CPSC 2500 Computer Organization
Unit 1: Data Representation

Recommended Reading:
From Unit 0 (Lecture 0.1): Ch 1.1, 1.3 (Ch. 1.2 and 1.4 are optional)
Lecture 1.1: Appendix A.1 – A.3
Lecture 1.2: Appendix A.4 – A.5 (focus on two's complement numbers)
Lecture 1.3: Appendix B

Lecture 1.1 Introduction to Data Representation

Binary Numbers

Powers of two: $2^0 = 1$ $\quad 2^5 = 32 \quad 2^{10} = 1{,}024$

$2^1 = 2 \quad 2^6 = 64$

$2^2 = 4 \quad 2^7 = 128$

$2^3 = 8 \quad 2^8 = 256$

$2^4 = 16 \quad 2^9 = 512$

The decimal number 947 in powers of 10 can be expressed as:

$9 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0$

$= 900 + 40 + 7$

$= 947$

Convert 101101_2 into base 10.

$\underset{2^5}{1}\ \underset{2^4}{0}\ \underset{2^3}{1}\ \underset{2^2}{1}\ \underset{2^1}{0}\ \underset{2^0}{1}$

$= 2^5 + 2^3 + 2^2 + 2^0$

$= 32 + 8 + 4 + 1 = \boxed{45}$

Use the subtraction method to convert 90 into binary.

2^n that is closest to 90 but not greater than.

$2^6 = 64 ; 90 - 64 = 26$

$2^4 = 16 ; 26 - 16 = 10$

$2^3 = 8 ; 10 - 8 = 2$

$2^1 = 2 ; 2 - 2 = 0$ ✓

$\underset{2^6}{1}\ \underset{2^5}{0}\ \underset{2^4}{1}\ \underset{2^3}{1}\ \underset{2^2}{0}\ \underset{2^1}{1}\ \underset{2^0}{0}$

$\boxed{1011010}$

Use the division / remainder method to convert 177 into binary.

Continuously divide by two and record the remainder.

k=0 n=177/2 =88 r=1 b=1
k=1 n=88/2=44 r=0 b=01
k=2 n=44/2=22 r=0 b=001
k=3 n=22/2=11 r=0 b=0001
k=4 n=11/2=5 r=1 b=10001
k=5 n=5/2=2 r=1 b=110001
k=6 n=2/2=1 r=0 b=0110001
k=7 n=1/2=0 r=1 b=$\boxed{10110001}$

Hexadecimal Numbers

Convert the binary number into hexadecimal.

$\underbrace{0011}_{3}\underbrace{0101}_{5}\underbrace{0001}_{1}\underbrace{1011}_{B}$

$\boxed{0x351B}$

Decimal	4-Bit Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Class Problem

Convert the number 489 into (a) binary and (b) hexadecimal.

k=0 n=489/2=244 r=1 b=1
k=1 n=244/2=122 r=0 b=01
k=2 n=122/2=61 r=0 b=001
k=3 n=61/2=30 r=1 b=1001
k=4 n=30/2=15 r=0 b=01001
k=5 n=15/2=7 r=1 b=101001
k=6 n=7/2=3 r=1 b=1101001
k=7 n=3/2=1 r=1 b=11101001
k=8 n=1/2=0 r=1 b=$\boxed{111101001}$
binary

$\underbrace{1}_{1}\underbrace{1110}_{E}\underbrace{1001}_{9}$

$\boxed{0x1E9}$
hexadecimal

Adding Binary Numbers

Works in the same way as base-10 addition. Math facts:

0 + 0 = 0 0 + 1 = 1

1 + 0 = 1 1 + 1 = 10

Find the sum of two bytes containing 139 and 46 using binary addition.

```
     11 1
  10001011
+ 00101110
----------
  10111001
```

Now find the sum of two bytes containing 214 and 93 using binary addition.

```
  1  111
  11010110
+ 01011101
----------
 100110011
```

Sound Representation

How much memory is needed to store a 30-second audio file (uncompressed)?

$$\left(\frac{30\text{ secs}}{\text{file}}\right)\left(\frac{44{,}100\text{ samples}}{\text{second}}\right)\left(\frac{2\text{ bytes}}{\text{sample}}\right) = 2.52\ MB$$

Lecture 1.2 Two's Complement Integers

Fixed Length Integers and Overflow

C++ Unsigned Integer Sizes

Data type	*Size (bytes)*	*Unsigned range (0 to …)*
`char`	1	255
`short`	2	65,535
`int, long`	4	4,294,967,295
`long long`	8	$\approx 1.84 \times 10^{19}$

Class Problem

Find the sums of these binary numbers. Assume a one-byte limit and indicate if overflow occurs.

```
   10100110          01011100
+  01101100       +  10001111
-----------       -----------
 1 00010010          11101011
```

Overflow

No overflow

Two's Complement Numbers

Two's complement numbers arrange negative and positive numbers in an ordered number line.

-4	1111 1100
-3	1111 1101
-2	1111 1110
-1	1111 1111
0	0000 0000
1	0000 0001
2	0000 0010
3	0000 0011
4	0000 0100

There will be different rules for overflow w/ negative #'s

This creates new endpoints. For one byte the endpoints are:

- bottom (most negative): 1000 0000 (-128)
- top (most positive): 0111 1111 (+127)

In general, if a number has *b* bits, the end points are:

- bottom (most negative): $-(2^{b-1})$
- top (most positive): $2^{b-1}-1$

Why is there one more negative value than positive value?

To handle 0.

0 consumes one of the "positive" bit patterns.

How do you determine if a value is negative?

The left most bit (sign bit).

If 1 ⇒ negative

0 ⇒ nonnegative (0 or positive).

Express -43 in two's complement.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	1	0	1	0	1	1

$$\begin{array}{r} 43 \\ -32 \\ \hline 11 \\ -8 \\ \hline 3 \\ -2 \\ \hline 1 \\ -1 \\ \hline 0 \end{array}$$

Positive 43; 0010 1011

Flip bits: 1101 0100

+1 : 1101 0101 → -43

Express -(-43) in two's complement.

Negative 43: 1101 0101

Flip: 0010 1010

+1 : 0010 1011 → 43

Add the numbers 75 and -39.

```
  1 11   11
  0100 1011          75
+ 1101 1001       + -39
-----------       -----
 10010 0100          36
```

Ignore this

Rule for detecting overflow when adding two's complement numbers: When the "carry in" and the "carry out" of the sign bit (left-most bit) are different, overflow has occurred.

Add the numbers 107 + 46.

```
 1 ← "carry in"
  11 1  111
  0110 1011         107
+ 0010 1110       +  46
-----------       -----
 01001 1001         153
```

"carry out"

153 > 127

Carry in ≠ carry out

Therefore overflow occurs

A = 0x36BE

Two's Complement Overflow Cases

Case 1: Adding a positive and a negative number.

The sign bits must be different (1 and 0):

⇒ Carry in must equal carry out

⇒ overflow is NOT possible

Case 2: Adding two positive numbers.

Sign bits are both zero

Carry out bit will be zero

⇒ When carry in of sign bit is 1, overflow occurs.

It means that the 7 bit addition result does not fit in 7 bits.

Case 3: Adding two negative numbers.

The sign bits are both 1.

The carry out bit will be 1.

⇒ If carry in is zero, overflow occurs

⇒ Most negative numbers have a zero in the bit directly to the right of the sign bit.

⇒ Least negative numbers (near 0) have a one in the bit to the right of the sign bit.

No overflow | overflow

Class Problem

Find the sums of these two's complement binary numbers. Assume a one-byte limit and indicate if overflow occurs.

```
  1111  111          11            111 111
   1001 1001         1011 0100      0010 0111
 + 0110 0111       + 1101 1010    + 0101 1001
 1 0000 0000       1 1000 1110    0 1000 0000
```

No overflow (carry in = carry out)

No overflow

overflow (carry in is 1)

Sign Extension

Convert a byte containing -39 to a 16 bit number.

-39 in 8 bits: 1101 1001

-39 in 16 bits: 1111 1111 1101 1001

Lecture 1.3 Floating-Point Numbers

Floating-Point Number Representation

Computers use a form of scientific notation for floating-point representation. Numbers written in scientific notation have three components:

$$\underline{\ \ }\ 7.638 \times 10^{5} \leftarrow \text{exponent}$$

sign significand

Converting Decimal to Floating-Point

Step 1. Convert the decimal number into a binary number.

Convert 10.625 into a binary number.

$$\frac{1}{b^3}\ \frac{0}{b^2}\ \frac{1}{b^1}\ \frac{0}{b^0}\ .\ \frac{1}{b^{-1}}\ \frac{0}{b^{-2}}\ \frac{1}{b^{-3}}$$

$$10.625 = 1010.101_2$$

$$\begin{array}{r} 10.625 \\ -8 \\ \hline 2.625 \\ -2 \\ \hline 0.625 \\ -0.5 \\ \hline 0.125 \\ -0.125 \\ \hline 0 \end{array}$$

Step 2. Express the floating-point number in scientific notation.

Recall in base-10 scientific notation, the number to the left of the decimal point must be 1-9 (unless the number is zero). Examples:

857.63 = 8.5763×10^{2}

0.00007634 = 7.634×10^{-5}

In binary, the number to the left of the floating point must be a 1 (unless the number is zero). Examples:

110111.01 = 1.1011101×2^{5}

0.011101 = 1.1101×2^{-2}

Express 10.625 as a binary number in scientific notation.

$$10.625 \rightarrow 1010.101$$
$$\downarrow$$
$$\boxed{1.010101 \times 2^{3}}$$

Step 3. Fill in the various fields of the floating-point number appropriately.

Convert 10.625 as an IEEE floating-point number in both binary and hexadecimal. 1.010101×2^3

Exponent: Two's Complement(3): 0000 0011

$+$ 127 $\quad$ + 0111 1111

Excess-127 bias $\quad$ 1000 0010

Sign	Exponent	Fraction
0	10000010	0101010 . . . 0

Binary: 0100 0001 0010 1010 0000 0000 0000 0000

Hexadecimal: 4 1 2 A 0 0 0 0

$\boxed{0x412A0000}$

Class Problem

Convert the IEEE floating-point number 0xC2AC8000 into decimal.

Binary: $\underbrace{1}_{\text{sign}}\ \underbrace{10000101}_{\text{exponent}}\ \underbrace{010\ 1100\ 1000\ 0000\ 0000\ 0000}_{\text{Fraction}}$

Excess-127 bias : 1000 0101

-127 $\quad$ 1000 0001

$\cancel{1}$ 0000 0110 $\Rightarrow 6$

$-1.01011001 \times 2^6 = -1010110.01$

64 32 16 8 4 2 1 0.5 0.25

$-(64+16+4+2+0.25) = \boxed{-86.25}$