23k-2001
BCS-3J
Date ____ 20__

COAL: Assignment 01

Answer #1:

a.

High level languages	Assembly language
• Closer resemblance to human natural language	• Closer resemblance to native machine language.
• A code-line is called as 'statement'.	• A code-line is called as 'instruction'.
• Multiply operations can be achieved in one statement.	• Only one operation can be performed by an instruction.
• Single statement corresponds to multiple instructions. (one-to-many)	• Each instruction corresponds to a single machine-language instruction. (one-to-one)
• High-level languages have data types such as int, char, float.	• Assembly language has no data type, rather data sizes such as byte, word, dword.
• High-level languages are portable and often system independent.	• Assembly languages are designed for a specific architecture, hence they are not portable.

Relationship and similarities:

→ Most high-level languages are compiled to a lower-level (assembly language) before being translated to machine code.

→ Assembly language is used in parallel with high-level languages in applications to optimize or target specific hardware.

→ All programming concepts such as conditions, selection, iteration & recursion are facilitated by both type of languages.

→ High-level languages are useful for faster development by abstracting hardware details, whereas assembly language optimizes performance by direct control over hardware.

23K-2001
BCS-3J

Date ____ 20__

b. Assemblers:

- Assembler translates assembly language directly into machine code, ready for execution.
- Translates instructions in correspondence to hardware-details as assembly language is closely tied to the cpu's architecture.
- Error-handling is limited, and code has to be proof read before-hand.

Compilers:

- Compilers translate high-level languages into a lower level (often assembly) which is passed on to be converted to machine language.
- Abstract details at the high level are translated to identify task that require further interpretation to operate on hardware.
- Compilers are highly optimized for error-checking and handling. e.g: syntax errors

c. Portability of HLLs as compared to Assembly:

High-level languages are regarded as more portable as they are designed to be hardware-independent. The compilers for HLLs abstract away the hardware details and make code ready for execution on any machine whereas Assembly language is directly assembled into machine code that is specific to the architecture it was written for.

→ E.g: A high-level language program such as Java can be executed on any system with JVM irrespective of the hardware.

→ E.g: A program written in C/C++ does not need to be rewritten for each platform. It can be compiled and run by any C++ compiler available.

An assembly language code written for a specific hardware cannot execute on another architecture without modification. e.g: x86, irvine32.

23k-2001
BCS-3J
Date ____ 20__

Answer #2:

Virtual Machine:

A software program that can emulate the functionality of another physical or virtual computer is known as a Virtual Machine.

According to the given example:

Level-3 : High-level language ↘ compilation] VM#1
Level-2 : Assembly language ↙ translation
Level-1 : Instruction-set computer architecture] VM#0
Level-0 : Digital logic ↙ execute instructions

- HLL is compiled to Assembly langvag at VM#1.
- Then, translated to ISA based on VM#0 system and executed at level-0 which is the hardware level on VM#0.

Answer #3:

a. MOV EAX, F2F0F0F1h
ADD EAX, 10000100h

Sol:

F2F0F0F1 ⇒ 1111 0010 1111 0000 1111 0000 1111 0001
+10000100 + 0001 0000 0000 0000 0000 0001 0000 0000
0001 0000 0010 1111 0000 1111 0001 1111 0001
⇒ 02F0F1F1h

CF : 1
OF : 0
ZF : 0
SF : 0
PF : 0 Ans.
AF : 0

b. Real Address Modes:

8086 processors operating in real-address mode can address 20-bit addresses following the range 0 to FFFFF (hex), which is why it can only access 1MB of RAM at a time.

c. Physical Address:

1 2AB : 025F

Linear physical address = segment x 10h + offset

$$\Rightarrow \begin{array}{r} 12AB \\ \times\ 10 \\ \hline 12^{1}AB0h \\ +\ 025F \\ \hline 12D0F \end{array}$$ Ans.

Physical address = 12D0Fh

d. 8086 is a 16-bit architecture that utilizes 16-bit registers for efficient usage. This is due to the fact that 16-bit data transfers were sufficient for most general-purpose tasks, and 20-bit address bus allow it to access 1MB at a time providing a balance between memory readability and transfer efficiency. The CPU is capable to store the 20-bit linear address based on the formula above.

23K-2001
BCS - 3J

Date ____ 20__

Answer #4

```
TITLE Question 4
INCLUDE Irvine32.inc
.data
    sunday = 0
    monday = 1
    tuesday = 2
    wednesday = 3
    thursday = 4
    friday = 5
    saturday = 6
    week BYTE sunday, monday, tuesday, wednesday, thursday, friday, saturday

.code
main PROC
    call DumRegs
    exit
main ENDP
END main
```

Answer #5

```
.data
    varA DWORD 5 DUP(?)
    varB BYTE 2 DUP(?)
    varC BYTE 15 DUP('&')
    varD BYTE 7 DUP('%')
    varE BYTE 1 DUP('M')
```

23K-2
BCS-3J

Date ____ 20

Answer #6:

i) mov al, 88h
add al, 90h

Sol:

```
  88h          1000      1000
+ 90h    ⇒   + 1001      0000
1  18h       0001 0001   1000
                ⇒ 18h
```

Ans: al = 18h
OF = 1 (result of the arithmetic too large to fit in destination)
SF = 0
CF = 1

ii) mov al, 5
add al, 123

Sol:

```
  5          '0000 '0101
+123   ⇒   + 0111 1011
128          1000 0000
               ⇒ 80h
```

Ans. al = 80
OF = 1 (MSB = 1 implies a sign change which also incurs an overflow condition as 128 is now being interpreted as a -ve number while the actual answer should have been +128)
SF = 1
CF = 0

23K-2001
BCS-3J

Date ____ 20 ____

Answer # 7:

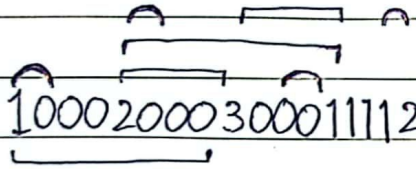
A) EAX = 20001000h — dwlist is of type DWORD, hence first 4 bytes will be stored

B) EBX = 00200010h — offset +1(byte), moves forward storing 00h from 3000h & the previous 3 bytes in order

C) ECX = 30002000h — offset +2 (bytes) stores 4 bytes ahead of the offset (1000h)

D) EDX = 11300020h — offset+3 (bytes), moves forward storing 11h from lower half of 1111h and the previous 3 bytes in order.

Memory Allocation for .data segment:



10002000300011112222 3333

(Little Endian representation)

0001 0002 0003 1111 2222 3333