C FILE

```
#include <cstdio>
#include <cstdlib>
using namespace std;
extern "C" int asmMain();
extern "C" int AllPrimeNumbers(int);
int main() {
	int num = asmMain();
	system("pause");
	return 0;    }
int AllPrimeNumbers(int num){
	//normal function    }
```

ASM FILE

```
.686p
.model flat, C
.stack 4096
includelib legacy_stdio_definitions.lib
ExitProcess PROTO, dwExitCode: DWORD
printf PROTO C,
	format: PTR BYTE, args: VARARG
scanf PROTO C,
	format: PTR BYTE, args: VARARG
AllPrimeNumbers PROTO C,
	dwInt: DWORD
TAB = 9
.data
format1 BYTE "Input an integer: ", 0
format2 BYTE "%d", 0
format3 BYTE "N =    %d",0dh, 0ah, 0
format4 BYTE "%d",0dh, 0ah, 0
newLine BYTE 0dh, 0ah, 0
space BYTE " ", 0
Number DWORD ?

.code

asmMain PROC C
	push ebp
	mov     ebp, esp
	INVOKE printf, ADDR format1
	INVOKE scanf, ADDR format2, ADDR Number
	INVOKE printf, ADDR format3, Number
	INVOKE AllPrimeNumbers,    Number
	INVOKE printf, ADDR Format4, eax
	mov     eax, 0              ; return 0
	pop     ebp
	ret
asmMain ENDP
END
```

AS-2

```
INCLUDE Irvine32.inc
.386
.model flat,stdcall
.stack 4096
ExitProcess proto,dwExitCode:dword
.data
minnum sdword ?
maxnum sdword ?
maxstr byte "Max : ",0
minstr byte "Min : ",0
nums sdword 200 dup(0)
datacount sdword ?
.code
main proc
	;read datacount
	call readint
	mov datacount,eax
	mov ecx,eax
	mov esi,OFFSET nums
	mov ebx,0
	mov bl,TYPE nums
	;input arr
makearr:
	call readint
	mov [esi],eax
	add esi,ebx
	loop makearr
	;set function regedit
	mov ecx,datacount
	mov esi,OFFSET nums
	;find function
	call findmax
	call findmin
	;print result
	mov edx,OFFSET minstr
	mov eax,minnum
```

```
        call writestring
        call writeint
        call crlf
        mov edx,OFFSET maxstr
        mov eax,maxnum
        call writestring
        call writeint
        call crlf
        invoke ExitProcess,0
main endp
findmax proc
        pushad
L1:
        mov eax,[esi]
        jmp returnL2
L2:
        add esi,ebx
        cmp [esi],eax
        jg L1
returnL2:
        loop L2
        mov maxnum,eax
        popad
        ret
findmax endp
findmin proc
        pushad
L1:
        mov eax,[esi]
        jmp returnL2
L2:
        add esi,ebx
        cmp [esi],eax
        jl L1
returnL2:
        loop L2
        mov minnum,eax
        popad
        ret
findmin endp
end main
```

Instr	Description
JO	Jump if overflow
JNO	Jump if not overflow
JS	Jump if sign
JNS	Jump if not sign
JE/	Jump if equal
JZ	Jump if zero
JNE/	Jump if not equal
JNZ	Jump if not zero
JP/	Jump if parity
JPE	Jump if parity even
JNP/	Jump if no parity
JPO	Jump if parity odd
JCXZ/	Jump if CX is zero
JECXZ	Jump if ECX is zero

Unsigned

JB/	Jump if below
JNAE/	Jump if not above or equal
JC	Jump if carry
JNB/	Jump if not below
JAE/	Jump if above or equal
JNC	Jump if not carry
JBE/	Jump if below or equal
JNA	Jump if not above
JA/	Jump if above
JNBE	Jump if not below or equal

signed

JL/	Jump if less
JNGE	Jump if not greater or equ
JGE/	Jump if greater or equal
JNL	Jump if not less
JLE/	Jump if less or equal
JNG	Jump if not greater
JG/	Jump if greater
JNLE	Jump if not less or equal