*9. Design and develop an assembly program to display messages "FIRE" and "HELP" alternately with flickering effects on a 7-segment display interface for a suitable period of time. Ensure a flashing rate that makes it easy to read both the messages (Examiner does not specify these delay values nor is it necessary for the student to compute these values).*

```
.model small

initds macro
    mov ax,@data        ; initializing the data segment
    mov ds,ax           ; it is ds, not dx
endm

init8255 macro
    mov al,cw       ; initialization of 8255 using control word
    mov dx,cr         by passing 80h to control reg.
    out dx,al         (to make port B as output & port C as output)
endm

outpb macro
    mov dx,pb           ; initialization of port B as output
    out dx,al
endm

outpc macro
    mov dx,pc           ; initialization of port C as output
    out dx,al
endm

printf macro msg
    lea dx,msg          ; load the effective address to dx
    mov ah,9            ; function number is 9
    int 21h             ; using dos interrupt 21h
endm

exit macro
    mov ah,4ch          ; to terminate
    int 21h
endm
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
.data
    pb equ 1191h    ;setting the port address for port B
    pc equ 1192h    ;setting the port address for port C
    cr equ 1193h    ;setting the port address for control reg.

    cw db 80h       ;80h is the value in control word 10000000, which
                    makes port B as output & port C as out put
    anykeytoexit db 10,13,"press any key to exit $"

            ;F    I    r    E
    fire db 8eh,0cfh,0afh,86h

            ;H    E    L    P
    help db 89h,86h,0c7h,8ch
```

As capital R cannot be displayed we are considering small r (if you want R, go ahead in terms of A - 88H)

variables FIRE & HELP contains hexa decimal values for FIRE & HELP to display on 7-segment display module. You can even try displaying your name with hexa decimal values.

```
.code
    initds
    init8255

    printf anykeytoexit   ; displays press any key to exit

    start:
        lea si,fire       ; loads the address of fire to si
        call disp_msg     ; displays the contents of table form fire
        call delay

        lea si,help       ; loads the address of help to si
        call disp_msg     ; displays the contents of table form help
        call delay

        mov ah,1        ; checks if any key from key board is  pressed
        int 16h
        jz start

        exit              ; terminate program


disp_msg proc       ; displaying char starts from this proc
    mov cx,4 ; count is taken 4 b'coz of 4 char in 1st string i.e. fire
    nextchar:
        mov bl,8          ; bl indicates 8 bits in single char
        mov al,[si]       ; char is moved to al from si which is in the
                        form of 8-bit data
        nextbit:
            rol al,1      ; rotate left will sends data out bit by bit
            outpb         ; sends bit by bit to output module

            push ax       ; keeps copy of ax in stack b'coz next
                                instruction changes it.
            mov al,00h    ; clock pulse 0 given to drive the bits on
             outpc                      led through port c

             mov al,11h   ; clock pulse 1 given to drive the bits on
             outpc                      led through port c

            pop ax        ; copy is retrieved from stack

            dec bl        ; decrements the bit count
            jnz néxtbit   ; repeats until bit count becomes 0

        inc si            ; si is pointed to next char
        loop nextchar    ; automatically decrements char count (cx)

        ret               ; returns the control to called instruction
disp_msg endp
```

```
delay proc

    mov bx,0ffffh      ; do a waste job waste number of times!!!!
    outerfor:
        mov cx,0ffffh

    innerfor:
        loop innerfor
      dec bx
      jnz outerfor
      ret

delay endp


end
```

```
for (bx = bignumber; bx >= 0; bx --)
{
      for(cx = bignumber; cx >= 0; cx --
)
      {
            Do nothing;
      }
}

basically, keep decrementing a huge
number till zero huge number of times.

By the time, microprocessor does this
huge decrements, you can actually see
```