

23K-0021

Kirsh Kumar

BSCS-3J

Question: 02

a) High-level Language:-

High level languages are closer to human languages so they are easier to learn, read & write.

- High-level languages are machine independent so code written in them can run across different hardware without much changes.
- it makes use of english statements for operation.

Assembly Language

It is a low level programming language just above machine code so it is difficult to learn.

- It is a machine ^{dependent} language as it runs on specific processor architecture (eg x86).
- It uses mnemonics to perform operations.

Similarities

- Both HLL & assembly need to be compiled or interpreted into machine code for computer to understand
- Both uses symbolic representations to refer to memory locations, data etc

Relationship:

- High level language has a one to many relationship with assembly language.
- As a single high-level language instruction expands into multiple assembly language instructions.

b)

Assemblers	Compilers
→ Translates assembly code into machine code	→ Translates high level language code into machine or assembly code
→ performs one to one translations	→ performs complex optimization, error checking, and code analysis
→ Debugging is difficult in assemblers	→ Debugging is easier as compared to assembler.

c)

• High-level languages are more portable b/c they are machine independent. The same code can run on different hardware.

A C++ code is written on Linux, it can run on a windows machine if it has the C++ compiler.

• whereas the assembly code is machine dependent, if the code is written on x86 architecture, it will not run on ARM without much changes.

Question: 02

• Virtual machine is a software program that emulates the function of some other physical or virtual computer.

• The level-0 or VM(0) is the actual hardware or machine language of the microcontroller arduino.

• The level-1 or VM(1) is the assembly language used to program the microcontroller which is generated by compiling C-code into assembly.

① compilation occurs at VM(1) which is level 2 - assembly language

② Translation occurs at VM(0) which is hardware level.

Question: 03

a) Roll No: 23K-0021

MOV EAX, 0F0F0F2Fh ; EAX = F0F0F2F2
ADD EAX, 10000100h

① converting both values in Binary

F0F0F2F2h = 1111 0000 1111 0000 1111 0010 1111 0001
10000100h = 0001 0000 0000 0000 0000 0001 0000 0000
+

100000000 1111 0000 1111 0011 1111 0001
Carry

CF=1, ZF=0, SF=0
AF=0, PF=0, OF=0

b) x86 processor can access 1MB ram at a time because it uses 20bit address ranging from 0 to FFFF to access the memory

c)

12AB:025F

physical address = (Segment \times 10) + offset

$$= (12AB \times 10) + 025F$$

$$= 12AB0 + 025F$$

$$\begin{array}{r} 12AB0 \\ + 025F \\ \hline \boxed{12D0F} \end{array}$$

d) As the ~~x86~~ 8086 processor is a 16bit architecture it uses 16bit registers for efficiency and simplicity.

- A 16 bit segment value is placed in segment register the CPU automatically converts a 16 bit segment and a 16 bit offset value into a 20bit linear address

Question: 04

Include Irvine32.inc
data

~~Sunday~~

Monday = 0

Tuesday = 1

Wednesday = 2

Thursday = 3

Friday = 4

Saturday = 5

Sunday = 6

120ys Array DWORD Monday, Tuesday, Wednesday, Thursday,
Friday, Saturday, Sunday

- code

main PROC

exit

main ENDP

END main

Question: 05

Include Irvine32.inc

.data

Var1 DWORD 5 DUP(?)

Var2 BYTE 2 DUP(?)

Var3 BYTE 15 DUP('2')

Var4 BYTE 7 DUP('1')

Var5 BYTE 1 DUP('M')

.code

main PROC

exit

~~end MAIN~~

main ENDP

END main

Question: 06

c)

mov al, 88h

add al, 90h

Binary

88h = 1000 1000

90h = 1001 0000

1 0001 1000

al = 18h, OF = 1, CF = 1

ii)

mov al, 5

add al, 123

8

5d:

^{4 3 2 1} 0000 ^{3 2 1} 0101

0111 1011

1000 0000

5

123

128

al = 80h, SF = 1, OF = 1, CF = 0

Question: 02

- 1) $ecx = \text{dwList}$; it would move 2 bytes from
start due to DWORD label
 $ecx = 20001000$
- 2) $ebx = [\text{dwList} + 1]$; it would take an offset
 $ebx = 00200010$ of 1 byte which
are 00 from 3000 and
store 2 bytes.
- 3) $ecx = [\text{dwList} + 2]$, it would take an offset
of 2 bytes which is 3000
 $ecx = 30002000$ and store 2 bytes.
- 4) $edx = [\text{dwList} + 3]$, it would take an offset
 $edx = 11300020$ of 3 bytes.