```
1
2
3
4   TITLE mult: per esercitazione Calcolatori 6 & 13/12/2016
5
6   comment *
7         Il programma mostra tre diversi modi per moltiplicare due interi non negativi
8   espressi su 8 bit in assembler 8086. Il primo modo utilizza l'istruzione di macchina
9   'mul', il secondo fa uso di somme a 2x8=16 bit e shifta sia il moltiplicando esteso
10  (a sinistra)che il moltiplicatore (a destra), il terzo fa uso di somme a 8 bit e shifta
11  (a destra) il solo moltiplicatore. La moltiplicazione con il secondo e terzo modo
12  riprende gli algoritmi gia' implementati in hardware sotto forma di macchine
13  sequenziali dedicate nella prima parte del corso: vederne una nuova implementazione
14  software ha il semplice scopo didattico di mostrare alcune istruzioni, direttive e
15  strutture in linguaggio assembler, e anche di mostrare come una macchina universale
16  possa implementare qualsiasi task di calcolo attraverso un adeguato programma scritto
17  facendo uso del suo particolare set di istruzioni.
18        Il programma mostra, in particolare:
19     -  la definizione dei segmenti di codice, dati e stack
20     -  la definizione di costanti, variabili, stringhe ed etichette
21     -  la codifica di semplici cicli (inizializzazione, calcolo, terminazione)
22     -  l'uso di vari modi di indirizzamento
23     -  la definizione di macro e di subroutine annidate
24     -  il passaggio di parametri attraverso i registri o lo stack
25     -  la gestione dei dati nello stack (uso combinato dei registri sp e bp)
26     -  l'uso dell'algoritmo delle divisioni successive per la conversione base 2/base
27        B e la stampa a video (formato ASCII) di un intero non negativo a 16 bit
28     -  l'uso di una look-up table
29
30              data creazione: 5 dicembre 2016
31              ultima revisione: 13 dicembre 2016
32  *
33
34
35
36  ;-----------------------------------------------------------------
37  ; Definizione costanti
38  CR EQU 13                       ; carriage return
39  LF EQU 10                       ; line feed
40  DOLLAR EQU '$'
41
42
43
44  ;-----------------------------------------------------------------
45  ;    M  A  C  R  O
46  ;-----------------------------------------------------------------
47
48  display macro xxxx              ; N.B. ogni stringa deve terminare con '$'
49        push dx
50        push ax
51        mov dx,offset xxxx
52        mov ah,9
53        int 21h                   ; servizio di stampa fornito dal S.O. (DOS)
54        pop ax
55        pop dx
56  endm
57
58
59
60  ;-----------------------------------------------------------------
61  ;
62  PILA SEGMENT STACK 'STACK'      ; definizione del segmento di stack
63        DB      64 DUP('STACK')   ; lo stack e' riempito con la stringa 'stack'
64                                  ; per identificarlo meglio in fase di debug
65  PILA ENDS                       ; fine del segment di stack
66
67
68
```

```
;-----------------------------------------------------------------
;
DATI SEGMENT PUBLIC 'DATA'    ; definizione del segmento dati

CRLF                db CR,LF,DOLLAR
SEP                 db ':',DOLLAR
TIMES               db ' x ',DOLLAR
EQUAL               db ' = ',DOLLAR

messaggio1          db "uso della istruzione MUL <op>:  ",DOLLAR
messaggio2          db "uso di SOMMATORE a 2n bit:  ",DOLLAR
messaggio3          db "uso di SOMMATORE a n bit:  ",DOLLAR

moltiplicando       db 174
moltiplicatore      db 239
prodotto            dw ?



; per conversione del risultato in ASCII e stampa a video
max_strlen equ 16
stringa_ax          db max_strlen dup (?),' $'
alfabeto_numerazione db "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ" ; look-up table

DATI ENDS             ; fine del segmento dati




;=================================================================

CSEG1 SEGMENT PUBLIC 'CODE'

MAIN proc far
        ASSUME CS:CSEG1,DS:DATI,SS:PILA,ES:NOTHING;

        MOV AX,SEG DATI
        MOV DS,AX

        ;****** 1. USO DELL'ISTRUZIONE 8086 'MUL' *****

            mov dl, moltiplicando
            mov bl, moltiplicatore

            ; stampa a video operandi
            display crlf
            display messaggio1
            xor dh,dh
            mov ax,dx
            call near ptr display_ax
            display times
            xor bh,bh
            mov ax,bx
            call near ptr display_ax
            display equal

            mov al,dl
            mul bl ; mul operando <===> ax <--- al*operando

            mov prodotto,ax

            ; stampa a video risultato
            call near ptr display_ax
            display crlf
```

```
137
138         ;****** 2. MULTIPLICAZIONE con SOMMATORE A 2n BIT *****
139
140             mov dl,moltiplicando
141             mov bl,moltiplicatore
142
143             ; stampa a video operandi
144             display crlf
145             display messaggio2
146             xor dh,dh
147             mov ax,dx
148             call near ptr display_ax
149             display times
150             xor bh,bh
151             mov ax,bx
152             call near ptr display_ax
153             display equal
154
155                             xor ax,ax
156                             mov cx,8
157             muloop2:
158                             test bl,1
159                             jz update2
160             accumulate2:    add ax,dx
161             update2:        shr bl,1
162                             shl dx,1
163                             sub cx,1
164                             cmp cx,0
165                             jnz muloop2
166
167             mov prodotto,ax
168
169             ; stampa a video risultato
170             call near ptr display_ax
171             display crlf
172
```

```
173
174          ;****** 3. MULTIPLICAZIONE con SOMMATORE A n BIT *****
175
176              mov dl,moltiplicando
177              mov al,moltiplicatore
178
179              ; stampa a video operandi
180              display crlf
181              display messaggio3
182              xor dh,dh
183              mov ax,dx
184              call near ptr display_ax
185              display times
186              xor bh,bh
187              mov ax,bx
188              call near ptr display_ax
189              display equal
190
191                                  xor ah,ah
192                                  mov cx,8
193              muloop3:
194                                  xor bh,bh
195                                  test al,1   ; check LSB di AL
196                                  jz update3
197              accumulate3:        add ah,dl
198                                  jnc update3 ; jump if NOT CARRY
199                                  mov bh,80h  ; N.B. 80h=10000000b
200              update3:
201                                  shr al,1
202                                  test ah,1   ; check LSB di AH
203                                  jz skip3
204                                  or al,80h   ; nel MSB di AL il LSB di AH
205              skip3:              shr ah,1
206                                  or ah,bh    ; nel MSB di AH il carry (MSB di BH)
207
208              check3:             sub cx,1
209                                  cmp cx,0
210                                  jnz muloop3
211
212              mov prodotto,ax
213
214              ; stampa a video risultato
215              call near ptr display_ax
216              display crlf
217
218          ;*************************************************************
219
220
221         exit:
222         MOV AH,4CH                  ; ritorno al DOS
223         INT 21H
224
225    main endp                        ; fine del programma principale
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
```

```
;================================================================================
; routine di conversione di un numero a 16 bit (il cui valore e' posto nello stack)
; nell'equivalente in una base prefissata B, espresso in ASCII e posto
; in una stringa di N caratteri, il cui indirizzo e' anch'esso inserito nello stack
; prima della chiamata. Il valore di ritorno N rimpiazza il valore immesso nello stack
; dal chiamante, che indica la base B in cui convertire il numero
;
bin2bascii proc near
        push ax                     ; salvataggio di tutti i registri usati
        push bx
        push cx
        push dx
        push si
        push di
        push bp
        pushf

        mov bp,sp
        add bp,16          ; rimuove l'effetto dei push qui sopra
        mov ax,SS:[bp+6]   ; prelievo numero da convertire (e' sepolto sotto 3 words)

        mov bx,[bp+2]      ; prelievo base numerazione
        mov cx,0           ; contatore caratteri ASCII stringa destinazione

again:  xor dx,dx
        div bx        ; in DX il resto, in AX il quoziente di [DX:AX]/[BX]
                      ; N.B. con or dx,0030h si renderebbe ASCII il contenuto di dx
        mov si,dx
        mov dl,alfabeto_numerazione[si]    ; si prende il simbolo di numerazione
        xor dh,dh                          ; dall'array (look-up table) apposito
        push dx       ; salvataggio nello stack della cifra (in ASCII) convertita
        inc cx
        cmp ax,0    ; le divisioni successive per la base terminano se quoziente zero
        je save_result
        jmp again
save_result:
        mov di,0
        mov bx,[bp+4]                ; prelievo offset della stringa destinazione
cycle:  pop dx              ; riempimento della stringa destinazione
        mov [bx][di],dl     ; con le cifre ASCII via via salvate nello stack
        inc di
        loop cycle
        mov [bp+2],di  ; salvataggio numero di caratteri nella stringa destinazione
                       ; (al posto della base di conversione, che non serve piu')

        popf                        ; ripristino di tutti i registri usati
        pop bp
        pop di
        pop si
        pop dx
        pop cx
        pop bx
        pop ax

        ret

bin2bascii endp
```

```
; ------------------------------------------------------
; routine di conversione e stampa a video formattata
; di un numero intero non negativo posto nel registro AX
;
display_ax proc near
        push ax
        push bx

        push ax                      ; in AX il numero da convertire
        mov ax,offset stringa_ax
        push ax                     ; in AX l'indirizzo della stringa destinazione
        mov ax,10
        push ax                      ; in AX la base di conversione
        call near ptr bin2bascii     ; chiamata (annidata) a subroutine
        pop bx                       ; in BX il numero di caratteri della stringa
        add sp,4

        mov stringa_ax[bx],'$'       ; '$' aggiunto per poter stampare
        display stringa_ax

        pop bx
        pop ax
        ret

display_ax endp



cseg1 ends                  ; fine del segment di codice

END MAIN                    ; il programma comincia all'indirizzo di MAIN
```