A. Veneziani - Analisi listato con assembly in line mixato a C (C++ Builder 6)
Individua se un numero e' primo e tutti i numeri primi da 1 a n

Il problema

Il listato vuole mostrare un caso di utilizzo dell'assembly in linea dei prodotti Borland, vale a dire assembly combinato con il normale codice C / C++, e come sia possibile sfruttare tale tecnologia, ad esempio, per scrivere del codice che effettui il controllo se un numero intero dato sia primo oppure no, o, come nel secondo codice considerato ricavare tutti i primi dal numero n inserito fino a 1.

Quella utilizzata è una modalità diversa da quella riportata dal libro di testo, nel quale l'assembly presentato è quello classico funzionante su una macchina 8088 / 8086 sotto DOS, ipoteticamente dotata di hardware tradizionale per l'epoca. In realtà in questa nuova visione della macchina (in realtà sempre la stessa visto che il vostro PC non cambia), più attinente al funzionamento sotto Windows e comunque in cui vengono a scomparire alcuni elementi (ad esempio i registri segmento), mentre altri vengono introdotti (un numero maggiore di flag ad esempio), oppure modificati (i registri sono tutti a 32 bit in questo nuovo "modello" ed essi hanno una E anteposta al nome proprio per indicare questa loro nuova caratteristica).

In realtà la differenza più grossa è che nell'assembly mixato la sintassi è assai diversa da quella dell'assembly tradizionale a se stante. Innanzitutto le zone dati qui sono usualmente le stesse variabili C / C++, che si possono interfacciare con registri assembly in modo opportuno.

Una volta passati gli opportuni valori ai registri è possibile elaborare tali valori utilizzando istruzioni assembly invece che istruzioni a basso livello. Mancano del tutto le chiamate ad interrupt (il DOS rende disponibili i suoi servizi con chiamate all'interrupt 21h) e l'utilizzo diretto di servizi forniti dal sistema operativo, come usualmente si faceva prima con DOS. Per effettuare tali operazioni di I/O però ci si può appoggiare a delle tradizionali istruzioni C / C++.

Test di primalità su un numero n

Il controllo se un numero sia primo si basa sul ripetuto controllo se lo stesso sia divisibile per qualche numero da ricercare nel range (n – 1) --> 2.

Esiste perciò un ciclo in cui si effettuano ripetuti test di divisibilità, per i numeri nel range suddetto. Ovviamente se n risulta in qualche modo divisibile si conclude che non è primo, mentre in caso contrario esso sarà comprovato essere primo. Quindi per tutti i numeri i quali non sono divisibili per nessun altro (test da (n – 1) a 2), è chiaro che esso sia un numero non primo.

Per capire come venga effettuato questo controllo analizziamo il seguente listato (assembly on line):

```
#include "stdio.h"
#include "conio.h"

int main(int argc, char* argv[])
{
     int n, ris;

     // lettura del numero la cui primalità è da controllare...
     printf("Inserisci il numero intero sotto test:  ");
```

```
     scanf("%i",&n);

     // segmento di codice assembly che determina se il numero n sia primo
     asm
     {
        MOV ECX,n     // inizializza il contatore di ciclo (da n-1 a 2)
        SUB ECX,1d
Ciclo:
        MOV EAX,n     // pone come dividendo n  (coppia EDX:EAX)
        MOV EDX,0h

        DIV ECX       // divide il dividendo per ECX
         CMP EDX,0h    // controlla se il resto, ossia se EDX:EAX e' divisibile
per ECX
        JE Non_Primo  // se è divisivbile esce dal ciclo di controllo
                      // (il numero n non è primo)

        DEC ECX       // effettua un nuovo test con il numero inferiore
        CMP ECX,2d    // se è pari a 2 esce dal ciclo di test...
        JNE Ciclo

        // se il programma è arrivato qui nessun divisore è stato trovato
        MOV ris,1d
        JMP Fine
Non_Primo:
        // nel caso in cui non sia primo, allora ris viene posta a 0
        MOV ris,0d
Fine:

    }

    // stampa il messaggio in base al risultato del codice assembly (ris 0 o 1)
     if (ris == 0)
        printf("Non e' primo \n");
     else
        printf("E' primo \n");
```

come si evince dal listato in un primo momento andiamo a leggere il numero da sottoporre a test, e questa operazione viene svolta, in questo caso tipicamente in C.

Una volta letto il dato in una variabile int, si apre una sezione assembly, indicando, come se fosse una istruzione asm { …. }, ossia l'indicazione asm seguita da una parentesi graffa aperta e poi successivamente una chiusa.

Siccome un int è un valore a 32 bit in formato numerico, è possibile trasferire senza problemi questo dato in un registro dato che entrambi hanno ampiezza 32 bit. La prima istruzione infatti sposta proprio il contenuto di n nel registro ECX. Siccome però noi dobbiamo considerare solo i valori di test da (n – 1) a 2, il valore del contatore viene prima decrementato di 1 (qui si è usata la SUB ma si poteva utilizzare anche la DEC in questo caso). Ci si predispone poi a caricare il dividendo usando i registri EDX e EAX, utilizzati assieme (64 bit). Il dividendo deve essere invece presente in ECX. In esso abbiamo messo il nostro valore "a scalare", che via via diminuisce testando la divisibilità per tutti i valori da (n – 1) a 2.

Dopo che la divisione viene eseguita il quoziente va in EAX ed il resto in EDX. E' proprio questo valore che verrà utilizzato per il controllo della divisibilità. Se esso è zero anche per un solo test il numero n risulta divisibile, ossia il numero non è primo e quindi un opportuna istruzione di salto induce lo stesso a saltare

fuori dal ciclo di controllo e porre un valore indicativo in una apposita variabile (Ris) a 0, viceversa, se per tutti i test esso non risulta mai divisibile, allora è primo, e quindi alla uscita (naturale) dal ciclo, si pone il valore di quella stessa variabile ad 1.
In base a questo valore si determina se il numero inserito è primo oppure non lo è. Questo valore è raccolto in questo esempio in una opportuna variabile C / C++ che permette di comunicare al programma C le conclusioni della parte di codice assembly. In ultimo una opportuna istruzione if(...) determina quale scritta rendere sullo schermo, tramite opportune printf(...), ognuna relativa alla conclusione a cui il programma assembly è giunto.
Il codice del programma considerato risulta quello sopra riportato (in C++ Builder 6).

Individuare tutti i numeri primi fino ad un certo n

Il secondo segmento di programma preso lo stesso valore su cui ha operato il primo (n), passa n nel registro EBX (che in questo caso fa le veci di contatore generale, ossia conteggia sui numeri dei quali si deve determinare la primalità); successivamente si passa lo stesso numero nel registro ECX, il quale è preposto a conteggiare (ciclo interno), la divisibilità del numero EBX, dal numero (EBX – 1) a 2, ossia che effettua il vero e proprio test di divisibilità. Per rendere divisore il numero EBX, esso viene spostato in EAX, mentre la parte alta del divisore viene messa ovviamente a 0. Il divisore anche in questo caso è il contenuto del registro ECX. Il resto viene raccolto, come nel caso precedente, nel registro EDX, e viene effettuato il controllo se esso sia pari a 0 o no. Se il resto risulta anche solo una volta pari a 0, il numero n sotto test risulta ovviamente non primo. In questo caso, dovendo il programma stampare tutti i numeri primi, viene semplicemente saltata l'operazione di output, e si effettua un decremento di EBX, per passare al numero successivo da analizzare. Altrimenti una opportuna istruzione printf(...) effettua la stampa del valore (intero a 32 bit) di EBX.
Il relativo codice risulta:

```
     ...
     // ricerca tutti i primi da n a 1
     printf("Primi da %i a scendere: \n", n);

     asm
     {
        // EBX - contatore che determina su quale numero va svolto il
        // test di primalità
        MOV EBX,n
Ciclo3:
        MOV ECX,EBX    // imposta il contatore ECX come il numero corrente - 1
        SUB ECX,1d
Ciclo2:
        // il dividendo viene posto a EBX, ossia il numero sotto test corrente
        MOV EAX,EBX
        MOV EDX,0h

        DIV ECX        // controllo della divisibilità di EBX per ECX
        CMP EDX,0h
        JE Non_Primo2

        // ciclo che imposta un divisore successivo
```

```
        DEC ECX
        CMP ECX,1d
        JA Ciclo2
     }
        // se nessun divisore è stato trovato allora EBX viene
        // stampato direttamente nella lista dei primi
        printf("%i \t",_EBX);

     asm
     {
Non_Primo2:
           DEC EBX              // ciclo che imposta il test per un numero
successivo...
        CMP EBX,1h
        JNE Ciclo3
    }
    ...
```

Esiste la possibilità di realizzare qualcosa del tutto equivalente anche nel classico Assembly tradizionale[1], i relativi listati sono del tutto analoghi, come logica, a quelli presentati in precedenza, essi leggono e scrivono da opportune zone di memoria, opportunamente predisposte:

Test di primalità su un numero n

```
Dati    SEGMENT
N       DW ?      ; numero di cui và controllata la primalità
Primo   DB ?      ; esito del test - 1 primo, 0 non primo
Dati    ENDS

Codice  SEGMENT

Inizio: MOV AX,Dati  ; setta il segmento dati a puntare alla corretta zona
        MOV DS,AX

        MOV CX,N     ; inizializza il contatore di ciclo (da n-1 a 2)
        DEC CX
Ciclo:
        MOV AX,N     ; pone come dividendo n  (coppia DX:AX)
        MOV DX,0h

        DIV CX       ; divide il dividendo per CX
        CMP DX,0h     ; controlla se il resto, ossia se DX:AX e' divisibile per
ECX
        JE Non_Primo  ; se è divisivbile esce dal ciclo di controllo
                      ; (il numero n non è primo)

        DEC CX       ; effettua un nuovo test con il numero inferiore
        CMP CX,2d    ; se è pari a 2 esce dal ciclo di test...
        JNE Ciclo

        ; se il programma è arrivato qui nessun divisore è stato trovato
        MOV Primo,1d
        JMP Fine
Non_Primo:
        ; nel caso in cui non sia primo, allora ris viene posta a 0
        MOV Primo,0d
Fine:
```

1 A parte le operazioni di input e output dei dati, usualmente assai più complesse da gestire in Assembly classico sotto DOS

```
        MOV AH,4Ch
        INT 21h
Codice  ENDS

END Inizio
```

ovviamente in questo listato sono presenti le sezioni del segmento codice e del segmento dati (che non hanno senso nel programma C + Assembly[2]) e l'utilizzo anche di un richiamo di una funzione del sistema operativo per permettere un ritorno del controllo al DOS.
Inoltre si noti come l'utilizzo dei registri a 32 bit (registri estesi) non ci sia più, ma si utilizzino i classici registri a 16 bit. Date le operazioni in modalità segmentata, è anche necessario all'inizio del programma indicare l'opportuna inizializzazione del registro del segmento dati DS, fatto puntare alla zona dati (Dati), pena il non regolare funzionamento del programma.
Considerazioni del tutto analoghe possono essere fatte per il programma che individua i numeri primi da (n – 1) a 2. Questo programma invece che stampare a video questi numeri li individua e li copia in una apposita zona della memoria predisposta dal programmatore (vettore di byte Primi):

```
Dati    SEGMENT
N       DW ?            ; numero da cui individuare i primi
Primi   DB 1000 DUP(?)   ; vettore dei primi individuati
Dati    ENDS


Codice  SEGMENT

        ; BX - contatore che determina su quale numero va svolto il
        ; test di primalità
Inizio:
        MOV AX,Dati
        MOV DS,AX

        MOV DI,0h

        MOV BX,N
Ciclo3:
        MOV CX,BX    ; imposta il contatore CX come il numero corrente - 1
        DEC CX
Ciclo2:
        ; il dividendo viene posto a BX, ossia il numero sotto test corrente
        MOV AX,BX
        MOV DX,0h

        DIV CX        ; controllo della divisibilità di BX per CX
        CMP DX,0h
        JE Non_Primo2

        ; ciclo che imposta un divisore successivo
        DEC CX
        CMP CX,1d
        JA Ciclo2

 ; memorizza in locazioni progressive (di Primi)
 ; i numeri che sono risultati primi
```

2 Essendo la memoria di quel sistema di programmazione flat (ossia non segmentata) ed essendo le zone dati normalmente definite come variabili C.

```
        MOV Word Ptr[Primi + DI],BX
        INC DI

Non_Primo2:
        DEC BX    ; ciclo che imposta il test per un numero successivo...
        CMP BX,1h
        JNE Ciclo3

        MOV AL,4Ch  ; uscita al sistema operativo
        INT 21h

Codice  ENDS

END  Inizio
```