Caruso Davide 4° B 30/11/03

RELAZIONE DI SISTEMI

Testo del problema:

Creare un programma in Assembly che stampi su video il prodotto , in forma esadecimale , di due numeri inseriti da tastiera in forma decimale. Visualizzare il prodotto tramite impiego di una procedura.

Risposta:

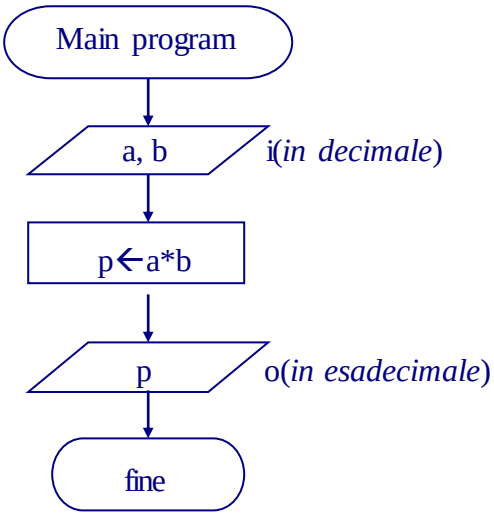
Individuazione dei dati di input e di output:

Dati di input: **a**(byte) , **b** (byte)
Dati di output: **p**(word).

Ricerca del algoritmo:

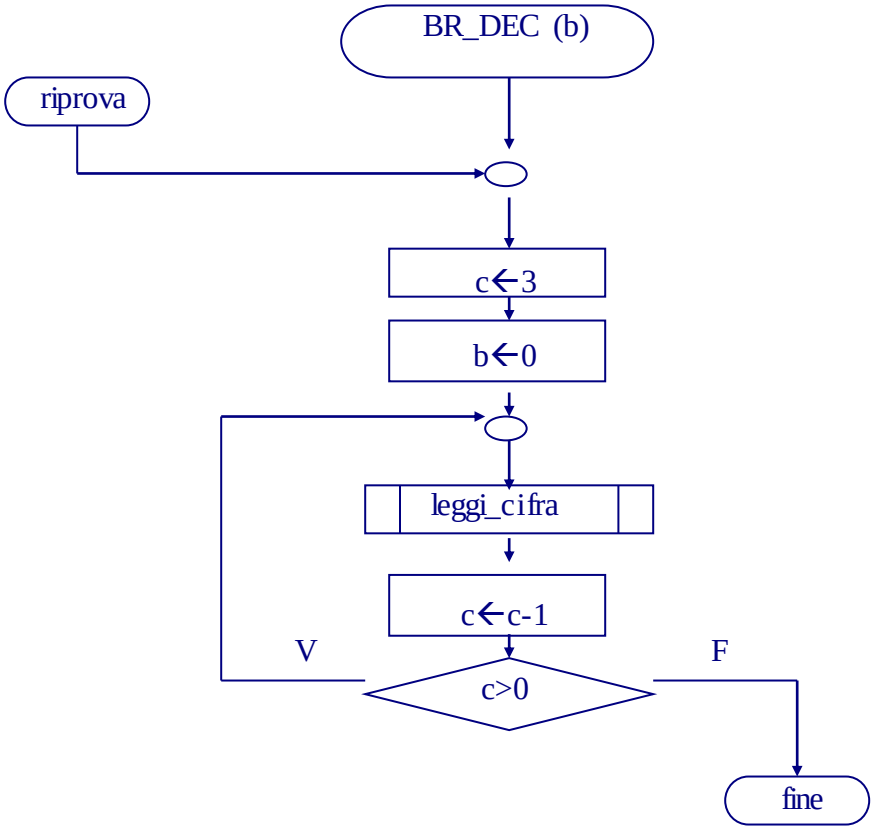
Attraverso l'utilizzo del modello di sviluppo Top-Down , ho suddiviso il problema proposto(main program) nel modo espresso dal seguente diagramma di flusso:

Diagramma di flusso



Per semplificare il problema proposto , ho deciso di considerare **a** e **b** dei numeri naturali , ognuno dalla dimensione di un byte .Di conseguenza **,** anche **p** verrà considerato un numero naturale , ma avrà la dimensione di una word , in modo che l'utente possa effettuare il prodotto per qualsiasi valore di **a** e **b**, senza ricevere alcun messaggio di overflow.
Per l'inserimento di **a** e **b** verrà utilizzata la macro BR_DEC , che avrà la funzione di leggere un byte da tastiera , in forma decimale , passato come parametro.
Questa macro , insieme alle altre di volta in volta elencate , durante la fase di implementazione , verrà salvata , in un file di testo che chiamerò STDLIB.lib , in modo da poter condividere tutte queste macro , con altri programmi , senza dover necessariamente ricopiarle , ed inoltre per non ingrandire il codice sorgente del programma principale , che sarà invece implementato nel file prodotto.asm.
Per quanto riguarda la macro BR_DEC , essa risulta basata sull'algoritmo alla pagina seguente.

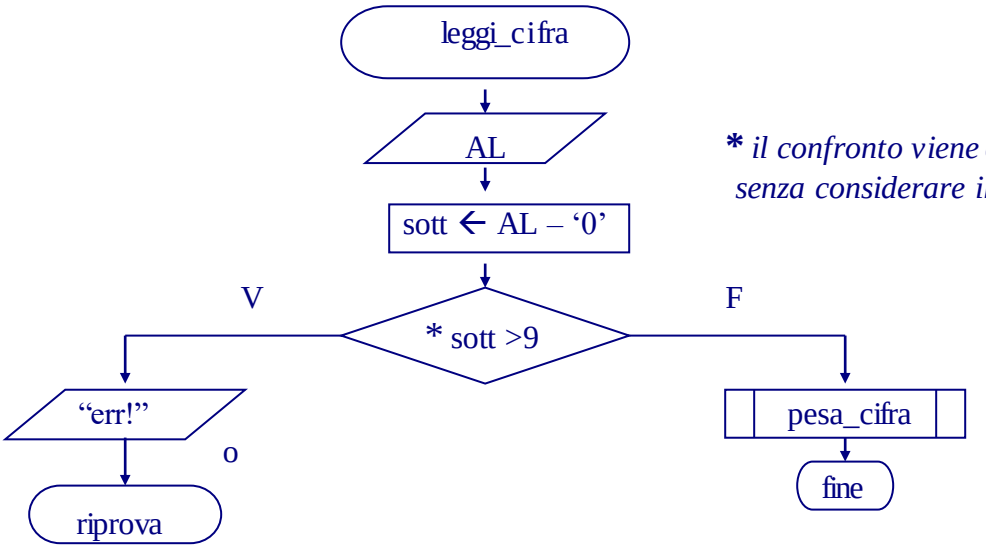
Diagramma di flusso



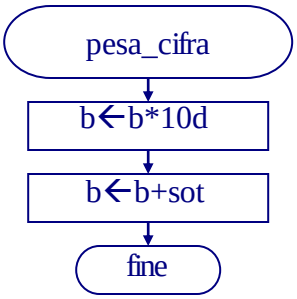
Come potete vedere , l'algoritmo risulta costituito da un ciclo for che utilizza come contatore la variabile **c** , per eseguire tre cicli , in cui viene letta la cifra di volta in volta inserita da tastiera , pesata e assegnata a **b**(byte da leggere) ,attraverso il blocco di istruzioni chiamato leggi _cifra . In questo blocco , una volta acquisito il codice ascii corrispondente al carattere del tasto premuto da tastiera , si dovrà in particolare:

- convertire tale carattere nella corrispondente cifra decimale (es: '1' diventa 1)
- individuare una possibile immissione errata (cioè verificare se l'utente , anziché digitare una cifra , digita , inavvertitamente , un carattere non numerico, oppure un numero non contenibile in un byte, cioè maggiore di 255) , e dare, in tal caso , la possibilità di reinserire il numero correttamente.
- in caso di inserimento corretto , si dovrà passare ad un altro blocco di istruzioni , che chiamerò pesa_cifra , che avrà la funzione di pesare la cifra inserita , moltiplicandola , la prima volta per 100 , la seconda 10 e in fine la terza volta per 1.

In base a quanto appena detto , gli algoritmi di entrambi i blocchi (LEGGI_CIFRA , PESA_CIFRA) risultano essere i seguenti:



***** *il confronto viene effettuato senza considerare il segno.*



Utilizzando il metodo descritto fin ora , il numero da leggere da tastiera sarà costituito sempre da tre cifre e pertanto , se , ad esempio , si vorrà inserire il numero 1 si dovrà digitare :[0][0][1] .

Traduzione in linguaggio assembly

```
;*****************************************************************
;** BR_DEC :Macro relativa all'acquisizione di un byte su video in decimale. *
;**                                                    file StdLib.lib  *
;*****************************************************************
BR_DEC MACRO B
LOCAL CICLO,INIZIO,ERR,FINE
        MOV CH,0AH
INIZIO:
        MOV DI,03H
        MOV B,00H
CICLO:
                                ;Inizio Blocco LEGGI CIFRA
        MOV AH,01H
        INT 21H
        SUB AL,30H
        CMP AL,09H
         JA  ERR                ; salto x un possibile errore causato dall'immissione di una
                                ;lettera al posto di una cifra
                                ;Fine Blocco LEGGI CIFRA
                                ;Inizio Blocco PESA CIFRA
        MOV CL,AL
        MOV AL,B
        XOR AH,AH
        MUL CH
        JB ERR                  ; salto x un possibile errore causato da un overflow in al
        ADD AL,CL
        JB ERR                  ; salto x un possibile errore causato da un overflow in al
        MOV B,AL
                                ;Fine Blocco PESA CIFRA
        DEC DI
        JA CICLO
        jmp fine
ERR:
   print_mess  MessErr          ;La stringa MessErr verrà dichiarata e inizializata con il
                                ;messaggio da stampare nel Data Segment , insieme alle altre
                                ;variabili del programma.
   JMP INIZIO
FINE:
ENDM
```

All'interno della precedente macro vi è un richiamo alla macro print_mess , utilizzata per stampare il messaggio di errore contenuto nella stringa MessErr che , come specificato nel commento , dovrà essere dichiarata in ds . La definizione della macro in questione è la seguente:

```
;*************************************************************
;* Macro relativa alla stampa di una stringa situata in ds.          *
;*                                                  file StdLib.lib  *
;*************************************************************
print_mess  macro s
   MOV DX ,OFFSET S
   MOV AH,09H
   INT 21H
ENDM
```

Questa macro utilizza la function del servizio 9 dell' interrupt 21h , definita dal sistema operativo Dos , avente la funzione di stampare su video i codici ascii contenuti all'interno della stinga situata in ds:dx e terminata dal carattere '$' . Per stampare la stringa **s** occorrerà , quindi , assegnare a dx l'indirizzo di offset di s , senza dover cambiare ds , perché questo punterà , per default in qualsiasi programma , al data segment , dove sarà situata la stringa .

Adesso inizierò a descrivere il codice sorgente del programma principale , costituito , riassumendo , dalla lettura di **a** e **b** da tastiera (attraverso la macro BR_DEC precedentemente descrittà) , dal blocco di istruzioni che calcolerà il prodotto **a** * **b** nella word **p** (che chiamerò in seguito CALCOLO_PRODOTTO) , e dalla stampa di **p** tramite la function HW_PRINT.
Per quanto riguarda il blocco CALCOLO_PRODOTTO , questo utilizzerà l'istruzione: mul (sorg) , che , come sappiamo , moltiplica , se sorg è un byte , il contenuto di AL per sorg , e mette il risultato in AX.
Invece , per quanto riguarda la function HW_PRINT , questa risulta costituita dal doppio richiamo della macro HB_PRINT , che ha la funzione di stampare su video, in esadecimale , un byte passato come parametro .Essendo possibile scindere un numero esadecimale contenuto in una word in due byte (parta alta e parte bassa) , al momento del richiamo a questa macro verrà passata la prima volta p[1] (parte alta di p) e la seconda p[0] (parte bassa di p).

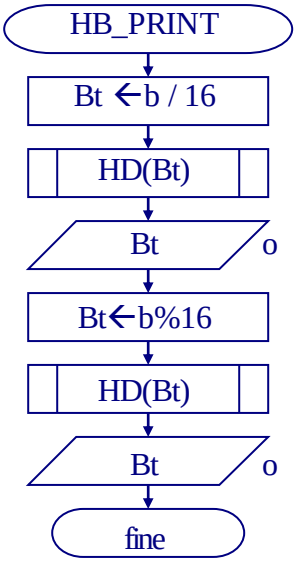
```
;********************
;** file prodotto.asm  **
;********************
include  c:\tasm\stdlib.lib

Dati segment
        a db (?)
        b db (?)
        P dw (?)
                                ;dichiarazione delle stringhe contenenti i messaggi di output su video
      messaggio1  DB "Inserisci  a:  $"
      messaggio2  DB 0ah,0dh,"Inserisci  b:  $"
                                ;i codici ASCII 0Ah e 0Dh hanno rispettivamente il compito di
                                ; cambiare riga e azzerare la  posizione del cursore
      messaggio3  DB 0ah,0dh,"a  * b =  $"     ;
      messaggio4  DB 0ah,0dh,"Premere  un tasto per uscire$"
                               ;dichiarazione della stringa MessErr utilizzata dalla macro BR_DEC ,
                               ; contenente  un messaggio di errore , che verrà stampato sul video nel
                               ;caso in cui si verifica un overflow durante l'esecuzione della macro
                               ;o se viene inserito un carattere
   MessErr DB "    Errore!",0ah,0dh,"Inserisci  un numero  inferiore  a 256: $"
Dati ends
```

```
Sistema  segment  stack 'stack'
        dw 100 dup (?)
        top label  word
Sistema  ends
Codice segment
        Assume  CS: codice, SS:sistema,  DS:dati,  ES:dati
inizio:                                 ; inizializzazione  dei registri  di segmento
        mov ax,Sistema
        mov ss,ax
        lea ax,top
        mov ax,dati
        mov ds,ax
        mov es,ax
                                        ;Inizio  del programma
        JMP START_PROGRAM
;Procedure
        HW_PRINT   PROC
             HB_PRINT  BH
             HB_PRINT  BL
             RET
        HW_PRINT ENDP
;fine  dello  spazio  riservato  alle  procedure
START_PROGRAM:
        print_mess  messaggio1      ;stampa del primo messaggio
        BR_DEC  A                   ;lettura del byte a in decimale
        print_mess  messaggio2      ;stampa del secondo messaggio
        BR_DEC  B                   ;lettura del byte b in decimale
         ;CALCOLO PRODOTTO
        MOV AL,B
        MUL  A
        MOV byte  ptr P[1],Al
        MOV byte  ptr P[0],ah
        ; Passaggio dei parametri e chiamata alla procedura che;stamperà la word p
         MOV BL, byte ptr P[1]
        MOV BH, byte ptr P[0]
        CALL  HW_PRINT
        ;blocco del programma fin quando non viene premuto un tasto
        print_mess  messaggio4
        MOV AH,01H
        INT 21H
        ;Ritorno al sistema operativo
        mov ax,4c00h
        int  21h
Codice ends
end inizio
```

Per terminare il programma mi rimane , soltanto , di definire la macro HB_PRINT , che sarà data dalla codifica dell'algoritmo alla pagina seguente , con un' unica considerazione sull'argomento da fare , che consiste nello specificare l'utilizzo delle istruzioni : *shift* e *and* , in sostituzione della divisione per sedici , dato che queste istruzioni risultano essere più efficienti dell'istruzione *div* .

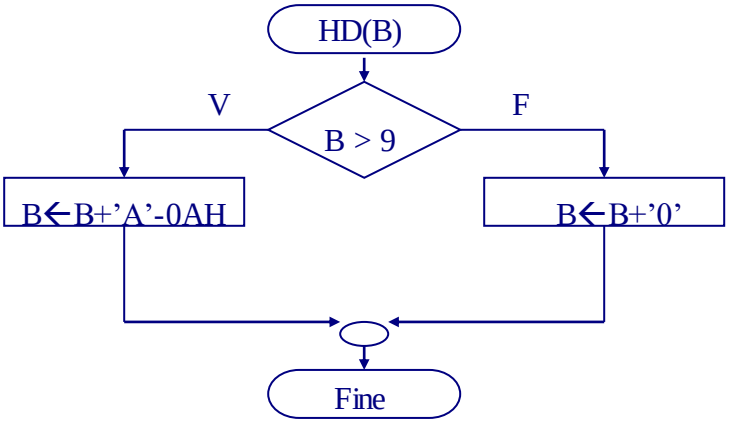
Diagramma di flusso



Traduzione in linguaggio assembly

```
;*****************************************************************
;** HB_PRINT Macro relativa alla stampa di un byte su video in   *
;** esadecimale.                                  file StdLib.lib *
;*****************************************************************
HB_PRINT MACRO B
        MOV DL,B
        MOV CL,04H
        SHR DL,CL
        HD DL
        MOV AH,02H
        INT 21H
        MOV DL,B
        AND DL,0FH
        HD DL
        INT 21H
ENDM
```

Nella precedente macro , le singole cifre di B, ottenute : sciftando di quattro posti verso destra B , per ricavare la cifra più significativa , e attraverso l'istruzione :and B,0FH , per eliminare la parte alta di B e estrarre la cifra meno significativa , al momento della loro conversione nei corrispettivi caratteri ascii , è stata utilizzata la sub-macro HD , il cui algoritmo risulta essere:



Che tradotto in linguaggio assembly diviene:

```
;*************************************************************
;** Macro x la conversione da cifra a caratere ASCII        file StdLib.lib  **
;*************************************************************

HD MACRO B
LOCAL VERO_IF,FINE_IF
        CMP B,09H
        JA VERO_IF
        ADD B,'0'
        JMP FINE_IF
VERO_IF:
        ADD B,'A'-0AH
FINE_IF:
ENDM
```

Commento:Dopo aver convertito il file sorgente(prodotto.asm) nel file eseguibile , attraverso le operazioni di compilazione e di linker , si potrà vedere il risultato finale dell'esecuzione del programma , che risulterà essere simile all'esempio riportato in basso: