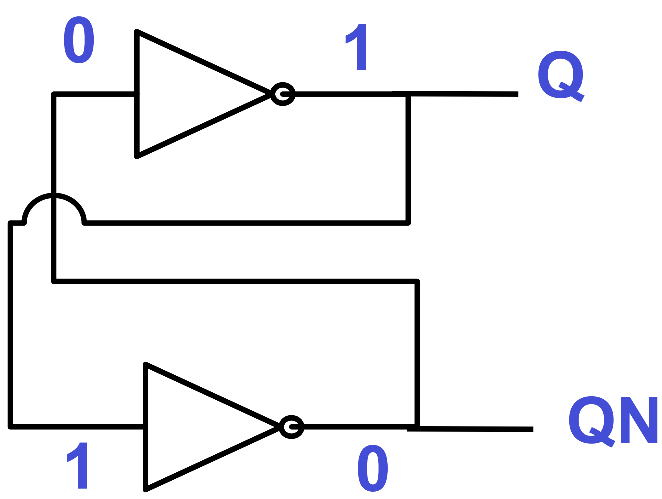
ECE 2300
Digital Logic & Computer Organization

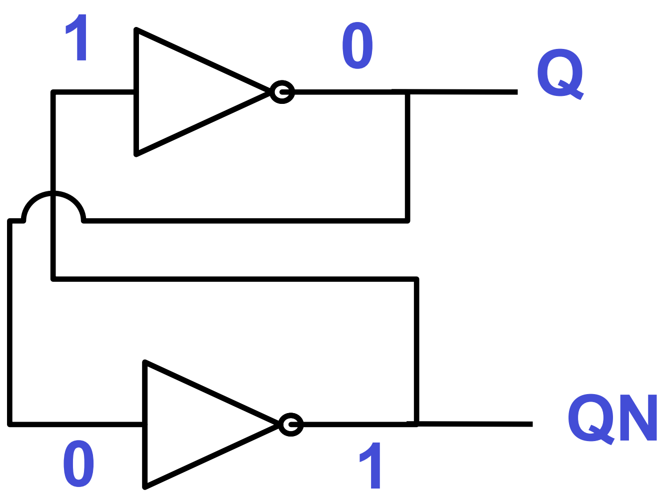
Fall 2016

Metastability
Binary Number Representations
Binary Arithmetic

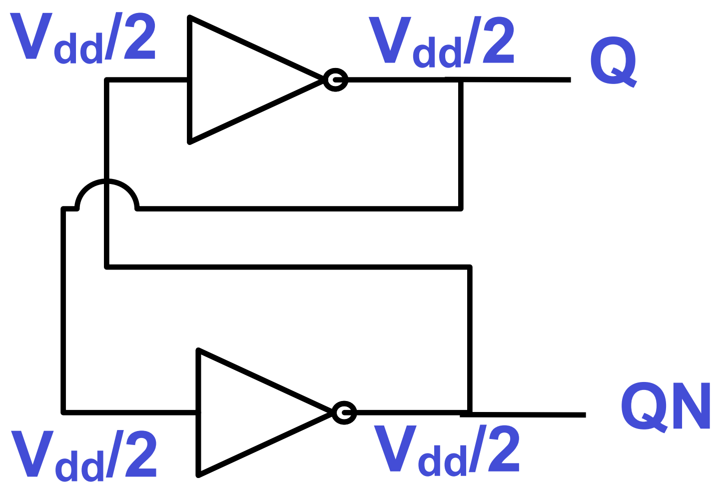
Cornell University

Bistable Element Stable States



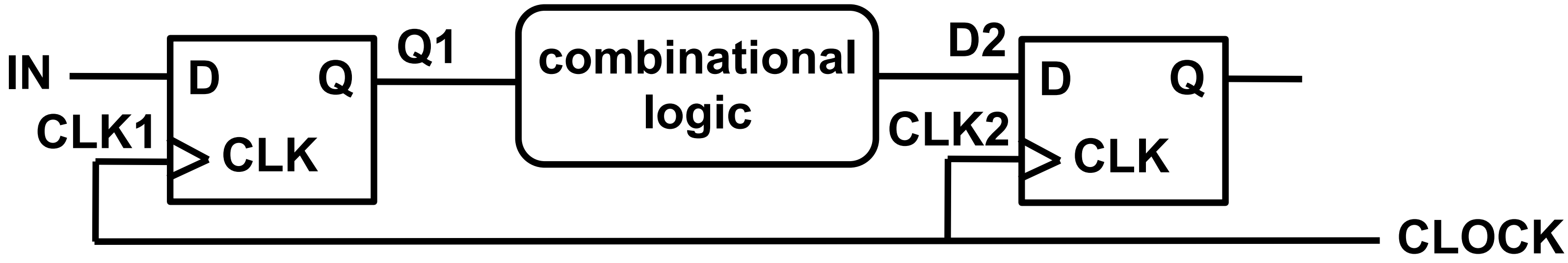


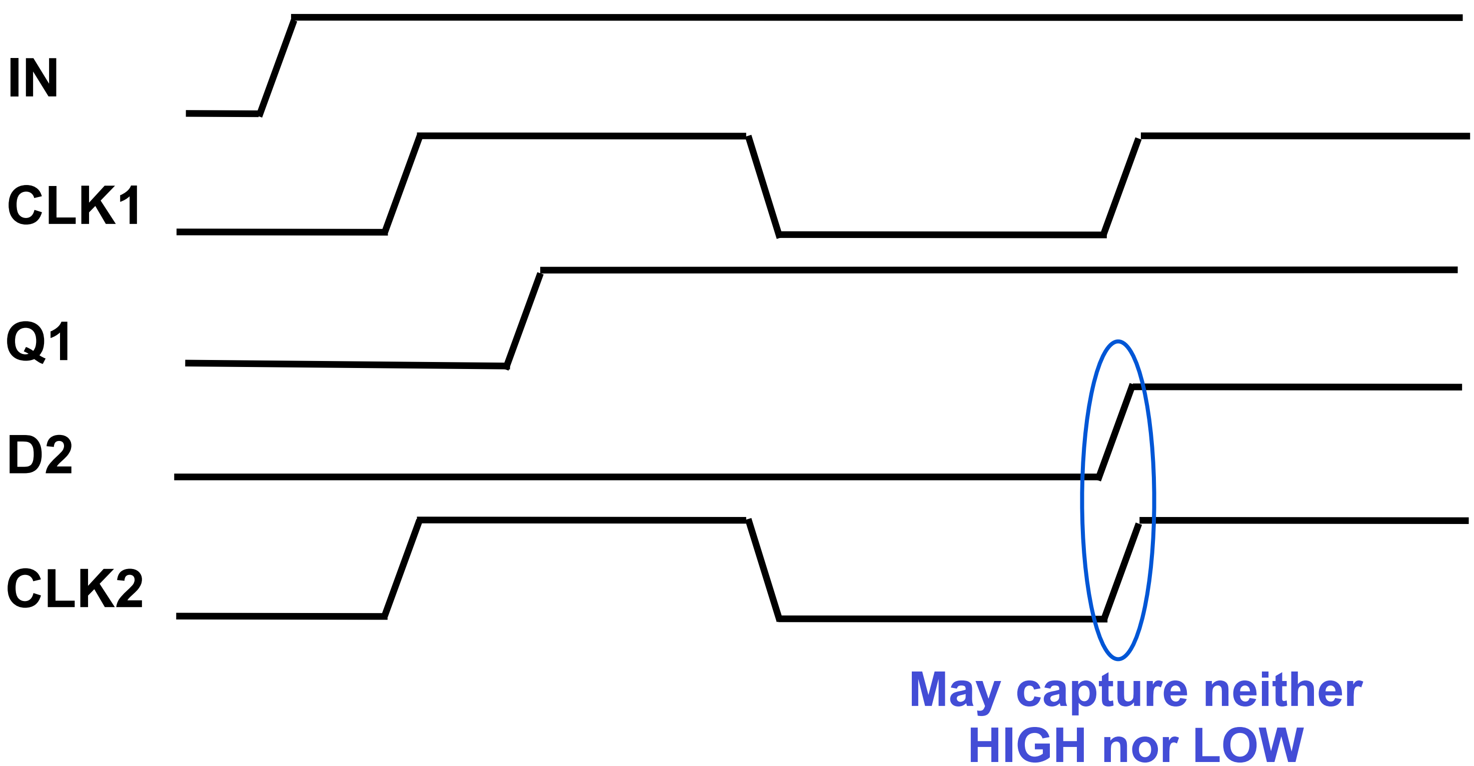
Metastable State



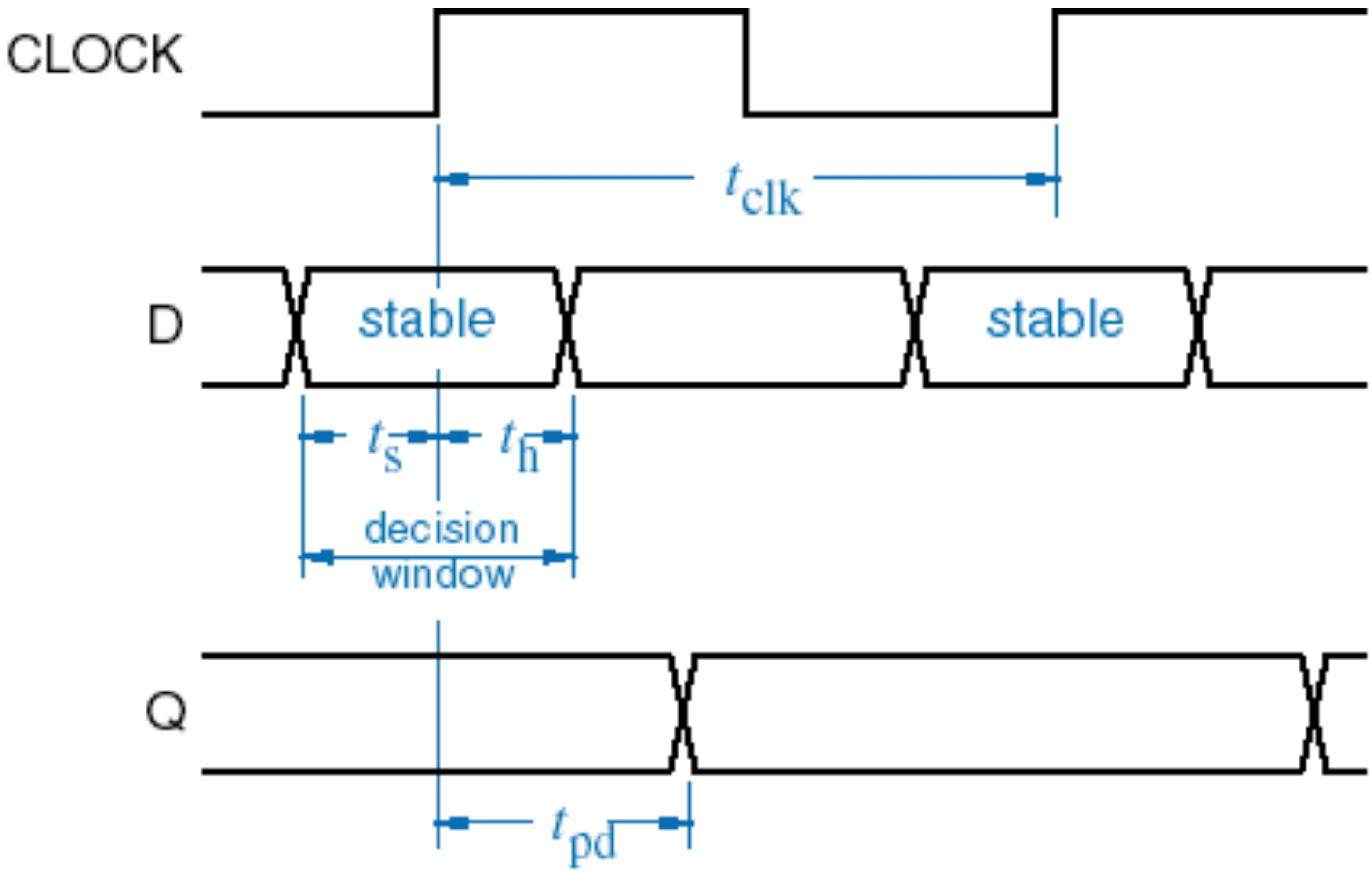
- Q and QN stuck in the undefined region between 0 and 1
- Eventually moves to stable state, but may take a while

How Can This Happen?

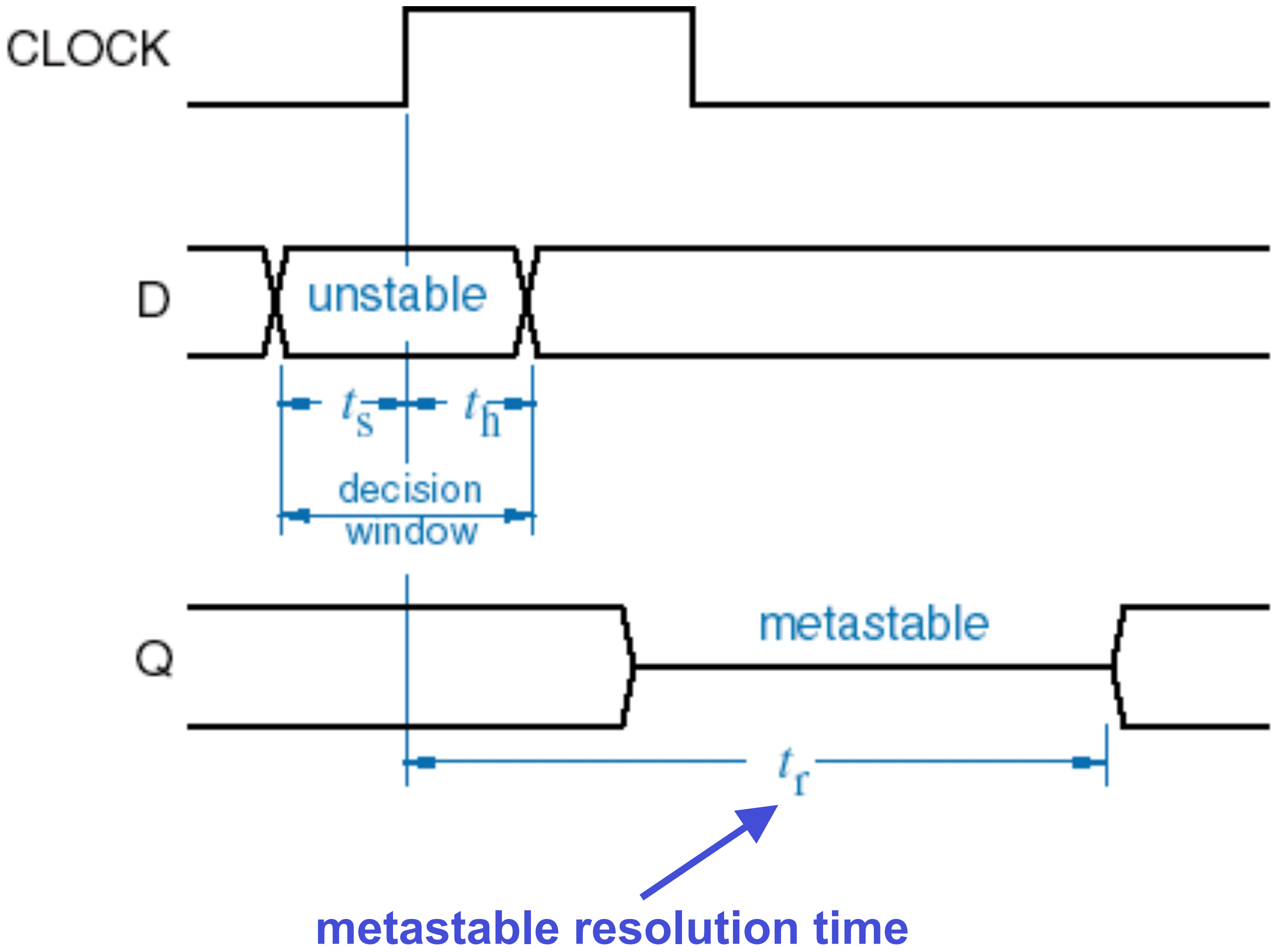




Stable FF Situation



Metastable Condition



Avoiding Metastability

- Causes of metastability
  - Input changes too soon before a clock edge
  - Input changes too soon after a clock edge
  - Clock pulse that is too narrow

- Avoid by meeting setup time, hold time, and minimum clock pulse width specifications

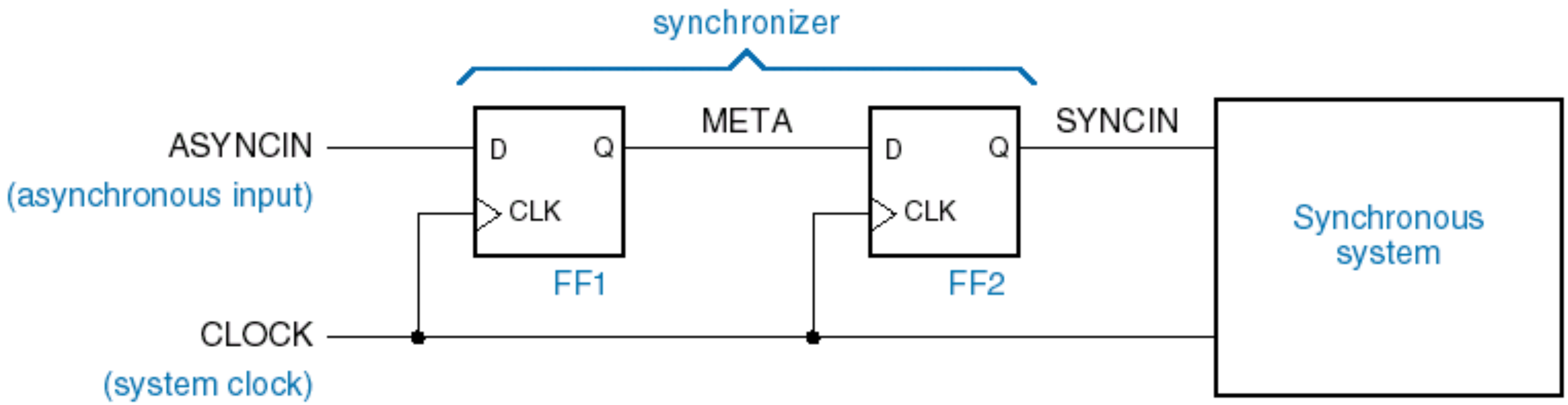
Asynchronous Inputs

- Inputs from the outside world may arrive at random times with respect to the clock
  - Keystrokes
  - Sensor inputs
  - Data received from a network
  - Sequential circuits with different clock sources that communicate

- Such *asynchronous inputs* may violate setup/hold times and cause metastability

- Must be *synchronized* before being sent to the sequential logic

Synchronizing Circuit



- **ASYNCIN may violate FF1 setup/hold times**
- **But META has a full cycle to settle to a 1 or 0 before it is sampled by FF2**
- **If META settles before the next triggering edge of the clock, SYNCIN will be stable**

Course Content

- Binary numbers and logic gates
- Boolean algebra and combinational logic
- Sequential logic and state machines

cut off for Prelim 1

- Binary arithmetic
- Memories

- Instruction set architecture
- Processor organization
- Caches and virtual memory
- Input/output
- Case studies

Positional Number Representation

- **What does 1432.67 mean?**

  $1432.67 = 1\times10^3 + 4\times10^2 + 3\times10^1 + 2\times10^0 + 6\times10^{-1} + 7\times10^{-2}$

  – Base 10 positional representation

  – Uses digits 0, 1, 2, …, 9

- **General base B positional representation**

  $a_n a_{n-1} \ldots a_2 a_1 a_0 = a_n B^n + a_{n-1} B^{n-1} + \ldots + a_2 B^2 + a_1 B^1 + a_0 B^0$

  – Uses digits 0, 1, 2, …, B-1

- **Bases of interest to computer designers**

  – Base 2, Binary (digits 0,1)

  – Base 16, Hexadecimal (digits 0,1,…,9,A,B,...,E,F)

Binary Numbers

- Recall weighted positional notation for decimal numbers

329

10^2 10^1 10^0

$\boxed{3x100 + 2x10 + 9x1 = 329}$

base 10
(decimal)

- Use similar weighted positional system for binary

Most Significant Bit (MSB)

101

Least Significant Bit (LSB)

2^2 2^1 2^0

$\boxed{1x4 + 0x2 + 1x1 = 5}$

base 2
(binary)

Binary Numbers

- For the binary number $b_{p-1}b_{p-2}\ldots b_1b_0.b_{-1}b_{-2}\ldots b_{-n}$ the decimal number is

$$D = \sum_{i=-n}^{p-1} b_i \bullet 2^i$$

- Examples

$$10011_2 = 1\times2^4 + 0\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$$
$$= 16 + 0 + 0 + 2 + 1$$
$$= 19_{10}$$

$$101.001_2 = 1\times2^2 + 1\times2^0 + 1\times2^{-3}$$
$$= 5.125_{10}$$

Unsigned Binary Numbers

- An *n*-bit unsigned number represents 2^n base 10 values
  - From 0 to 2^n-1

2^2	2^1	2^0	value
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Unsigned Binary Addition

- **Performed just like base-10**
  - **Add from right to left, propagating carry**

carry

$$\begin{array}{rl} 10010 & (18) \\ +\ \underline{01001} & (9) \\ 11011 & (27) \end{array} \qquad \begin{array}{rl} 10010 & (18) \\ +\ \underline{01011} & (11) \\ 11101 & (29) \end{array} \qquad \begin{array}{rl} 01111 & (15) \\ +\ \underline{00011} & (3) \\ 10010 & (18) \end{array}$$

$$\begin{array}{rl} 10111 & (23) \\ +\ \underline{111} & (7) \\ 11110 & (30) \end{array}$$

Signed Magnitude Representation

- Most significant bit is used as a sign bit
  - Sign bit of 0 for positive (0101 = 5, or 00000101 = 5)
  - Sign bit of 1 for negative (1101 = -5, or 10000101 = -5)

- *Range* is from $-(2^{n-1}-1)$ to $(2^{n-1}-1)$

- Drawbacks
  - Two representations for zero (+0 and -0)
  - Ordinary addition does not work

$$\begin{array}{rl} 00010 & (2) \\ +\underline{10010} & (-2) \\ 10100 & (\text{not } 0) \end{array}$$

Radix-Complement Representation

- Complement of an n-digit number formed by subtracting it from r^n, where r is the radix
- No sign bit
  - The number itself indicates positive/negative
- 10's complement
  - Example: 2372 → 10^4 - 2372 = 7628 [-2372]
- 2's complement
  - Example: 0101 [5] → 2^4 - 0101 = 1011 [-5]

Two's Complement Representation

- MSB has weight -2^{n-1}
- Range of an n-bit number: -2^{n-1} through $2^{n-1}-1$
  - Most negative number (-2^{n-1}) has no positive counterpart (one more negative than positive)

-2^3	2^2	2^1	2^0	value
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

-2^3	2^2	2^1	2^0	value
1	0	0	0	-8
1	0	0	1	-7
1	0	1	0	-6
1	0	1	1	-5
1	1	0	0	-4
1	1	0	1	-3
1	1	1	0	-2
1	1	1	1	-1

Two's Complement Representation

- Positive numbers and zero are same as unsigned binary representation

- To form a negative number
  - Start with the positive number
  - Flip every bit
  - Then add one

$$\begin{array}{rl} 00101 & (5) \\ 11010 & \text{(1's comp)} \\ +\quad\quad 1 & \\ \hline 11011 & (-5) \end{array} \qquad \begin{array}{rl} 01001 & (9) \\ 10110 & \text{(1's comp)} \\ +\quad\quad 1 & \\ \hline 10111 & (-9) \end{array}$$

Two's Complement Shortcut

- To take the two's complement of a number
  - Copy bits from right to left up to and including the first "1"
  - Flip remaining bits to the left

$$
\begin{array}{r}
011010000 \\
100101111 \\
+\underline{1} \\
100110000
\end{array}
$$

$$
\begin{array}{c|c}
0110 & 10000 \\
\big\downarrow \text{(flip)} & \big\downarrow \text{(copy)} \\
1001 & 10000
\end{array}
$$

Two's Complement Addition

- Procedure for addition is the same as unsigned addition regardless of the signs of the numbers

$$\begin{array}{rl} 00101 & (5) \\ +\underline{11011} & (-5) \\ 00000 & (0) \end{array} \qquad \begin{array}{rl} 01001 & (9) \\ +\underline{10111} & (-9) \\ 00000 & (0) \end{array}$$

Converting Binary (2's C) to Decimal

1. If MSB = 1, take two's complement to get a positive number
2. Add powers of 2 for bit positions that have a "1"
3. If original number was negative, add a minus sign

$$X = 01101000_{two}$$
$$= 2^6+2^5+2^3 = 64+32+8$$
$$= 104_{ten}$$

Assuming 8-bit 2's complement numbers

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

More Examples

$$X = 00100111_{two}$$
$$= 2^5+2^2+2^1+2^0 = 32+4+2+1$$
$$= 39_{ten}$$

$$X = 11100110_{two}$$
$$-X = 00011010$$
$$= 2^4+2^3+2^1 = 16+8+2$$
$$= 26_{ten}$$
$$X = -26_{ten}$$

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Assuming 8-bit 2's complement numbers

Converting Decimal to Binary (2's C)

First Method: *Division*

1. **Change to positive decimal number**
2. **Divide by two – remainder is least significant bit**
3. **Keep dividing by two until answer is zero, recording remainders from right to left**
4. **Append a zero as the MSB; if original number was negative, take two's complement**

$X = 104_{ten}$

104/2	=	52 r0	*bit 0*
52/2	=	26 r0	*bit 1*
26/2	=	13 r0	*bit 2*
13/2	=	6 r1	*bit 3*
6/2	=	3 r0	*bit 4*
3/2	=	1 r1	*bit 5*
1/2	=	0 r1	*bit 6*

$X = 01101000_{two}$

Converting Decimal to Binary (2's C)

Second Method: *Subtract Powers of Two*

1. Change to positive decimal number
2. Subtract largest power of two
   less than or equal to number
3. Put a one in the corresponding bit position
4. Keep subtracting until result is zero
5. Append a zero as MSB;
   if original was negative, take two's complement

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

$X = 104_{ten}$

104 - 64	=	40	*bit 6*
40 - 32	=	8	*bit 5*
8 - 8	=	0	*bit 3*

$X = 01101000_{two}$

Hexadecimal Notation

- **Often convenient to write binary (base-2) numbers as hexadecimal (base-16) numbers**
  - Fewer digits: 4 bits per hex digit
  - Less error prone: easy to misread long string of 1's and 0's

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

Converting from Binary to Hex

- Every group of four bits is a hex digit
  - Start grouping from right-hand side

011	1010	1000	1111	0100	1101	0111
⇓	⇓	⇓	⇓	⇓	⇓	⇓
3	A	8	F	4	D	7

This is not a new machine representation, just a convenient way to write the number

Before Next Class

- H&H 5.3

Next Time

More Binary Arithmetic