Multiple Source Files

When working on large project, splitting source file into several source files makes the project easier to manage.

The simplest source file splitting is to keep functions in separate files

Extern Statement

A function not located in the current source file can be called when **extern** statement is specified using the following format:

extern <symbolName>

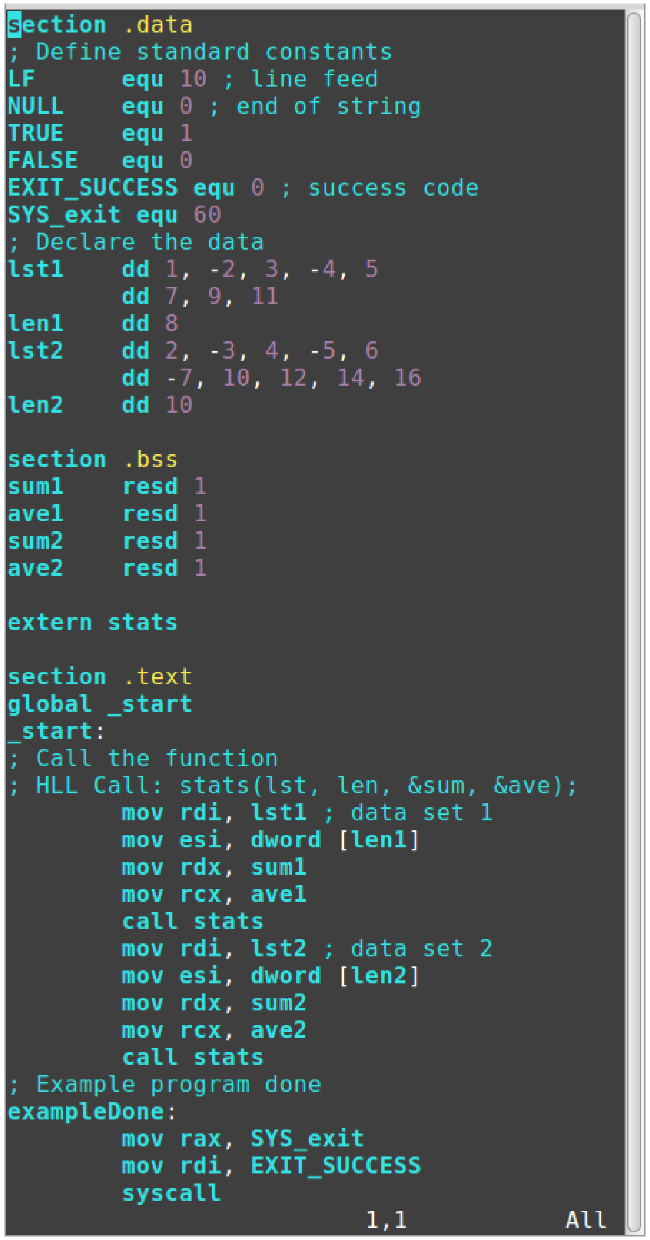
Compile & link command for multiple source files

```
yasm -f elf64 -g dwarf2 -o source1.o source1.s
yasm -f elf64 -g dwarf2 -o source2.o source2.s
yasm -f elf64 -g dwarf2 -o source3.o source3.s

ld -o executable_file source1.o source2.o source3.o
```

Multiple Source Files

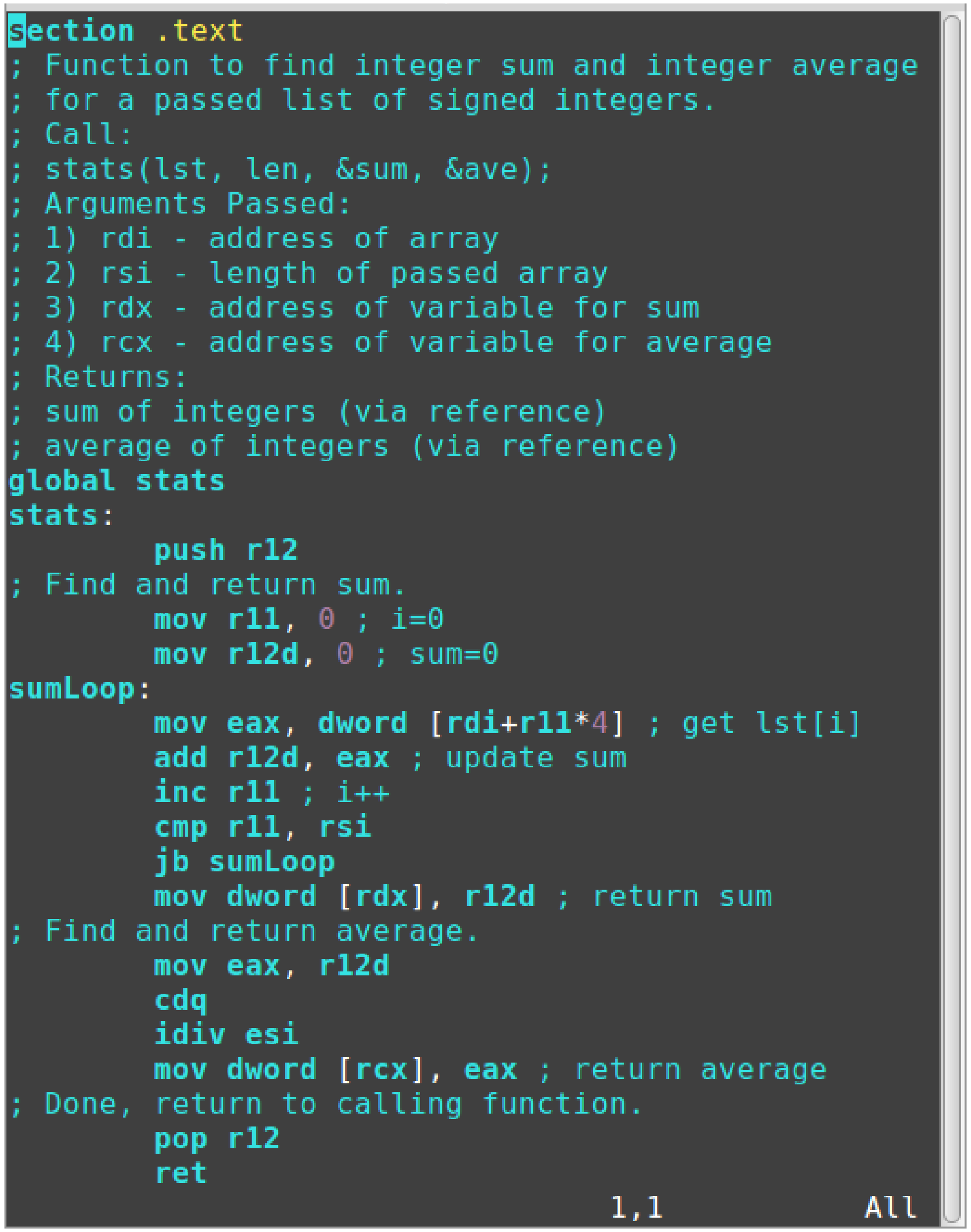
Example: extern statement

main.s



```
section .data
; Define standard constants
LF      equ 10 ; line feed
NULL    equ 0 ; end of string
TRUE    equ 1
FALSE   equ 0
EXIT_SUCCESS equ 0 ; success code
SYS_exit equ 60
; Declare the data
lst1    dd 1, -2, 3, -4, 5
        dd 7, 9, 11
len1    dd 8
lst2    dd 2, -3, 4, -5, 6
        dd -7, 10, 12, 14, 16
len2    dd 10

section .bss
sum1    resd 1
ave1    resd 1
sum2    resd 1
ave2    resd 1

extern stats

section .text
global _start
_start:
; Call the function
; HLL Call: stats(lst, len, &sum, &ave);
        mov rdi, lst1 ; data set 1
        mov esi, dword [len1]
        mov rdx, sum1
        mov rcx, ave1
        call stats
        mov rdi, lst2 ; data set 2
        mov esi, dword [len2]
        mov rdx, sum2
        mov rcx, ave2
        call stats
; Example program done
exampleDone:
        mov rax, SYS_exit
        mov rdi, EXIT_SUCCESS
        syscall
                         1,1           All
```

stats.s



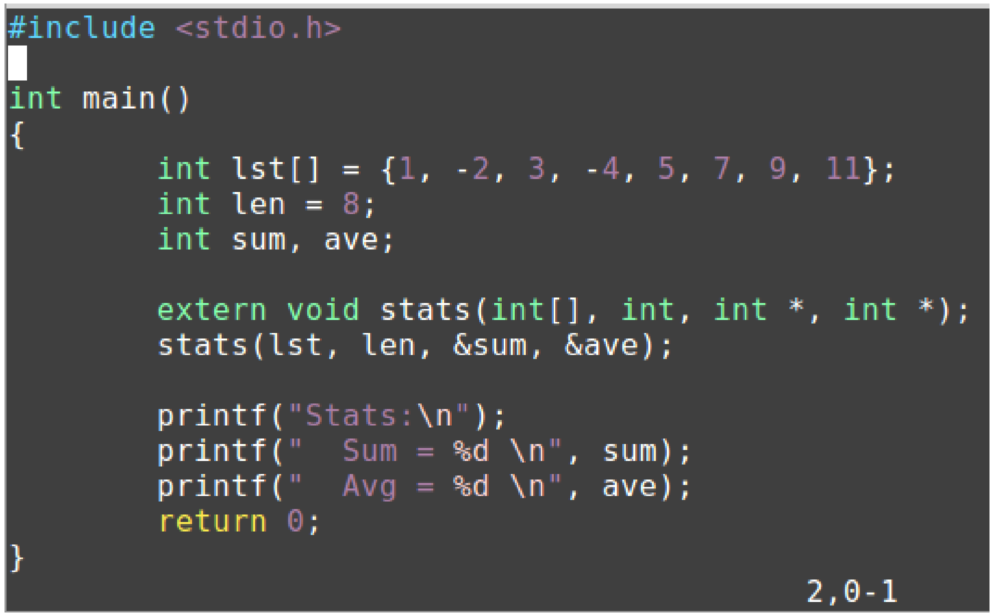
```
section .text
; Function to find integer sum and integer average
; for a passed list of signed integers.
; Call:
; stats(lst, len, &sum, &ave);
; Arguments Passed:
; 1) rdi - address of array
; 2) rsi - length of passed array
; 3) rdx - address of variable for sum
; 4) rcx - address of variable for average
; Returns:
; sum of integers (via reference)
; average of integers (via reference)
global stats
stats:
        push r12
; Find and return sum.
        mov r11, 0 ; i=0
        mov r12d, 0 ; sum=0
sumLoop:
        mov eax, dword [rdi+r11*4] ; get lst[i]
        add r12d, eax ; update sum
        inc r11 ; i++
        cmp r11, rsi
        jb sumLoop
        mov dword [rdx], r12d ; return sum
; Find and return average.
        mov eax, r12d
        cdq
        idiv esi
        mov dword [rcx], eax ; return average
; Done, return to calling function.
        pop r12
        ret
                                1,1           All
```

Multiple Source Files

Interfacing with high-level language (C)

Calling assembly from C

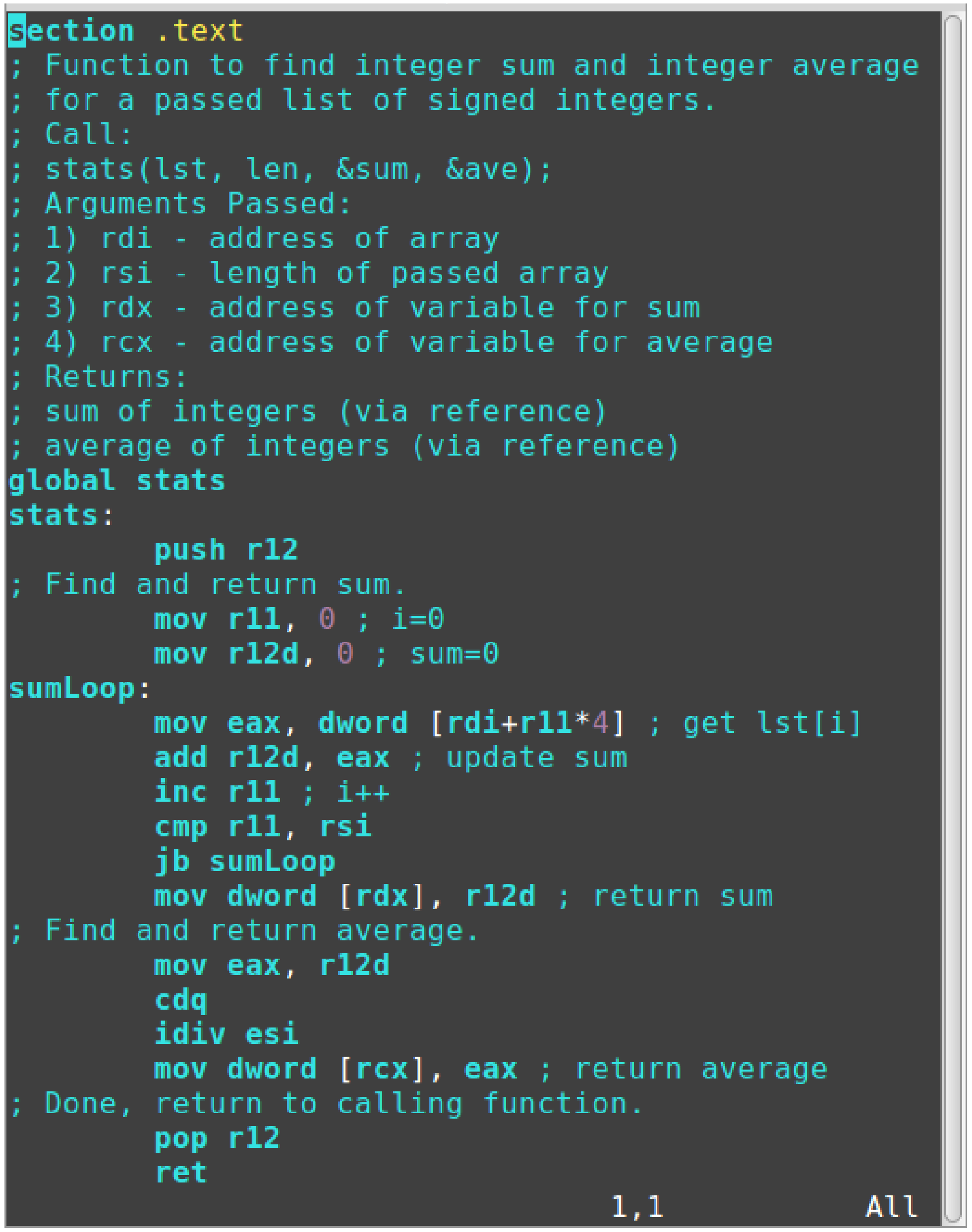
main.c



```c
#include <stdio.h>

int main()
{
        int lst[] = {1, -2, 3, -4, 5, 7, 9, 11};
        int len = 8;
        int sum, ave;

        extern void stats(int[], int, int *, int *);
        stats(lst, len, &sum, &ave);

        printf("Stats:\n");
        printf("  Sum = %d \n", sum);
        printf("  Avg = %d \n", ave);
        return 0;
}
                                            2,0-1
```

```
yasm -f elf64 -g dwarf2 -o stats.o stats.s
gcc main.c stats.o -o cstats
```

stats.s



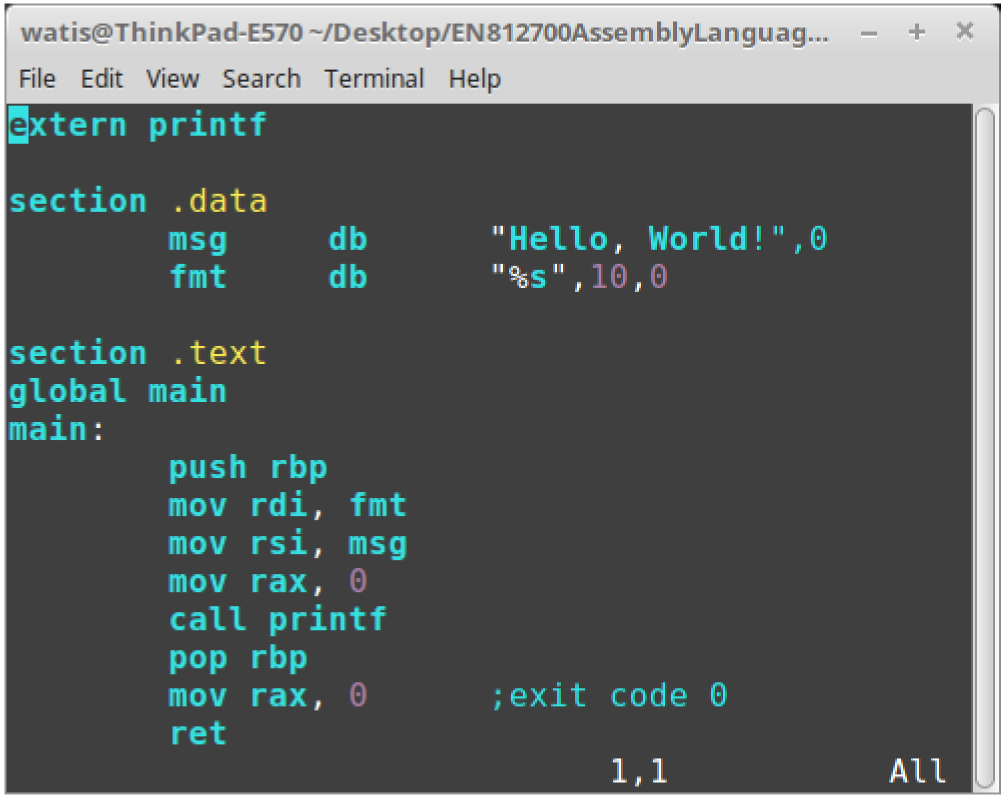
```
section .text
; Function to find integer sum and integer average
; for a passed list of signed integers.
; Call:
; stats(lst, len, &sum, &ave);
; Arguments Passed:
; 1) rdi - address of array
; 2) rsi - length of passed array
; 3) rdx - address of variable for sum
; 4) rcx - address of variable for average
; Returns:
; sum of integers (via reference)
; average of integers (via reference)
global stats
stats:
        push r12
; Find and return sum.
        mov r11, 0 ; i=0
        mov r12d, 0 ; sum=0
sumLoop:
        mov eax, dword [rdi+r11*4] ; get lst[i]
        add r12d, eax ; update sum
        inc r11 ; i++
        cmp r11, rsi
        jb sumLoop
        mov dword [rdx], r12d ; return sum
; Find and return average.
        mov eax, r12d
        cdq
        idiv esi
        mov dword [rcx], eax ; return average
; Done, return to calling function.
        pop r12
        ret
                                 1,1           All
```

Multiple Source Files

Interfacing with high-level language (C)

Calling C (glibc) from assembly

stats.s


watis@ThinkPad-E570 ~/Desktop/EN812700AssemblyLanguag...
File Edit View Search Terminal Help

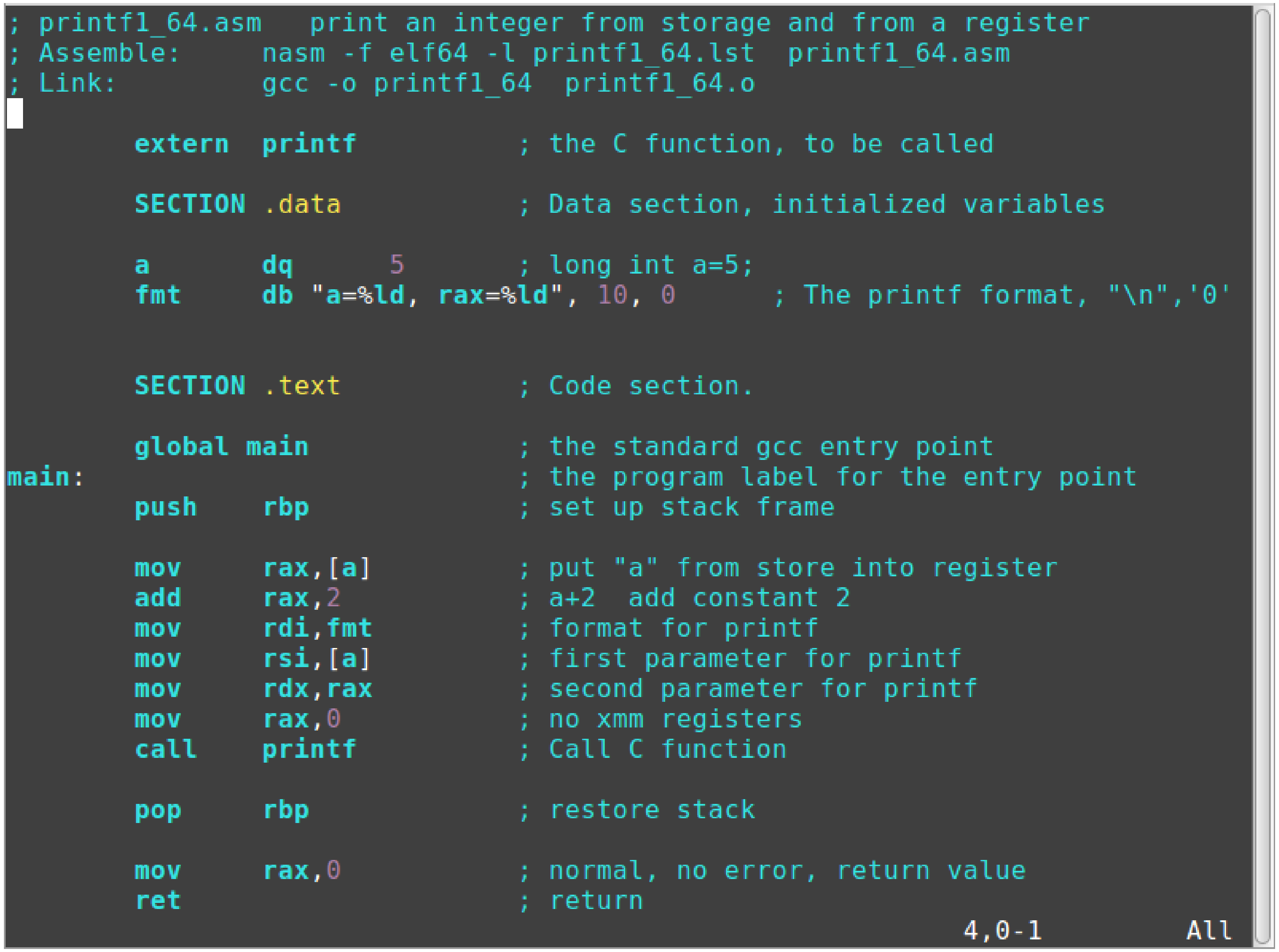
```
extern printf

section .data
        msg     db      "Hello, World!",0
        fmt     db      "%s",10,0

section .text
global main
main:
        push rbp
        mov rdi, fmt
        mov rsi, msg
        mov rax, 0
        call printf
        pop rbp
        mov rax, 0      ;exit code 0
        ret
                              1,1           All
```

```
yasm -f elf64 -g dwarf2 -o printf.o printf.s
gcc -o cprintf printf.o
```

Multiple Source Files

Interfacing with high-level language (C)

Calling C (glibc) from assembly



```
; printf1_64.asm   print an integer from storage and from a register
; Assemble:     nasm -f elf64 -l printf1_64.lst  printf1_64.asm
; Link:         gcc -o printf1_64  printf1_64.o

        extern  printf          ; the C function, to be called

        SECTION .data           ; Data section, initialized variables

        a       dq      5       ; long int a=5;
        fmt     db "a=%ld, rax=%ld", 10, 0      ; The printf format, "\n",'0'


        SECTION .text           ; Code section.

        global main             ; the standard gcc entry point
main:                           ; the program label for the entry point
        push    rbp             ; set up stack frame

        mov     rax,[a]         ; put "a" from store into register
        add     rax,2           ; a+2  add constant 2
        mov     rdi,fmt         ; format for printf
        mov     rsi,[a]         ; first parameter for printf
        mov     rdx,rax         ; second parameter for printf
        mov     rax,0           ; no xmm registers
        call    printf          ; Call C function

        pop     rbp             ; restore stack

        mov     rax,0           ; normal, no error, return value
        ret                     ; return
                                                                  4,0-1         All
```

```
yasm -f elf64 -g dwarf2 -o printf64.o printf64.s
gcc -o printf64 printf64.o
```