

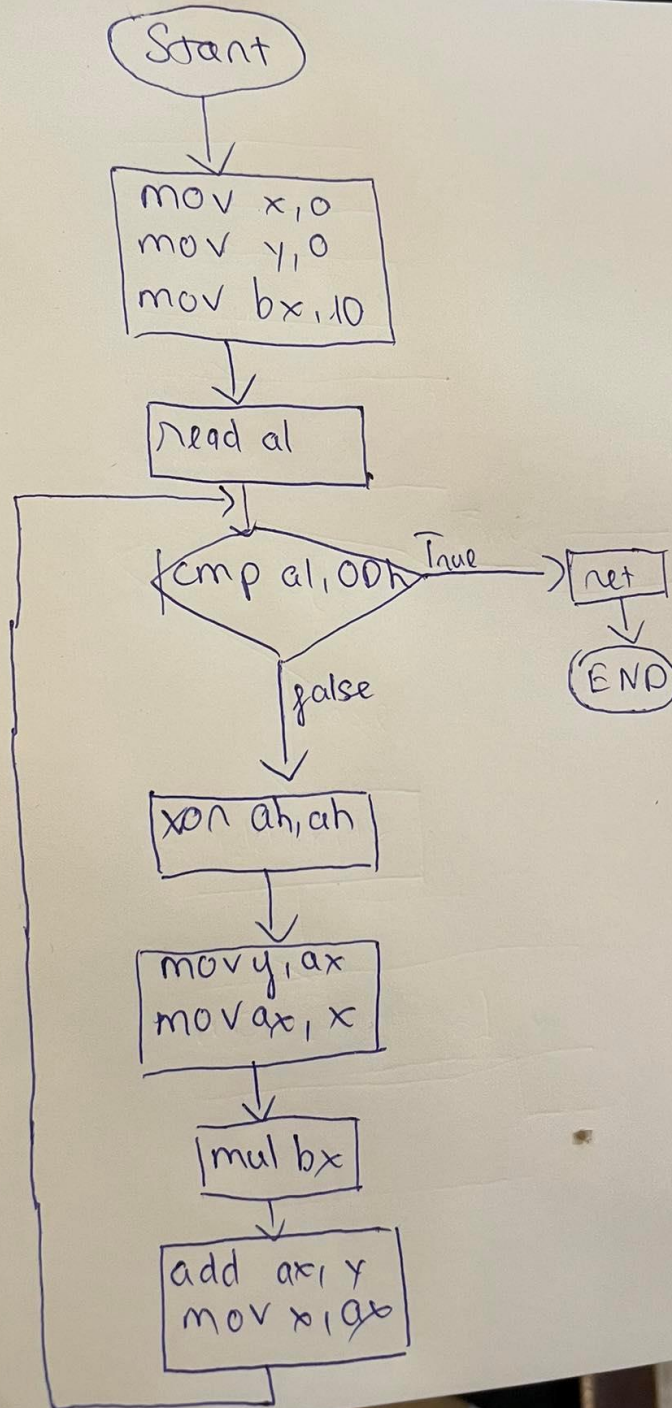
THE MICROPROCESSORS & MICROCONTROLLERS

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

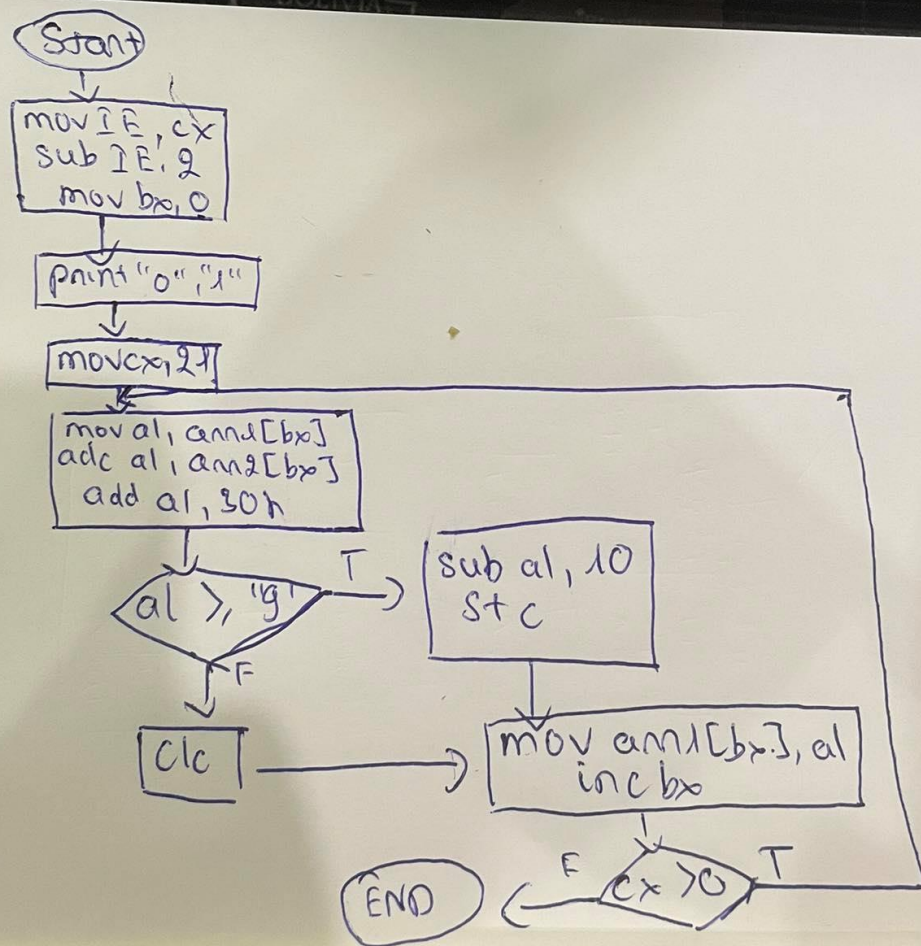
PRATICE EXERCISE #6:
IO PROCESSING, CALCULATION AND MEMORY ON THE 8086
MICROPROCESSOR

1. Flowchart of the above request processing algorithm

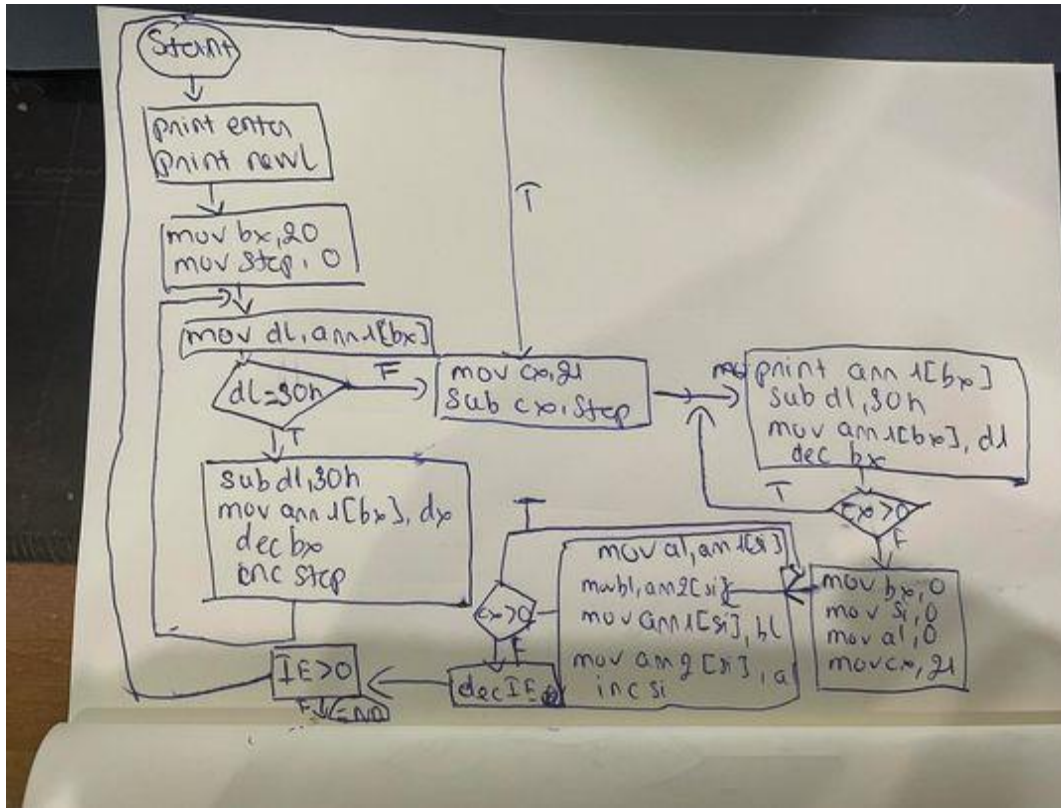
* Read_Number flowchart:



***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/1ru9flKq9odJSHx2ue9Gw7KIOSXtLKRuR?usp=sharing>

Source code (including English explanation how the algorithm works)

```

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

0001 org 100h
0002 .model small ; Specify the memory model as small <<64KB>
0003 .stack 100h ; Reserves 256 bytes for the stack segment
0004 .data
0005 ; Set up initial string to print
0006 ; $ = end of the string
0007 str1 db "Num1: $"
0008 str2 db 10,13, "Num2: $"
0009 str3 db 10,13, "Sum: $"
0010 str4 db 10,13, "Fibonacci of this sum: $"
0011 str5 db 10,13, "Fibonacci of this number: $"
0012 str6 db 10,13, "The sum is larger than 99, so we will have 2 FBNC series of each number $"
0013
0014 ; Set up initial value
0015 num_Print dd ?, '$' ; num_Print use to contain the SUM of val1 and val2
0016
0017 arr1 db 21 DUP(0) ; set up Array arr1 with 21 bytes 0
0018 arr2 db 21 DUP(0) ; set up Array arr2 with 21 bytes 0
0019
0020 x dw ? ; x, y is 2 terminal value use to store value for next instruction
0021 y dw ?
0022 step dw ? ; step can be known as bx for print_Number function
0023 IE dw ? ; IE to end the fibonacci function
0024
0025 val1 dd ? ; val1 to store value of number1
0026 val2 dd ? ; val2 to store value of number2
0027
0028
0029
0030
0031 .code
0032 main proc
0033 mov ax, 0data ; set data pointer
0034 mov ds, ax ; ds = data segment
0035
0036 ; set initial value
0037 mov bp, 0
0038 mov bx, 0
0039 mov arr1[bx], 0
0040 mov arr2[bx], 1
0041 mov cx, 2
0042 ; read Number 1
0043 mov ah, 9
0044 lea dx, str1 ; load the offset of str1 string into DX
0045 int 21h
0046 call readNumber
0047 mov ax, x
0048 mov val1, ax
0049
0050 ; read Number 2

```

```

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

0050 ; read Number 2
0051 mov ah, 9
0052 lea dx, str2 ; load the offset of str2 string into DX
0053 int 21h
0054 call readNumber
0055 mov ax, x
0056 mov val2, ax
0057
0058 ; add 2 Number
0059
0060
0061 mov ah, 9
0062 lea dx, str3 ; load the offset of str3 string into DX
0063 int 21h
0064 call plus_Func
0065
0066 ; print the Result
0067 call printnumber
0068
0069 ; print Fibonacci-series
0070 call fibo_Select
0071
0072
0073
0074
0075
0076 mov ah, 4ch ; Moves the immediate value 4Ch into the ah register.
0077 ; In the context of DOS interrupts, the value in the ah register specifies the function or service to be execute
0078 ; When this interrupt is triggered, the program terminates and control returns to the operating system.
0079 ; Stop the program
0080
0081 int 21h
0082
0083
0084
0085 main endp
0086
0087 readNumber proc
0088
0089 mov x, 0 ; x, y is 2 temporary number
0090 mov y, 0
0091 mov bx, 10 ; set bx for 10 for shifting decimal number
0092 read:
0093 mov ah, 1 ; set 1 to ah, call BIOS interrupt for reading number
0094 int 21h ; typed number will store in AL
0095 cmp al, 0Dh ; 0Dh = cret = enter, check if reading number ends or not
0096 je read_Done
0097 sub al, 30h ; Converts the ASCII value of the entered character to its corresponding numerical value.
0098 ; It subtracts the ASCII value of '0' (30h) from the entered character.
0099 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 convert options help about

100
101 ;how the readNumber works
102 ;first, mov ax contain the latest typed number into y
103 ;mov x to ax, x now store the previous value
104 ;mul bx = 10 to make ax__, then add ax,y to store the new value into ax
105 ;then store new value into x
106
107     mov y, ax
108     mov ax, x
109     mul bx
110     add ax, y
111     mov x, ax
112
113     jmp read
114 read_Done:
115     ret
116
117
118 readNumber endp
119
120 plus_Func proc
121
122 ; simple add function which add val1 and val2, store the result into AX
123 ; then store AX into num_Print
124
125     mov ax, val1
126     add ax, val2
127     mov num_Print, ax
128     ret
129
130 plus_Func endp
131
132 fibo_Select proc
133
134 ;fibo_Select is to select which fibonacci series of N number to print
135 ;according to lab6 request, if SUM > 99, we must print two fibonacci series of val1 and val2
136
137     cmp num_Print, 99
138     jle fibo_Sum
139     cmp num_Print, 99
140     jge fibo_Each_Number
141     ret
142
143 ;fibo_Sum will use to call fibo_32bit to calculate fibonacci series of the SUM
144
145 fibo_Sum proc
146     mov cx, num_Print
147     mov ah, 9
148
149

```

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 convert options help about

150
151     lea dx, str4 ; load the offset of str4 string into DX
152     int 21h
153     call fibo_32bit
154     ret
155
156 fibo_Sum endp
157
158 ;when Sum > 99, move to fibo_eachNumber to call fibo_32bit to calculate fibonacci series of 2 numbers
159
160 fibo_Each_Number proc
161
162     mov ah, 9
163     lea dx, str6 ; load the offset of str6 string into DX
164     int 21h
165
166     mov cx, val1
167     mov ah, 9
168     lea dx, str5 ; load the offset of str5 string into DX
169     int 21h
170     call fibo_32bit
171
172     mov cx, val2
173     mov ah, 9
174     lea dx, str5 ; load the offset of str5 string into DX
175     int 21h
176     call fibo_32bit
177
178 fibo_Each_Number endp
179
180
181 fibo_32bit proc
182
183     mov IE, cx ; mov cx into IE
184     sub IE, 2 ; print "0", "1" to begin fibonacci, so have to sub 2 to print N-2 fibonacci numbers
185     mov bx, 0
186
187     setup:
188     mov ah, 2 ; set up to print "0" and "1" to begin fibonacci
189     mov dl, '0'
190     int 21h
191
192     mov ah, 2
193     mov dl, 0dh
194     int 21h
195
196     mov ah, 2
197     mov dl, 0ah
198     int 21h
199

```

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 convert options help about

200
201    mov ah,2
202    mov dl,'1'
203    int 21h
204
205    start_Fibo:
206    ; arr1[bx] and arr2[2] contains digits of two numbers
207    ; add each digits of two array
208
209    mov cx,21
210    add_32bit:
211    mov al, arr1[bx]      ; corresponding digits from two arrays (arr1 and arr2) and add them together.
212    adc al, arr2[bx]
213    cmp al, 90h           ; compares the result (AL) with '9' to check if it is greater than '9'.
214    jna lower            ; If not, it jumps to the "lower" label.
215    sub al, 10            ; jna = jump if not above
216    stc                  ; subtract 10 from the result (AL)
217    jmp update           ; set carry flag = 1
218
219    lower:
220    clc                  ; set carry flag = 0
221    update:
222    mov arr1[bx],al      ; store the updated result (AL) back into the array (arr1)
223    inc bx               ; increase the index
224    loop add_32bit
225
226    print_Start:
227    mov ah,2
228    mov dl,0dh
229    int 21h
230
231    mov ah,2
232    mov dl,0ah
233    int 21h
234
235    mov bx, 20
236    mov step,0
237    delete_0:
238    mov dl, arr1[bx]      ; remove leading zeros from the printed Fibonacci number.
239    cmp dl, 30h           ; checks if the digit in the array (arr1) is '0'
240    jne continue
241    sub dl, 30h
242    mov arr1[bx],dl
243    dec bx
244    inc step              ; incrementing the step counter until a non-zero digit is encountered.
245    jmp delete_0
246
247
248
249
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 convert options help about

250
251    continue:
252    mov cx, 21            ; adjusts the loop counter (CX)
253    sub cx, step          ; subtract the step value, which represents the number of leading zeros removed.
254
255    print_Result:
256    mov ah, 2
257    mov dl, arr1[bx]      ; print each digit of the Fibonacci sequence stored in the array
258    int 21h
259    sub dl,30h            ; adjust the digit for ASCII representation by subtracting '30h'
260    mov arr1[bx],dl       ; store it back into the array (arr1)
261    dec bx
262    loop print_Result
263
264    mov bx,0
265    mov si,0
266    mov al,0
267
268    mov cx,21
269    fibo_Next_Step:
270    ; swap the digits of the current Fibonacci number (arr1) with the digits of the previous Fibonacci number
271    ; SI used as index
272    mov bl, arr2[si]
273    mov arr1[si], bl
274    mov arr2[si], al
275    inc si
276    loop fibo_Next_Step
277
278    dec IE
279    cmp IE,0
280    jne start_Fibo
281    ret
282
283    fibo_32bit endp
284
285    printnumber proc
286
287    mov ax, num_Print     ;moves num_Print into AX. The purpose of this instruction is to prepare for the print function.
288    mov step, 10          ;"step" is equivalent to the BX register in the "read_Number" function.
289    ;The value 10 is chosen because the code is intended for decimal number calculations.
290    mov cx, 0             ;CX is used to count the number of digits in the "num_Print" variable
291
292    ; extract individual digits from the number.
293    shift_Bit:
294    xor dx, dx            ;Clears the DX register to zero
295    div step              ;divides the value in the AX register
296    ;The quotient is stored in AX, and the remainder is stored in DX.
297    push dx              ;pushes the remainder (the extracted digit) onto the stack.
298
299
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
274 nov arr2[s1], al
275 inc si
276 loop fibo_Next_Step
277
278 dec IE
279 cmp IE, 0
280 jne start_Fibo
281 ret
282
283 fibo_32bit endp
284
285 printnumber proc
286
287
288 nov ax, num_Print ;moves num_Print into AX. The purpose of this instruction is to prepare for the print function.
289 nov step, 10 ;"step" is equivalent to the BX register in the "read_Number" function.
290 ;The value 10 is chosen because the code is intended for decimal number calculations.
291 nov cx, 0 ;CX is used to count the number of digits in the "num_Print" variable
292
293 ; extract individual digits from the number.
294 shift_Bit:
295
296 xor dx, dx ;Clears the DX register to zero
297 div step ;divides the value in the AX register
298 ;The quotient is stored in AX, and the remainder is stored in DX.
299 push dx ;pushes the remainder (the extracted digit) onto the stack.
300 inc cx ;increase the CX register to count the number of digits.
301 cmp ax, 0
302 je print
303 jmp shift_Bit
304
305 print:
306 pop dx ;pop the topmost value from the stack into the DX
307 add dl, 30h ;converts the digit from its numerical representation to its ASCII representation.
308 mov ah, 2
309 int 21h
310 dec cx ;reducing the digit count.
311 cmp cx, 0
312 jne print
313 ret
314
315 printnumber endp
316
317 end main
318
319
320
321

```

Result:

```

SCM emulator screen (80x25 chars)
14930352
24157817
39088169
63245986
102334155
165580141
267914296
433494437
701408733
1134903170
1836311903
2971215073
4807526976
7778742049
12586269025
20365011074
32951280099
53316291173
86267571272
139583862445
225851433717
365435296162
591286729879
956722026041

```

* Source code:

Code
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
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 interrupt 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
    ;according 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

fibonacci_Sum endp

;when Sum > 99, move to fibonacci_eachNumber to call fibonacci_32bit to calculate
;fibonacci series of 2 numbers

fibonacci_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 fibonacci_32bit

    mov cx, val2
    mov ah,9
    lea dx, str5      ; load the offset of str5 string into DX
    int 21h
    call fibonacci_32bit

fibonacci_Each_Number endp

fibonacci_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 fibonacci 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

```

- **Giai 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
```