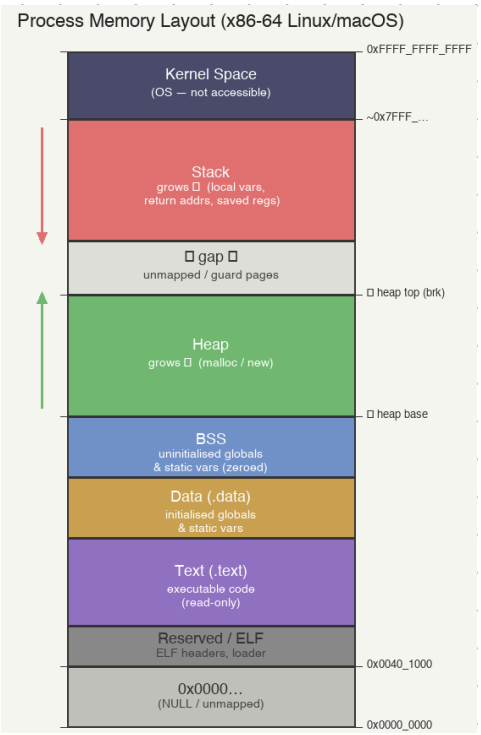
March 27, 2026

x86_64 Assembly

```
global _main
extern _printf
default rel

section .text
_main:
  push rbx          ; Align stack to 16-bytes (callee-saved)

  lea rdi, [msg]      ; Arg 1 (rdi): Format string
  mov rsi, [my_int]     ; Arg 2 (rsi): Integer
  movsd xmm0, [my_float]  ; Arg 1 Float (xmm0): Decimal

  mov rax, 1          ; Variadic Rule: Exactly 1 xmm register used

  call _printf

  pop rbx           ; Restore stack and rbx
  mov rax, 0          ; Return 0
  ret

section .data
  my_int dq 42
  my_float dq 3.14159
  msg db "Int: %d, Float: %f", 10, 0
```



* System V Calling Convention

- What is the relationship between function A which calls function B

  function A → caller

  function B → callee

  (i) Where are the arguments to callee stored?

  (ii) Where does the return value go?

  (iii) Which registers function B is allowed to write?

  (iv) How is stacked managed by callee?

Q. Why ↑ is important?

A. Our program will not be able to communicate with OS or C-standard library.

* 16 General Purpose Registers (64-bit)

- rax — accumulator, storing return values
- rbx — base, callee-saved registers
- rcx — counters for loops; stores 4th argument
- rdx — data; stores 3th argument
- rsi — source index; stores 2nd argument
- rdi — destination index; stores 1st argument

- %rsp — stack pointer, points to the top of the stack
- %rbp — based pointer, callee-saved
- %r8 & %r9 — store 5th and 6th argument
- %r10 & %r11 — caller-saved general-purpose registers
- %r12 & %r15 — callee-saved " "

* Accessing register chunks

Quad word	double word	word	byte
64 bit	32 bit	16 bit	8 bit
%rax	%eax	%ax	%al
%rcx	%ecx	%cx	%cl
%rsi	%esi	%si	%sil
%r8	%r8d	%r8w	%r8b

* Floating Point Registers

- xmm0 to xmm15 (vector registers) → 128-bit

* Special Registers

- %rip — points to the memory addres of next instruction
  - jmp, call
    - jmp → conditionals, switch statements
    - call → when a fn. is called
- %rflags — collection of 1-bit flags: zero, carry, etc.

* Caller-saved vs. Callee-saved

Function A calls function B

A: caller

B: callee

- Caller saved → volatile register.

  Function A better store the value somewhere if it needs that value later since function B may overwrite.

owned by callee

e.g. rax, rcx, rdx, rsi, rdi, r8 to r11

- Callee saved: Callee is responsible for returning these registers to their original state.

owned by caller

e.g. rbx, rbp, r12 to r15

↑ if callee needs to write to these registers, it must first pushes the values to stack and then pop them back before returning.

* Passing Arguments (System V ABI)

(i) Integers and memory addresses are passed in specific caller-saved registers.

e.g. rdi, rsi, rdx, rcx, r8, r9

(ii) Floating point arguments are passed using xmm0 to xmm7.

variadic rule: when a function is passed variable number of arguments, rax must contain the number of values.

So if no xmm is used al is 0.

printf ("%.d, %.d, %.d, %.d", x, y, z, w)

5 arguments.