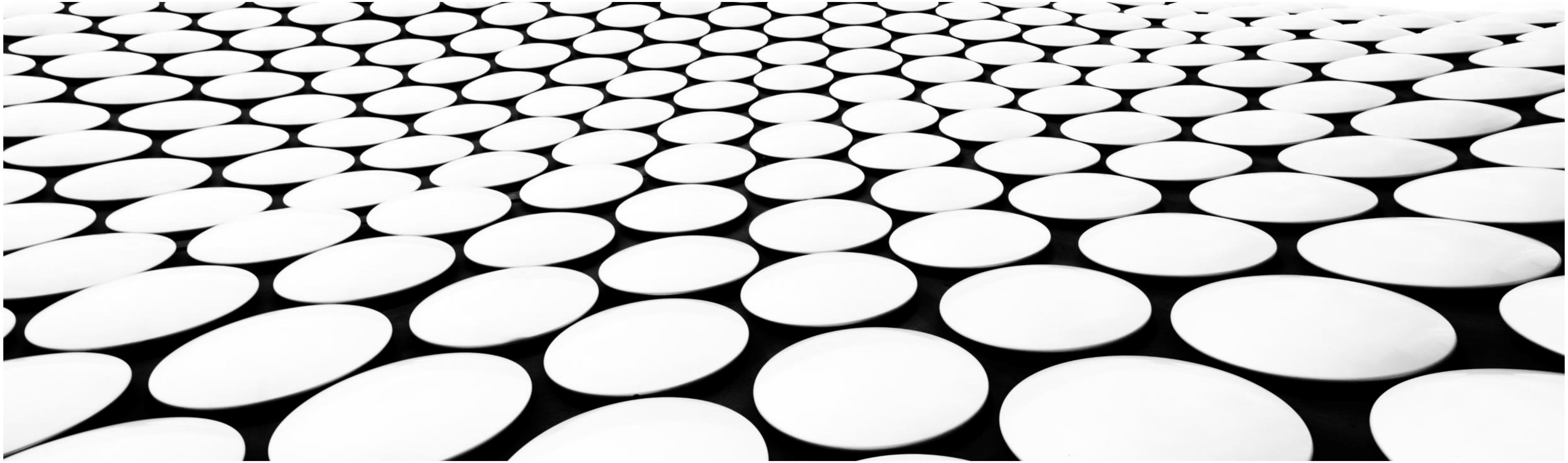
CS5102: FOUNDATIONS OF COMPUTER SYSTEMS

LECTURE 2: **BASICS OF DIGITAL LOGIC**

DR. ARIJIT ROY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY PATNA



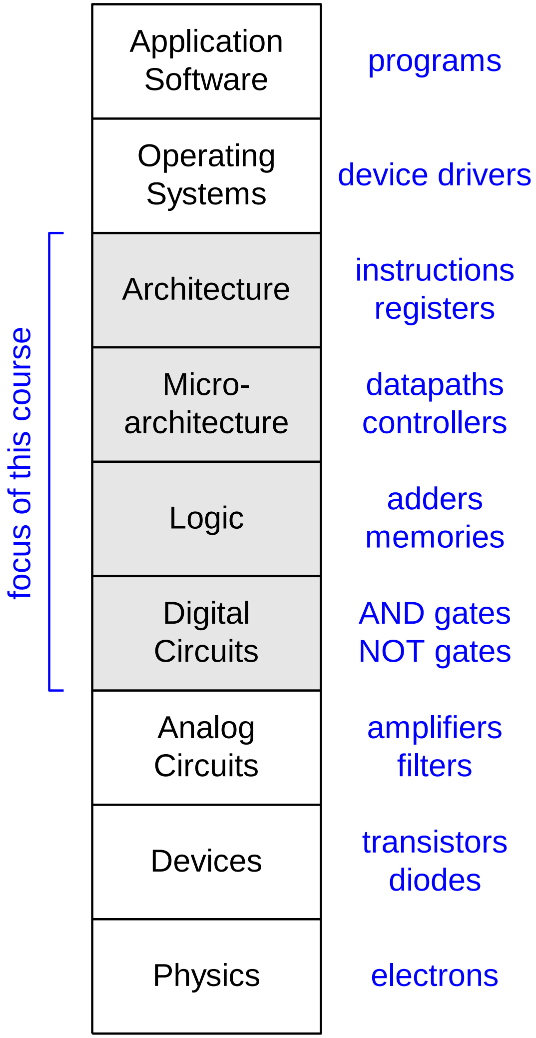
These slides are based on the book: David Money Harris and Sarah L. Harris, Computer Organization and Design

THE ART OF MANAGING COMPLEXITY

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

ABSTRACTION

- Hiding details when they aren’t important



DISCIPLINE

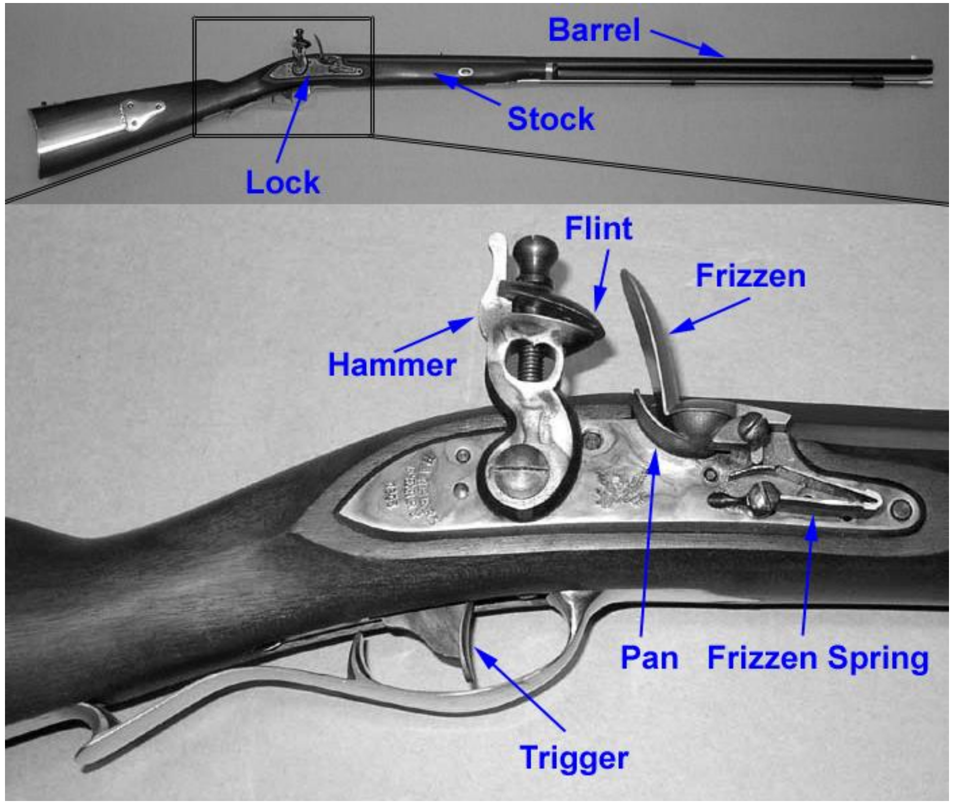
- Intentionally restricting your design choices
  - to work more productively at a higher level of abstraction
- Example: Digital discipline
  - Considering discrete voltages instead of continuous voltages used by analog circuits
  - Digital circuits are simpler to design than analog circuits – can build 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

EXAMPLE: FLINTLOCK RIFLE

- **Hierarchy**
  - Three main modules: lock, stock, and barrel
  - Submodules of lock: hammer, flint, frizzen, etc.
- **Modularity**
  - Function of stock: mount barrel and lock
  - Interface of stock: length and location of mounting pins
- **Regularity**
  - Interchangeable parts

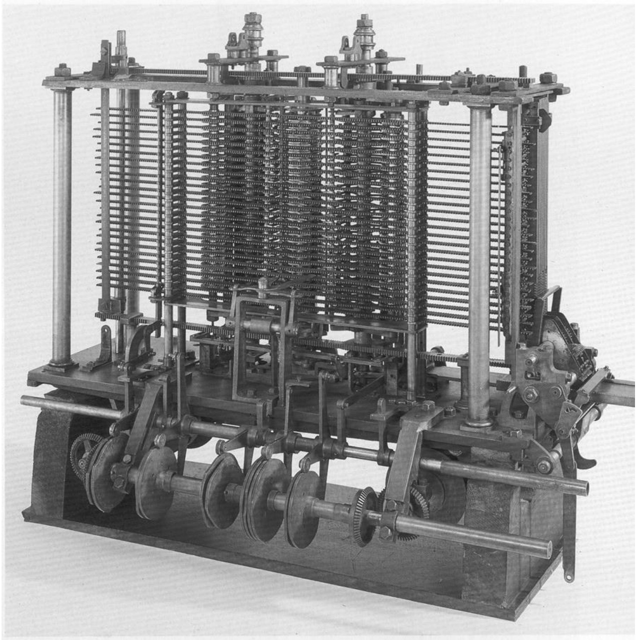


THE DIGITAL ABSTRACTION

- Most physical variables are continuous, for example
  - Voltage on a wire
  - Frequency of an oscillation
  - Position of a mass
- Instead of considering all values, the digital abstraction considers only a 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

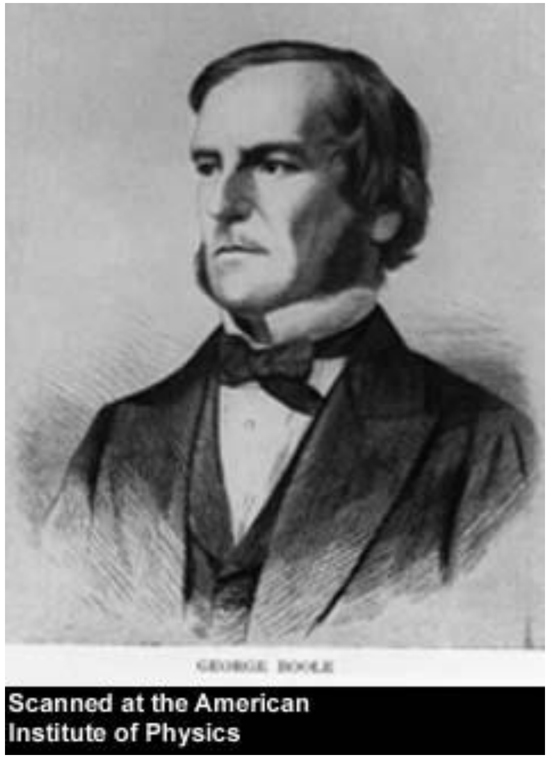


DIGITAL DISCIPLINE: BINARY VALUES

- Typically consider only two discrete values:
  - 1's and 0's
  - 1, TRUE, HIGH
  - 0, FALSE, LOW
- 1 and 0 can be represented by specific voltage levels, rotating gears, fluid levels, etc.
- Digital circuits usually depend on specific voltage levels to represent 1 and 0
- *Bit*: *B*inary dig*it*

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

1000's column
100's column
10's column
1's column

$$5374_{10} = \underbrace{5 \times 10^3}_{\text{five thousands}} + \underbrace{3 \times 10^2}_{\text{three hundreds}} + \underbrace{7 \times 10^1}_{\text{seven tens}} + \underbrace{4 \times 10^0}_{\text{four ones}}$$

- Binary numbers

8's column
4's column
2's column
1's column

$$1101_2 = \underbrace{1 \times 2^3}_{\text{one eight}} + \underbrace{1 \times 2^2}_{\text{one four}} + \underbrace{0 \times 2^1}_{\text{no two}} + \underbrace{1 \times 2^0}_{\text{one one}} = 13_{10}$$

POWERS OF TWO

- $2^0 =$
- $2^1 =$
- $2^2 =$
- $2^3 =$
- $2^4 =$
- $2^5 =$
- $2^6 =$
- $2^7 =$
- $2^8 =$
- $2^9 =$
- $2^{10} =$
- $2^{11} =$
- $2^{12} =$
- $2^{13} =$
- $2^{14} =$
- $2^{15} =$
- Handy to memorize up to 2^9

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

NUMBER CONVERSION

- Decimal to binary conversion:
  - Convert 10101_2 to decimal

- Decimal to binary conversion:
  - Convert 47_{10} to binary

NUMBER CONVERSION

- Decimal to binary conversion:
  - Convert 10011_2 to decimal
  - $16\times1 + 8\times0 + 4\times0 + 2\times1 + 1\times1 = 19_{10}$

- Decimal to binary conversion:
  - Convert 47_{10} to binary
  - $32\times1 + 16\times0 + 8\times1 + 4\times1 + 2\times1 + 1\times1 = 101111_2$

BINARY VALUES AND RANGE

- N-digit decimal number
  - How many values? $\mathbf{10^N}$
  - Range? $\mathbf{[0, 10^N - 1]}$
  - Example: 3-digit decimal number:
    - $10^3 = 1000$ possible values
    - Range: [0, 999]
- N-bit binary number
  - How many values? $\mathbf{2^N}$
  - Range: $\mathbf{[0, 2^N - 1]}$
  - Example: 3-digit binary number:
    - $2^3 = 8$ possible values
    - Range: $[0, 7] = [000_2 \text{ to } 111_2]$

HEXADECIMAL NUMBERS

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
A	10	
B	11	
C	12	
D	13	
E	14	
F	15	

HEXADECIMAL NUMBERS

Hex Digit	Decimal Equivalent	Binary Equivalent
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

HEXADECIMAL NUMBERS

- Base 16
- Shorthand to write long binary numbers

HEXADECIMAL TO BINARY CONVERSION

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

- Hexadecimal to decimal conversion:
  - Convert $4AF_{16}$ to decimal

HEXADECIMAL TO BINARY CONVERSION

- 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, BYTES, NIBBLES…

- Bits

10010110

most significant bit

least significant bit

- Bytes & Nibbles

byte

10010110

nibble

- Bytes

CEBF9AD7

most significant byte

least significant byte

POWERS OF TWO

- 2^{10} = 1 kilo ≈ 1000 (1024)
- 2^{20} = 1 mega ≈ 1 million (1,048,576)
- 2^{30} = 1 giga ≈ 1 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

$$\begin{array}{r} \color{blue}{11} \quad \leftarrow \color{blue}{\text{carries}} \\ 3734 \\ +\ 5168 \\ \hline 8902 \end{array}$$

- Binary

$$\begin{array}{r} \color{blue}{11} \quad \leftarrow \color{blue}{\text{carries}} \\ 1011 \\ +\ 0011 \\ \hline 1110 \end{array}$$

BINARY ADDITION EXAMPLES

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1001 \\ +\ 0101 \\ \hline \end{array}$$

- Add the following 4-bit binary numbers

$$\begin{array}{r} 1011 \\ +\ 0110 \\ \hline \end{array}$$

BINARY ADDITION EXAMPLES

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
1 \\
1001 \\
+\;\; 0101 \\
\hline
1110
\end{array}
$$

- Add the following 4-bit binary numbers

$$
\begin{array}{r}
111 \\
1011 \\
+\;\; 0110 \\
\hline
10001
\end{array}
$$

Overflow!

OVERFLOW

- Digital systems operate on a fixed number of bits
- Addition overflows when the 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

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

$$A: \{a_{N-1}, a_{N-2}, \cdots a_2, a_1, a_0\}$$

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

- 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

$$A:\{a_{N-1}, a_{N-2}, \cdots a_2, a_1, a_0\}$$

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

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

  +6 = **0110**

  - 6 = **1110**

- Range of an *N*-bit sign/magnitude number: $\mathbf{[-(2^{N-1}-1), 2^{N-1}-1]}$

SIGN/MAGNITUDE NUMBERS

- Problems:
  - Addition doesn't work, for example -6 + 6:

$$\begin{array}{r} 1110 \\ +\ 0110 \\ \hline 10100 \end{array}$$ (wrong!)

  - Two representations of 0 (± 0):

1000

0000

TWO’S COMPLEMENT NUMBERS

- Don’t have same problems as sign/magnitude numbers:
  - Addition works
  - Single representation for 0

TWO'S COMPLEMENT NUMBERS

- Same as unsigned binary, but the most significant bit (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

- Same as unsigned binary, but the most significant bit (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: **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: **$[-(2^{N-1}), 2^{N-1}-1]$**

"TAKING THE TWO'S COMPLEMENT"

- Flip the sign of a two's complement number
- Method:
  - Invert the bits
  - Add 1
- Example: Flip the sign of $3_{10} = 0011_2$

"TAKING THE TWO'S COMPLEMENT"

- 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. $\underline{+ \quad 1}$

     $1010_2 = -6_{10}$

- What is the decimal value of the two's complement number 1001_2?
  1. 0110
  2. $+ \quad 1$

     $0111_2 = 7_{10}$, so $1001_2 = -7_{10}$

TWO'S COMPLEMENT ADDITION

- Add 6 + (-6) using two's complement numbers

$$\begin{array}{r} 0110 \\ +\ 1010 \\ \hline \end{array}$$

- Add -2 + 3 using two's complement numbers

$$\begin{array}{r} 1110 \\ +\ 0011 \\ \hline \end{array}$$

TWO'S COMPLEMENT ADDITION

- Add 6 + (-6) using two's complement numbers

```
   111
    0110
+   1010
--------
   10000
```

- Add -2 + 3 using two's complement numbers

```
   111
    1110
+   0011
--------
   10001
```

INCREASING BIT WIDTH

- A value can be extended from N bits to M bits (where $M > N$) by using:
  - Sign-extension
  - Zero-extension

SIGN-EXTENSION

- Sign bit is copied into most significant bits.
- Number value remains the 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 are copied into most significant bits.
- Value will change for negative numbers.

- **Example 1:**
  - 4-bit value = $0011_2 = 3_{10}$
  - 8-bit zero-extended value: $00000011 = 3_{10}$
- **Example 2:**
  - 4-bit value = $1011 = -5_{10}$
  - 8-bit zero-extended value: $00001011 = 11_{10}$

LOGIC GATES

- Perform logic functions:
  - inversion (NOT), AND, OR, NAND, NOR, etc.
- Single-input:
  - NOT gate, buffer
- Two-input:
  - AND, OR, XOR, NAND, NOR, XNOR
- Multiple-input

THANK YOU!