```
TITLE unzip.asm - per compito 21/09/2005

COMMENT *
Il buffer CBUFF contiene N byte di dati compressi con la seguente
codifica:

                            1 -> 'A'
                           01 -> 'B'
                          001 -> 'C'
                         0001 -> 'D'

Bisogna decomprimere il buffer un byte alla volta, ponendo i dati
decompressi in un secondo buffer (UBUFF). Il programma principale
deve richiamare una macro, dopo aver posto il byte da decomprimere
in AL. La macro scrive in un buffer temporaneo USTRING. Il main
preleva il contenuto di USTRING e lo pone in UBUFF (simulando cosi'
l'output su stampante richiesto nel compito).
*




;-----------------------------------------------------------------
STACK SEGMENT STACK 'STACK'    ; definizione del segmento di stack
      DB      64 DUP('STACK')  ; lo stack e' riempito con la stringa 'stack'
                               ; per identificarlo meglio in fase di debug
TOS   LABEL WORD               ; identifica il top of stack
STACK ENDS


;-----------------------------------------------------------------
DATA SEGMENT PUBLIC 'DATA'           ; definizione segmento dati

                                     ; stringa originale (32 caratteri, codificati con N=9 byte)
    ORIGINAL DB "ORIGINALE:    ACCACABDAACCDBACAABCBADACCBDDBAB",13,10,'$'
    N equ 9                          ; questo statement NON alloca memoria
    CBUFF DB 10010011B,00110100B,01110010B,01000101B,10011101B,00101100B,01100100B,10100010B,00101101B,13,10,'$'
    Ustring DB 8 dup(?),'$'       ; al max un byte compresso da' luogo ad 8 byte decompressi ("AAAAAAAA")
    DECOMPRESSED DB "DECOMPRESSA: ",
    UBUFF DB N*8 dup(?),'$'
    NWRITTEN DW 0                    ; numero di byte gia' scritti in UBUFF: e' inizializzato a 0
    NRESIDUAL DW 0                   ; numero di bit a 0 nella parte meno significativa del byte
                                     ; compresso: e' inizializzato a 0
    listofsymbols db 'A','B','C','D' ; look-up table utilizzata dalla macro di decompressione
DATA ENDS
```

```
;-----------------------------------------------------------------
; MACRO
                                ; il byte da decomprimere e' in AL;
                                ; la stringa decompressa va in uncstrng;
                                ; la macro pone in dx il numero di byte decompressi,
                                ; e in nres il numero di bit a 0 residui
unzip macro uncstrng,nres       ; notare: 2 argomenti, separati da virgola
   local uncompressloop,writesymbol,closeiteration  ; le etichette dentro la macro si dichiarano local,

                                                    ; nell'evenienza che la macro sia chiamata piu' di una volta
                                                    ; e l'assemblatore dia errore per ridefinizione di etichetta
      push cx                   ; si usa cx come contatore, ma cx e' gia' in uso fuori della macro,
      mov cx,8                  ; e quindi va salvato prima di "sporcarlo"
      xor di,di                 ; contatore dei byte decompressi
      mov si,nres               ; conta il numero di 0 che precedono un 1: si=0->'A',1->'B', etc.
      mov ah,10000000B          ; maschera per il test dei bit a 1
uncompressloop:
      test al,ah                ; non modifica AL, ma scrive nella PSW
      jnz writesymbol
      inc si
      jmp closeiteration
writesymbol:
      mov dl,listofsymbols[si]  ; si scrive il simbolo decompresso nella stringa di uscita
      mov uncstrng[di],dl
      inc di
      mov si,0
closeiteration:
      shr ah,1                  ; si trasla a destra di 1 la maschera
      loop uncompressloop
      mov dx,di                 ; passaggio dei risultati di conteggio: il primo attraverso
      mov nres,si               ; un registro, il secondo attraverso la memoria
      pop cx                    ; ripristino del CX
endm



display macro xxxx              ; N.B. ogni stringa deve terminare con '$'
        push dx
        push ax
        mov dx,offset xxxx
        mov ah,9
        int 21h
        pop ax
        pop dx
endm
```

```
;-----------------------------------------------------------------
CSEG SEGMENT PUBLIC 'CODE'

MAIN PROC FAR
      ASSUME cs:cseg,DS:data;

        MOV AX,DATA             ; necessari 2 trasferimenti per assegnare a DS
        MOV DS,AX               ; l'indirizzo del segmento dati in modo esplicito

        DISPLAY ORIGINAL        ; DISPLAY DELLA STRINGA ORIGINALE,
                                ; la cui versione COMPRESSA e' in CBUFF
        mov cx,N
        mov bx, offset CBUFF
        ; MOV NWRITTEN,0          (superflua, perche' NWRITTEN inizializzata nel segmento dati)
        ; mov NRESIDUAL,0         (numero iniziale di bit residui, anch'esso gia' inizializzato sopra)

UNZIPLOOP:
        mov al,[bx]             ; nota: indirizzamento indiretto di registro
        unzip ustring,nresidual ; la macro pone in dx il numero di byte decompressi
                                ; e in nresidual il numero di bit a 0 residui

        MOV di,NWRITTEN         ; ad ogni ciclo si rimette in DI il numero di caratteri gia' posti in UBUFF
        xor si,si               ; inizializzazione di si per il ciclo XFERLOOP
XFERLOOP:                       ; si ricopiano i byte decompressi da UTEMP a UBUFF
        mov al,ustring[si]      ; si riusa al, che prima di questa istruzione
                                ; contiene il byte che e' appena stato decompresso
        mov ubuff[di],al        ; N.B. mov ubuff[di],utemp[si] non e' permessa con l'8086
        inc di
        inc si
        cmp si,dx               ; controllo di fine ciclo (DX contiene il numero di caratteri da porre in UBUFF)
        jne XFERLOOP

        mov nwritten,di         ; si risalva il conteggio di caratteri gia' scritti in UBUFF
        inc bx                  ; si incrementa BX per puntare al nuovo carattere di CBUFF
        LOOP UNZIPLOOP          ; equivale a dec cx, jnz unziploop
EXIT:
        mov bx,NWRITTEN         ; si aggiunge un '$' a fine stringa, per stampare a video
        mov ubuff[bx],'$'
        DISPLAY DECOMPRESSED    ; stampa la stringa omonima + quanto scritto in UBUFF (fino al '$')

        MOV AH,4CH              ; si restituisce il controllo al DOS
        INT 21h

MAIN ENDP

CSEG ENDS

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