**Experiment No**: 05

**Name of the Experiment:** Decimal Input up to 65535 and checking if Prime or not in 8086 assembly Language.

**Introduction:** This assembly code defines a procedure named CHECK\_PRIME aimed at determining whether a number stored in the AX register is a prime number or not. It initializes a divisor counter (CX) to 2, compares the input number with 1 (if it's 1, it immediately declares it as not prime), and then proceeds to check for divisors by iteratively dividing the number by incrementing divisors until reaching the number itself. If a divisor is found without a remainder, it concludes that the number is not prime and displays a message indicating this. Conversely, if no divisors are found, it determines the number is prime and displays a corresponding message. Finally, the procedure concludes by returning. The code uses conditional jumps and interrupts to print messages indicating whether the number is prime or not based on the division results and control flow.

**Program:**

.MODEL SMALL

.STACK 100H

.DATA

    msg1     DB 'Enter a number: $'

    msg2     DB 'The Entered Number is out of 16-bit range$'

    isPrime  DB 'The Entered Number is Prime$'

    notPrime DB 'The Entered Number is not Prime$'

.CODE

INDEC PROC

                PUSH BX

                PUSH CX

                PUSH DX

@BEGIN:

                MOV  AH, 2

                LEA  DX, msg1

                MOV  AH,9

                INT  21h

                XOR  BX, BX

                XOR  CX, CX

                MOV  AH, 1

                INT  21h

                CMP  AL, '-'

                JE   @MINUS

                CMP  AL, '+'

                JE   @PLUS

                JMP  @REPEAT2

@MINUS:

                MOV  CX, 1

@PLUS:

                INT  21h

@REPEAT2:

                CMP  AL, '0'

                JNGE @NOT\_DIGIT

                CMP  AL, '9'

                JNLE @NOT\_DIGIT

                AND  AX, 000FH

                PUSH AX

                MOV  AX, 10

                MUL  BX

                POP  BX

                ADD  BX, AX

                MOV  AH, 1

                INT  21h

                CMP  AL, 0DH

                JNE  @REPEAT2

                MOV  AX, BX

                OR   CX, CX

                JE   @EXIT

                NEG  AX

@EXIT:

                POP  DX

                POP  CX

                POP  BX

                RET

@NOT\_DIGIT:

                MOV  AH, 2

                MOV  DL, 0DH

                INT  21h

                MOV  DL, 0AH

                INT  21h

                JMP  @BEGIN

INDEC ENDP

CHECK\_PRIME PROC

                MOV  CX, 2

                CMP  AX, 1

                JE   @NOT\_PRIME

@CHECK\_DIVISOR:

                MOV  DX, 0

                MOV  BX, AX

                DIV  CX

                CMP  DX, 0

                JE   @NOT\_PRIME

                INC  CX

                MOV  AX, BX

                CMP  CX, AX

                JAE  @IS\_PRIME

                JMP  @CHECK\_DIVISOR

@IS\_PRIME:

                MOV  AH, 9

                LEA  DX, isPrime

                INT  21h

                JMP  @EXIT2

@NOT\_PRIME:

                MOV  AH, 9

                LEA  DX, notPrime

                INT  21h

@EXIT2:

                RET

CHECK\_PRIME ENDP

MAIN PROC

                MOV  AX, @DATA

                MOV  DS, AX

                CALL INDEC

                CALL CHECK\_PRIME

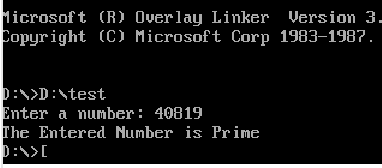
                MOV  AH, 4CH

                INT  21H

MAIN ENDP

END MAIN

**Input & Output:**



**Comments:**

1. The entered number should be positive.
2. The entered number should be between 1-65535.