# Assembly

Friday, October 16, 2020 12:17 PM

## GCC and assembly

→ Data is represented in binary; so is the code itself!

→ GCC converts C code to machine language

→ Assembly/asm – human-readable machine language

→ Very low-level – may need multiple asm instructions for single C instruction

## Looking at an executable

objump objump -d

Yields human-readable assembly for ELF executable

(mem addr of instruction, bytes, asm instructions)

Notes: $ – constant

%r – register (storage on CPU)

## Registers

Defn. Fast r/w mem slot that can hold variable values on CPU

→ 64 bits, 16 registers

→ Hold fn params, return values

→ Extremely fast

CPU instructions move data in/out of registers and perform arithmetic

## Assembly

→ Class will use X86-64 assembly (current Intel, AMD)

→ Others exist, like ARM, MIPS

Instruction Set Architecture (ISA) Contract between program/compiler, hardware

Defines operations executable by CPU

(data r/w/transfer, control mechanisms)

x86-64 backwards-compatible w/ OG 1978 16-bit arch

## The mov instruction

→ Copies bytes from one location to another

mov SRC, DST

→ src, dst can be

→ Immediate / constant (src only) $0x104

→ Register %rbx

→ Memory location (at most one) 0x60005d4

Ex.

```
mov   $0x104, ___
mov   %rbx, ___
mov   ___, %rbx
mov   0x104, ___
mov   ___, 0x104
```

Move forms

```
mov   (%rbx), ___
mov   ___, (%rbx)
```

parens mean "copy val at addr stored in %rbx"

```
mov   0x10(%rax), ___
mov   ___, 0x10(%rax)
```

Add 0x10 + addr in %rax, then copy

```
mov (,%rax, %rdx), ____
mov ____, (,%rax, %rdx)
```
Sum of addrs at %rdx, %rax

```
mov 0x10(,%rax,%rdx), ____
mov ____, 0x10(,%rax, %rdx)
```
Sum of 0x10 and addrs in %rax, %rdx

## Scaled Indexed Forms

Ex.
```
mov (, %rdx, 4), ____
mov ____, (, %rdx, 4)
```
Mem addr at 4x Val in %rdx
Scaling factor must be hardcoded 1, 2, 4, 8

```
mov 0x4(, %rdx, 4), ____
mov ____, 0x4(, %rdx, 4)
```
Addr at $(4 \times \%rdx) + 4$

```
mov (%rax, %rdx, 2), ____
mov ____, (%rax, %rdx, 2)
```
Addr at $(2 \times \%rdx) + \%rax$

```
mov 0x4(%rax, %rdx, 2), ____
____, 0x4(%rax, %rdx, 2)
```
Addr at $0x4 + (2 \times \%rdx) + \%rax$

General form $Imm(r_b, r_i, s)$

$= Imm + addr[r_b] + (s * addr[r_i])$

missing 0 missing 1