Task 1

```
;Program for Tutorial 3, Task 1
;Determine the addressing mode(s) for each of the following instructions:
.model small
.stack 100h

.code
main proc
     mov ax,8           ;immediate
     push ax            ;register
     call Factorial     ;direct
     mov ax,4C00h       ;immediate
     int 21h            ;immediate/direct?
main endp

Factorial proc
     push bp            ;register
     mov bp,sp          ;register
     mov ax,[bp+4]      ;indirect, based-relative
     cmp ax,1           ;immediate
     ja L1              ;direct
     mov ax,1           ;immediate
     jmp L2             ;direct
L1:  dec ax             ;register
     push ax            ;register
     call Factorial     ;direct
     mov bx,[bp+4]      ;indirect, based-relative
     mul bx             ;register
L2:  pop bp             ;register
     ret 2              ;immediate
Factorial endp

end
```

Task 2

```
1785:1234 = 17850+1234 = 18A84
FE01:5678 = FE010+5678 = 103688 (exceed 20-bit, i.e. wrapping)
1FA0:9ABC = 1FA00+9ABC = 294BC
0034:DEF0 = 00340+DEF0 = 0E230
5000:6789 = 50000+6789 = 56789
A000:2341 = A0000+2341 = A2341
B000:1A3C = B0000+1A3C = B1A3C
294B:000C = 294B0+000C = 294BC (overlapping with 1FA0:9ABC)
0123:4000 = 01230+4000 = 05230
FFFF:FFFF = FFFF0+FFFF = 10FFEF (exceed 20-bit, i.e. wrapping)
```

Task 3

What is the size of the address bus for Intel 8086?
Ans: 20-bit

What is the highest memory address accessible by Intel 8086?
Ans: FFFFFh

Are all the above physical addresses accessible by Intel 8086?
Ans: FE01:5678 and FFFF:FFFF are not accessible

Can you identify overlapping and wrapping?
Ans:
Overlapping: 1FA0:9ABC and 294B:000C points to same physical address

Wrapping: FE01:5678 and FFFF:FFFF exceeds 20-bit and will be wrapped to physical addresses 03688 and 0FFEF respectively

Task 4

```
;Program for Tutorial 3, Task 4
;Determine the effective address (logical) for each of the following instructions,
; except int 20h and ret:
.model small
.stack 100h

.code
; assuming DS=17D3, CS=17E3
0000 BE 0000 R    main:  mov si, offset num   ;DS:0000
0003 8B 04               mov ax,[si]          ;DS:0000
0005 8B 5C 02            mov bx,[si+2]        ;DS:0002
0008 E8 000D R           call proc1           ;CS:000D
                         ;display sum
000B CD 20               int 20h

000D BF 0004 R    proc1: mov di, offset sum   ;DS:0004
0010 C7 05 0000          mov word ptr [di],0  ;DS:0004
0014 01 05               add [di],ax          ;DS:0004
0016 01 1D               add [di],bx          ;DS:0004
0018 81 3D FF00          cmp [di],0ff00h      ;DS:0004
001C 74 03               je here1             ;CS:0021
001E C2 0001             ret 1
0021 C3           here1: ret 0

.data
0000 1122 3344    num dw 1122h,3344h
0004 0000         sum dw ?

End
```

Task 5

```
;Program for Tutorial 3, Task 5
;Predict the content of the affected memory/register(s)/ flag(s) after execution
; of each of the following instructions in sequence, except the int 20h (all
addresses in hex):
.model small
.stack 100h

.code
; assuming num is at DS:000A, proc1 is at CS:000D, here1 at CS:0028
main:  mov si, offset num          ;si = 000A
       mov ax,[si]                 ;ax = (000A) = (num) = 1122
       mov bx,[si+2]               ;bx = (000C) = (num+2) = 3344
       call proc1                  ;(ip) -> STACK, sp = (sp)-2, ip = offset proc1
       ;display sum
       int 20h

proc1: mov di, offset sum          ;di = 000E
       mov word ptr [di],0         ;(000E) = 0
       add [di],ax                 ;(000F)(000E) = (ah)(al) = (ax) = 11, 22
                                   ;FLAGS: NV PL NZ NA PE NC
                                   ;note [di] points to DS:000E and with 16-bit
                                   ;operation, it points to (DS:000F):(DS:000E)
       add [di],bx                 ;(000F)(000E) = 11 22 + (bx) = 1122 + 3344
                                   ;= 4466; FLAGS: NV PL NZ NA PE NC
       cmp [di],0ff00h             ;(000F)(000E) - ff00 = 4466 - ff00
                                   ;= 4466 + 0100 = 4566; FLAGS: NV PL NZ NA PO NC
                                   ;note ff00 is -0100, therefore 4466 -ff00
                                   ;= 4466 -(-0100)
       je here1                    ;since above condition is not equal,
                                   ;program proceed to next instruction as usual,
                                   ;i.e. no jump
       mov cx,[di]                 ;cx = (000F)(000E) = (sum) = 4566
       and ch,0ffh                 ;ch = (ch) AND ff = 45 AND ff = 45
                                   ;FLAGS: PL NZ PO
       not cl                      ;cl = NOT (cl) = NOT (45) = invert the bits
                                   ;= ba; FLAGS: NG NZ PO
       ret 0 ;ip <- STACK, sp = (sp)+2
here1: ret 0

.data
num dw 1122h,3344h
sum dw ?

End
```