

CSE 210: Computer Architecture

Lecture 6: Number Systems

Stephen Checkoway

Oberlin College

Oct. 15, 2021

Slides from Cynthia Taylor

Announcements

- Problem Set 1 due by the end of today
 - Submit via Gradescope
- Problem Set 2 due a week from today
- Lab 1 due a week from Sunday
 - On website, submit via GitHub
- Office hours 13:30 – 14:30 today

Positional Notation

- The meaning of a digit depends on its position in a number.
- A number, written as the sequence of digits $d_n d_{n-1} \dots d_2 d_1 d_0$ in base b represents the value

$$d_n * b^n + d_{n-1} * b^{n-1} + \dots + d_2 * b^2 + d_1 * b^1 + d_0 * b^0$$

Binary to Decimal

- We have $b = 2$

$$\begin{aligned} 10110_2 &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ &= 16 + 4 + 2 \\ &= 22 \end{aligned}$$

Decimal to binary

- Convert 115 to binary

- We know

$$\begin{aligned} 115 &= d_n \cdot 2^n + \dots + d_1 \cdot 2^1 + d_0 \cdot 2^0 \\ &= 2(d_n \cdot 2^{n-1} + \dots + d_1) + d_0 \end{aligned}$$

- 115 is odd and $2(d_n \cdot 2^{n-1} + \dots + d_1)$ is even so $d_0 = 1$

- Subtract 1, divide by 2, and repeat

$$\begin{aligned} 57 &= d_n \cdot 2^{n-1} + \dots + d_2 \cdot 2^1 + d_1 \\ &= 2(d_n \cdot 2^{n-2} + \dots + d_2) + d_1 \end{aligned}$$

Decimal to Binary

- Repeatedly divide by 2, recording the remainders
- The remainders form the binary digits of the number from the least significant to the most significant
- Converting 25 to binary

$$34_{10} = ?_2$$

- A. 010001
- B. 010010
- C. 100010
- D. 111110
- E. None of the above

Hexadecimal to binary

- Each hex digit corresponds directly to four binary digits
- $35AE_{16} =$

$$23C_{16} = ?_2$$

- A. 0010 0000 1100
- B. 0010 1111 0010
- C. 0010 0011 1100
- D. 1000 1101 1000
- E. None of the above

If every hex digit corresponds to 4 binary digits, how many binary digits does an octal digit correspond to?

- A. 2
- B. 3
- C. 4
- D. 5

Addition

- Use the same place-by-place algorithm that you use for decimal numbers, but do the arithmetic in the appropriate base

$$2A5C_{16} + 38BE_{16} = ?$$

A. 586A

B. 631A

C. 6986

D. None of the above

How We Store Numbers

- Binary numbers in memory are stored using a finite, fixed number of bits (typically 8, 16, 32, or 64)
 - 8 bits = byte
- Pad extra digits with leading 0s
- A byte representing $4_{10} = 00000100$

A byte (8 bits) can store positive values from 0 up to

A. 127

B. 128

C. 255

D. 256

E. None of the above

Java

- A `byte` is 8 bits
- A `short` is 16 bits
- An `int` is 32 bits
- A `long` is 64 bits

In C, an int is

A. 8 bits

D. It depends

B. 16 bits

E. None of the above

C. 32 bits

C specifies a *minimum size* for types

- chars are 1 byte and must be at least 8 bits
- shorts and ints must be at least 16 bits
- longs are at least 32 bits
- long longs are at least 64 bits
- sizeof(type) tells us how many bytes type is
- $1 = \text{sizeof}(\text{char}) \leq \text{sizeof}(\text{short}) \leq \text{sizeof}(\text{int}) \leq \text{sizeof}(\text{long}) \leq \text{sizeof}(\text{long long})$

So how do I know?

- Use `sizeof(int)` to check
- Or use C99 types like `int16_t` or `int32_t`

But how do we indicate a negative number?

- Sign and magnitude
- Ones' Complement
- Two's Complement

Short aside

- ones' complement involves taking each bit and taking the complement with respect to 1; there are many bits so many complements with respect to 1 hence "ones' complement"
- two's complement involves taking a complement with respect to a single power of 2, not bit-by-bit, hence "two's complement"
- Yes. It *is* confusing. No. No one remembers this.

Sign and Magnitude

- Have a separate bit for sign
- Set it to 0 for positive, and 1 for negative
- Can represent from -127 to 127 in 8 bits
- With n bits, can represent $-(2^{n-1} - 1)$ to $2^{n-1} - 1$

Addition and subtraction are a hassle

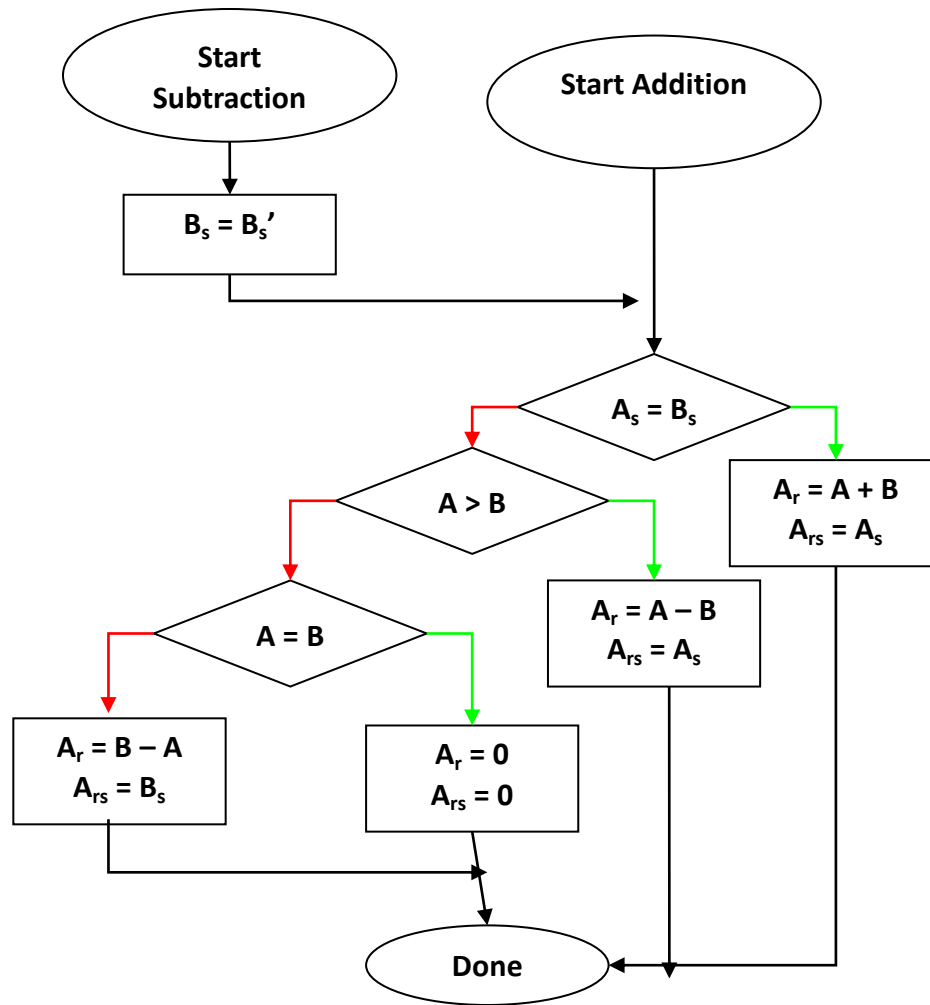


Diagram from Marek Andrzej Perkowski

A byte representing -6_{10} in Sign and Magnitude
(with leftmost sign bit) is

A. 0000 0111

E. None of the above

B. 1000 0110

C. 1000 0111

D. 1111 1110

Which is NOT a drawback of Sign and Magnitude?

- A. There are two zeros
- B. Unclear where to put the sign bit
- C. Complicated arithmetic
- D. Difficult to convert numbers to negative representation
- E. None of the above

Ones' Complement

- To make a number negative, just flip all its bits!
- For n bits, the unsigned version of $-x = 2^n - x - 1$

Reading

- Next lecture: Negatives in binary
 - Section 2.4
- Problem Set 1 due today
- Problem Set 2 due a week from today
- Lab 1 due a week from Sunday