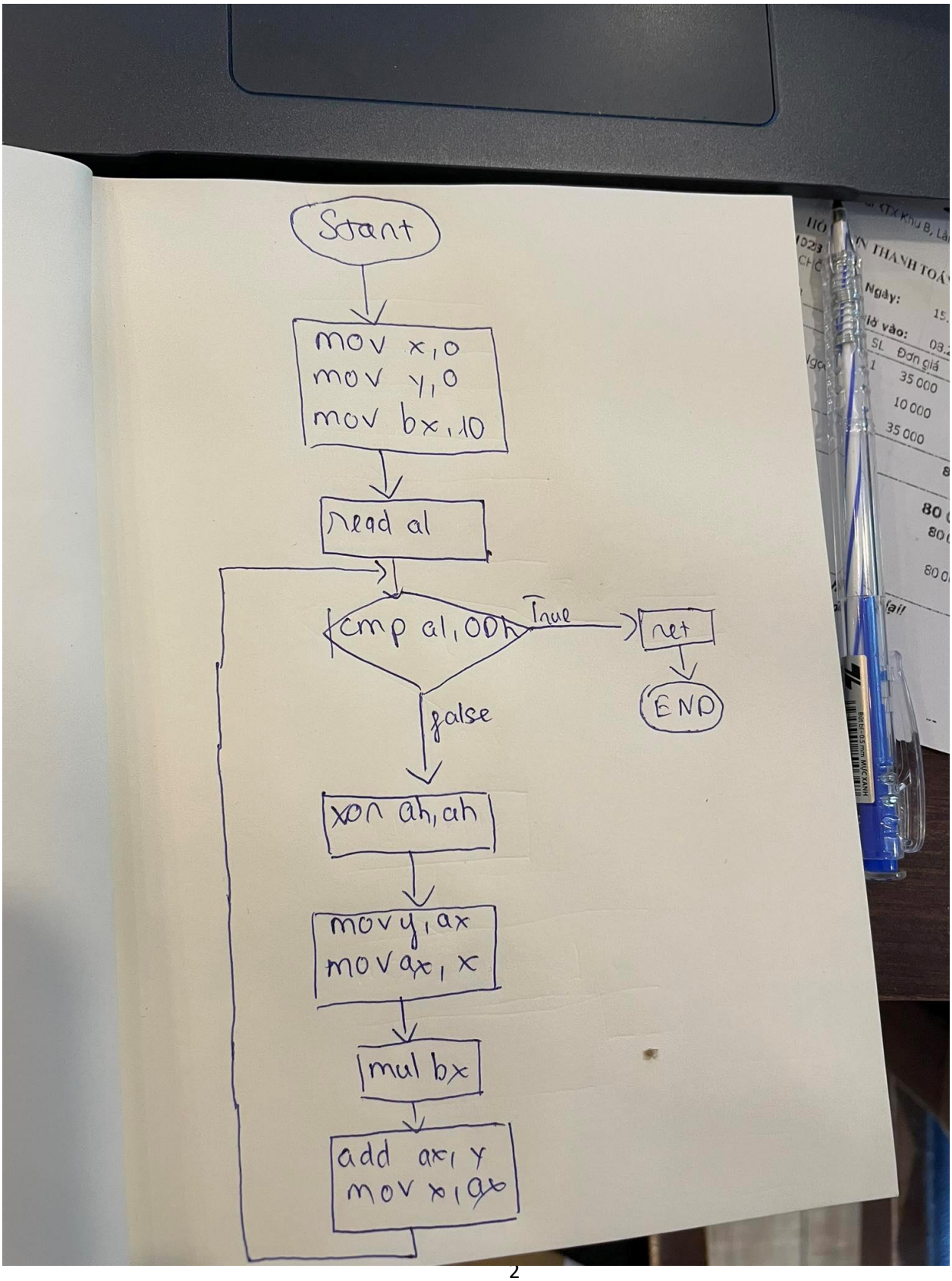
THE MICROPROCESSORS & MICROCONTROLLERS

Instructor : The Tung Than
Student: Bach Hoang Phan Thanh
ID: 21520599

PRATICE EXERCISE #6:
IO PROCESSING, CALCULATION AND MEMORY ON THE 8086 MICROPROCESSOR

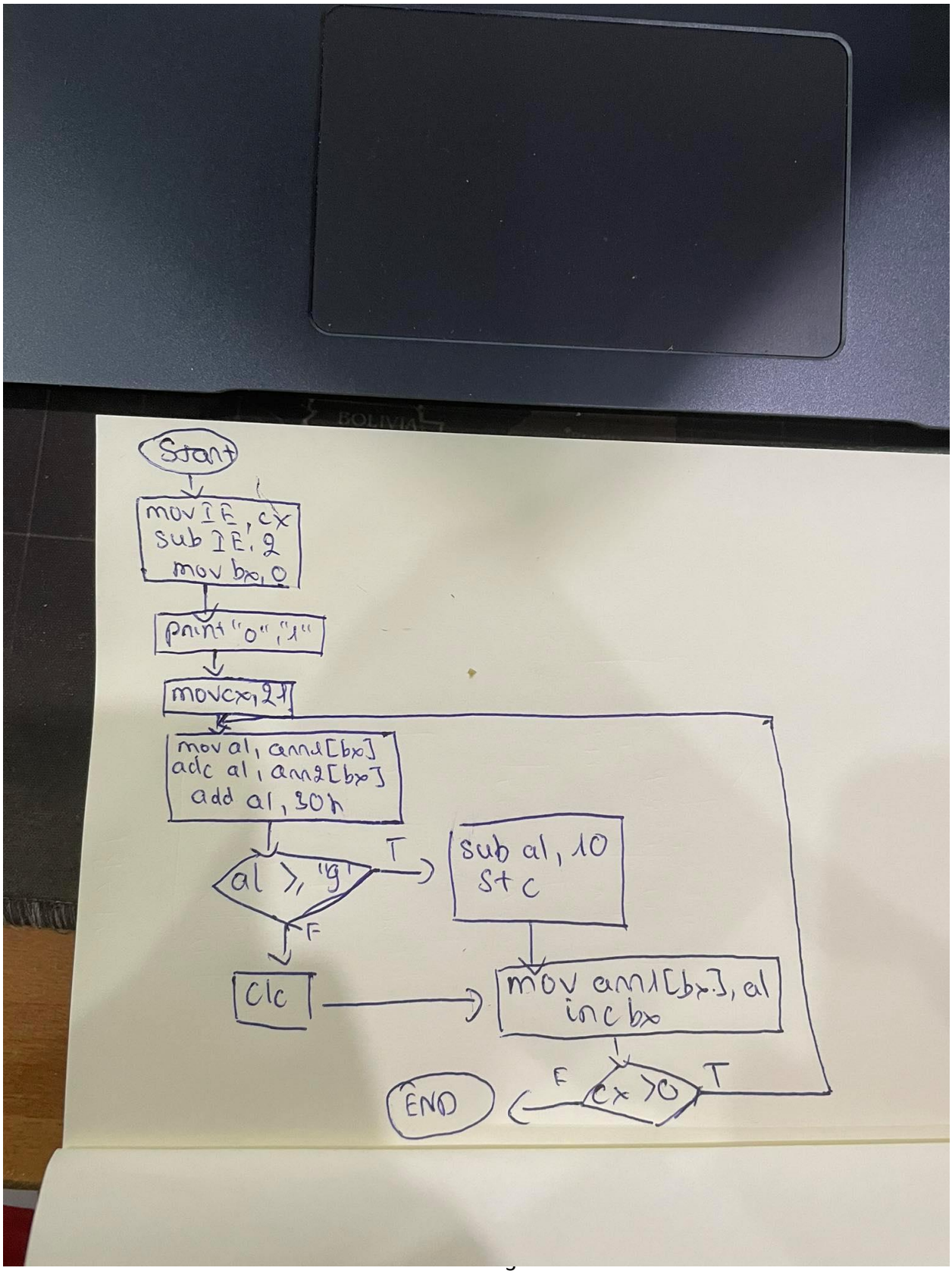
1. Flowchart of the above request processing algorithm

*** Read_Number flowchart:**

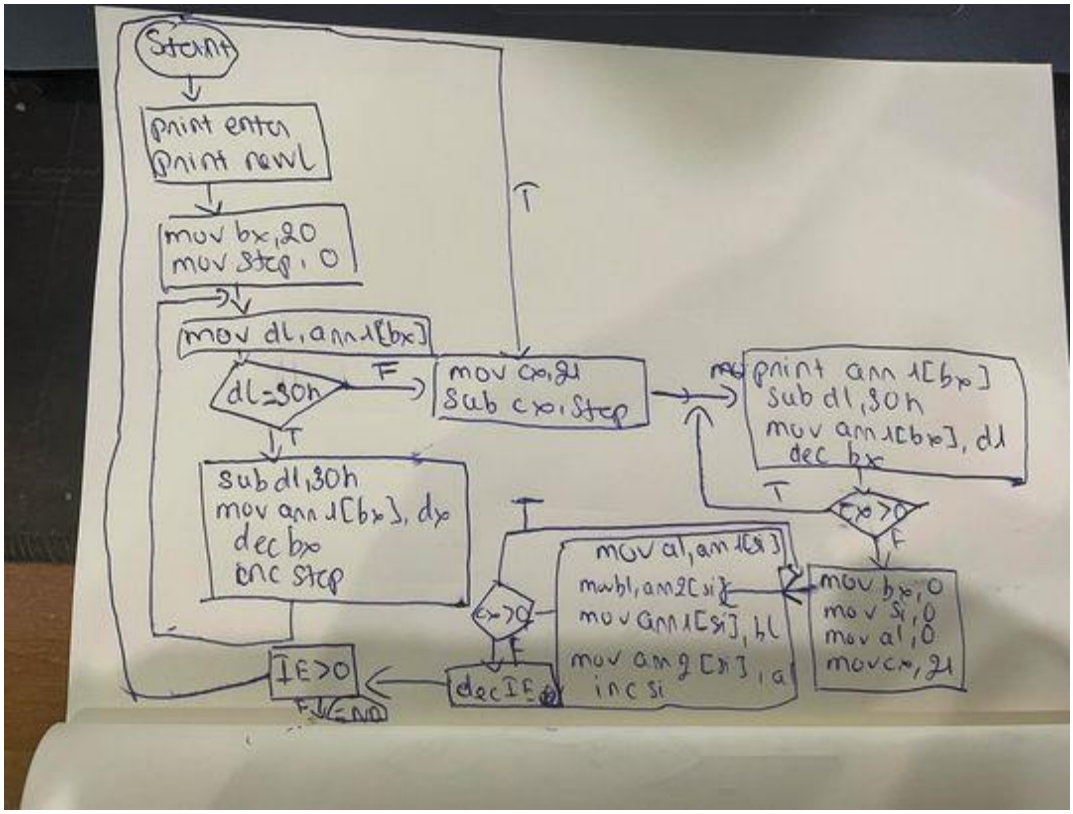


***Fibonacci_series flowchart:**

- Calculate Fibonacci:



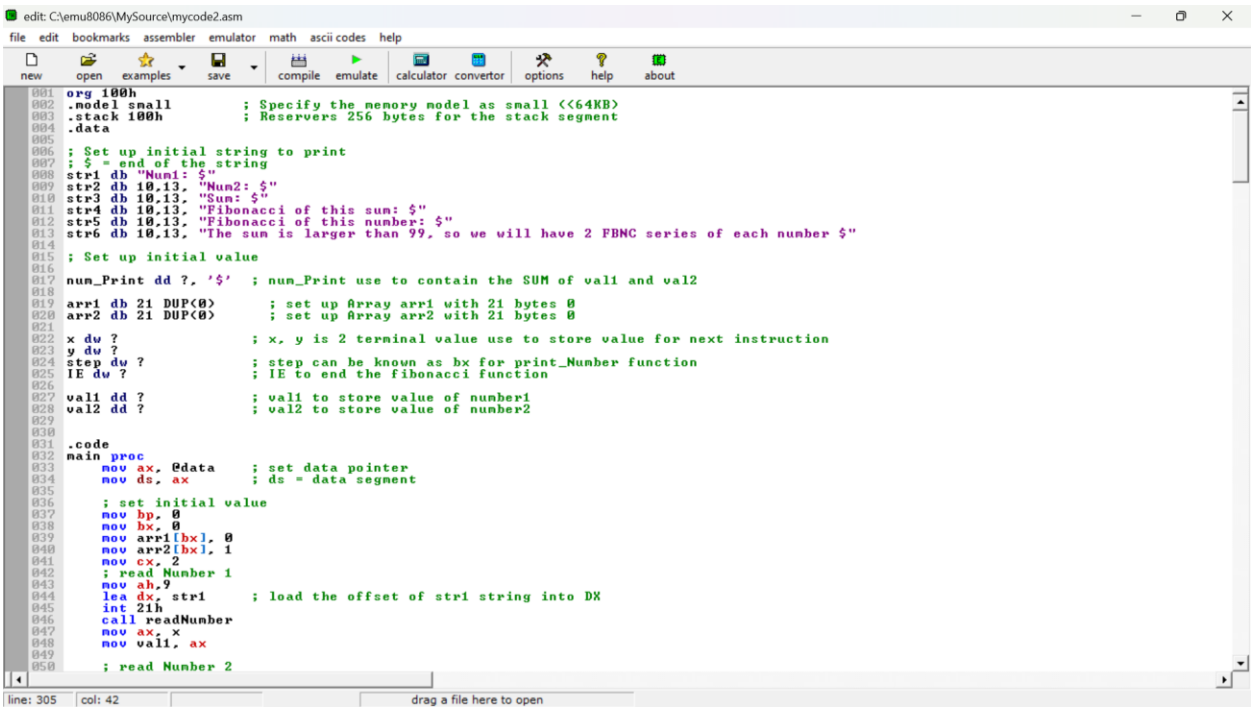
- Print Fibonacci



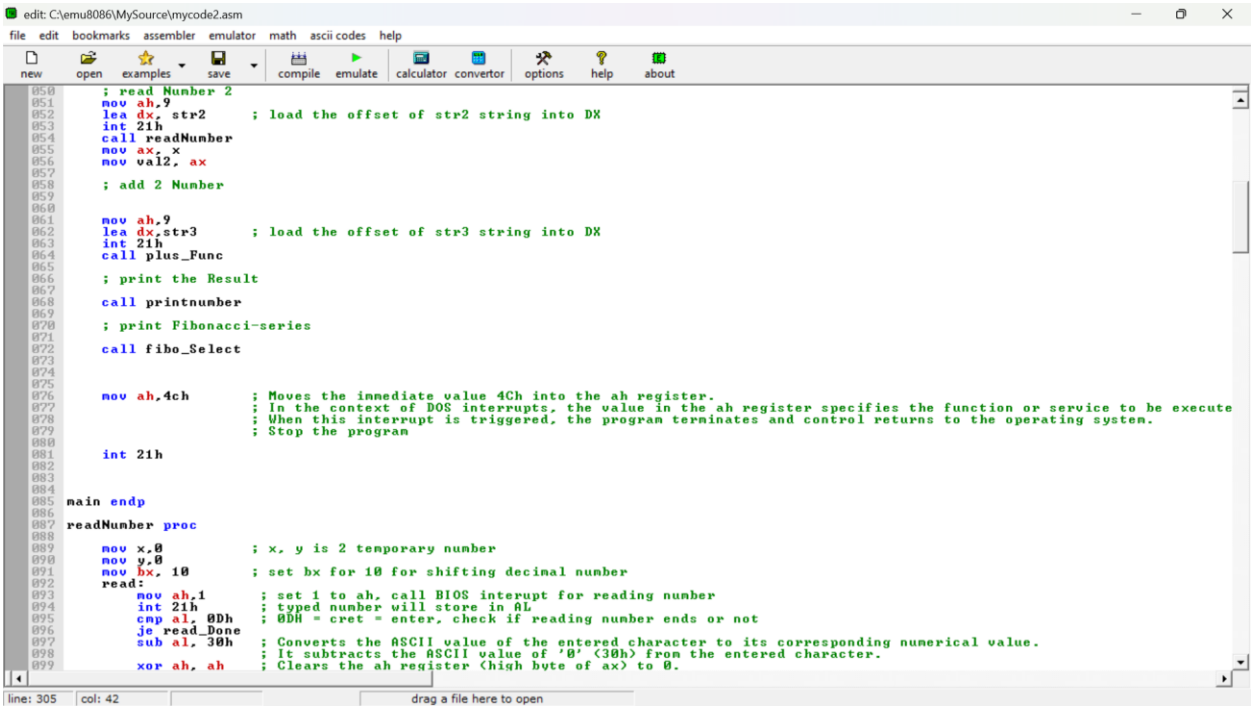
2.Explain how the algorithm works accompanied by a video (send a Google Drive link) to demonstrate the result

Link Result:
https://drive.google.com/drive/folders/1ru9flKq9odJSHx2ue9Gw7KlOSXtLKRuR?usp=sharing

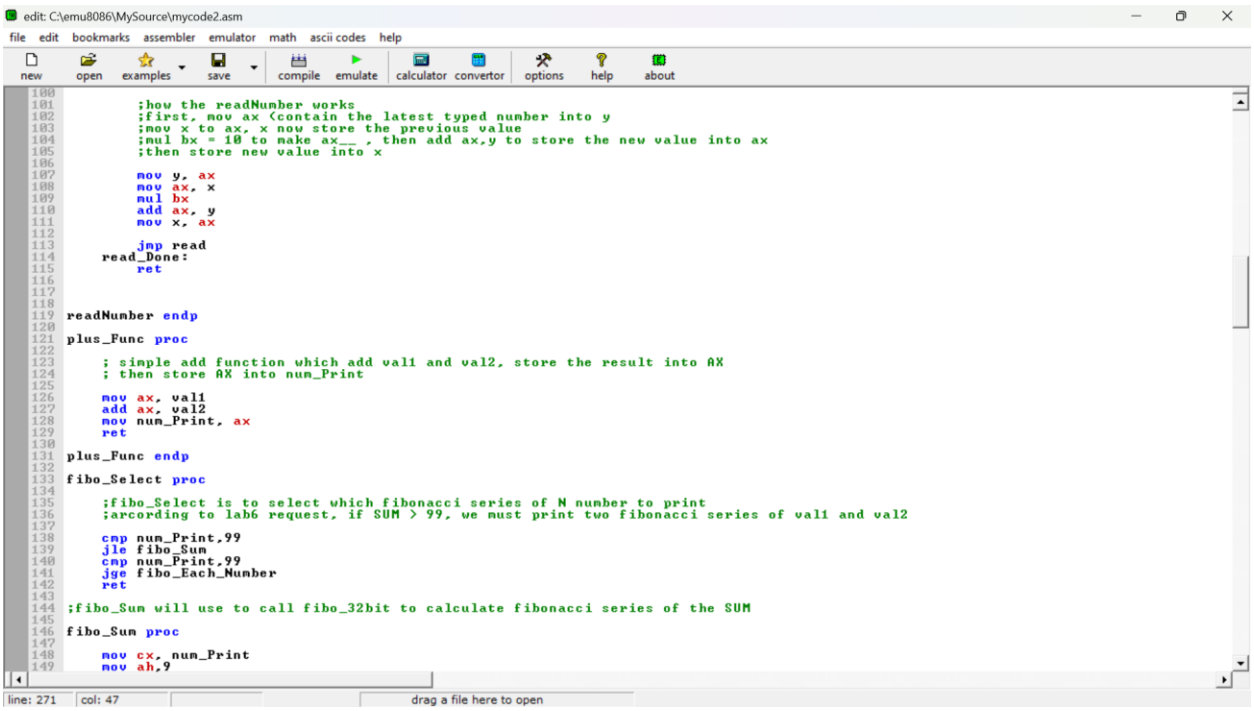
Source code (including English explanation how the algorithm works)

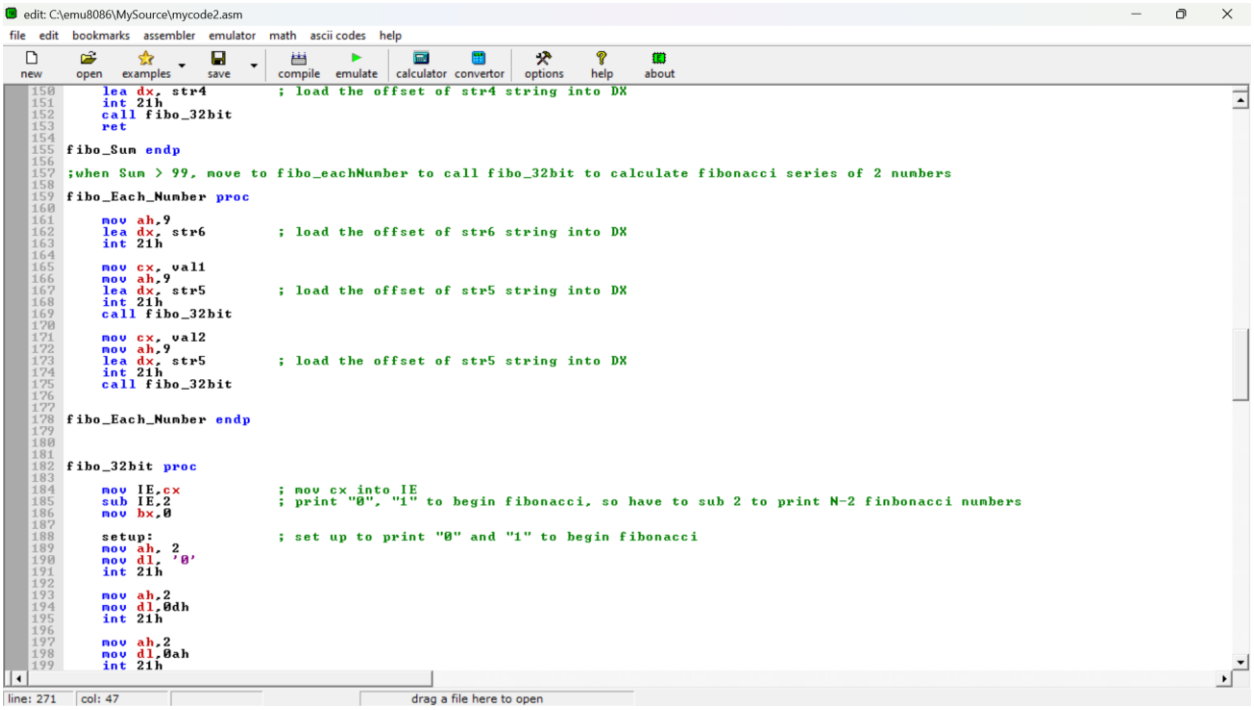


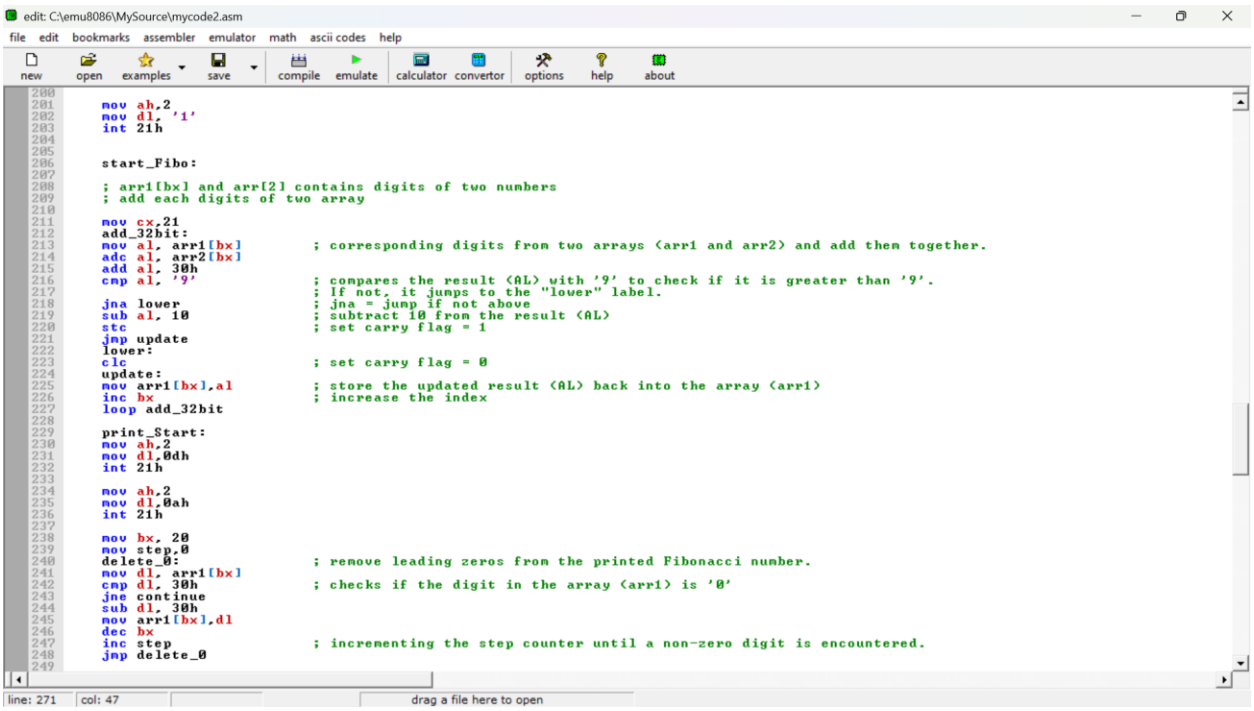

```
org 100h
.model small          ; Specify the memory model as small (<64KB)
.stack 100h           ; Reserves 256 bytes for the stack segment
.data

; Set up initial string to print
; $ = end of the string
str1 db "Num1: $"
str2 db 10,13, "Num2: $"
str3 db 10,13, "Sum: $"
str4 db 10,13, "Fibonacci of this sum: $"
str5 db 10,13, "Fibonacci of this number: $"
str6 db 10,13, "The sum is larger than 99, so we will have 2 FBNC series of each number $"

; Set up initial value

num_Print dd ?, '$'   ; num_Print use to contain the SUM of val1 and val2

arr1 db 21 DUP(0)       ; set up Array arr1 with 21 bytes 0
arr2 db 21 DUP(0)       ; set up Array arr2 with 21 bytes 0

x dw ?                ; x, y is 2 terminal value use to store value for next instruction
y dw ?
step dw ?             ; step can be known as bx for print_Number function
IE dw ?               ; IE to end the fibonacci function

val1 dd ?             ; val1 to store value of number1
val2 dd ?             ; val2 to store value of number2


.code
main proc
    mov ax, @data     ; set data pointer
    mov ds, ax        ; ds = data segment

    ; set initial value
    mov bp, 0
    mov bx, 0
    mov arr1[bx], 0
    mov arr2[bx], 1
    mov cx, 2
    ; read Number 1
    mov ah,9
    lea dx, str1      ; load the offset of str1 string into DX
    int 21h
    call readNumber
    mov ax, x
    mov val1, ax

    ; read Number 2
```

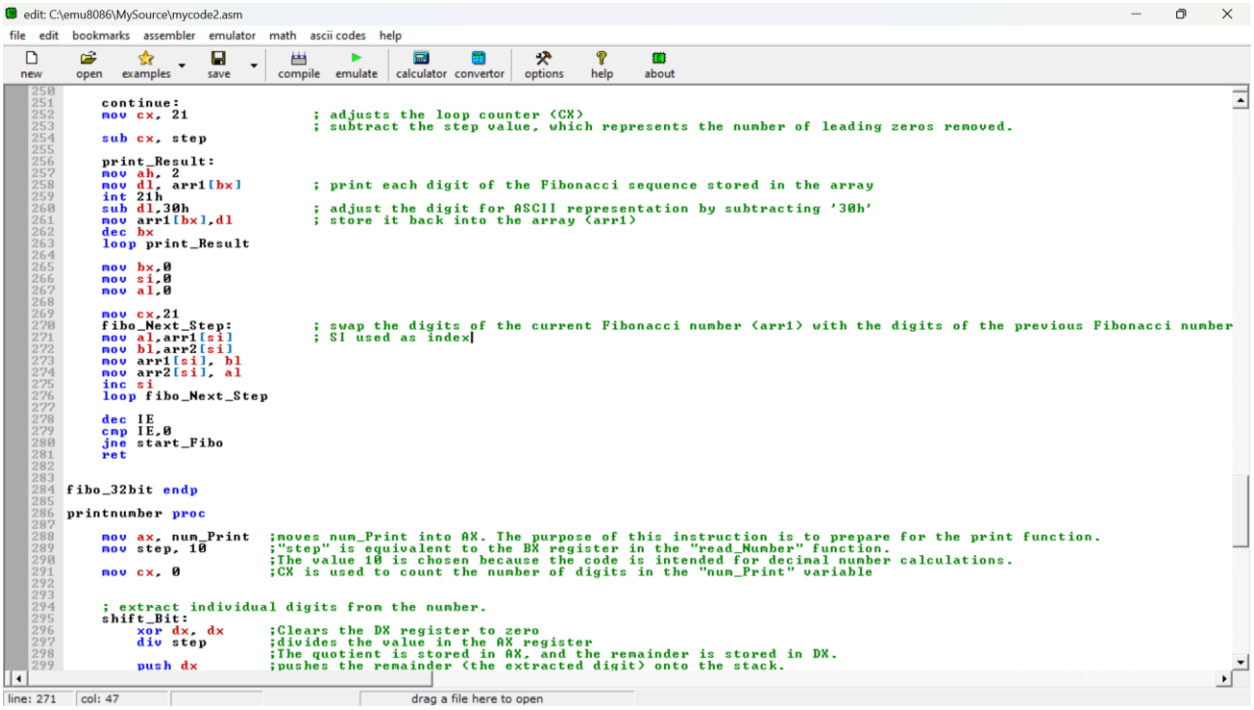


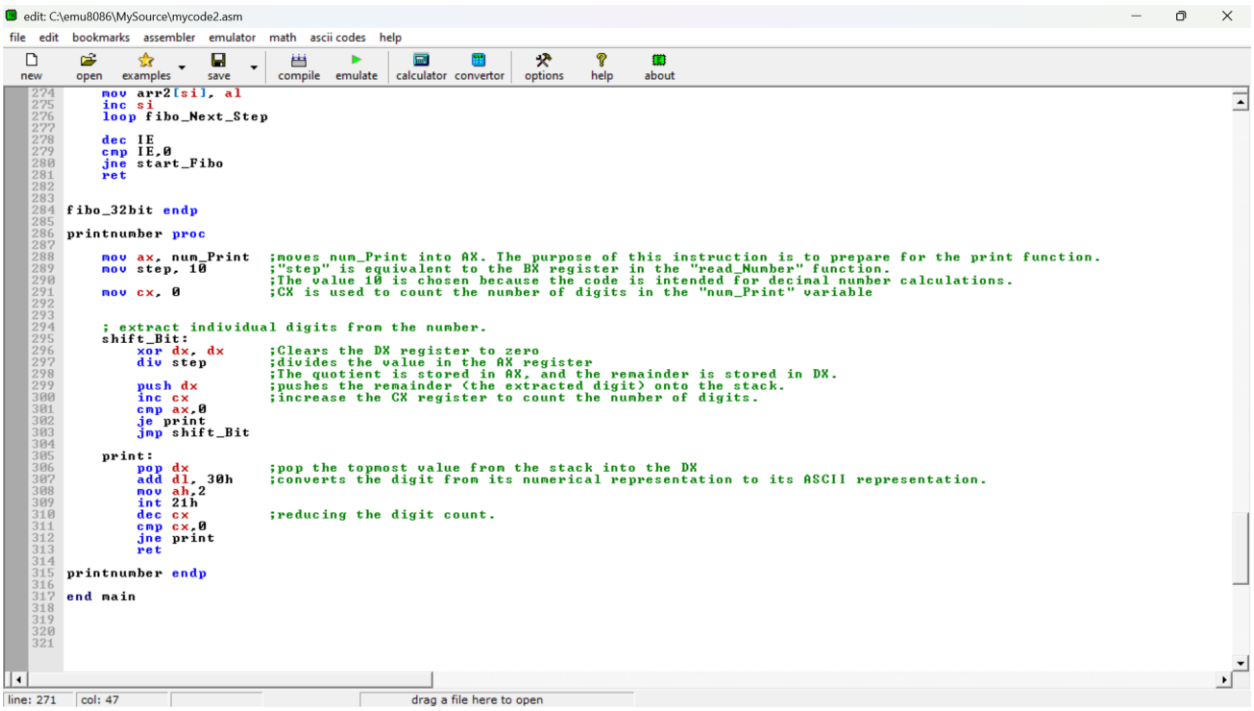

```
    ; read Number 2
    mov ah,9
    lea dx, str2      ; load the offset of str2 string into DX
    int 21h
    call readNumber
    mov ax, x
    mov val2, ax

    ; add 2 Number


    mov ah,9
    lea dx,str3       ; load the offset of str3 string into DX
    int 21h
    call plus_Func

    ; print the Result

    call printnumber

    ; print Fibonacci-series

    call fibo_Select


    mov ah,4ch        ; Moves the immediate value 4Ch into the ah register.
                      ; In the context of DOS interrupts, the value in the ah register specifies the function or service to be execute
                      ; When this interrupt is triggered, the program terminates and control returns to the operating system.
                      ; Stop the program

    int 21h


main endp

readNumber proc

    mov x,0           ; x, y is 2 temporary number
    mov y,0
    mov bx, 10        ; set bx for 10 for shifting decimal number
    read:
        mov ah,1       ; set 1 to ah, call BIOS interupt for reading number
        int 21h        ; typed number will store in AL
        cmp al, 0Dh    ; 0DH = cret = enter, check if reading number ends or not
        je read_Done
        sub al, 30h    ; Converts the ASCII value of the entered character to its corresponding numerical value.
                       ; It subtracts the ASCII value of '0' (30h) from the entered character.
        xor ah, ah     ; Clears the ah register (high byte of ax) to 0.
```


edit: C:\emu8086\MySource\mycode2.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
;how the readNumber works
;first, mov ax contain the latest typed number into y
;mov x to ax, x now store the previous value
;mul bx = 10 to make ax__ , then add ax,y to store the new value into ax
;then store new value into x
mov y, ax
mov ax, x
mul bx
add ax, y
mov x, ax
jmp read
read_Done:
ret
readNumber endp
plus_Func proc
; simple add function which add val1 and val2, store the result into AX
; then store AX into num_Print
mov ax, val1
add ax, val2
mov num_Print, ax
ret
plus_Func endp
fibo_Select proc
;fibo_Select is to select which fibonacci series of N number to print
;arcording to lab6 request, if SUM > 99, we must print two fibonacci series of val1 and val2
cmp num_Print,99
jle fibo_Sum
cmp num_Print,99
jge fibo_Each_Number
ret
;fibo_Sum will use to call fibo_32bit to calculate fibonacci series of the SUM
fibo_Sum proc
mov cx, num_Print
mov ah,9
line: 271 col: 47
drag a file here to open

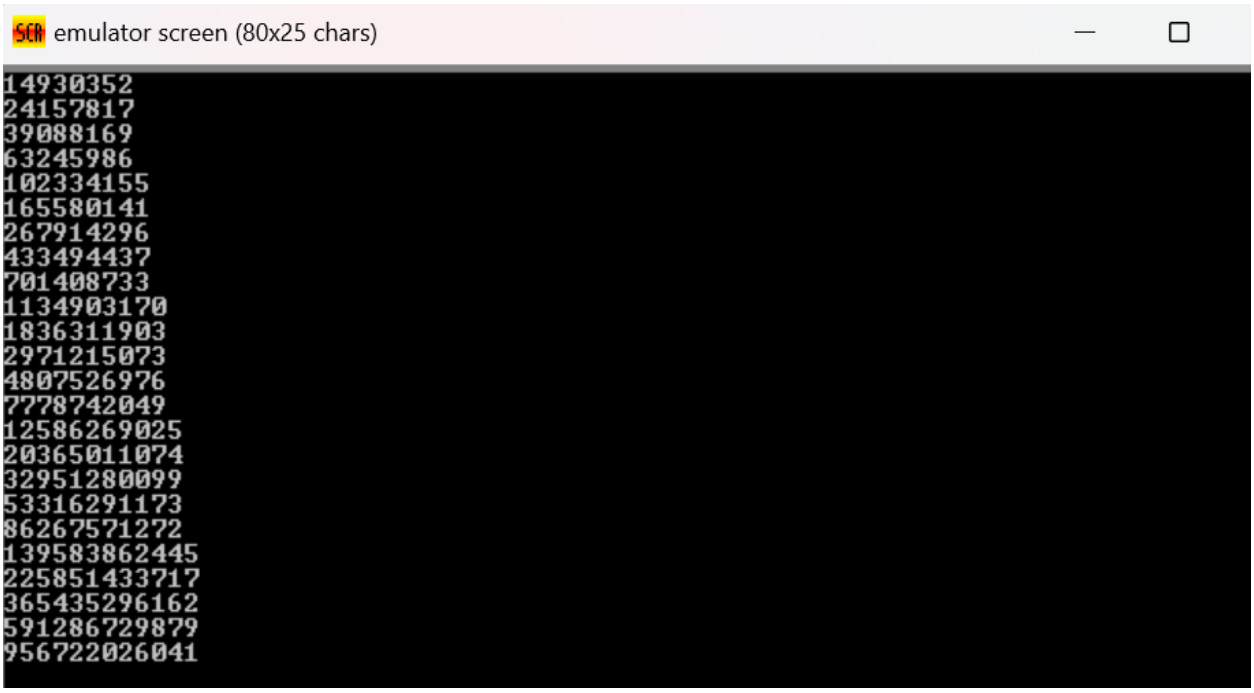

edit: C:\emu8086\MySource\mycode2.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
lea dx, str4 ; load the offset of str4 string into DX
int 21h
call fibo_32bit
ret
fibo_Sum endp
;when Sum > 99, move to fibo_eachNumber to call fibo_32bit to calculate fibonacci series of 2 numbers
fibo_Each_Number proc
mov ah,9
lea dx, str6 ; load the offset of str6 string into DX
int 21h
mov cx, val1
mov ah,9
lea dx, str5 ; load the offset of str5 string into DX
int 21h
call fibo_32bit
mov cx, val2
mov ah,9
lea dx, str5 ; load the offset of str5 string into DX
int 21h
call fibo_32bit
fibo_Each_Number endp
fibo_32bit proc
mov IE,cx ; mov cx into IE
sub IE,2 ; print "0", "1" to begin fibonacci, so have to sub 2 to print N-2 finbonacci numbers
mov bx,0
setup: ; set up to print "0" and "1" to begin fibonacci
mov ah, 2
mov dl, '0'
int 21h
mov ah,2
mov dl,0dh
int 21h
mov ah,2
mov dl,0ah
int 21h
line: 271 col: 47
drag a file here to open


edit: C:\emu8086\MySource\mycode2.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
mov ah,2
mov dl, '1'
int 21h
start_Fibo:
; arr1[bx] and arr[2] contains digits of two numbers
; add each digits of two array
mov cx,21
add_32bit:
mov al, arr1[bx] ; corresponding digits from two arrays (arr1 and arr2) and add them together.
adc al, arr2[bx]
add al, 30h
cmp al, '9' ; compares the result (AL) with '9' to check if it is greater than '9'.
; If not, it jumps to the "lower" label.
jna lower ; jna = jump if not above
sub al, 10 ; subtract 10 from the result (AL)
stc ; set carry flag = 1
jmp update
lower:
clc ; set carry flag = 0
update:
mov arr1[bx],al ; store the updated result (AL) back into the array (arr1)
inc bx ; increase the index
loop add_32bit
print_Start:
mov ah,2
mov dl,0dh
int 21h
mov ah,2
mov dl,0ah
int 21h
mov bx, 20
mov step,0
delete_0: ; remove leading zeros from the printed Fibonacci number.
mov dl, arr1[bx]
cmp dl, 30h ; checks if the digit in the array (arr1) is '0'
jne continue
sub dl, 30h
mov arr1[bx],dl
dec bx
inc step ; incrementing the step counter until a non-zero digit is encountered.
jmp delete_0
line: 271 col: 47
drag a file here to open


edit: C:\emu8086\MySource\mycode2.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
continue:
mov cx, 21 ; adjusts the loop counter (CX)
; subtract the step value, which represents the number of leading zeros removed.
sub cx, step
print_Result:
mov ah, 2
mov dl, arr1[bx] ; print each digit of the Fibonacci sequence stored in the array
int 21h
sub dl,30h ; adjust the digit for ASCII representation by subtracting '30h'
mov arr1[bx],dl ; store it back into the array (arr1)
dec bx
loop print_Result
mov bx,0
mov si,0
mov al,0
mov cx,21
fibo_Next_Step: ; swap the digits of the current Fibonacci number (arr1) with the digits of the previous Fibonacci number
; SI used as index
mov al,arr1[si]
mov bl,arr2[si]
mov arr1[si], bl
mov arr2[si], al
inc si
loop fibo_Next_Step
dec IE
cmp IE,0
jne start_Fibo
ret
fibo_32bit endp
printnumber proc
mov ax, num_Print ;moves num_Print into AX. The purpose of this instruction is to prepare for the print function.
mov step, 10 ;"step" is equivalent to the BX register in the "read_Number" function.
;The value 10 is chosen because the code is intended for decimal number calculations.
mov cx, 0 ;CX is used to count the number of digits in the "num_Print" variable
; extract individual digits from the number.
shift_Bit:
xor dx, dx ;Clears the DX register to zero
div step ;divides the value in the AX register
;The quotient is stored in AX, and the remainder is stored in DX.
push dx ;pushes the remainder (the extracted digit) onto the stack.
line: 271 col: 47
drag a file here to open



Result:



* Source code:

Code
org 100h

```
.model small        ; Specify the memory model as small (<64KB)
.stack 100h         ; Reservers 256 bytes for the stack segment
.data

; Set up initial string to print
; $ = end of the string
str1 db "Num1: $"
str2 db 10,13, "Num2: $"
str3 db 10,13, "Sum: $"
str4 db 10,13, "Fibonacci of this sum: $"
str5 db 10,13, "Fibonacci of this number: $"
str6 db 10,13, "The sum is larger than 99, so we will have 2 FBNC series of
each number $"

; Set up initial value

num_Print dd ?, '$'  ; num_Print use to contain the SUM of val1 and val2

arr1 db 21 DUP(0)      ; set up Array arr1 with 21 bytes 0
arr2 db 21 DUP(0)      ; set up Array arr2 with 21 bytes 0

x dw ?                 ; x, y is 2 terminal value use to store value for next
instruction
y dw ?
step dw ?           ; step can be known as bx for print_Number function
IE dw ?             ; IE to end the fibonacci function

val1 dd ?           ; val1 to store value of number1
val2 dd ?           ; val2 to store value of number2


.code
main proc
   mov ax, @data    ; set data pointer
   mov ds, ax       ; ds = data segment

   ; set initial value
   mov bp, 0
   mov bx, 0
   mov arr1[bx], 0
```

```
    mov arr2[bx], 1
    mov cx, 2
    ; read Number 1
    mov ah,9
    lea dx, str1     ; load the offset of str1 string into DX
    int 21h
    call readNumber
    mov ax, x
    mov val1, ax

    ; read Number 2
    mov ah,9
    lea dx, str2     ; load the offset of str2 string into DX
    int 21h
    call readNumber
    mov ax, x
    mov val2, ax

    ; add 2 Number


    mov ah,9
    lea dx,str3      ; load the offset of str3 string into DX
    int 21h
    call plus_Func

    ; print the Result

    call printnumber

    ; print Fibonacci-series

    call fibo_Select



    mov ah,4ch       ; Moves the immediate value 4Ch into the ah register.
                     ; In the context of DOS interrupts, the value in the ah register specifies the function or service to be executed when invoking the interrupt.
                     ; When this interrupt is triggered, the program terminates and
```

```
control returns to the operating system.
                ; Stop the program

   int 21h



main endp

readNumber proc

   mov x,0         ; x, y is 2 temporary number
   mov y,0
   mov bx, 10      ; set bx for 10 for shifting decimal number
   read:
      mov ah,1     ; set 1 to ah, call BIOS interupt for reading number
      int 21h      ; typed number will store in AL
      cmp al, 0Dh  ; 0DH = cret = enter, check if reading number ends or not
      je read_Done
      sub al, 30h  ; Converts the ASCII value of the entered character to its
corresponding numerical value.
                   ; It subtracts the ASCII value of '0' (30h) from the entered
character.
      xor ah, ah   ; Clears the ah register (high byte of ax) to 0.

      ;how the readNumber works
      ;first, mov ax (contain the latest typed number into y
      ;mov x to ax, x now store the previous value
      ;mul bx = 10 to make ax__ , then add ax,y to store the new value into
ax
      ;then store new value into x

      mov y, ax
      mov ax, x
      mul bx
      add ax, y
      mov x, ax

      jmp read
   read_Done:
```

```
        ret



readNumber endp

plus_Func proc

    ; simple add function which add val1 and val2, store the result into AX
    ; then store AX into num_Print

    mov ax, val1
    add ax, val2
    mov num_Print, ax
    ret

plus_Func endp

fibo_Select proc

    ;fibo_Select is to select which fibonacci series of N number to print
    ;arcording to lab6 request, if SUM > 99, we must print two fibonacci series of val1 and val2

    cmp num_Print,99
    jle fibo_Sum
    cmp num_Print,99
    jge fibo_Each_Number
    ret

;fibo_Sum will use to call fibo_32bit to calculate fibonacci series of the SUM

fibo_Sum proc

    mov cx, num_Print
    mov ah,9
    lea dx, str4        ; load the offset of str4 string into DX
    int 21h
    call fibo_32bit
```

```
   ret

fibo_Sum endp

;when Sum > 99, move to fibo_eachNumber to call fibo_32bit to calculate fibonacci series of 2 numbers

fibo_Each_Number proc

   mov ah,9
   lea dx, str6       ; load the offset of str6 string into DX
   int 21h

   mov cx, val1
   mov ah,9
   lea dx, str5       ; load the offset of str5 string into DX
   int 21h
   call fibo_32bit

   mov cx, val2
   mov ah,9
   lea dx, str5       ; load the offset of str5 string into DX
   int 21h
   call fibo_32bit


fibo_Each_Number endp



fibo_32bit proc

   mov IE,cx          ; mov cx into IE
   sub IE,2           ; print "0", "1" to begin fibonacci, so have to sub 2 to print N-2 finbonacci numbers
   mov bx,0

   setup:             ; set up to print "0" and "1" to begin fibonacci
   mov ah, 2
   mov dl, '0'
```

```
    int 21h

    mov ah,2
    mov dl,0dh
    int 21h

    mov ah,2
    mov dl,0ah
    int 21h

    mov ah,2
    mov dl, '1'
    int 21h


    start_Fibo:

    ; arr1[bx] and arr[2] contains digits of two numbers
    ; add each digits of two array

    mov cx,21
    add_32bit:
    mov al, arr1[bx]          ; corresponding digits from two arrays (arr1 and arr2) and add them together.
    adc al, arr2[bx]
    add al, 30h
    cmp al, '9'               ; compares the result (AL) with '9' to check if it is greater than '9'.
                              ; If not, it jumps to the "lower" label.
    jna lower                 ; jna = jump if not above
    sub al, 10                ; subtract 10 from the result (AL)
    stc                       ; set carry flag = 1
    jmp update
    lower:
    clc                       ; set carry flag = 0
    update:
    mov arr1[bx],al           ; store the updated result (AL) back into the array (arr1)
    inc bx                    ; increase the index
    loop add_32bit
```

```
print_Start:
mov ah,2
mov dl,0dh
int 21h

mov ah,2
mov dl,0ah
int 21h

mov bx, 20
mov step,0
delete_0:                ; remove leading zeros from the printed Fibonacci number.
mov dl, arr1[bx]
cmp dl, 30h          ; checks if the digit in the array (arr1) is '0'
jne continue
sub dl, 30h
mov arr1[bx],dl
dec bx
inc step             ; incrementing the step counter until a non-zero digit is encountered.
jmp delete_0


continue:
mov cx, 21           ; adjusts the loop counter (CX)
                     ; subtract the step value, which represents the number of leading zeros removed.
sub cx, step

print_Result:
mov ah, 2
mov dl, arr1[bx]       ; print each digit of the Fibonacci sequence stored in the array
int 21h
sub dl,30h                   ; adjust the digit for ASCII representation by subtracting '30h'
mov arr1[bx],dl        ; store it back into the array (arr1)
dec bx
```

```
    loop print_Result

    mov bx,0
    mov si,0
    mov al,0

    mov cx,21
    fibo_Next_Step:          ; swap the digits of the current Fibonacci number (arr1) with the digits of the previous Fibonacci number (arr2) to prepare for the next Fibonacci calculation
    mov al,arr1[si]        ; SI used as index
    mov bl,arr2[si]
    mov arr1[si], bl
    mov arr2[si], al
    inc si
    loop fibo_Next_Step

    dec IE
    cmp IE,0
    jne start_Fibo
    ret


fibo_32bit endp

printnumber proc

    mov ax, num_Print  ;moves num_Print into AX. The purpose of this instruction is to prepare for the print function.
    mov step, 10           ;"step" is equivalent to the BX register in the "read_Number" function.
                    ;The value 10 is chosen because the code is intended for decimal number calculations.
    mov cx, 0              ;CX is used to count the number of digits in the "num_Print" variable



    ; extract individual digits from the number.
    shift_Bit:
       xor dx, dx     ;Clears the DX register to zero
```

```
        div step       ;divides the value in the AX register
                       ;The quotient is stored in AX, and the remainder is stored in DX.
        push dx        ;pushes the remainder (the extracted digit) onto the stack.
        inc cx         ;increase the CX register to count the number of digits.
        cmp ax,0
        je print
        jmp shift_Bit

    print:
        pop dx         ;pop the topmost value from the stack into the DX
        add dl, 30h    ;converts the digit from its numerical representation to its ASCII representation.
        mov ah,2
        int 21h
        dec cx         ;reducing the digit count.
        cmp cx,0
        jne print
        ret

printnumber endp

end main
```

- Gỉai thích Tiếng Việt:

Trong chương trình in ra N số fibonacci, thay vì chọn cách in ra kết quả của các số Fibonacci, em chọn in ra từng chữ số.

Nếu ghép các thanh ghi thì ta có thể ra được kết quả lên đến 32-bit, nhưng số thứ 99 của dãy Fibonacci cho kết quả lên đến hơn 2^66, cũng hơn 10^20. Vì thế em chọn dung mảng có 21 phần tử để in ra từng chữ số kết quả.

Dù thời gian thực thi sẽ lâu hơn, nhưng nó giúp mình in được tới số Fibonacci thứ 99 bằng cách in được chữ số, cộng từng chữ số. Quy tắc vẫn đảm bảo tính chất của dãy số Fibonacci

Method 2: (All set up, print and read Number is the same as Method 1)
Giải thích tiếng Việt:

- Thay vì in từng chữ số, cách thứ 2 sẽ tiết kiệm thời gian bằng cách in trực tiếp kết quả. Tuy nhiên phương pháp này không thể in được tới số Fibonacci thứ 99 vì thanh ghi không chứa được hết

```
fbnc proc

    mov ax, 0
    mov bx, 1
    mov cx, num_Print
    mov IE, cx    ; IE = N-fibonacci number to print
    print_fbnc:
        add ax, bx   ; add ax and bx, store in AX
        mov num_Print, ax ; store the value of add func to num_print
        call printnumber
        mov dx, ' '      ; print ' ' (SPACE)
        mov ah,2
        int 21h
        mov ax, bx       ; revert number, prepare for next fibonacci calculation
        mov bx, num_Print
        dec IE
        cmp IE,0
        jne print_fbnc
        ret
    ret

fbnc endp
```