Introduction To Memory (ROM)

Read-Only Memery

Akilesh Praveen — CMSC398E

UMD

March 6, 2020

Agenda

1. Announcements
   - Class Cancelled

2. Introduction to Memory
   - Introduction

Announcements

Class cancelled next week!

- We won’t be having class next week :)
- Go enjoy your spring break!!

Extension Policy

- It's fine to ask for extensions, but please do so reasonably and **beforehand**.
- We're already pretty lenient with grading in this class, but we will draw a line somewhere.
- Note: If you have a medical note or a university excusal, this policy can be overriden.

Read Only Memory

What is Read Only Memory?

- It's a pretty simple type of memory to understand, so we'll start off with it
- Memory that you can write **once**, but you can only read from after
- When you power off the machine, the memory you wrote will still remain the way you set it

Why Read Only Memory?

- ROM has a lot of uses in modern electronics
    - Things like BIOS in computers + other startup functions
    - Calculators for startup routines + repeated values
    - Put to heavy use in gaming consoles
    - Things like digital clocks and hair dryers also will have a fair bit of this stuff if you take them apart

Why Read Only Memory?

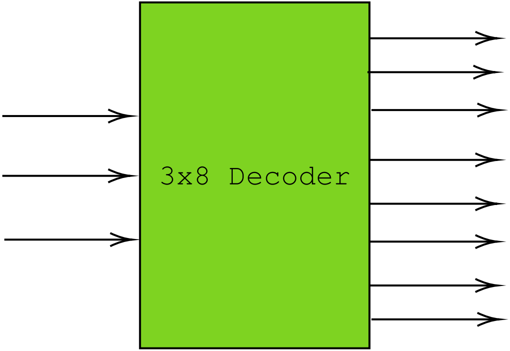
- Incidentally, this is also the easiest memory to build
- We get the concept- and it turns out, there are easy ways to represent ROM as a set of functions

Some types of ROM

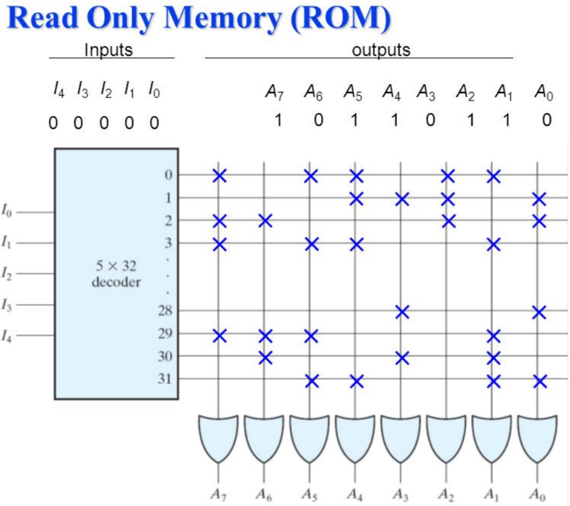
- **ROM** → Read Only Memory
    - Data assigned during the manufacturing process
- **PROM** → Programmable Read Only Memory
    - Programmed after manufacture
- **EPROM** → Erasable Programmable Read Only Memory
    - Same as above, but can be erased (UV)
- **EEPROM** → Electrically Erasable Programmable Read Only Memory
    - Can be erased electrically, unlike above

Let's make ROM

- Remember decoders?
- Turns out, ROM can be thought of as a basic decoder, but with custom outputs



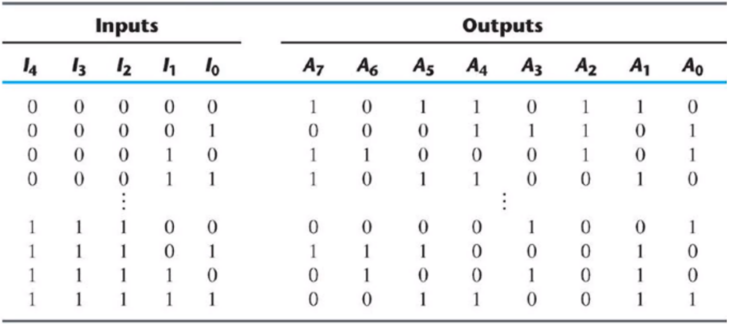
ROM Example



Read Only Memory (ROM)

Inputs					outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0

ROM Example



Inputs						Outputs							
I_4	I_3	I_2	I_1	I_0		A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0		1	0	1	1	0	1	1	0
0	0	0	0	1		0	0	0	1	1	1	0	1
0	0	0	1	0		1	1	0	0	0	1	0	1
0	0	0	1	1		1	0	1	1	0	0	1	0
		⋮							⋮				
1	1	1	0	0		0	0	0	0	1	0	0	1
1	1	1	0	1		1	1	1	0	0	0	1	0
1	1	1	1	0		0	1	0	0	1	0	1	0
1	1	1	1	1		0	0	1	1	0	0	1	1

Disadvantages of ROM

- Real Life
  - Can never be changed
  - Only realistic to manufacture in huge batches, and takes a lot of R&D to get right
  - No software patches
- Minecraft
  - Can't be changed with outside influence (as easily)
  - In that sense, lines can never be repurposed without manual reconfiguration

Other types of ROM

- **PROM**, **EPROM**, **EEPROM**
- *"Finally, a PROM that all computer science/engineering majors can enjoy"*
  -Aki, probably
- Essentially, these are just QOL improvements upon ROM
- EPROM and EEPROM are the same in functionality, it's just that one is erased via ultraviolet light and electrical signals

PROM Advantages

- PROMs are highly versatile, and it turns out that they are highly useful to implement minimum functions
- e.g. $\neg$ AB+AB can be minimized to B
- We can explore this further using Karnaugh Maps
    - Karnaugh Maps and minimization will be covered in an online video (this is a 1 credit class)

Grading / Open OH / Project 5