# Assembly, Patents, Copyright

CS 2130: Computer Systems and Organization 1

March 17, 2023

## Announcements

- Homework 5 due Monday 3/20 at 11pm
- Quiz 5 opens tonight, due Sunday

## Patents and Copyright

Can we patent our ISA? Should we?

| icode | b | meaning |
|---|---|---|
| 0 | | `rA = rB` |
| 1 | | `rA += rB` |
| 2 | | `rA &= rB` |
| 3 | | `rA` = read from memory at address `rB` |
| 4 | | write `rA` to memory at address `rB` |
| 5 | 0 | `rA = ~rA` |
| | 1 | `rA = -rA` |
| | 2 | `rA = !rA` |
| | 3 | `rA = pc` |
| 6 | 0 | `rA` = read from memory at `pc + 1` |
| | 1 | `rA` += read from memory at `pc + 1` |
| | 2 | `rA` &= read from memory at `pc + 1` |
| | 3 | `rA` = read from memory at the address stored at `pc + 1` |
| | | For icode 6, increase `pc` by 2 at end of instruction |
| 7 | | Compare `rA` as 8-bit 2's-complement to `0` |
| | | if `rA <= 0` set `pc = rB` |
| | | else increment `pc` as normal |

## Patents and Copyright

### Copyright

- “Everyone is a copyright owner. Once you create an original work and fix it, like taking a photograph, writing a poem or blog, or recording a new song, you are the author and the owner.”

from https://www.copyright.gov/what-is-copyright/

### Patent

- “Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.”

from 35 U.S.C. 101

## Patents

In software and hardware, patents become messy

- Code is a description of a process we want the computer to do
- Do not have to implement the process to patent it

Question: Should we patent something like our ISA?

## Patents

In software and hardware, patents become messy

- Code is a description of a process we want the computer to do
- Do not have to implement the process to patent it

Question: Should we patent something like our ISA?

What is the current state of the art?

## Common Approaches to Software

How can we get value from what we create?

- Copyright - distribute closed source software
- License Agreements (in contract law)
- Always innovate

## Back to Assembly

## Compilation Pipeline

Turning our code into something that runs

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

## Most Common Instructions

- *`mov`* - =
- *`lea`* - load effective address
- *`call`* - push PC and jump to address
- *`add`* - +=
- *`cmp`* - set flags as if performing subtract
- *`jmp`* - unconditional jump
- *`test`* - set flags as if performing &
- *`je`* - jump iff flags indicate == 0
- *`pop`* - pop value from stack
- *`push`* - push value onto stack
- *`ret`* - pop PC from the stack