MICROPROCESSOR REPORT

Name: Rania Hamada Muhammad

ID: 83

Faculty of Engineering, Alexandria University

Table of Contents

Task1

Hello world

Code:

```
name "hi-world"

; this example prints out  "hello world!"
; by writing directly to video memory.
; in vga memory: first byte is ascii character, byte that follows is
character attribute.
; if you change the second byte, you can change the color of
; the character even after it is printed.
; character attribute is 8 bit value,
; high 4 bits set background color and low 4 bits set foreground color.

; hex    bin        color
;
; 0      0000      black
; 1      0001      blue
; 2      0010      green
; 3      0011      cyan
; 4      0100      red
; 5      0101      magenta
; 6      0110      brown
; 7      0111      light gray
; 8      1000      dark gray
; 9      1001      light blue
; a      1010      light green
; b      1011      light cyan
; c      1100      light red
; d      1101      light magenta
; e      1110      yellow
; f      1111      white



org 100h

; set video mode
mov ax, 3     ; text mode 80x25, 16 colors, 8 pages (ah=0, al=3)
int 10h       ; do it!

; cancel blinking and enable all 16 colors:
mov ax, 1003h
mov bx, 0

int 10h
```

```
; set segment register:
mov     ax, 0b800h
mov     ds, ax

; print "Name"
; first byte is ascii code, second byte is color code.

mov [02h], 'N'

mov [04h], 'a'

mov [06h], 'm'

mov [08h], 'e'

mov [0ah], ':'

mov [0ch], ' '

mov [0eh], 'R'

mov [10h], 'a'

mov [12h], 'n'

mov [14h], 'i'

mov [16h], 'a'

mov [18h], ' '

mov [1ah], 'H'

mov [1ch], 'a'

mov [1eh], 'm'

mov [20h], 'a'

mov [22h], 'd'

mov [24h], 'a'
```

```
;Print seat number
mov [142h], 'S'
mov [144h], 'e'
mov [146h], 'a'
mov [148h], 't'
mov [14Ah], ' '
mov [14ch], 'N'
mov [14eh], 'u'
mov [150h], 'm'
mov [152h], 'b'
mov [154h], 'e'
mov [156h], 'r'
mov [158h], ':'
mov [15ah], '8'
mov [15ch], '3'
```

```
;Print Academic ID
mov [282h], 'A'
mov [284h], 'c'
mov [286h], 'a'
mov [288h], 'd'
mov [28ah], 'e'
mov [28ch], 'm'
mov [28eh], 'i'
mov [290h], 'c'
mov [292h], ' '
mov [294h], 'N'
mov [296h], 'u'
mov [298h], 'm'
mov [29ah], 'b'
mov [29ch], 'e'
mov [29eh], 'r'
mov [2a0h], ':'
mov [2a2h], '0'
mov [2a4h], '1'
mov [2a6h], '7'
mov [2a8h], '0'
mov [2aah], '0'
mov [2ach], '7'
mov [2aeh], '2'
mov [2b0h], '4'
```

```
; color all characters:
mov cx, 18  ; number of characters.
mov di, 03h ; start from byte after 'h'

c:  mov [di], 11101100b   ; light red(1100) on yellow(1110)
    add di, 2 ; skip over next ascii code in vga memory.
    loop c


mov cx, 14  ; number of characters.
mov di, 143h ; start from byte after 'h'

d:  mov [di], 11101100b   ; light red(1100) on yellow(1110)
    add di, 2 ; skip over next ascii code in vga memory.
    loop d


mov cx, 24  ; number of characters.
mov di, 283h ; start from byte after 'h'

e:  mov [di], 11101100b   ; light red(1100) on yellow(1110)
    add di, 2 ; skip over next ascii code in vga memory.
    loop e

; wait for any key press:
mov ah, 0
int 16h

ret
```

Screen shot of the output

Figure 1: Output of 'Hello World' Example

Palindrome

Code

```
; this sample checks if string is a palindrome or not.
; palindrome is a text that can be read backwards
; and give the same meaning as if it was read forward.
; for example: "abba" is polindrome.
; note: this program is case sensitive, "abba" is not "abba".

name "pali"

org 100h

; set video mode
mov ax, 3     ; text mode 80x25, 16 colors, 8 pages (ah=0, al=3)
int 10h       ; do it!

; cancel blinking and enable all 16 colors:
mov ax, 1003h
mov bx, 0
int 10h


jmp studentName

m1:
s db ' Name: Rania Hamada'
s_size = $ - m1
  db 0Dh,0Ah,'$'

studentName:

; first let's print it:
mov ah, 9
mov dx, offset s
int 21h

stop1:
    ; print new line
    mov dl, 10
    mov ah, 02h
    int 21h
    mov dl, 13
    mov ah, 02h
    int 21h
```

```
jmp seatNo

m2:
no db ' Seat number: 83'
no_size = $ - m1
  db 0Dh,0Ah,'$'

seatNo:

; first let's print it:
mov ah, 9
mov dx, offset no
int 21h

stop2:
    ; print new line
    mov dl, 10
    mov ah, 02h
    int 21h
    mov dl, 13
    mov ah, 02h
    int 21h

jmp AcademicNo

m3:
Ano db ' Academic number: 01700724'
Ano_size = $ - m3
  db 0Dh,0Ah,'$'

AcademicNo:

; first let's print it:
mov ah, 9
mov dx, offset Ano
int 21h


stop:

; wait for any key press:
;mov ah, 0
;int 16h


; set segment register:
mov     ax, 0b800h
mov     ds, ax
```

```
; color all characters:
mov cx, 18  ; number of characters.
mov di, 03h ; start from byte after 'h'

c:  mov [di], 11101100b   ; light red(1100) on yellow(1110)
    add di, 2 ; skip over next ascii code in vga memory.
    loop c

mov cx, 15  ; number of characters.
mov di, 143h ; start from byte after 'h'

d:  mov [di], 11101100b   ; light red(1100) on yellow(1110)
    add di, 2 ; skip over next ascii code in vga memory.
    loop d


mov cx, 25  ; number of characters.
mov di, 283h ; start from byte after 'h'

e:  mov [di], 11101100b   ; light red(1100) on yellow(1110)
    add di, 2 ; skip over next ascii code in vga memory.
    loop e

ret


;msg1 db "  this is palindrome!$"
;msg2 db "  this is not a palindrome!$"
```

Screen shot of the output

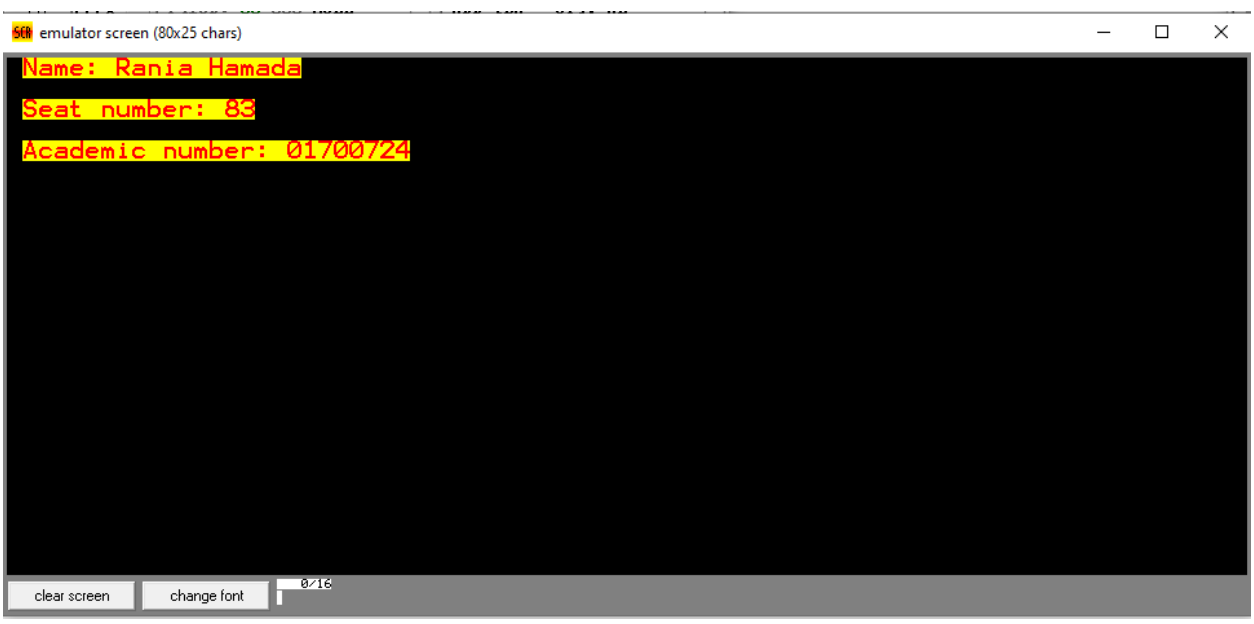

Figure 2: Output of 'Palindrome' Example

Task 2

Calculating prime factors, GCD and LCM

Code

The code is divided into 4 parts, each part has to do something. The first part is to take two unsigned 8-bit numbers from the user.

```
;This code is used to get the prime factors of 2 unsigned numbers and then
calculating their gcd and lcm

; It's devided into 4 parts, the first part is to take the inputs from the
user

; the second part is to calculate the prime factors for each number

; the third part is to calculate the gcd for each number

; the last part is to calculate the lcm for each number

INCLUDE "EMU8086.INC"

name "Prime Factors - GCD - LCM"

org 100h
;---------------PRINT A WELCOMING MESSAGE -----------------------

; first let's print it:
mov ah, 9
mov dx, offset msg1
int 21h

;PRINT A NEW LINE
MOV dl, 10
MOV ah, 02h
INT 21h
MOV dl, 13
MOV ah, 02h
INT 21h

; first let's print it:
mov ah, 9
mov dx, offset msg2
int 21h

;PRINT A NEW LINE
MOV dl, 10
MOV ah, 02h
INT 21h
MOV dl, 13
MOV ah, 02h
INT 21h
```

```
;------------------ CODE FOR GETTING THE PRIME FACTORS OF THE FIRST NUMBER
---------

;PRINT A NEW LINE
MOV dl, 10
MOV ah, 02h
INT 21h
MOV dl, 13
MOV ah, 02h
INT 21h


; PRINT MSG5
MOV AH, 9
MOV DX, OFFSET MSG5
INT 21H

MOV AH, 0
MOV AL, NUM1

CALL PRINT_NUM  ;print the first number

MOV AH, 9
MOV DX, OFFSET MSG6
INT 21H

;-----------------------------------------

MOV AX, 0000H  ;INITIALIZE AX
MOV AL, NUM1   ;COPY THE FIRST NUMER
MOV BL, 2      ;TO EVEN OR ODD CHECK
JMP CLOOP1
```

Then do the algorithm shown in flow chart to calculate the prime factors

```
;------------------ CODE FOR GETTING THE PRIME FACTORS OF THE FIRST NUMBER
---------

;PRINT A NEW LINE
MOV dl, 10
MOV ah, 02h
INT 21h
MOV dl, 13
MOV ah, 02h
INT 21h


; PRINT MSG5
MOV AH, 9
MOV DX, OFFSET MSG5
INT 21H

MOV AH, 0
MOV AL, NUM1

CALL PRINT_NUM  ;print the first number

MOV AH, 9
MOV DX, OFFSET MSG6
INT 21H
;----------------------------------------
MOV AX, 0000H  ;INITIALIZE AX
MOV AL, NUM1   ;COPY THE FIRST NUMER
MOV BL, 2      ;TO EVEN OR ODD CHECK
JMP CLOOP1
CLOOP1:
    MOV DL, AL ;COPY AL TO USE AGAIN
    DIV BL     ;DIVIDE BY 2

    CMP AH, 0  ;even
    JE L1      ;GO TO L1

    CMP AH, 1  ;odd

    MOV AL, DL
    MOV CL, 1 ;START LOOP FROM 3

    MUL AL
    MOV BX, AX

    MOV AL, DL
    MOV DX, BX

    JE L2

LOOP CLOOP1
```

```
L1:
    MOV VAR3, AL
    ;PRINT A NEW LINE
    MOV dl, 10
    MOV ah, 02h
    INT 21h
    MOV dl, 13
    MOV ah, 02h
    INT 21h

    PRINT '2'   ;PRINT 2

    PUSH 2   ; TO STORE IN STACK

    MOV AX, 0000H
    MOV AL, VAR3


    CMP AL, 1   ;CHECK AL VALUE

    JE LOOPEND  ;IF IT'S EQUAL 1

    JNE CLOOP1  ;
RET
L2:
    MOV AH, 0
    MOV VAR2, AL
    ADD CL, 2

    DIV CL
    CMP AH, 0
    JE L3

    CMP AH, 1
    JGE L6
RET
```

```
L3:

    MOV VAR5, AL
    MOV VAR4, CL

    ;PRINT A NEW LINE
    MOV dl, 10
    MOV ah, 02h
    INT 21h
    MOV dl, 13
    MOV ah, 02h
    INT 21h

    MOV AX, 0000H
    MOV CX, 0000H

    MOV CL, VAR4
    MOV AL, CL

    CALL PRINT_NUM

    PUSH AX  ; STORE IT IN STACK

    MOV AX, 0000H
    MOV AL, VAR5
    CMP AL, 1
    JE LOOPEND

    JG L4


RET

L4:

    DIV CL
    CMP AH, 0
    JE L3

    JNE L5


RET

L5:
MOV AL, VAR5
JMP L2
RET

L6:
MOV AL, VAR2
JMP L2
RET

LOOPEND:
```

```
;------------------ CODE FOR GETTING THE PRIME FACTORS OF THE SECOND NUMBER
---------

PUSH 0

;PRINT A NEW LINE
MOV dl, 10
MOV ah, 02h
INT 21h
MOV dl, 13
MOV ah, 02h
INT 21h


; PRINT MSG5
MOV AH, 9
MOV DX, OFFSET MSG5
INT 21H

MOV AH, 0
MOV AL, NUM2

CALL PRINT_NUM  ;print the first number

MOV AH, 9
MOV DX, OFFSET MSG6
INT 21H

;----------------------------------------

MOV AX, 0000H  ;INITIALIZE AX
MOV AL, NUM2   ;COPY THE FIRST NUMER
MOV BL, 2      ;TO EVEN OR ODD CHECK
JMP CLOOP2

CLOOP2:
    MOV DL, AL ;COPY AL TO USE AGAIN
    DIV BL     ;DIVIDE BY 2

    CMP AH, 0  ;even
    JE L1_1      ;GO TO L1

    CMP AH, 1  ;odd

    MOV AL, DL
    MOV CL, 1 ;START LOOP FROM 3

    MUL AL
    MOV BX, AX

    MOV AL, DL
    MOV DX, BX

    JE L2_1

LOOP CLOOP2
```

```
L1_1:
    MOV VAR3, AL

    ;PRINT A NEW LINE
    MOV dl, 10
    MOV ah, 02h
    INT 21h
    MOV dl, 13
    MOV ah, 02h
    INT 21h

    PRINT '2'   ;PRINT 2

    PUSH 2   ; STORE IT IN STACK

    MOV AX, 0000H
    MOV AL, VAR3


    CMP AL, 1   ;CHECK AL VALUE

    JE LOOPEND_1  ;IF IT'S EQUAL 1

    JNE CLOOP2  ;
RET


L2_1:
    MOV AH, 0
    MOV VAR2, AL
    ADD CL, 2

    DIV CL
    CMP AH, 0
    JE L3_1

    CMP AH, 1
    JGE L6_1


RET
```

```
L3_1:

    MOV VAR5, AL
    MOV VAR4, CL

    ;PRINT A NEW LINE
    MOV dl, 10
    MOV ah, 02h
    INT 21h
    MOV dl, 13
    MOV ah, 02h
    INT 21h

    MOV AX, 0000H
    MOV CX, 0000H

    MOV CL, VAR4
    MOV AL, CL

    CALL PRINT_NUM

    PUSH AX    ; STORE IT IN STACK

    MOV AX, 0000H
    MOV AL, VAR5
    CMP AL, 1
    JE LOOPEND_1

    JG L4_1


RET

L4_1:

    DIV CL
    CMP AH, 0
    JE L3_1

    JNE L5_1


RET

L5_1:
MOV AL, VAR5
JMP L2_1
RET

L6_1:
MOV AL, VAR2
JMP L2_1
RET

LOOPEND_1:
```

Then getting the common factors using the stored values of the prime factors in stack and store it in two arrays and then store the common factors in a new array.

```
;------------------------- GET COMMON FACTORS ----------------------------
------------


MOV AX, 00H
MOV BX, 00H
MOV CX, 00H
MOV DX, 00H
MOV SI, 00H

FACTORS2:

MOV AX, SP

POP AX

MOV BX, AX

CMP BX, 0
JNE STORE2

MOV SI, 0
JE FACTORS1


STORE2:
    INC PRIMESLEN2
    MOV PRIMES2[SI], BL
    INC SI
    JMP FACTORS2
RET


FACTORS1:
MOV AX, SP

POP AX

MOV BX, AX

CMP BX, 0
JE ENDFACTORS
JNE STORE1

STORE1:
    INC PRIMESLEN1
    MOV PRIMES1[SI], BL
    INC SI
    JMP FACTORS1
RET

ENDFACTORS:
```

```
MOV SI, OFFSET PRIMES1
MOV DI, OFFSET PRIMES2

MOV DIHELP, DI
MOV CX, 0

CMPLOOP1:
    MOV DX, CX

    MOV AL, [SI]  ;[SI] ELEMENT IN AL
    CMPLOOP2:
    MOV BL, [DI]
    INC DI

    CMP AL, BL

    JNE NOTSAME

    JE STORECOMMON ;STORE IN COMMON FACTOR ARRAY

    LOOP CMPLOOP2
LOOP CMPLOOP1

STORECOMMON:

MOV SIPRIMES, SI  ;SI OF PRIMES1 ARRAY

MOV SI, SIFACTORS

MOV COMMONFACTORS[SI], AL

INC SI
MOV SIFACTORS, SI

MOV [DI-1], 0

INC DL
CMP DL, PRIMESLEN1
JGE ENDCMP
JL RESETL1AGAIN
RET
```

```
RESETL1AGAIN:
MOV CX, DX
INC SIPRIMES
MOV SI, SIPRIMES
MOV DI, DIHELP
JMP CMPLOOP1
RET

NOTSAME:
INC CL
CMP CL, PRIMESLEN2
JG RESETL1   ; RESET CMPLOOP2
JLE CMPLOOP2
RET

RESETL1:
INC DX
CMP DL, PRIMESLEN1
JG ENDCMP

MOV CX, DX
;INC SI
;MOV DI, DIHELP
JMP CMPLOOP1
RET

ENDCMP:
```

After that we used the common factors to calculate GCD

```
;------------------------ GET GCD FROM COMMON FACTORS ARRAY ----------------
-------------

MOV SI, 0
MOV AL, 1

GCDLOOP:
    CMP COMMONFACTORS[SI], 0
    JE ENDGCD
    JNE CALCGCD ;CALCULATE GCD
LOOP GCDLOOP


CALCGCD:
    MUL COMMONFACTORS[SI]
    MOV GCD, AX
    INC SI
    JMP GCDLOOP
RET

ENDGCD:

;PRINT A NEW LINE
MOV dl, 10
MOV ah, 02h
INT 21h
MOV dl, 13
MOV ah, 02h
INT 21h


; PRINT GCD
MOV AH, 9
MOV DX, OFFSET MSG7
INT 21H

MOV AX, GCD

CALL PRINT_NUM  ;print gcd
```

Final, calculating LCM using the following equation:

$$\text{LCM} = \frac{\text{NUM1} \cdot \text{NUM2}}{\text{GCD(NUM1,NUM2)}}$$

```
;---------------------------- CALCULATE LCM -------------------------------
---------

MOV AX, 00H
MOV BX, 00H

MOV AL, NUM1
MOV BL, NUM2

MUL BL

MOV DX, GCD
DIV DL

MOV LCM,AX

;PRINT A NEW LINE
MOV dl, 10
MOV ah, 02h
INT 21h
MOV dl, 13
MOV ah, 02h
INT 21h

; PRINT GCD
MOV AH, 9
MOV DX, OFFSET MSG8
INT 21H

MOV AX, LCM

CALL PRINT_NUM

; wait for any key press:
mov ah, 0
int 16h

ret
```

Data, procedures and macros have been used

```
primes1 db 100 dup(?)
primes2 db 100 dup(?)
commonfactors db 100 dup(?)

primeslen1 db 0
primeslen2 db 0

SIPRIMES DW ?
SIFACTORS DW 0
DIHELP DW ?

GCD DW 1
LCM DW ?

msg1 db 'Welcome to our program for calculating prime factors$'

msg2 db 'This program takes two numbers and then prints the prime factors
for each one$'

msg3 db 'Please, Enter a number between 2 and 255: $'

msg4 db 'Please, Enter a VALID number between 2 and 255: $'

msg5 db 'Prime factors of $'

msg6 db ' are: $'

msg7 db 'GCD = $'

msg8 db 'LCM = $'

NUM1 DB ?
NUM2 DB ?

VAR1 DB ?
VAR2 DB ?
VAR3 DB ?
VAR4 DB ?
VAR5 DB ?
```

```
; PROC TO TAKE INPUT FROM THE USER

TAKEINPUT PROC


    ;PRINT A NEW LINE
    MOV dl, 10
    MOV ah, 02h
    INT 21h
    MOV dl, 13
    MOV ah, 02h
    INT 21h


    mov ah, 9
    mov dx, offset msg3
    int 21h


    CALL SCAN_NUM  ; GET NUMBER IN CX

    MOV AX, CX     ; COPY THE NUMBER TO AX

    MOV BX, AX


    RET            ; RETURN TO CALLER

    TAKEINPUT ENDP


; PROC TO TAKE INPUT AGAIN FROM THE USER

TAKEINPUTAGAIN PROC

    ;PRINT A NEW LINE
    MOV dl, 10
    MOV ah, 02h
    INT 21h
    MOV dl, 13
    MOV ah, 02h
    INT 21h

    mov ah, 9
    mov dx, offset msg4
    int 21h


    CALL SCAN_NUM  ; GET NUMBER IN CX

    MOV AX, CX     ; COPY THE NUMBER TO AX
    MOV BX, AX
    RET
    TAKEINPUTAGAIN ENDP
```

```
; MACROS TO DEFINE PROCS

DEFINE_SCAN_NUM

;DEFINE_PRINT_STRING

DEFINE_PRINT_NUM

DEFINE_PRINT_NUM_UNS  ;REQUIRED FOR PRINT_NUM

;DEFINE_PTHIS
```

All codes are attached here: https://github.com/raniaelhagin/Microprocessor-Tasks

Output

Asking for the first number until it gets a valid number and the same for the second number

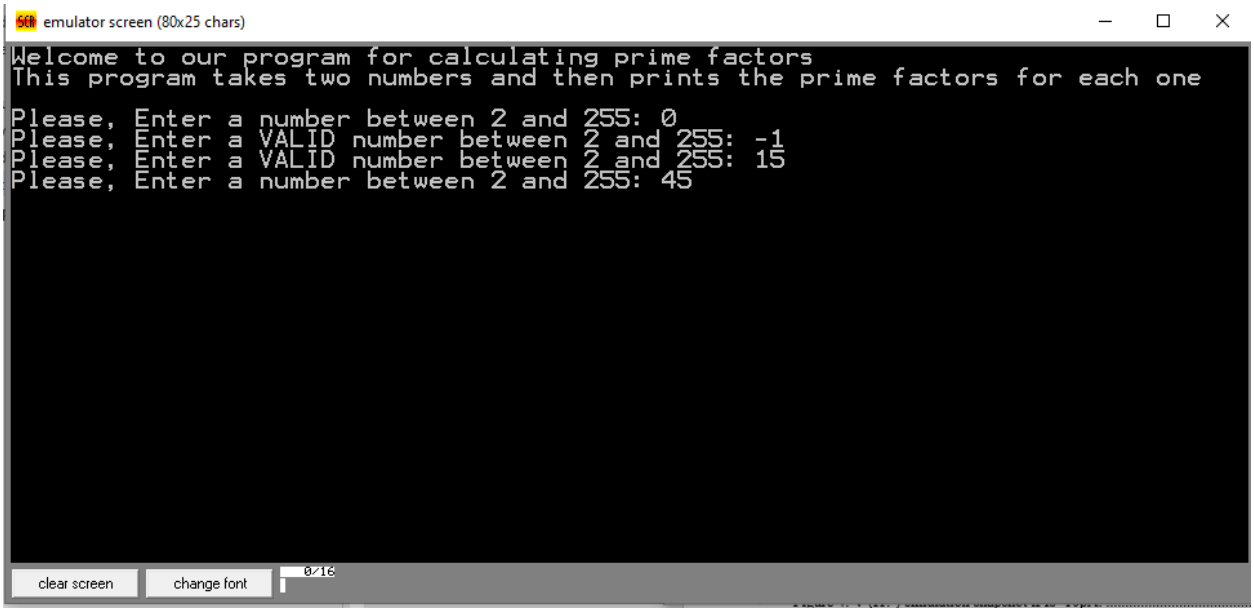

Figure 3: Asking the user for numbers

Then it prints the prime factors for each number

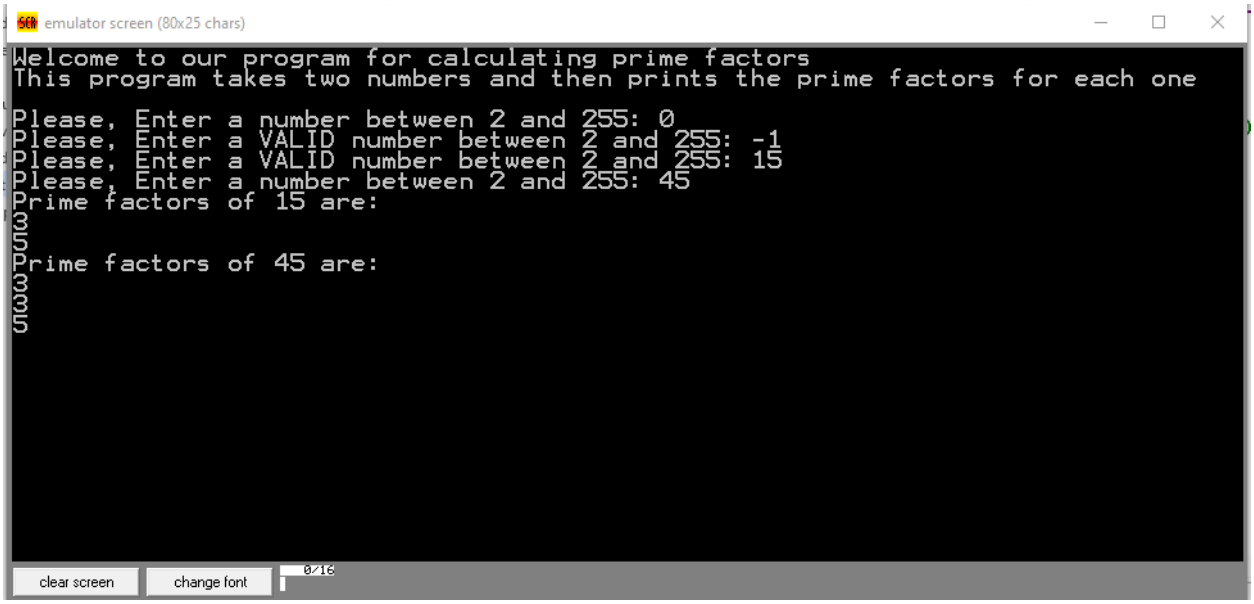

Figure 4: Print prime factors

The prime factors stored in stack separated by a memory location with 0000h value

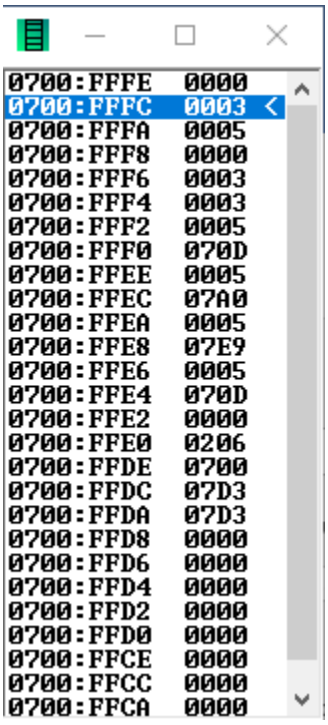

Figure 5: Prime factors in stack

Storing the prime factor in two arrays and storing the common factor in another array

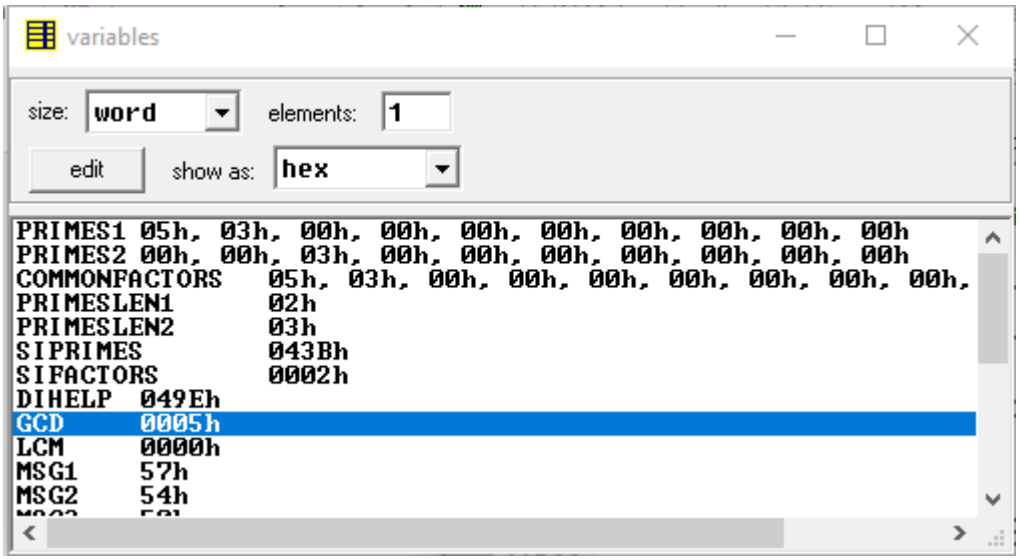

Figure 6: Common factors and Prime factors

Then calculating and printing GCD and LCM

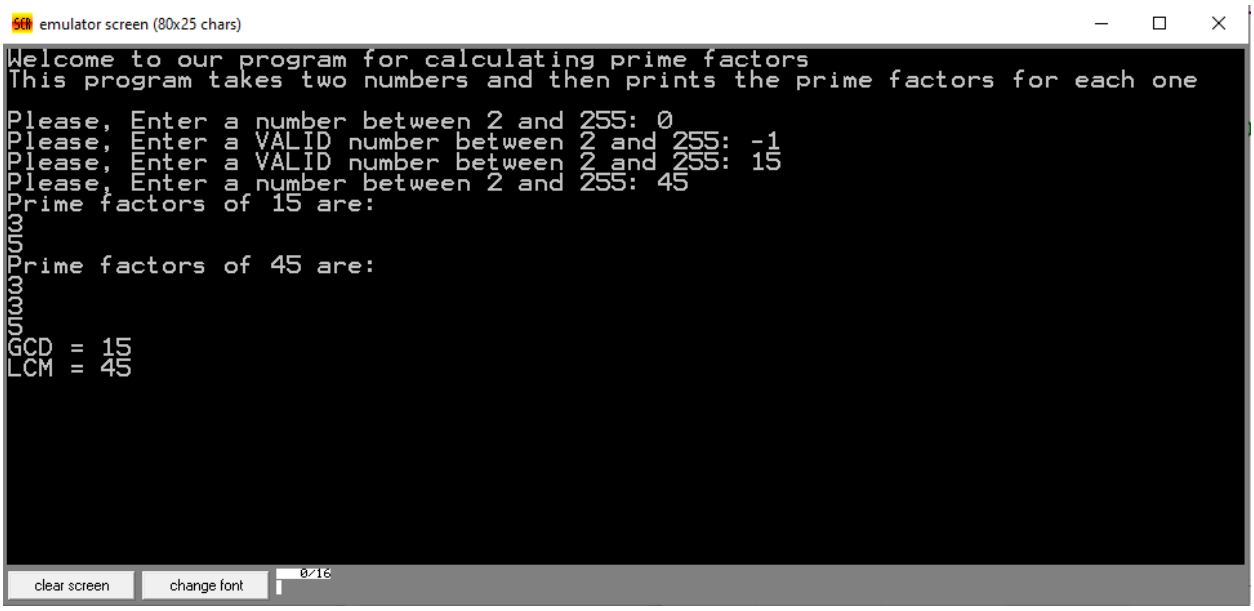

Figure 7: GCD and LCM

Figure Captions