LA CPU INTEL 8086

Vantaggi dei programmi Assembly

L'utilizzo del linguaggio Assembly anzichè di un linguaggio ad alto livello (tipo C o Pascal) è talvolta giustificato dalla maggiore efficienza del codice;

Infatti i programmi in Assembly sono tipicamente

- più veloci,
- più corti,
- ma più complessi

dei programmi scritti in linguaggi ad alto livello.

La maggior complessità è data dal fatto che anche le più comuni routines devono essere sintetizzate dal programmatore (talvolta per semplificare la programmazione e per aumentare la compatibilità del codice, si utilizzano librerie general purpose, ma sono ovviamentemeno efficienti).

ESEMPIO:

Come esempio si consideri un programma per stampare i numeri pari da 0 a 100:

Il programma BASIC è:

```
100    I=0
110    PRINT  I
120    I=I+2
130     IF  I 100 GOTO 110
```

Il codice Assembly generato da un compilatore BASIC è il seguente

```
        I       DW          ?
L00100:         MOV         I, 0
L00110:         MOV         AX, I
                CALL        STAMPA
L00120:         MOV         AX, I
```

```
            ADD     AX, 2
            MOV     I, AX
L00130:     MOV     AX,I
            CMP     AX,100
            JB      L00110
```

Si notano almeno due semplici modifiche, che ne migliorano notevolmente le prestazioni:

-L'uso di registri al posto di locazioni di memoria

-L'uso di particolari caratteristiche dell'Assembly

Il programma scritto direttamente in Assembly è il seguente:

```
            MOV     AX,0            ; inizializza il valore del contatore
CICLO:      CALL    STAMPA          ; stampa il valore corrente di AX
            INC     AX              ;calcola il nuovo numero pari
            INC     AX              ; a partire dal vecchio AX
            CMP     AX,100          ; se AX non ha raggiunto il valore massimo
            JB      CICLO           ;ritorna a CICLO
```

Il programma così ottenuto presenta rispetto a quello prodotto dal compilatore BASIC due vantaggi fondamentali:

è più veloce (perché utilizza i registri e non locazioni di memoria)

è composto da un numero minore di istruzioni e quindi occupa una minor e estensione di memoria.

Set di Istruzioni

L'Assembly dell'8086 rende disponibili 92 tipi di istruzioni, raggruppabili nelle seguenti classi:

- Trasferimento DATI
- Aritmetiche (Somma, sottrazione, moltiplicazione, divisione)
- Manipolazioni di Bit (logiche, traslazione, rotazione)
- Trasferimento di Controllo (CALL, RET, JMP e tutti i salti condizionati e non, LOOP)
- Manipolazioni di Interruzioni (INT , INT0, IRET)
- Controllo del Processore(modifica dei Flag, Sincronizzazione, Istruzione nulla NOP).

Modi di Indirizzamento

Il Modo di Indirizzamento di un'istruzione definisce il metodo da utilizzare per determinare dove è memorizzato un particolare dato (operando).

Gli operandi possono essere contenuti:

- in registri;
- nell'istruzione stessa;
- in memoria;
- su una porta di I/O.

I Modi di Indirizzamento possono essere raggruppati in 7 classi:

- Register; MOV AX,BX
- Immediate; MOV AX,10h
- Direct;
- Register Indirect; MOV AX, [BX]
  MOV AX, [SI]
- Base Relative; MOV AX, [BX+4]
- Direct Indexed;
- Base Indexed.

ISTRUZIONI DI TRASFERIMENTO

Nel microprocessore 8086 per spostare gli operandi in memoria si utilizza il comando MOV da Move cioè spostare muovere.
Esso ha la seguente sintassi:

MOV destinazione, sorgente

Cioè sposta il contenuto di sorgente in destinazione;

In questo caso si ha che la sorgente che è scritta a destra del comando viene spostata a sinistra del comando MOV

destinazione←sorgente

La destinazione può essere un registro (a 16 o a 8 bit) o una locazione di memoria; la sorgente può essere un registro,una locazione di memoria o un dato immediato.

Vi sono molti modi per spostare un dato, tra questi ricordiamo:

Indirizzamento immediato:

```
MOV  AX , 43FF          ; sposta il contenuto 43FFH in AX
MOV AL, 43H             ; sposta il contenuto 43H in AL
```

Indirizzamento con registro:

```
MOV BX ,CX              ; sposta il contenuto di CX in BX
MOV AH, AL              ; sposta il contenuto di AL in AH
```

Indirizzamento diretto (assoluto):

```
MOV DATO, AX        ; sposta il contenuto di AX nella variabile DATO

MOV AX,DATO         ; sposta il contenuto di DATO nel registro AX

MOV AX, ES:DATO   ;sposta in AX il contenuto della variabile DATO
                   ;allocata nel segmento ES

MOV AX, OFFSET DATO  ; sposta in AX l'offset della variabile DATO
```

Indirizzamento indiretto:

L'operando si trova ad un indirizzo il cui offset è specificato dal contenuto di un registro puntatore (BX,BP, SI o DI); il registro segmento di default è DS per BX, DI per SI mentre SS è per BP;

MOV AL,[SI]
MOV AX, [BP]
MOV [DI],AL

ISTRUZIONI ARITMETICHE:

Istruzione ADD

L'operazione di somma tra due operandi viene realizzata attraverso il comando ADD (da adder cioè sommare). La sua sintassi è:

ADD sorgente1, sorgente2

In questo caso la somma tra sorgente1 e sorgente2 viene trasferita in sorgente**1**

Sorgente1←sorgente1 + sorgente2

ADD AX, DATO ; somma il contenuto di DATO a quello di AX e lo trasferisce in AX
ADD DATO,BX ; somma il contenuto di BX e DATO e lo trasferisce in DATO

Istruzione SUB

L'operazione di sottrazione tra due operandi viene realizzata tramite l'istruzione SUB (da subtract o sottrai). La sua sintassi è:

SUB sorgente1,sorgente2

N questo caso la sottrazione tra sorgente1 e sorgente2 viene trasferita in sorgente1

Sorgente1 ← sorgente1 –sorgente2

Istruzione MUL

L'istruzione MUL esegue la moltiplicazione per sorgente del moltiplicando che si trova in AL se la moltiplicazione è ad 8 bit, mentre oppure esegue la moltiplicazione di sorgente per il moltiplicando che si trova in AX se la moltiplicazione è a 16 bit.

Il risultato del prodotto di sorgente per il moltiplicando (AL O AX) viene trasferito in AX se il moltiplicando è a 8 bit mentre viene trasferito in DX:AX se il moltiplicando è a 16 bit(Word

alta in DX e word bassa in AX). I fag C ed O vengono impostati ad 1 se la metà più significativa del risultato è diversa da zero.

MUL sorgente

Moltiplica senza segno interi a 8 bit o 16 bit con il risultato in doppia precisione a 16 o a 32 bit.

ESEMPIO:

```
MOV AL, 2    ;sposta il numero 2 in AL

MOV BL,3     ;sposta il numero 3 in BL

MUL  B3      ; AX← AL* BL ciè sposta in AX il numero dato dal prodotto di AL per BL

             ; nel nostro caso AX← 0006
```

ESEMPIO:

```
MOV AX ,10  ; sposta in AX il numero 10h

MOV BX, 2 ; sposta in BX il numero 02h

MUL BX      ;sposta in DX: AX il risultato di AX per BX nel nostro caso DX:AX← 0000:0020
```

Istruzione DIV

Per eseguire la divisione si segue la seguente sistassi:

DIV sorgente

Divisione senza segno;

il divisore a 8 o a 16 bit è in sorgente; il dividendo di precisione doppia , è in AX o nella coppia di registri DX:AX;

il quoziente va in AL (8bit) o in AX (16 bit) e il resto nei due casi, in AX o in DX.

I flag sono indefiniti e una divisione per zero genera una interruzione di tipo 0.