Mixed Assembly and C Primer ver 1.1

I. C program calling nasm x86 assembly function:

C program (testc.c):

```
#include <stdio.h>

extern int sum();

int main()
{
int a1, a2, x;
a1 =2;
a2=3;
x = sum(a1, a2);
printf("%d + %d = %d", a1, a2, x) ;
   getchar();
   return 0;
}
```

NASM x86 program (funcsum.asm):

```
  segment .text
        global _sum

_sum:   push ebp            ; create stack frame
        mov ebp, esp        ; save stack
        mov eax, [ebp+8]    ; first parameter
        mov edx, [ebp+12]   ; second parameter
        add eax, edx        ; add.  EAX is the default parameter to pass back
        leave               ; restore base pointer → mov esp,ebp, pop ebp
        ret
```

* Compile assembly
Nasm –fwin32 –o funcsum.o funcsum.asm
* Compile C
gcc –c testc.c
* combine assembly and C
gcc testc.o funcsum.o –o testc.exe

---optional: to view it in gdb---
nasm –felf32 -Fstabs –g funcsum.o funcsum.asm
gcc –c testc.c
gcc testc.o funcsum.o –o testc.out

II. Nasm x86 assembly function calling C library:

NASM x86 assembly program:

```
; scanf sample ask 2 numbers and add

        global _main
        extern _printf, _scanf, _getchar

    SECTION .text               ; Code section.

_main:
        push prompt1
        call _printf
        add esp, 4                      ; remove parameter

        push input1
        push scanformat
        call _scanf
        add esp, 8                      ; address is 4 bytes

        push crlf
        call _printf
        add esp, 4

        push prompt2
        call _printf
        add esp, 4                      ; remove parameter

        push input2
        push scanformat
        call _scanf
        add esp, 8                      ; address is 4 bytes

        mov eax, [input1]
        add eax, [input2]
        mov [ans], eax

        push dword [ans]
        push dword [input2]
        push dword [input1]
        push dword ansmsg
        call _printf
        add esp, 16

        ret


    SECTION .data               ; Data section, initialized variables

prompt1 db "Enter first number = ",0    ; string prompt
prompt2 db "Enter second number = ",0
ansmsg          db "%d + %d = %d", 13, 10, 0
crlf db 13,10,0
input1  dd 0
```

```
input2  dd 0
ans     dd 0
scanformat db "%d",0
```

* To assemble:

```
nasm –fwin32 callscaf.asm
gcc callscaf.obj callscaf.exe
```