23K0671 BCS 3J

A 1 → COAL.

Date:

Q.1 a). HighLevel languages and Assembly Language are both human readable. Both are such levels which cannot be directly interpreted by the hardware.

- Highlevel language has a one to many relationship with the Assembly language.

b) Assemblers, translate the assembly language code into machine code, which can directly be decoded by the architecture. Contrarily, Compilers breakdown highlevel code into assembly code, which needs to be further translated into machine code.

c) Since, assembly code is transformed into machine code, and the machine code is a different environment for different physical architecture, assembly code cannot be definetly run on different dirices, meaning ~~if~~ that an assembly code written for intel achitecture won't run on amd.

Examples :-

- Java programs run on many devices while assembly code doesn't.

- Highlevel languages are portable since the different types of compilers can be used on ~~a~~ any device and can safely and accurately change highlevel into assembly code.

Date:

Q.2

- Virtual Machines are a concept, in which a virtualized form of architecture is layed out
- Vm(0) refers to Level 0 of the architecture, which is the native machin language, such as machine code.
- Vm(1) refer to level 1 of the architecture, which is the better and closer to human environment, such as assembly or highlevel.

→ Level 3 is Vm(1)
→ Level 2 is Vm(1) — compilation
→ level 1 is Vm(0) — translation
→ Level 0 is Vm(0) — decoding.

Q.3 a) F2F4F0F4 = 1111 0010 1111 0100 1111 0000 1111 0100

+ 0001 0000 0000 0000 0000 0001 0000 0000

~~Numerical~~ → $\boxed{0001}$ 0000 0010 1111 0100 1111 0001 1111 0100

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

b) The 8086 can generate 20 bit addresses, from 00000h to FFFFFh. The total storage turns out to be 2^20 = 1MB of addressable memory.

Date:

c) (12ABh) × 10h + 025Fh

=> 12AB0h + 025Fh.

=> 12AB0
+ 025F
12D0Fh

=> $\boxed{12D0Fh}$ physical address.

d) Despite 20bit addresses, the 8086 architecture is a 16 bit processor system, ~~bcoz~~ the registers and instruction pointers are in 16bits. Hence to make instructions less complicated the addresses are represented in 16 bits.

Q.4

- data

Monday = "monday", 0
Tuesday = "tuesday", 0
Wednesday = "Wednesday", 0
Thursday = "thrsday", 0
Friday = "friday", 0
Saturday = "saturday", 0
Sunday = "Sunday", 0

- ~~array~~ array BYTE Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday

Date:

Q.5

```
.data
    v1  DWORD  5  DUP(?)
    v2  BYTE   2  DUP(?)
    v3  BYTE   15 DUP('&')
    v4  BYTE   7  DUP('%')
    v5  BYTE   1  DUP('M')
```

Q.6

i) 88h = 1000 1000
90h + 1001 0000
1 0001 1000
1 8 h

=> al = 18h , OF = 1 , CF = 1 (larger result and carry generated)

ii) 5 = 0000 0101
123 = + 0111 1011
1000 0000
8 0 h

=> al = 80h , OF = 0 , CF = 0 , SF = 1. (sign bit changed).

Date:

Q.7

- Error :- Syntax has no issues and the code will run.
But the use of singular increments to the table address causes disalignment and unintended use.

=> step 1 => eax = dwlist (takes 2 bytes starting from the start).
=> eax = 20001000 h
↳ because DWORD table.

=> step 2 => ebx = [dwlist + 1] (an offset of 1 byte ~~start~~ and store 2 bytes)
=> ebx = 00200010
lower zeros from 3000h

=> step 3 => ecx = [dwlist + 2] (an offset of 2 byts and store 2 bytes)
cxc = 30002000

=> similarly step 4 = edx = 11300020
lower from 1111h.

=> eax = 20001000 h
ebx = 00200010 h
ecx = 30002000 h
edx = 11300020 h.