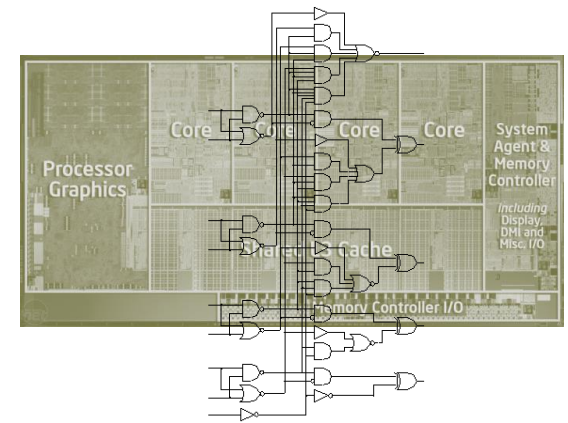




고려대학교
KOREA UNIVERSITY



COSE221, COMP211 Logic Design

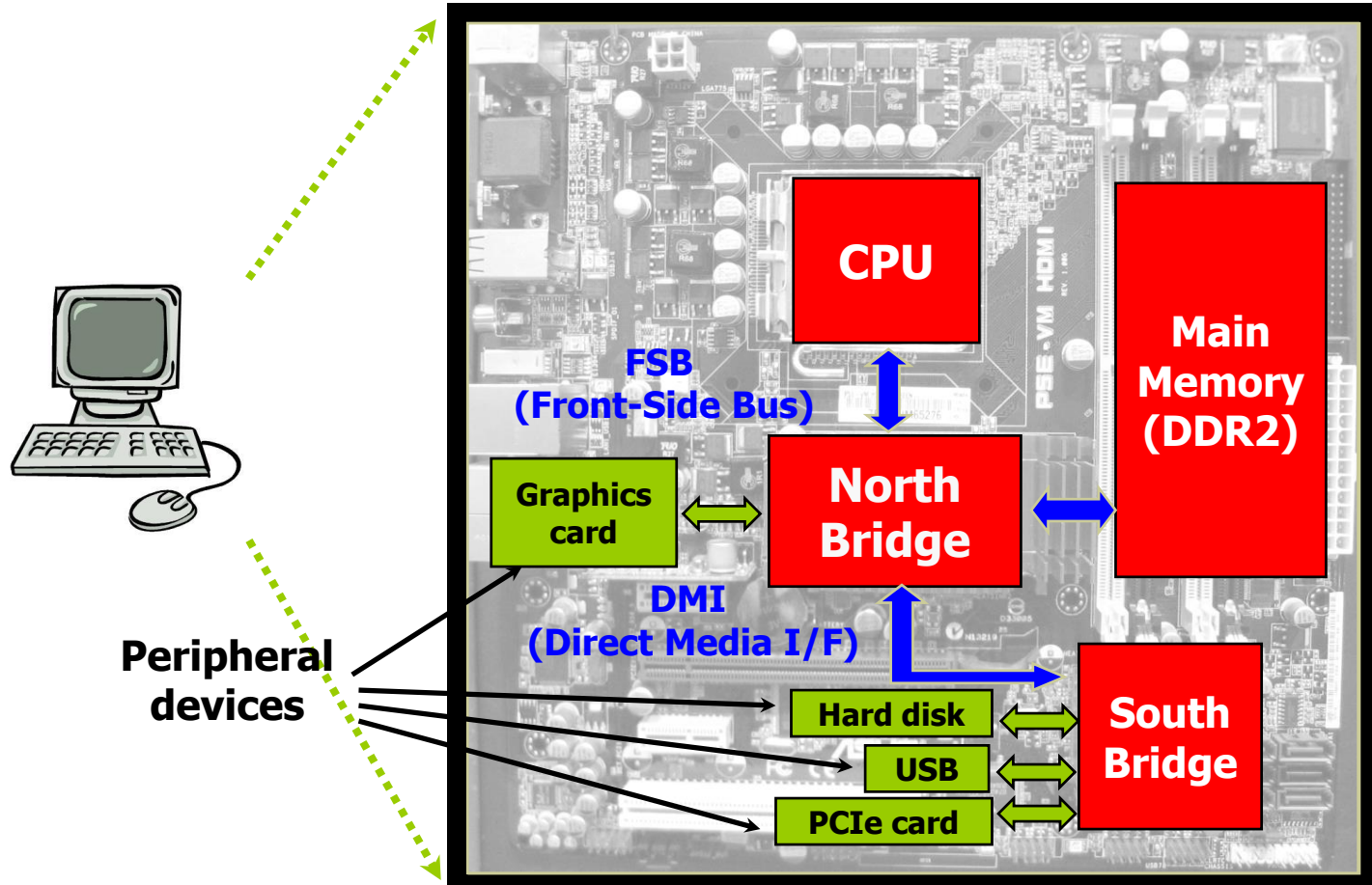
Lecture 1. Number Systems

Prof. Taeweon Suh

Computer Science & Engineering

Korea University

A Desktop Computer System (as of 2008)



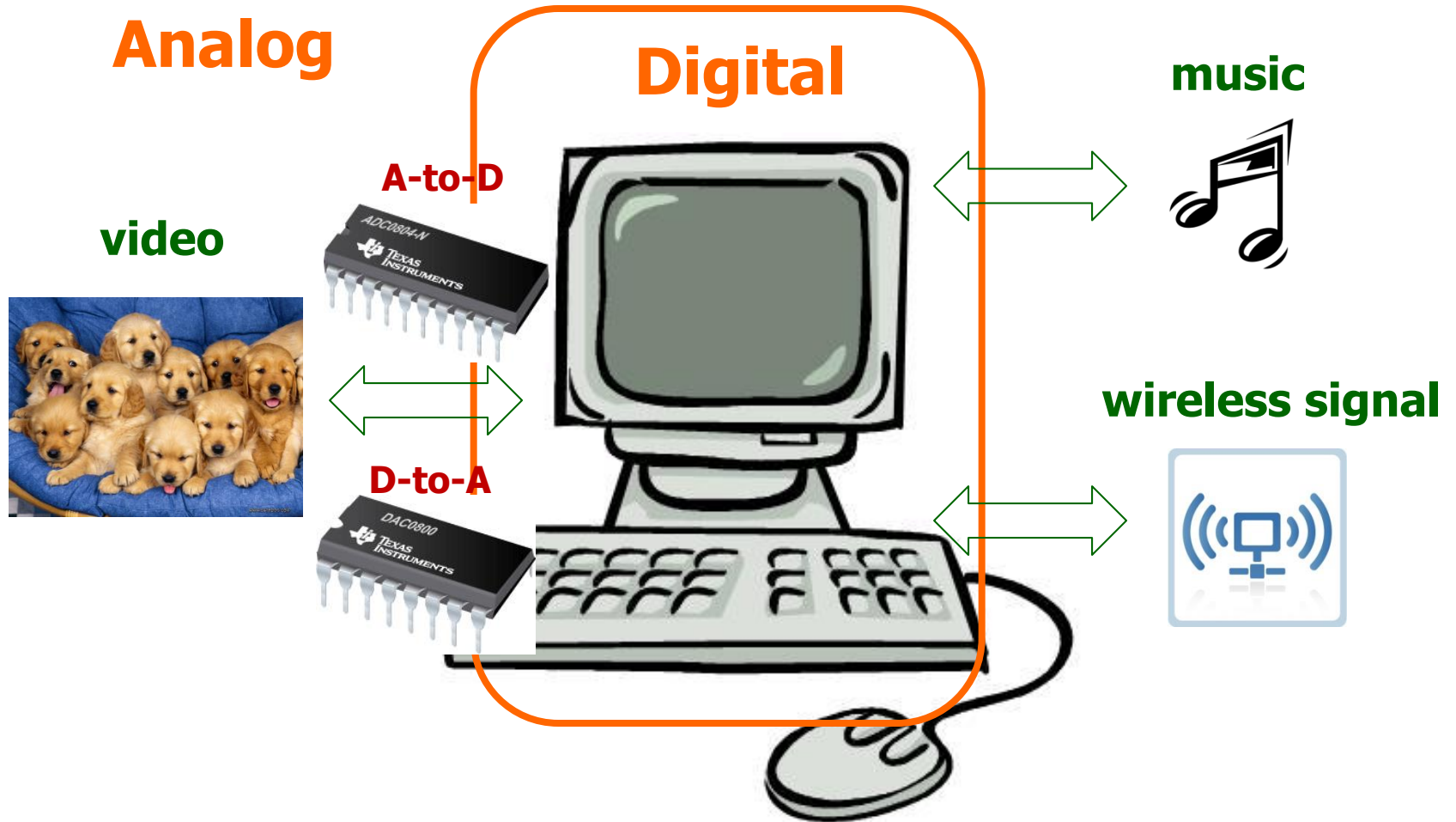
- The term “Processor” is used to refer to CPU

A Computer System

- Computer is composed of many components
 - CPU (Intel's Core 2 Duo, AMD's Opteron etc)
 - Memory (DDR3)
 - Chipsets (North Bridge and South Bridge)
 - Power Supply
 - Peripheral devices such as Graphics card and Wireless card
 - Monitor
 - Keyboard/mouse
 - etc

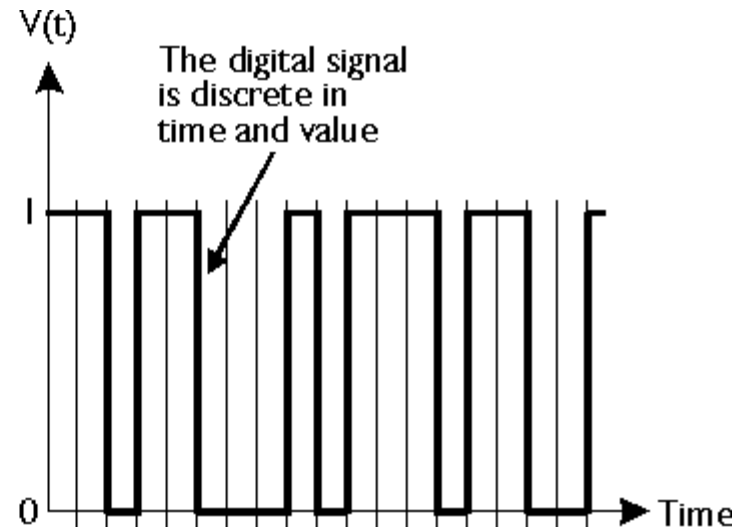
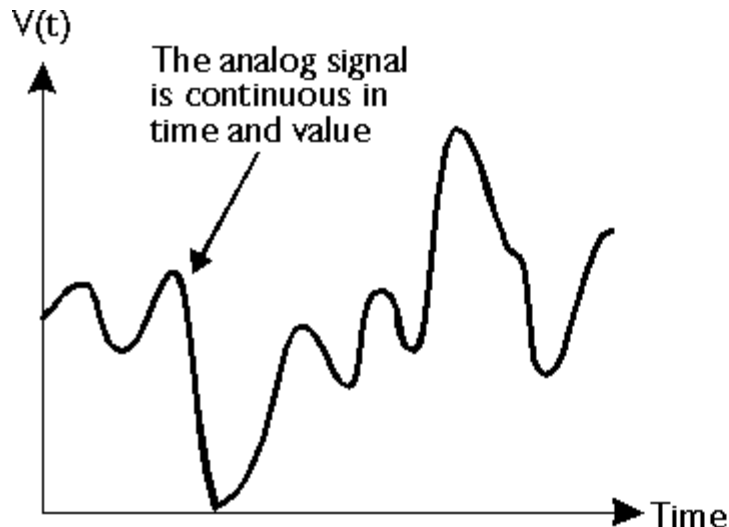


Digital vs Analog



Bottom layer of a Computer

- Each component inside a computer is basically made based on **analog** and **digital** circuits
 - Analog
 - Continuous signal
 - Digital
 - Only knows 1 and 0



What you mean by 0 or 1 in Digital Circuit?

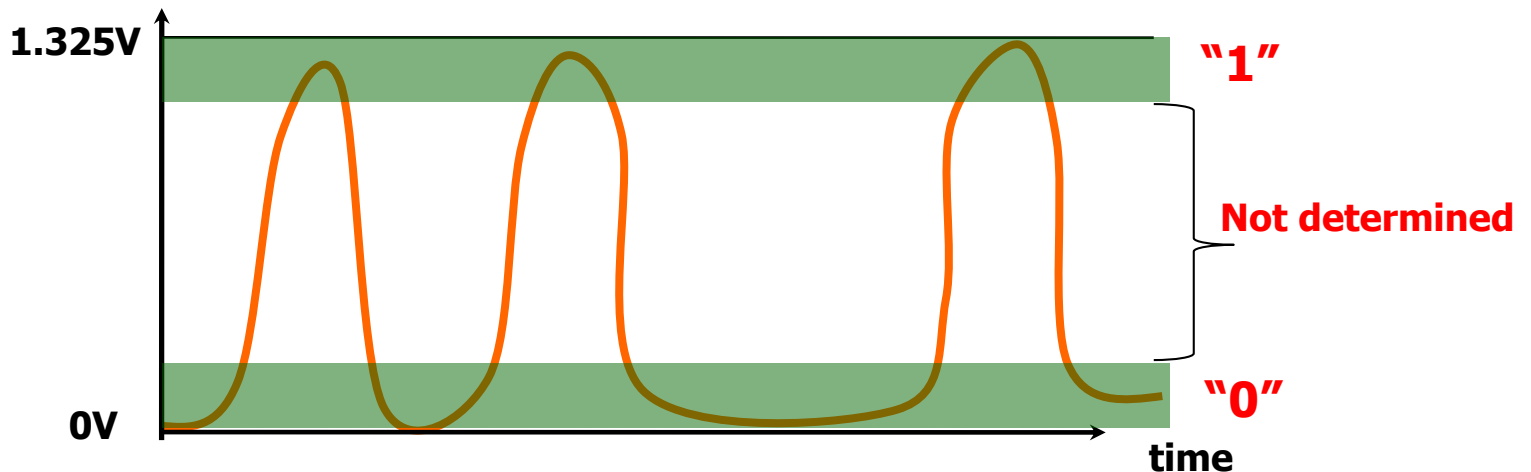
- In fact, everything in this world is analog
 - For example, sound, light, electric signals are all analog since they are continuous in time
 - Digital circuit is a special case of analog circuit
- Power supply provides power to the computer system
 - Power supply has several outlets (such as 3.3V, 5V, and 12V)



What you mean by 0 or 1 in Digital Circuit?

- Digital circuit treats a signal above a certain level as "1" and a signal below a certain level as "0"
 - Different components in a computer have different voltage requirements
 - CPU (Core 2 Duo): 1.325 V
 - Chipsets: 1.45 V
 - Peripheral devices: 3.3V, 1.5V

Note: Voltage requirements change as the technology advances



Number Systems

- Analog information (video, sound etc) is converted to a digital format for processing
- Computer processes information in digital
- Since digital knows "1" and "0", we use different number systems in computer
 - **Binary** and **Hexadecimal** numbers

Number Systems - Decimal

- **Decimal** numbers

- Most natural to human because we have ten fingers (?) and/or because we are used to it (?)
- Each column of a decimal number has 10x the weight of the previous column

- **Decimal number has 10 as its base**

ex) $5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$

- N-digit number represents one of 10^N possibilities
ex) 3-digit number represents one of 1000 possibilities: 0 ~ 999

Number Systems - Binary

- **Binary** numbers

- Bit represents one of 2 values: 0 or 1
- Each column of a binary number has 2x the weight of the previous column

- **Binary number has 2 as its base**

ex) $10110_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 22_{10}$

- N-bit binary number represents one of 2^N possibilities
ex) 3-bit binary number represents one of 8 possibilities: 0 ~ 7

Power of 2

- $2^0 =$

- $2^1 =$

- $2^2 =$

- $2^3 =$

- $2^4 =$

- $2^5 =$

- $2^6 =$

- $2^7 =$

- $2^8 =$

- $2^9 =$

- $2^{10} =$

- $2^{11} =$

- $2^{12} =$

- $2^{13} =$

- $2^{14} =$

- $2^{15} =$

Power of 2

- $2^0 = 1$
- $2^1 = 2$
- $2^2 = 4$
- $2^3 = 8$
- $2^4 = 16$
- $2^5 = 32$
- $2^6 = 64$
- $2^7 = 128$
- $2^8 = 256$
- $2^9 = 512$
- $2^{10} = 1024$
- $2^{11} = 2048$
- $2^{12} = 4096$
- $2^{13} = 8192$
- $2^{14} = 16384$
- $2^{15} = 32768$

*** Handy to memorize up to 2^9**

Number Systems - Hexadecimal

- **Hexadecimal** numbers

- Writing long binary numbers is tedious and error-prone
- **We group 4 bits to form a hexadecimal (hex)**
 - A hex represents one of 16 values
 - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F
- Each column of a hex number has 16x the weight of the previous column
 - **Hexadecimal number has 16 as its base**
ex) $2ED_{16} = 2 \times 16^2 + E(14) \times 16^1 + D(13) \times 16^0 = 749_{10}$
- N-hexadigit number represents one of 16^N possibilities
ex) 2-hexadigit number represents one of 16^2 possibilities: 0 ~ 255

Number Systems

Hex Number	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

Number Conversions

- Hexadecimal to binary conversion:
 - Convert $4AF_{16}$ (also written $0x4AF$) to binary number
 - $0100\ 1010\ 1111_2$
- Hexadecimal to decimal conversion:
 - Convert $0x4AF$ to decimal number
 - $4 \times 16^2 + A(10) \times 16^1 + F(15) \times 16^0 = 1199_{10}$

Bits, Bytes, Nibbles

- Bits (b)

10010110

most significant bit least significant bit

- Bytes & Nibbles

- Byte (B) = 8 bits
 - Used everyday
- Nibble (N) = 4 bits
 - Not commonly used

byte

10010110

nibble

CEBF9AD7

most significant byte least significant byte

KB, MB, GB ...

- In computer, the basic unit is byte (B)
- And, we use KB, MB, GB many many many times
 - $2^{10} = 1024 = \mathbf{1KB \text{ (kilobyte)}}$
 - $2^{20} = 1024 \times 1024 = \mathbf{1MB \text{ (megabyte)}}$
 - $2^{30} = 1024 \times 1024 \times 1024 = \mathbf{1GB \text{ (gigabyte)}}$
- How about these?
 - $2^{40} = \mathbf{1TB \text{ (terabyte)}}$
 - $2^{50} = \mathbf{1PB \text{ (petabyte)}}$
 - $2^{60} = \mathbf{1EB \text{ (exabyte)}}$
 - $2^{70} = \mathbf{1ZB \text{ (zettabyte)}}$
 - ...

Quick Checks

- $2^{22} = ?$
 - $2^2 \times 2^{20} = 4 \text{ Mega}$
- How many different values can a 32-bit variable represent?
 - $2^2 \times 2^{30} = 4 \text{ Giga}$
- Suppose that you have 2GB main memory in your computer. How many bits you need to address (cover) 2GB?
 - $2^1 \times 2^{30} = 2 \text{ GB, so 31 bits}$

Addition

- Decimal

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

- Binary

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

Binary Addition Examples

- Add the following 4-bit binary numbers

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

$$\begin{array}{r} 1011 \\ + 0110 \\ \hline \mathbf{0001} \end{array}$$

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
- Example:
 - add 13 and 5 using 4-bit numbers

$$\begin{array}{r} 11\ 1 \\ 1101 \\ +\ 0101 \\ \hline 10010 \end{array}$$

Signed Binary Numbers

- How does the computer represent positive and negative integer numbers?
- There are 2 ways
 - **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
 - Negative number: sign bit = 1
 - Positive number: sign bit = 0
- Example: 4-bit representations of ± 5 :
 $+5 = 0101_2$
 $-5 = 1101_2$
- Range of an N -bit sign/magnitude number:
 $[-(2^{N-1}-1), 2^{N-1}-1]$

Sign/Magnitude Numbers

- Problems

- Addition doesn't work naturally
- Example: $5 + (-5)$

$$\begin{array}{r} 0101 \\ + 1101 \\ \hline 10010 \end{array}$$

- Two representations of 0 (± 0)

0000 (+0)

1000 (-0)

Two's Complement Numbers

- Ok, so what's a solution to these problems?
 - **2's complement numbers!**
- Don't have the same problems as sign/magnitude numbers
 - Addition works fine
 - Single representation for 0
- **So, hardware designer likes it and uses 2's complement number system when designing adders (inside CPU)**

Two's Complement Numbers

- Same as unsigned binary numbers, but the most significant bit (MSB) has value of -2^{N-1}
 - Example
 - Biggest positive 4-bit number: $0111_2 (7_{10})$
 - Lowest negative 4-bit number: $1000_2 (-2^3 = -8_{10})$
- The most significant bit still indicates the sign
 - If MSB == 1, a negative number
 - If MSB == 0, a positive number
- Range of an N -bit two's complement number
 $[-2^{N-1}, 2^{N-1}-1]$

How to Make 2's Complement Numbers?

- Reversing the sign of a two's complement number

- Method:

1. Flip (Invert) the bits
2. Add 1

- Example

-7: 2's complement number of **+7**

$$\begin{array}{rcl} 0111 & (+7) & \\ 1000 & (\text{flip all the bits}) & \\ + & 1 & (\text{add 1}) \\ \hline 1001 & (-7) & \end{array}$$

Two's Complement Examples

- Take the two's complement of 0110_2

$$\begin{array}{r} 1001 \text{ (flip all the bits)} \\ + \quad 1 \text{ (add 1)} \\ \hline 1010 \end{array}$$

- Take the two's complement of 1101_2

$$\begin{array}{r} 0010 \text{ (flip all the bits)} \\ + \quad 1 \text{ (add 1)} \\ \hline 0011 \end{array}$$

Two's Complement Addition

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

$$\begin{array}{r} 111 \\ 0110 \\ + 1010 \\ \hline 10000 \end{array}$$

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

$$\begin{array}{r} 111 \\ 1110 \\ + 0011 \\ \hline 10001 \end{array}$$

How do We Check it in Computer?

```
1 #include <stdio.h>
2
3 int main()
4 {
5     int sa; // 1. "int implies a signed integer
6             // 2. "signed" is implied in "int" (int == signed int)
7             // 3. "int" value is stored in 32-bit
8
9     sa = -1 ;
10    printf("A signed number d'%d is stored as h'%0X at address %p in computer\n", sa, *(&sa), &sa);
11
12    sa = -2 ;
13    printf("A signed number d'%d is stored as h'%0X at address %p in computer\n", sa, *(&sa), &sa);
14
15    sa = -3 ;
16    printf("A signed number d'%d is stored as h'%0X at address %p in computer\n", sa, *(&sa), &sa);
17
18    return 0;
19
20 }
```

```
suhtw@suhtw-desktop:~/classes/comp211$ ./unsigned_signed
A signed number d'-1 is stored as h'FFFFFFFF at address 0x7fffedcc9e4c in computer
A signed number d'-2 is stored as h'FFFFFFFE at address 0x7fffedcc9e4c in computer
A signed number d'-3 is stored as h'FFFFFFFD at address 0x7fffedcc9e4c in computer
```

Increasing Bit Width

- Sometimes, you need to increase the bit width when you design a computer
 - For example, read a 8-bit data from main memory and store it to a 32-bit
- 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
- Examples
 - 4-bit representation of 3 = 0011
 - 8-bit sign-extended value: 00000011
 - 4-bit representation of -5 = 1011
 - 8-bit sign-extended value: 11111011

Zero-Extension

- Zeros are copied into most significant bits.
 - Number value may change.
- Examples
 - 4-bit value = 0011
 - 8-bit zero-extended value: 00000011
 - 4-bit value = 1011
 - 8-bit zero-extended value: 00001011

Number System Comparison

Number System	Range
Unsigned	$[0, 2^N-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

For example, 4-bit representation:

