NAME

CIS40/CNET220 EXERCIZE 4

1. Assemble the following code at the default segment, and an offset of 100

```
0100    PUSH   DS
0101    MOV    AX,0000
0104    PUSH   AX
0105    MOV    AH,02
0107    MOV    CX,0010
010A    ROL    DX,1
010C    PUSH   DX
010D    AND    DX,0001
0111    ADD    DX,+30
0114    INT    21
0116    POP    DX
0117    DEC    CX
0118    JNZ    010A
011A    RET
```

2. Save the program to the hard disk as c:\student\hextobin
3. Place ABCD into register DX
4. Type g for go (if you're feeling lucky) or g = 100 11A (if you're not sure what the odd's are)
5. Write out the resultant binary number _____

6. Modify the code to send the number out in the reverse order as input into DX register
    a. You need to change lines 010A instructions rotation type, and it relative position in the code sequence

7. Save the modification as c:\student\bitrev
8. Place ABCD into register DX
9. Type g for go
10. Write out the resultant binary number _____ (verify that it is the reversed from above)

Instructor Verification of Code and Operation _____

Hextobin Code Description
Note the DX register is initialized using the debug register command before running this code.

The First three lines save a reentry point for debug to display the message “program terminated normally” after the program finishes executing (after the RET statement at the end)

```
0100    PUSH  DS
0101    MOV   AX,0000
0104    PUSH  AX
```

This line initializes AH register with write to standard output command used on line 0114 Int 21 instruction

```
0105    MOV   AH,02
```

This line initializes register CX so that lines 117 and 118 will countdown/loop 16 decimal times

```
0107    MOV   CX,0010
```

ROL rotates the information in the DX register left 1 bit position, to align the next bit from the MSB into the LSB position. This allows information to be output MSB to LSB one bit at a time

```
010A    ROL   DX,1
```

This PUSH saves the rotated information in DX onto the stack since it will be modified the next step

```
010C    PUSH  DX
```

This AND operation, removes all bits from DX except for the LSB

```
010D    AND   DX,0001
```

ADDing 30 to DX converts the LSB to an ascii 1 or 0 so that it can be displayed

```
0111    ADD   DX,+30
```

INT 21 instruction uses the information in the DL, and AH registers to output to the monitor

```
0114    INT   21
```

POP DX removes the saved DX information from the stack (see line 010C) and restores it back into DX

```
0116    POP   DX
```

DEC CX counts the loop counter down by one each loop pass, when it equals zero line 118 does not jump but finishes by executing the RET statement.

```
0117    DEC   CX
```

JNZ jumps to offset 010A if the CX register is zero

```
0118    JNZ   010A
011A    RET
```