x86-64 Assembly Functions and the Stack

CS 2130: Computer Systems and Organization 1
March 16, 2026

Announcements

- Homework 5 **due tonight at 11:59pm** on Gradescope
- Homework 6 coming **tomorrow** during lab **due next Monday**

Functions

`f(x,y):` $\longrightarrow$ f:

```
    ...
    ...
```

$\downarrow$ $\equiv$

`    return 4` $\longrightarrow$ retq

```
...
```

`z = f(2,5)` $\longrightarrow$ call f

Function Calls: Calling Conventions

`callq myfun`

- Push return address, then jump to myfun
- Convention: Store arguments in registers and stack before call
  - First 6 arguments (in order): `rdi`, `rsi`, `rdx`, `rcx`, `r8`, `r9`
  - If more arguments, pushed onto stack (last to first)

`retq`

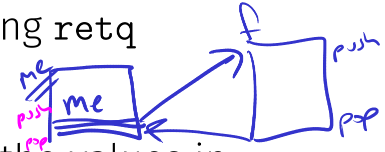
- Pop return address from stack and jump back
- Convention: store return value in `rax` before calling `retq`

This is similar to our Toy ISA's function calls in homework 4

Calling Conventions: Registers

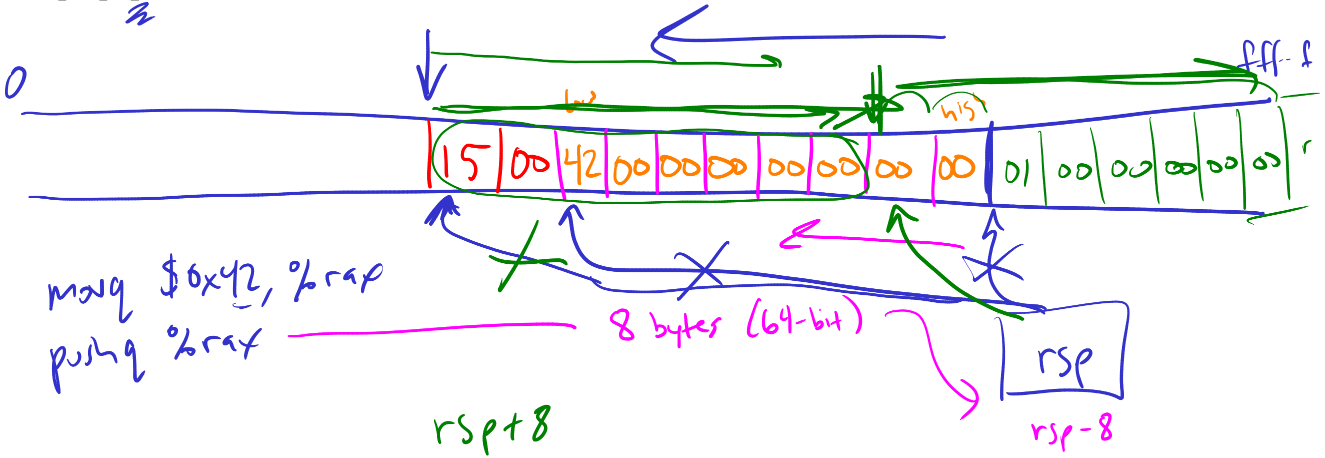
Calling conventions - recommendations for making function calls

- Where to put arguments/parameters for the function call?
- Where to put return value? in `rax` before calling `retq`
- What happens to values in the registers?
  - **Callee-save** - The function should ensure the values in these registers are unchanged when the function returns
    - `rbx`, `rsp`, `rbp`, `r12`, `r13`, `r14`, `r15`
  - **Caller-save** - Before making a function call, save the value, since the function may change it



The Stack

```
pushq %rax
popq %rdx
```



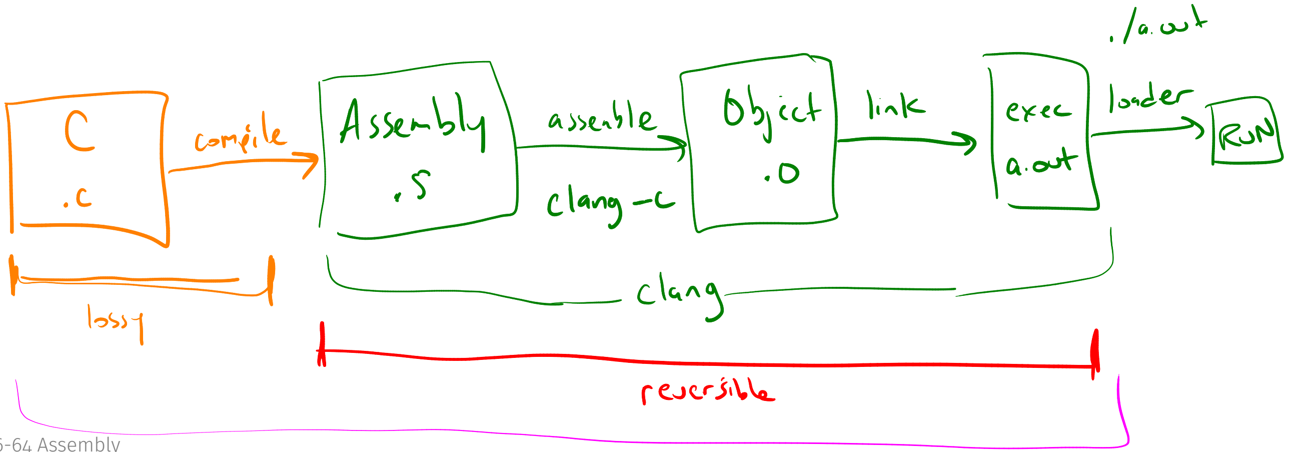
The Stack

`stack.s` – Example with lldb

Compilation Pipeline

Turning our code into something that runs

- **Pipeline** - a sequence of steps in which each builds off the last



./a.out
C
.c
compile
Assembly
.s
assemble
clang -c
Object
.o
link
exec
a.out
loader
RUN
lossy
clang
reversible