Task2: number request.asm

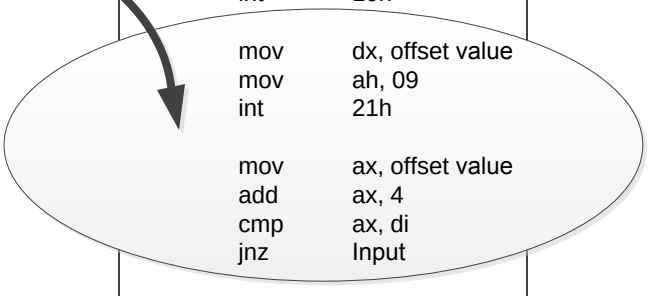
Description of the task

The main goal of this lesson to practice the functions, we have learned so far, and learn the way how we can modify the content of the memory at runtime.
The program waiting a four-digit decimal number through the keyboard. The number is contained in the memory using a variable character by character at ASCII code.
The program clears the screen and only numbers accepts. Pressing other buttons causes error message.

Introducing the program:

- The program has to be prepared with the exception of the storage program. It should be replaced by a JMP instruction. The compiled program does nothing at the first sight if we press a number, pressing any other key causes error message. Pressing ESC key the program terminated.
- the full prepared program displays the pressed sequence of numbers. If the length of the typed numbers reached 4, it displays and the program terminated (jumps to the end of the program).

Optimatization possibility

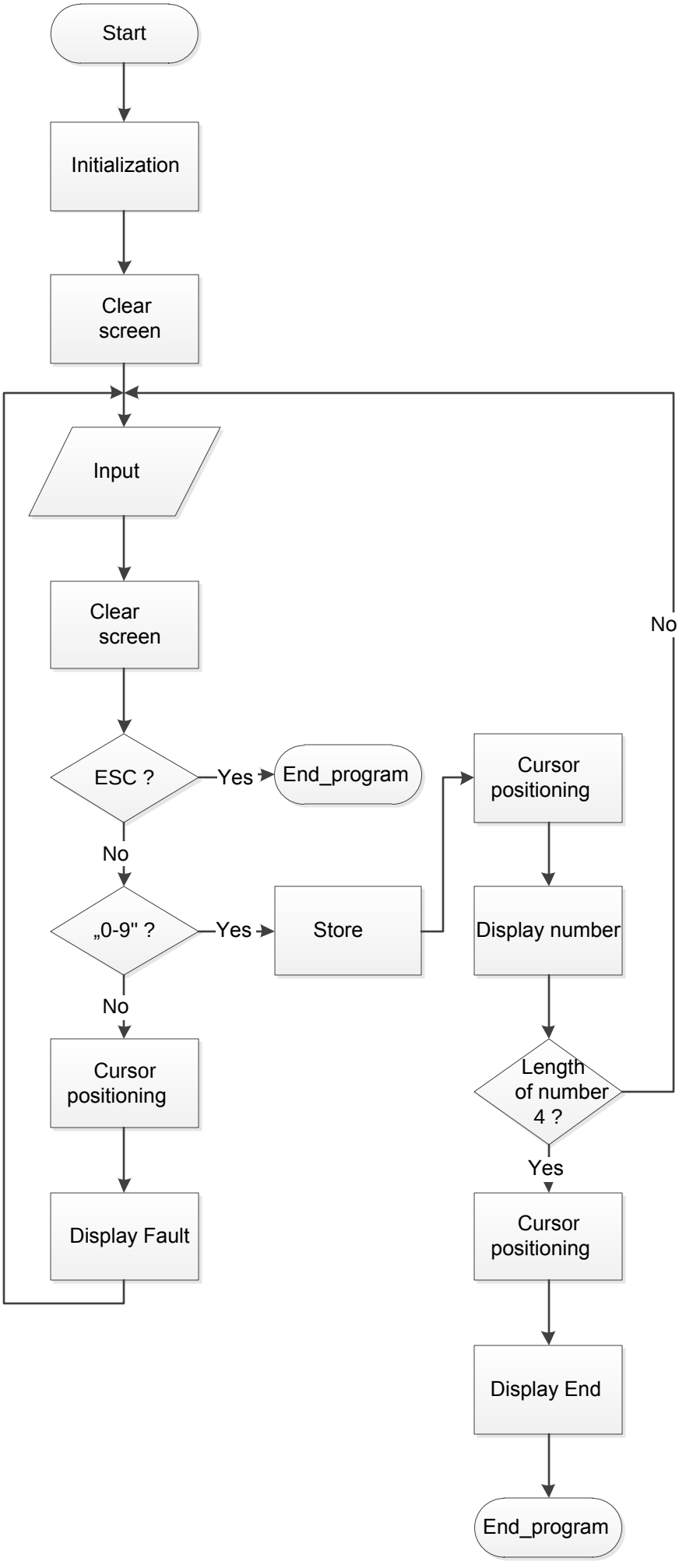


Implementation

```
Code    Segment
        assume CS:Code,
        DS:Data, SS:Stack
Start:
        mov     ax, Code
        mov     ds, ax

        mov     di, offset value

        mov     ax, 03
        int     10h
Input:
        xor     ax, ax
        int     16h

        mov     bx,ax
        mov     ax, 03
        int     10h
        mov     ax,bx

        cmp     al, 27
        jz      End_program

        mov     cx, 10
        mov     ah, '0'
Exam:
        cmp     al, ah
        jz      Store
        inc     ah
        loop    Exam

        mov     ah, 02h
        mov     bh, 0
        mov     dh, 10
        int     10h

        mov     dx, offset fault
        mov     ah, 09
        int     21h

        jmp     Input
Store:
        mov     [di], al
        inc     di
        mov     al, '$'
        mov     [di], al

        mov     ah, 02h
        mov     bh, 0
        mov     dh, 5
        mov     dl, 28
        int     10h

        mov     dx, offset value
        mov     ah, 09
        int     21h

        mov     ax, offset value
        add     ax, 4
        cmp     ax, di
        jnz     Input

        mov     ah, 02h
        mov     bh, 0
        mov     dh, 7
        mov     dl, 0
        int     10h
```

Implementation

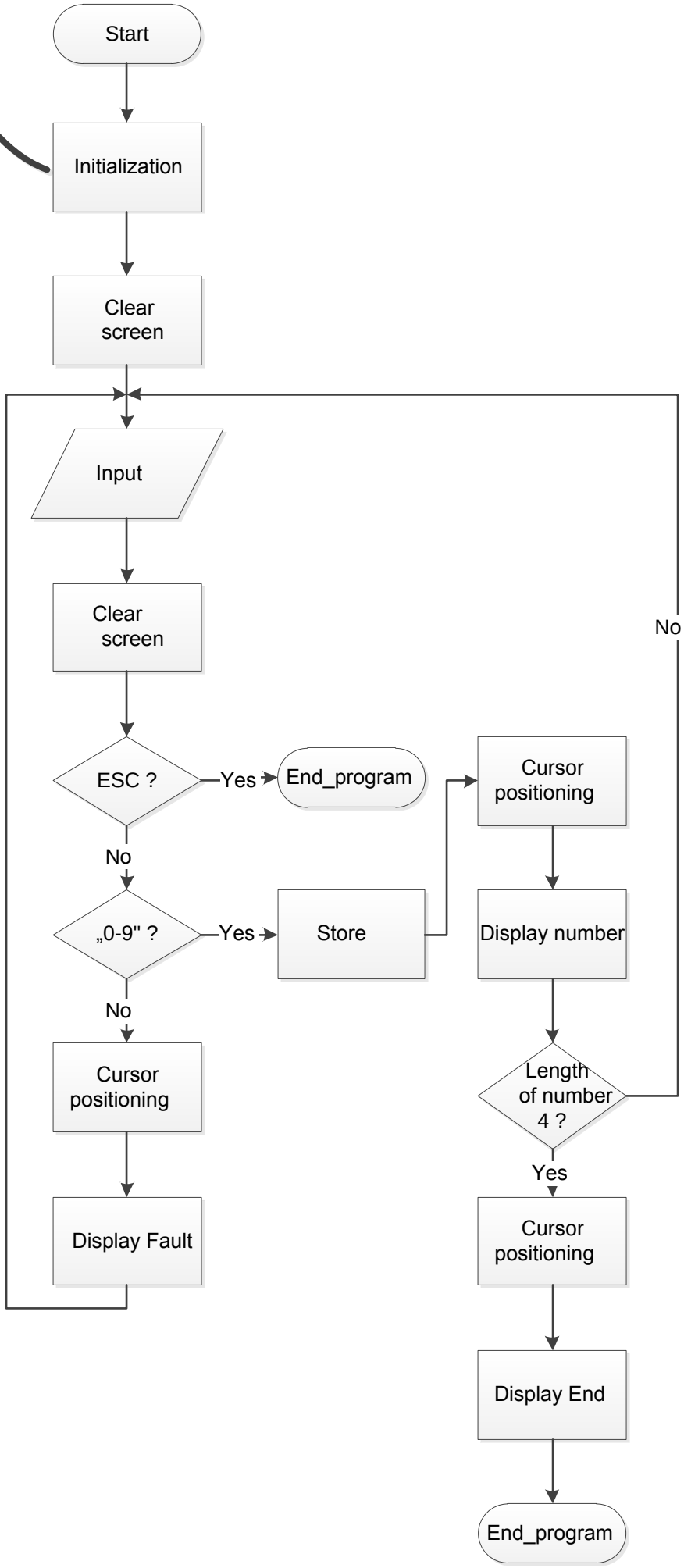
```
        mov     dx, offset message
        mov     ah, 09h
        int     21h
End_program:
        mov     ax, 4c00h
        int     21h

fault:  db      'illegal character!$'

value:  db      '*****$'

message: db     'End of input$'

Code    Ends

Data    Segment
Data    Ends

Stack   Segment
Stack   Ends

        End Start
```

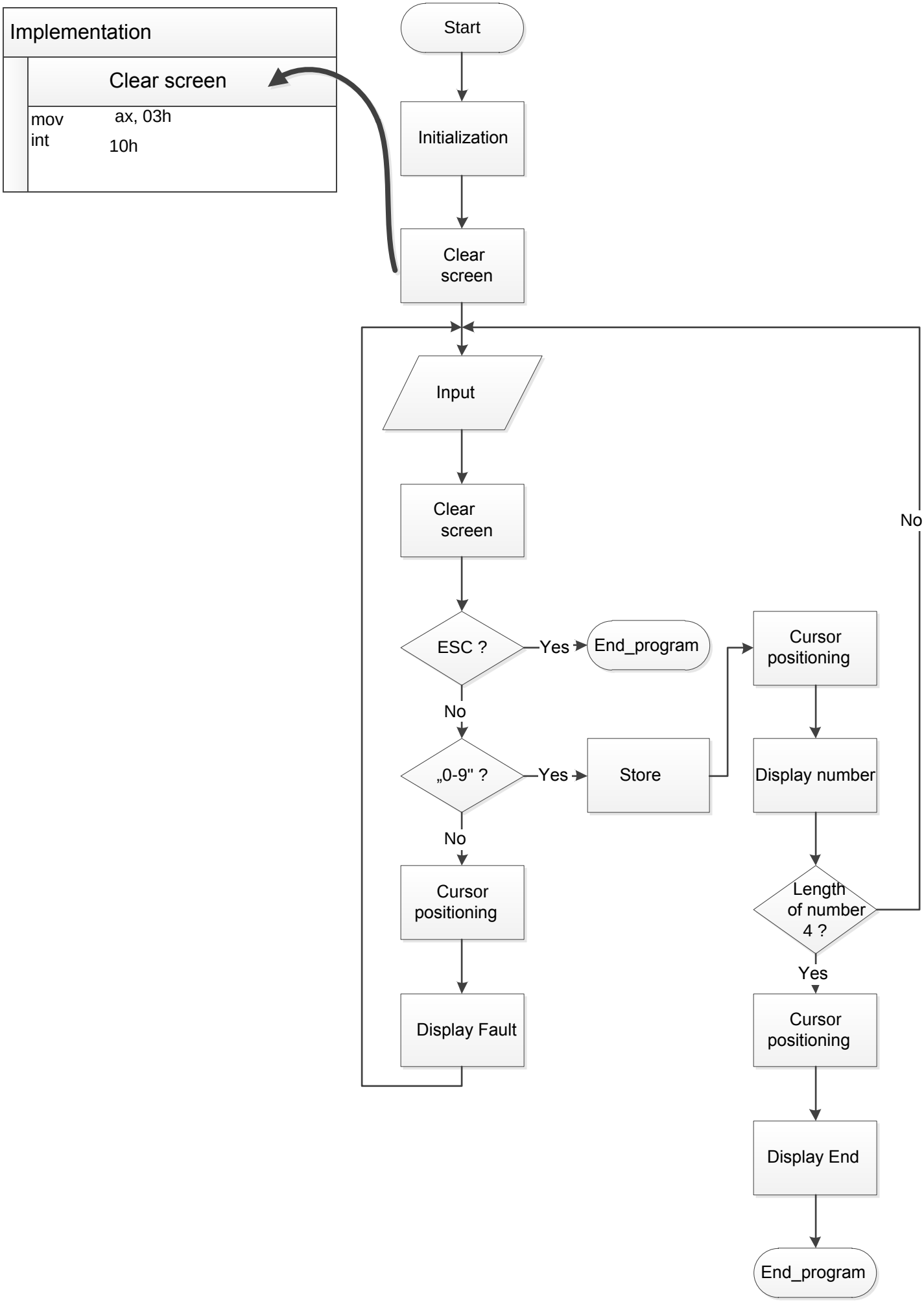

Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
Cursor positioning
Display number
No
Cursor positioning
Display Fault
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program

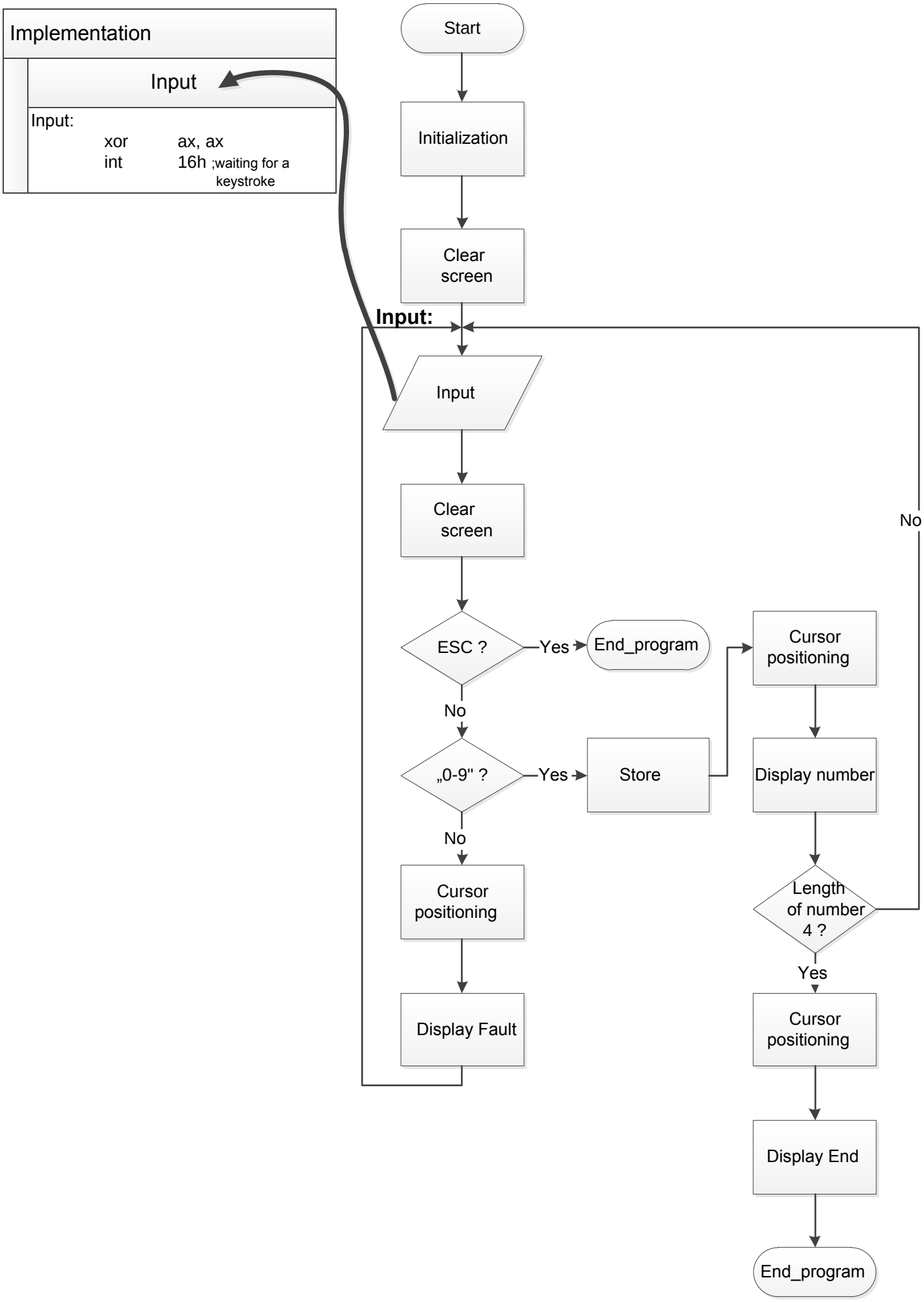
Implementation

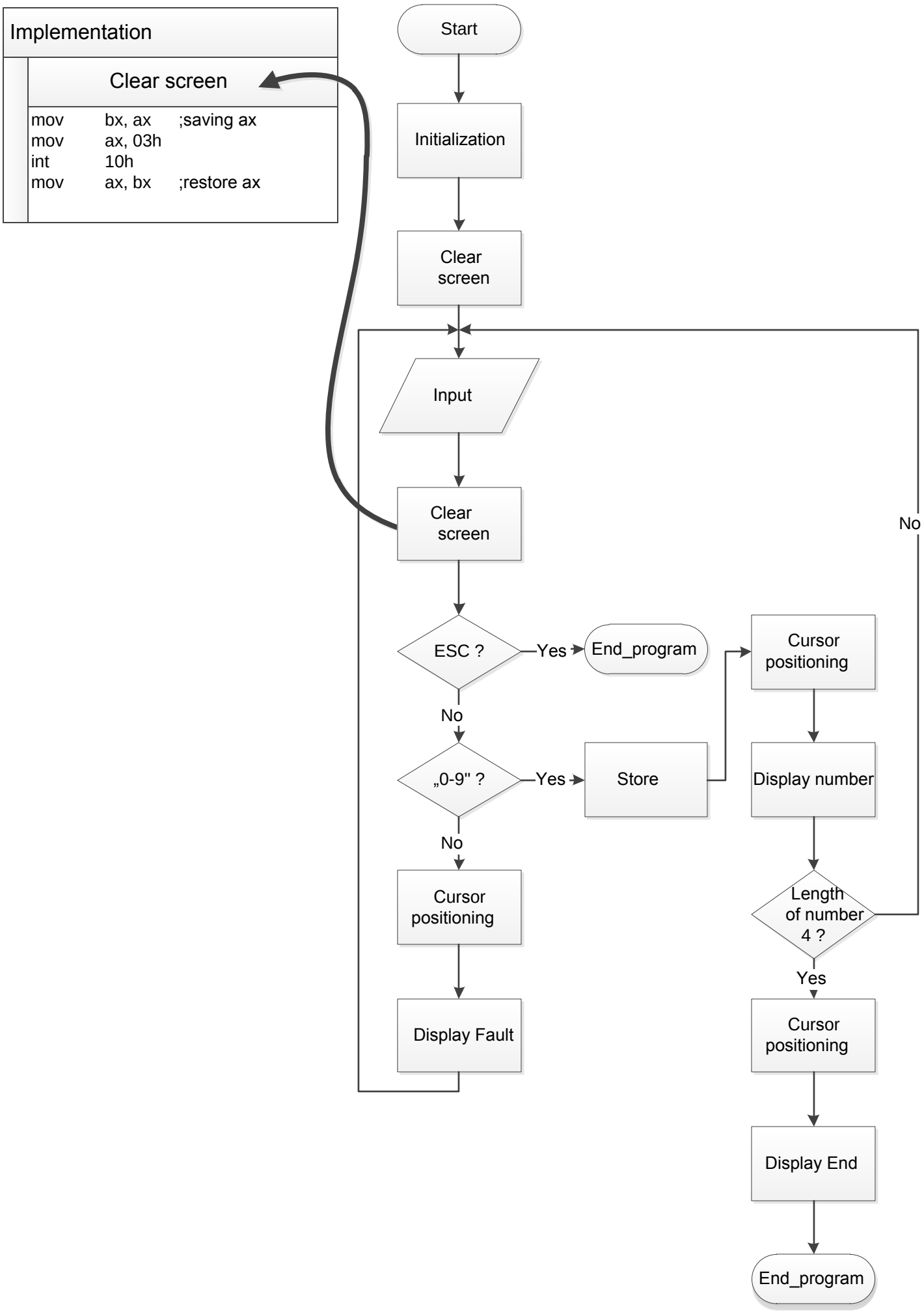
Initialization

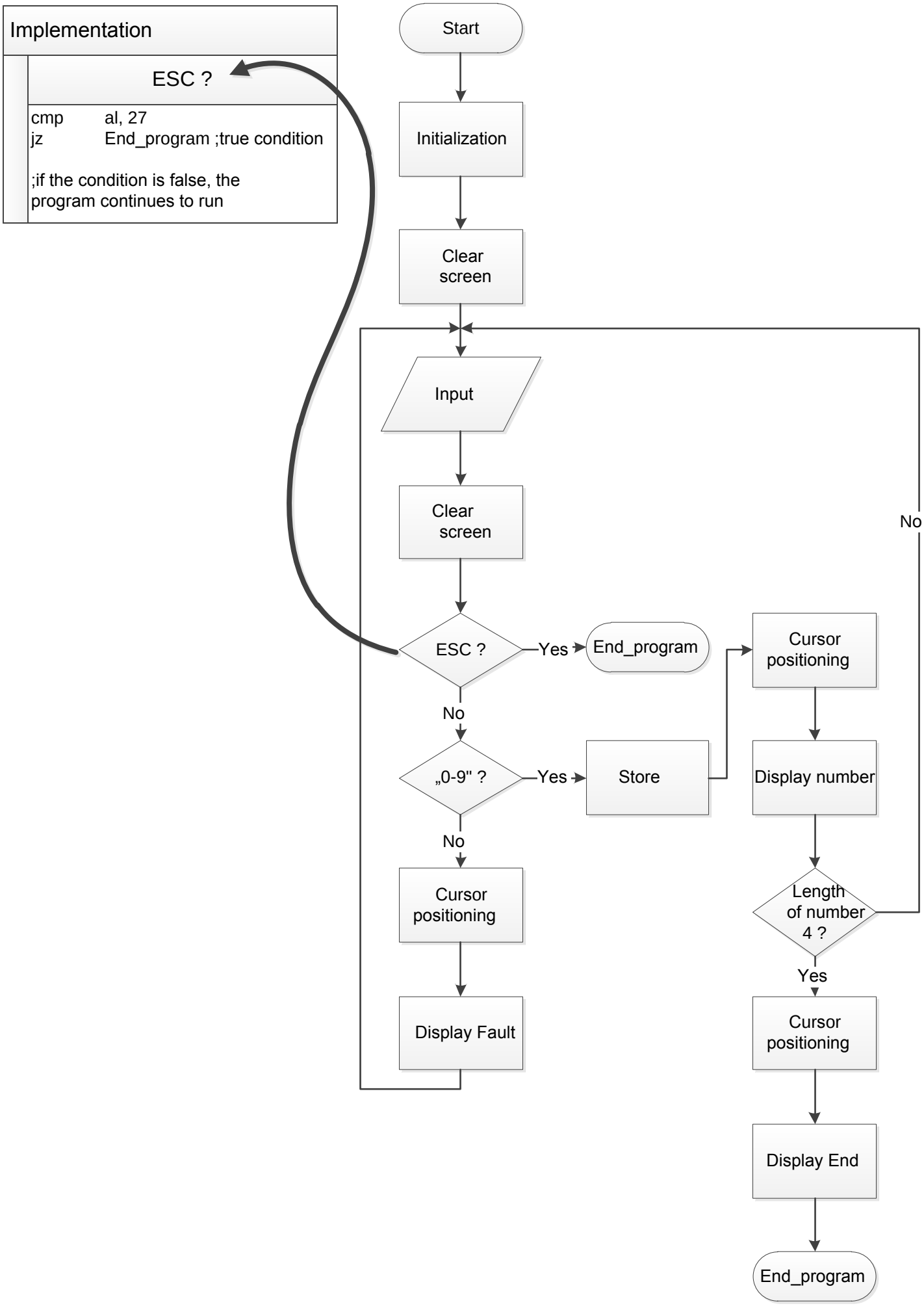
```
;after End_program, to the Code
segment
fault:      db ”Illegal character!$”
value:      db ”****$”
message:    db ”end of input$”
-------------------------------------------------
--- ;the beginning of the program
mov         di, offset value
```

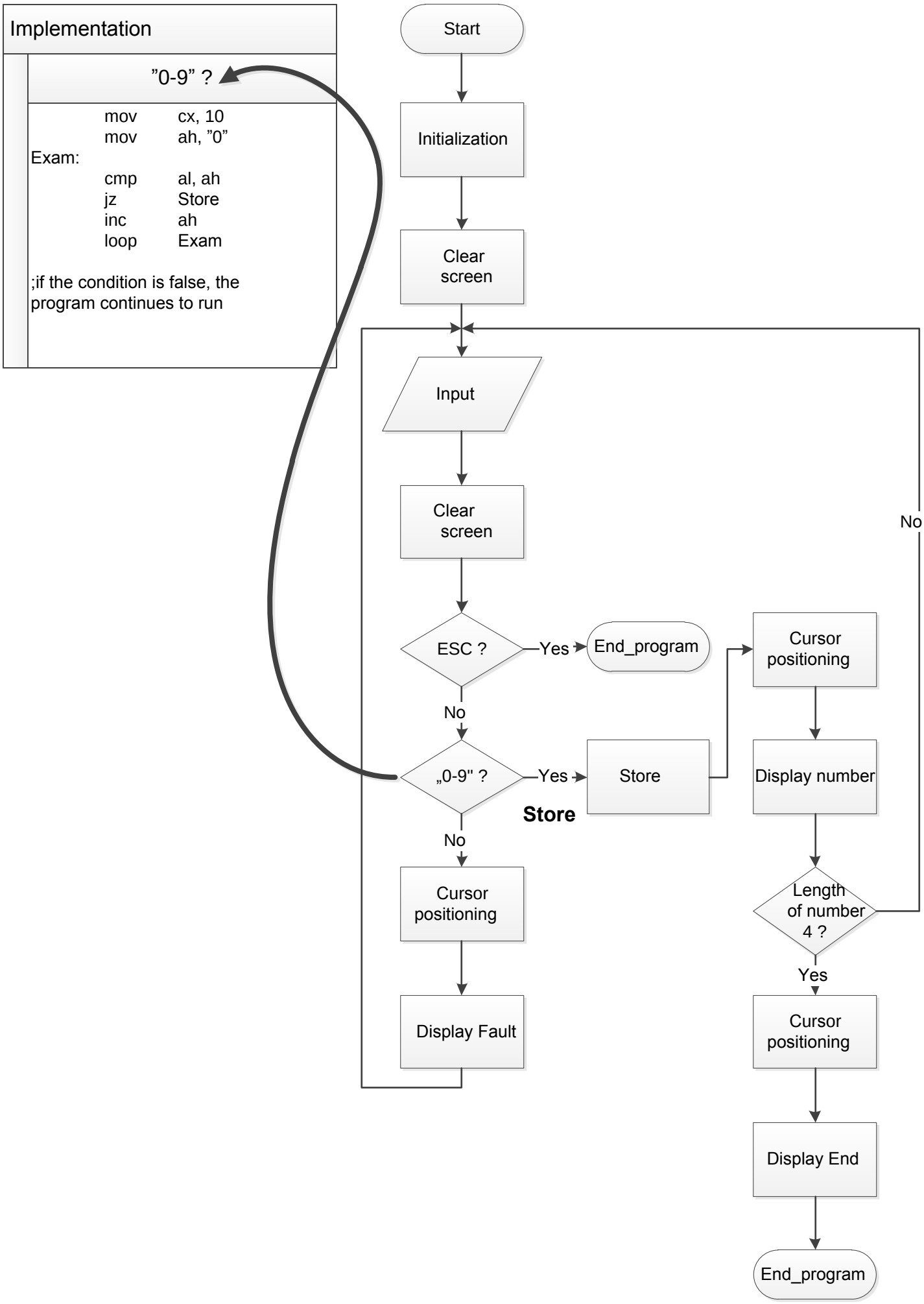


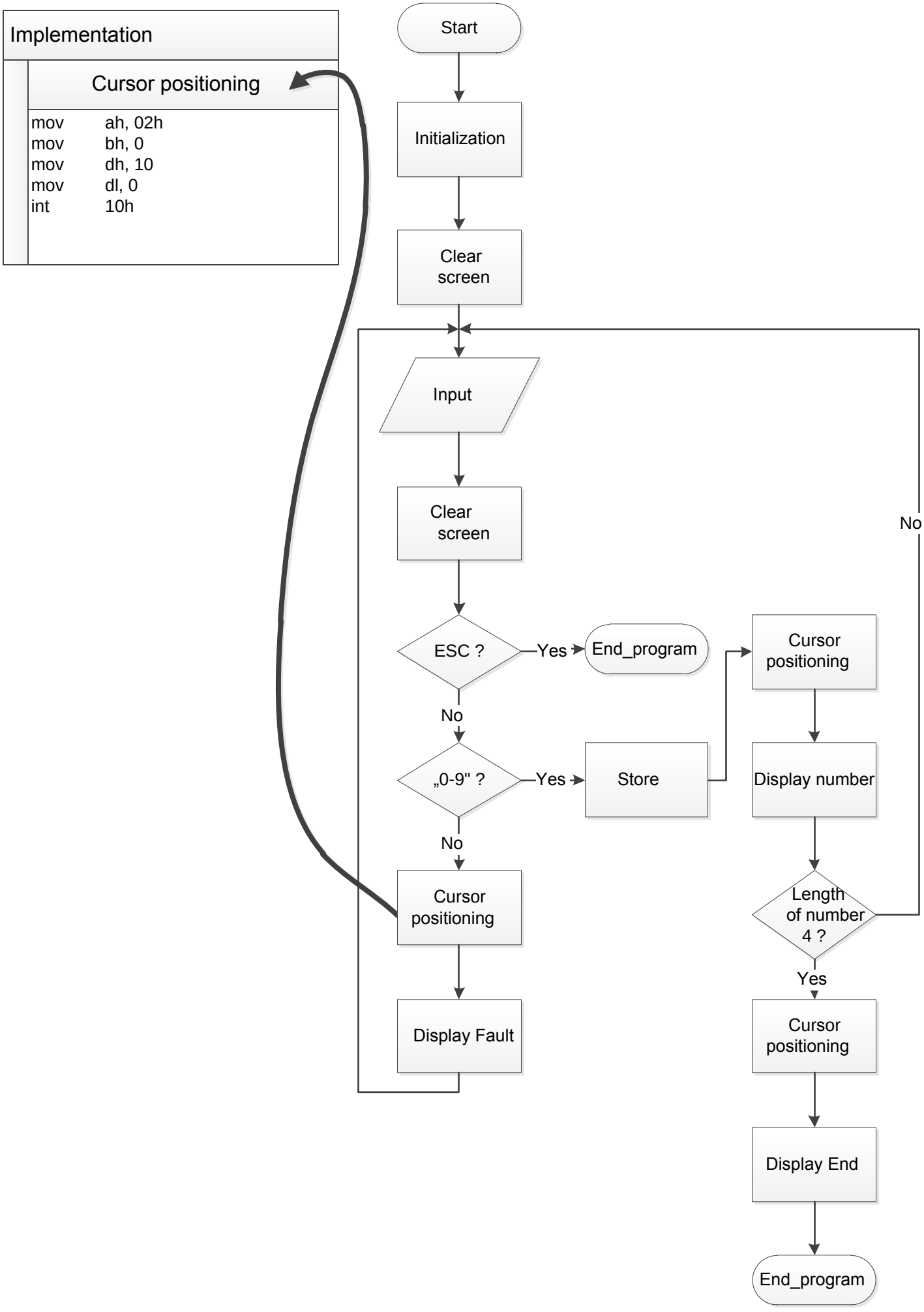

Implementation
Clear screen
mov ax, 03h
int 10h
Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program

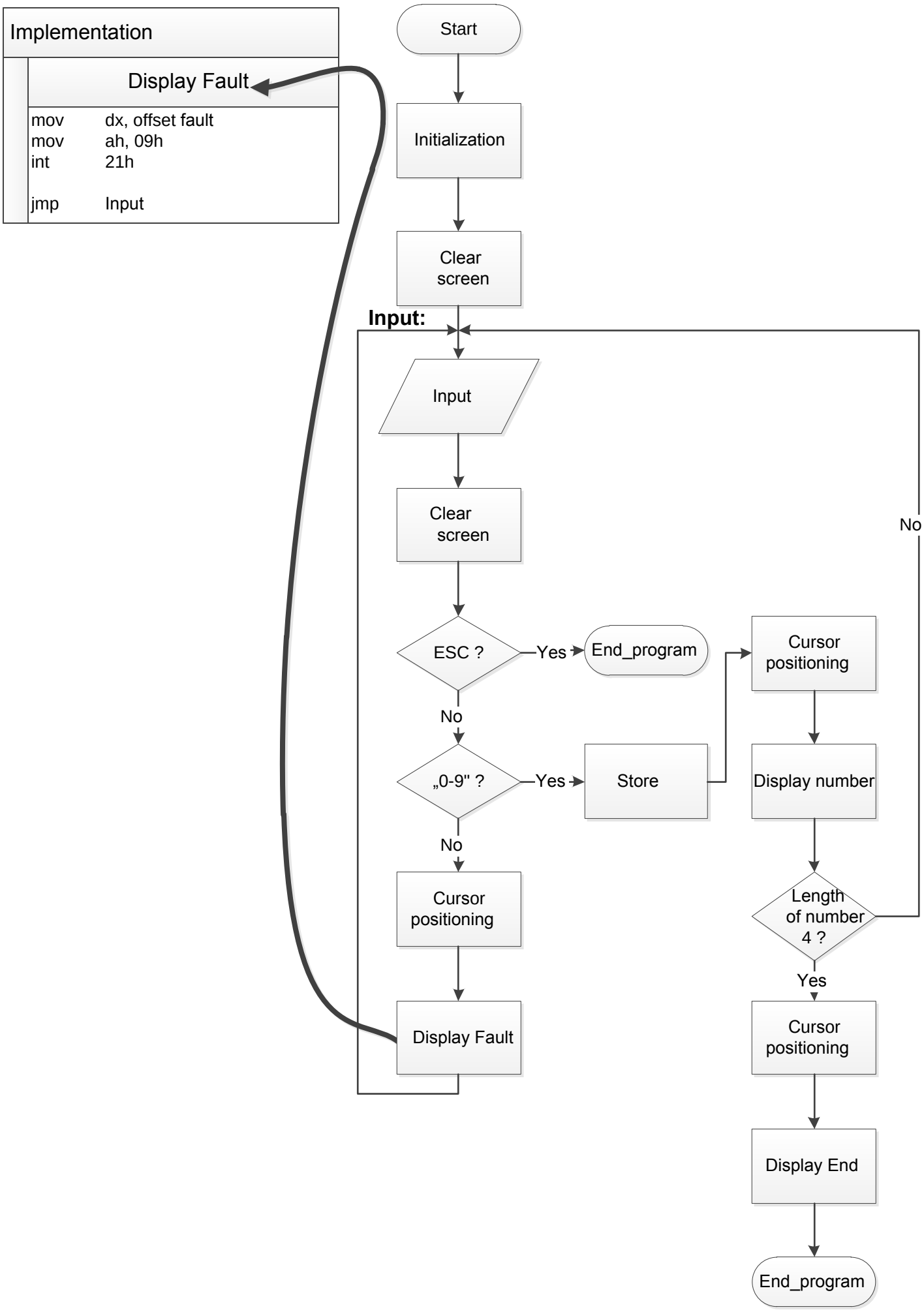

Implementation
Input
Input:
xor ax, ax
int 16h ;waiting for a keystroke
Start
Initialization
Clear screen
Input:
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program

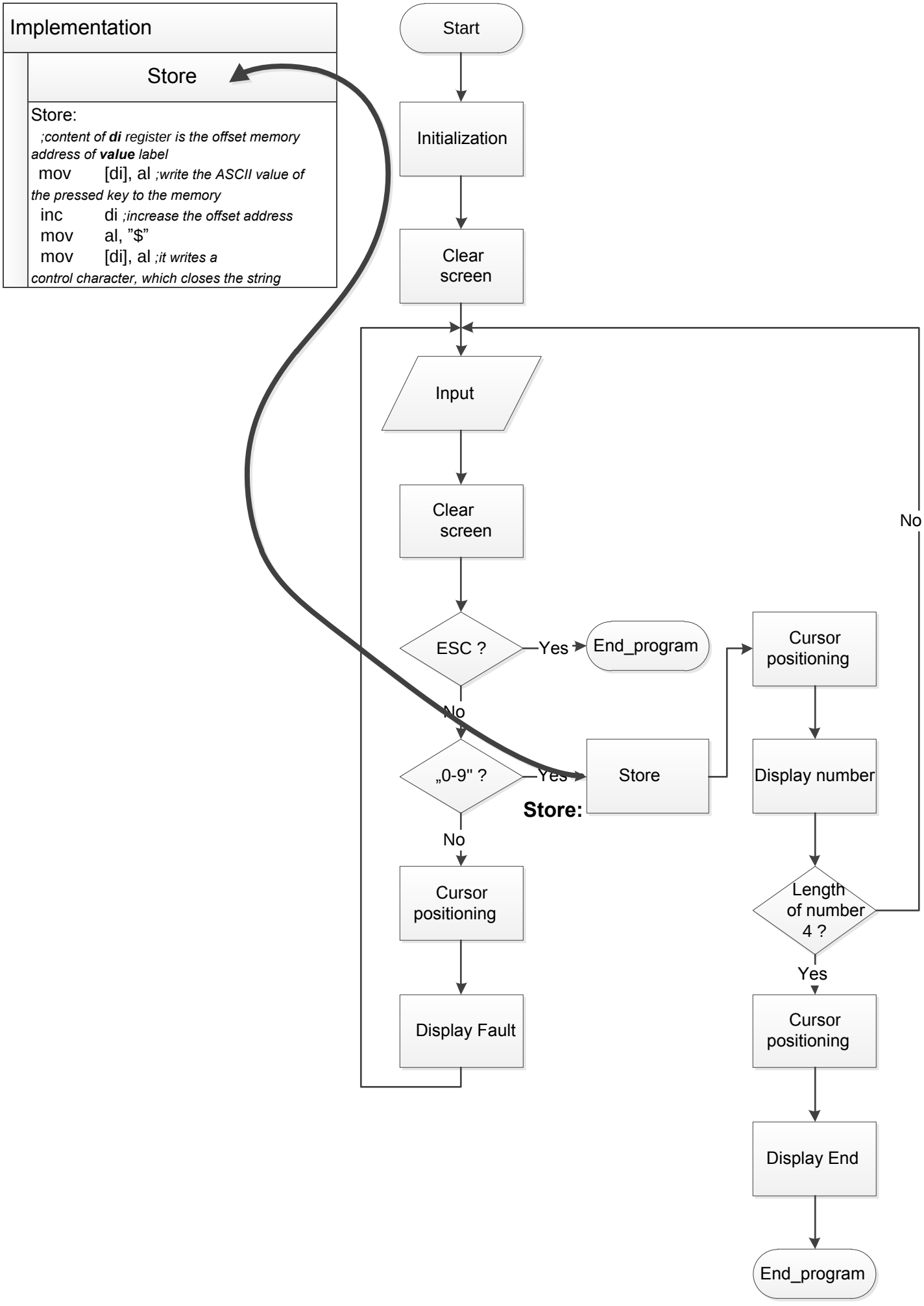

Implementation
Clear screen
mov bx, ax ;saving ax
mov ax, 03h
int 10h
mov ax, bx ;restore ax
Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program

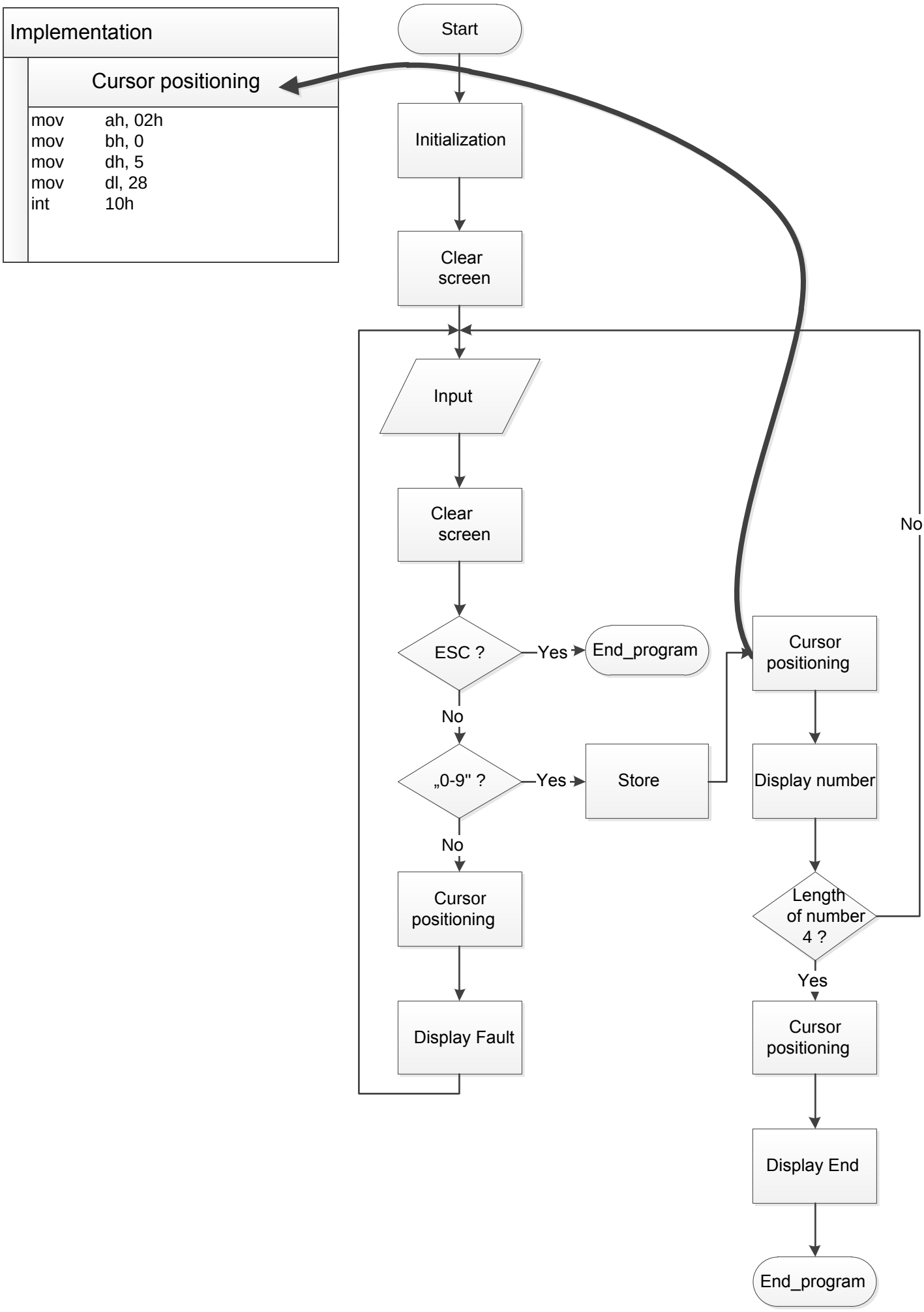

Implementation
ESC ?
cmp al, 27
jz End_program ;true condition
;if the condition is false, the
program continues to run
Start
Initialization
Clear
screen
Input
Clear
screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor
positioning
Display Fault
Cursor
positioning
Display number
Length
of number
4 ?
No
Yes
Cursor
positioning
Display End
End_program

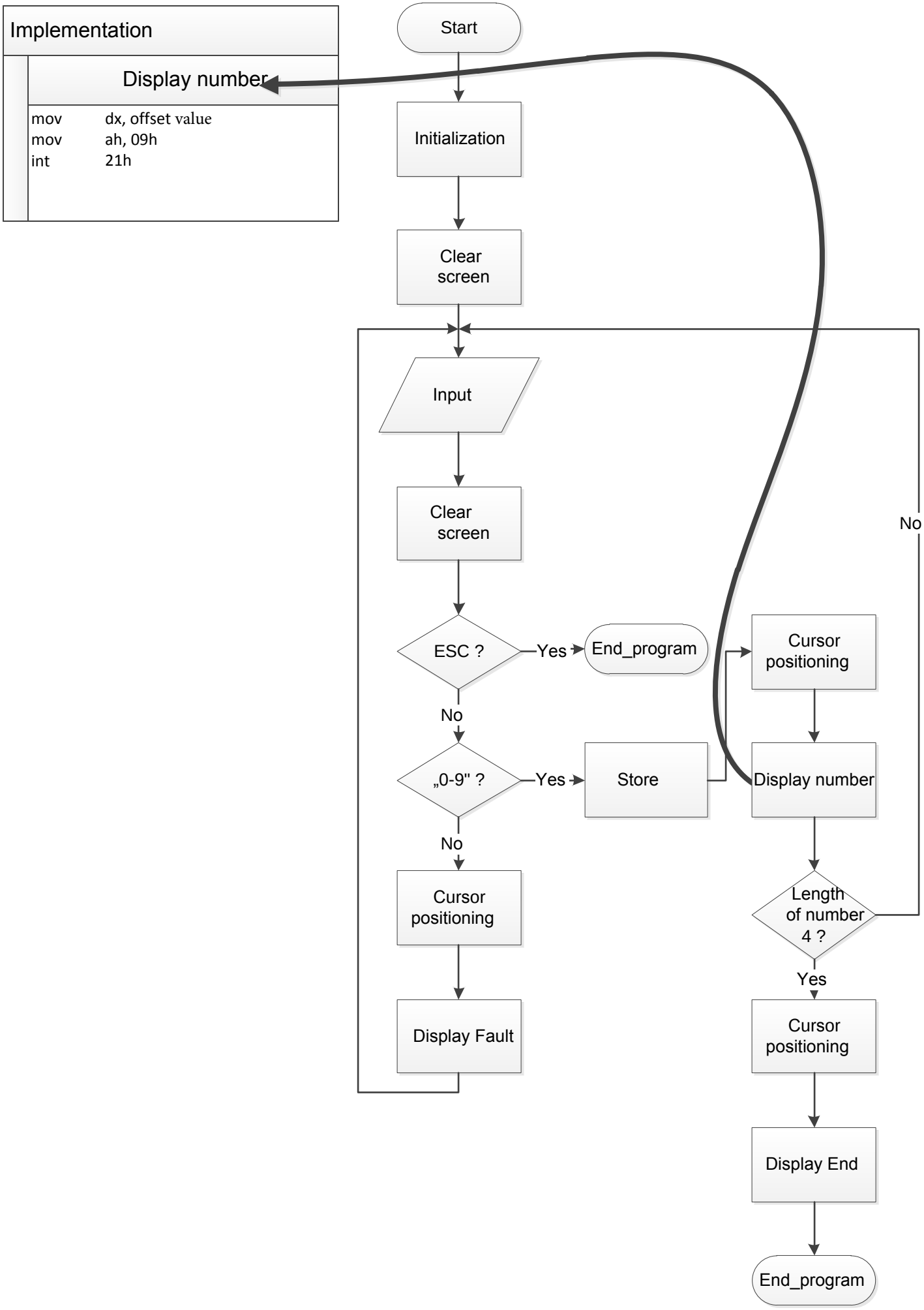

Implementation
”0-9” ?
mov cx, 10
mov ah, ”0”
Exam:
cmp al, ah
jz Store
inc ah
loop Exam
;if the condition is false, the
program continues to run
Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Store
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program


Implementation
Cursor positioning
mov ah, 02h
mov bh, 0
mov dh, 10
mov dl, 0
int 10h
Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program


Implementation
Display Fault
mov dx, offset fault
mov ah, 09h
int 21h
jmp Input
Start
Initialization
Clear screen
Input:
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program


Implementation
Store
Store:
;content of di register is the offset memory address of value label
mov [di], al ;write the ASCII value of the pressed key to the memory
inc di ;increase the offset address
mov al, "$"
mov [di], al ;it writes a control character, which closes the string
Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store:
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program

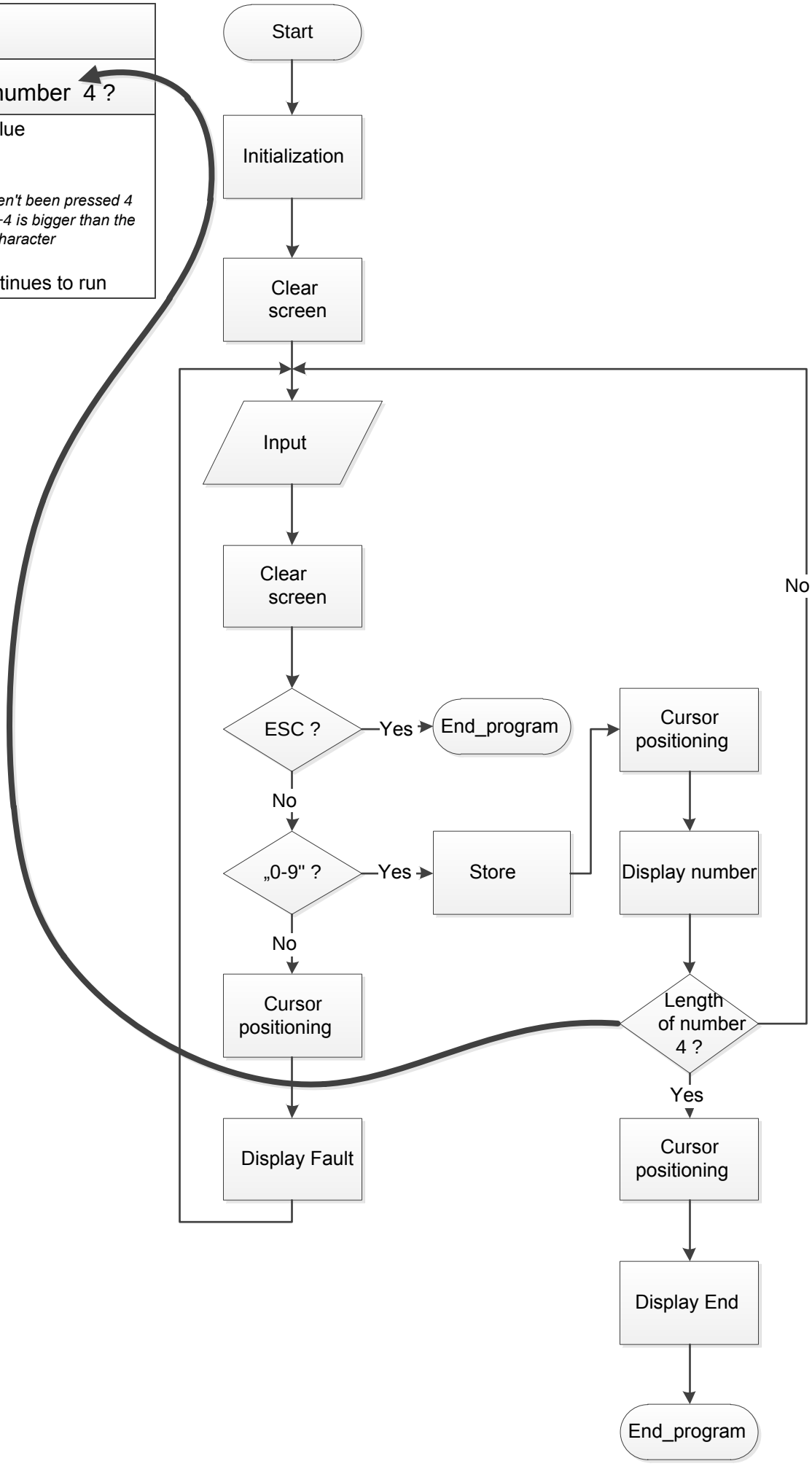

Implementation
Cursor positioning
mov ah, 02h
mov bh, 0
mov dh, 5
mov dl, 28
int 10h
Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program

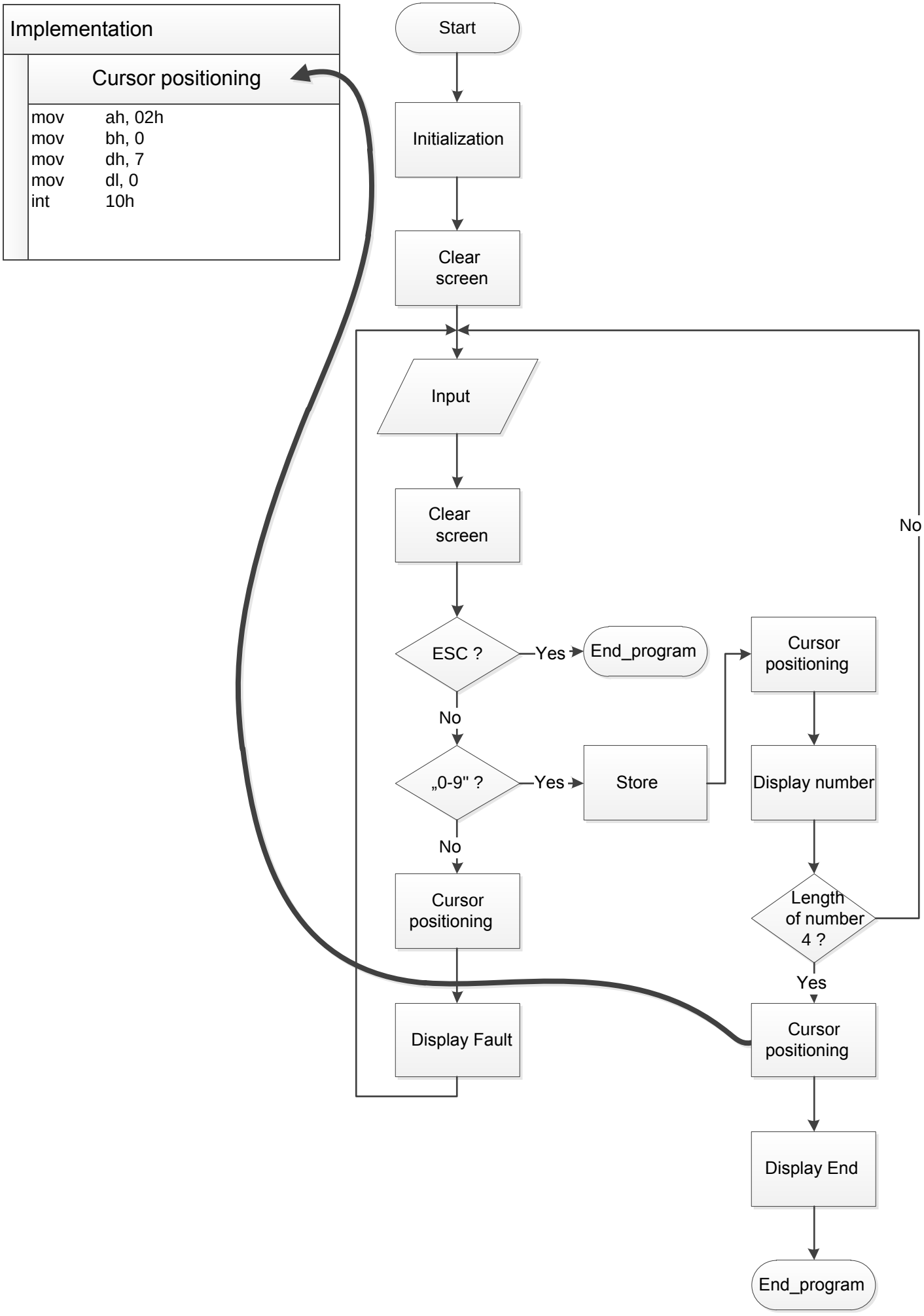

Implementation
Display number
mov dx, offset value
mov ah, 09h
int 21h
Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program

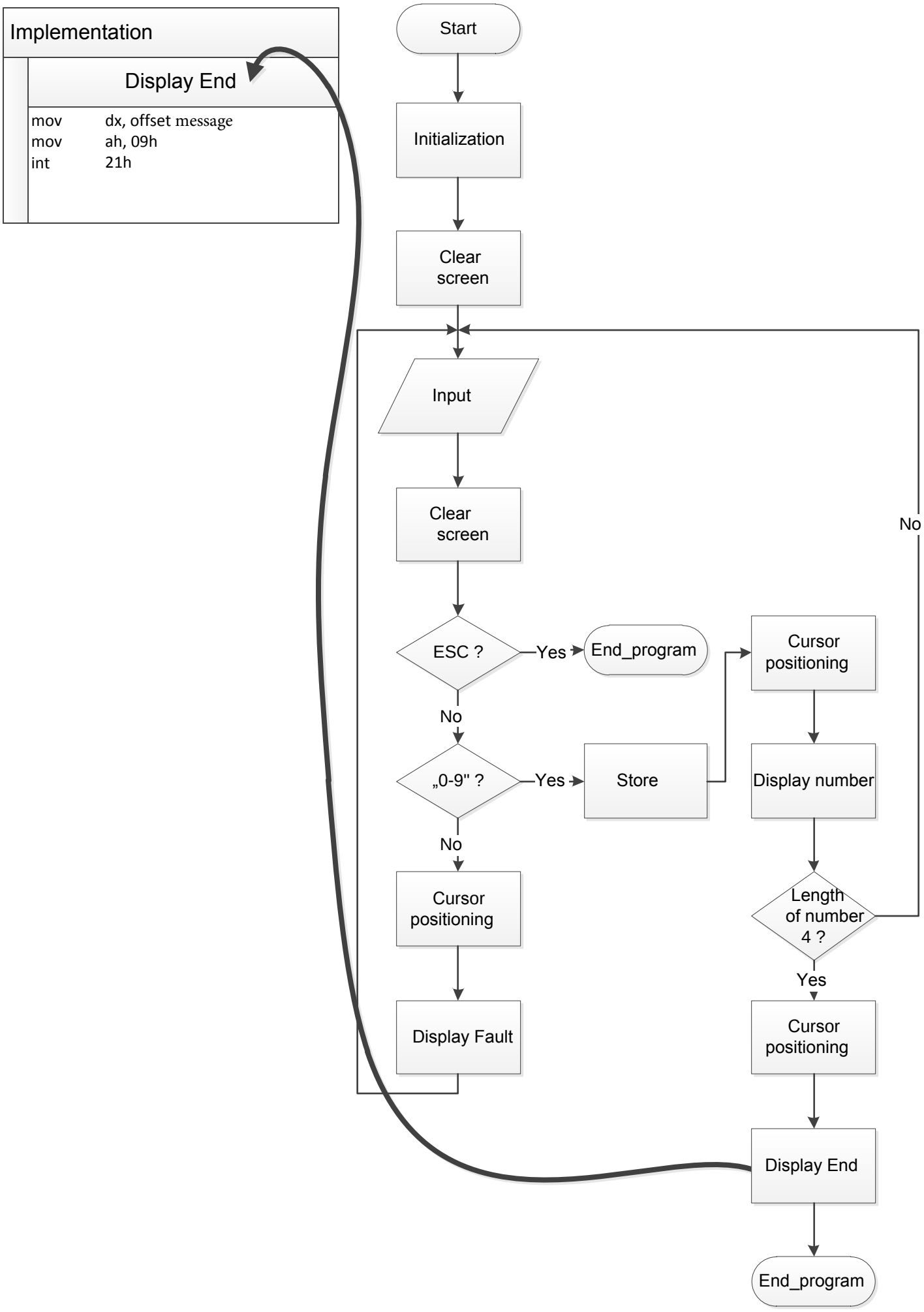
Implementation

Is the length of number 4 ?

```
mov     ax, offset value
add     ax, 4
cmp     ax, di
jnz     Input ;we haven't been pressed 4 numeric key, start address+4 is bigger than the address of the last stored character

;else the program continues to run
```




Implementation
Cursor positioning
mov ah, 02h
mov bh, 0
mov dh, 7
mov dl, 0
int 10h
Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program


Implementation
Display End
mov dx, offset message
mov ah, 09h
int 21h
Start
Initialization
Clear screen
Input
Clear screen
ESC ?
Yes
End_program
No
„0-9" ?
Yes
Store
No
Cursor positioning
Display Fault
Cursor positioning
Display number
Length of number 4 ?
No
Yes
Cursor positioning
Display End
End_program