1. A software interrupt is a call to an operating system procedure. Mos of these procedures provide input-output capability to application programs. An example of a software interrupt is the INT 10h video services. A hardware interrupt is an electronic alerting signal sent to the processor from an external device. An example of a hardware interrupt is the interrupt that is generated by the PIC which signals the CPU to suspend execution of the current program. A maskable interrupt is a hardware interrupt that can be disabled or ignored by the instructions of CPU. A non-maskable interrupt is a hardware interrupt that cannot be ignored by the CPU. An example of a maskable interrupt is RST6.5, and an example of a non-maskable interrupt is trap of 8085.

2. The steps of interrupt vectoring are as follows:
    1. The operand of the INT instruction is multiplied by 4 to locate the matching interrupt vector table entry.
    2. The CPU pushes the flags and a 32 bit segment/offset return address on the stack, disables hardware interrupts, and executes a far call to the address stored at location 10h * 4 in the interrupt vector table.
    3. The interrupt handler at F000:F065 executes until it reaches an IRET instruction.
    4. The IRET instruction pops the flags and the return address off the stack, causing the processor to resume execution immediately following the INT 10h instruction in the calling program.

3. .data
   myString BYTE “asdfghjkl”
   .main
   string_len = ($  - myString)
   mov ax, string_len
   call WriteInt
   exit
   main ENDP
   END Main

   4. INCLUDE Irvine32.inc
   .data
   ALIGN WORD
   ALLPoints COORD NumPoints DUP (<0, 0, 0>)
   .code
   main PROC
   mov edi, 0
   mov ecx, NumPoints
   mov ax, 1
   L1: mov (COORD PTR ALLPoints[edi]).X, ax
   mov (COORD PTR ALLPoints[edi]).Y, ax

```
        mov (COORD PTR ALLPoints[edi]).Z, ax
        add edi, TYPE COORD
        inc ax
        loop L1
       exit
main ENDP
END MAIN
```

5. RISC makes hardware simpler by using an instruction set composed of a few basic steps for loading, evaluating, and storing operations just like a load command will load data, store command will store the data. In CISC, a single instruction will do all loading, evaluating and storing operations just like a multiplication command will do stuff like loading data, evaluating and storing it.