Name : Sajid Islam

Instructor : Muhammad Hassaan

COAL

Assessment No 13

Roll No : 24p-0745

Q1

```
[org 0x0100]


mov ax, 0B800h     ; video memory

mov es, ax


mov cx, 80         ; 80 chars

mov di, 0          ; start at 0


mov ah, 1Eh        ; attribute

mov al, 'X'        ; character


cld

rep stosw          ; fill row


mov ah, 00h        ; wait key

int 16h
```

```
mov ax, 4C00h
int 21h
```

Q2

```
[org 0x0100]

mov ax, 0B800h        ; video memory
mov es, ax
mov ds, ax

mov ax, 80            ; calc row 5 offset
mov bx, 5
mul bx
shl ax, 1
mov si, ax

mov ax, 80            ; calc row 10 offset
mov bx, 10
mul bx
shl ax, 1
```

```
mov di, ax

mov cx, 80          ; 80 words
cld
rep movsw           ; copy row

mov ah, 00h         ; wait key
int 16h

mov ax, 4C00h
int 21h
```

Q3

```
[org 0x0100]

mov ax, 0B800h      ; video memory
mov es, ax
mov ds, ax

mov ax, 79          ; last col of row 0
```

```
shl ax, 1

mov si, ax


mov ax, 80+79        ; last col of row 1

shl ax, 1

mov di, ax


mov cx, 80           ; 80 words

std                  ; backward copy

rep movsw            ; copy row


mov ah, 00h          ; wait key

int 16h


mov ax, 4C00h

int 21h
```

Q4

```
[org 0x0100]

jmp start

row: dw 3              ; row
col: dw 20             ; column
mgs: db "COAL LAB 13",0

start:
mov ax, 0B800h         ; video memory
mov es, ax

mov ax, 80             ; row offset
mov bx, [row]
mul bx
add ax, [col]
shl ax, 1              ; convert to bytes
mov di, ax

mov ah, 07h            ; attribute
```

```
mov si, mgs

cld

print:
    lodsb               ; get char
    mov [es:di], ax     ; write
    add di, 2
    cmp byte [si], 0    ; end?
    jnz print

mov ah, 00h             ; wait key
int 16h

mov ax, 4C00h
int 21h
```

Q5

```
[org 0x0100]

jmp start

vptr: dw 0, 0B800h     ; offset, segment

start:
   les di, [vptr]     ; load ES:DI

   mov ax, 0F48h      ; 'H' + attribute

   stosw              ; write to video memory

   mov ah, 00h        ; wait key
   int 16h

   mov ax, 4C00h
   int 21h
```

Q6

```
[org 0x100]
start:
    mov ah, 0x35
    mov al, 0x00
    int 0x21
    mov [old_off], bx  ; save offset of old handler
    mov ax, es
    mov [old_seg], ax  ; save segment of old handler

-
    mov dx, new_handler ; offset of new handler
    mov ax, cs
    mov ds, ax          ; DS = CS before DOS call
    mov ah, 0x25
    mov al, 0x00        ; interrupt number = 0
    int 0x21            ; INT 0 now points to new_handler


    mov ax, 5
```

```
    mov bl, 0          ; denominator = 0
    div bl             ; triggers INT 0



    mov dx, old_off
    mov ax, old_seg
    mov ds, ax
    mov ah, 0x25
    mov al, 0x00
    int 0x21



    mov ax, 0x4C00
    int 0x21



new_handler:
    ; Save all registers
    push ax
    push bx
    push cx
```

```
    push dx
    push si
    push di
    push bp
    push ds
    push es

    mov ax, cs
    mov ds, ax          ; DS = CS to access message

    mov si, message     ; point to message string

print_loop:
    lodsb               ; load next character into AL
    cmp al, 0           ; check for null terminator
    je done_print
    mov ah, 0x0E        ; BIOS teletype function
    mov bh, 0x00        ; page 0
    int 0x10            ; print character
    jmp print_loop
```

```
done_print:

    pop es
    pop ds
    pop bp
    pop di
    pop si
    pop dx
    pop cx
    pop bx
    pop ax

    iret                ; return from interrupt


old_off dw 0            ; old INT 0 offset
old_seg dw 0            ; old INT 0 segment
message db 'Division Error!', 0
```

Q7

```
[org 0x0100]          ; COM program start

start:

    mov ax, cs
    mov es, ax          ; set ES = CS for string scanning
    mov di, str_start
    mov cx, str_length
    mov al, 'a'         ; character to search for

    repne scasb         ; scan string until match or end

    jne char_not_found    ; if not found, jump

    dec di              ; adjust DI to point to matched character
    mov dx, di          ; store offset of found character
    jmp exit_program

char_not_found:
    mov dx, 0FFFFh       ; set DX = FFFFh if character not found
```

```
exit_program:
    mov ax, 4C00h
    int 21h             ; terminate program

str_start db "Assembly Language"
str_length db 17
```

Q8

```
[org 0x0100]            ; COM program starts at offset 100h

start:

    mov ax, cs
    mov ds, ax

    mov ah, 0x35
    mov al, 0x60
    int 0x21            ; get current INT 60h handler

    mov [old_off], bx       ; save old ISR offset
```

```
mov ax, es
mov [old_seg], ax          ; save old ISR segment

mov dx, new_int60_handler  ; offset of new ISR
mov ax, cs
mov ds, ax
mov ah, 0x25
mov al, 0x60
int 0x21                   ; install new INT 60h handler

int 0x60                   ; call new ISR directly

mov dx, [old_off]
mov ax, [old_seg]
mov ds, ax
mov ah, 0x25
mov al, 0x60
int 0x21                   ; restore original INT 60h handler

mov ax, 4C00h
int 21h                    ; terminate program
```

```
; --- New INT 60h handler ---
new_int60_handler:

    push ax
    push bx
    push cx
    push dx
    push si
    push di
    push bp
    push ds
    push es                 ; save all registers

    mov ax, cs
    mov ds, ax              ; DS = CS to access message

    mov si, msg             ; SI → message

.print_loop:
    lodsb                   ; load next character
```

```
    cmp al, 0
    je .done
    mov ah, 0x0E            ; BIOS teletype print
    mov bh, 0x00
    int 0x10
    jmp .print_loop

.done:
    pop es
    pop ds
    pop bp
    pop di
    pop si
    pop dx
    pop cx
    pop bx
    pop ax                  ; restore registers
    iret                    ; return from interrupt

old_off dw 0                ; old INT 60h offset
old_seg dw 0                ; old INT 60h segment
```

```
msg db "Custom Interrupt Called!", 0   ; message printed by ISR
```