1 – Utilizzo di variabili dichiarate nel programma C

All'interno di un blocco di istruzioni assembly inline, è possibile fare riferimento a *variabili* dichiarate nel programma C.

In questo modo è possibile riservare una zona di memoria (es. una word, un vettore di byte, …) tramite istruzioni in linguaggio C e accedervi tramite *indirizzamento diretto* mediante istruzioni assembler.

In tutti gli esercizi che seguiranno, l'input e l'output dei programmi saranno costituiti da variabili dichiarate in C; poiché la conoscenza del linguaggio C non è richiesta in questo corso, le istruzioni C necessarie per la dichiarazione delle variabili e la stampa dei risultati saranno fornite nel testo degli esercizi stessi.

Il lucido seguente mostra alcuni esempi.

2 – Utilizzo di variabili C: esempio

Creare un nuovo progetto contenente il seguente programma e verificarne il funzionamento passo a passo (con il debugger):

```
#include <stdio.h>


void main()
{

    unsigned int a=2;
    unsigned int b=556;
    unsigned int somma;

    __asm
    {
        MOV EAX,a
        ADD EAX,b
        MOV somma,EAX
    }

    printf("Il risultato e': %d\n",somma);

}
```

Dichiarazione di tre variabili double word (a,b,somma), di cui due inizializzate.

Accesso alle zone di memoria corrispondenti alle variabili tramite indirizzamento diretto

Stampa a video del valore della variabile somma

- *unsigned int* è il tipo di variabile C corrispondente alla double word
- È possibile dare un valore iniziale alle variabili, oppure lasciarle non inizializzate.

3 – Tipi di variabili

Alcuni esempi di dichiarazioni di variabili numeriche in C che saranno utilizzate negli esercizi:

- Un BYTE con segno: *char pluto;*
- Un BYTE senza segno: *unsigned char pluto;*
- Una WORD con segno: *short int pluto;*
- Una WORD senza segno: *unsigned short int pluto;*
- Una DOUBLE WORD con segno: *int pluto;*
- Una DOUBLE WORD senza segno: *unsigned int pluto;*
- Un vettore di 20 BYTE con segno: *char vettore[20];*
- Un vettore di 15 DOUBLE WORD senza segno: *unsigned int vettore[15];*

4 – Note preliminari sugli esercizi

- Per risolvere ciascun esercizio è necessario fare riferimento al corrispondente *diagramma di flusso* che illustra l'algoritmo da utilizzare
- Seguire sempre il funzionamento del programma con il debugger e verificarne il corretto comportamento nei vari casi
- Ogni esercizio deve essere risolto creando *un nuovo progetto* con il Visual C. Il nome del progetto dovrebbe essere legato al numero dell'esercizio (es. Prog001, Prog002, …); ogni progetto deve contenere un unico file C che deve essere salvato nella directory del progetto stesso (per maggior ordine, il nome del file C potrebbe essere lo stesso del progetto che lo contiene, es. Prog001.c)
- Se si desidera copiare un progetto da un computer all'altro (o salvarlo su un dischetto), eliminare prima la sottodirectory "debug" dalla directory del progetto. Infatti tale directory contiene file che possono essere ricreati mediante la compilazione e che quindi è inutile copiare.

5 – Esercizio 001

Moltiplicazione di due numeri mediante somme ripetute

Input: due WORD unsigned

Output: una DWORD unsigned

"//" permette di inserire una linea di commento nel programma C, (es. // Stampa su video)

Esempi di casi importanti da verificare:

Num1=0, Num2=5

Num1=1, Num2=5

Num1=4, Num2=0

Num1=4, Num2=1

Num1=4, Num2=3

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    unsigned short Num1 = 3;
    unsigned short Num2 = 5;
    unsigned int Prodotto;


// Blocco assembler
    __asm {



    }


// Stampa su video
    printf("Prodotto=%u\n",Prodotto);
}
```

6 – Esercizio 002

Algoritmo per l'estrazione di radice

Input: una DWORD unsigned

Output: una DWORD unsigned

Esempi di casi importanti da verificare:

Num=0

Num=1

Num=2

Num=10

Num=1089

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    unsigned int Num = 10;
    unsigned int Radice;


// Blocco assembler
    __asm {


    }


// Stampa su video
    printf("Radice=%d\n",Radice);
}
```

7 – Esercizio 003

Ricerca di un carattere in una stringa

Input: un vettore di BYTE (la stringa), una DWORD (la lunghezza della stringa), un BYTE (il carattere da cercare)

Output: una DWORD signed (la posizione del carattere, 0=primo carattere, …; -1 se il carattere non è presente)

- Per indicare un singolo carattere in C si utilizzano gli apici singoli, per le stringhe gli apici doppi
- L'operatore *sizeof* permette di impostare automaticamente la variabile Lunghezza al numero di caratteri in Stringa

Esempi di casi importanti da verificare:

Carattere in posizione zero

Carattere in ultima posizione

Carattere in posizione intermedia

Carattere non presente

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    char Stringa[] = "Questa è una stringa";
    int Lunghezza = sizeof(Stringa)-1;
    char Carattere = 'è';
    int Posizione;


// Blocco assembler
    __asm {


    }


// Stampa su video
    printf("Posizione=%d\n",Posizione);
}
```

8 – Esercizio 004

Dato un numero, dire se è una potenza di due

Input: una DWORD unsigned

Output: una DWORD unsigned (con valore 1 se il numero in input è potenza di 2, 0 altrimenti)

Per scrivere numeri esadecimali in C, utilizzare il prefisso "0x", es. 0xFF = 255

Esempi di casi importanti da verificare:

Num = 0

Num = 1

Num = 2

Num = 3

Num = 65536

Num = 65537

…

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    unsigned int Num = 0x80000001;
    unsigned int Ris;    // Risultato: 1 se è pot. di 2,
                         // 0 altrimenti


// Blocco assembler
    __asm {


    }


// Stampa su video
    printf("Ris=%d\n",Ris);
}
```

9 – Esercizio 005

Somma e media degli elementi in un vettore

Input: un vettore di DWORD, una DWORD (la lunghezza del vettore)

Output: due DWORD signed

Lung è inizializzato automaticamente alla lunghezza del vettore, tramite *sizeof*

Esempi di casi importanti da verificare:

Vettore di un solo elemento

Vettore di due elementi

Vettore di numeri positivi

Vettore di numeri negativi

Vettore composto sia da numeri positivi che negativi

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    int Vettore[] = {1,2,3,4,5,6,7,8,9,10,1,2,3,-1,-2,-3};
    unsigned int Lung = sizeof(Vettore)/sizeof(int);
    int Somma;
    int Media;


// Blocco assembler
    __asm {


    }


// Stampa su video
    printf("Somma=%d\n",Somma);
    printf("Media=%d\n",Media);
}
```

10 – Esercizio 006

Calcolare la parità di un vettore di byte

Input: un vettore di BYTE, una DWORD (la lunghezza del vettore)

Output: una DWORD unsigned (0 se la parità del vettore è pari, 1 se dispari)

Lung è inizializzato automaticamente alla lunghezza del vettore, tramite *sizeof*

Esempi di casi importanti da verificare:

Vettore composto solo da zeri

Vettore con un solo bit a 1

Vettore con tutti i bit a 1

…

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    unsigned char Vettore[] = {3,7,3,7,5};
    unsigned int Lung = sizeof(Vettore)/sizeof(Vettore[0]);
    int Ris; // 0: parità pari, 1: parità dispari


// Blocco assembler
    __asm {


    }


// Stampa su video
    printf("Ris=%d\n",Ris);
}
```

11 – Esercizio 007

Ordinamento "ingenuo" di un vettore

Input: un vettore di DWORD signed, una DWORD (la lunghezza del vettore)

Output: il vettore stesso ordinato

La stampa a video del vettore viene fatta attraverso un ciclo in linguaggio C che stampa i singoli elementi

Esempi di casi importanti da verificare:

Vettore già ordinato

Vettore in ordine inverso (10,7,5,4,2,…)

Vettore di numeri positivi

Vettore di numeri negativi

Vettore di numeri positivi e negativi

…

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    int Vettore[] = {3,7,3,7,5,1,4,-3,-7,-9,2,6};
    unsigned int Lung = sizeof(Vettore)/sizeof(Vettore[0]);


// Blocco assembler
    __asm {


    }


// Stampa su video
    {
        unsigned int i;
        for (i=0;i<Lung;i++) {
            printf("Vettore[%2d] = %d\n",i,Vettore[i]);
        }
    }
}
```

12 – Esercizio 008

Conversione di un numero in esadecimale (stringa)

Input: una DWORD unsigned

Output: un vettore di 8 BYTE (la stringa corrispondente al numero esadecimale)

La stampa a video del vettore viene fatta attraverso un ciclo in linguaggio C che stampa gli 8 byte (caratteri)

Esempi di casi importanti da verificare:

Num = 0

Num = 1

Num = 256

Num = 255

Num = 439041615

Num = 11259375

…

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    unsigned int Num = 1254154;
    char Ris[8];


// Blocco assembler
    __asm {


    }


// Stampa su video
    {
        unsigned int i;
        for (i=0;i<8;i++) {
            printf("%c",Ris[i]);
        }
        printf("\n");
    }
}
```

13 – Esercizio 009

Determinare se un numero è primo

Input: una DWORD unsigned

Output: una DWORD (1 se il numero è primo, 0 altrimenti)

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    unsigned int Num = 2;
    int Ris;    // Risultato: 1 se Num è primo, 0 altrimenti


// Blocco assembler
    __asm {


    }


// Stampa su video
    printf("Ris=%d\n",Ris);
}
```

Esempi di casi importanti da verificare:

Num = 0

Num = 1

Num = 2

Num = 3

Vari numeri primi (5, 7, 31, 100003, …)

Vari numeri non primi

14 – Esercizio 010 (Facoltativo)

Distanza euclidea fra due vettori floating point (single precision)

Input: due vettori di numeri floating point, una DWORD (la lunghezza dei vettori)

Output: un numero floating point

- *float* è il tipo di variabile C per i numeri floating point in precisione singola
- Lung è inizializzato automaticamente alla lunghezza del vettore Vet1 (Vet2 deve avere la stessa lunghezza)

Esempi di casi importanti da verificare:

Vettori di un solo elemento

Vettori con numeri vari (positivi e negativi)

…

Scheletro da utilizzare per il programma:

```
#include <stdio.h>


void main()
{
// Variabili
    float Vet1[] = {-2,4,3,4};
    float Vet2[] = {1,8,7,9};
    int Lung = sizeof(Vet1)/sizeof(Vet1[0]);
    float Ris;


// Blocco assembler
    __asm {


    }


// Stampa su video
    printf("Ris=%f\n",Ris);
}
```