Basics of Computing – Chapter 1.1-1.3
Data Manipulation and Storage Media

Cory L. Strope
Boolean Symbol System

Circuits

Data Storage
0's and 1's

Binary **number system** vs. Boolean **symbol system**

Binary:
- Operations: $+\,,\times\,,\,/,\,-$
- $\{0, 1, 10, 11, 100, 101, \ldots\}$

Boolean:
- Operations: AND, OR, XOR, NOT, \ldots
- $\{0, 1\}$ or $\{\text{true, false}\}$ or $\{\text{on, off}\}$
Boolean Logic
Binary Nature

Boolean logic is binary in nature, thus there are two values for:

▶ Any given condition
▶ Any answer based on the condition

For example, is the weather pleasant?

▶ Three conditions: Wind, warmth, and rain.
▶ These values can be set to either true (1), or false (0).

Today: warm = false, wind = false, rain = true.
Weather pleasant? **False**

Allowing degrees of warmth, wind, and rain would require levels of each condition.
Boolean Logic

Gate

The circuitry to carry out boolean logic are called logical gates.

- Designed to accept a small number of conditions
- Conditions expressed as true or false

Consider logic gates with only two inputs. For a given gate, there will be particular response if:

1. Both inputs are false,
2. Only the first input is false,
3. Only the first input is true, and
4. Both inputs are true.

To show the output of a gate for each pair of inputs, we build truth tables.
Truth Tables

Truth tables compute functional values of logical expressions w.r.t. all possible combinations their logical variables may take.

Column headings on a truth table show:

1. Inputs (variables)
2. Expression result, based on variables and operators

Row headings show:

1. All possible T/F assignment to variables
2. Result of T/F assignment to Expression

<table>
<thead>
<tr>
<th>Inputs:</th>
<th>Outputs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Q</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Boolean Logic
Operations and Circuitry

- **Input**: 2 bits
- **Number of operations for 2-bit input operations**: $2^4 = 16$
- **Given the following setup for a truth table**:

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P op Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$X_1$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$X_2$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$X_3$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$X_4$</td>
</tr>
</tbody>
</table>

where result $X_i \in \{0, 1\}$.

16 combinations possible using the given setup:

<table>
<thead>
<tr>
<th>$X_1X_2X_3X_4$</th>
<th>Description</th>
<th>Gate Name?</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Inputs ignored</td>
<td>$False$</td>
</tr>
<tr>
<td>0001</td>
<td>$P$ and $Q$ true</td>
<td>$AND$</td>
</tr>
<tr>
<td>0010</td>
<td>Only $Q$ true</td>
<td>$Q \text{ AND NOT } P$</td>
</tr>
<tr>
<td>0011</td>
<td>$Q$ true</td>
<td>$Q$</td>
</tr>
<tr>
<td>0100</td>
<td>Only $P$ true</td>
<td>$P \text{ AND NOT } Q$</td>
</tr>
<tr>
<td>0101</td>
<td>$P$ true</td>
<td>$P$</td>
</tr>
<tr>
<td>0110</td>
<td>Only $P$ or $Q$ true</td>
<td>$XOR$</td>
</tr>
<tr>
<td>0111</td>
<td>$P$ or $Q$ true</td>
<td>$OR$</td>
</tr>
<tr>
<td>1000</td>
<td>$P$ and $Q$ false</td>
<td>$NOR$</td>
</tr>
<tr>
<td>1001</td>
<td>$P = Q$</td>
<td>$NXOR$</td>
</tr>
<tr>
<td>1010</td>
<td>$P$ false</td>
<td>$NOT P$</td>
</tr>
<tr>
<td>1011</td>
<td>Not only $P$ true</td>
<td>$P$ implies $Q$</td>
</tr>
<tr>
<td>1100</td>
<td>$Q$ false</td>
<td>$NOT Q$</td>
</tr>
<tr>
<td>1101</td>
<td>Not only $Q$ true</td>
<td>$Q$ implies $P$</td>
</tr>
<tr>
<td>1110</td>
<td>$P$ or $Q$ false</td>
<td>$NAND$</td>
</tr>
<tr>
<td>1111</td>
<td>Inputs ignored</td>
<td>$True$</td>
</tr>
</tbody>
</table>
Boolean Logic
Operations

Out of the 16 possible gates, we will study only four:

- Three basic gates:
  1. NOT
  2. AND
  3. OR

- Combinations of these gates construct our other common gate:
  1. XOR
Boolean Operations

NOT – \((\NOT P, \NOT Q)\)

The simplest boolean operation is **NOT**.

- Takes 1 input and inverts it.

\[
\begin{array}{c|c|c|c}
P & Q & \NOT P & \NOT Q \\
\hline
0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 \\
\end{array}
\]
Boolean Gates

NOT

\[ \begin{array}{c|c|c}
\text{Inputs} & 0 & 1 \\
\text{Output} & 1 & 0 \\
\end{array} \]

\[ ^1 \text{From Pearson Education, Inc.} \]
Boolean Operations

AND

Boolean AND works similar to binary multiplication.

- Takes 2 inputs, P and Q.

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P AND Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Boolean Gates
AND

\[ \text{AND} \]

\[
\begin{array}{c|c|c}
\text{Inputs} & & \text{Output} \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

\(^2\text{From Pearson Education, Inc.}\)
Boolean Operations

OR

Boolean OR works similar to binary addition.

- Takes 2 inputs, P and Q.

\[
\begin{array}{c|c}
0 & 0 \\
\hline
0 & 1 \\
\hline
1 & 0 \\
\hline
1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c}
0 & 0 \\
\hline
1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
P & Q & P \text{ OR } Q \\
\hline
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
Boolean Gates

**OR**

Inputs | Output
--- | ---
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 1

---

From Pearson Education, Inc.
**Boolean Operations**

**XOR**

Boolean XOR = \[ \begin{cases} & \text{output 0} \\ & \text{input 1's even} \\ & \text{output 1} \\ & \text{input 1's odd} \end{cases} \]

- Takes 2 inputs, P and Q.

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P XOR Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Boolean Gates

XOR

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
</tr>
<tr>
<td>1 1</td>
<td>0</td>
</tr>
</tbody>
</table>
Boolean Symbol System

Circuits

Data Storage

Cory L. Strope: Basics of Computing – Chapter 1.1-1.3 Data Manipulation and Storage Media
Creating a Circuit

As it was previously mentioned, the XOR gate can be produced by the three basic gates

- AND
- OR
- NOT

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P XOR Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Creating a Circuit

Steps

1. Highlight the rows in which a 1 shows up in the output.

2. For each row, use AND and NOT gates to make the inputs equal 1.
   - \( r_{01} \) \((\text{NOT } P \text{ AND } Q)\)
   - \( r_{10} \) \((P \text{ AND } \text{NOT} Q)\)

3. If there is more than 1 row highlighted, connect gates with OR
   - \((\text{NOT } P \text{ AND } Q) \text{ OR } (P \text{ AND } \text{NOT} Q)\)

4. Draw circuit (on board)
Creating a Circuit

XOR using AND’s and NOT’s
Boolean Addition

Two-bit Adder

Addition can be performed using the general equation:

\[
\begin{array}{c}
P \\
+ \quad Q \\
\hline
C_S \\
\end{array}
\]

P and Q are the inputs, C is the **carry** bit, and S is the **sum** bit. To add two bits, we need to match all four cases of addition:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The sum in all 4 cases is two bits, therefore, two outputs. This implies that there needs to be at least two logic gates.
Boolean Addition

To create a circuit with more than one output, create a circuit for each output.

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
## Boolean Addition

### Two-bit Adder

Highlight the rows with 1’s in the output. *Note that each output (S and C) is treated independently of the other!*

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
### Boolean Addition

**Two-bit Adder**

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>C</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Boolean Addition

Three-bit Adder

The two-bit adder is restricted to adding bit-strings of length 1 only.

Multi-column addition is preferable. For this, we need an adder that can propagate carry bits:
Boolean Addition

Three-bit Adder

The Logic table for a 3-bit Adder:

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>C_{in}</th>
<th>C_{out}</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
### Boolean Addition

#### Three-bit Adder

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>Cin</th>
<th>Cout</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- **Cout**: Occurs when either P & Q, P & Cin, or Q & Cin are 1.
- **S**: True when an odd number of inputs are 1.
- Bullet point above: 3 &’s, 1 or!
- Suggests we need only to compute the sum, as in the two-bit adder
  - 3-bit XOR: (P XOR Q) XOR Cin
Boolean Addition

Three-bit Adder

Cory L. Strope: Basics of Computing – Chapter 1.1-1.3 Data Manipulation and Storage Media
Flip-Flops

In previous digital computations, the inputs were given to a gate, and stage-by-stage, the output of each gate is found by a lookup in it’s logic table, until the end of the computation is reached, and the output is determined.

More difficult (though important) computations require feedback from outputs into earlier inputs. One example of such a gate is the flip-flop.

Definition
A flip-flop is a circuit that produces an output value of 0 or 1, which remains constant until a temporary pulse from another circuit causes it to shift to the other value.
Flip-Flops
Like a Light Switch

Diagram of flip-flops circuit.
Flip-Flops

Why?

Once set, flip-flops will maintain it’s current state (output value). Other circuits can change the state of the flip-flop by sending a pulse (1-bit) to the inputs of the flip-flop.

- Maintenance of the output state ← storing a bit within a computer.
- To store information, many flip-flops can be constructed on a computer chip.
Storage Devices in Computer Systems

Issues

Storing “stuff” brings up many issues, depending on what it is:

- How much storage do we need?
- How do we get our stuff?
- How fast can we get our stuff?
- How much are we willing to spend on our storage materials?
- How do we organize our stored stuff?
- How long will the items last?

We can answer these questions for many things:

- Food,
- papers (like old homework),
- etc.
Storage Devices in Computer Systems

Bits and Bytes

- How much storage do we need? **Size**
- How do we get our stuff? **Access Method**
- How fast can we get our stuff? **Access Speed**
- How much are we willing to spend on our storage materials? **Cost**
- How do we organize our stored stuff? **Organization and Structure**
- How long will the items last? **Life-time of stored elements**.

It is important to understand the impact of each of these questions when working with computers.
We want to find the phone number of someone fast
Assume phone book is not in alphabetical order
Sequentially search entry-by-entry.

Very slow. Sequential.

http://www.lynnlake.ca
Storage Devices in Computer Systems
Paper Clipped Numbers

▶ We keep the business cards of numbers we use frequently in our wallet.
  ▶ Look at the cards on a regular basis.
  ▶ Faster and easier than the phonebook.
  ▶ Wallet is small.
  ▶ Still flip through the cards.

Faster than phone book, but still slow. Sequential or Random?

^http://www.sfc-webdesign.com
Storage Devices in Computer Systems
Number that is Found, taken from paper clip.

Making calls, take a couple of cards out.
  Find correct number on business card (Fax, office, home, ...)
  Business card is very small.
  Easier to find number.
  Limitations?

Fast.
Random

---

Mrs. Barbara Gantwarg
Math Tutor

“I Solve Math Problems”
Certified Teacher of Mathematics

http://www.toadwebsites.com
Storage Devices in Computer Systems

Written Number

867-5309

- Found the number, write it down
  - Very fast.
  - Cannot write more than one number.
  - After phone call, what happens to number?

Very Fast.
Random.
Comparisons of Storage Options

<table>
<thead>
<tr>
<th></th>
<th>Speed</th>
<th>Size</th>
<th>Volatility</th>
<th>Access Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phone Book</td>
<td>Very Slow</td>
<td>Very Large</td>
<td>Very Low</td>
<td>Sequential</td>
</tr>
<tr>
<td>Paper Clip</td>
<td>Slow</td>
<td>Large</td>
<td>Low</td>
<td>Random</td>
</tr>
<tr>
<td>Biz. Card</td>
<td>Fast</td>
<td>Small</td>
<td>High</td>
<td>Random</td>
</tr>
<tr>
<td>Whiteboard</td>
<td>Very Fast</td>
<td>Very Small</td>
<td>Very high</td>
<td>Random</td>
</tr>
</tbody>
</table>
Storage Devices in Computer Systems
From Slowest to Fastest

- Mass Storage: Hard Drive, CD, DVD, Tape backup
- Main Memory
- Cache
- Registers
Storage Devices in Computer Systems

Main Memory

Memory is stored in units called cells.

- Each cell is 1 byte.
- Each cell has a “name”, or address.
  - Similar to house address
- Main Memory is usually called *Random Access Memory* (RAM)
  - We can access any cell directly, without scanning other cells.

<table>
<thead>
<tr>
<th>Cell</th>
<th>01011011</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell 1</td>
<td>01001111</td>
</tr>
<tr>
<td>Cell 2</td>
<td>01100111</td>
</tr>
<tr>
<td>Cell 3</td>
<td>00000010</td>
</tr>
<tr>
<td>Cell 4</td>
<td>01101011</td>
</tr>
<tr>
<td>Cell 5</td>
<td>10001101</td>
</tr>
<tr>
<td>Cell 6</td>
<td>00110010</td>
</tr>
<tr>
<td>Cell 7</td>
<td>10111011</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Cell n</td>
<td>11011011</td>
</tr>
</tbody>
</table>
There are two operations in memory:

- Read
- Write

Read(Cell 5) → 10001101

Write(00000000, Cell 5) replaces the contents of Cell 5 with 00000000.

<table>
<thead>
<tr>
<th>Cell</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>01011011</td>
</tr>
<tr>
<td>1</td>
<td>01001111</td>
</tr>
<tr>
<td>2</td>
<td>01100111</td>
</tr>
<tr>
<td>3</td>
<td>00000010</td>
</tr>
<tr>
<td>4</td>
<td>01101011</td>
</tr>
<tr>
<td>5</td>
<td>10001101</td>
</tr>
<tr>
<td>6</td>
<td>00110010</td>
</tr>
<tr>
<td>7</td>
<td>10111011</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>11011011</td>
</tr>
</tbody>
</table>
Main memory plays a vital role in any computer

- All read and write operations go through the Main memory
- Main memory is faster than the Hard drive
Storage Devices in Computer Systems
Main Memory – Cons

Main memory, cache, and registers all work using electrical pulses. As in flip-flops, before any pulses there is no state

- Without electricity, a flip-flop loses its state.
- Shutting down a computer resets the state of memory, cache, and registers!
Storage Devices in Computer Systems
Mass Storage

- Magnetic Systems
  - Disk
  - Tape
- Optical Systems
  - CD
  - DVD
Storage Devices in Computer Systems
Mass Storage

Pros:

- Can hold huge amounts of data
- Do not lose data when the power is off

Cons:

- Very slow
- Some mass storage devices are *sequential*.
  - You cannot directly access the data you need. For example, tape drives.
## Storage Devices in Computer Systems

### Mass Storage vs. Main Memory

<table>
<thead>
<tr>
<th></th>
<th>Speed</th>
<th>Size</th>
<th>Volatility</th>
<th>Access Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass Storage</td>
<td>Slow</td>
<td>Huge</td>
<td>Very Low</td>
<td>Sequential or Random</td>
</tr>
<tr>
<td>Main Memory</td>
<td>Fast</td>
<td>Small</td>
<td>High</td>
<td>Random</td>
</tr>
</tbody>
</table>
Mass Storage
On-line vs. Off-line

Definition
On-line means the device or information is connected and readily available to the machine without human intervention.

Definition
Off-line means that human intervention is required before the device or information can be accessed by the machine.
Mass Storage
Magnetic Disk Drives

A good example of a magnetic disk drive is the floppy disk drive:
Mass Storage
Hard Disk Drives – Time

Disk Speed = \begin{cases} 
\text{Access Time} = \begin{cases} 
\text{Seek Time} \\
\text{Latency Time} \\
\text{Transfer Time} 
\end{cases} \\
\text{Seek time:} \text{ The time required to move the read/write heads from one track to another.} \\
\text{Rotation (Latency) time:} \text{ Half the time required for the disk to make a complete rotation.} \\
\text{Transfer time:} \text{ The rate at which data can be transferred to or from the disk.} 
\end{cases} \)}}
Example: What is the average access time of a hard disk that spins at 75 revolutions per second\(^5\) with a seek time of 15 milliseconds?

1. Calculate the rotation time: (one half of the rotation time)

\[
0.5 \times \frac{1 \text{ second}}{75 \text{ revolutions}} = \frac{1}{150}
\]

2. Add rotation time and seek time:

\[
0.00667 + 0.015 = 0.02167 \text{ s} = 21.67 \text{ ms}
\]

\(^5\)75 revolutions per second is the same as 4500 revolutions per minute. 4500 RPM is the slowest hard drive speed for commonly sold disks.
Mass Storage
Tape drive

To access data from a tape drive, the tape is mounted in a device called a tape drive that can read, write, and rewind the tape.

Pros: Low cost, High capacity.
Cons: Extremely slow, so often done overnight.
Magnetic tapes are still used as a popular off-line data storage for archival purposes.
Optical Storage
Compact Disk (CD)

CDs store information on a single track that spirals around the CD, from inside to out.
Questions over Chapters 1.1–1.3?