**Nand2Tetris Notes**

**About this Course**

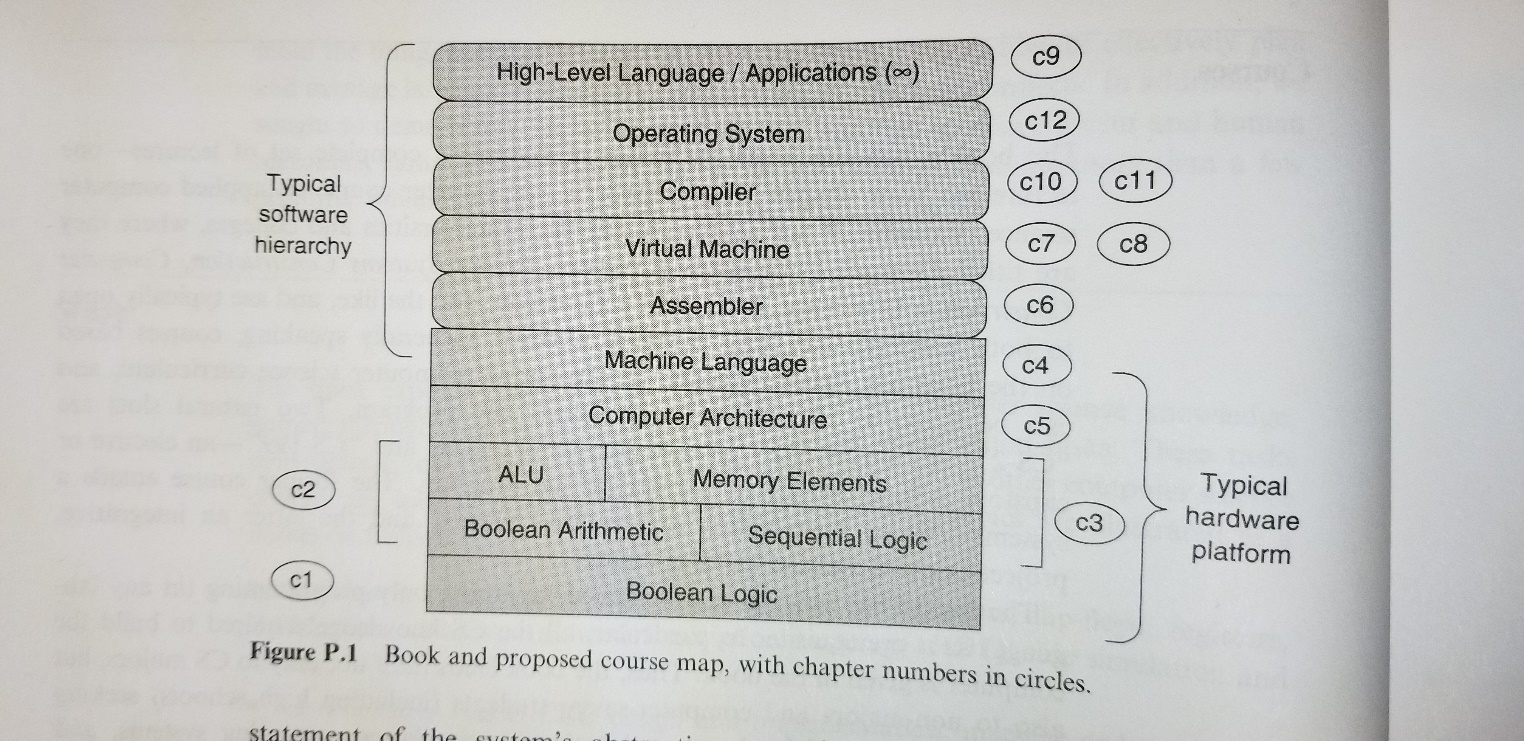
The goal of the course is to build a complete working computer system from simple Nand logic gates. We start by implementing basic logic chips(nand, xor, nor, etc..) with a supplied hardware description language and we finish with a full fledged computer capable of running programs like Tetris.

Figure 1- This is an overview of the Course

**Scope**

1. ***Hardware:*** Logic gates, Boolean arithmetic, multiplexors, flip-flops, registers, RAM units, counters, Hardware Description Language (HDL), chip simulation and testing.
2. ***Architecture:*** ALU/CPU design and implementation, machine code, assembly language programming, addressing modes, memory-mapped input/output (I/O).
3. ***Operating Systems:*** Memory management, math library, basic I/O drivers, screen management, file I/O, high level language support.
4. ***Programming Languages:*** Object-based design and programming, abstract data types, scoping rules, syntax and semantics, references.
5. ***Compilers:*** Lexical analysis, top-down parsing, symbol tables, lists, recursion, arithmetic algorithms, running time considerations.
6. ***Data Structures and Algorithms:*** Stacks, hash tables, lists, recursion, arithmetic algorithms, geometric algorithms, running time considerations.
7. ***Software engineering:***  Modular design, the interface/implementation paradigm, API design and documentation, proactive testing planning, programming at the large, quality assurance.

Chapter 1 Notes:

**Goal: Implement basic and composite logics gates only using the primitive Nand gate.**

Gate List – Not, And, Or/Xor, Multiplexor/Demultiplexor, Multi bit versions and multiway gates.

-Similar to the role of axioms in mathematics, primitive gates provide a set of elementary building blocks from which everything else can be built.

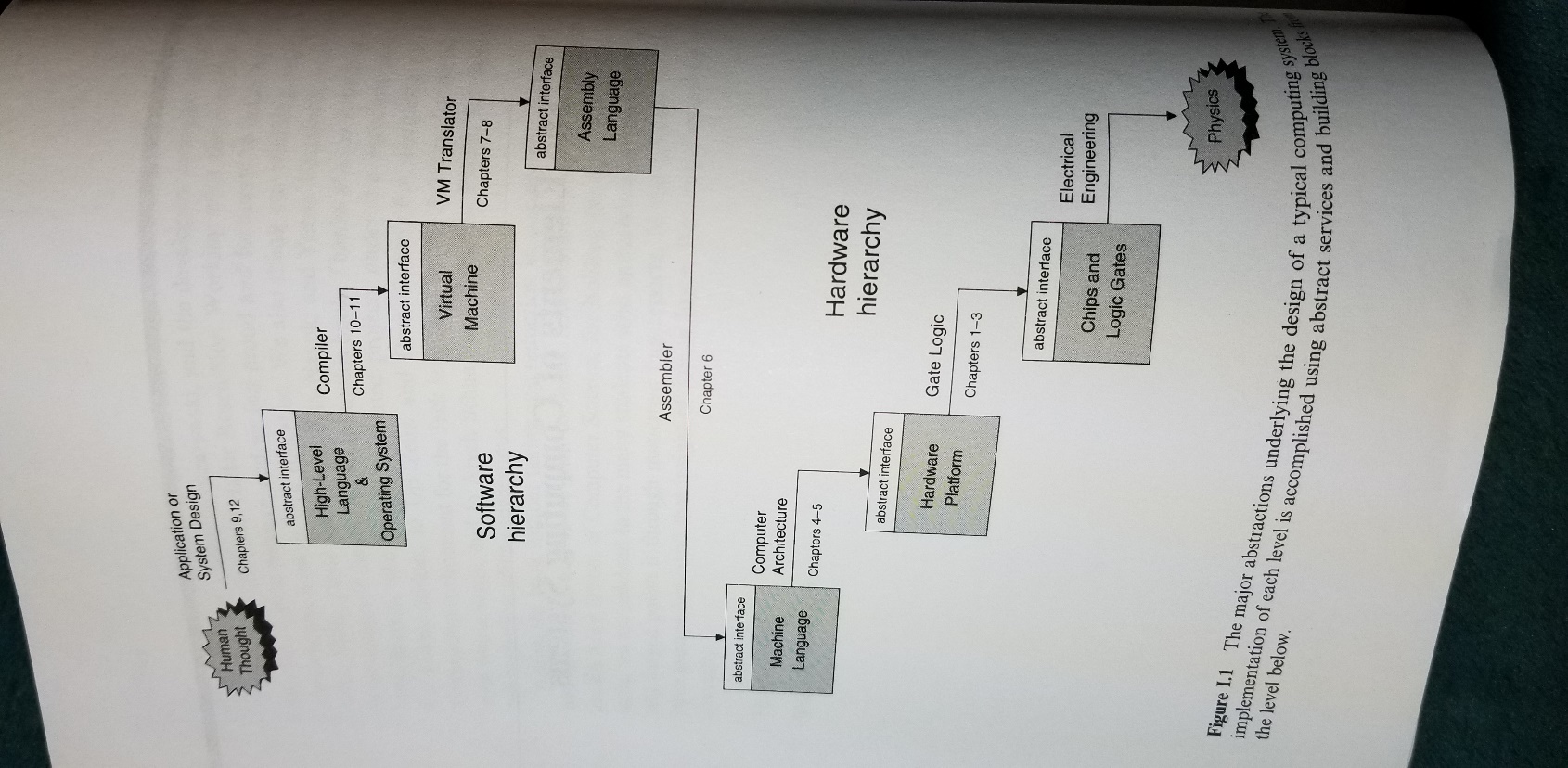
**This link is very helpful in implementing the chips-**

<http://nand2tetris-questions-and-answers-forum.32033.n3.nabble.com/Hardware-Construction-Survival-Kit-td3385741.html>

http://nand2tetris-questions-and-answers-forum.32033.n3.nabble.com/

**Notes:**

1. Read about HDL software to learn which software is best.
2. If time permits go back and implement a few more of the chips but read / ask in forums about the most logical way of going about doing it.
3. There is no visual way of connecting the chips available in this HDL, but there are HDL software tools that are best for visualization and optimization.
4. All chips can be built from the primitive Nand gate.



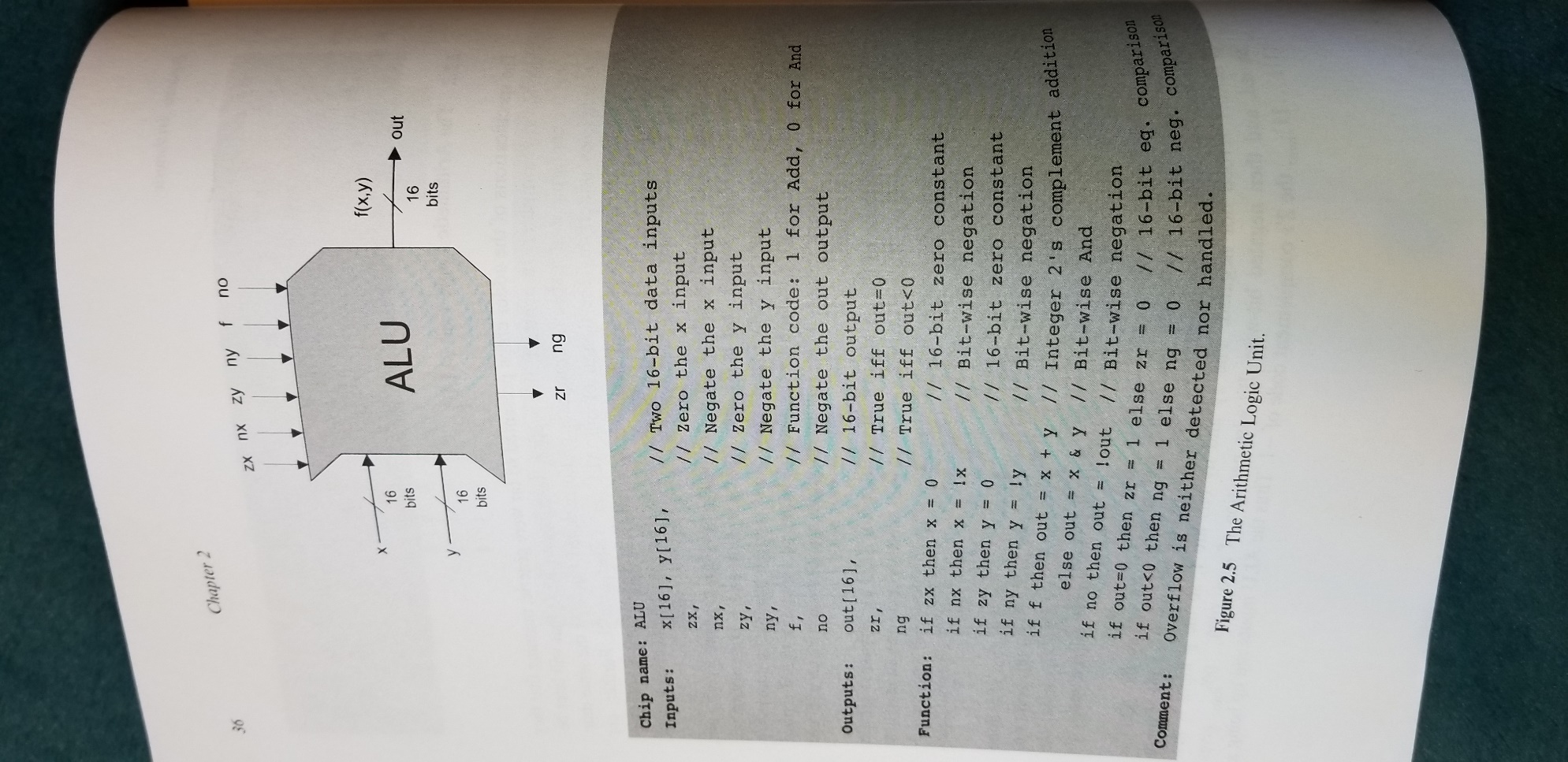
Chapter 2 Notes:

**Goal: Implement an ALU by integrating all the chips we have made using simple Boolean operations.**

Chip List – Half-adder, Full-Adder, Adder, Incrementor, ALU.

**Notes-**

1. Coding schemes like 2’s complement have been developed to represent signed(+,-) binary numbers.



1. I am not going to go into understanding the logic behind 2’s complement. Can review later if necessary.
2. The important thing to remember is that the 2’s complement scheme is very useful because it allows us to implement addition and subtraction with only an adder, and no other special hardware is needed.
3. This leads to the result that a single chip ALU can perform all basic arithmetic and logical operations in hardware.
4. There is a design trade off between hardware and software (ALU and OS). For the ALU in this course we have chosen an ALU with limited functionality, so we will implement other operations in software (multiplication, division, floating point arithmetic). We can implement this at the OS level. Usually ALU functionality is a cost/performance issue. More can be learned in computer architecture books.

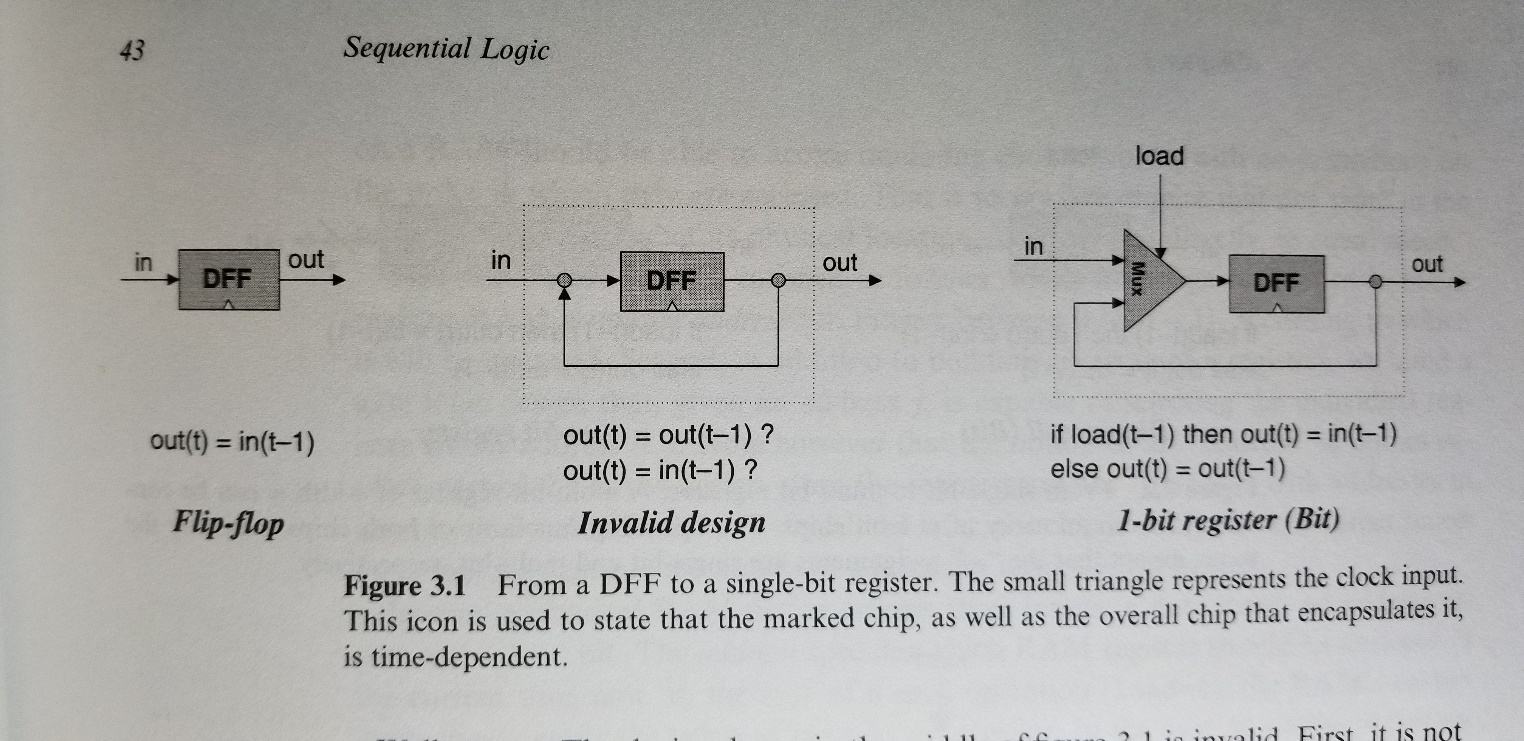
Chapter 3 Notes:

**Goal: In the previous chapters we used combinational logic chips with Nand as the primitive logic gate. In this chapter we will implement sequential chips (flip-flop as primitive) in order to form the basis for computer memory(RAM).**

Chip List – Clock, Data Flip Flops, Registers, Memories, Counters

**Notes-**

1. Sequential chips are ‘clocked’ meaning they update and can change during a computer clock cycle even if the input has not changed. The time unit starts at tick and ends at tock at which point the sequential chips values are updated. We can control the clock manually or with a script.
2. We can build a 1 bit register (our first memory) by using a flip-flop and a multiplexor. RAM is stacked registers and accepts data input, an address input, and a load bit (read/write).
3. When we build the computers clock we have to make sure the length of the cycle is longer than the time it takes for a bit to travel from one chip in the architecture to another.
4. The cornerstone of all memory systems in this chapter is the Data flip-flop, which can in turn be built from Nand Gates. There is a book called ‘Code’ by Charles Petzold that helps in understanding the Flip-Flop since this book deems it primitive.
5. Modern memory chips are not always built from flip-flops but are carefully optimized, by exploiting physical properties.



Chapter 4 Notes:

**Goal: In this chapter we will get a taste of low-level programming in machine language by using the Hack computer platform. We will use an assembler and a CPU emulator that are supplied to run binary programs on a simulated Hack platform. We will use a plain text editor to write the program in assembly, then translate it with the supplied assembler and test it with the supplied CPU emulator. Once we finish building the hardware platform in the next chapter, we can run the program directly on the hardware.**

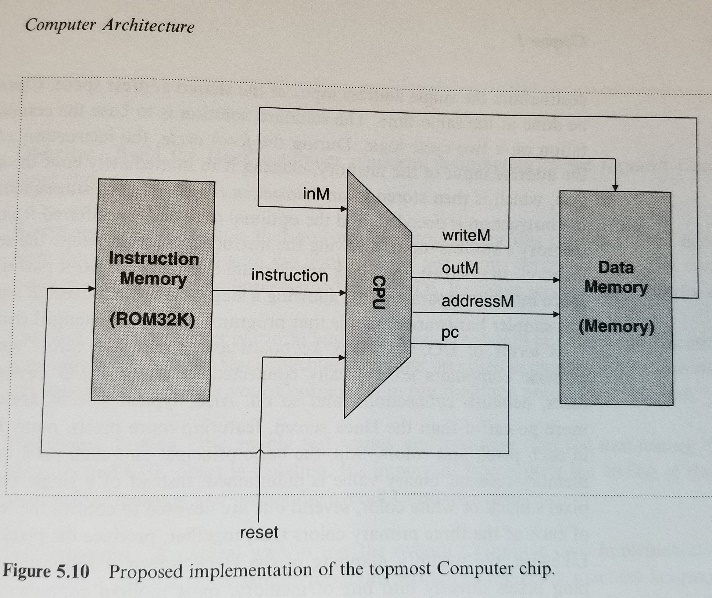
Chip List – Clock, Data Flip Flops, Registers, Memories, Counters

**Notes-**

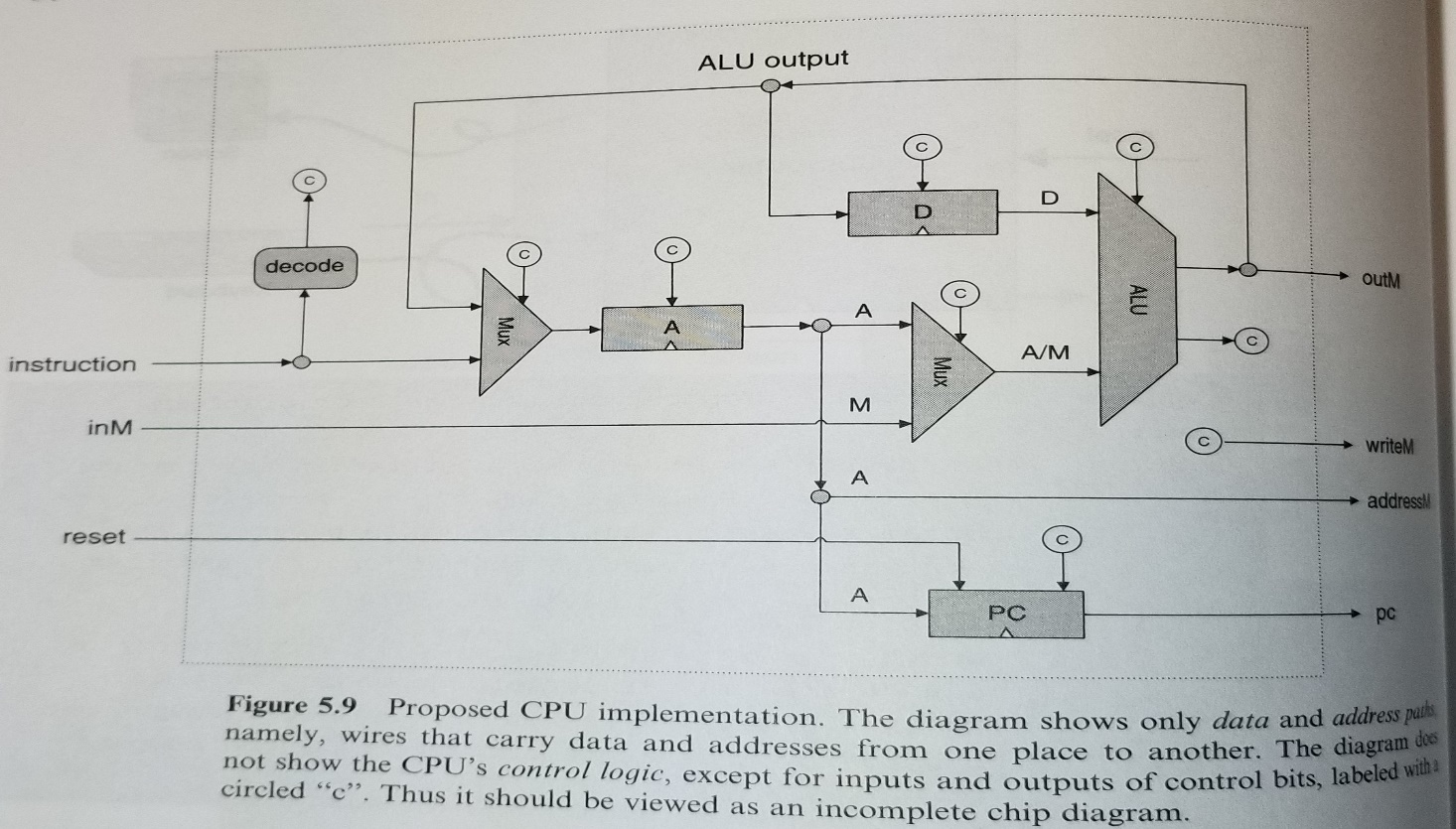
1. The Hack computer is a 16-bit machine, consisting of a CPU, two separate memory modules serving as instruction memory and data memory, and two memory-mapped I/O devices: a screen and a keyboard.

Chapter 5 Notes:

**Goal: Build the Hack computer platform culminating in the topmost computer chip. In chapter 3 we build a RAM16k chip. The Screen and Keyboard chips are built in already. We can use the ALU and register chips built in the previous chapters complete a full CPU.**

**Notes-**

1. The Hack computer is rather minimal. Typical computer platforms have more registers, more data types, more powerful ALU’s and richer instruction sets. However, from a qualitative standpoint Hack is quite similar to most digital computers, as they all follow the same conceptual paradigm: the Von Neumann architecture.



Chapter 6 Notes:

**Goal: Develop an assembler that translates programs written in Hack assembly language into the binary code understood by the Hack hardware platform.**