```
TITLE algosqrt.asm: per esercitazione 16/12/2020

comment *
      Scrittura in linguaggio assembly dell'algoritmo per il calcolo della radice quadrata di un numero N,
nella forma radice intera e residuo. L'algoritmo fa uso di sottrazioni e shift. Cfr. compito 17/6/2019.
Rielaborazione di un esercizio svolto dagli studenti Lorenzo Corsini e Cristiano Narducci.

            data creazione: 2 dicembre 2020
           ultime modifiche: 16 dicembre 2020

*

CR EQU 13                       ; carriage return
LF EQU 10                       ; line feed
DOLLAR EQU '$'

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

PILA SEGMENT STACK 'STACK'     ; definizione del segmento di stack
      DB      24 DUP('STACK')  ; lo stack e' riempito con la stringa 'stack'
                               ; per identificarlo meglio in fase di debug
PILA ENDS                      ; non necessario a prima vista in questo esercizio

DATI SEGMENT PUBLIC 'DATA'
CRLF                 db CR,LF,DOLLAR
COMMA                db ',',DOLLAR
N dw 3768
pot equ 0100000000000000B ;16384
; per conversione del risultato in ASCII e stampa a video
max_strlen equ 16
```

```
stringa_ax           db max_strlen dup (?),' $'
alfabeto_numerazione db "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"

DATI ends

CSEG1 segment public 'CODE'

MAIN proc far

ASSUME CS:CSEG1,DS:DATI,SS:PILA

mov AX, seg DATI
mov DS,AX




;INIZIALIZZAZIONE


MOV BX,N ;n in BX

; stampa a video di N
mov ax,bx
display crlf
call near ptr display_ax; si stampa N
display crlf

XOR AX,AX ; r in AX
MOV CX,pot ;s in CX

;PRIMO CICLO ADJUST while(s > n)
ADJUST:
cmp CX,BX
jle COMPUTE
shr cx,1    ;s <- s >> 2
shr cx,1
jmp ADJUST
```

```
;CICLO PRINCIPALE
COMPUTE:
cmp CX,0
je TERMINATE
mov DX,AX ; dx <- r
add DX,CX ; dx <- r+s
cmp BX,DX ; confronto tra n e r+s
jl COMPUTE_ELSE ; n < (r+s)
sub BX,DX
shr AX,1
add ax,cx
jmp COMPUTE_AFTER_ELSE

COMPUTE_ELSE: ; r >> 1
shr AX,1

COMPUTE_AFTER_ELSE: ;s <- s >> 2
shr cx,1
shr cx,1

jmp COMPUTE

TERMINATE:       ;stampa a video i risultati e ritorno al DOS
display crlf
call near ptr display_ax ; si stampa a
mov AX,BX
display crlf
call near ptr display_ax ; si stampa b

display crlf

MOV AH,4CH                ; ritorno al DOS
INT 21H

MAIN endp
```

```
bin2bascii proc near
        push ax
        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]                    ; numero da convertire (e' sepolto sotto 3 words)

        mov bx,[bp+2]
        mov cx,0
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 dall'array apposito
        xor dh,dh
        push dx
        inc cx
        cmp ax,0
        je save_result
        jmp again
save_result:
        mov di,0
        mov bx,[bp+4]                       ; offset della stringa destinazione
cycle:  pop dx
        mov [bx][di],dl
        inc di
        loop cycle
        mov [bp+2],di                       ; numero di caratteri nella stringa destinazione

        popf
        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
        mov ax,offset stringa_ax
        push ax
        mov ax,10
        push ax
        call near ptr bin2bascii
        pop bx                      ; in bx il numero di caratteri della stringa
        add sp,4

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

        pop bx
        pop ax
        ret

display_ax endp

cseg1 ends
END MAIN
```