Computer OS and Programming Notes

Notes based on CS 2110 Textbook

Krish Katariya

Last updated: **February 17, 2024**

Contents

1. Digital Logic Structures

1.1. Sequential Logic

1.1.1. K-Maps

Note 1.1.1.1

Karnaugh maps are an easy way to make a truth table and convert it into a circuit using the least number of gates.

1.1.1.1. K-Maps Setup

Karnaugh maps are setup using gray code, which means that only one variable changes between two-adjacent cells. If you examine the values across the top from left to right, or down the side from top to bottom, you'll also see that the activated bits follow a pattern like 00, 01, etc

Definition 1.1.1.1.1

A **gray code** is a binary numerical system that is ordered such that two subsequent values only differ in one bit. It is also known as reflected binary code because the codes are reflected in the first and last n/2 values

In this case, we differ by only 1 bit at time. So for example, 01 -> 10 is NOT a gray code, because 2 bits had to be flipped

1.1.1.2. K-Maps Grouping Rules

1. We want the biggest groups where the size of the groups are a power of 2
2. We want the least number of groups
3. We can build groups with adjacent cells including wrapping around corners

If something doesn't matter we can just put it to X and we can group with it if wanted.

You make the biggest groups possible, and you analyze the groups for what values don't change and use that to create a logical expression. From there, it is easy to turn the logical expression into circuits.

1.1.2. Level Triggered Logic

There are two types of sequential logic: **level triggered logic and edge triggered logic**. Both rely on the signals of a clock, which is a circuit component that oscillates between a 1 and 0 at a set frequency to help synchronize operations in a circuit.

The difference is when the output changes based on the input signal. In **level triggered logic**, when the clock has a 1 output, the circuit output will match the input, and when the clock has 0 output, the circuit output stays the same. Think of the changes happening when the clock is 1 and level.

> RS Latches, D Latches, and memory are all level triggered

1.2. Basic Storage Elements

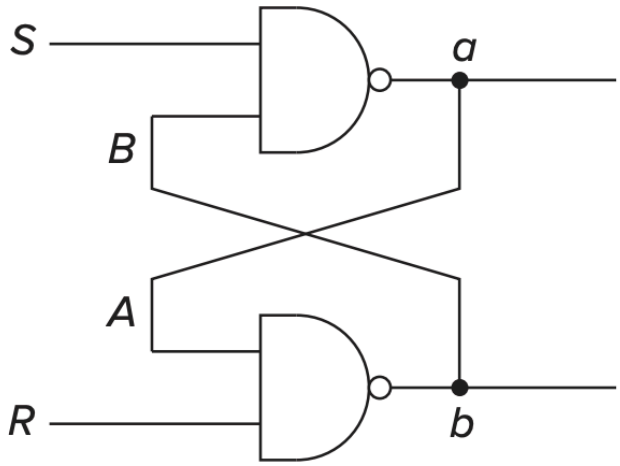
- The other kind of storage element are those that involve the storage of information and those that do not

1.2.1. The RS Latch

Definition 1.2.1.1

The **RS Latch** can store one bit of information, a 1 or a 0. Generally, two 2-input NAND gates are connected such that the output of each is connected to one of the inputs of the other. The other inputs are usually held to be zero.

Setting the latch to store a 1 is known as **setting** the latch, while setting the latch to store a 0 is referred to **resetting** the latch



Definition 1.2.1.2

The **quiescent** (or quiet) state of a latch is the state when the latch is storing a value, either 0/1, and nothing attempts to change that value.

This happens when S and R are both equal to 1. So as long as the inputs S and R remain as 1, the state of the circuit will not change.

Note 1.2.1.1

Setting the latch to a 1 or 0

The latch can be sent to 1 by momentarily setting S to 0, provided that we keep the value of R at 1. Similarly, we can set the patch to 0 by setting R to zero (known as clearing or resetting), provided we keep the value of S at 1.

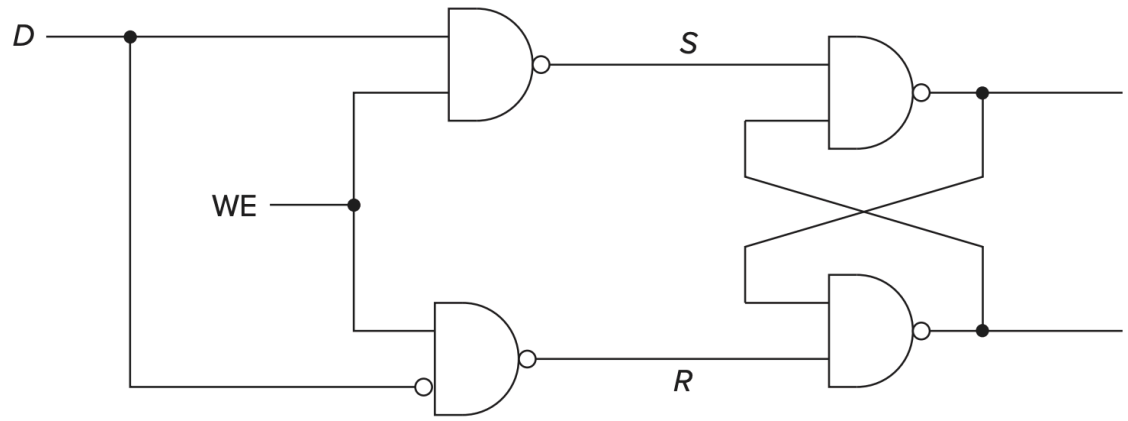
Logic behind setting to 1: If we set S to 0 for a brief period of time, this causes a and thus A to be equal to 1. Since R is 1 and A is 1, b must be 0, This causes B to be 0, which makes a equal to 1 again. Now when we return S to 1, a remains the same since B is also 0, and 1 0 input to a nand gate is enough to make sure that the NAND gate stays at 1.

1.2.2. The Gated D Latch

Definition 1.2.2.1

The D latch helps control when a latch is set and when it is cleared. In the following figure, the latch is set to the value of D whenever WE is asserted. When WE is not asserted, the outputs S and R are both equal to 1.

When WE is momentarily set to 1, exactly one of the outputs S or R is set to 0 depending on the value of D. If D is set to 1, the S is set to 0, else



re 3.19 A gated D latch.

1.3. The Concept of Memory

Memory is made up of a (usually large) number of locations, each uniquely identifiable and each having the ability to store a value. We refer to the unique memory location as its *address*. We refer to the number of bits of information stored ine ach location as its *addressability*

1.3.1. Address Space

Definition 1.3.1.1

We refer to the total number of uniquely identifiable locations as the **memory's address space**.

For example, 2GB memory has two billion memory locations.

1.3.2. Addressability

Definition 1.3.2.1

Addressability: the number of bits stored in each memory location.

1.4. 2^2 by 3-Bit Memory Example

In this case, the memory has an address space of 4 locations and an addressability of three bits. Since it is 2^2 memory, it takes two bits to specify the address. We specify it using A[1:0]. Since its addressability is 3, that means in each location, it stores 3 bits worth of information/data.

Note 1.4.1

When specifying a memory location in terms of **A[high:low]**, we are starting from the rightmost spot as index of 0. This means we are looking at the sequence of $h - l + 1$ bits such that high is the leftmost bit number, and low is the rightmost bit number in the sequence.

Access of memory first starts with *decoding the address bits*, using a decoder. We also have WE, which defines whether we are in write-enable mode of not.

The input of A[1:0] defines what the decoder has to select for the correct *word line*. From there, the decoder outputs a line of 1 which is anded across all three D-latches producing the output of that position.

1.5. State

Definition 1.5.1

State: a snapshot of that system in which all relevant items are explicitly expressed.

Ex: for a lock, the state would be open, or 0/1/2 correct operations leading to opening the lock.

Definition 1.5.2

A **finite state machine** consists of the following elements

1. A finite number of states
2. A finite number of external inputs
3. A finite number of external outputs
4. An explicit specification of all state transitions
5. An explicit specification of what determines each external output value

The state machines we have talked about so far are **asynchronous**, because there is no fixed amount of time in between when these inputs should be fed into the state machine. On the other hand, a **synchronous** state machine (such as most computers) have a fixed amount of time in between inputs.

Note 1.5.1

The control for the fixed time between state machine changes is controlled by a clock, *whos values alternate between 0 volts and some specified fixed voltage*

1.6. Datapath of LC-3

Definition 1.6.1

Datapath of the LC-3 consists of all the logic structures that combine to process information at the core of the computer.

2. The Von Neumann Model

2.1. Basic Components

A **computer program** consists of a set of instructions, each specifying a well defined piece of work for the computer to carry out. The **instruction** is the smallest piece of work specified in a computer program.

Theorem 2.1.1

The **von Neumann model** consists of 5 parts:

1. Memory
2. Processing Unit
3. Input
4. Output
5. Control Unit

2.2. Memory

A typical memory for today's computer systems is 2^{34} by 8 bits, meaning it has 2^34 distinct memory locations (its address space), with 8 bits of data (its addressability).

Definition 2.2.1

MAR: Memory Address Reader.

To read the contents of a memory location, we place the address of that location in the MAR, and interrogate the computers memory. The information stored in the location having that address is stored in the **memory data register (MDR)**.

In order to write to this location instead, we do the same process but with *write enable on*.

2.3. Process Unit

The actual processing of information in the computer is done at the **process unit**. The simplest processing unit is the **Arithmetic and Logic Unit (ALU)**, which can Add and perform basic bitwise operations.

Definition 2.3.1

ALU's normally process data elements of a fixed size known as the **word lengths** of the computer, and the data elements are known as the **words**. Most modern day processors have 64 bit word lengths, but LC-3 has a 16 bit word length.

Note 2.3.1

Most computers provide a small storage close to the ALU to allow results to be temporarily stored while doing calculations. The easiest form of quick storage is a **register** with the same length as the word length.

2.4. Input and Output

For the LC-3, our inputs are only the keyboard, and our outputs are only the monitor.

2.5. Control Unit

Definition 2.5.1

Control Unit: the component in charge of making sure all the other parts of the computer play together. The control unit keeps track of where we are within the process of executing a program.

To keep track of which instruction is being executed, the control unit has an **instruction register** which contains that information.

The control unit also has a **program counter**, which is a register pointing to the "next" instruction to be carried out.

2.6. Instruction Processing

The central idea of computer processing is that the program and data are both stored as sequences of bits in the computers memory, and the program is executed one instruction at a time under the direction of the control unit.

2.6.1. The Instruction

The most basic unit of processing is the **instruction**.

There are three types of instructions:

Definition 2.6.1.1

Operate instructions: instructions that operate on data. For LC-3, these are ADD, AND, and NOT instructions

Definition 2.6.1.2

Data movement instructions: instructions that move information from the processing unit to and from memory to input/output defives.

Definition 2.6.1.3

Control instructions: instructions that are necessary for altering the sequential processing of instructions.

2.6.2. Instruction Cycle

Instructions are carried out by the control unit in a systematic, step by step manner. There are six sequential phases, each of which requires zero or more steps.

```
FETCH
DECODE
EVALUATE ADDRESS
FETCH OPERANDS
EXECUTE
STORE RESULT
```

Definition 2.6.2.1

Fetch phase: obtains the next instruction from memory and loads it into the instruction register (IR) of the control unit.

The instructions are stored in computer memory. In order to carry out the work of an instruction, we must identify where it is. The PC contains the address of the next instruction to be processed.

Steps:

1. The MAR is loaded with the contents of the PC, and increment the PC
2. The memory is interrogated, which results in the next instruction being placed by the memory into the MDR
3. The IR is loaded with the contents of the MDR

Definition 2.6.2.2

Decode phase: examines the instructions in order to figure out what the micro architecture is being used to do. In the LC-3, a decoder figures out which of the 16 opcodes to be used.

Definition 2.6.2.3

Execute phase: carries out the execution of the Instruction.

There are other phases but we don't really need them for this class (?)

3. The LC-3

3.1. Memory Organization

The LC-3 memory has an address space of 2^{16} locations and an addressability of 16 bits. For the LC-3, we refer to 16 bits *as one word*, and we say that the LC-3 is *word addressable*

Note 3.1.1

Reminder (because I forgot lol):

addressability: the amount of bits stored for each locations **address space**: the amount of unique storage locations

3.2. Registers

Definition 3.2.1

Registers: used to store data temporarily because if often takes more than one clock cycle to access memory/do other tasks.

The LC-3 has 8 unique registers, each identifiable by a three-digit register number.

3.3. The Instruction Set

Instructions are made of two things, their **opcode** (what the instruction is asking the computer to do) and its **operands** (who the computer is expected to do it to)

Definition 3.3.1

Instruction Set: defined by its opcodes, datatypes, and addressing modes.

Example 3.3.1

The INSTRUCTION [ADD R2, R0 R1] has an opcode of ADD, one addressing mode (register mode}, and one data type (2's complement), In this case we define two registers from which to add to and a register to store the value in.

Other instructions: AND, BR, JMP, JSR, LD, STI, etc

Theres too many maybe ill desscribe them if its required.

3.4. Opcodes

The LC-3 ISA has 15 instructions, each defined by its unique opcodes, meaning 4 bits are used for the opcode.

Note 3.4.1

The LC3 only has 15 opcodes, even when there are 16 possibilities. The code 1101 has been left unspecified.

There are three different kinds of opcodes:

1. **Operates**: process information
2. **Data movement**: move information between memory and the registers and between them and I/O
3. **Control**: change the sequence of instructions that will be executed

3.5. Data Types

Every opcode will interpret the bit patterns of its operands according to the data type it is designed to support. For ADD opcode, this is 2's complement.

3.6. Addressing Modes

Definition 3.6.1

Addressing Modes: a mechanism is a mechanism for specifying where the operand is located.

An operand can generally be found in one of three places:

1. In memory
2. In a register
3. As a part of the instruction
   - If part of the instruction, we refer to it as a *literal* or as an *immediate operand.*

The LC-3 supports five addressing modes, immediate, register, and three memory addressing modes (PC-relative, indirect, and Base+offset)

Definition 3.6.2

PC relative operands are calculated relative to the Program Counter value, For example `LD R0, 50(PC)` means to load the content of memory at an address calculated by adding 50 to the PC into register R0

Definition 3.6.3

Indirect Addressing operands involve accessing memory indirectly through a pointer.

Definition 3.6.4

Base+offset Addressing operands are calculated by adding a base value (usually in a register) to some offset. Useful for accessing elements of arrays/structures in memory

3.7. Condition Codes

The LC-3 has three single bit registers that are individually set each time one of the 8 general purpose registers (GPR's) is written into as a result of execution of one of the operate instructions/load instructions.

Definition 3.7.1

The three single bit registers are the **N, Z, and P** registers corresponding to negative, zero, and positive. These three are referred to as **condition codes** because the condition of thehose bits are used to change the sequence of execution of instructions in a program.