8086 Microprocessor Experiments: Code and Flowcharts

Q1: 16-Bit Addition

Assembly Code

```
data segment
n1 dw 1234h
n2 dw 6578h
sum dw ?
data ends
code segment
assume cs:code, ds:data
start:
  mov ax,data
  mov ds,ax
  mov ax,n1
  mov bx,n2
  add ax,bx
  mov sum,ax
  mov ah,4ch
  int 21h
code ends
end start
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+-------------------------+
| Initialize DS           |
+----+--------------------+
     |
     v
+-------------------------+
| Load AX=n1, BX=n2       |
+----+--------------------+
     |
     v
+-------------------------+
| AX = AX + BX            |
+----+--------------------+
     |
     v
+-------------------------+
| Store SUM = AX         |
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q1: 16-Bit Subtraction

Assembly Code

```
data segment
n1 dw 5678h
n2 dw 1234h
diff dw ?
data ends
code segment
assume cs:code, ds:data
start:
  mov ax,data
  mov ds,ax
  mov ax,n1
  mov bx,n2
  sub ax,bx
  mov diff,ax
  mov ah,4ch
  int 21h
code ends
end start
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS          |
+----+-------------------+
     |
     v
+------------------------+
| Load AX=n1, BX=n2      |
+----+-------------------+
     |
     v
+------------------------+
| AX = AX - BX           |
+----+-------------------+
     |
     v
+------------------------+
| Store DIFF = AX        |
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q2: 32-Bit Addition

Assembly Code

```
data segment
no1lw dw 5678h
no1hw dw 1234h
no2lw dw 2253h
no2hw dw 5678h

sumlwlb db ?
sumlwhb db ?
sumhwlb db ?
sumhwhb db ?
data ends

code segment
assume cs:code, ds:data
start:
    mov ax, data
    mov ds, ax
    mov bx, no1lw
    mov ax, no1hw
    mov dx, no2lw
    mov cx, no2hw
    add bx, dx
    mov sumlwlb, bl
    mov sumlwhb, bh
    adc ax, cx
    mov sumhwlb, al
    mov sumhwhb, ah
    mov ah, 4Ch
    int 21h
code ends
end start
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS          |
+----+-------------------+
     |
     v
+------------------------+
| Load Operands:         |
| BX=no1lw, AX=no1hw     |
| DX=no2lw, CX=no2hw     |
+----+-------------------+
     |
     v
+------------------------+
| 1. ADD BX, DX (Low Words) |
+----+-------------------+
     |
     v
+------------------------+
| 2. ADC AX, CX          |
| (High Words + Carry)   |
+----+-------------------+
     |
     v
+------------------------+
| Store Results in       |
| SUMLW, SUMHW (by bytes)|
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q2: 32-Bit Subtraction

Assembly Code

```
data segment
no1lw dw 5678h
no1hw dw 1234h
no2lw dw 2253h
no2hw dw 5678h
difflwlb db ?
difflwhb db ?
```

```
diffhwlb db ?
diffhwhb db ?
data ends

code segment
assume cs:code, ds:data
start:
    mov ax, data
    mov ds, ax
    mov bx, no1lw
    mov ax, no1hw
    mov dx, no2lw
    mov cx, no2hw
    sub bx, dx
    mov difflwlb, bl
    mov difflwhb, bh
    sbb ax, cx
    mov diffhwlb, al
    mov diffhwhb, ah
    mov ah, 4Ch
    int 21h
code ends
end start
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS          |
+----+-------------------+
     |
     v
+------------------------+
| Load Operands:         |
| BX=no1lw, AX=no1hw     |
| DX=no2lw, CX=no2hw     |
+----+-------------------+
     |
     v
+------------------------+
| 1. SUB BX, DX (Low Words) |
+----+-------------------+
     |
     v
+------------------------+
| 2. SBB AX, CX          |
| (High Words - Borrow)  |
+----+-------------------+
     |
     v
+------------------------+
| Store Results in       |
| DIFFLW, DIFFHW (by bytes)|
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q3: 16-Bit Multiplication

Assembly Code

```
data segment
n1 dw 1234h
n2 dw 5678h
prodlw dw ?
prodhw dw ?
data ends
code segment
assume cs:code, ds:data
start:
  mov ax,data
  mov ds,ax
  mov ax,n1
  mov bx,n2
  mul bx
  mov prodlw,ax
  mov prodhw,dx
  mov ah,4ch
  int 21h
code ends
end start
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS          |
+----+-------------------+
     |
     v
+------------------------+
| Load AX=n1, BX=n2      |
+----+-------------------+
     |
     v
```

```
+-------------------------+
| Perform MUL BX:         |
| DX:AX = AX * BX         |
+----+--------------------+
     |
     v
+-------------------------+
| Store PRODLW = AX       |
| Store PRODHW = DX       |
+----+--------------------+
     |
     v
+-------------------------+
| Terminate (INT 21H)     |
+----+--------------------+
     |
     v
+-----------+
|   END     |
+-----------+
```

Q4: 16-Bit Division

Assembly Code

```
data segment
dvr dw 1234h
dndlw dw 0063h
dndhw dw 0620h
quo dw ?
rem dw ?
data ends
code segment
assume cs:code, ds:data
start:
  mov ax,data
  mov ds,ax
  mov dx,dndhw
  mov ax,dndlw
  mov bx,dvr
  div bx
  mov quo,ax
  mov rem,dx
  mov ah,4ch
  int 21h
code ends
end start
```

Flowchart

```
+-----------+
|  START    |
+----+------+
     |
     v
+-----------------------+
| Initialize DS         |
+----+------------------+
     |
     v
+-----------------------+
| Load Dividend:        |
| DX=dndhw, AX=dndlw    |
| Load Divisor: BX=dvr  |
+----+------------------+
     |
     v
+-----------------------+
| Perform DIV BX:       |
| AX = Quotient         |
| DX = Remainder        |
+----+------------------+
     |
     v
+-----------------------+
| Store QUO = AX        |
| Store REM = DX        |
+----+------------------+
     |
     v
+-----------------------+
| Terminate (INT 21H)   |
+----+------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q5: Finding Largest Number from Block

Assembly Code

```
DATA SEGMENT
    array   DB 2H, 4H, 5H, 11H, 10H
    largest DB ?
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV CX, 04H
    LEA SI, array
    MOV AL, [SI]
AGAIN:
    INC SI
    CMP AL, [SI]
    JNC NEXT
    MOV AL, [SI]
NEXT:
    LOOP AGAIN

    MOV largest, AL

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS, CX=4    |
| LEA SI, array          |
| AL = [SI] (First element)|
+----+-------------------+
     |
   AGAIN:
     v
+------------------------+
| INC SI                 |
| (Point to next element)|
+----+-------------------+
     |
     v
+--------------------+
| Is AL < [SI]?     | (CMP AL, [SI])
| (Current Largest < Next)  |
+---------+---------+
|   YES   |    NO   |
+---------+---------+
     |         |
     v         v
+----------+   |
| MOV AL, [SI] |   |
| (Update AL)  |   |
+----------+   |
     |         |
   NEXT: <-----+
     v
+------------------------+
| LOOP AGAIN             |
| (Decrement CX, JNZ)    |
+---------+---------+
|   CX > 0|    CX = 0 |
+---------+---------+
     |         |
     v         v
   AGAIN <-----+
+-----------------------+
| Store largest, AL      |
+----+------------------+
     |
     v
+-----------------------+
| Terminate (INT 21H)    |
+----+------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q6: Finding Smallest Number from Block

Assembly Code

```
DATA SEGMENT
    array   DB 2H, 4H, 5H, 11H, 10H
    smallest DB ?
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV CX, 04H
    LEA SI, array
    MOV AL, [SI]
AGAIN:
    INC SI
    CMP AL, [SI]
    JBE NEXT
    MOV AL, [SI]
NEXT:
    LOOP AGAIN

    MOV smallest, AL

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS, CX=4    |
| LEA SI, array          |
| AL = [SI] (First element)|
+----+-------------------+
     |
   AGAIN:
     v
+------------------------+
| INC SI                 |
+----+-------------------+
     |
     v
+--------------------+
| Is AL > [SI]?     | (CMP AL, [SI])
| (Current Smallest > Next) |
+---------+---------+
|   YES   |    NO   |
+---------+---------+
     |         |
     v         v
+----------+   |
| MOV AL, [SI] |   |
| (Update AL)  |   |
+----------+   |
     |         |
   NEXT: <-----+
     v
+------------------------+
| LOOP AGAIN             |
+---------+---------+
|   CX > 0|    CX = 0 |
+---------+---------+
     |         |
     v         v
   AGAIN <-----+
+------------------------+
| Store smallest, AL     |
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q7(a): Packed to Unpacked BCD

Assembly Code

```
DATA SEGMENT
    PACKED  DB 45H
    UNPACK1 DB ?
    UNPACK2 DB ?
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV AL, PACKED
    MOV AH, AL
    AND AL, 0FH
    AND AH, 0F0H
    MOV CL, 04
    ROR AH, CL

    MOV UNPACK1, AH
    MOV UNPACK2, AL

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+-----------------------+
| Initialize DS         |
+----+------------------+
     |
     v
+-----------------------+
| Load AL=PACKED (45H)  |
| Copy AH = AL          |
+----+------------------+
     |
     v
+-----------------------+
| 1. Get Low Nibble:    |
|    AND AL, 0FH (AL=05H)|
+----+------------------+
     |
     v
+-----------------------+
| 2. Get High Nibble:   |
|    AND AH, 0F0H (AH=40H)|
|    ROR AH, 4 (AH=04H)  |
+----+-------------------+
     |
     v
+------------------------+
| Store UNPACK1=AH       |
| Store UNPACK2=AL       |
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q7(b): Unpacked to Packed BCD

Assembly Code

```
DATA SEGMENT
    UNPACK1 DB 04H
    UNPACK2 DB 05H
    PACKED  DB ?
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV AL, UNPACK1
    MOV CL, 04
    ROL AL, CL
    OR  AL, UNPACK2
    MOV PACKED, AL

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS          |
+----+-------------------+
     |
     v
+------------------------+
| Load AL=UNPACK1 (04H)  |
+----+-------------------+
     |
     v
+------------------------+
| Shift High Digit:      |
| ROL AL, 4 (AL=40H)     |
+----+-------------------+
     |
     v
+------------------------+
| Combine with Low Digit:|
| OR AL, UNPACK2 (AL=45H)|
+----+-------------------+
     |
     v
+------------------------+
| Store PACKED = AL      |
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q7(c): Packed BCD to ASCII

Assembly Code

```
DATA SEGMENT
    PACKED  DB 45H
    ASCII1  DB ?
    ASCII2  DB ?
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV AL, PACKED
    MOV AH, AL
    AND AL, 0FH
    AND AH, 0F0H
    MOV CL, 04
    ROR AH, CL

    ADD AH, 30H
    ADD AL, 30H

    MOV ASCII1, AH
    MOV ASCII2, AL

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+-------------------------+
| Unpack Digits (Q7a steps)|
| AH = High Digit (04H)  |
| AL = Low Digit (05H)   |
+----+--------------------+
     |
     v
+-------------------------+
| Convert High Digit:    |
| ADD AH, 30H (AH='4')   |
+----+--------------------+
     |
     v
+-------------------------+
| Convert Low Digit:     |
| ADD AL, 30H (AL='5')   |
+----+--------------------+
     |
     v
+-------------------------+
| Store ASCII1=AH        |
| Store ASCII2=AL        |
+----+--------------------+
     |
     v
+-------------------------+
| Terminate (INT 21H)    |
+----+--------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q8(A) / Q9: Block Copy With String Instructions

Assembly Code

```
DATA SEGMENT
    block1 DB 10H, 20H, 30H, 40H, 50H
    block2 DB 5 DUP(?)
    n      DB 05H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA, ES:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    MOV ES, AX

    MOV CL, n
    MOV CH, 00H

    LEA SI, block1
    LEA DI, block2

    CLD
    REP MOVSB

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+-------------------------+
| Initialize DS=ES        |
| CX = count              |
| LEA SI, source          |
| LEA DI, dest            |
+----+--------------------+
     |
     v
+-------------------------+
| Set Direction: CLD      |
| (Forward copy)          |
+----+--------------------+
     |
     v
+-------------------------+
| REP MOVSB               |
| (Copy CX bytes)         |
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q8(B) / Q10: Block Copy Without String Instructions

Assembly Code

```
DATA SEGMENT
    block1 DB 10H, 20H, 30H, 40H, 50H
    block2 DB 5 DUP(?)
    n      DB 05H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV CL, n
    MOV CH, 00H

    LEA SI, block1
    LEA DI, block2
COPY_LOOP:
    MOV AL, [SI]
    MOV [DI], AL
    INC SI
    INC DI
    LOOP COPY_LOOP

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS          |
| CX = count             |
| LEA SI, source         |
| LEA DI, dest           |
+----+-------------------+
     |
  COPY_LOOP:
     v
+------------------------+
| AL = [SI]              |
| [DI] = AL              |
+----+-------------------+
     |
     v
+------------------------+
| INC SI, INC DI         |
| (Increment pointers)   |
+----+-------------------+
     |
     v
+------------------------+
| LOOP COPY_LOOP         |
+---------+---------+
|   CX > 0|    CX = 0 |
+---------+---------+
     |         |
     v         v
 COPY_LOOP <-----+
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------+
```

Q11: Block Exchange

Assembly Code

```
DATA SEGMENT
    block1 DB 10H, 20H, 30H, 40H, 50H
    block2 DB 1H,  2H,  3H,  4H,  5H
    n      DB 05H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV CL, n
    MOV CH, 00H
    LEA SI, block1
    LEA DI, block2
EXCHANGE_LOOP:
    MOV AL, [SI]
    MOV BL, [DI]

    MOV [SI], BL
    MOV [DI], AL

    INC SI
    INC DI
    LOOP EXCHANGE_LOOP

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+-------------------------+
| Initialize DS           |
| CX = count              |
| LEA SI, block1          |
| LEA DI, block2          |
+----+--------------------+
     |
 EXCHANGE_LOOP:
     v
+-------------------------+
| AL = [SI] (From block1)|
| BL = [DI] (From block2)|
+----+--------------------+
     |
     v
+-------------------------+
| [SI] = BL (Store to B1)|
| [DI] = AL (Store to B2)|
+----+--------------------+
     |
     v
+-------------------------+
| INC SI, INC DI          |
+----+--------------------+
     |
     v
+-------------------------+
| LOOP EXCHANGE_LOOP      |
+---------+---------+
|   CX > 0|    CX = 0 |
+---------+---------+
     |         |
     v         v
EXCHANGE_LOOP <--+
+-----------------------+
| Terminate (INT 21H)    |
+----+------------------+
     |
     v
+----------+
|   END    |
+----------
```

Q12: Count Odd and Even Numbers

Assembly Code

```
DATA SEGMENT
    ARR  DB 10H, 21H, 32H, 43H, 54H, 65H, 76H, 87H
    N    DB 08H
    EVEN DB 00H
    ODD  DB 00H
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV CL, N
    MOV CH, 00H
    LEA SI, ARR

    MOV BL, 00H
    MOV BH, 00H
NEXT_NUM:
    MOV AL, [SI]
    TEST AL, 01H
    JZ EVEN_NUM
    INC BH
    JMP CONTINUE
EVEN_NUM:
    INC BL
CONTINUE:
    INC SI
    LOOP NEXT_NUM

    MOV EVEN, BL
    MOV ODD,  BH

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS          |
| CX = N (count)         |
| LEA SI, ARR            |
| BL=0 (Even), BH=0 (Odd)|
+----+-------------------+
     |
  NEXT_NUM:
     v
+------------------------+
| AL = [SI]              |
+----+-------------------+
     |
     v
+-------------------+
| Is AL Odd?        | (TEST AL, 01H / JZ EVEN_NUM)
| (LSB is 1)        |
+---------+---------+
|   NO    |    YES  |
+---------+---------+
 EVEN_NUM: |         |
     v         v
+----------+   +----------+
| INC BL   |   | INC BH   |
| (Even++) |   | (Odd++)  |
+----------+   +----------+
     |         |
     v         v
   <-----------+
   CONTINUE:
     v
+------------------------+
| INC SI                 |
+----+-------------------+
     |
     v
+------------------------+
| LOOP NEXT_NUM          |
+---------+---------+
|   CX > 0|    CX = 0 |
+---------+---------+
     |         |
     v         v
 NEXT_NUM <----+
+------------------------+
| Store EVEN=BL, ODD=BH  |
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+------------------+
     |
     v
+----------+
|   END    |
+----------
```

Q13: Arrange in Ascending Order (Bubble Sort)

Assembly Code

```
DATA SEGMENT
    arr DB 09h, 03h, 07h, 02h, 04h
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV CH, 4
ASC_PASS:
    MOV CL, CH
    LEA SI, arr
ASC_CMP:
    MOV AL, [SI]
    CMP AL, [SI+1]
    JC  ASC_NOSWAP
    XCHG AL, [SI+1]
    MOV [SI], AL
ASC_NOSWAP:
    INC SI
    DEC CL
    JNZ ASC_CMP

    DEC CH
    JNZ ASC_PASS

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+-----------------------+
| Initialize DS         |
| CH = N-1 (Outer Loop)  |
+----+-------------------+
     |
   ASC_PASS:
     v
+------------------------+
| CL = CH (Inner Count)  |
| LEA SI, arr            |
+----+-------------------+
     |
   ASC_CMP:
     v
+-------------------+
| Is AL < [SI+1]?   | (CMP AL, [SI+1] / JC ASC_NOSWAP)
| (Already in order)|
+---------+---------+
|   YES   |    NO   |
+---------+---------+
     |         v
     |    +-------------------+
     |    | Swap AL and [SI+1]|
     |    | [SI] = AL         |
     |    +-------------------+
     |         |
ASC_NOSWAP: <----+
     v
+------------------------+
| INC SI, DEC CL         |
+----+-------------------+
     |
     v
+-------------------+
| Is CL > 0?        | (JNZ ASC_CMP)
+---------+---------+
|   YES   |    NO   |
+---------+---------+
     |         v
 ASC_CMP <-----+
     |
     v
+------------------------+
| DEC CH                 |
+----+-------------------+
     |
     v
+-------------------+
| Is CH > 0?        | (JNZ ASC_PASS)
+---------+---------+
|   YES   |    NO   |
+---------+---------+
     |         v
 ASC_PASS <----+
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------
```

Q14: Arrange in Descending Order (Bubble Sort)

Assembly Code

```
DATA SEGMENT
    arr DB 09h, 03h, 07h, 02h, 04h
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV CH, 4
DESC_PASS:
    MOV CL, CH
    LEA SI, arr
DESC_CMP:
    MOV AL, [SI]
    CMP AL, [SI+1]
    JGE DESC_NOSWAP
    XCHG AL, [SI+1]
    MOV [SI], AL
DESC_NOSWAP:
    INC SI
    DEC CL
    JNZ DESC_CMP

    DEC CH
    JNZ DESC_PASS

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+-----------------------+
| Initialize DS         |
| CH = N-1 (Outer Loop) |
+----+------------------+
     |
  DESC_PASS:
     v
+------------------------+
| CL = CH (Inner Count)  |
| LEA SI, arr            |
+----+-------------------+
     |
   DESC_CMP:
     v
+-------------------+
| Is AL >= [SI+1]?  | (CMP AL, [SI+1] / JGE DESC_NOSWAP)
| (Already in order)|
+---------+---------+
|   YES   |    NO   |
+---------+---------+
     |         v
     |   +-------------------+
     |   | Swap AL and [SI+1]|
     |   | [SI] = AL         |
     |   +-------------------+
     |         |
DESC_NOSWAP: <---+
     v
+------------------------+
| INC SI, DEC CL         |
+----+-------------------+
     |
     v
+-------------------+
| Is CL > 0?        | (JNZ DESC_CMP)
+---------+---------+
|   YES   |    NO   |
+---------+---------+
     |         v
 DESC_CMP <----+
     |
     v
+------------------------+
| DEC CH                 |
+----+-------------------+
     |
     v
+-------------------+
| Is CH > 0?        | (JNZ DESC_PASS)
+---------+---------+
|   YES   |    NO   |
+---------+---------+
     |         v
 DESC_PASS <---+
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------
```

Q15: Display College Name 5 Times

Assembly Code

```
DATA SEGMENT
    msg DB 'Vidyalankar Institute of Technology$', 0Dh, 0Ah
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX

    MOV CX, 5
DISPLAY_LOOP:
    LEA DX, msg
    MOV AH, 09H
    INT 21H

    LOOP DISPLAY_LOOP

    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+------------------------+
| Initialize DS          |
| CX = 5 (Loop Count)    |
+----+-------------------+
     |
   DISPLAY_LOOP:
     v
+------------------------+
| Display String:        |
| LEA DX, msg            |
| INT 21H (AH=09H)       |
+----+-------------------+
     |
     v
+------------------------+
| LOOP DISPLAY_LOOP      |
+---------+---------+
|   CX > 0|    CX = 0 |
+---------+---------+
     |         |
     v         v
```

```
DISPLAY_LOOP <---+
+-----------------------+
| Terminate (INT 21H)    |
+----+------------------+
     |
     v
+----------+
|   END    |
+----------
```

Q16: Reverse User Entered String

Assembly Code

```
DATA SEGMENT
    msg1 DB 'Enter a String: $'
    msg2 DB 0Dh,0Ah,'Original String: $'
    msg3 DB 0Dh,0Ah,'Reversed String: $'
    str  DB 100 DUP('$')
    rev  DB 100 DUP('$')
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV AX, DATA
    MOV DS, AX
    LEA DX, msg1
    MOV AH, 09H
    INT 21H
    LEA DX, str
    MOV AH, 0AH
    INT 21H
    MOV CL, [str+1]
    MOV CH, 0
    LEA SI, str+2
    LEA DI, rev
    ADD SI, CX
    DEC SI
REV_LOOP:
    MOV AL, [SI]
    MOV [DI], AL
    INC DI
    DEC SI
    LOOP REV_LOOP
    MOV BYTE PTR [DI], '$'
    LEA DX, msg2
    MOV AH, 09H
    INT 21H

    LEA DX, str+2
    MOV AH, 09H
    INT 21H
    LEA DX, msg3
    MOV AH, 09H
    INT 21H
    LEA DX, rev
    MOV AH, 09H
    INT 21H
    MOV AH, 4CH
    INT 21H

CODE ENDS
END START
```

Flowchart

```
+----------+
|  START   |
+----+-----+
     |
     v
+-------------------------+
| Initialize DS           |
+----+--------------------+
     |
     v
+-------------------------+
| Display Prompt (msg1)   |
| Read String (INT 21H, AH=0AH)|
| Store in STR buffer     |
+----+--------------------+
     |
     v
+-------------------------+
| Initialize Pointers:    |
| CX = length of STR      |
| SI = last char address |
| DI = start of REV buffer |
+----+--------------------+
     |
   REV_LOOP:
     v
+-------------------------+
| AL = [SI] (Read char)  |
| [DI] = AL (Store char) |
+----+--------------------+
     |
     v
+-------------------------+
| INC DI, DEC SI          |
+----+--------------------+
     |
     v
+-------------------------+
| LOOP REV_LOOP           |
+---------+---------+
|   CX > 0|    CX = 0 |
+---------+---------+
     |         v
 REV_LOOP <----+
+-------------------------+
| Terminate REV string   |
| with '$'               |
+----+-------------------+
     |
     v
+------------------------+
| Display Original String|
| Display Reversed String|
+----+-------------------+
     |
     v
+------------------------+
| Terminate (INT 21H)    |
+----+-------------------+
     |
     v
+----------+
|   END    |
+----------
```

Q17: Check if String is Palindrome

Assembly Code

```
DATA SEGMENT
    Arr     DB 9 DUP(00h)
    Msg     DB 'Enter the string:$'
    Msg1    DB 'String is PALINDROME$'
    Msg2    DB 'String is NOT PALINDROME$'
DATA ENDS

CODE SEGMENT
ASSUME CS:CODE, DS:DATA
START:
    MOV     AX, DATA
    MOV     DS, AX

    MOV     CX, 0009H
    MOV     BX, 0000H

    LEA     DX, Msg
    MOV     AH, 09H
    INT     21H
NXT_CHR:
    MOV     AH, 01H
    INT     21H
    CMP     AL, 0DH
    JE      END_INPUT
    MOV     [BX], AL
    INC     BX
    LOOP    NXT_CHR
END_INPUT:
    DEC     BX
    MOV     SI, 0000H
    MOV     DI, BX
    MOV     CX, BX
    SHR     CX, 1

    CLD
NXT_CMP:
    MOV     AL, [SI]
    CMP     AL, [DI]
    JNE     NOT_PALIN
    INC     SI
    DEC     DI
    LOOP    NXT_CMP
PALIN:
    LEA     DX, Msg1
    MOV     AH, 09H
    INT     21H
    JMP     STOP
NOT_PALIN:
    LEA     DX, Msg2
    MOV     AH, 09H
    INT     21H
STOP:
    MOV     AH, 4CH
    INT     21H

CODE ENDS
END START
```

Flowchart

```
+-----------+
|  START    |
+----+------+
     |
     v
+-------------------------+
| Initialize DS           |
| Display Prompt (Msg)    |
+----+--------------------+
     |
     v
+-------------------------+
| Read String (Looping    |
| with INT 21H, AH=01H)   |
| Store in ARR until ENTER|
+----+--------------------+
     |
 END_INPUT:
     v
+-------------------------+
| Initialize Pointers/Cnt:|
| SI = start (0)          |
| DI = end (BX-1)         |
| CX = length / 2         |
+----+--------------------+
     |
   NXT_CMP:
     v
+-------------------------+
| AL = [SI]              |
| Compare AL with [DI]   |
+----+-------------------+
     |
     v
+-------------------+
| Is AL = [DI]?     | (JNE NOT_PALIN)
+---------+---------+
|   YES   |    NO   |
+---------+---------+
     |         v
     |   NOT_PALIN:
     v         v
+-----------------------+  +-----------------------+
| INC SI, DEC DI         |  | Display Msg2 (NOT PAL)|
+----+-------------------+  +----+-------------------+
     |                        |
     v                        v
+-----------------------+   STOP:
| LOOP NXT_CMP           |      |
+---------+---------+        v
|   CX > 0|    CX = 0 |      +-----------------------+
+---------+---------+      | Terminate (INT 21H)    |
     |         v           +----+-------------------+
     v         |                |
   NXT_CMP <---+                v
     |                    +----------+
 PALIN: <---------------------+   END    |
     v                    +----------+
+-----------------------+
| Display Msg1 (PALINDROME)|
+----+-------------------+
     |
     v
+-----------------------+
| JMP STOP               |
+----+-------------------+
```