico VETT1 e VETT2 hanno come elementi dei byte. Il numero di elementi di VETT1 è 20H. Scrivere un programma che elebori VETT1 e lasci in VETT2 delle codifiche ASCII secondo la seguente legge: i primi due elementi di VETT2 contengono le codifiche ASCII delle cifre esadecimali corrispondenti ai quattro bit più significativi del primo elemento di VETT1, e cosi' via.
5. Scrivere un programma che una volta in memoria permetta di fare quanto segue:
    a) prelevare dall'interfaccia seriale d'ingresso 100H byte e collocare in memoria i dati prelevati a partire dalla locazione DATI;
    b) esaminare il bit 3 di ogni byte al fine di contare quanti byte hanno quel bit a 0, e ogni volta che cio' accade, settarlo ad 1;
    c) scrivere nella locazione RIS il numero di byte di cui al punto b).
6. Scrivere un programma che effettui quanto segue: i registri SI, DI, BX contengono gli offset al primo elemento di tre vettori V1, V2 e V3 rispettivamente. Gli elementi dei vettori sono lunghi un byte. Il numero degli elementi di V1 è contenuto in CX. I vettori V2 e V3 hanno lo stesso numero di elementi e tale numero è contenuto nel byte successivo a quello indirizzato da CX. Il programma considera ciascun elemento di V2, conta quanti elementi di V1 hanno il valore di quell'elemento e lascia tale informazione nel corrispondente elemento di V3. Ad esempio, al termine dell'esecuzione del programma, se lo i-esimo elemento di V3 vale 5, e lo i-esimo elemento di V2 vale 100, questo significa che in V1 ci sono 5 elementi uguali a 100. Nota i vettori stanno nel segmento dati specificato da DS.
7. Scrivere un programma che faccia quanto segue:
    a) nel registro SI trova l'offset di una cella contenente l'offset di partenza di un vettore (DS specifica il segmento dati corrente);
    b) nel registro CX trova un numero che indica quanti sono gli elementi del vettore;
    c) gli elementi sono parole di 16 bit;
    d) se *elem* è un elemento del vettore, il sottoprogramma lo sostituisce con -*abs(elem*) nell'ipotesi di lavorare in complemento a due.
8. La locazione di nome ENNE contiene un numero *unsigned* N > 0. Nella doppia locazione di nome FAT viene lasciato il fattoriale di N. Nella locazione di nome ERROR viene lasciato 1 o 0 a seconda che si sia avuto o no traboccamento nel calcolo del fattoriale.
9. Riconoscitore di coppie di bit 01 nella parola contenuta nella doppia locazione di nome alpha. Il numero di coppie di bit trovate è lasciato nella locazione di nome RIS.
10. Due vettori hanno elementi da 16 bit e sono memorizzati in memoria dalle locazioni di nome simbolico VETT1 e VETT2 rispettivamente. Il programma elabora tali vettori facendo la somma elemento per elemento e lascia il risultato in un vettore che inizia dalla locazione di memoria di nome simbolico VETT3. Lo spazio di memoria richiesto per ciascuno dei tre vettori è di 100H byte. Il numero di elementi effettivi non può quindi superare 100H. Nota: eventuali traboccamenti nelle somme sono ignorati.

11. Viene prelevato un carattere tramite un'interfaccia seriale di ingresso: se il carattere è una lettera minuscola viene trasformato nella corrispondente lettera maiuscola. Di ogni carattere, eventualmente modificato, viene fatto l'eco tramite l'interfaccia seriale di uscita.
12. Scrivere un programma principale che chiamando un sottoprogramma calcoli l'integrale di una funzione. Il programma principale elabora i campioni f(0), f(D), f(2D), ... , f((n -1)D) della funzione all'interno di un vettore di byte di nome simbolico VETT. Le quantità n e D sono contenute nelle semicelle di nome simbolico LUNG e PASSO. Il risultato del calcolo deve essere messo nelle due celle (16 bit ognuna) di nome simbolico INTEGRALE, e INTEGRALE + 2. Il sottoprogramma riceve gli offset di VETT, LUNG, PASSO, in SI, DI, BX il nome del segmento dati in DS, e ritorna il risultato in AX, DX. Si supponga che il risultato del calcolo sia senza traboccamento su 32 bit.
13. Scrivere un programma che faccia quanto segue:
    a) nella cella di nome ALFA c'è l'indirizzo di un vettore;
    b) nella cella di nome BETA c'è un numero che indica quanti sono gli elementi di un vettore;
    c) gli elementi sono parole da 16 bit;
    d) se *elem* è un elemento del vettore, il programma lo sostituisce con *abs(elem)* nell'ipotesi di lavorare in complemento a 2.
14. Scrivere un programma monitor che andando in esecuzione al reset iniziale:
    a) gestisca un terminale connesso ad un'interfaccia seriale;
    b) faccia uscire per dieci volte la lettera "A";
    c) attenda che sia premuto il tasto "ritorno carrello"
    d) quindi faccia uscire il simbolo "*" all'inizio di una nuova linea.
15. Tre celle ALFA, BETA, e GAMMA, contengono rispettivamente l'offset di partenza di un primo vettore, l'offset di partenza di un secondo vettore e la lunghezza in byte dei vettori. Gli elementi dei vettori sono interi con segno ad 8 bit e sono al più 1K. Il programma elabora le coppe di elemeneni aventi lo stesso indice nei due vettori. Gli elementi sono lasciati inalterati se concordi, altrimenti, quelli di modulo minore viene cambiato di segno. Nota: ALFA, BETA, GAMMA ed i vettori stanno nel segmento dati Specificato da DS.
16. Scrivere un programma che elabora il contentuto di due celle consecutive, la prima delle quali avente nome simbolico ALFA, e lascia il risultato in una semicella BETA. L'elaborazione consiste nel calcolare quante coppie del tipo 00 o 01 o 10 sono in bit corrispondenti delle due celle consecutive.
17. Scrivere un programma che elabori il contenuto della cella di nome simbolico ALFA e metta il risultato dell'elaborazione nella cella di nome simbolico BETA; l'elaborazione consiste nel contare le triple 010 (interallacciate) contenute nella cella di nome ALFA.
18. Scrivere un programma che elabori il contenuto di una parola di nome simbolico ABC e che faccia apparire su un terminale connesso all'interfaccia seriale di uscita:
    a) la lettera "L" se il numero di coppie di bit 10 nella parola esaminata è minore di 4;
    b) la lettera "E" se il numero di coppie di bit 10 nella parola esaminata è uguale a 4;
    c) la lettera "G" se il numero di coppie di bit 10 nella parola esaminata è maggiore di 4.
19. Una matrice avente come elementi dei byte è memorizzata in memoria colonna dopo colonna. La matrice è quadrata 10H x 10H. Scrivere un programma che lasci nella semicella di nome RISU:
    a) 00H se la matrice è diagonale;
    b) FFH se la matrice non è diagonale.
20. Un programma assembler andando in esecuzione fa quanto segue: esamina il contenuto di una semicella di nome simbolico ALFA: se il contenuto di tale elemento è tale che gli 1 sono in numero pari termina, altrimenti sostituisce uno 0 con un 1 e termina.
21. Scrivere un programma che faccia apparire su un terminale video connesso ad un interfaccia di uscita la figura così descritta: rettangolo(xs=21,ys=6,xd=60,yd=20) ove ogni punto del rettangolo è rappresentato dalla lettera X (la cui codifica ASCII è 78H). L'indirizzo dell'interfaccia nello spazio di I/O sono a partire da 2000H in esadecimale, il video ha dimensioni 80x25 e il cursore è inizialmente posizionato nella posizione (1,1). Nota la codifica ASCII dello spazio è 20H, del ritorno carrello 0DH, dell'avanzamento di linea 0AH.
22. Un vettore di nome simbolico VETT ha 256 elementi, essendo ogni elemento un byte. In ogni elemento i quattro bit meno significativi rappresentano una riga della tabella di verità di una rete a 8 variabili d'ingresso e a quattro variabili di uscita (chiamiamole z3, z2, z1 e z0). Scrivere un programma che imposti il bit più significativo di ogni elemento del vettore in modo tale che la colonna così costruita sia la tabella di verità della rete caratterizzata da y=z3/z2 + z2z1z0 + z1/z0 + z2/z1.

23. Scrivere un programma che calcoli il determinante di una matrice 3x3 i cui elementi sono interi con segno, usando un sottoprogramma che calcola il determinante di una matrice 2x2.
24. Scrivere un programma che una volta tradotto in linguaggio macchina e messo in esecuzione faccia quanto segue: sul terminale video connesso ad un interfaccia seriale di uscita appare la seguente sequenza:
    000
    002
    ...
    996
    998
25. Scrivere un programma che effettui quanto segue: nella cella di nome ALFA è contenuto un numero n>0. Il programma lascia nella cella di nome RISU lo n-esimo elemento della successione di Fibonacci.

Nota.
Per tutti gli esercizi: "cella" indica una locazione di due byte (16 bit), "semicella" indica una locazione di un byte (8 bit).

Esercizio 2005-12-02

Si realizzi il programma in linguaggio Assembler 8086 che faccia quanto segue:

1. attenda che l'utente digiti una numero **`N`** compreso tra **`1`** e **`5`** attraverso una tastiera connessa all'interfaccia seriale di ingresso posta agli indirizzi simbolici RBR e RSR (i numeri da '`1`' a '`5`' hanno codifiche ASCII da 31H a 35H);
2. si depositi questo numero nella locazione ampia un byte avente nome simbolico **`N`**;
3. per **`N`** volte:
    a. si attenda che l'utente digiti **due** cifre comprese tra **`0`** e **`9`** seguite dal tasto '**`a`**';
    b. interpretando la stringa di caratteri digitati come un numero in base dieci, si memorizzi tale numero nella posizione *i-esima* del vettore collocato a partire dalla locazione **`VETT`** (**`VETT`** ha 5 elementi di un byte ciascuno).
4. si memorizzi nella locazione ampia due byte avente nome simbolico **`RISU`**, la somma degli elementi del vettore **`VETT`**;
5. si invii sull'interfaccia di uscita i caratteri ASCII necessari a stampare il valore di **`RISU`**.

Esercizio 2005-12-12

Si realizzi il programma in linguaggio **Assembler 8086** che faccia quanto segue:

1. attenda che l'utente digiti due cifre numeriche attraverso una tastiera connessa all'interfaccia seriale di ingresso posta agli indirizzi simbolici RBR e RSR (i numeri da '`0`' a '`9`' hanno codifiche ASCII da 30H a 39H);
2. memorizzi tale numero, interprato come numero in base 10, in formato binario nella locazione ampia un byte avente nome simbolico **`A`**;
3. invii sull'interfaccia di uscita i caratteri necessari per *andare a capo* e spostarsi **`5`** righe più in basso (0x0D è il codice ASCII del ritorno carrello e 0x0A è il codice ASCII dell'avanzamento a capo);
4. quindi invia all'interfaccia di uscita la sequenza:
    `00 02 04 … NN-2 NN`
    ove **`NN`** è il numero pari non maggiore del numero **`A`** (ogni numero è costituito da due caratteri ASCII ed è separato dal successivo da uno spazio).

Esercizio 2006-01-10

Considerati due vettori di nome simbolico VETT1 e VETT2 rispettivamente, aventi ciascuno 32 elementi da 8 bit, scrivere un programma in linguaggio **Assembler 8086** che faccia quanto segue:

1. memorizzi all'interno di un vettore VETT3, avente le stesse caratteristiche di VETT1 e VETT2, il risultato della somma degli elementi corrispondenti di VETT1 e VETT2;
2. stampi su un terminale connesso alla porta dello spazio di I/O avente nome simbolico TBR e registro di stato TSR, il risultato della somma espresso decimale e rappresentato tramite caratteri ASCII e seguito dai caratteri di ritorno carrello e *line feed*;
3. terminate tali operazioni ponga a 0 ogni elemento di VETT1 e di VETT2.

Esercizio 2006-03-31

Si considerino due vettori di nome simbolico VETT1 e VETT2 ognuno composto da 32 elementi da 8 bit. Scrivere un programma in linguaggio **Assembler 8086** che elabori il contenuto del vettore VETT1 facendo quanto segue: per ogni elemento di VETT1

1. se il bit più significativo dell'elemento vale 1, l'elemento di indice corrispondente nel vettore VETT2 viene posto uguale al valore dei quattro bit meno significativi dell'elemento considerato;
2. se il bit più significativo dell'elemento vale 0, il valore dell'elemento viene copiato nell'elemento di indice corrispondente del vettore VETT2
3. al termine della scansione del vettore VETT1, nella locazione ENNE deve essere presente il numero di volte in cui si e' verificato il caso 1 e nella locazione EMME deve essere presente il numero di volte in cui si e' verificato il caso 2.

Esercizio 2006-04-26

Si considerino tre vettori di nome simbolico VETT1, VETT2, e VETT3 ognuno composto da 16 elementi da 8 bit. Scrivere un programma in linguaggio **Assembler 8086** che esegua quanto segue: per ogni elemento di VETT1

1. conti quanti bit 1 sono presenti in tale elemento e scriva tale valore nell'elemento di indice corrispondente del vettore VETT2;
2. conti quante coppie 01 sono presenti in tale elemento e scriva tale valore nell'elemento di indice corrispondente del vettore VETT3.

Esercizio 2006-09-11

Si considerino tre vettori di nome simbolico VETT1, VETT2, e VETT3 ognuno composto da 32 elementi da 8 bit. Scrivere un programma in linguaggio **Assembler 8086** che esegua quanto segue:

1. per ogni elemento di VETT1 e VETT2 conta quanti bit sono a "1" e memorizza la loro somma all'interno dell'elemento corrispondente di VETT3;
2. quindi azzera il contenuto di tutte le celle di VETT1 e VETT2.

Esercizio 2006-09-25

Scrivere un programma in linguaggio **Assembler 8086** che esegua quanto segue:

1. legga dalla locazione di memoria ampia 8 bit e di nome simbolico ALFA un numero n senza segno;
2. se n <= 20 calcoli lo n-*esimo* elemento della successione di Fibonacci, e memorizzi tale valore nella locazione di memoria ampia 16 bit di nome simbolico RISU;
3. altrimenti, memorizzi il valore (2^{16}-1) nella locazione di nome simbolico RISU.

Esercizio 2006-12-01

Utilizzando il linguaggio Assembler 8086 e l'insieme di *interrupt* a disposizione sui processori compatibili x86, si realizzi il programma che faccia quanto segue:

1. stampa sullo schermo 10 volte il carattere '*' nelle posizioni di coordinate definite nel vettore VETT;
2. attende che l'utente prema uno dei seguenti tasti: '`a`', '`d`', '`s`', '`w`', e '`z`': se il tasto premuto corrisponde al carattere '`z`' il programma termina;
3. a seconda del tasto premuto sposta il cursore di una posizione rispetto a quella attuale in direzione:
   - tasto '`a`': sinistra,
   - tasto '`d`': destra,
   - tasto '`s`': basso,
   - tasto '`w`': alto,
4. controlla la nuova posizione del cursore sia posta all'interno dello schermo (dimensioni 80x25);
5. controlla che nella nuova posizione sia già presente il carattere '`*`':
   a. se è presente passa al punto 6,
   b. altrimenti scrive il carattere '*', quindi torna al punto 2;
6. stampi la stringa "Game Over!", quindi stampi su linee successive dello schermo il numero di spostamenti del cursore:
   a. verso sinistra,
   b. verso destra,
   c. verso il basso,

   d. verso l'altro;
7. invii sull'interfaccia di uscita, disponibile agli indirizzi RSR e RBR, i caratteri ASCII necessari a stampare la somma totale di tali spostamenti.

Esercizio 2006-12-11

Utilizzando il linguaggio Assembler 8086 e l'insieme di *interrupt* a disposizione sui processori compatibili x86, si realizzi il programma che faccia quanto segue:

1. stampa carattere '`$`' nella posizione di coordinate definite nei byte aventi nome simbolico STARTCOL e STARTROW;
2. attende che l'utente prema il tasto '`b`' (senza farne l'eco);
3. cancella il carattere "`$`" dalla posizione precedente e lo stampa nella una posizione più a sinistra nello schermo;
4. controlla se il carattere "`$`" appena spostato ha raggiunto il bordo sinistro dello schermo (colonna 0), in tal caso posiziona il carattere nella colonna 79 mantenendo la stessa riga e continua dal punto 3;
5. ad ogni spostamento del carattere "`$`" stampa il numero di colonna della posizione corrente nell'angolo in alto a sinistra dello schermo (si tenga presente che questo valore varia tra 0 e 79 e pertanto sono necessari due caratteri ASCII).

Esercizio 2007-01-12

Utilizzando il linguaggio Assembler 8086 e l'insieme di *interrupt* a disposizione sui processori compatibili x86, si realizzi il programma che faccia quanto segue:

1. attenda che l'utente digiti un numero compreso tra 3 e 9.
2. memorizzi il numero digitato nella cella ampia un byte di nome simbolico N;
3. calcoli il valore della successione di Fibonacci per il numero N, memorizzando tale valore nella cella di nome R;
4. stampi su una nuova riga il valore decimale (codificato da cifre ASCII) del numero R;
5. invii le cifre ASCII del risultato appena stampato anche all'interfaccia parallela di uscita collegata agli indirizzi simbolici TSR e TBR.

Nota. La successione di Fibonacci è una sequenza di numeri interi naturali definibile assegnando i valori dei due primi termini, F_0:= 0 ed F_1:= 1, e chiedendo che per ogni successivo sia $F_n := F_{n-1} + F_{n-2}$. Essa produce i seguenti valori: 1, 1, 2, 3, 5, 8, 13, 21, 34, … .

Esercizio 2007-03-29

Utilizzando il linguaggio Assembler 8086 e l'insieme di *interrupt* a disposizione sui processori compatibili x86, si realizzi il programma che faccia quanto segue:

1. conti le triple `010` presente nella locazione ampia 16 bit di nome simbolico `ALFA`;
2. memorizzi tale numero di triple nella locazione ampia un byte di nome simbolico `T`;
3. memorizzi il quadrato di `T` nella locazione ampia un byte di nome simbolico `Q`;
4. stampi a video il valore decimale (codificato da cifre ASCII) del numero `Q`.

Esercizio 2007-04-19

Utilizzando il linguaggio Assembler 8086 e l'insieme di *interrupt* a disposizione sui processori compatibili x86, si realizzi il programma che faccia quanto segue:

1. conta il numero di bit posti uguale a '`1`' della locazione ampia 8 bit di nome simbolico `ALFA`;
2. memorizzi tale numero nella locazione ampia un byte di nome simbolico `T`;
3. se tale numero è pari salta al punto 4, altrimenti sostituisce ogni bit a '`0`' contenuto in `alfa` con un bit a '`1`';
4. stampi a video il valore decimale (codificato da cifre ASCII) del numero `T`.

Esercizio 2007-09-06

Si considerino tre vettori di nome simbolico VETT1, VETT2, e VETT3 ognuno composto da 32 elementi da 8 bit. Scrivere un programma in linguaggio **Assembler 8086** che esegua quanto segue:

1. per ogni elemento di VETT1 e VETT2 esegue la somma e memorizza il risultato nel vettore VETT3;
2. dopo ogni somma stampa a schermo la rappresentazione ASCII del numeri estratti da VETT1, VETT2 e VETT3.

Esercizio 2007-09-26

Si considerino tre vettori di nome simbolico VETT1, VETT2, e VETT3 ognuno composto da 32 elementi da 8 bit. Scrivere un programma in linguaggio **Assembler 8086** che esegua quanto segue:

1. per ogni elemento di VETT1 e VETT2 esegue la somma e memorizza il risultato nel vettore VETT3;
2. dopo ogni somma stampa a schermo la rappresentazione ASCII del numeri estratti da VETT1, VETT2 e VETT3.

Esercizio 2007-12-04

Utilizzando il linguaggio Assembler 8086 e l'insieme di *interrupt* disponibili sui processori compatibili x86, si realizzi il programma che faccia quanto segue:

1. chiede all'utente di digitare da tastiera un numero compreso tra 01 e 20 (due cifre codificate ASCII) (2pt);
2. memorizza tale numero all'interno di una variabile avente nome simbolico N, ampia 1 byte (1pt);
3. dati due vettori VETT1 e VETT2 di 20 elementi ciascuno, considera i primi N elementi di ciascuno, e per ognuno di essi calcola e stampa a video ogni volta su una linea diversa:
    a. l'indice dell'elemento considerato seguito dal carattere ':' (1pt);
    b. quanti bit sono a 1 nell'elemento considerato di VETT1 (1pt),
    c. quanti bit sono a 0 nell'elemento considerato di VETT2 (1pt),
    d. quale è il valore numerico decimale codificato ASCII dell'elemento considerato di VETT1 interpretato come numero binario naturale (2pt),
    e. quale è il valore numerico decimale codificato ASCII dell'elemento considerato di VETT2 interpretato come numero binario naturale (2pt),
    f. quale è la media intera troncata tra i gli elementi corrispondenti di VETT1 e VETT2 interpretati come numeri binari naturali (1pt);
4. scambia il contenuto dell'elemento *i-esimo* di VETT1 con quello *N-i esimo* di VETT2 (2pt);
5. ripete il punto 3 per i vettori così trasformati (1pt).

Esempio di output.

```
Inserire numero: 02
1: 3 7 5 1 3
2: 2 5 3 10 6
1: 3 7 10 3 6
2: 1 5 1 5 3
```

Esercizio 2007-12-13

Utilizzando il linguaggio Assembler 8086 e l'insieme di *interrupt* disponibili sui processori compatibili x86, si realizzi il programma che faccia quanto segue:

1. chiede all'utente di digitare da tastiera un numero compreso tra 1 e 5 (una cifra codificate ASCII) (1pt);
2. memorizza tale numero all'interno di una variabile avente nome simbolico N, ampia 1 byte (1pt);
3. per N volte chiede all'utente di inserire due cifre codificate ASCII comprese tra 00 e 99 precedendo ogni richiesta dalla stringa posta su una nuova linea dello schermo avente contenuto "Inserire `il valore n. #:` ", dove al posto di # viene stampato l'indice corrente del numero da inserire (4pt);
4. gli N numeri immessi vengono memorizzati in un vettore VETT1 (1pt);
5. terminata l'immissione degli N numeri, chiama una procedura che riceve in ingresso l'indirizzo del vettore ed il numero N (3pt), e che per ogni numero inserito calcola:
    a. la rappresentazione binaria e la stampa su schermo in binario codificato ASCII (1pt);
    b. la rappresentazione binaria del complemento a 1 e la stampa su schermo in binario codificato ASCII (1pt),
    c. la rappresentazione binaria del complemento a 2 e la stampa su schermo in binario codificato ASCII (1pt),
6. quindi, per moltiplica per 2 ogni numero inserito e lo memorizza in un vettore VETT2 (2pt);
7. ripete il punto 5 per i vettore 2 (1pt).

Esempio di output.

```
Inserire numero: 2
Inserire il valore n.1: 12
Inserire il valore n.2: 07
00001010 11110101 11110100
00000111 11111000 11111001
00010100 11101011 11101000
00001110 11110001 11110010
```

Esercizio 2008-01-08

Utilizzando il linguaggio Assembler 8086 e l'insieme di *interrupt* disponibili sui processori compatibili x86, si realizzi il programma che faccia quanto segue:

1. chiede all'utente di digitare da tastiera un numero compreso tra `01` e `99` (due cifre codificate ASCII) (1pt);
2. memorizza tale numero all'interno di una variabile avente nome simbolico `N1`, ampia 1 byte (1pt);
3. stampa tale numero nei formati binario ed esadecimale codificati con i caratteri ASCII (2pt);
4. chiede all'utente di digitare da tastiera un altro numero compreso tra `01` e `99` (due cifre codificate ASCII) (1pt);
5. memorizza tale numero all'interno di una variabile avente nome simbolico `N2`, ampia 1 byte (1pt);
6. stampa tale numero nei formati binario ed esadecimale codificati con i caratteri ASCII (2pt);
7. chiede all'utente di digitare da tastiera il simbolo ASCII dell'operazione che intende effettuare, limitato a '`+`', '`-`', '`*`', '`/`' (1pt);
8. effettua l'operazione corrispondente memorizzando il risultato nella variabile avente nome simbolico `R`, ampia 2 byte (2pt);
9. stampa a video il risultato dell'operazione corrispondente nei formati decimale, binario ed esadecimale, tutti codificati con i caratteri ASCII (2pt);
10. quindi chiede all'utente se vuole effettuare un altro calcolo (attende la pressione del tasto '`y`'), oppure se vuole stampare nuovamente il risultato (attende la pressione del tasto '`a`'), oppure se vuole uscire (attende la pressione del tasto '`z`') (1pt).