PROJECT 3

PRIME NUMBER

CSCI 150
ASSEMLY LANGUAGE

MAI PHAM

DEVELOPMENT ENVIRONMENT
VM - VISUAL STUDIO 2017

TABLE OF CONTENTS

Mai Pham
CSCI 150 - Project 3
Prof. T. Vo

PROJECT NOTE

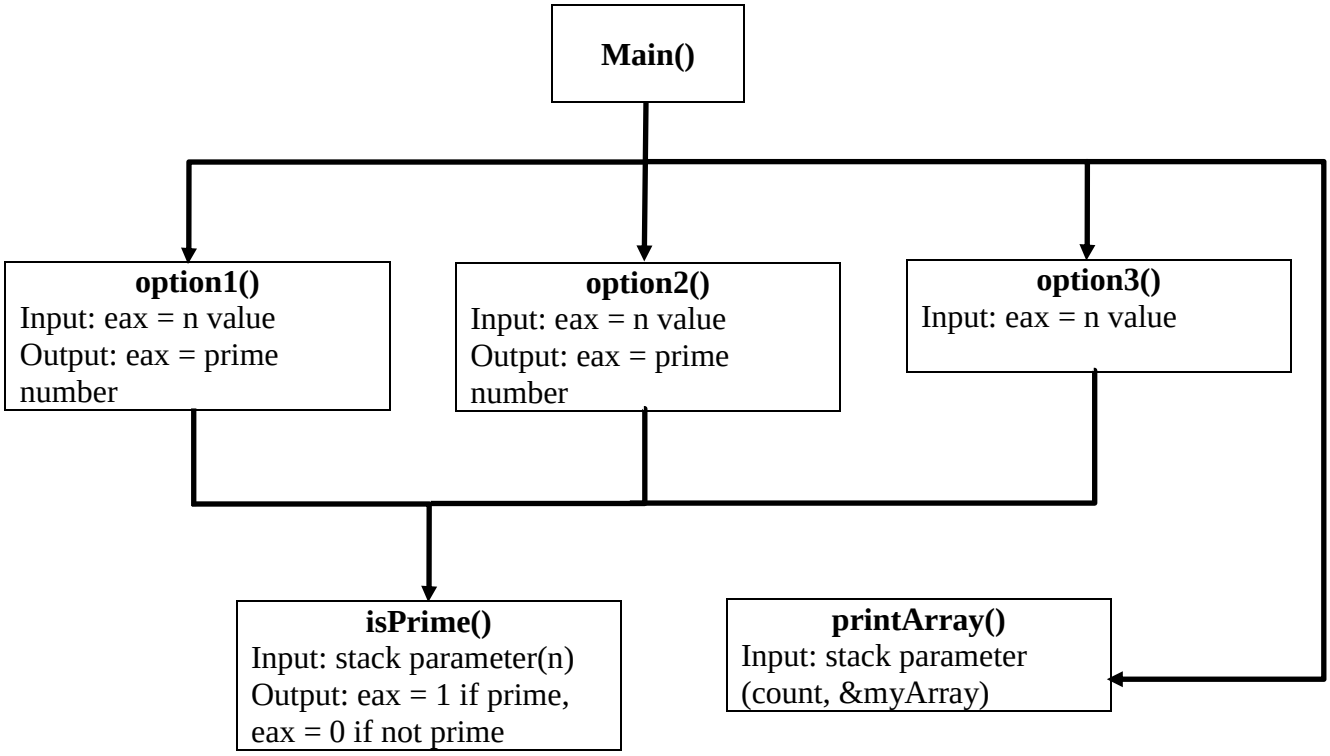
OBJECTIVE:

- Write a program that prompt for an input n value between 2 to 2 million and an option to either find the largest, smallest, or all prime numbers. If invalid value or option is entered, the program would prompt user to renter the information.

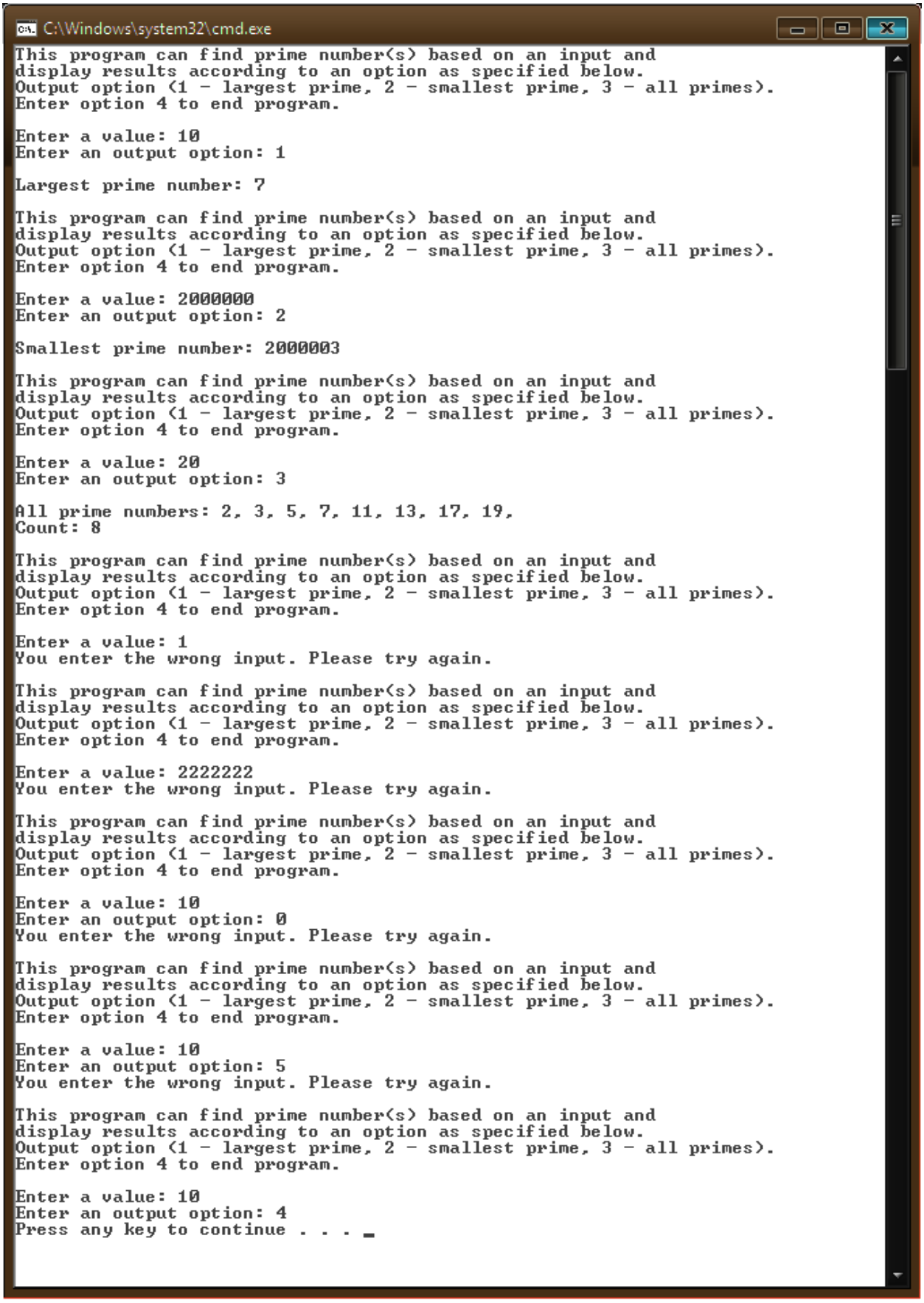
STATUS:

- The project is fairly easy. I did not encounter major issues with it and had successfully run the main project. Did not attempt any extra credit.

HIERARCHY CHART



OUTPUT


C:\Windows\system32\cmd.exe
This program can find prime number(s) based on an input and
display results according to an option as specified below.
Output option (1 - largest prime, 2 - smallest prime, 3 - all primes).
Enter option 4 to end program.
Enter a value: 10
Enter an output option: 1
Largest prime number: 7
This program can find prime number(s) based on an input and
display results according to an option as specified below.
Output option (1 - largest prime, 2 - smallest prime, 3 - all primes).
Enter option 4 to end program.
Enter a value: 2000000
Enter an output option: 2
Smallest prime number: 2000003
This program can find prime number(s) based on an input and
display results according to an option as specified below.
Output option (1 - largest prime, 2 - smallest prime, 3 - all primes).
Enter option 4 to end program.
Enter a value: 20
Enter an output option: 3
All prime numbers: 2, 3, 5, 7, 11, 13, 17, 19,
Count: 8
This program can find prime number(s) based on an input and
display results according to an option as specified below.
Output option (1 - largest prime, 2 - smallest prime, 3 - all primes).
Enter option 4 to end program.
Enter a value: 1
You enter the wrong input. Please try again.
This program can find prime number(s) based on an input and
display results according to an option as specified below.
Output option (1 - largest prime, 2 - smallest prime, 3 - all primes).
Enter option 4 to end program.
Enter a value: 2222222
You enter the wrong input. Please try again.
This program can find prime number(s) based on an input and
display results according to an option as specified below.
Output option (1 - largest prime, 2 - smallest prime, 3 - all primes).
Enter option 4 to end program.
Enter a value: 10
Enter an output option: 0
You enter the wrong input. Please try again.
This program can find prime number(s) based on an input and
display results according to an option as specified below.
Output option (1 - largest prime, 2 - smallest prime, 3 - all primes).
Enter option 4 to end program.
Enter a value: 10
Enter an output option: 5
You enter the wrong input. Please try again.
This program can find prime number(s) based on an input and
display results according to an option as specified below.
Output option (1 - largest prime, 2 - smallest prime, 3 - all primes).
Enter option 4 to end program.
Enter a value: 10
Enter an output option: 4
Press any key to continue . . .

SOURCE CODE

```
TITLE ASM Template
INCLUDE Irvine32.inc

.data
prompt1       BYTE   "This program can find prime number(s) based on an input and", 0Dh,
0AH, 0
prompt2       BYTE   "display results according to an option as specified below.", 0Dh, 0AH,
0
prompt3       BYTE   "Output option (1 - largest prime, 2 - smallest prime, 3 - all
primes).", 0Dh, 0AH, 0
prompt4       BYTE   "Enter option 4 to end program.", 0Dh, 0AH, 0
again         BYTE   "You enter the wrong input. Please try again.", 0Dh, 0AH, 0

value         BYTE   "Enter a value: ", 0
selection     BYTE   "Enter an output option: ", 0
largest              BYTE   "Largest prime number: ", 0
smallest      BYTE   "Smallest prime number: ", 0
allPrime      BYTE   "All prime numbers: ", 0
time          BYTE   "Count: ", 0
count         DWORD  0
myArray       DWORD  2000 DUP(0), 0

.code
main PROC
       L1:
              mov edx, OFFSET prompt1                 ; display all the instructions
              call WriteString
              mov edx, OFFSET prompt2
              call WriteString
              mov edx, OFFSET prompt3
              call WriteString
              mov edx, OFFSET prompt4
              call WriteString
              call crlf

              mov edx, OFFSET value                   ; get input n value
              call WriteString
              call ReadDec
              cmp eax, 2                              ; if < 2, input again
              jb Wrong
              cmp eax, 2000000                        ; if > 2mil, input again
              ja Wrong
              push eax                                ; else save n value

              mov edx, OFFSET selection               ; get option
              call WriteString
              call ReadDec
              cmp eax, 1                              ; if < 1, input again
              jb wrong
              je OP1                                  ; if 1 = option 1
              cmp eax, 2
              je OP2                                  ; if 2 = option 2
              cmp eax, 3
              je OP3                                  ; if 3 = option 3
              cmp eax, 4                              ; if 4 = quit
              je Done
```

```
		ja Wrong					; if > 4, input again

	OP1:
		pop eax					; get n value from stack
		call option1				; get largest prime numer
		call crlf
		mov edx, OFFSET largest			; display the largest prime number
		call WriteString
		call WriteDec
		call crlf
		call crlf
		jmp L1
	OP2:
		pop eax					; get n value from stack
		call option2				; get smallest prime number
		call crlf
		mov edx, OFFSET smallest		; display the smallest prime number
		call WriteString
		call WriteDec
		call crlf
		call crlf
		jmp L1
	OP3:
		push OFFSET myArray			; send in myArray address
		push OFFSET count			; send in count address
		call option3				; get list of prime numbers
		call crlf

		mov edx, OFFSET allPrime		; display list of prime numer
		call WriteString
		push OFFSET myArray			; send in myArray address
		push count				; send in # of count
		call printArray				; print update myArray
		call crlf

		mov edx, OFFSET time			; print # of prime number
		call WriteString
		mov eax, count
		call WriteDec
		call crlf
		call crlf
		jmp L1
	OP4:						; quit program
		jmp Done
	Wrong:
		mov edx, OFFSET again			; wrong input, do again
		call WriteString
		call crlf
		jmp L1
	Done:
		exit
main ENDP

; This procedure would find the largest prime number by decrement
; the input n number (store in eax) until a prime number is reach.
; INPUT: eax = n
; RETURN: eax = largest prime number
option1 PROC
		mov ecx, eax				; ecx = n
```

```
	L1:
		push ecx				; save n
		call isPrime				; eax: 1 = prime, 0 = not prime
		pop ecx					; remove n
		cmp eax, 1				; found prime, stop
		je done
		dec ecx					; else, check next number
		jne L1
	done:
		mov eax, ecx				; eax = prime
		ret
option1 ENDP

; This procedure would find the smallest prime number by increment
; the input number (store in eax) until a prime number is reach.
; INPUT: eax = n
; RETURN: eax = smallest prime number
option2 PROC
		mov ecx, eax				; ecx = n
	L1:
		push ecx				; save n
		call isPrime				; eax: 1 = prime, 0 = not prime
		pop ecx					; remove n
		cmp eax, 1				; found prime, stop
		je done
		inc ecx					; else, check next number
		jne L1
	done:
		mov eax, ecx				; eax = prime
		ret
option2 ENDP

; This procedure would store a list of prime numbers from 2 to
; input number into an array and update number of prime numbers to count
; INPUT: stack input (n, &array, &count)
option3 PROC
		push ebp
		mov ebp, esp
		mov ecx, [ebp + 16]			; ecx = n
		mov esi, [ebp + 12]			; esi = myArray address
		mov ebx, [ebp + 8]			; ebx = count address
	L1:
		cmp ecx, 1				; n value = 1, stop
		je weDone
		push ecx				; save n
		call isPrime				; eax: 1 = prime, 0 = not prime
		pop ecx					; remove n
		cmp eax, 1				; found prime, add to array
		je done
		jne next				; else, check next number
	done:
		mov [esi], ecx				; add to array
		add esi, 4
		inc DWORD PTR [ebx]
	next:
		loop L1
	weDone:
		pop ebp
		ret 12					; return and clear stack
```

```
option3 ENDP

; This procedure would take in 1 number (from stack parameter)
; and check if that number is a prime number or not.
; INPUT: stack parameter (n)
; RETURN: eax = 1 if prime and eax = 0 if not prime.
isPrime PROC
              push ebp
              mov ebp, esp
              mov ecx, [ebp + 8]                    ; ecx = input number
              dec ecx
       L1:
              mov eax, [ebp + 8]                    ; eax = input number
              mov edx, 0                            ; clear edx
              div ecx                               ; edx::eax = eax/ecx
              cmp ecx, 1                            ; ecx = 1 = stop loop
              je done                               ; jump out of loop
              cmp edx, 0                            ; edx = remainder
              je notPrime                           ; edx = 0 = not prime
              loop L1

       notPrime:                                    ; not prime
              mov eax, 0                            ; eax = 0
              jmp over
       done:                                        ; prime
              mov eax, 1                            ; eax = 1
       over:
              pop ebp                               ; restore ebp
              ret                                ; return but keep last stack for reuse
isPrime ENDP

; This procedure would print the array backward
; INPUT: stack parameter (count, &myArray)
printArray    PROC
              push ebp
              mov ebp, esp
              mov ecx, [ebp + 8]                    ; get count
              mov esi, [ebp + 12]                   ; get myArray address
              mov eax, ecx                          ; compute to go to last number
              mov ebx, 4
              mul ebx
              add esi, eax                          ; go to last number of the address
       L3:
              sub esi, 4
              mov eax, [esi]                        ; display number
              call WriteDec
              mov eax, ','                          ; display ','
              call WriteChar
              mov eax, ' '                          ; display ' '
              call WriteChar
              loop L3
              pop ebp
              ret 8                                 ; return and clear stack
printArray    ENDP

END main
```