Q1

How the physical address is calculated and find it if segment address is 4000H and offset address is 3000H. Why are these two addresses required in 8086?

PHY ADD = SEG ADD * 10H + OFFSET ADD
PHY ADD = 43000H
1MB of memory - interfaced with 8086 - requires 20-bit address lines.
8086 has16-bit registers – 20-bit address is generated from two 16-bit address (seg address and offset address)

Find the errors in the following instructions (if so).
I. **MOV DS,5000H** ---- MOV DS, AX
II. **ADD [AX], [4000H]** ---- ADD AX, [4000H]
III. **ADD 0200H** ---- (or) ADD AX 0200H
IV. **INC 3000H** ---- INC [3000H]
V. **CMP BX** ---- CMP AX, BX

Q2

Write an 8086 ALP programs to find the average of 64 numbers stored in the memory location 5000H with and without using DIV instruction. Assume that the sum does not exceed 16 bits.

i) WITH DIV

```
ASSUME CS:CODE, DS:DATA

DATA SEGMENT

   NUMLIST DB 45H, 78H, 97H, 67H, 89H, 78H, 57H, 79H

   NEXT DB 38H  DUP (00H)  // REMAINING VALUES WILL BE ZEROS

   COUNT EQU 40H

   RESULT DW 01H DUP (?)

DATA ENDS


CODE SEGMENT

   START:

   MOV AX, DATA      ; get the segment address of DATA segment in DS register

   MOV DS, AX

   MOV CX, COUNT     ; COUNT is the number of data has to be added

   MOV SI, OFFSET NUMLIST    ; get the offset address of NUMLIST

   MOV AX, 0000H
```

```
    MOV BX, 0000H
LP: MOV BL, [SI]
    ADD AX, BX
    INC SI
    DEC CX
    JNZ LP
    MOV BX, 0040H
    DIV BX
    MOV DI, OFFSET RESULT   ; get the offset address of RESULT in DI register
    MOV [DI], AX
    MOV AH, 00H
    INT 21H
    CODE ENDS
    END START

RET
```

ii) WITHOUT DIV

```
ASSUME CS:CODE, DS:DATA

DATA SEGMENT

    NUMLIST DB 45H, 78H, 97H, 67H, 89H, 78H, 57H, 79H

    NEXT DB 38H  DUP (00H)  // REMAINING VALUES WILL BE ZEROS

    COUNT EQU 40H

    RESULT DW 01H DUP (?)
```

```
DATA ENDS

CODE SEGMENT

    START:

    MOV AX, DATA      ; get the segment address of DATA segment in DS register

    MOV DS, AX

    MOV CX, COUNT     ; COUNT is the number of data has to be added

    MOV SI, OFFSET NUMLIST    ; get the offset address of NUMLIST

    MOV AX, 0000H

    MOV BX, 0000H

LP: MOV BL, [SI]

    ADD AX, BX

    INC SI

    DEC CX

    JNZ LP

    MOV CL, 06H
```

```
SHR AX, CL

MOV DI, OFFSET RESULT   ; get the offset address of RESULT in DI register

MOV [DI], AX

MOV AH, 00H

INT 21H

CODE ENDS

END START
```

```
RET
```

Q3

Write an 8051 ALP to find (compute) the numbers which are divisible by both 3 and 7 in the range 1 to 99 and store them in memory. Interface two seven segment displays in P0 and P1 and display the stored numbers one after other.

```
ORG 0000H
; 7 SEGMENT LOOKUP TABLE
MOV 30H, #03FH
MOV 31H, #06H
MOV 32H, #5BH
MOV 33H, #04FH
MOV 34H, #066H
MOV 35H, #06DH
MOV 36H, #07DH
MOV 37H, #07FH
```

```
MOV 38H, #07FH
MOV 39H, #067H

MOV R0, #20H
MOV R2, #00H
MOV R3, #01H
LOOP:MOV A, R3
MOV B, #15H
DIV AB
MOV A, B
CJNE A, #00H,J1
MOV A, R3
MOV @R0, A
INC R0
INC R2
J1:INC R3
CJNE R3, #64H, LOOP


MOV R0, #20H
BACK: MOV A, @R0
MOV B, #100D
DIV AB
MOV A, B
MOV B, #10D
DIV AB
ADD A, #30H
MOV R1, A
MOV A, @R1
MOV P0, A

MOV A, B
```

```
ADD A, #30H
MOV R1, A
MOV A, @R1
MOV P1, A
INC R0
ACALL DELAY
DJNZ R2, BACK


ORG 100H
DELAY:
MOV R5,#255
H3: MOV R4,#255
H2: MOV R3,#255
H1: DJNZ R3,H1
DJNZ R4,H2
DJNZ R5,H3
RET
END
```

Q4

Write an 8051 (AT89C51) ALP to generate a pulse waveform with frequency 2KHz using timer. Don't access TL0/1 register in the ALP. Provide the necessary timer calculations.

Time period = 1/(2KHz) = 0.5ms

ON period = OFF period = 0.25ms

Number of count = 0.25ms/1.085us = 230.41 = 230

TH0 = 255 – 230 = 25 or 19H

```
ORG 0000H
LJMP MAIN ;by-pass interrupt vector table
ORG 000BH ;Timer 0 interrupt vector table
CPL P2.1 ;toggle P2.1 pin
```

```
RETI

ORG 0050H ;after vector table space
MAIN: MOV TMOD,#02H ;Timer 0, mode 2
MOV P0,#0FFH ;make P0 an input port
MOV TH0,#019H ;TH0=A4H for -92
MOV IE,#82H ;IE=10000010 (bin) enable
;Timer 0
SETB TR0 ;Start Timer 0
BACK: SJMP BACK ;get data from P0

END
```

Q5

Write 8051 (AT89C51) ALP for the following logic. The existing values of TCON=01H and IE = 0FFH.

```
ORG 0000H
LJMP MAIN ;by-pass interrupt
;vector table
;--ISR for INT1 to turn on LED
ORG 0003H ;INT0 ISR
SETB P1.3 ;turn on LED
MOV R3,#255
BACK: DJNZ R3,BACK ;keep LED on for a while
CLR P1.3 ;turn off the LED
MOV R3,#255
BACK1: DJNZ R3,BACK1
RETI ;return from ISR
;--MAIN program for initialization
ORG 30H
MAIN: MOV IE,#0FFH
```

```
MOV TCON, 01H;enable external INT 1
HERE: SETB P1.3 ;turn on LED
MOV R3,#255
BACK4: DJNZ R3,BACK4 ;keep LED on for a while
CLR P1.3 ;turn off the LED
MOV R3,#255
BACK3: DJNZ R3,BACK3
MOV R3,#255
BACK2: DJNZ R3,BACK2;stay here until get interrupted
SJMP HERE
END
```