Communication Technical Engineering Department

1st Stage

Digital Logic- UOMU028021

Lecture 3 – Conversion and Arithmetic Operations

Dr. Mohammed Fadhil
PhD in Computer Networks
Email: mohammed.fadhil1@uomus.edu.iq

Octal-to-Decimal Conversion

- Method:
  - **Multiply** each digit by its positional weight (**powers of 8**) and **sum** the results.
- Example:
  - Convert 372_8 → Decimal
  - **372_8** $= 3 \times 8^2 + 7 \times 8^1 + 2 \times 8^0$
  - $= 3 \times 64 + 7 \times 8 + 2 \times 1$
  - $= 192 + 56 + 2$
  - $= 250_{10}$

Octal to Decimal Conversion (with Fraction)

Note: Multiply each digit by 8^position
Integer part: left to right, powers start at 0
Fractional part: right to left, negative powers

- **Example 1:**
  - Convert **24.6_8** to its decimal equivalent
  - $24.6_8 = 2\times8^1 + 4\times8^0 + 6\times8^{-1}$
  - $= 2\times8 + 4\times1 + 6\times0.125$
  - $= 16 + 4 + 0.75$
  - $= 20.75_{10}$
- Result:
  - $24.6_8 = 20.75_{10}$

- **Example 2:**
  - Convert **15.2_8** to its decimal equivalent
  - $15.2_8 = 1\times8^1 + 5\times8^0 + 2\times8^{-1}$
  - $= 1\times8 + 5\times1 + 2\times0.125$
  - $= 8 + 5 + 0.25$
  - $= 13.25_{10}$
- Result:
  - $15.2_8 = 13.25_{10}$

Decimal-to-Octal Conversion

- **Integer Conversion (Repeated Division by 8)**
  - Method: **Divide** by 8, **track** remainders **bottom-up**.
  - Example: Convert 266_{10} → Octal

Division	Quotient	Remainder
266 ÷ 8	33	**2** (LSB)
33 ÷ 8	4	**1**
4 ÷ 8	0	**4** (MSB)

  - **Remainders** in **reverse** order: 4 1 2 → $\mathbf{412_8}$

Decimal-to-Octal Conversion

- **Fraction Conversion (Repeated Multiplication by 8)**
  - Method: **Multiply** by 8, **track** carries (integer parts) **top-down**.
  - Example: Convert $\mathbf{0.23_{10}}$ → Octal (**3 places**)

Multiplication	Product	Carry (Integer)	Fraction
0.23 × 8	1.84	**1** (MSD)	0.84
0.84 × 8	6.72	**6**	0.72
0.72 × 8	5.76	**5** (LSD)	0.76

  - **Carries** in **order**: 1 6 5 → $\mathbf{0.165_8}$
  - The process is terminated after **three places**; if more accuracy were required, we continue multiplying to obtain more octal digit.

Octal-to-Binary Conversion

- **Key Advantage:**
  - The primary advantage of the octal number system is the ease with which conversion can be made between binary and octal numbers.
  - Each octal digit maps directly to a 3-bit binary group, simplifying conversions.
  - Octal-Binary Reference Table

Octal Digit	0	1	2	3	4	5	6	7
3-Bit Binary	000	001	010	011	100	101	110	111

Octal-to-Binary Conversion

- **Integer Conversion**
- **Convert $472_8 \rightarrow$ Binary**
  - Step-by-Step:
    - **Split** each octal digit: 4 | 7 | 2
    - **Map** to 3-bit binary using the table:
      - 4 → 100
      - 7 → 111
      - 2 → 010
    - **Combine** results: 100 111 010 → 100111010_2
  - Visual Guide:

Octal Digit	4	7	2
Binary	100	111	010

  - Result:
    - $472_8 = 100111010_2$

Octal-to-Binary Conversion

- **Fractional Conversion**
- **Convert $34.562_8 \rightarrow$ Binary**
  - Step-by-Step:
    - **Split integer and fractional parts**: 3 4 • 5 6 2
    - **Map** each to 3-bit binary using the table:
    - 3 → 011 | 4 → 100 | 5 → 101 | 6 → 110 | 2 → 010
    - **Combine** results (preserve the octal point): 011 100 • 101 110 010 → 011100.101110010_2
  - Visual Guide:

Octal Part	3	4	•	5	6	2
Binary	011	100	•	101	110	010

  - Result:
    - $34.562_8 = 011100.101110010_2$
    - (Leading zero can be omitted: 11100.10111001_2)

Binary-to-Octal Conversion

- **Conversion Rules:**
  - Group binary digits into **sets of 3** (right → left for integers, left → right for fractions)
  - Add leading/trailing zeros if needed to complete groups
  - Convert each 3-bit group to octal using reference table
- **Example: Simple Conversion**
  - **Convert 100111010_2 → Octal**
  - **Steps**:
    - **Group from right (LSB): 100 | 111 | 010**
    - **Convert each group:**
      - **100 → 4**
      - **111 → 7**
      - **010 → 2**
  - **Result**:
    - **$100111010_2 = 472_8$**

Binary-to-Octal Conversion

- **Padding with Leading/Trailing Zeros**
- **Key Rule:**
  - For integer parts (left of point):
    - Add zeros to the left of the MSB (most significant bit)
  - For fractional parts (right of point):
    - Add zeros to the right of the LSB (least significant bit)
- **Visual Examples:**
  - **Integer Padding (Left Side)**
    - **Original:** 1 1 0 1 0 1 1 0 **← 8 bits (not divisible by 3)**
    - **Padded:** 0 1 1 | 0 1 0 | 1 1 0 **← Added 1 leading zero**
    - 3 2 6 **← Octal equivalent**
  - **Fractional Padding (Right Side)**
    - **Original:** .1 0 1 1 **← 4 bits (needs 2 more for 3-bit groups)**
    - **Padded:** .1 0 1 | 1 0 0 **← Added 2 trailing zeros**
    - 5 4 **← Octal equivalent**

- **Step-by-Step Padding Guide**

Binary Number	Action	Padded Version
11010110	Add 1 leading zero	011 010 110
101.01101	Integer: Leave as-is + 1 trailing zero (fraction)	101 • 011 010
.1101	Add 2 trailing zeros	.110 100

Binary-to-Octal Conversion

- **Example: With Padding Zeros**

- **Convert 11010110_2 → Octal**
  - **Steps:**
    - Add **1 leading zero** to make complete groups:
      - 011 | 010 | 110
    - Convert each group:
      - 011 → 3
      - 010 → 2
      - 110 → 6
  - **Visual Guide:**

    Original: 1 1 0 1 0 1 1 0

    Padded: 0 1 1 | 0 1 0 | 1 1 0

    3 2 6
  - **Result:**
    - $11010110_2 = 326_8$

- **Example: Fractional Binary**

- **Convert 1011.01101_2 → Octal**
  - **Steps:**
    - Integer part (pad left):
      - Original: 1 0 1 1 → Needs 2 leading zeros
      - Padded: 001 | 011
      - Convert: 1 3
    - Fraction part (pad right):
      - Original: .01101 → Needs 1 trailing zero
      - Padded: .011 | 010
      - Convert: 3 2
  - **Visual Guide:**
    - Original: 1 0 1 1 • 0 1 1 0 1
    - Padded: 0 0 1 | 0 1 1 • 0 1 1 | 0 1 0
    - 1 3 • 3 2
  - **Result:**
    - $1011.01101_2 = 13.32_8$

Hexadecimal Number System (Base-16)

- Key Advantages:
  - Compact representation of binary data (1 hex digit = 4 binary bits)
  - Direct mapping to memory addresses and machine code
  - Universal standard in web design (CSS colors), assembly language, and debugging
- Why Hexadecimal
  - Compact Representation:
    - **1 byte (8 bits) = 2 hex digits**
      - **Example: $11010011_2 = D3_{16}$**
    - **Easy Binary Conversion:**
      - **Binary: 1101 0011 → Split into nibbles**
      - **Hex: D 3 → $D3_{16}$**

Hex	Decimal	Binary
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

Hexadecimal-to-Decimal Conversion

- **Key Principle:**
  - Every hex digit's value depends on its positional weight (powers of 16)
- **Method:**
  - Multiply each hex digit by its positional weight (powers of 16)
- **Formula:**
  - Decimal = $d_n \times 16^n + \dots + d_1 \times 16^1 + d_0 \times 16^0$
  - (Where d = hex digit value, n = position from right starting at 0)
- **Example 1:**
  - Convert 356_{16} to decimal
  - 3 5 6
  - | | └ $6 \times 16^0 = 6 \times 1 = 6$
  - | └── $5 \times 16^1 = 5 \times 16 = 80$
  - └──── $3 \times 16^2 = 3 \times 256 = 768$
  - **Total = 768 + 80 + 6** = 854_{10}

- **Example 2:**
  - Convert $2AF_{16}$ to decimal
  - **2 A F**
  - | | └ **$15 \times 16^0 = 15 \times 1 = 15$**
  - | └── **$10 \times 16^1 = 10 \times 16 = 160$**
  - └──── **$2 \times 16^2 = 2 \times 256 = 512$**
  - **Total = 512 + 160 + 15 =** 687_{10}

TRY YOUSELF! – CLASS ACTIVITY-

Convert $2AC_{16}$ to decimal

Decimal-to-Hexadecimal Conversion

- **Steps:**
  - Divide by 16, track remainders (convert ≥10 to A-F).
  - Read remainders bottom-up.
- **Example 1:**
  - Convert 423_{10} to hex

Division	Quotient	Remainder	Hex Digit
423 ÷ 16	26	7	7
26 ÷ 16	1	10	A
1 ÷ 16	0	1	1

  - Result: Read ↑ → $1A7_{16}$

- **Example 2:**
  - Convert 214_{10} to hex
  - 214 ÷ 16 = 13 R6 (6)
  - 13 ÷ 16 = 0 R13 (D)
  - Result: ↑ Read up → $D6_{16}$

TRY YOUSELF! – CLASS ACTIVITY-

Convert 1024_{10} to Hexadecimal

Hexadecimal-to-Binary Conversion

- **Direct 4-Bit Grouping Method**
- **Key Rule:**
  - ‣ Each hexadecimal digit converts to exactly 4 binary bits (nibble)
  - ‣ Works for both integers and fractions
- **Conversion Steps:**
  - Separate each hex digit
  - Convert to 4-bit binary using the reference table
  - Combine all binary groups

- **Example: Convert $9F2_{16}$ to Binary**
  - 9 F 2
  - ↓ ↓ ↓
  - 1001 1111 0010
  - Result: $9F2_{16} = 100111110010_2$

- **Example: Convert $A3.C5_{16}$ to Binary**
  - A 3 . C 5
  - ↓ ↓ ↓ ↓
  - 1010 0011 . 1100 0101
  - Result: $A3.C5_{16} = 10100011.11000101_2$

Binary-to-Hexadecimal Conversion

- 4-Bit Grouping Method (Reverse of Hex-to-Binary)
- Steps:
  - Group binary digits into 4-bit nibbles (start from right for integers, left for fractions)
  - Pad with leading/trailing zeros if needed
  - Convert each group to its hex equivalent

- **Example: Convert 101110100110_2 to hex**
  - Binary: 1011 1010 0110
  - ↓ ↓ ↓
  - Hex: B A 6
  - **Result: $101110100110_2 = BA6_{16}$**

Hexadecimal-to-Octal Conversion

- Two-Step Method:
  - Hex → Binary → Octal
- Steps:
  - Convert each hex digit to 4-bit binary
  - Regroup binary into 3-bit chunks (for octal)
  - Pad with zeros if needed
  - Convert to octal digits

- **Example: Convert $2F_{16}$ to octal**
  - **Step 1: Hex → Binary**
  - 2 F → 0010 1111
  - **Step 2: Regroup for octal**
  - 00 101 111 → Pad → 00 101 111 (drop unnecessary leading zero)
  - **Step 3: Binary → Octal**
  - 101=5, 111=7
  - **Result: $2F_{16} = 57_8$**

Hexadecimal-to-Octal Conversion

- **Example: Convert $9F_{16}$ to octal**
  - Step 1: Hex → Binary (4-bit groups)
    - 9 F → 1001 1111
  - Step 2: Binary → Octal (3-bit groups, pad leading zeros)
    - Original: 1001 1111
    - Pad left: 010 011 111
    - Regroup: 010 011 111
  - Step 3: Binary → Octal
    - 010 = 2, 011 = 3, 111 = 7
  - **Result: 237_8**

Review Questions: Number System Conversions

1. Binary Conversion
   – Find the binary equivalent of decimal 363. Then, convert the resulting binary number to octal.
2. Maximum 8-Bit Value
   – What is the largest decimal number that can be represented using 8 bits in binary?
3. Binary-to-Decimal
   – Convert the binary number 1101011_2 to its decimal equivalent.
4. Binary Counting Sequence
   – Determine the next binary number in the sequence after 10111_2.
5. MSB Weight Calculation
   – Calculate the positional weight (decimal value) of the Most Significant Bit (MSB) in a 16-bit binary number.
6. Octal-to-Decimal
   – Convert the octal number 614_8 to decimal.
7. Decimal-to-Octal
   – Convert the decimal number 146_{10} to octal.
8. Hexadecimal-to-Decimal
   – Convert the hexadecimal number $24CE_{16}$ to decimal.
9. Decimal-to-Hex-to-Binary
   – First, convert the decimal number 3117_{10} to hexadecimal. Then, convert the resulting hexadecimal number to binary.
10. Binary-to-Decimal (Fractional)
   – Solve for x in the equation: $1011.11_2 = x_{10}$
11. Octal-to-Decimal (Fractional)
   – Solve for x in the equation: $174.3_8 = x_{10}$
12. Decimal-to-Binary (Fractional)
   – Solve for x in the equation: $10949.8125_{10} = x_2$
13. Hexadecimal-to-Binary (Fractional)
   – Solve for x in the equation: $2C6B.F2_{16} = x_2$

Binary Arithmetic Fundamentals

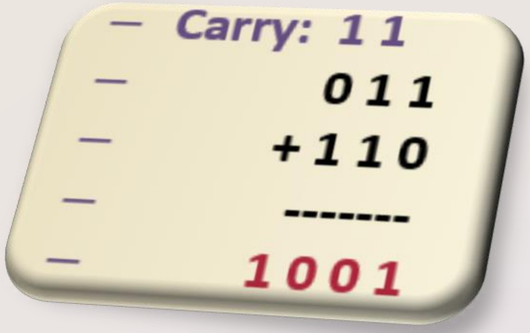
- Key Definitions
  - Binary Addition
    - Definition: Bitwise operation following four rules with possible carry propagation.
    - Carry: Overflow when the sum of bits exceeds 1 (similar to decimal "carrying the 1").
  - Binary Subtraction (Direct Method)
    - Definition: Bitwise operation requiring borrowing from higher positions when subtracting a larger digit from a smaller one.
  - 2's Complement Method
    - 1's Complement: Inverting all bits of a number (1→0, 0→1).
    - 2's Complement: Adding 1 to the 1's complement to represent negative numbers.
    - Purpose: Simplifies subtraction by converting it to addition (used in CPUs).

Binary Addition

- **Rules Table**

A + B	Sum	Carry
0 + 0	0	0
0 + 1	1	0
1 + 1	0	1
1+1+1	1	1

- **Example: 011_2 (3) + 110_2 (6)**
  - 011
  - + 110
  - -----
  - 1001 (9)
  - **Step-by-Step:**
    - LSB (Right): 1 + 0 = 1
    - Middle: 1 + 1 = 0 (carry 1)
    - MSB (Left): 0 + 1 + 1 (carry) = 0 (carry 1)
    - Final Carry: Leading 1 → 1001_2

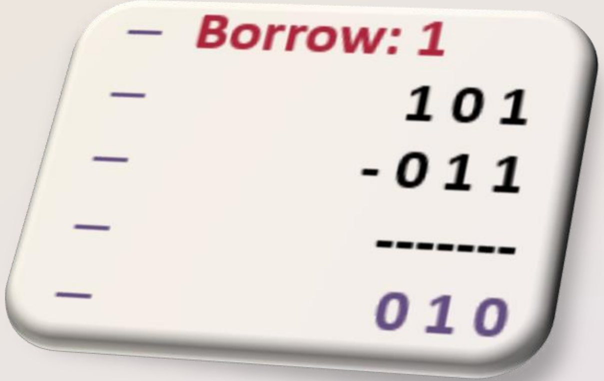


Binary Subtraction (Direct Method)

- **Rules Table**

A – B	Result	Borrow
0 – 0	0	0
1 – 0	1	0
1 – 1	0	0
0 – 1	1	1

- **Example: 101_2 (5) – 011_2 (3)**
  - 101
  - \- 011
  - -----
  - 010 (2)
  - Step-by-Step:
    - LSB: 1 – 1 = 0
    - Middle: 0 – 1 → Borrow 1 → 10 – 1 = 1
    - MSB: (0 after borrow) – 0 = 0
    - Final result: 010



1's and 2's Complement Conversion

- **Definitions**
  - **1's Complement:**
    - Definition: Flip all bits of a binary number (0→1, 1→0).
    - Example: 0101 → 1010.
  - **2's Complement:**
    - Definition: 1's complement + 1. Represents negative numbers.
    - Example: 0101 → 1010 (1's) → 1011 (2's).
- **Key Advantages of 2's Complement:**
  - Simplifies hardware design (uses same circuits for + and -)
  - Eliminates separate subtraction logic
  - Represents negative numbers in binary
  - The leftmost bit (MSB) is the sign bit (0 = positive, 1 = negative).

- **Example: Convert 1011 0101_2 (Original)**

Step	Binary Value	Visual Representation
Original Number	1011 0101	1 0 1 1 0 1 0 1
1's Complement	0100 1010	0 1 0 0 1 0 1 0
2's Complement	0100 1010 + 1 = 0100 1011	0 1 0 0 1 0 1 1

1’s and 2’s Complement Conversion

- **Why Are They Used?**

Feature	1’s Complement	2’s Complement
Negative Rep.	Two zeros (0000 and 1111)	Single zero (0000)
Hardware	Requires extra step for subtraction	Simplifies arithmetic (no carry logic)
Overflow	Detected by end-around carry	Detected by sign-bit mismatch

1's Complement Examples

- **Example 1: Representing Negative Numbers**
  - Number: -3 in 4-bit binary
    - Step 1: Write +3 → 0011
    - Step 2: Flip all bits → 1100 (This is -3 in 1's complement).

------------ ------------ ----------- ----------

------------ ------------ ----------- ----------

- **Example 2: Subtraction Using 1's Complement**
  - Calculate: 5 – 3
    - Step 1: Represent -3 → 1100 (1's complement of 0011).
    - Step 2: Add to 5 (0101):
  - 0101 (5)
  - \+ 1100 (-3 in 1's)
  - \-------
  - 10001
- **Step 3: End-around carry → Add the overflow 1 back:**
  - 0001 (from 10001)
  - \+ 1 (carry)
  - \-------
  - 0010 (2) → Correct!
- **Problem: Without the extra carry step, you'd get 0001 (wrong!).**

2's Complement Examples

- **Example 1: Representing Negative Numbers**
  - Number: -3 in 4-bit binary
    - Step 1: Write +3 → 0011
    - Step 2: Flip all bits → 1100 (This is -3 in 1's complement).
    - Step 3: Add 1 → 1101 (This is -3 in 2's complement).

  ------------ ------------ ----------- ----------

  ------------ ------------ ----------- ----------

- **Example 2: Subtraction Using 2's Complement**
  - Calculate: 5 – 3
    - Step 1: Represent -3 → 1101 (2's complement of 0011).
    - Step 2: Add to 5 (0101):
  - 0101 (5)
  - + 1101 (-3 in 2's)
  - -------
  - 10010 → Discard overflow → `0010` (2) → Correct!
- **No extra steps needed**

Key Differences in Examples

Operation	1's Complement	2's Complement
Represent -3	1100 (flipped bits)	1101 (flipped bits + 1)
Calculate 5-3	Needs end-around carry (10001 → 0010)	Simple addition (10010 → discard → 0010)

Homework!

1. Convert -6 to 8-bit 2's complement.
2. Subtract 7 - 4 using 2's complement.

THANK YOU ☺