CSE 390 B Spring 2021

# Final Class! :(

Social Computing Reflection, CSE 390B victory lap, TA-led activity

# Agenda

- **Social Computing Discussion III**
- Victory Lap
- End of Quarter Activity led by your lovely TAs
- Reminders

# Social Computing Reflection Discussion III

- Share the article you found for your third social reflection!

- Things to include:
  - Give a summary of the article
  - Share how it changed/influenced your thinking
  - Share your follow up questions
  - Feel free to have discussions beyond these bullet points too!

# What can we do to change things as computer scientists?

- We've spent some time exploring social issues in computing, and you may feel like you want to help change things about the current computing culture!
  - I know there are things I would definitely like to change

- But it can be hard to feel like you have any power or control over what are very complex issues

# Individualistic Ethics Can be Misleading

- Often our society portrays solutions to systemic issues as a sum of a bunch of individual actions
- While individual choices do add up, this view does not acknowledge the structural pressures placed on individuals that lead them to make certain choices
- It also doesn't account for the things that individuals don't have control over (or very indirect control over)
- Systemic problems need systemic solutions
  - Changing systems to incentivize better choices by individuals
  - Changing systems to prioritize better choices at all levels (not just the individual level)

# Example: Carbon Emissions

- Often our society portrays solutions to systemic issues as a sum of a bunch of individual actions
  - Carbon emissions are so bad because of all the individual choices we make to drive/fly/use lawn mowers/etc.
- While individual choices do add up, this view does not acknowledge the structural pressures placed on individuals that lead them to make certain choices
  - You may have to drive to work because the only bus option requires 2 transfers and takes 2 hours
- It also doesn't account for the things that individuals don't have control over (or very indirect control over)
  - You have little direct control over how energy companies generate their energy (aside from your power to vote)
- Systemic problems need systemic solutions
  - Individual incentives: Providing more options for public transport/greener commutes
  - Better choices at all levels: taxing carbon emissions produced by organizations

# Individualistic Ethics in Computing

- Movements towards ethics/social issues/change in computing often emphasize individual responsibilities

- In reality, there may be very few times where you will be programming and have a direct ethical question in front of you
  - These are often decided at other levels and then passed down as instructions for "what to do"
  - Obviously depends on where you are working, what you are working on, etc.

- But there are still ways in which you can pressure the systems to change!

# Lobbying Your Skills as a Computer Scientist

- Y'all have amazing skills! You get to decide what you want to use them on
- This might be the career choice you make
  - Whether you work at a company, nonprofit, university, etc.
  - What organizations you choose to work for
  - Example: Finding a company/nonprofit that works on an issue you think is meaningful
  - Example: Researching technologies aimed a social good (lots of labs at UW that aim to do this)
- This might be choices you make within a role
  - Advocating for best practices on a project you are working on
  - Declining to work on a project because of concerns over the ethics of it
  - Example: pointing out harmful bias in the design of a product

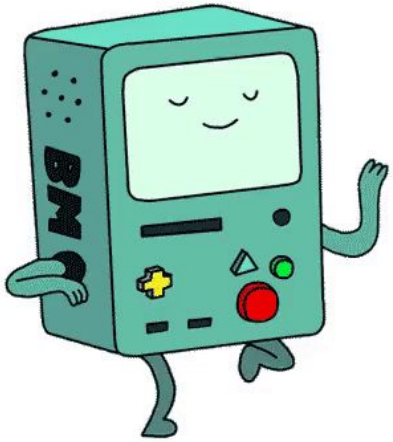
# Pressuring Systems to Change

- You might run into a system (company, institution, committee, etc.) that you think is really difficult to change
  - I super relate to this feeling
- The only way for that system to change is for there to be consistent pressure for it to do so
  - These systems want us to give up trying to change them!
- One way to create pressure is to come together as a collective and advocate for your beliefs
  - There is strength in numbers!
  - A bunch of google employees recently created a union
  - Amazon workers have discussed unionizing

# Agenda

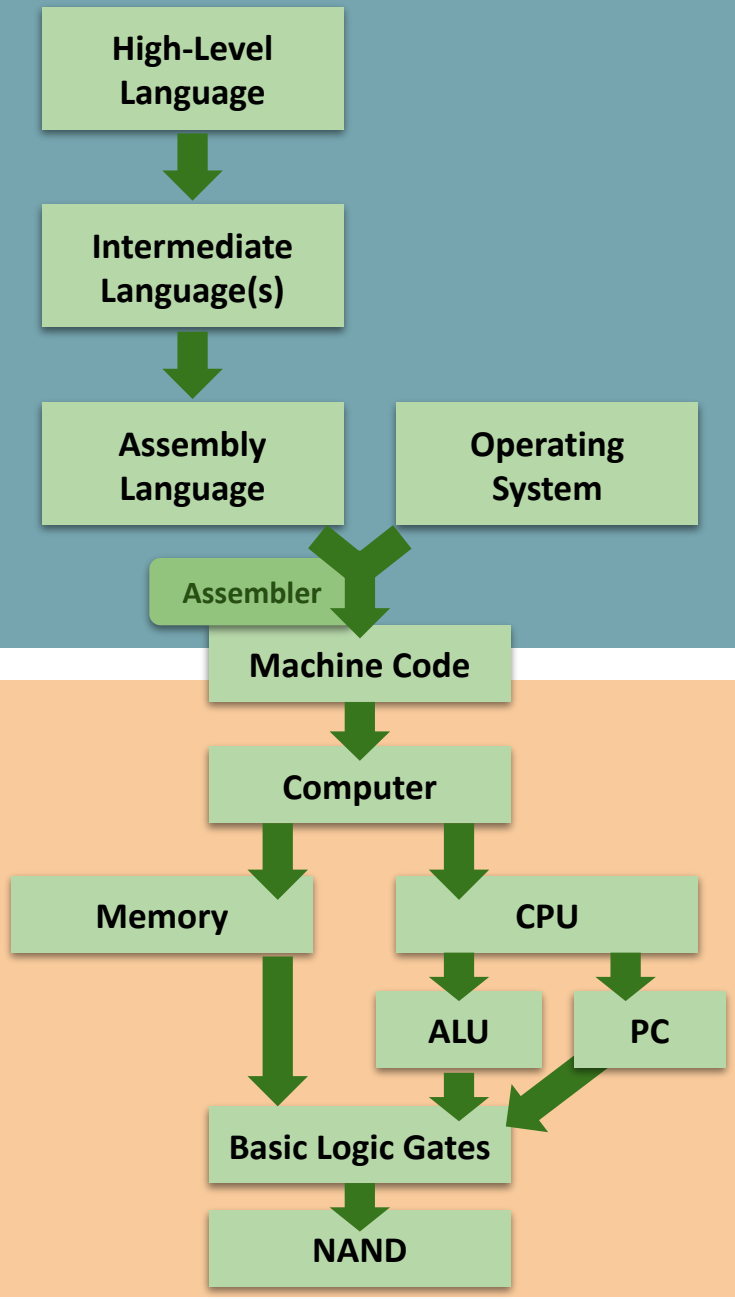

- Social Computing Discussion III
- **Victory Lap**
- End of Quarter Activity led by your lovely TAs
- Reminders

# Nand2Tetris Projects




- By **building a computer**, you've accomplished something that very few CSE students have done!
  - Many software writers consider building the computer as Somebody Else's Problem™
  - But so many technical skills and CSE courses tie in to this task
  - And even if you only write Java for the rest of time…
    - Understanding the "layer below" makes you a better engineer in the "layer above"!

# Roadmap




SOFTWARE

HARDWARE

## Refining Our Theorem Even More

**Theorem**

Any function can be represented by a combination of And & Not.

Example

Not(z) And Not(*x* And *y*)

Not & And can be represented with Nand:

**N x = (x Nand x)**

**t(x Nand y)**

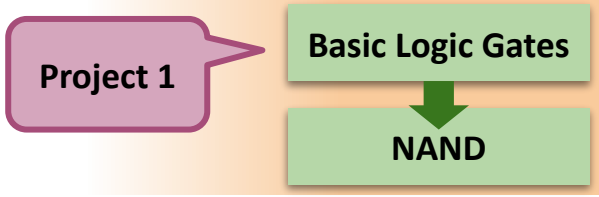
**Theorem**

Any function can be represented solely by Nand operations.

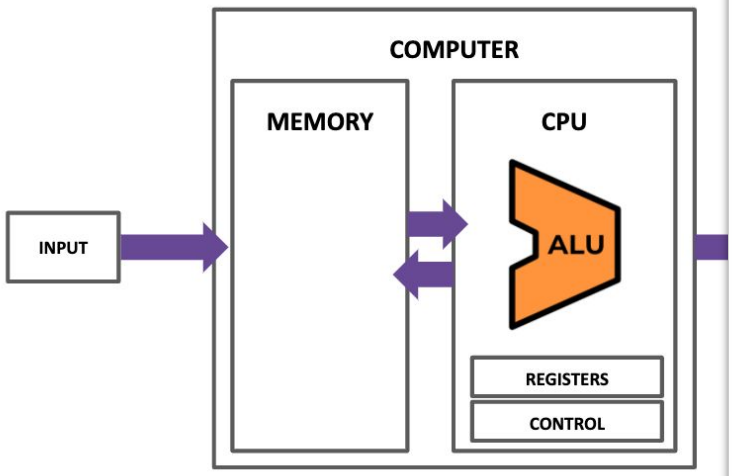
nand (*x* Nand *y*)) Nand

nd z) Nand (*x* Nand *y*))

- Boolean function synthesis
- Practice with an unfamiliar, declarative style of programming

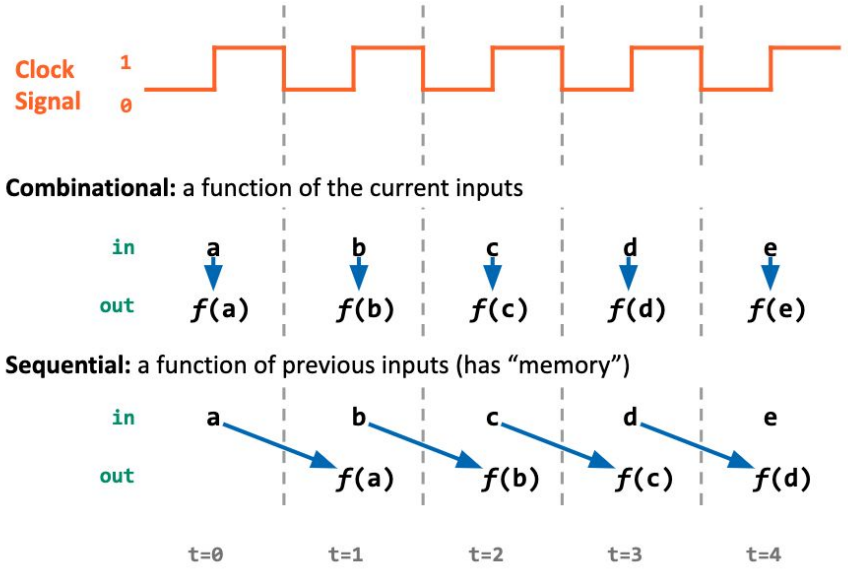


## The Von Neumann Architecture

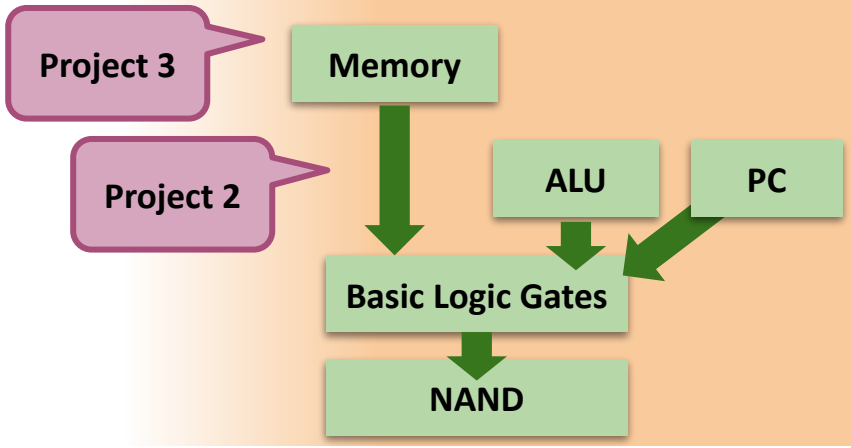



(This picture will get more detailed as we go!)

## Combinational vs. Sequential Abstraction




- Components found in "real-world" computers: ALU, PC, Memory...
- Learning a mental model for sequential logic



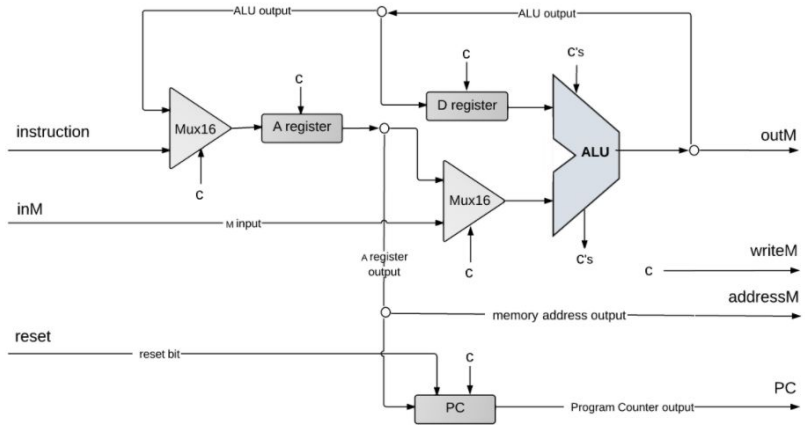
## Hack: C-Instructions

Symbolic: `dest = comp ; jump`

Binary: `1 1 1 a c1 c2 c3 c4 c5 c6 d1 d2 d3 j1 j2 j3`

- **Family:** C-Instruction
- *unused*
- **Comp:** ALU Operation (a bit chooses between A and M)
- **Dest:** Where to store result
- **Jump:** Condition for jumping

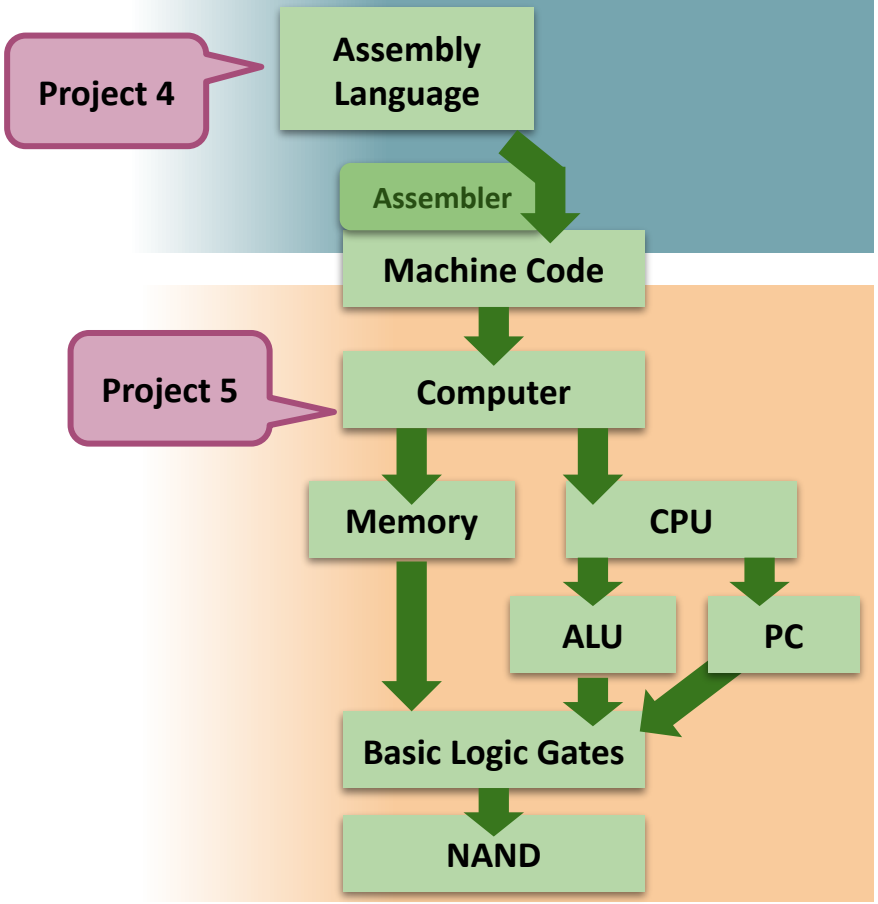
## Hack CPU Implementation




(each "C" symbol represents a control bit)

26

- The connection between software and hardware through binary instructions
- What has to happen in a clock cycle to process one instruction



## The Compiler: Implementation

```
public int fact(int n) {
  if (n == 0) {
    return 1;
  } else {
    return n * fact(n - 1);
  }
}
```
High-Level Language

```
(fact)
  @R0
  M=M+1
  @R1
  D=A
  @ifbranch
  D;JEQ
```
Assembly Language

Scanner → Parser → Type Checker → Optimizer → Code Generator

Break string into discrete **tokens**: IF ( ID(n) == NUM(0) etc.

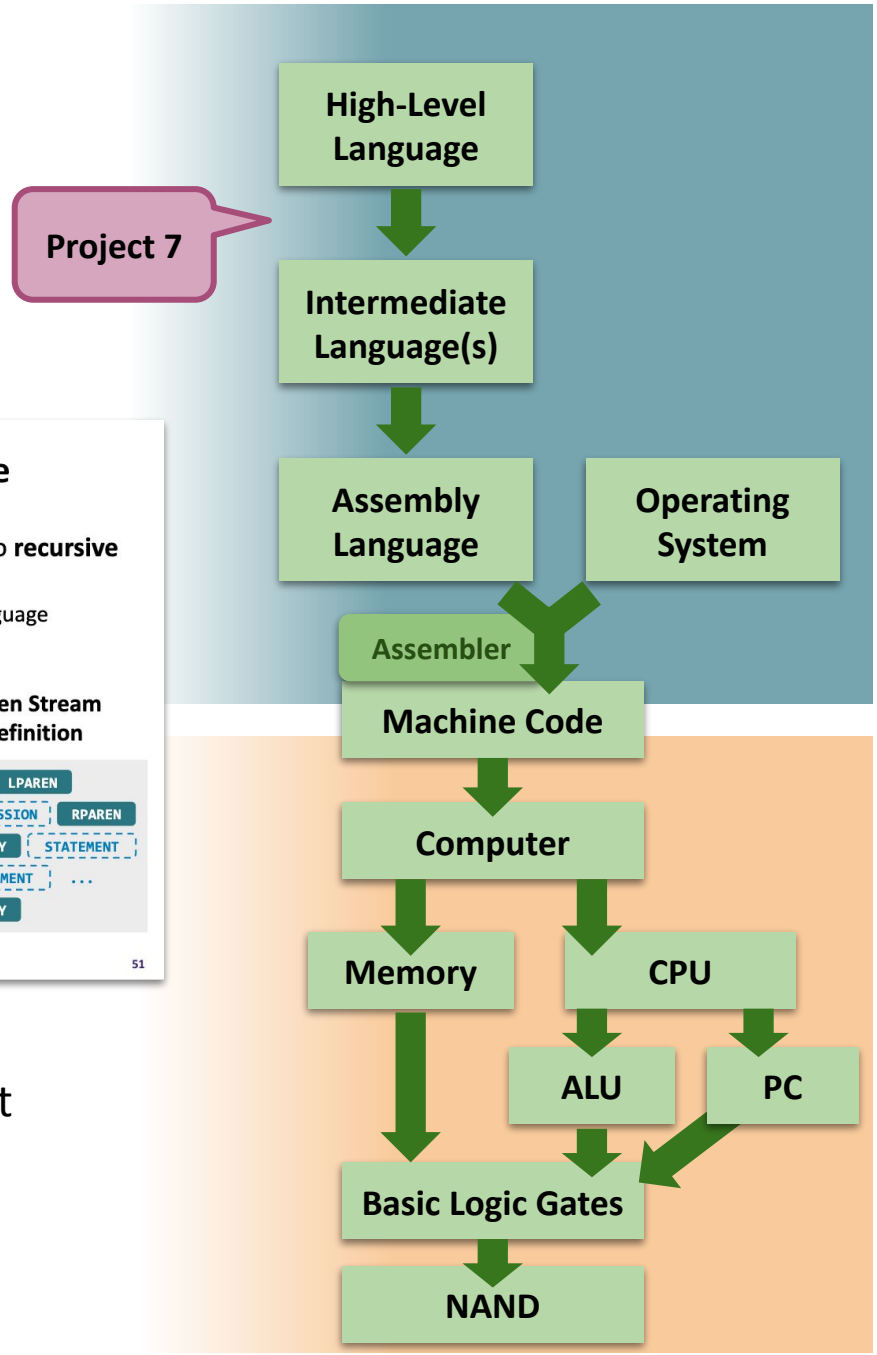
Arrange tokens into **syntax tree**: + (x, 10)

## Describing a Programming Language

- These broad categories lend themselves well to **recursive definitions**
  - Easily express all possible configurations of the language constructs

| Symbolic Example | General Definition of an `if` Statement | Token Stream Definition |
|---|---|---|
| `if (x == 0) { x = y; }` | `if ( EXPRESSION ) { STATEMENT STATEMENT ... }` | IF LPAREN EXPRESSION RPAREN LCURLY STATEMENT STATEMENT ... RCURLY |

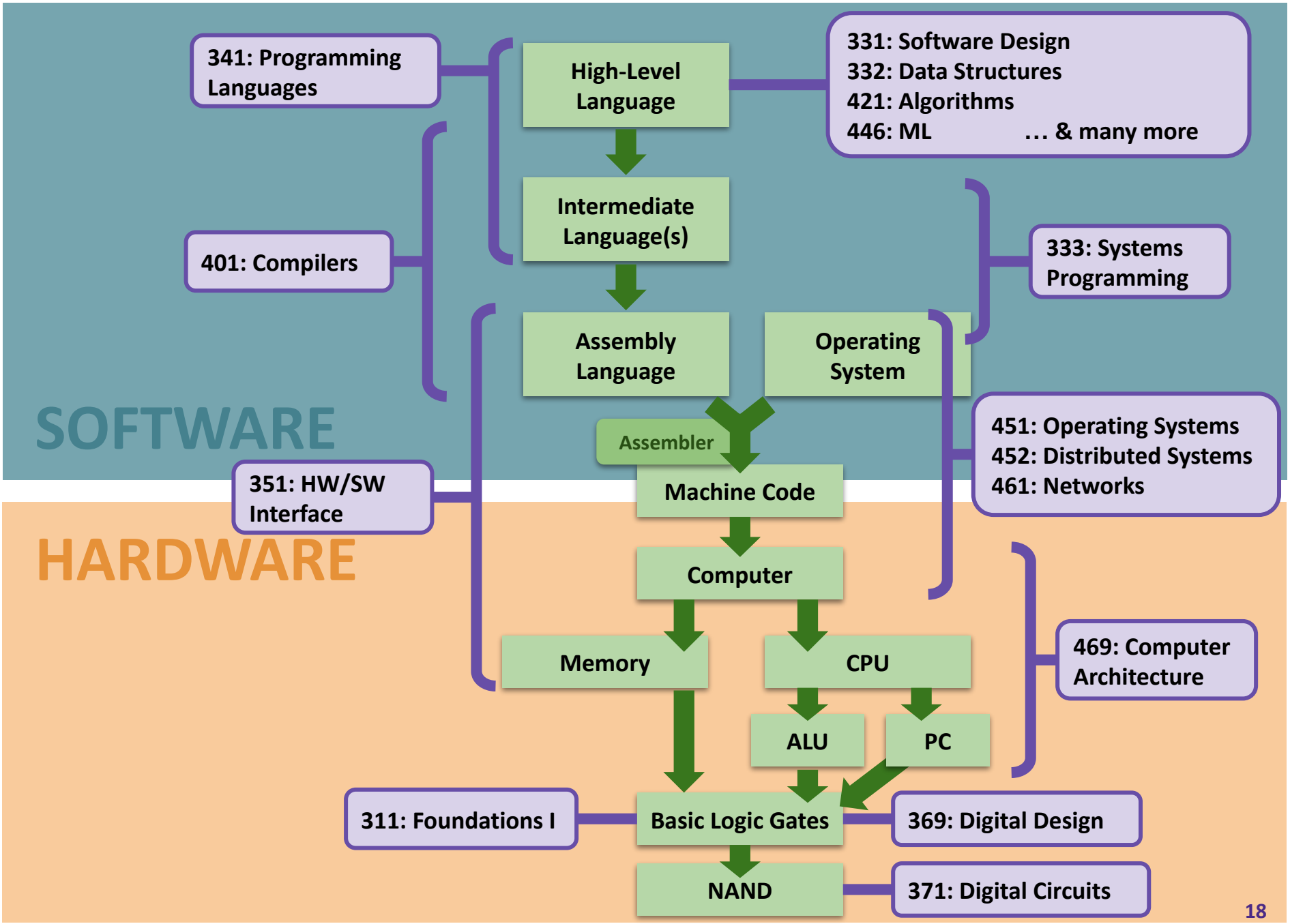
51



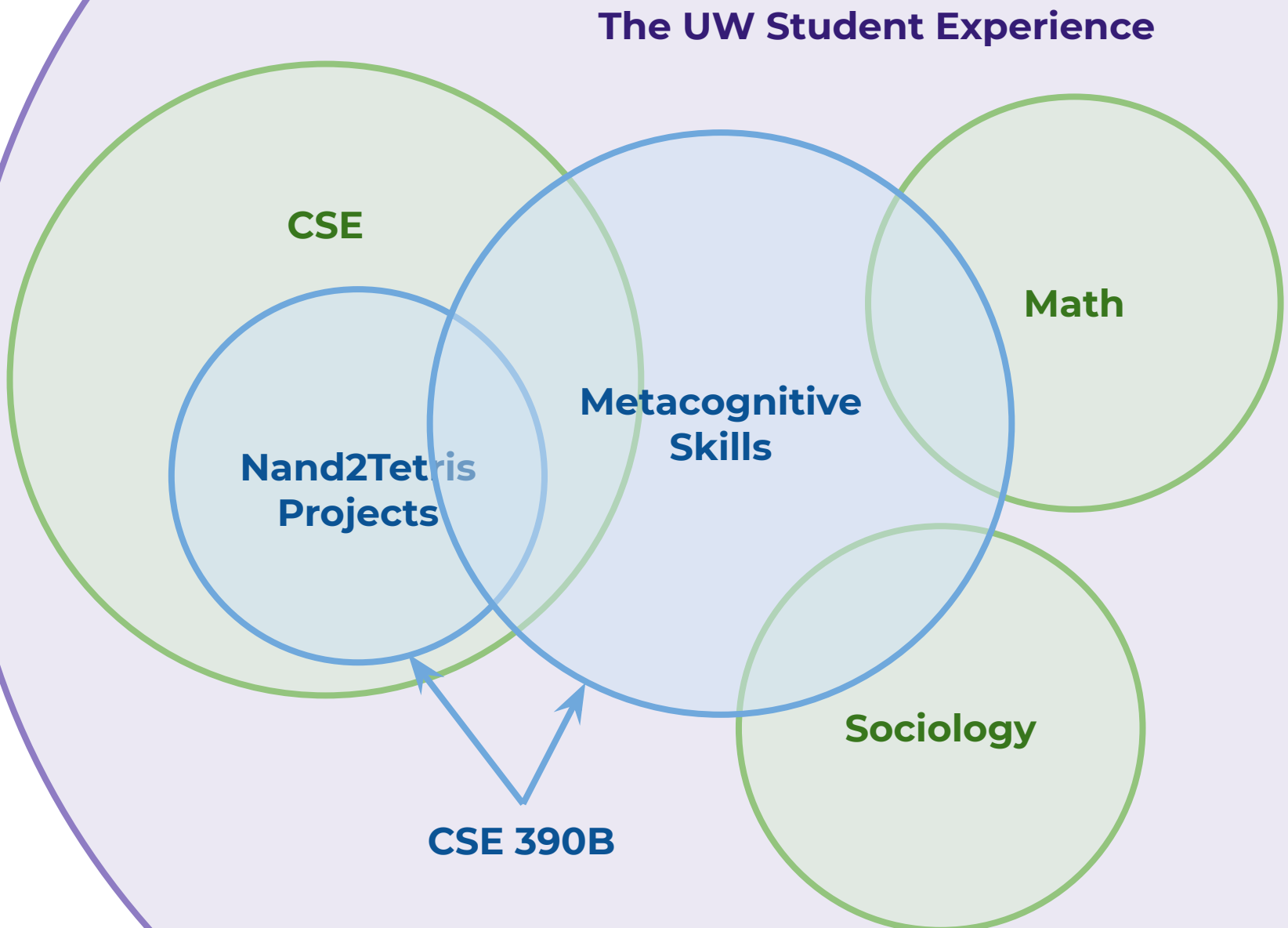

- What actually happens when you click that green run button?
- Programs can read in programs and then spit out equivalent programs

# Takeaways: Why Build a Computer?

- Undeniably a **significant engineering effort**
  - You practiced so many skills and programmed with so many different languages, tools, & paradigms -- and you can do it again!
- We hope this was **demystifying experience**
  - To see CSE not as isolated courses, but as a big inter-connected jigsaw puzzle.
- And hopefully at least a little **fun** :)
  - CSE has something for everyone, and with any luck you found something exciting enough to pursue further.

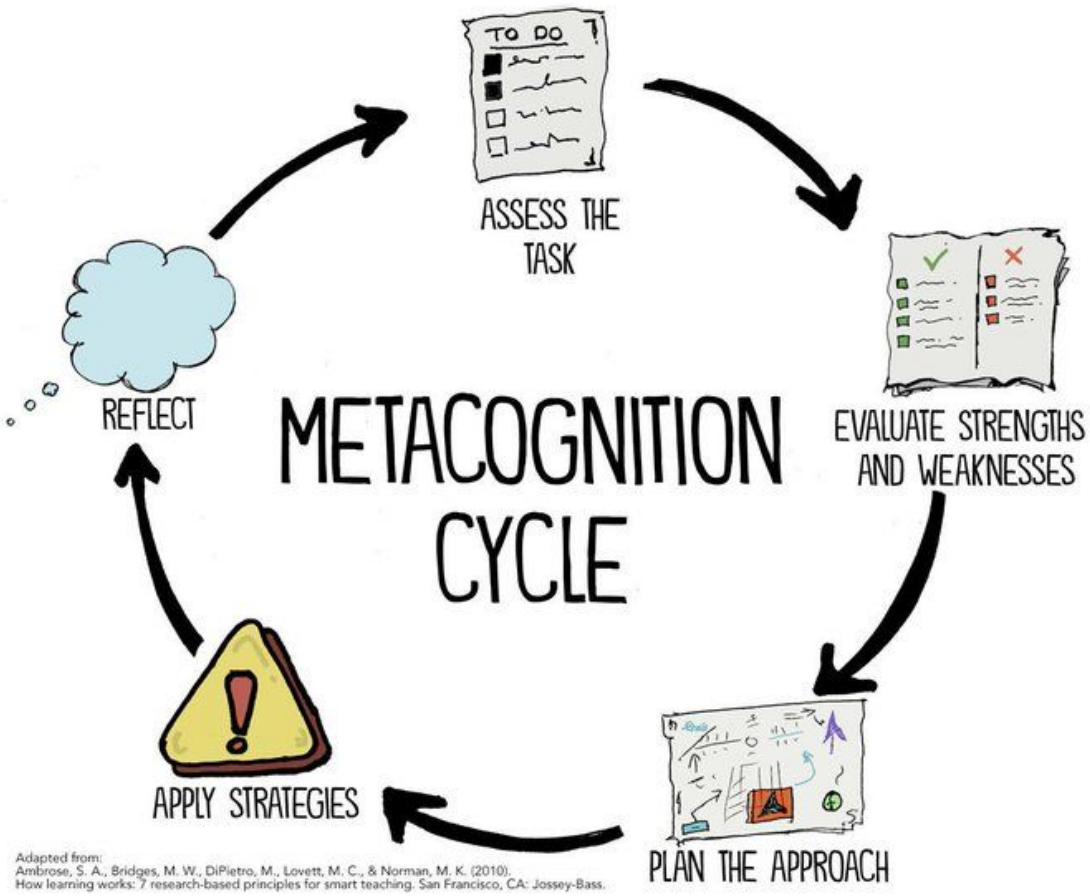

341: Programming Languages
High-Level Language
331: Software Design
332: Data Structures
421: Algorithms
446: ML
… & many more
Intermediate Language(s)
401: Compilers
333: Systems Programming
Assembly Language
Operating System
SOFTWARE
Assembler
451: Operating Systems
452: Distributed Systems
461: Networks
351: HW/SW Interface
Machine Code
HARDWARE
Computer
Memory
CPU
469: Computer Architecture
ALU
PC
311: Foundations I
Basic Logic Gates
369: Digital Design
NAND
371: Digital Circuits

# Remember this?



# Metacognitive Skills Victory Lap!

- Time Management
- Note-Taking
- Annotation
- Exam Preparation
- Test-Taking
- Debugging
- Working with Instructors & TAs
- Design Decisions
- Oral Communication




VICTORY SCREECH

# Agenda

- Social Computing Discussion III
- Victory Lap
- **End of Quarter Activity led by your lovely TAs**
- Reminders

# Agenda

- Social Computing Discussion III
- Victory Lap
- End of Quarter Activity led by your lovely TAs
- **Reminders**

# Reminders!

- Office Hours ending this week
  - Happy to meet during finals week by appointment
- Final Project Part I: Project Outline
  - Due TONIGHT 11:59PM PST
- Final Project Part II: Mock Presentation
  - Due next Wed 6/9 11:59PM PST
- Final Project Presentations Thursday June 10th @ 10:30AM PST
- Course Evaluations
  - Please, please, pleaseeeeeee complete this!

# Thanks for an incredible quarter!

Stay in touch :)

*-- The CSE 390B Staff*