THỰC HÀNH VI XỬ LÝ – VI ĐIỀU KHIỂN

GVHD: Bùi Phùng Hữu Đức

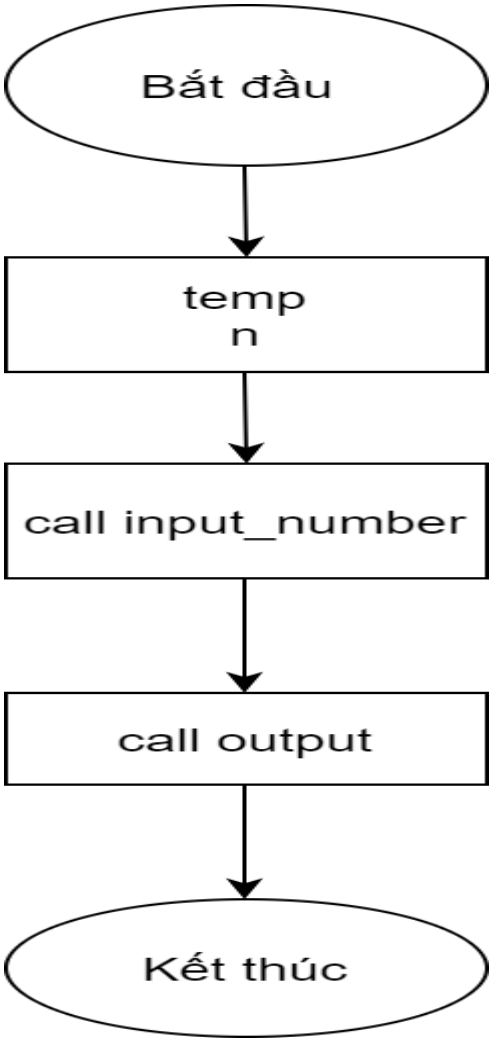
Họ và tên sinh viên thực hiện: Nguyễn Hữu Tứ

Mã số sinh viên: 19522453

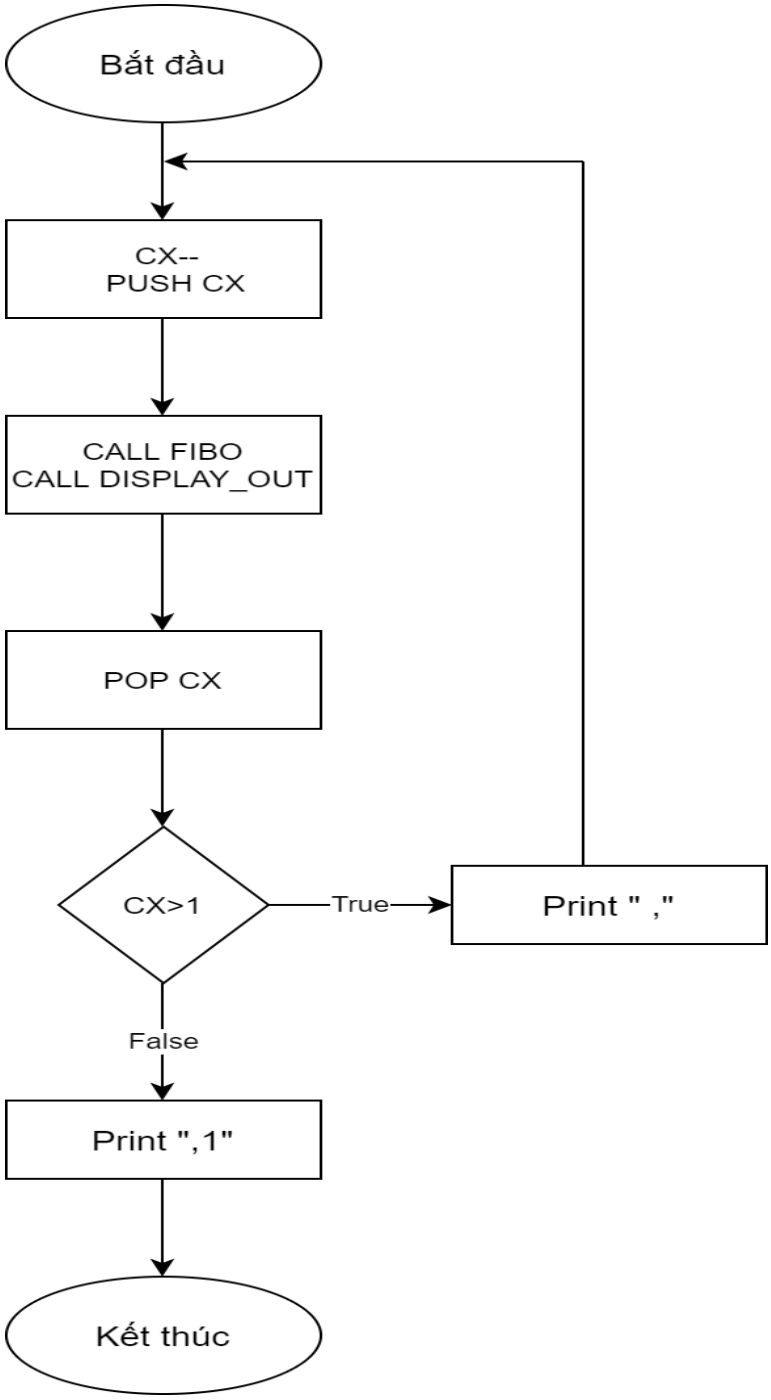
BÁO CÁO THỰC HÀNH SỐ 06
XỬ LÝ IO, TÍNH TOÁN VÀ BỘ NHỚ TRÊN 8086

I. Nhập vào một số nguyên N và in ra N số fibonaci đầu tiên :

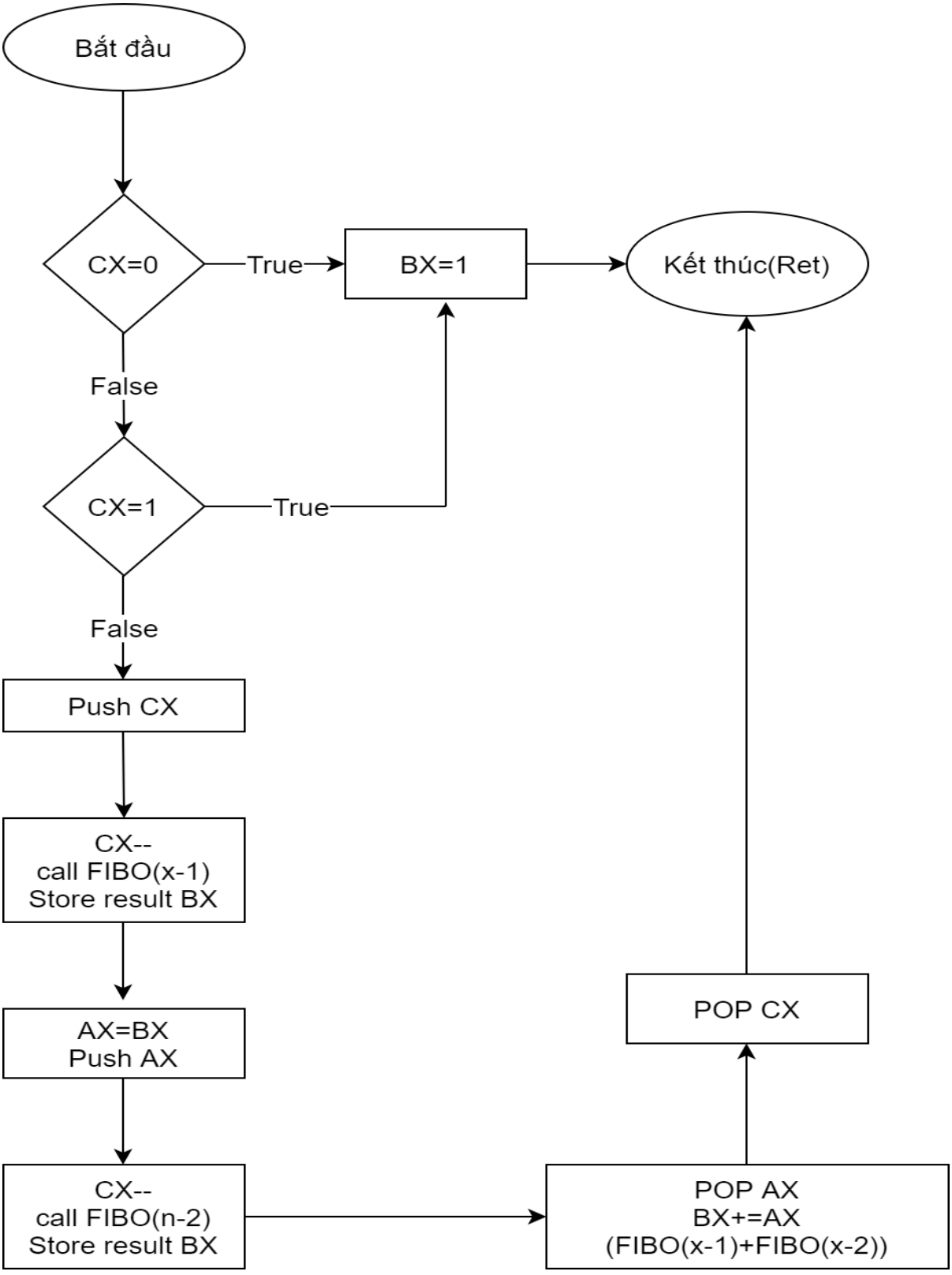
1. Lưu đồ thuật toán



Hình 1 : Lưu đồ hàm main



Hình 2 :Sơ đồ hàm Output



Hình 3 :Sơ đồ hàm FIBO

Code	Giải thích
.MODEL SMALL .STACK 100H	

.DATA	Biến temp lưu giá trị tạm với không có giá trị khởi tạo.
temp DW ?	
tb1 db "Nhap N :$"	Chuỗi tb1 thông báo nhập n
under_line DB 0AH,0DH,"$"	Chuỗi under_line dùng để xuống dòng
.CODE	.Phần code
MAIN PROC	Hàm main
MOV AX,@DATA	
MOV DS,AX	
MOV AH,9	Dùng hàm ngắt để xuất ra chuỗi thông báo nhập n
LEA DX,tb1	
INT 21h	
MOV AH,01H	
CALL INPUT_NUMBER	Hàm nhập số cần tính toán
MOV AH,9	In ra kí tự xuống dòng để xuất ra chuỗi các số
LEA DX,under_line	
INT 21h	Ta dùng dùng ngắt
DB 13,10,0	
CALL OUTPUT	Hàm output xuất giá trị và tính toán
MAIN ENDP	Kết thúc hàm main
;//////////////////////	
DISPLAY_ PROC	Hàm dùng để display
MOV CS:temp, SI	Gán temp = SI
POP SI	Lấy giá trị vào SI
PUSH AX	Đưa giá trị AX vào stack
nextchar:	
MOV AL, CS:[SI]	Gán AL = temp[SI]
INC SI	Tăng SI (tăng địa chỉ)
CMP AL, 0	So sánh AL với 0
JZ print	Nhảy nếu ZF = 1
MOV AH, 0Eh	Gán AH = 0Eh để ngắt
INT 10h	
JMP nextchar	Nhảy tới nextchar
print:	
POP AX	
PUSH SI	
MOV SI, CS:temp	
RET	
DISPLAY_ ENDP	Kết thúc hàm
;//////////////////////	
INPUT_NUMBER PROC	Hàm INPUT_NUM
MOV BX,0h	BX = 0
MOV CX, 0h	CX = 0
next_digit:	AH = 0
MOV AH, 00h	

INT 16h	
MOV AH, 0Eh	Interupt
INT 10h	
CMP AL, 13	Interupt
JNE next	So sánh AL = 13 (nhấn Enter)
JMP STOP_INPUT_NUMBER	Nhảy nếu không bằng
next:	Nhảy về stop_input_num
MOV BL,AL	
cmp BX,'0'	BL = AL
JB not_minus	So sánh BX >= ‘0’(So sánh ascii)
CMP BX, '9'	Nhảy nếu điều kiện không đúng
JA not_minus	So sánh BX <= ‘9’(So sánh ascii)
SUB BX,30h	Nhảy nếu điều kiện trên không đúng
MOV AX, CX	BX - 30h (Chuyển về kí tự ascii)
MOV CX, 0AH	AX = CX
MUL CX	CX = 10
MOV CX, AX	AX*CX (= AX * 10)
ADD CX, BX	CX = AX
JMP next_digit	CX + BX
not_minus:	Nhảy
MOV AH,4CH	
INT 21h	Interupt terminal
RET	
STOP_INPUT_NUMBER:	Trở về từ lệnh call
RET	
INPUT_NUMBER ENDP	
;//////////////////////	
FIBO PROC	Hàm tính FIBO
CMP CX,0H	So sánh CX = 0
JE OUT_FIBO	Bằng thì nhảy OUT_FIBO
CMP CX,01H	So sánh CX = 1
JE OUT_FIBO	Bằng thì nhảy OUT_FIBO
PUSH CX	Lưu CX
DEC CX	CX -= 1
CALL FIBO	Gọi FIBO (Fibo(CX - 1))
MOV AX,BX	AX = BX (Hàm trên kết quả lưu BX)
DEC CX	CX -= 1
PUSH AX	Lưu AX
CALL FIBO	Gọi FIBO(Fibo(CX - 2))
POP AX	Lấy AX
ADD BX,AX	BX = AX+BX= (Fibo(CX-1) + Fibo(CX-2))
POP CX	
RET	Trở về từ hàm gọi
OUT_FIBO:	
MOV BX,01H	BX = 1

RET	Trở về từ hàm gọi
FIBO ENDP	Kết thúc hàm
;//////////////////////	
DISPLAY_OUT PROC	Hàm DISPLAY_OUT
CMP BX, 0AH	So sánh BX > 10
JNB OUTPUT_NEXT	Nhảy nếu điều kiện trên sai
ADD BX,30h	BX + 30h
MOV DX,BX	DX = BX
MOV AH,02H	Interupt In kí tự
INT 21H	Trở về từ lệnh gọi
RET	
OUTPUT_NEXT:	
MOV CX, 0H	CX = 0
MOV AX, BX	AX = BX
DISPLAY:	
MOV BX, 10D	BX = 10
MOV DX, 00H	DX = 0
DIV BX	AX = AX/BX
ADD DX, 30H	DX + 30 (DX = AX%BX)
PUSH DX	Lưu DX
INC CX	CX + 1
CMP AX, 0H	So sánh AX = 0
JE END_DISPLAY	Nhảy nếu bằng
JMP DISPLAY	Nhảy DISPLAY
	Lấy DX
END_DISPLAY:	
POP DX	Interupt In kí tự
MOV AH, 02H	
INT 21H	
LOOP END_DISPLAY	LOOP
RET	Trở về từ lệnh CALL
DISPLAY_OUT ENDP	Kết thúc hàm
;//////////////////////	
OUTPUT PROC	Giảm CX
	Lưu CX
DEC CX	Gọi hàm FIBO(tính số fibo)
PUSH CX	Gọi DISPLAY_OUT
CALL FIBO	
CALL DISPLAY_OUT	So sánh CX > 1
POP CX	Không đúng thì nhảy OUT_SP (in dấu ,)
CMP CX,01H	Nhảy PRINT_1
JA OUT_SP	
JMP PRINT_1	Hàm in ra màn hình
OUT_SP:	
CALL DISPLAY_	

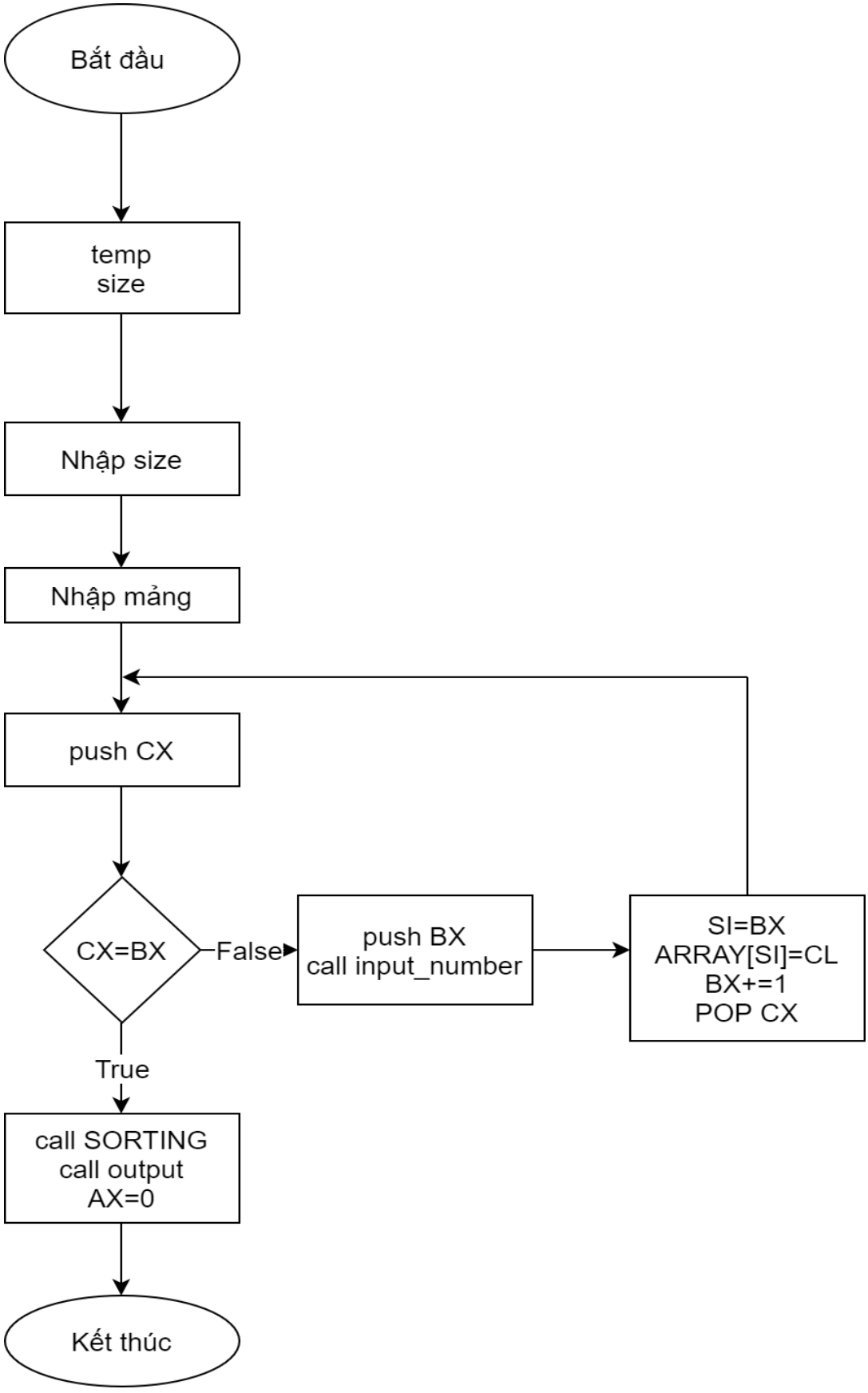
DB ", ",0
JMP OUTPUT
PRINT_1:
CALL DISPLAY_
DB ", 1",0
MOV AH, 4CH
INT 21H
 	
OUTPUT ENDP
;//////////////////////
END MAIN
	Nhảy về OUTPUT(lặp lại với giá trị CX giảm dần)
	Hàm in ra màn hinh
	In kí tự 1 (Chuỗi fibo đã in xong)
	Ngắt Terminal
	
	Kết thúc hàm
	
	Kết thúc chương trình

-Kết quả mô phỏng:

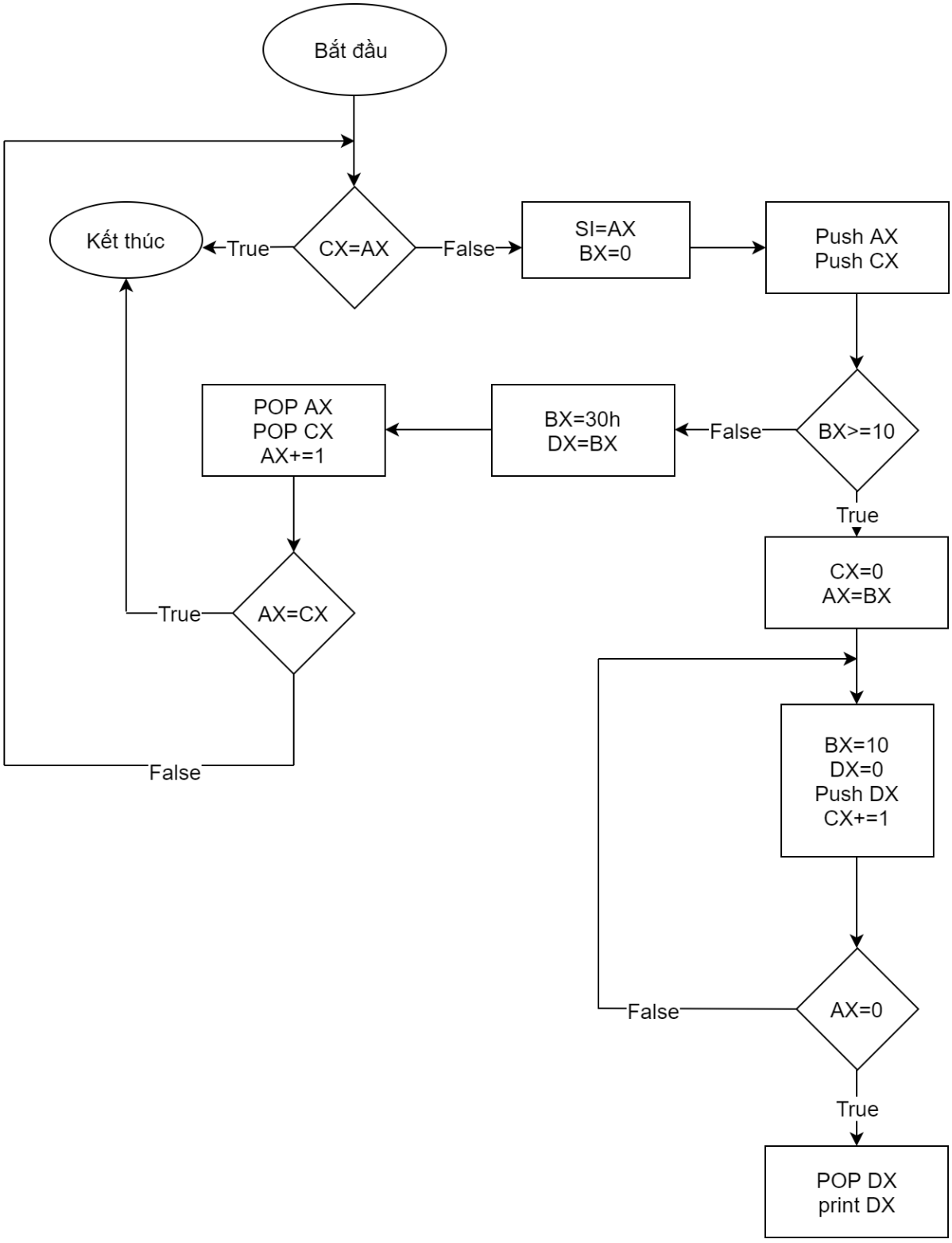
Hình 4 : Kết quả mô phỏng

II. Nhập mảng N(N<15), sắp xếp theo thứ tự tăng dần và in ra màn hình console:

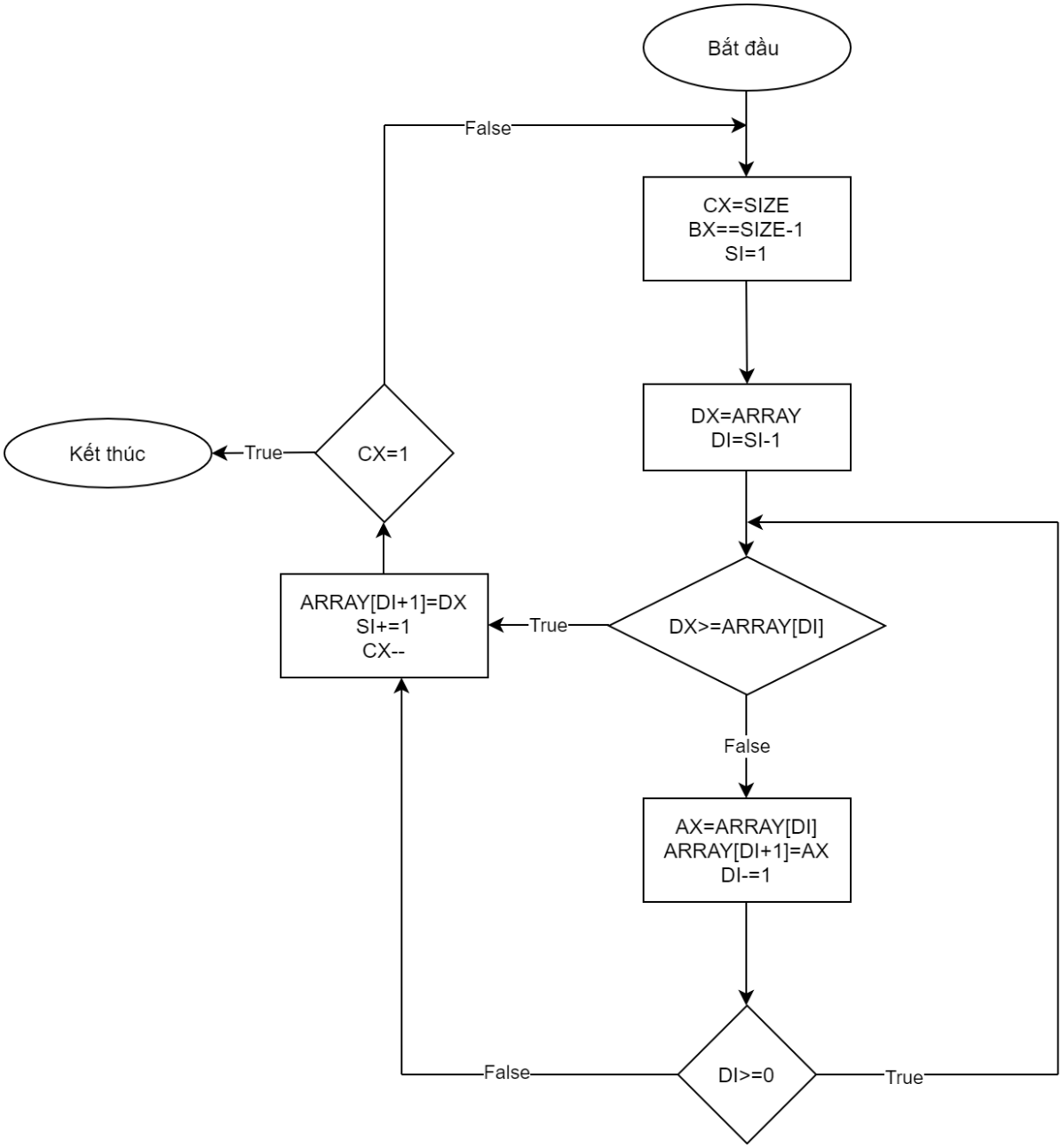
1. Sơ đồ giải thuật :



Hình 5 : Sơ đồ hàm main



Hình 6 : Sơ đồ hàm output



Hình 7 : Sơ đồ hàm Sort

Code	Ý nghĩa
.MODEL SMALL org 100h .STACK 100H .DATA ARRAY DB 15 DUP(?) tb1 db "Nhap so luong phan tu cua mang: $" tb2 db "Nhap phan tu cua mang: $" tb3 db "Mang sau khi duoc sap xep la: $"	 Tạo mảng Các thông báo báo nhập xuất

tb4 db '', $" under_line DB 0AH,0DH,"$"	 Xuống dòng
SIZE DW ? temp DW ?	Kích thước mảng Biến tạm temp
.CODE MAIN PROC	Phần .code Hàm chính

MOV AX,@DATA
MOV DS,AX
MOV AL , ARRAY
	AX = @DATA
	DS = AX
	AL = ARRAY

OUTPUT_TB1:	
MOV AH,9 LEA DX,tb1 INT 21h	Hàm ngắt in ra màn hình In ra màn hình

MOV AH,01H
CALL INPUT_NUMBER
MOV SIZE,BX
	AH = 1
	Hàm nhập từ bàn phím
	SIZE = BX

OUTPUT_TB2:	
MOV BX ,00H	BX = 0
MOV AH,9 LEA DX,under_line INT 21h	Hàm ngắt in ra màn hình Xuống dòng Hàm in ra màn hình
MOV AH,9 LEA DX,tb2 INT 21h	
MOV AH,9 LEA DX,under_line INT 21h	
INPUT_ARRAY:	
MOV AH,9 LEA DX,under_line INT 21h	
PUSH CX CMP BX, CX	Lưu vào CX So sánh BX = CX Bằng thì nhảy

JE OUTPUT_SORT PUSH BX CALL INPUT_NUMBER	Gọi hàm nhập (nhập các phần tử)

STOP_INPUT:
POP BX
MOV SI, BX
MOV ARRAY[SI], CL
INC BX
POP CX
JMP INPUT_ARRAY
	SI = BX
	ARRAY[SI]= CL
	Tăng BX lên 1
	Nhảy đến INPUT_ARRAY để nhập tiếp phần tử tiếp theo

OUTPUT_SORT: CALL SORTING MOV AH,9 LEA DX,tb3 INT 21h MOV AX , 00H	Gọi hàm sort Hàm in ra màn hình AX = 0
CALL OUTPUT	Hàm xuất
MAIN ENDP ;//////////////////////////	Kết thúc hàm chính

INPUT_NUMBER PROC
MOV BX,0h
MOV CX, 0h
next_digit:
MOV AH, 00h
INT 16h
MOV AH, 0Eh
INT 10h
CMP AL, 13
JNE next
JMP STOPINPUT_NUMBER
next:
MOV BL,AL
cmp BX,'0'
JB not_minus
CMP BX, '9'
JA not_minus
	Hàm nhập INPUT
	BX = 0
	CX = 0
	
	AH = 0
	Interupt
	AH = 0EH
	Interupt
	So sánh AL = 13 (Enter)
	Nhảy nếu không bằng
	Nhảy
	BL = AL
	So sánh BX >= ‘0’(so sánh ascii)
	
	Nhảy nếu không đúng
	So sánh BX < =‘9’(so sánh ascii)
	Nhảy nếu không đúng

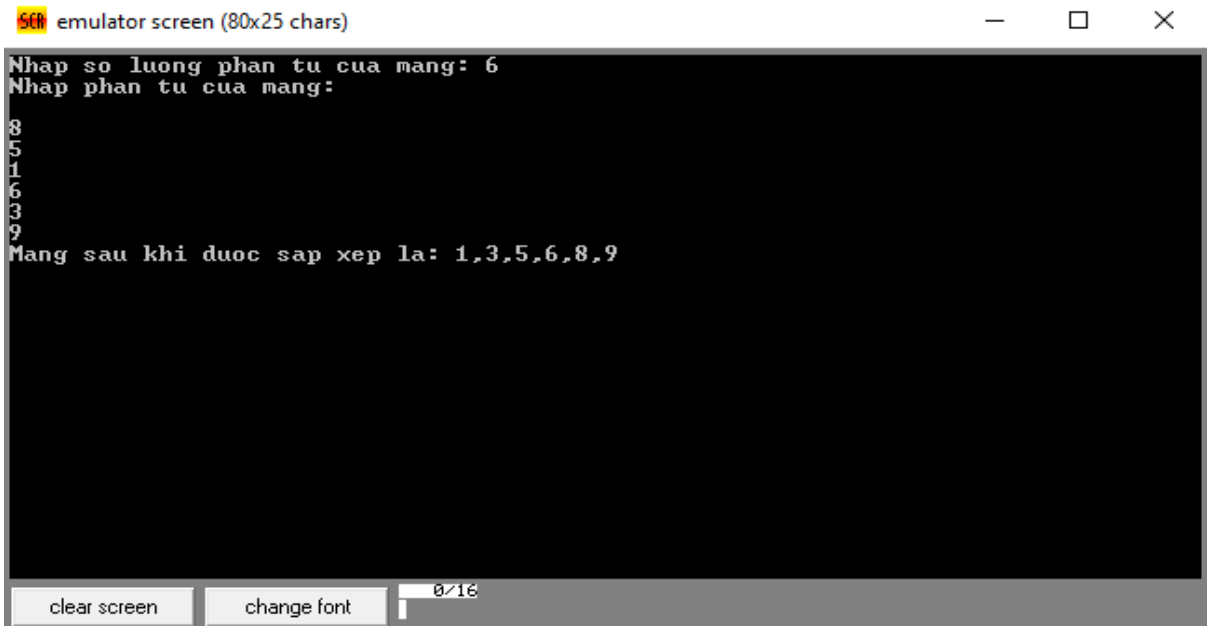
SUB BX,30h	BX – 30H(Chuyển về mã ascii)
MOV AX, CX	AX = CX
MOV CX, 0AH	CX = 10
MUL CX	AX = AX*10
MOV CX, AX	CX = AX
ADD CX, BX	CX = BX (Cộng số còn lại)
JMP next_digit	Nhảy
not_minus:	
MOV AH,4CH	Interupt
INT 21h	
RET	Quay về từ lệnh gọi
STOPINPUT_NUMBER:	
RET	
INPUT_NUMBER ENDP	Kết thúc hàm
;//////////////////////////	
SORTING PROC	Hàm sort
PUSHA	Lưu register
MOV CX,CS:SIZE	CX = SIZE
MOV BX,CX	BX = SIZE
SUB BX,1	BX = SIZE – 1
MOV SI,1	SI = 1
FOR_LOOP:	DL = ARRAY[SI]
MOV DL,ARRAY[SI]	DI = SI
MOV DI,SI	
SUB DI,1	DI = DI – 1
WHILE_LOOP:	
CMP DL,ARRAY[DI]	So sánh DL >= ARRAY[DI]
JGE EXIT_WHILE_LOOP	Đúng thì nhảy
MOV AL,ARRAY[DI]	AL = ARRAY[DI]
MOV ARRAY[DI+1],AL	ARRAY[DI+1] = AL
SUB DI,1	DI = DI – 1
CMP DI,0	So sánh DI > = 0
JGE WHILE_LOOP	Đúng thì nhảy
EXIT_WHILE_LOOP:	
MOV ARRAY[DI+1],DL	ARRAY[DI+1] = DL
ADD SI,1	SI = SI + 1
DEC CX	Giảm CX
CMP CX,1	So sánh CX = 1
JNE FOR_LOOP	Sai thì nhảy
SORT_DONE:	
POPA	
RET 4	
SORTING ENDP	Kết thúc hàm sort
;//////////////////////////	
OUTPUT_C PROC	

MOV CS:temp, SI
POP SI
PUSH AX
nextchar:
MOV AL, CS:[SI]
INC SI
CMP AL, 0
JZ print
MOV AH, 0Eh
INT 10h
JMP nextchar
print:
POP AX
PUSH SI
MOV SI, CS:temp
RET
;////////////////////
	Hàm in ra display
	Gán temp = SI
	Lấy giá trị vào SI
	Đưa giá trị AX vào stack
	
	Gán AL = temp[SI]
	Tăng SI (tăng địa chỉ)
	So sánh AL với 0
	Nhảy nếu ZF = 1
	
	Gán AH = 0Eh để ngắt
	Nhảy tới nextchar
	
	Kết thúc hàm

OUTPUT PROC
 	
CMP AX, CX
JE not_minus
 	
MOV SI , AX
MOV BX, 00H
MOV BL , ARRAY[SI]
PUSH CX
PUSH AX
 	
CMP BX, 0AH
JNB OUTPUT_NEXT
ADD BX ,30h
MOV DX, BX
MOV AH, 02H
INT 21H
JMP OUT_1
OUTPUT_NEXT:
MOV CX, 00H
MOV AX, BX
DISPLAY:
MOV BX, 10D
MOV DX, 00H
DIV BX
ADD DX, 30H
PUSH DX
INC CX
CMP AX, 0
	
	So sánh AX = CX
	Đúng thì nhảy
	
	SI = AX
	BX = 0
	BL = ARRAY[SI]
	
	So sánh BX >= 10
	Không đúng thì nhảy
	BX + 30h (chuyển sang ascii)
	DX = BX
	Interupt để in kí tự
	Nhảy đến OUT_1
	
	CX = 0
	AX = BX
	
	BX = 10
	DX = 0
	AX = AX / 10
	DX + 30h
	
	Tăng CX thêm 1
	So sánh AX = 0

JE END_DISPLAY	Bằng thì nhảy
JMP DISPLAY	Nhảy đến DISPLAY
END_DISPLAY:	
POP DX	
MOV AH, 02H	
INT 21H	
LOOP END_DISPLAY	
OUT_1:	
POP AX	
POP CX	
INC AX	Tăng AX thêm 1
CMP AX, CX	So sánh AX = CX
JE not_minus	Nhảy nếu bằng
CALL OUTPUT_C	In ra màn hình
DB ",",0	
JMP OUTPUT	
RET	
OUTPUT ENDP	Kết thúc hàm
;//////////////////////	
END MAIN	Kết thúc chương trình

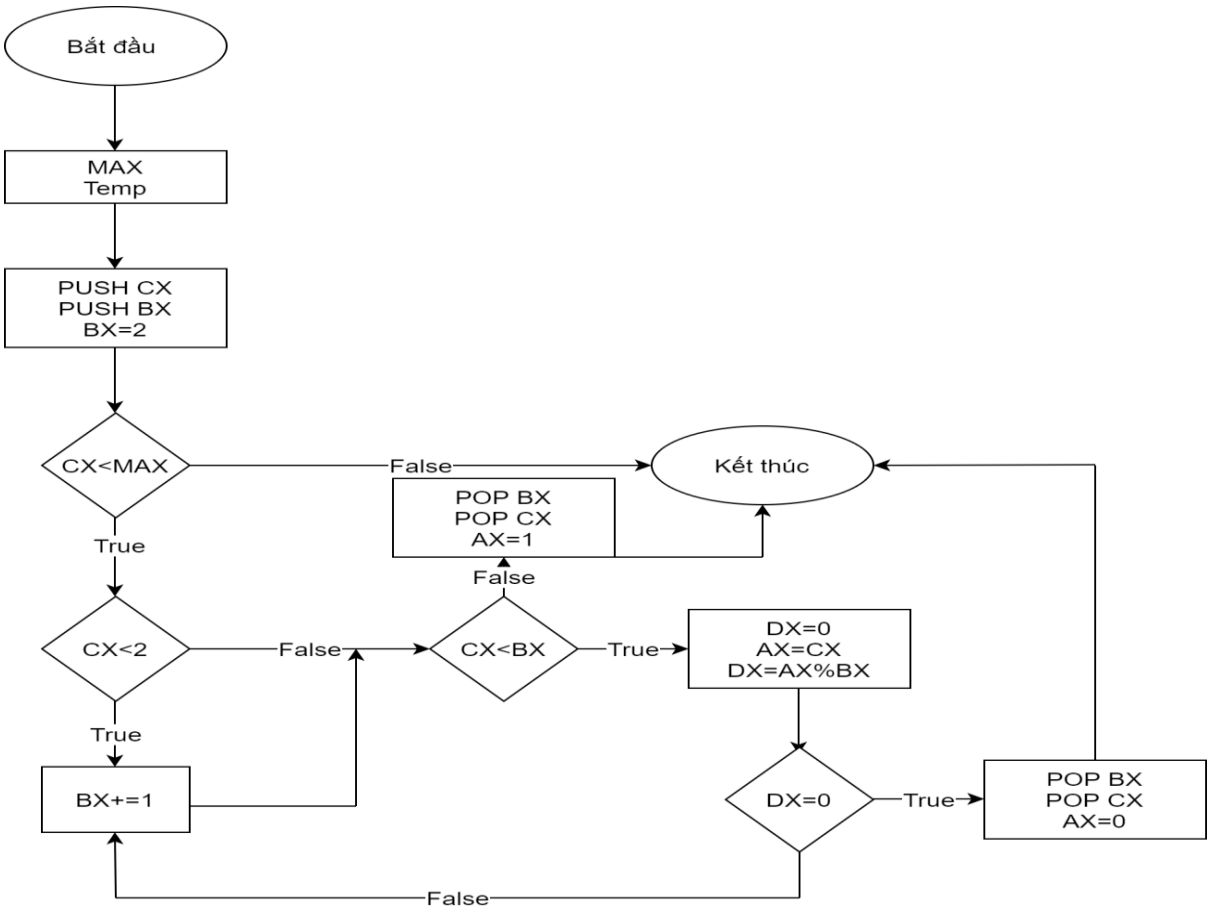
-Kết quả mô phỏng :



Hình 8 : Mô phỏng kết quả

III. Nhập vào số nguyên N (N<150),in ra các số nguyên tố nhỏ hơn N vào file.txt(mỗi số cách nhau một khoảng trắng)

1. Sơ đồ giải thuật:



Hình 9 : Sơ đồ giải thuật bài 3

Code	Ý nghĩa
.MODEL SMALL include 'emu8086.inc' org 100h	
.DATA	Phần .Data
tb1 db "Nhap so N: $"	Khai báo nhập xuất
tb2 db "Cac so nguyen to: $"	
under_line DB 0AH,0DH,"$"	Phần xuống dòng
space db " $"	Phần khoảng trắng
outfile DB "c:\result.txt",0	File để ghi kết quả vào
MAX dw ?	Biến MAX
temp db ?	Biến tạm Temp
.CODE	Phần Code
MAIN PROC	Hàm chính
MOV AX,@DATA MOV DS,AX	
DISPLAY_TB1:	Hàm thông báo nhập số
MOV AH,9 LEA DX,tb1 INT 21h	Dùng hàm ngắt để in ra chuỗi cần thông báo
CALL INPUT_NUMBER	Gọi hàm input_number để nhập số từ bàn phím
MOV AH,9 LEA DX,under_line INT 21h	Dùng hàm ngắt để hiển thị xuống dòng
DISPLAY_TB2: MOV AH,9 LEA DX,tb2 INT 21h	Dùng hàm ngắt để thông báo in các số nguyên tố

PUSH CX
mov ah, 3ch
mov cx, 0
mov dx, offset outfile
int 21h
	Lưu CX
	AH = 3CH
	CX = 0
	DX, offset outfile

jc EXIT	So sánh AX với 0, đúng thì thoát CT
mov MAX, ax	MAX=ax
POP CX	Lấy giá trị CX

MOV BX, 00H	BX=0
LOOP_PRINT:	Vòng lặp in ra các số nguyên tố
PUSH CX	Lưu giá trị của CX
MOV CX , BX	CX=BX
CALL PRIME_FUNCTION	Hàm sử lí nguyên tố
POP CX	Lấy giá trị của CX
CMP AX, 01H	So sánh AX với 01
JE PRINT	Nếu AX=1 thì nhảy tới hàm print
JMP NEXT	Nếu không thì nhảy tới hàm NEXT
MAIN ENDP	Kết thúc hàm chính
;//////////////////////////	
INPUT_NUMBER PROC	Hàm nhập giá trị
MOV CX, 0H	CX=0
MOV BX, 0H	BX=0
READ:	
MOV AH,01H	AH=01
INT 21H	Interrupt
CMP AL,0DH	So sánh AL với 0Dh
JZ EXIT1	Nhảy nếu =0
MOV BL,AL	BL = AL
cmp BX,'0'	So sánh BX >= ‘0’(so sánh ascii)
JB EXIT	Nhảy nếu không đúng
cmp BX,'9'	So sánh BX < =‘9’(so sánh ascii)
JA EXIT	Nhảy nếu không đúng
SUB BX, 30H	BX – 30H(Chuyển về mã ascii)
MOV AX, CX	AX = CX
MOV CX, 0AH	CX = 10
MUL CX	AX = AX*10
MOV CX, AX	CX = AX
ADD CX, BX	CX = BX (Cộng số còn lại)
JMP READ	Nhảy
EXIT1:	
RET	

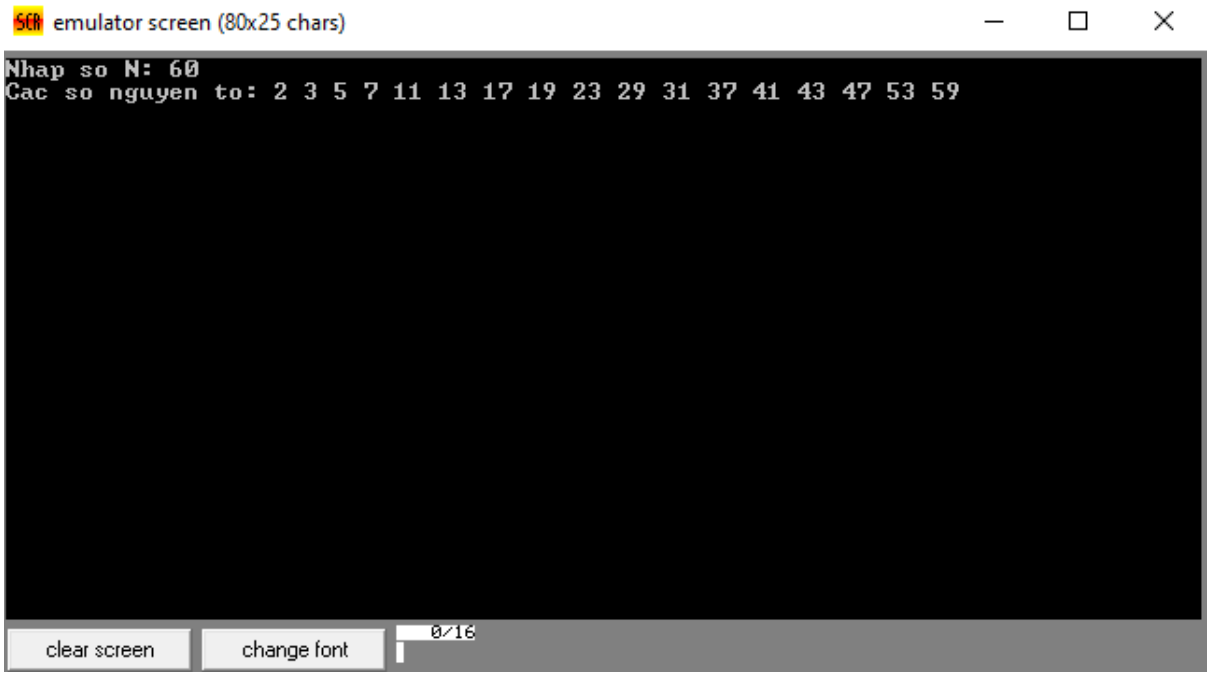
INPUT_NUMBER ENDP	Kết thúc hàm nhập
;//////////////////////////	
PRIME_FUNCTION PROC	Hàm xử lý số nguyên tố
PUSH CX	Lưu CX
PUSH BX	Lưu BX
CMP CX , 02H	So sánh CX với 2
JB NOT_PRIME	Nhảy nếu CX nhỏ hơn 2
MOV BX, 02H	BX=2
LOOP_PRIME_NUMBER:	
CMP BX, CX	So sánh BX với CX
JB CONTINUE_PRIME_FUNCTION	Nhảy nếu BX<CX
JMP PRIME	Nhảy không thì nhảy tới hàm PRIME
CONTINUE_PRIME_FUNCTION:	
MOV DX, 00H	DX = 0
MOV AX, CX	AX = CX
DIV BX	AX = AX/BX
CMP DX, 00H	So sánh DX = 0 (DX = AX%BX)
JE NOT_PRIME	Bằng thì nhảy NOT_PRIME
INC BX	BX++
JMP LOOP_PRIME_NUMBER:	
PRIME:	
POP BX	Lấy lại BX
POP CX	Lấy CX
MOV AX, 01H	AX = 01
RET	Trở về
NOT_PRIME:	
POP BX	Lấy lại BX
POP CX	Lấy CX
MOV AX, 00H	AX = 0
RET	Trở về
PRIME_FUNCTION ENDP	Kết thúc hàm
;//////////////////////////	

DISPLAY_NUMBER PROC	
CMP BX, 0AH	So sánh BX với 10
JB LESS_THAN_10	Nhảy nếu BX<10
MOV CX, 00H	CX=0
MOV AX, BX	AX=BX
CONTINUE_DISPLAY:	
MOV BX, 10D	BX=10
MOV DX, 00H	DX=0
DIV BX	AX = AX/BX
ADD DX, 30H	DX + 30 (DX = AX%BX)
PUSH DX	Lưu DX
INC CX	CX++
CMP AX, 0	So sánh AX với 0
JE DISPLAY_END	Nhảy nếu bằng
JMP CONTINUE_DISPLAY	Nhảy CONTINUE_DISPLAY
LESS_THAN_10:	Hàm nhở hơn 10
ADD BX , 30h	BX+=30h
CALL WRITE_FILE	Gọi hàm ghi vào file
MOV DX, BX	DX=BX
MOV AH, 02H	Interrupt in kí tự
INT 21H	
RET	Trở về hàm chính
DISPLAY_END:	
POP DX	Lấy DX
MOV BX, DX	BX=DX
CALL WRITE_FILE	Gọi hàm ghi file
MOV AH, 02H	Interrupt in kí tự
INT 21H	
LOOP DISPLAY_END	
RET	
DISPLAY_NUMBER ENDP	Kết thúc hàm
;//////////////////////////	

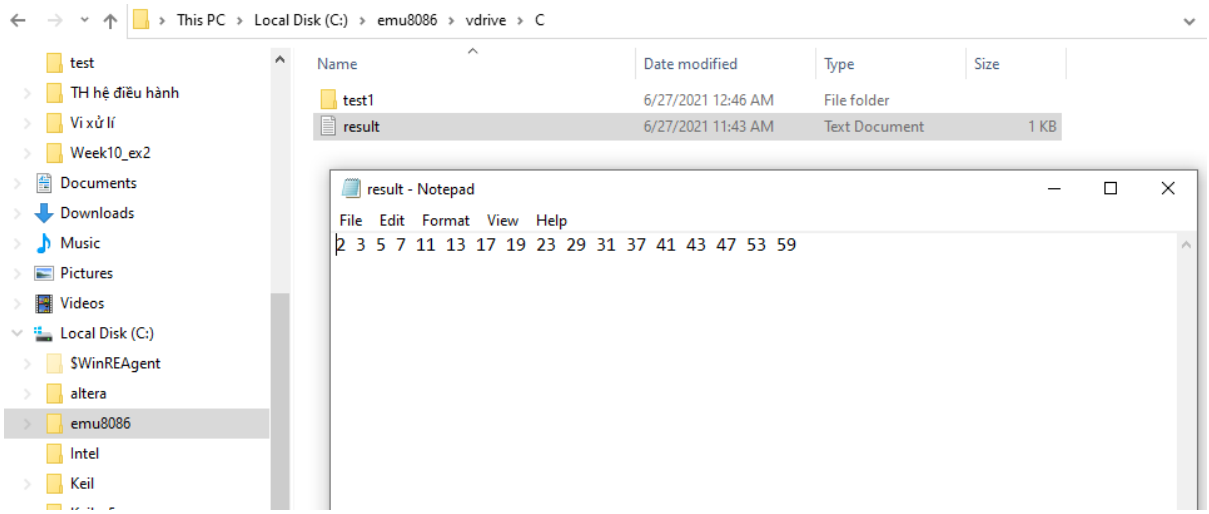
WRITE_FILE PROC	Hàm ghi vào file
PUSH BX	Lưu giá trị của BX
PUSH cx	Lưu giá trị của CX
PUSH DX	Lưu giá trị của DX
mov ah, 40h	Ah=40h
MOV temp, BL	Temp=BL
mov bx, MAX	Bx =MAX
mov dx, offset temp	Thực hiện ghi
mov cx, 01h	Interrupt
int 21h	
POP DX	Lấy giá trị của BX
POP cx	Lấy giá trị của CX
POP BX	Lấy giá trị của DX
RET	Trở về hàm chính
WRITE_FILE ENDP	Kết thúc hàm ghi
;//////////////////////////	
PRINT:	
PUSH CX	Lưu giá trị CX
PUSH BX	Lưu giá trị BX
CALL DISPLAY_NUMBER	Gọi hàm DISPLAY_NUMBER
mov ah, 40h	Ah=40h
MOV temp, 32d	Temp=32
mov bx, MAX	Bx=MAX
mov dx, offset temp	Thực hiện ghi
mov cx, 01h	Interrupt
int 21h	
POP BX	Lấy giá trị của BX
POP CX	Lấy giá trị của CX
MOV AH,9	
LEA DX,space	In ra kí tự khoảng trắng
INT 21h	
NEXT:	
INC BX	
	BX++
LOOP LOOP_PRINT	

EXIT:
mov ah, 4ch
int 21h
RET
;////////////////////////////////
END MAIN
	Hàm thoát chương trình
	
	Kết thúc chương trình

-Kết quả mô phỏng:



Hình 10 : Mô phỏng kết quả bài 3



Hình 11 : Ghi vào file Result.txt