# TASK 1:

org 0x0100

jmp start

message: db 'Huzaifa.', 0

; Divide by zero interrupt handler

myisrfor0:

push ax ; push all registers

push bx

push cx

push dx

push si

push di

push bp

push ds

push es

push cs

pop ds ; point ds to our data segment

call clrscr ; clear the screen

; Set up screen position and attribute

mov ax, 30

push ax ; push x position

mov ax, 20

push ax ; push y position

mov ax, 0x71 ; white on blue attribute

push ax ; push attribute

; Set up message address and call printstr

mov ax, message

push ax ; push offset of message

call printstr ; print message

pop es

pop ds

pop bp

pop di

pop si

pop dx

pop cx

pop bx

pop ax

iret ; return from interrupt

; Subroutine to clear the screen

clrscr:

push es

push ax

push cx

push di

mov ax, 0xb800

mov es, ax ; point es to video base

xor di, di ; point di to top left column

mov ax, 0x0720 ; space char in normal attribute

mov cx, 2000 ; number of screen locations

cld ; auto increment mode

rep stosw ; clear the whole screen

pop di

pop cx

pop ax

pop es

ret

; Subroutine to calculate the length of a string

strlen:

push bp

mov bp, sp

push es

push cx

push di

les di, [bp+4] ; point es:di to string

mov cx, 0xffff ; load maximum number in cx

xor al, al ; load a zero in al

repne scasb ; find zero in the string

mov ax, 0xffff ; load maximum number in ax

sub ax, cx ; find change in cx

dec ax ; exclude null from length

pop di

pop cx

pop es

pop bp

ret 4

; Subroutine to print a string

printstr:

push bp

mov bp, sp

push es

push ax

push cx

push si

push di

push ds ; push segment of string

mov ax, [bp+4]

push ax ; push offset of string

call strlen ; calculate string length

cmp ax, 0 ; is the string empty

jz exit ; no printing if string is empty

mov cx, ax ; save length in cx

mov ax, 0xb800

mov es, ax ; point es to video base

mov al, 80 ; load al with columns per row

mul byte [bp+10] ; multiply with y position

add ax, [bp+12] ; add x position

shl ax, 1 ; turn into byte offset

mov di,ax ; point di to required location

mov si, [bp+6] ; point si to string

mov ah, [bp+8] ; load attribute in ah

cld ; auto increment mode

nextchar: lodsb ; load next char in al

stosw ; print char/attribute pair

loop nextchar ; repeat for the whole string

exit: pop di

pop si

pop cx

pop ax

pop es

pop bp

ret 8

start:

; Set up the interrupt vector table

xor ax, ax

mov es, ax ; load zero in es

mov word [es:0\*4], myisrfor0 ; store offset at n\*4

mov [es:0\*4+2], cs ; store segment at n\*4+2

; Generate a divide by zero interrupt

mov ax, 0x8432 ; load a big number into ax

mov bl, 0 ; use a zero divisor to generate interrupt 0

div bl ; interrupt 0 will be generated

; Terminate the program

mov ax, 0x4c00 ; terminate program

int 0x21 ; call DOS

# TASK 2:

[org 0x100]

jmp start

msg1: db 'hello world', 0

msg2: db 'hello world again', 0

msg3: db 'hello world again and again', 0

start:

; Set VGA attributes - toggle blinking

mov ah, 0x10 ; service 10 – VGA attributes

mov al, 03 ; subservice 3 – toggle blinking

mov bl, 01 ; enable blinking bit

int 0x10 ; call BIOS video service

; Get keystroke from the user

mov ah, 0 ; service 0 – get keystroke

int 0x16 ; call BIOS keyboard service

call clrscr ; clear the screen

; Print 'hello world' at (0,0) with blue on black attributes

mov ax, 0

push ax ; push x position

mov ax, 0

push ax ; push y position

mov ax, 1 ; blue on black attribute

push ax ; push attribute

mov ax, msg1

push ax ; push offset of string

call printstr ; print the string

; Wait for another keystroke

mov ah, 0 ; service 0 – get keystroke

int 0x16 ; call BIOS keyboard service

; Similar print and wait sequences for different colors and messages

; ...

; Terminate the program

mov ax, 0x4c00 ; terminate program

int 0x21 ; call DOS