


```

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

ADDNUMSTR PROC FAR
    ASSUME CS:CSEG,DS:DATA,SS:STACK,ES:NOTHING;

    MOV AX,DATA          ; Necessary 2 transfers to assign to DS
    MOV DS,AX            ; l'indirizzo del segmento dati in modo esplicito

    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ; 1. Conversione della stringa STR1 in complemento a due (il risultato e' posto in AX)
    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

    ; 1.1 Calcolo del modulo del numero NUM1 corrispondente a STR1 (il risultato e' posto in AL)
    ; Nota: la conversione e' realizzata come somma di potenze della base di arrivo (10)
    ;
    mov accum,0           ; Inizializzazione (qui superflua) dell'accumulatore
    mov ah,0              ; Azzeramento della parte alta di AX
    mov al,STR1[3]        ; In AL il carattere '8' (=38h)
    and al,0Fh            ; In AL il numero 8 (=08h)
    add accum,al          ; In ACCUM il numero 8=8+0
    mov al,STR1[2]        ; In AL il carattere '2' (=32h)
    and al,0Fh            ; In AL il numero 2 (=02h)
    mul dieci             ; ax <--- al*dieci. In AL (ed AX) il numero 20=2*10 (AH resta =0)
    add accum,al          ; In ACCUM il numero 28=20+8
    mov al,STR1[1]        ; In AL il carattere '1' (=31h)
    and al,0Fh            ; In AL il numero 1 (=01h)
    mul cento             ; ax <--- al*cento. In AL (ed AX) il numero 100=1*100 (AH resta =0)
    add accum,al          ; In ACCUM il numero 128=100+28
    mov al,accum           ; In AL il numero 128 (=80h)

    ; 1.2 Rappresentazione del numero con segno NUM1 in complemento a due su 16 bit (AL -> AX)
    ; con estensione di segno sul byte ah
    ;
    mov ah,STR1[0]        ; Controllo del segno di NUM1
    cmp ah,'-'
    je num1_negativo      ; Se NUM1 positivo, AL resta invariato
    mov ah,0              ; Estensione di segno per numeri positivi
    jmp num1_in_ax
num1_negativo:           ; Se NUM1 negativo, AL <--- complemento a due di AL
    not al               ; e si estende il segno su ah con tutti i bit a 1
    add al,1             ; (e' quello che accade nel caso d'esempio: AX=FF80h)
    mov ah,0Ffh
num1_in_ax:

```

```

; //////////////////////////////////////
; 1bis. conversione della stringa STR2 in complemento a due (il risultato e' posto in DX)
; //////////////////////////////////////
;
; Nota: questa conversione, prevista nell'esercizio d'esame del 27/4/2005, e' identica
; a quella vista sopra per STR1. Nel presente programma si considera invece la variabile
; di memoria NUM2, il cui valore (che e' gia' espresso in compl. a due) viene posto in dx.
; Nel caso in esempio NUM2>0, percio' DX=0039h, con 39h = 00111001b = 57d.
; Nota: l'istruzione mov dx, word ptr num2 non funzionerebbe, perche' l'operatore word ptr
; forza il byte num2 al formato di parola richiesto da dx prelevando dalla memoria il byte
; successivo a num2 e ponendolo in DH, e non facendo, come dovrebbe, l'estensione del segno.
;
mov dl,num2
controlla_dl:
mov dh,0 ; Estensione di segno di default (NUM2 positivo)
cmp dl,0
jge num2_in_dx
mov dh,0ffh ; Estensione di segno se NUM2 negativo
num2_in_dx:

; //////////////////////////////////////
; 2. somma algebrica NUM1+NUM2 (il risultato viene posto in AX)
; //////////////////////////////////////

add ax,dx ; AX <-- AX+DX ( = -128+(+057) = -71 ),
; con bypass dell'overflow, che si potrebbe verificare
; con AL <-- AL+DL. Infatti, poiche' il range di
; rappresentazione di NUM1 (e NUM2) e' [-128,+127], il
; range della somma NUM3=NUM1+NUM2 e' [-256,+254], la cui
; rappresentazione in complemento a due richiede 10 bit.
; Si noti che, fatta eccezione per -256, i restanti numeri
; del range di uscita sono rappresentabili in
; complemento a due con 9 bit; ne segue che i loro moduli sono
; rappresentabili in rappresentazione naturale su 8 bit: [0,255].

num3_in_ax:

; //////////////////////////////////////
; 3. ri-conversione del numero in complemento a due in AX nel formato stringa (il risultato posto in STR3)
; //////////////////////////////////////

; 3.1 Scrittura del segno di NUM3 nella stringa STR3 e del modulo di NUM3 in AX
;
cmp ax,0
jl ax_negativo
ax_positivo:
mov STR3[0], '+' ; Se AX e' positivo, allora AH contiene tutti zeri
jmp modulo_NUM3_in_ax ; e si puo' procedere a convertire AL
ax_negativo:
mov STR3[0], '-' ; Se AX e' negativo (e nel caso d'esempio lo e'), allora
not ax ; viene anzitutto reso positivo con ilcomplemento a due
add ax,1 ;
modulo_NUM3_in_ax:

```

```

; 3.2 Conversione del modulo di NUM3 posto in AX nella stringa STR3
;
; Nota 1: per la conversione inversa si usano le divisioni
; successive per la base 10, "ASCIIificando" i resti ottenuti,
; che corrispondono alle cifre via via piu' significative.
; Al posto della costante 30h, nella conversione cifra->carattere
; si puo' utilizzare direttamente '0', e/o al posto della or la add

div dieci
or ah,30h
mov STR3[3],ah
mov ah,0
; In AH il resto, in AL il quoziente di AX/dieci (qui: AH=1)
; Si rende ASCII l'intero positivo (qui: AH='1')
; Scrittura della cifra decimale meno significativa di AX in STR3
; Dopo la prima divisione per 10, ah si puo' porre a 0
; perche' modulo(NUM3)/10 vale al piu' 25; invece, in preparazione
; della prima divisione per 10 non si pone AH a 0 con MOV AH,0
; perche' (nell'unico caso NUM3=-256) potrebbe essere ah=1

div dieci
or ah,30h
mov STR3[2],ah
mov ah,0
; In AH il numero 7
; In AH il carattere '7'

div dieci
or ah,30h
mov STR3[1],ah
; In AH il numero 0
; In AH il carattere '0'

stampa_a_video:
display crlf
display STR1
display STR2
display uguale
display STR3
display crlf

mov ah,4ch
int 21h
; Uscita dal programma

ADDNUMSTR ENDP

CSEG ENDS

END ADDNUMSTR
; il programma comincia all'indirizzo di ADDNUMSTR

```

Esercizio #2 compito #1 esame 27/4/2005

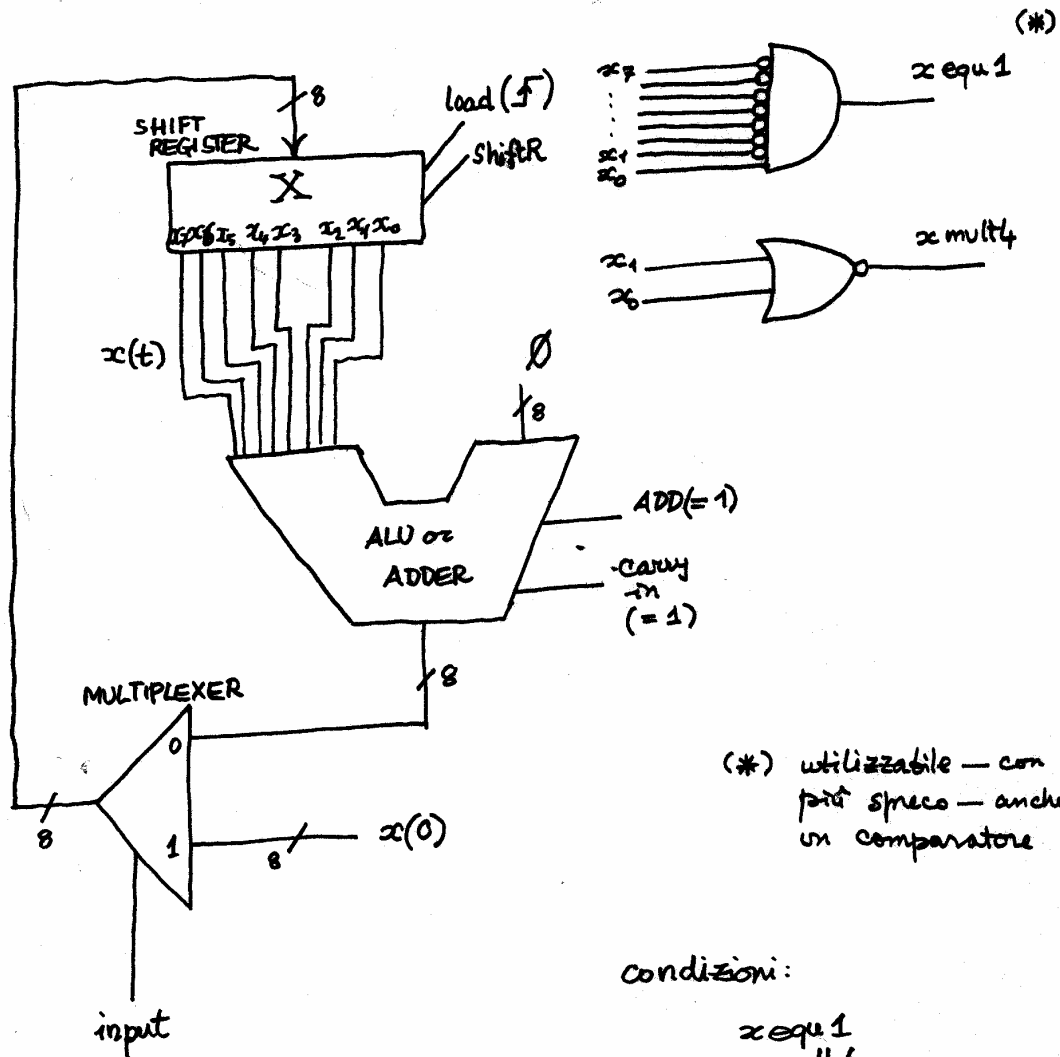
Algoritmo:

if $x=1$ stop;

if $x \% 4 = 0$ $x \leftarrow x/4$

else $x \leftarrow x+1$

Parte Operativa:



(*) utilizzabile — con
pois' spaco — anche
un comparatore

condizioni:

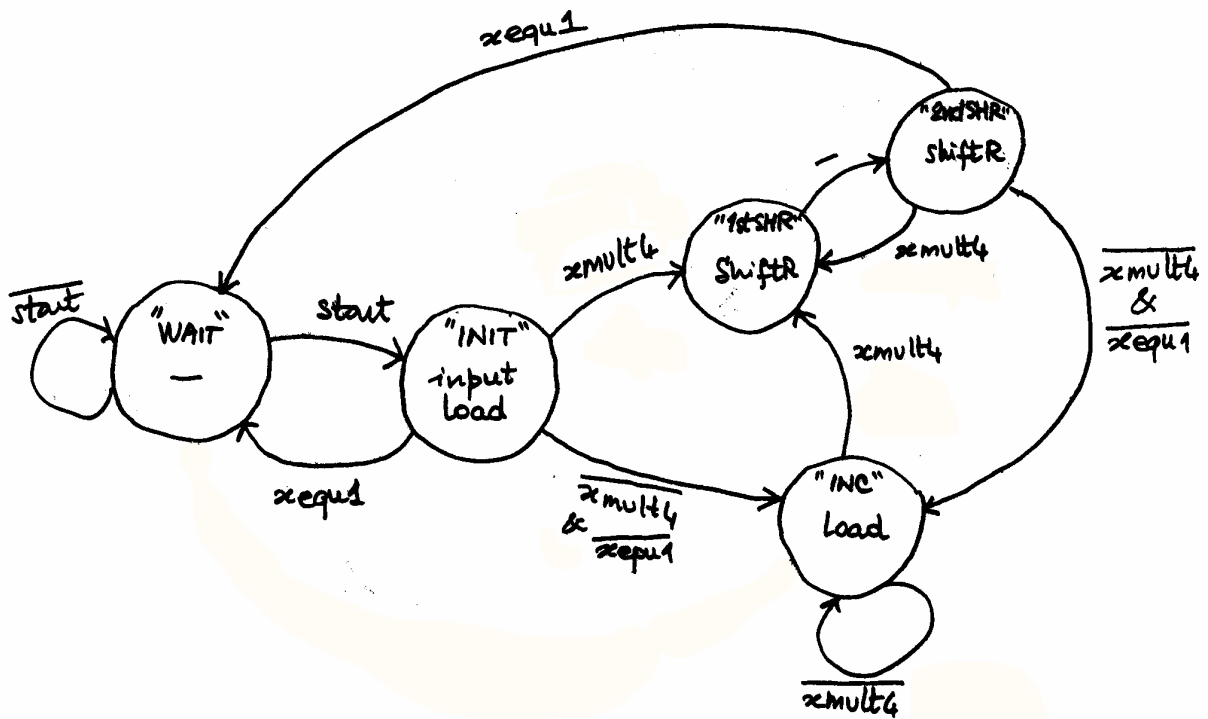
$x \text{ equ } 1$
 $x \text{ mult } 4$

controlli:

input
load
shifter

Nota: sono
mutuamente
esclusivi \Rightarrow
 \Rightarrow si potrebbe
utilizzare
un singolo
segnale
per definirli

Parte Controllo



Esempio:

t	x(t)	xeqv1	xmult4	input	load	shift R	Stato
0	71	0	0	1	1	0	INPUT
1	72	0	1	0	1	0	INC
2	18	0	0	0	0	1	2nd SHR
3	19	0	0	0	1	0	INC
4	20	0	1	0	1	0	INC
5	5	0	0	0	0	1	2nd SHR
6	6	0	0	0	1	0	INC
7	7	0	0	0	1	0	INC
8	8	0	1	0	1	0	INC
9	2	0	0	0	0	1	2nd SHR
10	3	0	0	0	1	0	INC
11	4	0	1	0	1	0	INC
12	1	1	0	0	0	1	2nd SHR