Lecture 1 Outline

Reminders to self:

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

- Last Lecture
  - Overview
  - Number Systems and Conversion – almost done
- Today’s Lecture
  - Finish Number Systems and Conversion
  - Binary Arithmetic
  - Introduction to Negative Numbers in Binary

Handouts and Announcements

- Announcements
  - Homework Problems 1-5
    - Already on Carmen
    - Due in Carmen 11:59pm, Thursday 1/19
  - Homework Problems 1-2, 1-3, & 1-4 reminder
    - Assigned on Carmen on 1/12
    - Due in Carmen 11:59pm, Tuesday 1/17
  - Reminder: no class Monday – University Holiday
  - Read for Wednesday: Pages 21-23

Number Systems and Conversion

Example: Convert $2AB.13_{13}$ to base 5

$$2AB.13_{13} = 2 \times 13^2 + 10 \times 13 + 11 + \frac{1}{13} + \frac{3}{13^2} = 479.0947_{10}$$

Where we left off last lecture

- I rounded to 4 digits base 10
- Extra precision at intermediate step to not introduce rounding error into final result

Check: $3 \times 5^3 + 4 \times 5^2 + 4 + \frac{2}{25} + \frac{1}{125} + \frac{4}{5^4} = 479.0944$

Number Systems and Conversion

Conversion from binary to hexadecimal and binary to octal:

- Conversion from binary to hexadecimal (and conversely) can be done by inspection because each hexadecimal digit corresponds to exactly four binary digits (bits).

$$1001101.010111_2 = \underbrace{0100}_{4}\ \underbrace{1101}_{D}\ .\ \underbrace{0101}_{5}\ \underbrace{1100}_{C} = 4D.5C_{16}$$

- A similar conversion can be done from binary to octal, base 8 (and conversely), except each octal digit corresponds to three binary digits, instead of four.

$100101101011010_2 = 100\quad 101\quad 101\quad 011\quad 010 = 45532_8$

- If the number of bits is not a multiple of 4 (for hex) or of 3 (for octal)
  - For the integer part, add leading 0s
  - For the fractional part, add trailing 0s

Number Systems and Conversion

Binary Addition:

- For simplicity, arithmetic in digital systems is performed in binary (base 2)

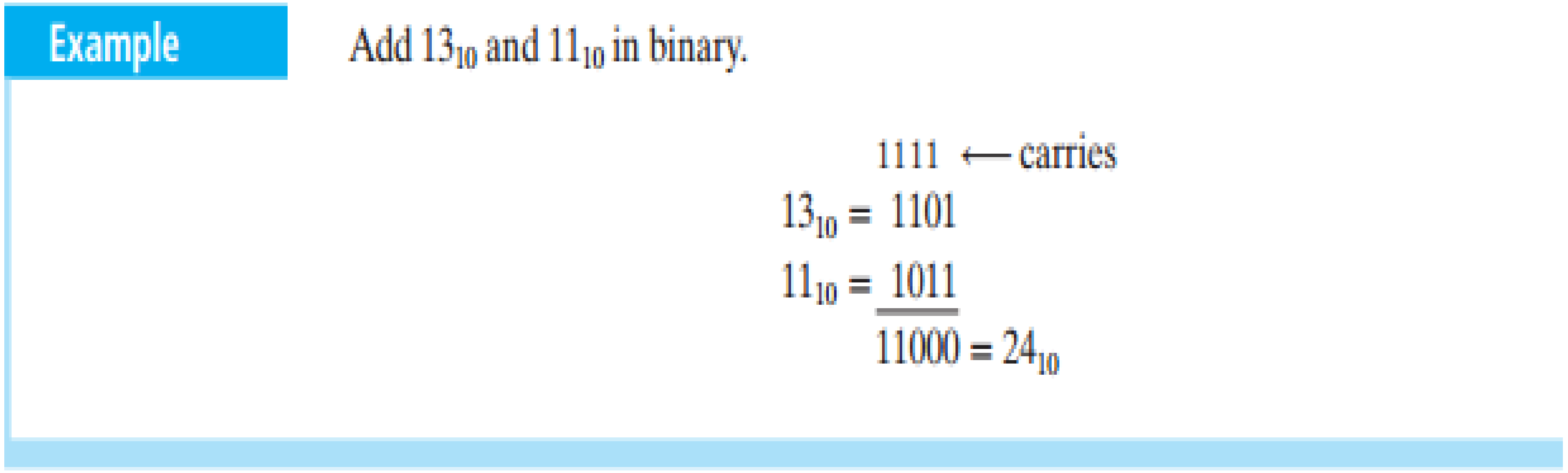
- Addition table for binary:

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 0 \quad \text{and carry 1 to the next column}$$

Carrying 1 to a column is equivalent to adding 1 to that column.

Binary Arithmetic

Example 5: Binary addition



Example Add 13_{10} and 11_{10} in binary.

$$\begin{array}{rl} & 1111 \longleftarrow \text{carries} \\ 13_{10} = & 1101 \\ 11_{10} = & \underline{1011} \\ & 11000 = 24_{10} \end{array}$$

Binary Arithmetic

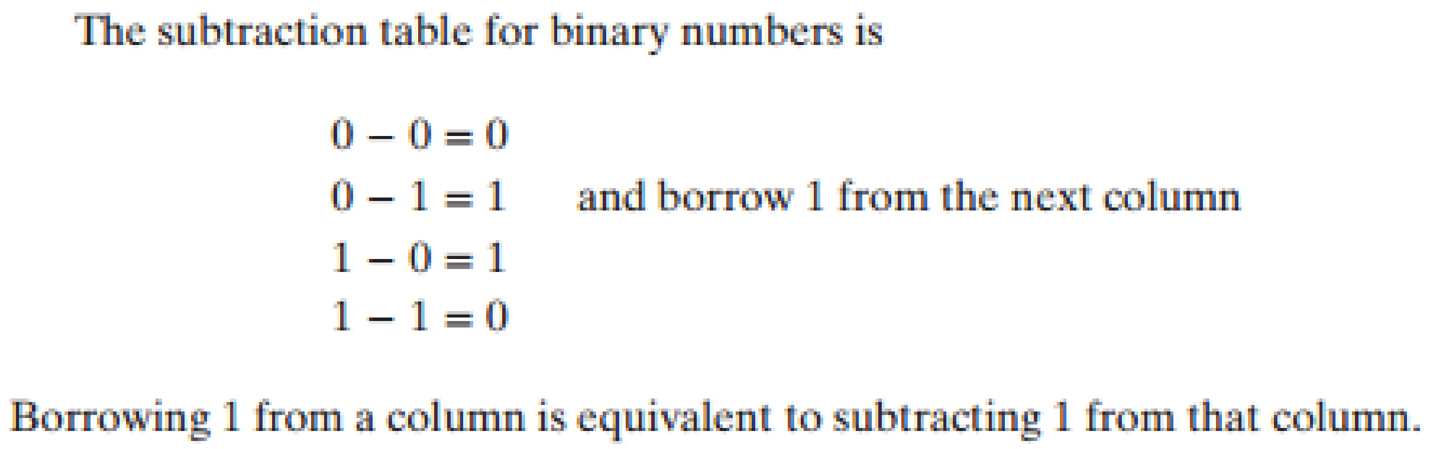
Example: Add 14_{10} and 8_{10} in binary

$$\begin{array}{r} 1110 \\ +\ 1000 \\ \hline 10110_2 \end{array} = 22_{10}$$

Binary Arithmetic

Binary Subtraction:

- Subtraction table for binary:



The subtraction table for binary numbers is

$0 - 0 = 0$

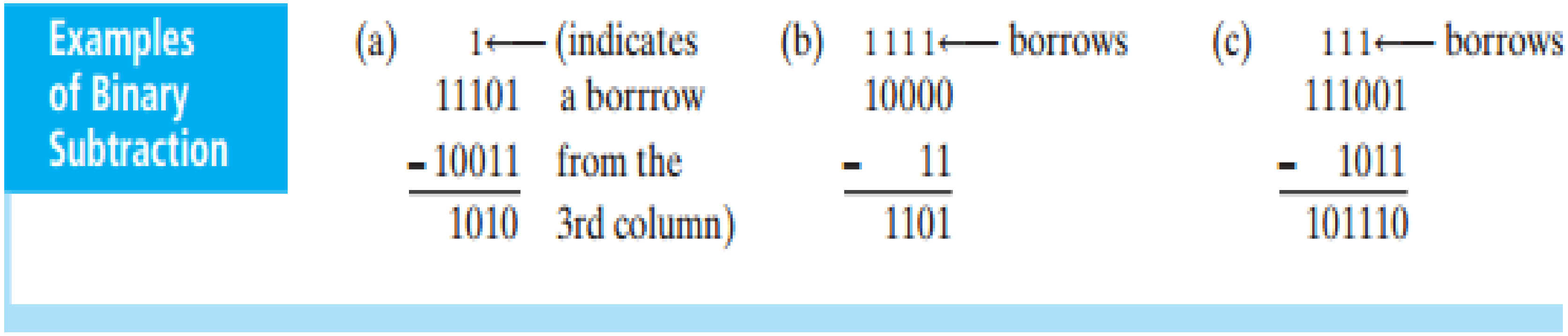
$0 - 1 = 1$ and borrow 1 from the next column

$1 - 0 = 1$

$1 - 1 = 0$

Borrowing 1 from a column is equivalent to subtracting 1 from that column.

Example 6: Subtractions in Binary



Examples of Binary Subtraction

```
(a)       1←— (indicates     (b)  1111←— borrows     (c)      111←— borrows
      11101    a borrow            10000                    111001
    - 10011    from the          -    11                  -   1011
      ─────                      ───────                  ────────
       1010    3rd column)          1101                    101110
```

Binary Arithmetic

Example: Subtract binary 1110_2 from 10010_2

14_{10} 18_{10}

$$
\begin{array}{r}
\overset{1}{}\,\overset{1}{1}0010 \\
-\ 1110 \\
\hline
00100_2 = 4_{10}
\end{array}
$$

Binary Arithmetic

Binary Multiplication:

- Multiplication table for binary:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Binary Arithmetic

Example 7: Binary Multiplication

13_{10} by 11_{10} in binary:

$$
\begin{array}{r}
1101 \\
1011 \\
\hline
1101 \\
'1101 \\
'0000 \\
|\ '1101 \\
\hline
10001111 = 143_{10}
\end{array}
$$

Each partial product is either
- Multiplicand shifted over
- Zero

Binary Arithmetic

When doing binary multiplication, a common way to avoid carries greater than 1 is to add in the partial products one at a time as illustrated by the following example: Multiply binary 1011 by 1101

$$
\begin{array}{rcccccccc}
 & & & & & 1 & 0 & 1 & 1 \\
\times & & & & & 1 & 1 & 0 & 1 \\
\hline
 & & & & & 1 & 0 & 1 & 1 \\
 & {}^{1} & {}^{1} & {}^{1} & 0 & 0 & 0 & 0 & \\
 & & & 1 & 0 & 1 & 1 & & \\
+ & & 1 & 0 & 1 & 1 & & & \\
\hline
 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
$$

$\leftarrow$ not a partial product example lol

Binary Arithmetic

Binary Division:

- Binary division is similar to decimal division, except it is much easier because the only two possible quotient digits are 0 and 1.
- We start division by comparing the divisor with the most significant bits of the dividend.
  - If we cannot subtract without getting a negative result, we move one place to the right and try again.
  - If we can subtract, we place a 1 for the quotient above the number we subtracted from and append the next dividend bit to the end of the difference and repeat this process with this modified difference until we run out of bits in the dividend.

Binary Arithmetic

Example 8: Binary Division

145_{10} by 11_{10} in binary

```
           1101
       ________
  1011 )10010001
         1011
         ----
          111001
          1011
          ----
            1101
            1011
            ----
              10
```

The quotient is 1101 with a remainder of 10.

Binary Arithmetic

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

$$\begin{array}{r} 1100 \overline{)\,1011111} \\ \underline{1100} \\ 101111 \end{array}$$

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:

- **Sign & Magnitude**: Most significant bit is the sign
  - Ex: $-5_{10} = 1101_2$
- **2's Complement**: $N^* = 2^n - N$
  - 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

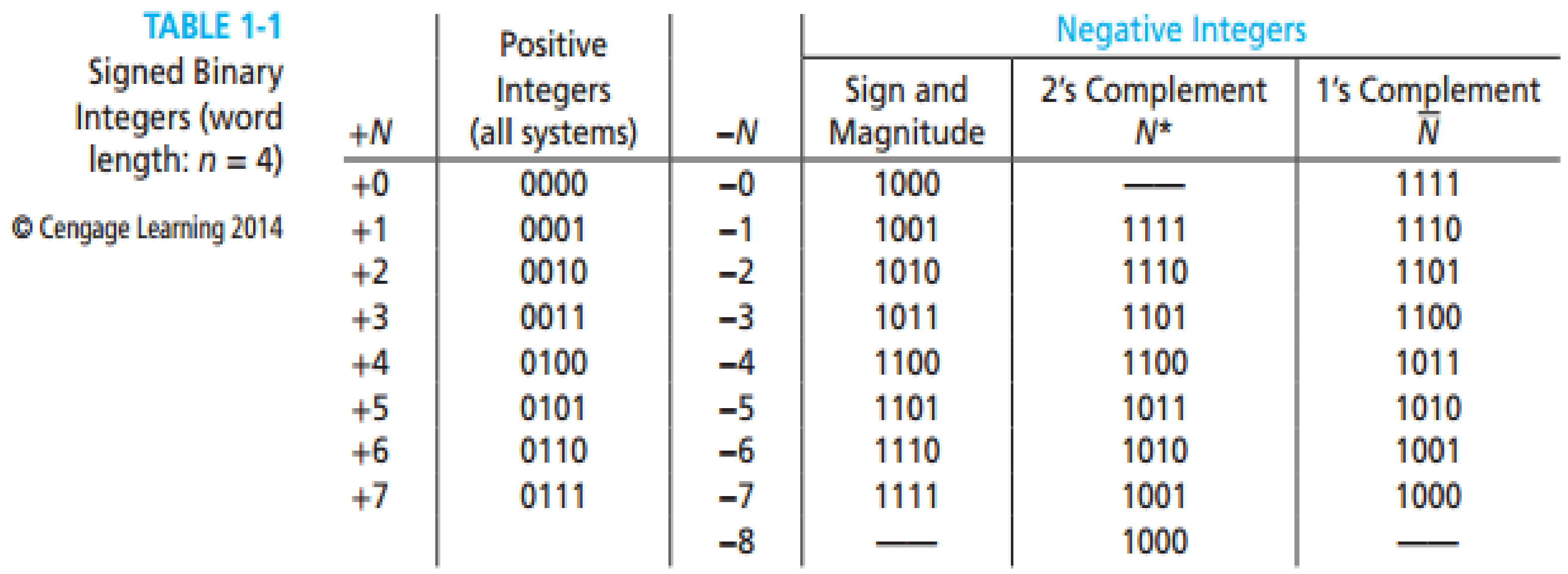


TABLE 1-1 Signed Binary Integers (word length: $n = 4$)

			Negative Integers		
$+N$	Positive Integers (all systems)	$-N$	Sign and Magnitude	2's Complement N^*	1's Complement $\bar{N}$
+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	——

- 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