17 /EU22/043

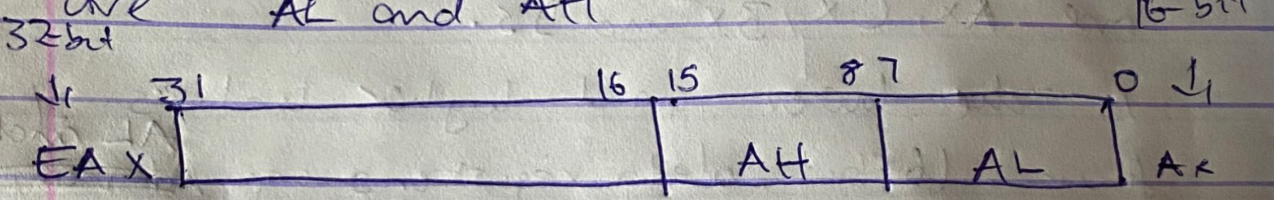
g) Using numeric tags you to a specific locations in memory. The addresses coded in the instructions would have to be updated whenever new variables were inserted before existing ones

5) Object files and listing files

17/ENG02/043 Ano

2b) No, They're not becaus it's not purtable

c) The 32-bit register EAX has a lower half AX that can be used as a 16bit data register and this 16-bit register has a lower and a higher halves which are AL and AH



a) Assembly language programs are not purtable betw various families of processors (they are processor dependent)

17/ENG02/043

3) b) Main Proc - This identifies the beginning of the code

MOV Ax, 47104 - This tells the program to move '47104' in to the register Ax

ADD EAx, 127Q - This tells the program to add 127 (in base 8) to the value already exesisting in EAX register

MOV DS, Ax - This tells the program to move the value in AX in to Ds

Main ENDP - This is the exit statement that calls a predefined MS-Windows function that halts the program

c) Value 1 Byte 0Dh - This tells the system to store the byte 0Dh under Value 1 label (it is an unsigned byte)

Value 1 = Label
Byte = Directive
0Dh = initializer
0Dh → radix

d) Value 2 DWORD ? - This is an uninitialized variable and it's value will be assigned at runtime

17/EG/GO2/0443

iii) Values SBYTE -10, -20, -30, -40, -50

These are multiple initializers (Signed variables). Here, one label is used to declare multiple Signed Variables

a) Segmented memory model divides the system ~~memory~~ memory into groups of independent segments represented by pointers located in ~~Code~~ the segment registers. Each segement is used to contain specific data types

① Data Segment (.data) identifies the area of a program containing variables

② Code Segment (.code) identifies the area of a program containing executable instructions

③ The Stack Segment (.stack) identifies the area of a program holding the runtime stack, setting it's size

17/ENG02/043

4) TITLE Subtraction (Sub.asm)

; This program subtracts 3 16-bit integers

INCLUDE Irvine32.inc

.data

val1 WORD 5000h

val2 WORD 2000h

val3 WORD 1000h

final val WORD ?

.Code

main PROC

mov ax, val1

sub ax, val2

sub ax, val3

mov final val, ax

DumpRegs

exit

main ENDP

END main