CENG 3420
Computer Organization & Design
Lecture 04: Binary Number

Textbook: Chapter 2.4

Zhengrong Wang

CSE Department, CUHK

zhengrongwang@cuhk.edu.hk

Binary Number

Recap

- Arithmetic instructions to perform computation on registers.
  - E.g., add x1, x2, x3.
- Memory instructions to move value between registers and memory.
  - E.g., lw x1, 4(x2).

- But how does the computer perform the actual computation?
  - How to do $2 + 3$?
  - What about $2.3 + 3.4$?

Representation of Natural Number

- A natural number N can be written as M digits (d_i) in some base B:

$$N = (d_{M-1} d_{M-2} \dots d_1 d_0)_B$$
$$= d_{M-1} \times B^{M-1} + d_{M-2} \times B^{M-2} + \cdots + d_1 \times B + d_0$$
$$= \sum_{0}^{M-1} d_i \times B^i$$

- E.g., we have 10 fingers → naturally base 10

$$1234_{10} = 1 \times 10^3 + 2 \times 10^2 + 3 \times 10^1 + 4$$

- E.g., computer uses electronic signal (high/low voltage means 1/0) → base 2

$$1011_2 = 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 = 11_{10}$$

- E.g., For human readability, we often use base 16

$$beef_{16} = 11 \times 16^3 + 14 \times 16^2 + 14 \times 16^1 + 15 = 48879_{10}$$

Common Numerical System

Decimal (10)	Binary (2)	Octal (8)	Hexadecimal (16)
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

- Some interesting numerical systems:
  - Traditional Chinese weight used base-16. E.g. 半斤八两. (Why?)
  - Mayan used base-20.
  - Ancient Babylonians used base-60 (still used in our time system, e.g. 60 seconds for 1 minute).
  - Yuki people (California) used base-8 (spaces between fingers).
  - Soviet Union developed ternary computers (three values, -1, 0, 1).
- The choice of numerical base reflects the nature of the system.
- Digital computers operate using binary logic.

32-bit Unsigned Integers

- RV32-I uses 32-bit unsigned integers, with range $[0, 2^{32} - 1]$

```
+------------------------+----------------------------------+
| Unsigned Decimal Value | 32-bit Binary Representation     |
+------------------------+----------------------------------+
| 0                      | 00000000000000000000000000000000 |
| 1                      | 00000000000000000000000000000001 |
| ...                    | ...                              |
| 4294967295 (2^32-1)    | 11111111111111111111111111111111 |
+------------------------+----------------------------------+
```

- Right-most bit is least significant bit (LSB).
- Left-most bit is most significant bit (MSB).

32-bit Signed Integers

- We use two's complement to represent signed integers.
  - MSB = 0 → Non-negative number follows normal representation.
  - MSB = 1 → Negative number, the magnitude from two's complement.
- Convert between negative and positive number: Invert, then add one.
  - 2 = 0010 → Invert 1101 → Add one 1110 = -2
  - -2 = 1110 → Invert 0001 → Add one 0010 = 2

Signed Decimal Value	32-bit Binary Representation
-2,147,483,648 (-2^31)	10000000000000000000000000000000
-2,147,483,647	10000000000000000000000000000001
...	...
0	00000000000000000000000000000000
1	00000000000000000000000000000001
...	...
2,147,483,647 (2^31 - 1)	01111111111111111111111111111111

- Range $[-2^{31}, 2^{31}-1]$
- Note the asymmetry.

Why Two's Complement

- All modern processor uses two's complement for signed integers.
- In two's complement, $-x = 2^n - x$
  - E.g., $5 = 0101_2, -5 = 2^4 - 5 = 16 - 5 = 11 = 1011_2$

- This unifies addition for signed and unsigned integers!
  - Since we drop the overflow bit, addition is modulo 2^n
  - E.g., $8 - 5 = 1000_2 + 1011_2 = 10011_2 = 0011_2 = 3$
  - For addition, simply treat everything unsigned.

Signed and Unsigned Extension

- To extend a n-bit integer to m-bit integer (m > n).
- Signed extension: Duplicate the most significant bit (MSB), i.e. the sign bit.
  - Keep the sign unchanged!
- Unsigned extension: Fill with 0.

- Exercise: check that after signed extension, -4 is still -4.
- Exercise: what is the final value of 4-bit 8u signed extended into 8-bit?

4-bit Decimal	4-bit Binary	8-bit Binary	8-bit Decimal
4	0100	00000100	4
-4	1100 (2's comp)	11111100	-4

Conversion for Decimal Number

- Step 1: Divide the decimal number by the base.
- Step 2: Save the remainder (first remainder is the least significant digit).
- Repeat steps 1 and 2 until the quotient is zero.
- Result is in reverse order of remainders

- EX1: Convert 36_8 to binary value.
- EX2: Convert 36_{10} to binary value.
- EX3: Convert -6_{10} to binary value.

Addition and Subtraction

- Just like in primary school (carry & borrow 1s)

```
  0111          0111          0110
+ 0110        - 0110        - 0101
-------       -------       -------
```

- Two's complement operations are easy: do subtraction by negating then adding.

```
  0111          0111
- 0110   ->   + 1010
-------       -------
```

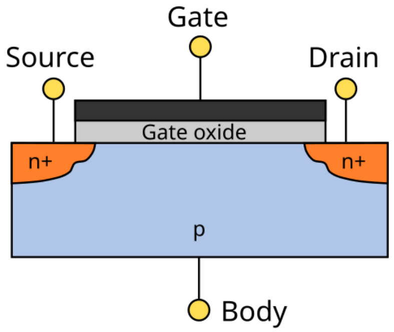
- Overflow (result too large for finite computer word).

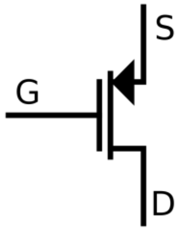
```
  0111
+ 1110
-------
```

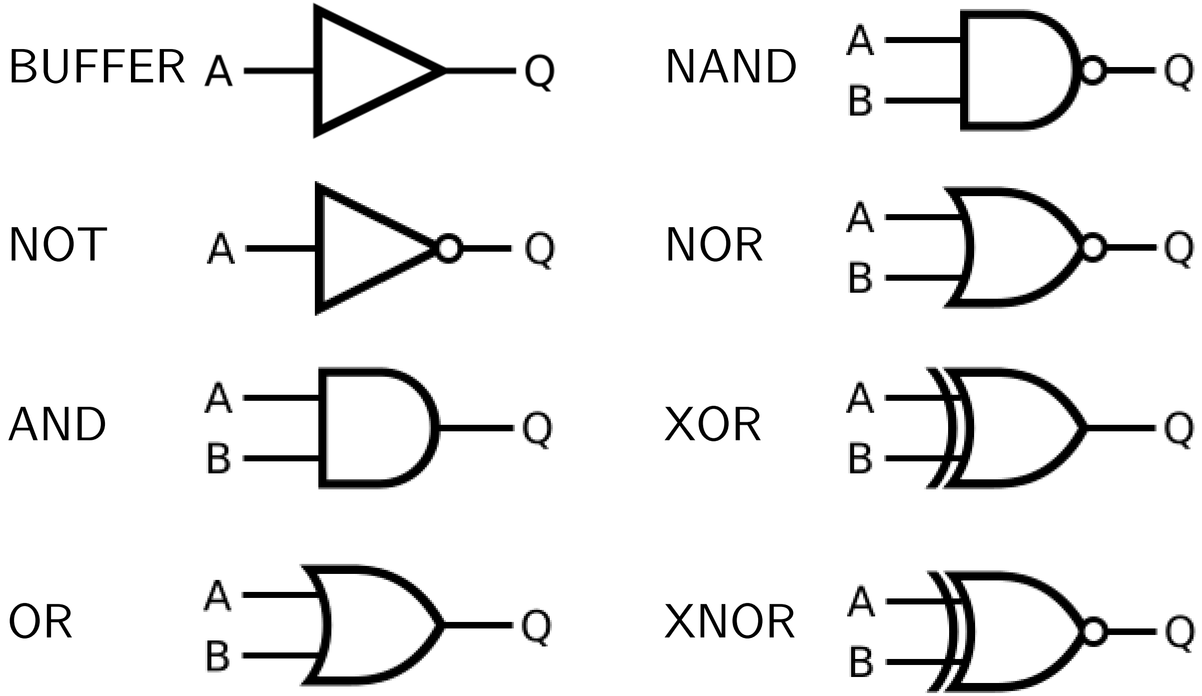
Logical Gates (Optional)

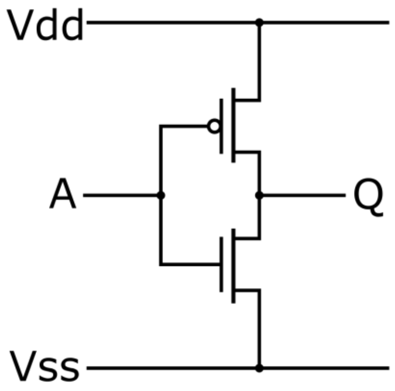
Transistors to Logical Gates

- Transistor:
  - Voltage on Gate controls conductivity between Source and Drain.
- You can implement logical gates with transistors.
  - Can you implement AND gate with transistors?







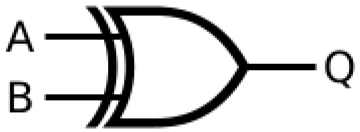


NOT w. MOSFET

Truth Table

- A means for describing how a logic circuit's output depends on the logic levels present at the circuit's inputs.
- The number of input combinations will equal 2^N for an N-input truth table.
  - Determine the true table of a three-input AND gate.

XOR



Truth Table of Q=XOR(A, B)

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0