Computer Architecture

Module 1: Data Representation and Basic Computer Arithmetic

Comprehensive Study Notes

1. Number Systems

Number systems are fundamental to understanding how computers represent and process data.

1.1 Types of Number Systems

Number System	Base	Digits Used	Example
Binary	2	0, 1	1011_2
Octal	8	0–7	17_8
Decimal	10	0–9	15_{10}
Hexadecimal	16	0–9, A–F	$1F_{16}$

1.2 Number System Conversions

Binary to Decimal:

$$1011_2 = 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$$
$$= 8 + 0 + 2 + 1 = 11_{10}$$

Decimal to Binary (Repeated Division):

25_{10} = ?
25 ÷ 2 = 12 remainder 1
12 ÷ 2 = 6 remainder 0
6 ÷ 2 = 3 remainder 0
3 ÷ 2 = 1 remainder 1
1 ÷ 2 = 0 remainder 1
Reading upward: 11001_2

Binary to Octal (Group 3 bits):

$1011011_2 = 001\ 011\ 011 = 133_8$

Binary to Hexadecimal (Group 4 bits):

$1011011_2 = 0101\ 1011 = 5B_{16}$

2. Complements

Complements simplify subtraction in digital systems.

2.1 Complements in Base-r System

r's Complement:

- Formula: $r^n - N$
- Where n = number of digits, N = given number

Example (2's Complement):

8-bit number: $12_{10} = 00001100_2$
2's complement = $256 - 12 = 244_{10} = 11110100_2$

r−1's Complement:

- Formula: $(r^n - 1) - N$

Example (1's Complement):

8-bit number: $12_{10} = 00001100_2$
1's complement = $255 - 12 = 243_{10} = 11110011_2$

2.2 Finding 2's Complement

Method 1: Invert all bits and add 1

Original: 00001100
1's comp: 11110011
Add 1: 11110100 (2's complement)

Method 2: Copy from LSB up to first 1, complement remaining

Original: 01101000
Copy up to first 1: 1000
Complement remaining: 1001
Result: 10011000

2.3 Subtraction Using Complements

Using 2's Complement:

Subtract: $16 - 9$
$16_{10} = 00010000_2$
$9_{10} = 00001001_2$

9's 2's complement = 11110111_2
Add: 00010000 + 11110111 = 100000111
Discard carry: $00000111_2 = 7_{10}$ ✓

3. Fixed-Point Representation

3.1 Sign-Magnitude Representation

- Most significant bit (MSB) = sign bit → 0 = positive, 1 = negative
- Remaining bits = magnitude

Example (8-bit):

+25 = 00011001
−25 = 10011001

Disadvantages:

- Two representations for zero (+0 and −0)
- Complex addition/subtraction

3.2 1's Complement Representation

- Positive numbers: same as sign-magnitude
- Negative numbers: invert all bits of positive number

Example (8-bit):

```
+25 = 00011001
−25 = 11100110
```

Disadvantages:

- Two zeros (+0 = 00000000, −0 = 11111111)
- End-around carry needed

3.3 2's Complement Representation

- Positive numbers: same as sign-magnitude
- Negative numbers: invert bits and add 1

Example (8-bit):

```
+25 = 00011001
−25 = 11100111
```

Advantages:

- Single representation for zero
- Simple addition/subtraction
- No end-around carry

Range for n-bit 2's Complement:

- Minimum: -2^{n-1}
- Maximum: $2^{n-1} - 1$
- For 8-bit: −128 to +127

3.4 Comparison of Representations

Property	Sign-Magnitude	1's Complement	2's Complement
Zero Representation	Two (+0, −0)	Two (+0, −0)	One
Addition Circuit	Complex	Simple with end-around carry	Simplest
Range (8-bit)	−127 to +127	−127 to +127	−128 to +127

4. Floating-Point Representation

4.1 IEEE 754 Standard

Single Precision (32-bit):

Sign (1 bit)	Exponent (8 bits)	Mantissa (23 bits)
S	E	M

Double Precision (64-bit):

Sign (1 bit)	Exponent (11 bits)	Mantissa (52 bits)
S	E	M

4.2 Normalized Representation

The number is represented as:

$$\text{Value} = (-1)^s \times (1.M) \times 2^{(E - \text{bias})}$$

Bias Values:

- Single precision: 127
- Double precision: 1023

4.3 Converting Decimal to IEEE 754

Example: Convert 25.625 to IEEE 754 Single Precision

Step 1: Convert to binary
$25 = 11001_2$
$0.625 = 0.101_2$
$25.625 = 11001.101_2$

Step 2: Normalize
$11001.101 = 1.1001101 \times 2^4$

Step 3: Determine components
Sign: Positive → 0
Exponent: $4 + 127 = 131 = 10000011_2$
Mantissa: 10011010000000000000000 (23 bits)

Step 4: Combine
0 10000011 10011010000000000000000

4.4 Special Values

Condition	Exponent	Mantissa	Value
Zero	All 0	All 0	0
Denormalized	All 0	Non-zero	±0.M × 2^(E_min)
Infinity	All 1	All 0	±∞
NaN	All 1	Non-zero	Not a Number

4.5 Floating-Point Arithmetic Operations

Addition/Subtraction:

- Align exponents (shift smaller mantissa right)
- Add/subtract mantissas
- Normalize result
- Round result

Multiplication:

- Add exponents (subtract bias)
- Multiply mantissas

- Normalize and round

Division:

- Subtract exponents (add bias)
- Divide mantissas
- Normalize and round

5. Arithmetic Operations

5.1 Binary Addition

Rules:

```
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 (carry 1)
1 + 1 + 1 = 1 (carry 1)
```

Example:

```
 1011
+ 0110
 ----
 10001
```

5.2 Binary Subtraction

Using 2's complement method:

- Find 2's complement of subtrahend
- Add to minuend
- Discard final carry

```
Subtract: 13 − 7
13 = 1101₂
7  = 0111₂
2's complement of 7: 1001₂
Add: 1101 + 1001 = 10110
Discard carry: 0110₂ = 6₁₀ ✓
```

5.3 Binary Multiplication

Rules:

```
0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1
```

Example:

```
  1101  (13)
× 1011  (11)
--------
  1101
 1101
0000
1101
--------
10001111  (143)
```

5.4 Binary Division

```
1101 (13) ÷ 0110 (6)
13 ÷ 6 = 2 remainder 1
2 = 0010₂
1 = 0001₂
```

6. Alphanumeric Representation

6.1 ASCII (American Standard Code for Information Interchange)

Character	Decimal	Binary
'A'	65	1000001
'B'	66	1000010
'a'	97	1100001
'0'	48	0110000
'1'	49	0110001
SPACE	32	0100000

6.2 Extended ASCII (8-bit)

- Adds 128 additional characters
- Latin characters, symbols, accented letters

6.3 Unicode

- UTF-8: Variable length (1–4 bytes)
- UTF-16: 2 or 4 bytes
- UTF-32: 4 bytes

Example:

```
'你好' in UTF-8:
你 = E4 BD A0
好 = E5 A5 BD
```

7. Common Errors in Computing

7.1 Overflow

When result exceeds maximum representable value.

```
8-bit 2's complement: 127 + 1 = −128  (overflow!)
01111111 + 00000001 = 10000000
```

7.2 Underflow

When result is too small to be represented.

7.3 Rounding Errors

Floating-point arithmetic has inherent rounding errors.

```
0.1 + 0.2 ≠ 0.3 in binary
(Approximately 0.30000000000000004)
```

7.4 Precision Loss

Loss of Precision:

```
Large + Small = Large  (if small << large)
Example: 1,000,000 + 0.000001 = 1,000,000  (in single precision)
```

8. Quick Reference Summary

8.1 Key Formulas

Concept	Formula
r's Complement	$r^n - N$
r−1's Complement	$(r^n - 1) - N$
2's Complement Range	-2^{n-1} to $2^{n-1} - 1$
IEEE 754 Value	$(-1)^s \times (1.M) \times 2^{(E-bias)}$

8.2 Common Conversions

Convert From	Convert To	Method
Decimal	Binary	Repeated division
Binary	Decimal	Sum of powers
Binary	Octal	Group 3 bits
Binary	Hex	Group 4 bits

8.3 Number System Comparison

System	Base	Applications
Binary	2	Computer hardware, logic gates
Octal	8	Unix file permissions, assemblers
Decimal	10	Human interface
Hex	16	Memory addresses, debugging

9. Practice Problems

1. Convert 37_{10} to binary, octal, and hex
2. Find 2's complement of 01011010_2
3. Add: $10110110_2 + 01101011_2$
4. Represent −27 in 8-bit 2's complement
5. Convert 13.625 to IEEE 754 single precision

10. Key Takeaways

✅ Binary is the fundamental number system for computers
✅ 2's complement is the most common signed representation
✅ IEEE 754 is the standard for floating-point arithmetic
✅ Fixed-point has limited range but is precise
✅ Floating-point covers large range but with rounding errors
✅ Different number systems serve different purposes in computing