Alexandria University

Faculty of Engineering

Communication and Electronics Department

Third Year 2020

Software Microprocessor Assignment-2

Section_1

Name:	**ID**
Alaa Mohamed Morsy	4

TABLE OF CONTENT

ASSUMTIONS

The program assume that the user does not input any string or letters.

Frist part

Printing the prime factors of two numbers, the Least common multiple (L.C.M) and the Highest common factor (H.C.F)

THE CODE

```
#make_bin#
; BIN is plain binary format similar to .com format, but not limited to 1 segment;
; All values between # are directives, these values are saved into a separate .binf file.
; Before loading .bin file emulator reads .binf file with the same file name.
; All directives are optional, if you don't need them, delete them.
; set loading address, .bin file will be loaded to this address:
#LOAD_SEGMENT=0500h#
#LOAD_OFFSET=0000h#
; set entry point:
#CS=0500h#   ; same as loading segment
#IP=0000h#    ; same as loading offset
; set segment registers
#DS=0500h#   ; same as loading segment
#ES=0500h#   ; same as loading segment
; set stack
#SS=0500h#   ; same as loading segment
#SP=FFFEh#  ; set to top of loading segment
; set general registers (optional)
#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
#DI=0000h#
```

```
#BP=0000h#
include emu8086.inc

JMP START
NOTE DB 0Dh,0Ah,'NOTE:PLEASE INTER UNSIGNED NUMBERS FROM 0 TO 255$'
msg1 db 0Dh,0Ah, 'Enter the first number:$'
msg2 db 0Dh,0Ah, 'Enter the second number:$'
err_minus db 0Dh,0Ah, 'The input must be unsigned!!$'
err_great db 0Dh,0Ah, 'The input mustnot be greater than 255!!$'
; 1004h  ;the address of the memory where num1 is stored
; 1014h  ;the address of the memory where num2 is stored
prim_num db 2h,3h,5h,7h,0Bh,0Ch,11h,13h,1fh

msg3 db 0Dh,0Ah, 'the prime factors of $'
space db ' $'
new_l db 0DH,0AH,' are: $'

msg4 db 0Dh,0Ah, 'the HCF of the two numbers= $'
;1020h  ;where prime mo. of 1st digit is stored
;1050h  ;where prime mo. of 2st digit is stored

msg5 db 0Dh,0Ah, 'the LCM of the two numbers= $'

START:
  ; first print NOTE:
  mov ah, 9
  mov dx, offset NOTE
  int 21h
;input of 1st number
   mov ah, 9
```

```
 mov dx, offset msg1
 int 21h
 Mov di,1000h
 call inp_NUM
 MOV  [1004h],BX    ; 1004h the address of the memory where num1 is stored


;input of 2st number
  mov ah, 9
 mov dx, offset msg2
 int 21h
 Mov di,1010h
 call inp_NUM
 MOV  [1014h],BX    ; 1014h the address of the memory where num2 is stored

;Calculate the primefactors
 Mov Bx, 1004h
 Mov cx,[bx]
 mov ah, 9
 mov dx, offset msg3  ;
 int 21h
mov ax,cx
call print_num_uns
 mov ah, 9
 mov dx, offset new_l ;print new line
 int 21h
 Mov di,1020h  ;1020h  ;where prime mo. of 1st digit is stored
 call prime_fac
```

```
;****second no***
 Mov Bx, 1014h  ; 1014h the address of the memory where num2 is stored
Mov cx,[bx]
 mov ah, 9
 mov dx, offset msg3
 int 21h
mov ax,cx
call print_num_uns

 mov ah, 9
 mov dx, offset new_l ;print new line
 int 21h
Mov di,1050h  ;1050h  ;where prime mo. of 2st digit is stored
call prime_fac
;Calculate the HCF
mov ax,0
mov al,[1004h]
mov bl,[1014h]
cmp al,bl
jl  re
back:
mov ah,0
div bl
cmp ah,0
je  end

mov al,bl
mov bl,ah
jmp back
```

```
end:
 mov ah, 9
 mov dx, offset msg4
 int 21h
 mov ax,0
 mov al,bl
 call PRINT_NUM_UNS
 mov dl,al ;the HCF

 ;Calculate the LCM
 mov ax,0
 mov bx,0
 mov al,[1004h]
 mov bl,[1014h]
 mul bl ;Multiplication of the 2 numbers
 div dl ;divide the HCF
 ; al have the LCM
 mov bh,al

 mov ah, 9
 mov dx, offset msg5
 int 21h
 mov ax,0
 mov al,bh
 call PRINT_NUM_UNS



 HLT          ; halt!
```

```
;*********************************************
;************functions************************
;*********************************************
;This procedure used to input and check the input
 inp_NUM       PROC    NEAR
  PUSH  DX
  PUSH  AX
  PUSH  SI

  MOV   CX,0 ;reset counter
  MOV   BX,0
READ:
 ; get char from keyboard
 ; into AL:
 MOV  AH, 1
 INT  21h

 ; check for MINUS:
 CMP     AL, '-'
 JE      error_minus

 ; check for ENTER key:
 CMP  AL, 0Dh
 JE   CALC_num

 sub  al,30h
 MOV  [di],AL
 inc  di
 inc  cx
 CMP  CX,4 ;check if overflow
```

```
jnl  overfl

JMP  READ
error_minus:  ;message if the input is signed

 mov ah, 9
 mov dx, offset err_minus
 int 21h
 jmp START

overfl:  ;message if overflow
  mov ah, 9
 mov dx, offset err_great
 int 21h
 jmp START

 greater:   ;message if the input is greater than 255
  mov ah, 9
 mov dx, offset err_great
 int 21h
 jmp START

CALC_num:
dec  di
mov  al,1h
mul  [di]
ADD  bX,AX
dec  di
Mov  al,0Ah
MUL  [di]
```

```
ADD  BX,AX
DEC  di
mov  al,64h
MUL  [di]
ADD  BX,AX
CMP  BX,255
JNL greater
 POP   SI
 POP   AX
 POP   DX
 RET
 inp_NUM   ENDP
;************************
 newprime:
 mul bl
 ADD AL,cl
 inc si
 jmp fac

 printf:
  mov  cx,ax
 Mov  ax,bx
 mov [di],ax
 inc  di
 call print_num_uns
 mov ah, 9
 mov dx, offset space
 int 21h
 mov  ax,cx
 jmp next
```

```
;This procedure used to input and check the input
 prime_fac       PROC    NEAR
  PUSH  DX
  PUSH  AX
  PUSH  SI

  MOV   si,0 ;reset counter
  mov   bx,0 ;reset the div
  Mov   ax,cx
  mov   cx,0 ;reset for get reminder
  fac:
  MOV  bl,prim_num[si]
  div  bl
  cmp  AH,0
  mov  cl,ah
  JNE  newprime
  jmp  printf ;to print the prime no and store them
next:
  cmp  ax,1
  je   exit
  jmp fac

 exit:
  mov [di],'$' ;termination for prime factors
  POP   SI
  POP   AX
  POP   DX
  RET
  prime_fac   ENDP
```

```
;********
re:
 mov al,[1014h]
 mov bl,[1004h]
 jmp back

 DEFINE_PRINT_NUM_UNS
 END
```

The method of operation

The user enters two numbers then the program checks if they are unsigned and within the range (from 0 to 255) and allow the user to re-enter the previous values if there was any problem using the procedure **inp_num**.

The program stores the two number in [1004h] and [1014h] memory locations.

Then, the program uses the procedure **prime_fac** to calculate the prime factors of each number and print them.

For calculating H.C.F, the program uses division method which is consists of 4 steps:

- **I.** Divide the large number by the smaller one.
- **II.** Then the remainder is treated as divisor and the divisor as dividend.
- **III.** Divide the first divisor by the first remainder.
- **IV.** Divide the second divisor by the second remainder.
- **V.** Continue this process till the remainder becomes 0.

For calculating L.C.M, the program multiples the two numbers then divide the result with the H.C.F .

The output:

Entering 30 and 12 as Example1

```
NOTE:PLEASE INTER UNSIGNED NUMBERS FROM 0 TO 255
Enter the first number:30
Enter the second number:12
the prime factors of 30
 are: 2 3 5
the prime factors of 12
 are: 2 2 3
the HCF of the two numbers= 6
the LCM of the two numbers= 60
```

Entering 80 and 16 as Example2

```
NOTE:PLEASE INTER UNSIGNED NUMBERS FROM 0 TO 255
Enter the first number:16
Enter the second number:80
the prime factors of 16
 are: 2 2 2 2
the prime factors of 80
 are: 2 2 2 2 5
the HCF of the two numbers= 16
the LCM of the two numbers= 80
```

Second part

Getting from the user an array of numbers then, classify the numbers into even and odd arrays. Then, display odd numbers ascendingly and even numbers in descending order.

THE CODE

```
#make_bin#
; BIN is plain binary format similar to .com format, but not limited to 1 segment;
; All values between # are directives, these values are saved into a separate .binf file.
; Before loading .bin file emulator reads .binf file with the same file name.
; All directives are optional, if you don't need them, delete them.
; set loading address, .bin file will be loaded to this address:
#LOAD_SEGMENT=0500h#
#LOAD_OFFSET=0000h#
; set entry point:
#CS=0500h#   ; same as loading segment
#IP=0000h#    ; same as loading offset
; set segment registers
#DS=0500h#   ; same as loading segment
#ES=0500h#   ; same as loading segment
; set stack
#SS=0500h#   ; same as loading segment
#SP=FFFEh#  ; set to top of loading segment
; set general registers (optional)
#AX=0000h#
#BX=0000h#
#CX=0000h#
#DX=0000h#
#SI=0000h#
```

```
#DI=0000h#
#BP=0000h#
;The code
include emu8086.inc

JMP start
err_minus db 0Dh,0Ah, 'The input must an integer!!$'
err_great db 0Dh,0Ah, 'The input mustnot be greater than 255!!$'
Note db ' You are alowed to enter numbers up to 255$'
msg1 db 0dh,0ah,'Please, Enter how many numbers you want to enter: $'
msg2 db 0dh,0ah,'Enter the nmber : $'
;1010h Number of numbers
;1011h the numbers
;1100 odd no.s sorted asending
;1200 even no. sorted descending
msg_odd db 0dh,0ah,'The odd numbers are: $'
msg_even db 0dh,0ah,'The even numbers are: $'
space db ' $'

start:
; first print NOTE:
  mov ah, 9
  mov dx, offset Note
  int 21h
;print msg1
   mov ah, 9
  mov dx, offset msg1
  int 21h
  ;get the no. of numbers in AL
```

```
 call inp_NUM
 MOV  [1009h],BX  ; 1009h the address of the memory where no. of numbers is stored

 ;Entering the numbers
 mov ch,0
 mov cl,[1009h]
   mov di,1010h   ;memory location begin to store the numbers
NO.Enter:
 ;print msg2
   mov ah, 9
  mov dx, offset msg2
  int 21h
  ;get the number
 call inp_NUM
 mov [di],bx
 INC di

 LOOP NO.Enter

push si
 mov bx,0
 mov cx,0
 mov si,0
 mov di,0
odd/even:   ;classification
 mov cl,[1010h+si]
 mov ah,0
 mov al,[1010h+si] ;mov the no. to the accumlator
 mov ch,2
 div ch    ; div by 2
```

```
 cmp ah,1
 je  odd
 jmp  even
endl:
 inc si
 cmp si,[1009h] ;compare with no. of numbers
 jne odd/even

 mov [1300h],bx ;record no. of odd
 mov [1302h],di ;record no. of even
 pop si

 ;for odd numbers
 mov al,0
 mov ah,0
 mov bx,0
 mov cx,0

 mov al,[1300h]
 mul [1300h]   ;check n^2
 mov dx,ax

odd_sort:  ;BUbble sorting
 mov al,[1100h+bx]
 mov ah,[1100h+bx+1]
 inc cx
 cmp al,ah
 ja  al_bigger  ;ascenging order
 e:
 inc bx
```

```
cmp bx,[1300h]
je  resetO
contO:
cmp cx,dx
jne odd_sort

 ;for even numbers
mov al,0
mov ah,0
mov bx,0
mov cx,0

mov al,[1302h]
mul [1302h]   ;check n^2
mov dx,ax

even_sort:
mov al,[1200h+bx]
mov ah,[1200h+bx+1]
inc cx
cmp al,ah
jb  al_smaller ;descending order
f:
inc bx
cmp bx,[1302h]
je  resetE
contE:
cmp cx,dx
jne even_sort
```

```
 mov bx,0
;Printing odd numbers
 mov ah, 9
 mov dx, offset msg_odd
 int 21h
print_odd:
 mov dl,[1101h+bx]
 mov ax,0
 mov al,dl

 call PRINT_NUM_UNS

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

 inc bx
 cmp bx,[1300h]
 jne print_odd

  mov bx,0
;Printing even numbers
 mov ah, 9
 mov dx, offset msg_even
 int 21h
print_even:
 mov dl,[1200h+bx]
 mov ax,0
 mov al,dl
```

```
call PRINT_NUM_UNS

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

inc bx
cmp bx,[1302h]
jne print_even

HLT          ; halt!

odd:
mov [1100h+bx],cl
inc bx
jmp endl

even:
mov [1200h+di],cl
inc di
jmp endl

al_bigger:  ;odd sorting(ascending)
mov [1100h+bx+1],al
mov [1100h+bx],ah
jmp  e

al_smaller:
mov [1200h+bx+1],al
mov [1200h+bx],ah
```

```
 jmp  f

  resetO:   ;reset odd loop
 mov bx,0
 jmp contO

  resetE:  ;reset even loop
 mov bx,0
 jmp contE

 ;*********************************************
 ;************functions************************
 ;*********************************************
;This procedure used to input and check the input
 inp_NUM        PROC    NEAR
  PUSH  DX
  PUSH  AX
  PUSH  CX
  PUSH  SI
  PUSH  DI

  MOV   CX,0 ;reset counter
  MOV   BX,0
  MOV   di,1000h
READ:
 ; get char from keyboard
 ; into AL:
 MOV  AH, 1
 INT  21h
```

```
; check for MINUS:
CMP     AL, '-'
JE      error_minus

; check for ENTER key:
CMP  AL, 0Dh
JE   CALC_num

sub  al,30h
MOV  [di],AL
inc  di
inc  cx
CMP  CX,4 ;check if overflow
jnl  overfl

JMP  READ
error_minus:  ;message if the input is signed

 mov ah, 9
 mov dx, offset err_minus
 int 21h
 jmp START

overfl:  ;message if overflow
  mov ah, 9
 mov dx, offset err_great
 int 21h
 jmp START

 greater:   ;message if the input is greater than 255
```

```
 mov ah, 9
 mov dx, offset err_great
 int 21h
 jmp START

CALC_num:
dec  di
mov  al,1h
mul  [di]
ADD  bX,AX
dec  di
Mov  al,0Ah
MUL  [di]
ADD  BX,AX
DEC  di
mov  al,64h
MUL  [di]
ADD  BX,AX
CMP  BX,255
JNL greater

 POP   DI
 POP   SI
 POP   CX
 POP   AX
 POP   DX
 RET
 inp_NUM   ENDP
DEFINE_PRINT_NUM_UNS
END
```

The method of operation

The program asks the user to enter how many numbers he wants to enter. The program checks if the number is signed or out of range if that the program shows an error message and restart the program. The number of numbers is stored in memory location 1009h.

The program asks the user to enter the numbers and store them starting with memory location 1010h.

After entering the last number, the program begins to classify the numbers putting odd numbers in an array - begin with 1100h memory location – and even numbers in another array – begin with 1200h memory location-.

Note: the beginning of each array is choosing to allow 255 numbers.

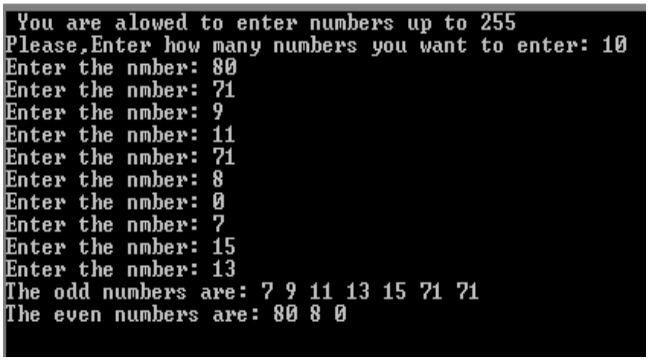
The program uses Bubble sorting algorithm to sort odd numbers in ascending order and even numbers in descending order.

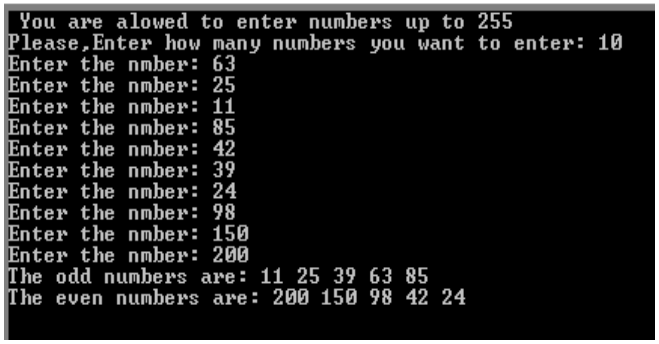
Note:

- number of elements in odd array is stored in 1300h memory location.
- number of elements in even array is stored in 1302h memory location.

The output:

Example1:





Example2: