Lecture 1 Outline

Reminders to self:

- ❑ Turn on lecture recording to Cloud
- ❑ Turn on Zoom microphone

- Last Lecture
  - Finished Number Systems and Conversion
  - Binary Arithmetic – almost done
- Today's Lecture
  - Binary Arithmetic – one more division example
  - Representation of negative numbers in binary
  - Addition involving signed binary numbers
  - Binary codes

Handouts and Announcements

- Announcements
  - Homework Problems 1-6, 1-7, 1-8
    - Already on Carmen
    - Due in Carmen
      - HW 1-6: 11:25am, Monday 1/23
      - HW 1-7, 1-8: 11:25am Wednesday 1/25
  - Homework Problem 1-5 reminder
    - Assigned on Carmen on 1/13
    - Due in Carmen 11:59pm, Thursday 1/19
  - Read for Friday: Pages 29, 36-46

Handouts and Announcements

- Announcements
  - Mini-Exam 1 Reminder
    - Available 5pm Monday 1/23 through 5:00pm Tuesday 1/24
    - Due in Carmen PROMPTLY at 5:00pm on 1/24
    - Designed to be completed in ~36 min, but you may use more
    - When planning your schedule:
      - I recommend building in 10-15 min extra
      - To allow for downloading exam, signing and dating honor pledge, saving solution as pdf, and uploading to Carmen
    - I also recommend not procrastinating
  - Exam review topics available on Carmen

Binary Arithmetic

Example: divide binary 1011111 by 1100 (decimal 95 by 12, in binary)

```
             111
      ----------
1100 | 1011111
       1100
       ---------
        101111
         1100
        --------
         10111
          1100
         -------
          1011
```

result: 111 with remainder 1011

Representation of Negative Numbers

- Last lecture, if an arithmetic operation resulted in a number with more bits, we added bits
- In digital systems number of bits typically fixed by hardware
- Generically, n-bits where n is a positive integer
- Last lecture all numbers were positive
- Common methods of representing both positive and negative numbers are:

	Sign and magnitude	1's complement	2's complement
Range for n-bits	$-(2^{n-1}-1)$ to $+(2^{n-1}-1)$		-2^{n-1} to $+(2^{n-1}-1)$
Representation of zero	Both +0 and -0 Causes complications with arithmetic		Only +0

- In each of these methods, the leftmost bit of a number is 0 for positive numbers and 1 for negative numbers.

Representation of Negative Numbers

Three systems for representing negative numbers in binary - Overview: (with 4-bit examples)**:**

- **Sign & Magnitude**: Most significant bit is the sign
  - Ex: $-5_{10} = 1101_2$
- **2's Complement**: $N^* = 2^n - N$ $\quad N \equiv$ magnitude of number
  - Ex: $-5_{10} \rightarrow N = 5;\ N^* = 2^4 - 5 = 16 - 5 = 11_{10} = 1011_2$
  - Note: This approach is a "human" approach. You should learn it, but for the purposes of Digital Logic (this course) we ultimately want to know a "logic circuit" approach to 2's Complement
- **1's Complement**: $\bar{N} = (2^n - 1) - N$
  - Ex: $-5_{10} \rightarrow N = 5;$
    $\bar{N} = (2^4 - 1) - 5 = 16 - 1 - 5 = 10_{10} = 1010_2$

Representation of Negative Numbers

TABLE 1-1 Signed Binary Integers (word length: $n = 4$)

$+N$	Positive Integers (all systems)	$-N$	Negative Integers: Sign and Magnitude	Negative Integers: 2's Complement N^*	Negative Integers: 1's Complement $\overline{N}$
+0	0000	−0	1000	——	1111
+1	0001	−1	1001	1111	1110
+2	0010	−2	1010	1110	1101
+3	0011	−3	1011	1101	1100
+4	0100	−4	1100	1100	1011
+5	0101	−5	1101	1011	1010
+6	0110	−6	1110	1010	1001
+7	0111	−7	1111	1001	1000
		−8	——	1000	——

$+(2^{n-1}-1)$

$-(2^{n-1}-1)$

-2^{n-1}

$N^* = 2^n - N$

$\overline{N} = (2^n - 1) - N$

- Designing logic circuits to do arithmetic for sign and magnitude binary numbers is awkward
- One method: convert into 2's (or 1's) complement, do arithmetic, convert back
- We will look at addition in 2's and 1's complement

Representation of Negative Numbers

2’s compliment:

- Positive number, N, is represented by a 0 followed by the magnitude of N as in the sign and magnitude system
- For negative numbers, $-N$, 2’s compliment is represented by N^*, as follows

$$N^* = 2^n - N$$

where the word length is n bits

Representation of Negative Numbers

2's compliment:

- Alternative ways to find 2's complement
  1. Do bit-by-bit complement, then add 1
  2. Working right-to-left
     a) Leave 0's as 0 until first 1 is encountered
     b) Leave first 1 as 1
     c) Complement all remaining bits to the left

Representation of Negative Numbers

Example: 4-bits, complement $+6_{10}$

- All three ways
- Then work it backward

$N^* = 2^n - N$

$+6 \Rightarrow 0110$ Working in decimal $2^4 - 6 = 16 - 6 = 10_{10} = 1010_2$

Working in binary $10000 - 0110 = 1010$

Using first alternative approach: $0110 \Longrightarrow 1001 + 1 = 1010$
(bit-by-bit complement +1)

To first 1

Using second alternative approach: $\underbrace{01}_{\text{flip}}\underbrace{10}_{\text{keep}} \Longrightarrow 1010$

The alternative approaches are more amenable to logic circuit implementation than is the first approach (logic inverter gate, to flip a bit)

Representation of Negative Numbers

Example: 4-bits, complement $+6_{10}$

- All three ways
- $\boxed{\text{Then work it backward}}$

Starting from 1010_2 for -6_{10}, find code for $+6_{10}$

$N = 2^n - N^*$

$10000 - 1010 = 0110$

Two ways to use first alternative approach:

1. Do same algorithm as before $\quad 1010 \xRightarrow{\text{Bit-by-bit complement}} 0101 + 1 = 0110$

2. Reverse steps of algorithm $\quad 1010 - 1 = 1001 \xRightarrow[\text{Bit-by-bit complement}]{} 0110$

Using second alternative approach: $\quad \underbrace{10}_{\text{flip}}\underbrace{10}_{\text{keep}} \Rightarrow \quad 0110$

To first 1

Representation of Negative Numbers

Addition is straightforward. Like unsigned binary addition, except:

- Ignore any carry from the sign position
- Watch out for overflows (results that exceed range for # of bits)

Example: 2's Compliment Addition

$n = 4$ examples

1. Addition of two positive numbers, sum $< 2^{n-1}$ — $+3_{10}$ and $+4_{10}$

$$\begin{array}{r} 0011 \\ \underline{0100} \\ 0111 \end{array} \quad \text{(correct answer)} \qquad\qquad \begin{array}{r} +3 \\ \underline{+4} \\ +7 \end{array}$$

2. Addition of two positive numbers, sum $\geq 2^{n-1}$ — $+5_{10}$ and $+6_{10}$

$$\begin{array}{r} +5 \\ \underline{+6} \\ \end{array} \qquad \begin{array}{r} 0101 \\ \underline{0110} \\ 1011 \end{array} \quad \longleftarrow \text{wrong answer because of overflow (+11 requires 5 bits including sign)}$$

Example: 2's Compliment Addition (continued)

3. Addition of positive and negative numbers (negative number has greater magnitude)

$$\begin{array}{rr} +5 & 0101 \\ -6 & 1010 \\ \hline -1 & 1111 \end{array} \quad \text{(correct answer)}$$

4. Same as case 3 except positive number has greater magnitude

$$\begin{array}{rr} -5 & 1011 \\ +6 & 0110 \\ \hline +1 & (1)0001 \end{array} \quad \longleftarrow \text{correct answer when the carry from the sign bit is ignored (this is } \textit{not} \text{ an overflow)}$$

When adding two numbers of opposite sign the result is always closer to zero than either operand. Never produces an overflow.

Example: 2's Compliment Addition (continued)

5. Addition of two negative numbers, $|\text{sum}| \le 2^{n-1}$

$$\begin{array}{rr} -3 & 1101 \\ -4 & 1100 \\ \hline -7 & (1)1001 \end{array} \quad \longleftarrow$$ correct answer when the last carry is ignored (this is *not* an overflow)

6. Addition of two negative numbers, $|\text{sum}| > 2^{n-1}$

$$\begin{array}{rr} -5 & 1011 \\ -6 & 1010 \\ \hline & (1)0101 \end{array} \quad \longleftarrow$$ wrong answer because of overflow (-11 requires 5 bits including sign)