BBM 231

Hours: Mondays: 13:00 – 16:00

Instructors:

• Ufuk Çelikcan: ufuk.celikcan@gmail.com
• Mehmet Köseoğlu: mkoseoglu@gmail.com

Register the course on Piazza:
piazza.com/hacettepe.edu.tr/fall2020/bbm231233
BBM 233 Lab

• We will announce the details later.
• You will be using online circuit simulators and verilog software to do the lab.
• Assistant: Selma Dilek.
Grading

• 1 midterm exam: 35%
• Final exam: 45%
• Quizzes:
  – 10 Quizzes: 20%
• We will take attendance through quizzes.
• A student will get an F1 if s/he does not attend enough quizzes.
Quizzes

• Will be given through Google Classroom.
• We will invite you through your cs.hacettepe.edu.tr emails.
• You need to open a Google account with your cs.hacettepe.edu.tr email.
• Quizzes will be given every week and will also count towards your attendance.
Textbook & References

• Textbook
  Digital Design and Computer Architecture 2nd edition, Harris and Harris

• Other references
  – Tons of digital design books
  – Lectures from MIT Open Courseware and Stanford
Chapter 1 :: Topics

- Background
- The Game Plan
- The Art of Managing Complexity
- The Digital Abstraction
- Number Systems
- Logic Gates
- Logic Levels
- CMOS Transistors
- Power Consumption
Background

- Microprocessors have revolutionized our world
  - Cell phones, Internet, rapid advances in medicine, etc.
- The semiconductor industry has grown from $21 billion in 1985 to $300 billion in 2011
The Game Plan

• Purpose of course:
  – Understand what’s under the hood of a computer
  – Learn the principles of digital design
  – Learn to systematically debug increasingly complex designs
  – Design and build a microprocessor
The Art of Managing Complexity

- Abstraction
- Discipline
- The Three –Y’s
  - Hierarchy
  - Modularity
  - Regularity
Abstraction

- Hiding details when they aren’t important

<table>
<thead>
<tr>
<th>Application Software</th>
<th>“hello world!”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Systems</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td></td>
</tr>
<tr>
<td>Micro-architecture</td>
<td></td>
</tr>
<tr>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td>Digital Circuits</td>
<td></td>
</tr>
<tr>
<td>Analog Circuits</td>
<td></td>
</tr>
<tr>
<td>Devices</td>
<td></td>
</tr>
<tr>
<td>Physics</td>
<td></td>
</tr>
</tbody>
</table>

- programs
- device drivers
- instructions
- registers
- datapaths
- controllers
- adders
- memories
- AND gates
- NOT gates
- amplifiers
- filters
- transistors
- diodes
- electrons
• Intentionally restrict design choices
• Example: Digital discipline
  – Discrete voltages instead of continuous
  – Simpler to design than analog circuits – can build more sophisticated systems
  – Digital systems replacing analog predecessors:
    • i.e., digital cameras, digital television, cell phones, CDs
The Three -Y’s

- **Hierarchy**
  - A system divided into modules and submodules

- **Modularity**
  - Having well-defined functions and interfaces

- **Regularity**
  - Encouraging uniformity, so modules can be easily reused
• **Hierarchy**
  
  – Three main modules: lock, stock, and barrel
  
  – **Submodules of lock:** hammer, flint, frizzen, etc.

---

**Example: The Flintlock Rifle**

- **Lock**: hammer, flint, frizzen, etc.
- **Stock**: supports the firearm.
- **Barrel**: where the bullet travels before being fired.

---

Example: The Flintlock Rifle

- **Modularity**
  - **Function of stock**: mount barrel and lock
  - **Interface of stock**: length and location of mounting pins

- **Regularity**
  - Interchangeable parts
The Digital Abstraction

• Most physical variables are **continuous**
  – Voltage on a wire
  – Frequency of an oscillation
  – Position of a mass

• Digital abstraction considers **discrete subset** of values
The Analytical Engine

- Designed by Charles Babbage from 1834 – 1871
- Considered to be the first digital computer
- Built from mechanical gears, where each gear represented a discrete value (0-9)
- Babbage died before it was finished
Digital Discipline: Binary Values

• Two discrete values:
  – 1’s and 0’s
  – 1, TRUE, HIGH
  – 0, FALSE, LOW

• 1 and 0: voltage levels, rotating gears, fluid levels, etc.

• Digital circuits use **voltage** levels to represent 1 and 0

• **Bit**: Binary digit
George Boole, 1815-1864

• Born to working class parents
• Taught himself mathematics and joined the faculty of Queen’s College in Ireland.
• Wrote *An Investigation of the Laws of Thought* (1854)
• Introduced binary variables
• Introduced the three fundamental logic operations: AND, OR, and NOT.
Number Systems

- Decimal numbers
  \[5374_{10} = \]

- Binary numbers
  \[1101_2 = \]
Number Systems

• **Decimal numbers**

\[ 5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0 \]

- five thousands
- three hundreds
- seven tens
- four ones

• **Binary numbers**

\[ 1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10} \]

- one eight
- one four
- no two
- one one
Powers of Two

- $2^0 =$
- $2^1 =$
- $2^2 =$
- $2^3 =$
- $2^4 =$
- $2^5 =$
- $2^6 =$
- $2^7 =$
- $2^8 =$
- $2^9 =$
- $2^{10} =$
- $2^{11} =$
- $2^{12} =$
- $2^{13} =$
- $2^{14} =$
- $2^{15} =$
### Powers of Two

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$
- Handy to memorize up to $2^9$
• Decimal to binary conversion:
  – Convert $10011_2$ to decimal

• Decimal to binary conversion:
  – Convert $47_{10}$ to binary
Number Conversion

• Decimal to binary conversion:
  – Convert $10011_2$ to decimal
  – $16 \times 1 + 8 \times 0 + 4 \times 0 + 2 \times 1 + 1 \times 1 = 19_{10}$

• Decimal to binary conversion:
  – Convert $47_{10}$ to binary
  – $32 \times 1 + 16 \times 0 + 8 \times 1 + 4 \times 1 + 2 \times 1 + 1 \times 1 = 101111_2$
Binary Values and Range

- \( N \)-digit decimal number
  - How many values?
  - Range?
  - Example: 3-digit decimal number:

- \( N \)-bit binary number
  - How many values?
  - Range:
  - Example: 3-digit binary number:
Binary Values and Range

• $N$-digit decimal number
  – How many values? $10^N$
  – Range? $[0, 10^N - 1]$
  – Example: 3-digit decimal number:
    • $10^3 = 1000$ possible values
    • Range: $[0, 999]$

• $N$-bit binary number
  – How many values? $2^N$
  – Range: $[0, 2^N - 1]$
  – Example: 3-digit binary number:
    • $2^3 = 8$ possible values
    • Range: $[0, 7] = [000_2$ to $111_2]$
# Hexadecimal Numbers

<table>
<thead>
<tr>
<th>Hex Digit</th>
<th>Decimal Equivalent</th>
<th>Binary Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>
# Hexadecimal Numbers

<table>
<thead>
<tr>
<th>Hex Digit</th>
<th>Decimal Equivalent</th>
<th>Binary Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0001</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0010</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0011</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>0100</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>0101</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>0110</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>A</td>
<td>10</td>
<td>1010</td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>1011</td>
</tr>
<tr>
<td>C</td>
<td>12</td>
<td>1100</td>
</tr>
<tr>
<td>D</td>
<td>13</td>
<td>1101</td>
</tr>
<tr>
<td>E</td>
<td>14</td>
<td>1110</td>
</tr>
<tr>
<td>F</td>
<td>15</td>
<td>1111</td>
</tr>
</tbody>
</table>
Hexadecimal Numbers

- Base 16
- Shorthand for binary
Hexadecimal to Binary

• Hexadecimal to binary conversion:
  – Convert $4AF_{16}$ (also written 0x4AF) to binary

• Hexadecimal to decimal conversion:
  – Convert 0x4AF to decimal
Hexadecimal to Binary

- **Hexadecimal to binary conversion:**
  - Convert \(4AF_{16}\) (also written \(0x4AF\)) to binary
  - \(0100\ 1010\ 1111_2\)

- **Hexadecimal to decimal conversion:**
  - Convert \(4AF_{16}\) to decimal
  - \(16^2 \times 4 + 16^1 \times 10 + 16^0 \times 15 = 1199_{10}\)
• Bits

\[ 10010110 \]

most significant bit
least significant bit

byte

• Bytes & Nibbles

\[ 10010110 \]
nibble

• Bytes

CEBF9AD7

most significant byte
least significant byte
Large Powers of Two

- $2^{10} = 1 \text{ kilo} \approx 1000 \ (1024)$
- $2^{20} = 1 \text{ mega} \approx 1 \text{ million} \ (1,048,576)$
- $2^{30} = 1 \text{ giga} \approx 1 \text{ billion} \ (1,073,741,824)$
Estimating Powers of Two

• What is the value of $2^{24}$?

• How many values can a 32-bit variable represent?
Estimating Powers of Two

• What is the value of $2^{24}$?
  - $2^4 \times 2^{20} \approx 16$ million

• How many values can a 32-bit variable represent?
  - $2^2 \times 2^{30} \approx 4$ billion
Addition

- Decimal
  \[
  \begin{array}{c}
  3734 \\
  + 5168 \\
  \hline
  8902
  \end{array}
  \text{carries}
  \]

- Binary
  \[
  \begin{array}{c}
  1011 \\
  + 0011 \\
  \hline
  1110
  \end{array}
  \text{carries}
  \]
Binary Addition Examples

- Add the following 4-bit binary numbers
  
  \[
  \begin{array}{c}
  \text{1001} \\
  + \text{0101} \\
  \hline
  \text{1010}
  \end{array}
  \]

- Add the following 4-bit binary numbers
  
  \[
  \begin{array}{c}
  \text{1011} \\
  + \text{0110} \\
  \hline
  \text{1101}
  \end{array}
  \]
Binary Addition Examples

- Add the following 4-bit binary numbers
  
  
  \[
  \begin{array}{c}
  1001 \\
  + 0101 \\
  \hline
  1110
  \end{array}
  \]

- Add the following 4-bit binary numbers
  
  \[
  \begin{array}{c}
  111 \\
  + 1011 \\
  + 0110 \\
  \hline
  10001
  \end{array}
  \]

Overflow!
Overflow

- Digital systems operate on a fixed number of bits
- Overflow: when result is too big to fit in the available number of bits
- See previous example of $11 + 6$
Signed Binary Numbers

- Sign/Magnitude Numbers
- Two’s Complement Numbers
• 1 sign bit, $N$-1 magnitude bits
• Sign bit is the most significant (left-most) bit
  – Positive number: sign bit = 0
  – Negative number: sign bit = 1

$$ A = (-1)^a_{n-1} \sum_{i=0}^{n-2} a_i 2^i $$

• Example, 4-bit sign/mag representations of ±6:
  +6 =
  - 6 =

• Range of an $N$-bit sign/magnitude number:
**Sign/Magnitude Numbers**

- 1 sign bit, \( N-1 \) magnitude bits
- Sign bit is the most significant (left-most) bit
  - Positive number: sign bit = 0
  - Negative number: sign bit = 1
- Example, 4-bit sign/mag representations of \( \pm 6 \):
  - \(+6 = 0110\)
  - \(-6 = 1110\)
- Range of an \( N \)-bit sign/magnitude number:
  \([-2^{N-1}-1, 2^{N-1}-1]\)
• Problems:
  – Addition doesn’t work, for example \(-6 + 6\): 
    \[
    \begin{array}{c}
    1110 \\
    + 0110 \\
    \hline
    10100 \text{ (wrong!)}
    \end{array}
    \]
  – Two representations of 0 (± 0):
    \[
    \begin{array}{c}
    1000 \\
    0000
    \end{array}
    \]
Two’s Complement Numbers

• Don’t have same problems as sign/magnitude numbers:
  – Addition works
  – Single representation for 0
Two’s Complement Numbers

- Msb has value of $-2^{N-1}$

$$A = a_{n-1} \left(-2^{n-1}\right) + \sum_{i=0}^{n-2} a_i 2^i$$

- Most positive 4-bit number:
- Most negative 4-bit number:
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an $N$-bit two’s comp number:
Two’s Complement Numbers

- Msb has value of $-2^{N-1}$

$$A = a_{n-1} (-2^{n-1}) + \sum_{i=0}^{n-2} a_i 2^i$$

- Most positive 4-bit number: 0111
- Most negative 4-bit number: 1000
- The most significant bit still indicates the sign (1 = negative, 0 = positive)
- Range of an $N$-bit two’s comp number: $[-(2^{N-1}), 2^{N-1}-1]$
• Flip the sign of a two’s complement number
• Method:
  1. Invert the bits
  2. Add 1
• Example: Flip the sign of $3_{10} = 0011_2$
Flip the sign of a two’s complement number

Method:
1. Invert the bits
2. Add 1

Example: Flip the sign of $3_{10} = 0011_2$

1. 1100
2. $+ 1$

$1101 = -3_{10}$
Two’s Complement Examples

- Take the two’s complement of $6_{10} = 0110_2$

- What is the decimal value of $1001_2$?
Two’s Complement Examples

• Take the two’s complement of $6_{10} = 0110_2$
  
  1. $1001$
  2. $+ 1$
  
  $$1010_2 = -6_{10}$$

• What is the decimal value of the two’s complement number $1001_2$?
  
  1. $0110$
  2. $+ 1$
  
  $$0111_2 = 7_{10}, \text{ so } 1001_2 = -7_{10}$$
Two’s Complement Addition

- Add $6 + (-6)$ using two’s complement numbers
  
  $0110$
  
  $+ 1010$
  
  $1100$

- Add $-2 + 3$ using two’s complement numbers
  
  $1110$
  
  $+ 0011$
  
  $0001$
Two’s Complement Addition

- Add $6 + (-6)$ using two’s complement numbers
  \[
  \begin{array}{c}
  111 \\
  0110 \\
  + 1010 \\
  \hline
  10000
  \end{array}
  \]

- Add $-2 + 3$ using two’s complement numbers
  \[
  \begin{array}{c}
  111 \\
  1110 \\
  + 0011 \\
  \hline
  10001
  \end{array}
  \]
Increasing Bit Width

• Extend number from $N$ to $M$ bits ($M > N$):
  – Sign-extension
  – Zero-extension
Sign-Extension

- Sign bit copied to msb’s
- Number value is same

- Example 1:
  - 4-bit representation of 3 = 0011
  - 8-bit sign-extended value: 00000011

- Example 2:
  - 4-bit representation of -5 = 1011
  - 8-bit sign-extended value: 11111011
Zero-Extension

- Zeros copied to msb’s
- Value changes for negative numbers

- **Example 1:**
  - 4-bit value = 0011₂ = 3₁₀
  - 8-bit zero-extended value: 00000011 = 3₁₀

- **Example 2:**
  - 4-bit value = 1011 = -5₁₀
  - 8-bit zero-extended value: 00001011 = 11₁₀
### Number System Comparison

<table>
<thead>
<tr>
<th>Number System</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unsigned</td>
<td>[0, (2^N-1)]</td>
</tr>
<tr>
<td>Sign/Magnitude</td>
<td>([- (2^{N-1}-1), 2^{N-1}-1])</td>
</tr>
<tr>
<td>Two’s Complement</td>
<td>([-2^{N-1}, 2^{N-1}-1])</td>
</tr>
</tbody>
</table>

For example, 4-bit representation:

- **Unsigned**
  - \(0000\) to \(1111\)
  - \(0000\) to \(1111\)

- **Sign/Magnitude**
  - \(0000\) to \(1111\)
  - \(0000\) to \(1111\)

- **Two’s Complement**
  - \(1000\) to \(1111\)
  - \(1000\) to \(1111\)

- **Sign/Magnitude**
  - \(1111\) to \(1111\)
  - \(1000\) to \(1111\)