Number Systems

Class 39

Positional Notation

- for counting quantities from 0 through 9, we use single digits
- because most of us have 10 fingers

- for values larger than 9, we use a positional notation
- the number 7305 really means:

$$7 \times 10^3 + 3 \times 10^2 + 0 \times 10^1 + 5 \times 10^0$$

- we call this the decimal number system because
  - it uses ten digits (0 – 9)
  - the coefficients are multiplied by powers of 10
- when necessary to disambiguate the radix of 10, we write

$$7305_{10}$$

Binary Numbers

- we can use any positive integer for the radix
- in computer science, because computers typically use only two values, we use binary numbers, radix 2

$$
\begin{aligned}
1101_2 &= 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
&= 13_{10}
\end{aligned}
$$

- 1101_2 and 13_{10} are the exact same value
- they are simply expressed in two different notation systems

Binary and Decimal

- binary numbers are essential for working efficiently with computers
- but:
  - humans can't deal with typical binary numbers

$$11110101011011111010101010101101$$

  - there is no obvious correlation between binary and decimal

$$1101_2 = 13_{10}$$

- hexadecimal to the rescue

Bytes

- eight bits is one byte
- one-half of a byte, four bits, is a nibble
- there's nothing magic about a byte, it's just convenient
- nibbles make it easier for humans to see a byte's value

$$11010110 \qquad \text{vs} \qquad 1101\,0101$$

- a nibble is about the largest number of bits you can comfortably see

Hexadecimal

- hexadecimal numbers are base-16 numbers
- this requires 16 different digits
- but only 10 digits exist in the Hindu-Arabic system
- so to represent hexadecimal numbers, we use the ten decimal digits 0 – 9
- these have the same values in base-10 and base-16
- plus the six letters a – f, which have the decimal values 10 – 15 respectively
- thus we have

$$\begin{aligned} 7b05 &= 7 \times 16^3 + b \times 16^2 + 0 \times 16^1 + 5 \times 16^0 \\ &= 7 \times 4096_{10} + 11_{10} \times 256_{10} + 0 \times 16_{10} + 5 \times 1 \\ &= 28672_{10} + 2816_{10} + 0 + 5 \\ &= 31493_{10} \end{aligned}$$

- again, there's no obvious correlation between hexadecimal and decimal

Binary and Hexadecimal

- the reason we care about hexadecimal is because of nibbles
- since one nibble represents one of 16 values, one nibble is exactly one hexadecimal digit

$$1101010100101011_2 = c52b_{16}$$

- thus a byte, 8 bits, is exactly 2 hex digits

Binary $\leftrightarrow$ Hex Conversion

		$2^0 = 1$
		$2^1 = 2$
$0000 = 0$	$1000 = 8$	$2^2 = 4$
$0001 = 1$	$1001 = 9$	$2^3 = 8$
$0010 = 2$	$1010 = \text{a}\ (10)$	$2^4 = 16$
$0011 = 3$	$1011 = \text{b}\ (11)$	$2^5 = 32$
$0100 = 4$	$1100 = \text{c}\ (12)$	$2^6 = 64$
$0101 = 5$	$1101 = \text{d}\ (13)$	$2^7 = 128$
$0110 = 6$	$1110 = \text{e}\ (14)$	$2^8 = 256$
$0111 = 7$	$1111 = \text{f}\ (15)$	$2^9 = 512$
		$2^{10} = 1024$

Conversion Decimal $\rightarrow$ Binary

- earlier slides showed how to convert a base-2 or a base-16 representation to decimal
- what about a conversion in the other direction? how to convert a value in decimal notation into binary notation?
- this is done by a series of subtractions: each power of two either contributes to a value or does not
- must know the powers of 2
- it's exactly like making change with coins

example: convert 1304 to binary

Conversion Decimal $\rightarrow$ Binary

- earlier slides showed how to convert a base-2 or a base-16 representation to decimal
- what about a conversion in the other direction? how to convert a value in decimal notation into binary notation?
- this is done by a series of subtractions: each power of two either contributes to a value or does not
- must know the powers of 2
- it's exactly like making change with coins

example: convert 1304 to binary

$$1304 = 101\,0001\,1000_2$$

- in math, we use the subscript 2 to indicate base-2
- in C++, we use the 0b prefix: 0b10100011000

Conversion Decimal $\rightarrow$ Hexadecimal

- the easiest way is to convert decimal $\rightarrow$ binary $\rightarrow$ hexadecimal

example: convert 1304 to hexadecimal

Conversion Decimal $\rightarrow$ Hexadecimal

- the easiest way is to convert decimal $\rightarrow$ binary $\rightarrow$ hexadecimal

example: convert 1304 to hexadecimal

$$\begin{aligned} 1304 &= 0\text{b}10100011000 \\ &= 0\text{x}518 \end{aligned}$$

Hex Arithmetic

- all pointers are expressed in hex notation
- in the previous lab, we had to subtract pointers to find out how many characters were in a C-string
- in the next lab, we'll have to do this again
- we need to be able to add and subtract hex values
- it's just like normal addition and subtraction except
  - when we carry, we carry 16, not 10
  - when we borrow, we borrow 16, not 10

Hex Arithmetic

example: add 0x518 + 0xe9

Hex Arithmetic

example: add 0x518 $+$ 0xe9

example: subtract 0x4a6 $-$ 0x1bf

Hex Arithmetic

example: add 0x518 + 0xe9

example: subtract 0x4a6 − 0x1bf

- when a C++ program prints an address, it is typically a very large number
- example show_struct_size program
- but usually any arithmetic will be with close-together values
- so we can ignore most of the high-order digits