

# OS Assignment1

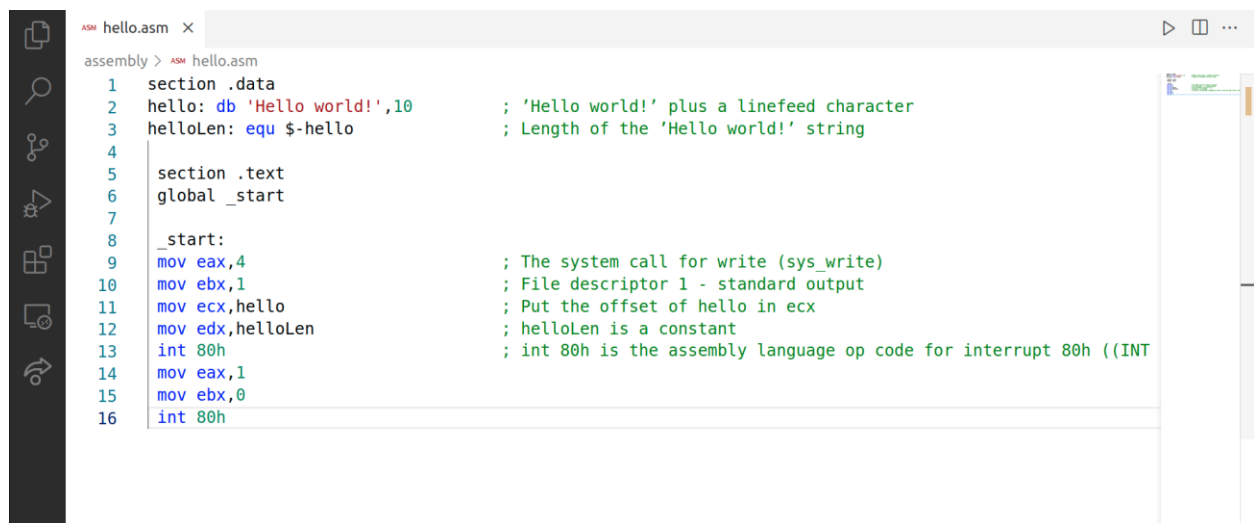
Name:

Subhan Khalid

Roll No:

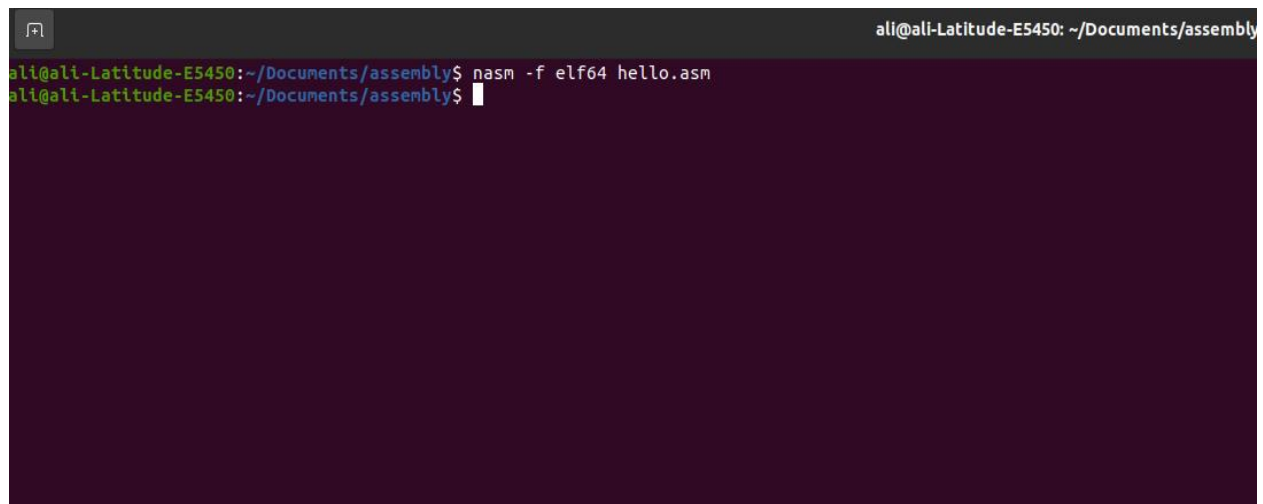
P200086 (4A)

1:



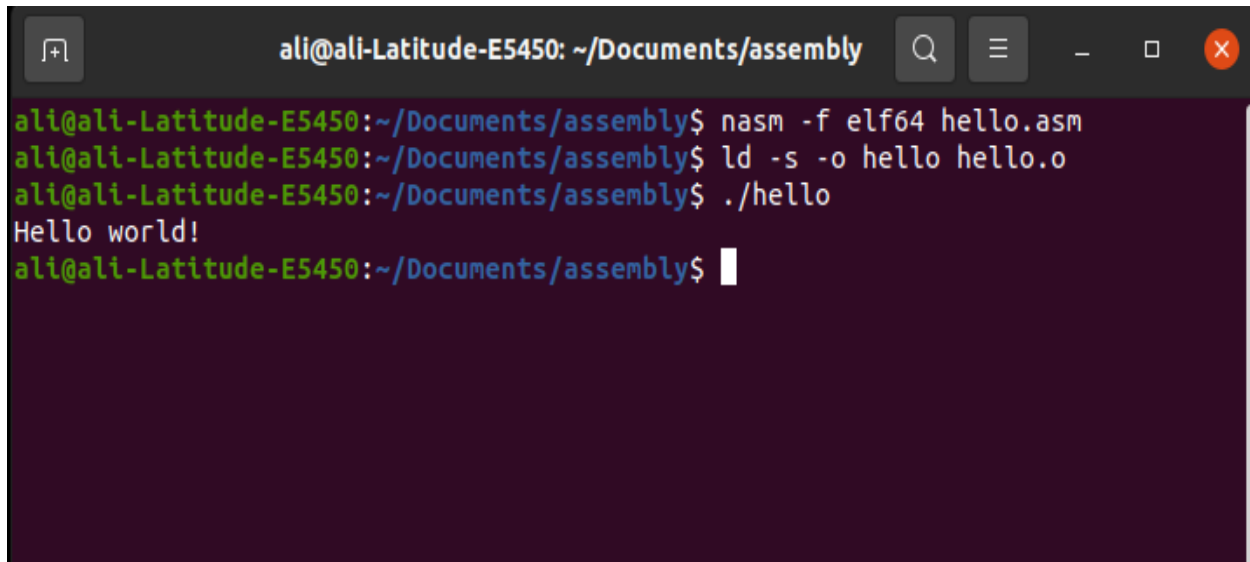
```
ASM hello.asm x
assembly > ASM hello.asm
1  section .data
2  hello: db 'Hello world!',10      ; 'Hello world!' plus a linefeed character
3  helloLen: equ $-hello           ; Length of the 'Hello world!' string
4
5  section .text
6  global _start
7
8  _start:
9  mov eax,4                       ; The system call for write (sys_write)
10 mov ebx,1                       ; File descriptor 1 - standard output
11 mov ecx,hello                   ; Put the offset of hello in ecx
12 mov edx,helloLen               ; helloLen is a constant
13 int 80h                       ; int 80h is the assembly language op code for interrupt 80h ((INT
14 mov eax,1
15 mov ebx,0
16 int 80h
```

2. Compilation



```
ali@ali-Latitude-E5450: ~/Documents/assembly
ali@ali-Latitude-E5450:~/Documents/assembly$ nasm -f elf64 hello.asm
ali@ali-Latitude-E5450:~/Documents/assembly$
```

### 3-4. Linking & Run the new executable:

A terminal window with a dark background and light-colored text. The window title is 'ali@ali-Latitude-E5450: ~/Documents/assembly'. The terminal shows the following commands and output:

```
ali@ali-Latitude-E5450:~/Documents/assembly$ nasm -f elf64 hello.asm
ali@ali-Latitude-E5450:~/Documents/assembly$ ld -s -o hello hello.o
ali@ali-Latitude-E5450:~/Documents/assembly$ ./hello
Hello world!
ali@ali-Latitude-E5450:~/Documents/assembly$
```

### 5: int 80h

**int 80h** is basically interrupt call use in 32bit assembly language.

After loading the registers. **int 80h** interrupt call which Linux / Ubuntu understands. We'll load things up in the registers. What do we want to do? When an interrupt call occurs, the control goes to OS. The OS goes or reads the register or sees what the user wants to do and respective output generate according to the instruction of OS.