CSE 390B, Autumn 2022

Building Academic Success Through Bottom-Up Computing

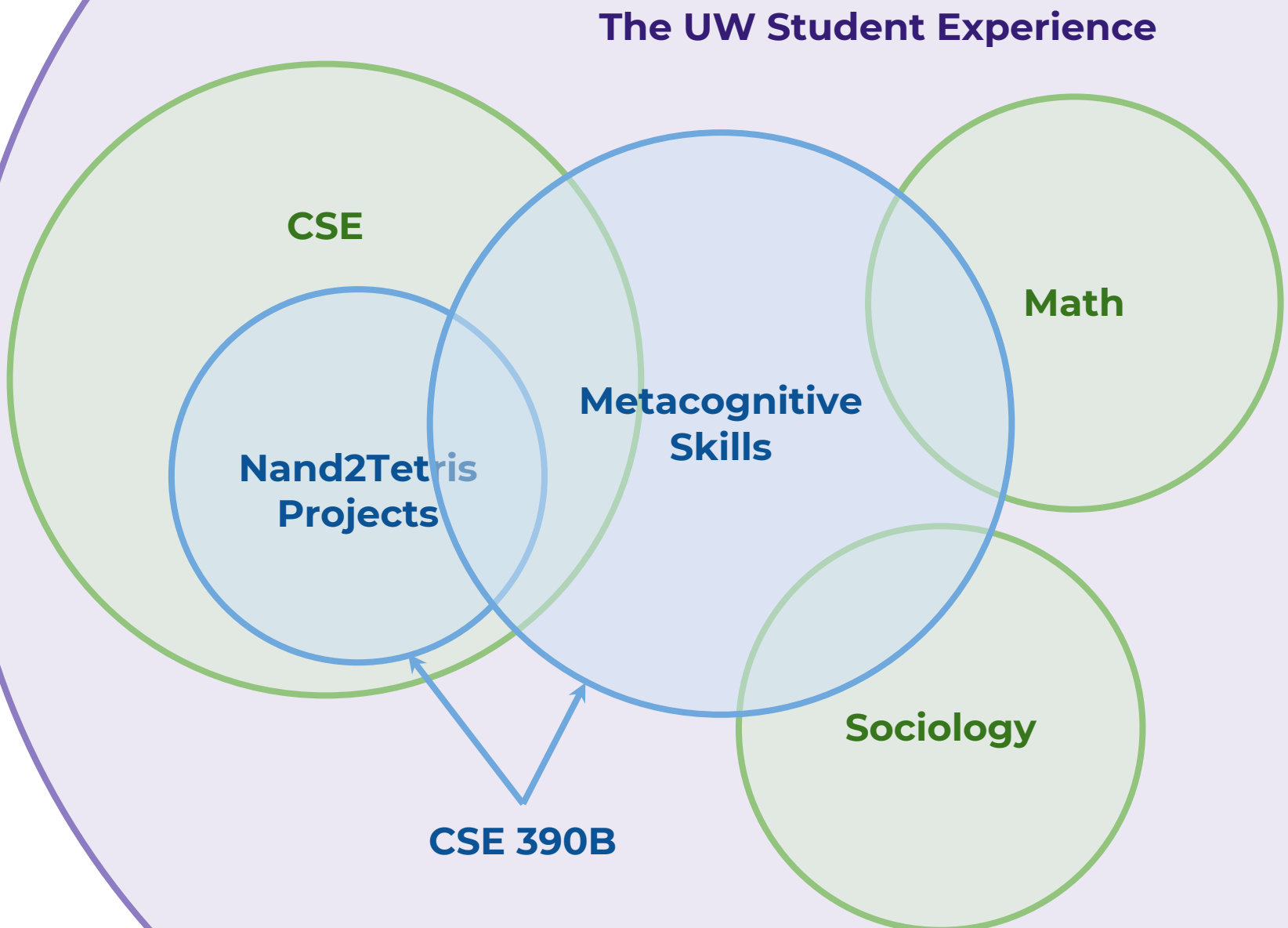
# Course Wrap-up & TA-led Activities

CSE 390B Reflection and Victory Lap, CSE 390B Course Staff Panel, Jeopardy Game

# Lecture Outline

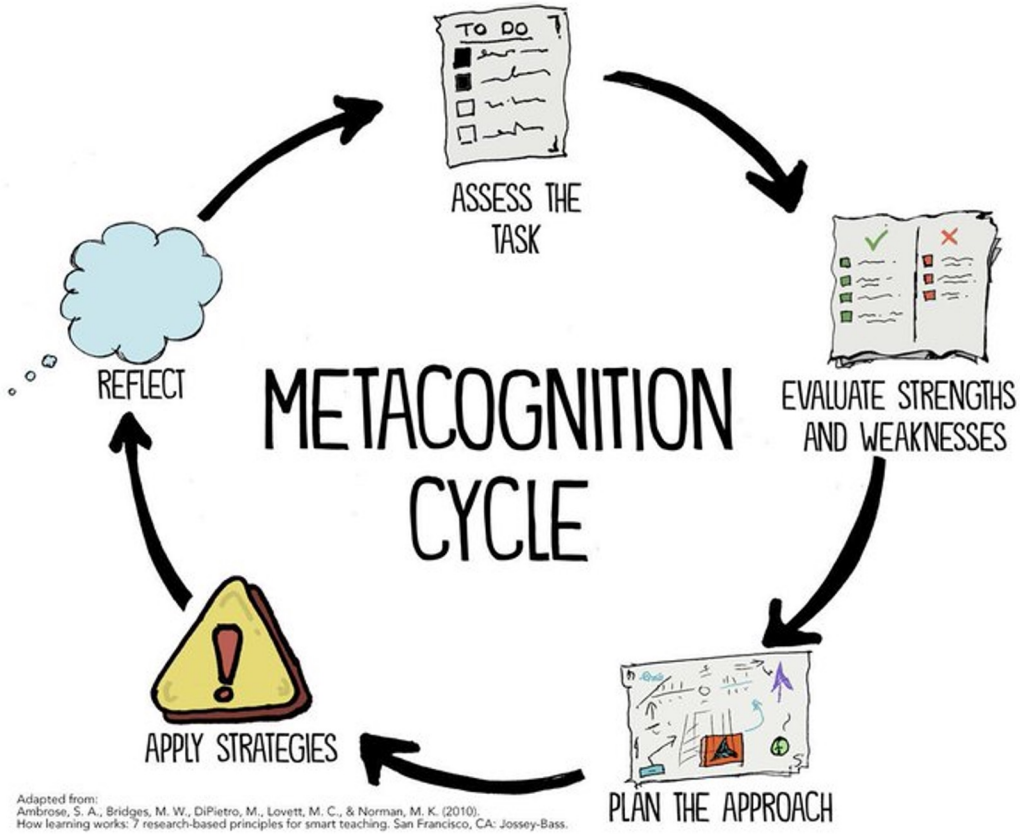
- **CSE 390B Reflection and Victory Lap**
  - **Metacognitive Skills**
  - **Nand2Tetris Projects**
- CSE 390B Course Staff Panel
  - Ask About Class Recommendations, Extracurriculars, etc.
- Jeopardy Game
  - Topics: UW History, CSE 390B Course Staff, Seattle, Pop Culture

# Remember this?

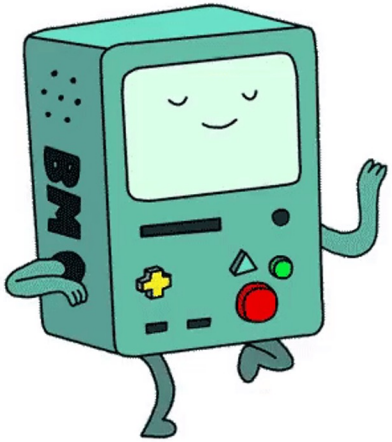


# Metacognitive Skills Victory Lap!

- Time Management
- Note-Taking
- Annotation
- Exam Preparation
- Test-Taking
- Post-exam Reflection
- Debugging
- Working with Instructors & TAs
- Reflection on Metacognition



# Nand2Tetris Projects



- By building a computer, you've accomplished something that very few others have done!
  - Many software writers consider building the computer as Somebody Else's Problem™
  - But so many technical skills and CSE courses tie into 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"!

## Project 1 Example: Xor (cont'd)

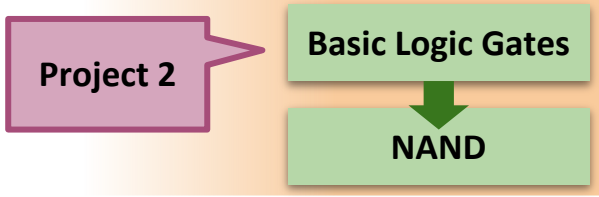
- Step 2: Use truth table to generate a Boolean function
  - Let's use the Boolean function synthesis strategy from the reading
  - Row 2 = NOT(A) AND B
  - Row 3 = A AND NOT(B)
  - A XOR B = Row 2 OR Row 3

= (NOT(A) AND B) OR (A AND NOT(B))

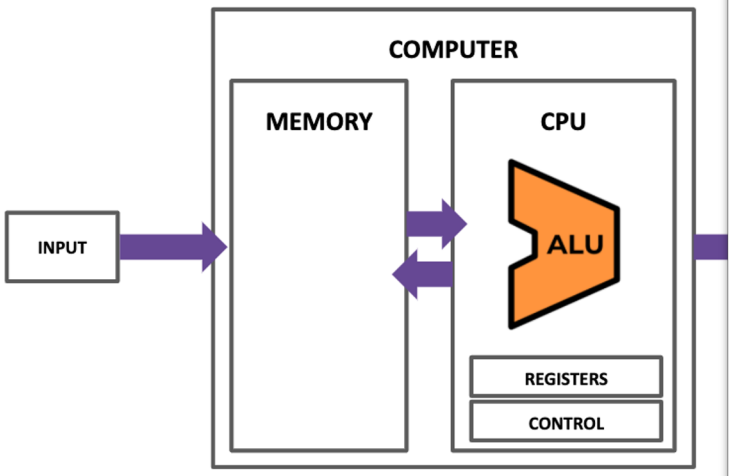
| A | B | F | |
|---|---|---|---|
| 0 | 0 | 0 | (Row 1) |
| 0 | 1 | 1 | (Row 2) |
| 1 | 0 | 1 | (Row 3) |
| 1 | 1 | 0 | (Row 4) |

F = A XOR B

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

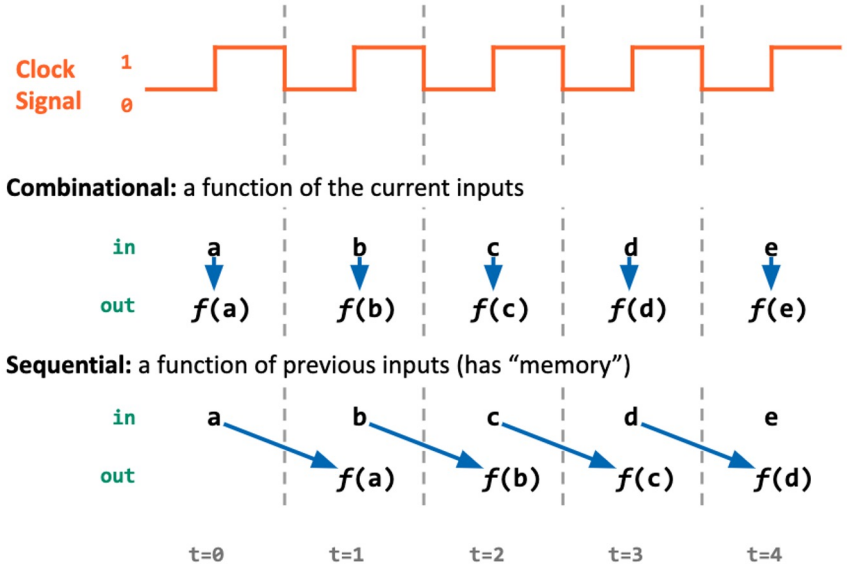


## The Von Neumann Architecture

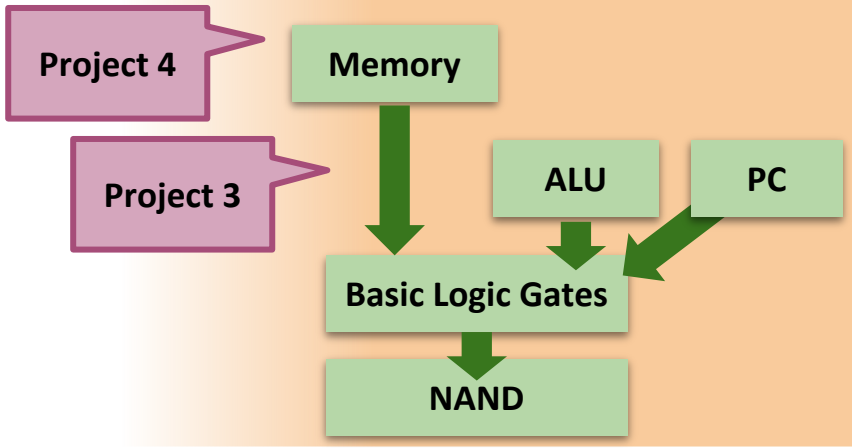



(This picture will get more detailed as we go!)

## Combinational vs. Sequential Abstraction




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



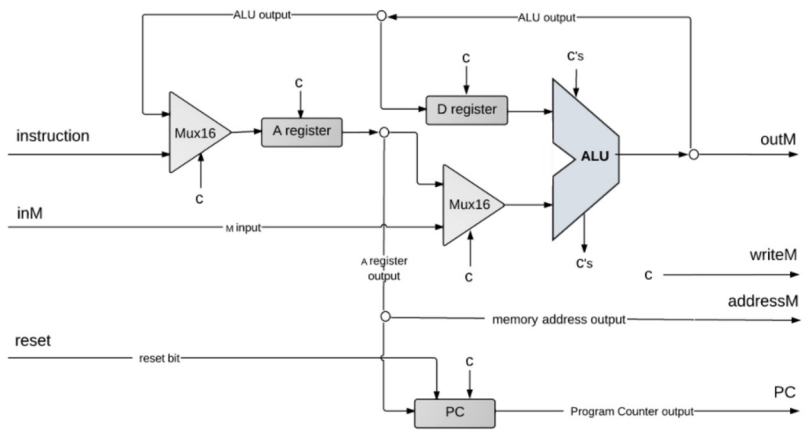
## Hack: C-Instructions

Symbolic: `dest = comp ; jump`

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

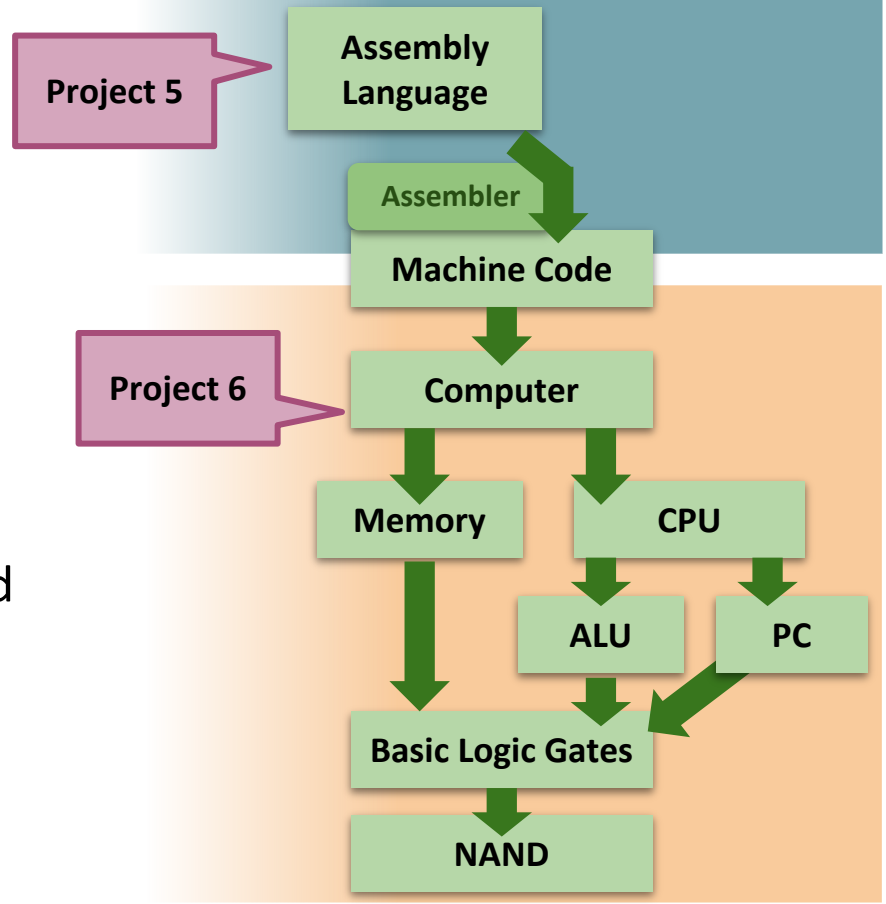
- Family: C-Instruction
- unused
- Comp: ALU Operation (a bit chooses between A and M)
- Dest: Where to store result
- Jump: Condition for jumping

## Hack CPU Implementation




(each "C" symbol represents a control bit)

26




- The connection between software and hardware through binary instructions
- What must 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.

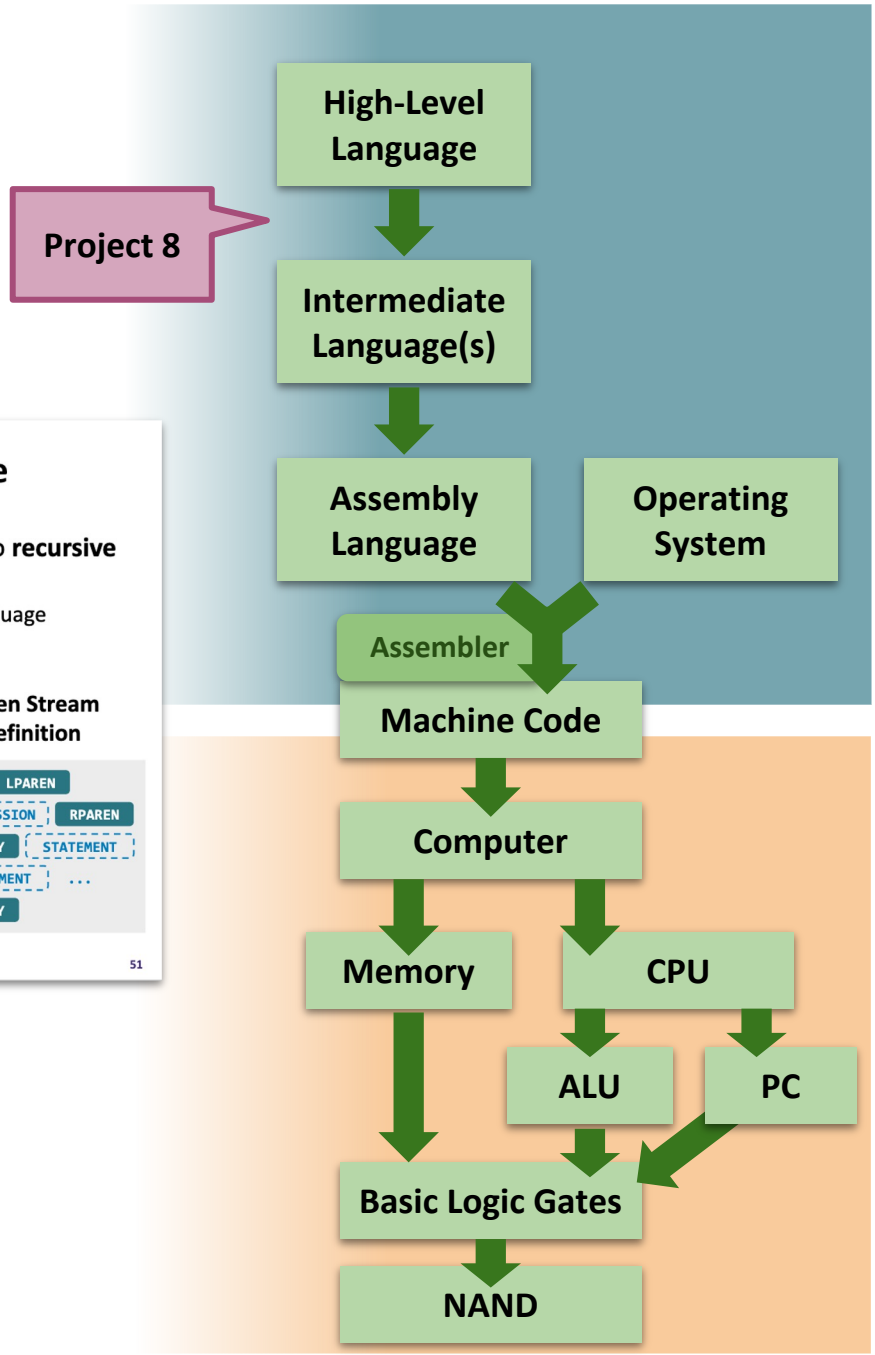
Arrange tokens into **syntax tree**: + (x, 10)

## Describing a Programming Language

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

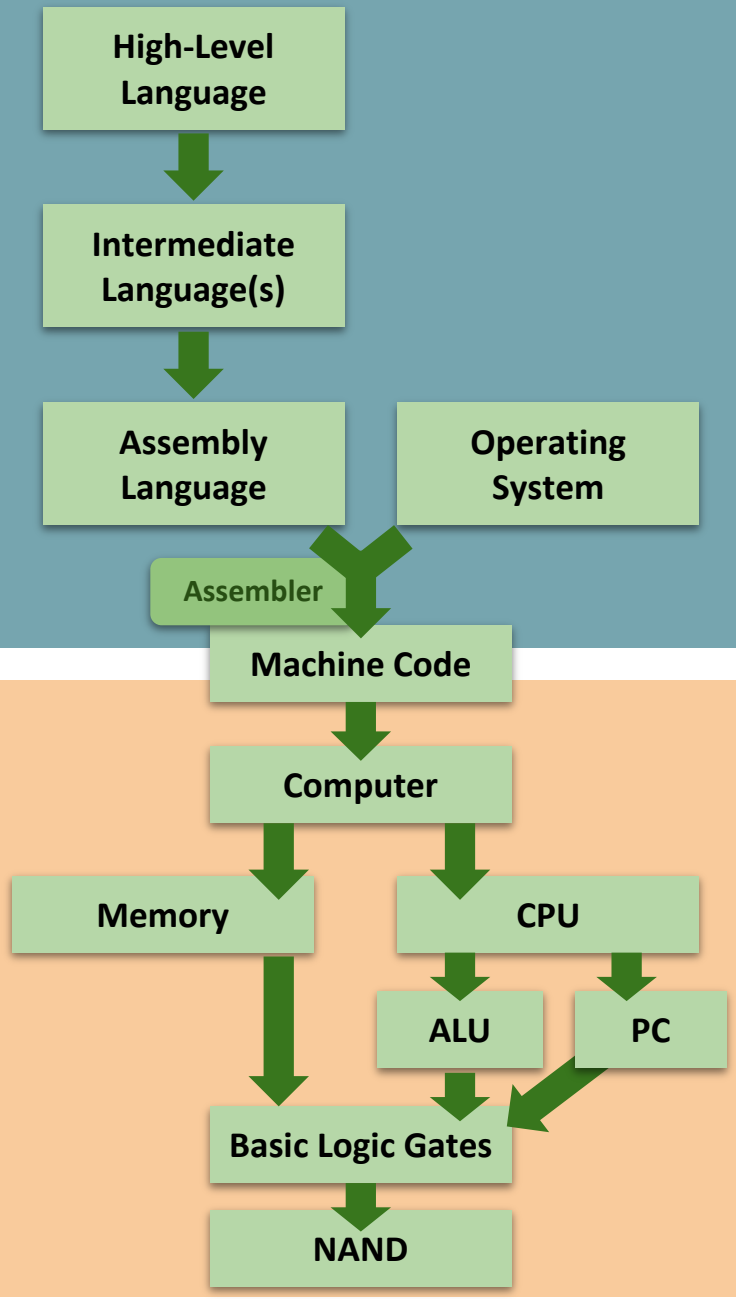
| Symbolic Example | General Definition of an `if` Statement | Token Stream Definition |
|---|---|---|
| `if (x == 0) { x = y; }` | `if ( EXPRESSION ) { STATEMENT STATEMENT ... }` | IF LPAREN EXPRESSION RPAREN LCURLY STATEMENT STATEMENT ... RCURLY |

51



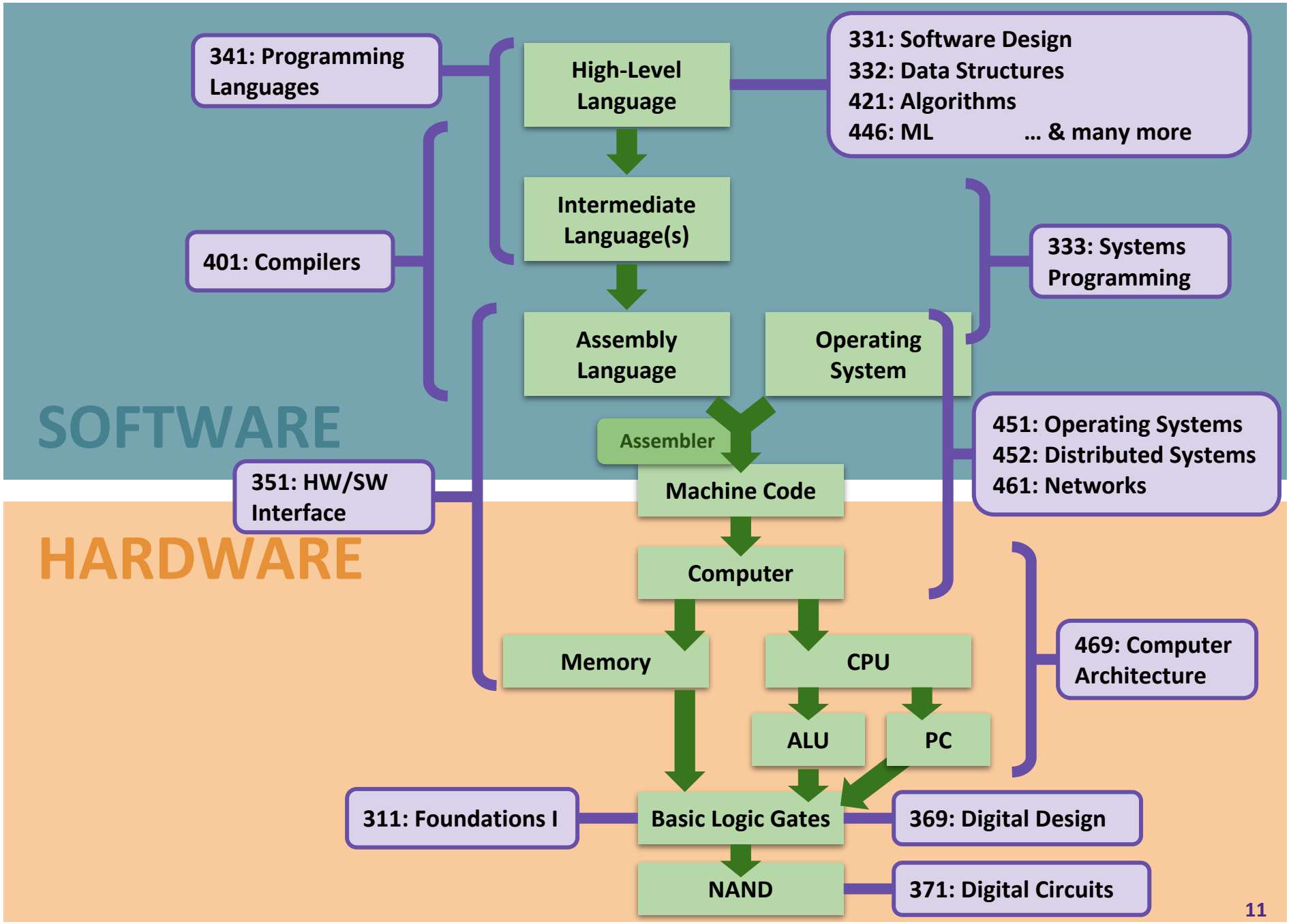

- What happens when you click that green run button?
- Programs can read in programs and then spit out equivalent programs

# Roadmap




SOFTWARE

HARDWARE


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

# Takeaways: Why Build a Computer?

- **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 a demystifying experience**
  - To not see CSE courses as isolated but as interconnected puzzle pieces
- **We hope you had fun in this learning journey!**
  - The computing field is broad and has much for you to explore
  - We are hopeful you found a topic you want to pursue further, both technically and metacognitively

# Lecture Outline

- CSE 390B Reflection and Victory Lap
  - Metacognitive Skills
  - Nand2Tetris Projects
- **CSE 390B Course Staff Panel**
  - **Ask About Class Recommendations, Extracurriculars, etc.**
- Jeopardy Game
  - Topics: UW History, CSE 390B Course Staff, Seattle, Pop Culture

# CSE 390B Course Staff Panel

Ask us about:

- Classes
  - Recommendations for easy, hard, useful, etc. classes
  - What classes go well with each other
- Extracurricular activities
  - TAing
  - Research
  - Allen School RSOs
  - UW RSOs
- Internships

# Lecture Outline

- CSE 390B Reflection and Victory Lap
  - Metacognitive Skills
  - Nand2Tetris Projects
- CSE 390B Course Staff Panel
  - Ask About Class Recommendations, Extracurriculars, etc.
- **Jeopardy Game**
  - **Topics: UW History, CSE 390B Course Staff, Seattle, Pop Culture**

# Jeopardy Game

- Organize into teams of 3-4 students

- The first team to raise their hand and answer the question wins that round and chooses the next question

# Post-Lecture 20 Reminders

- Office hours and student-TA 1:1 meetings end this week
  - Course staff open to meeting during finals week by appointment

- Project Reminders
  - **Final Project, Part I: E-Portfolio Outline due tonight (12/8) at 11:59pm**
  - Final Project, Part II: Final E-Portfolio due next Tuesday (12/13) at 4pm
  - If you have any uncompleted projects, the last day to turn them in is next Friday (12/16) at 11:59pm

- Please fill out course evaluations if you haven't already