CS2100

TUTORIAL #1

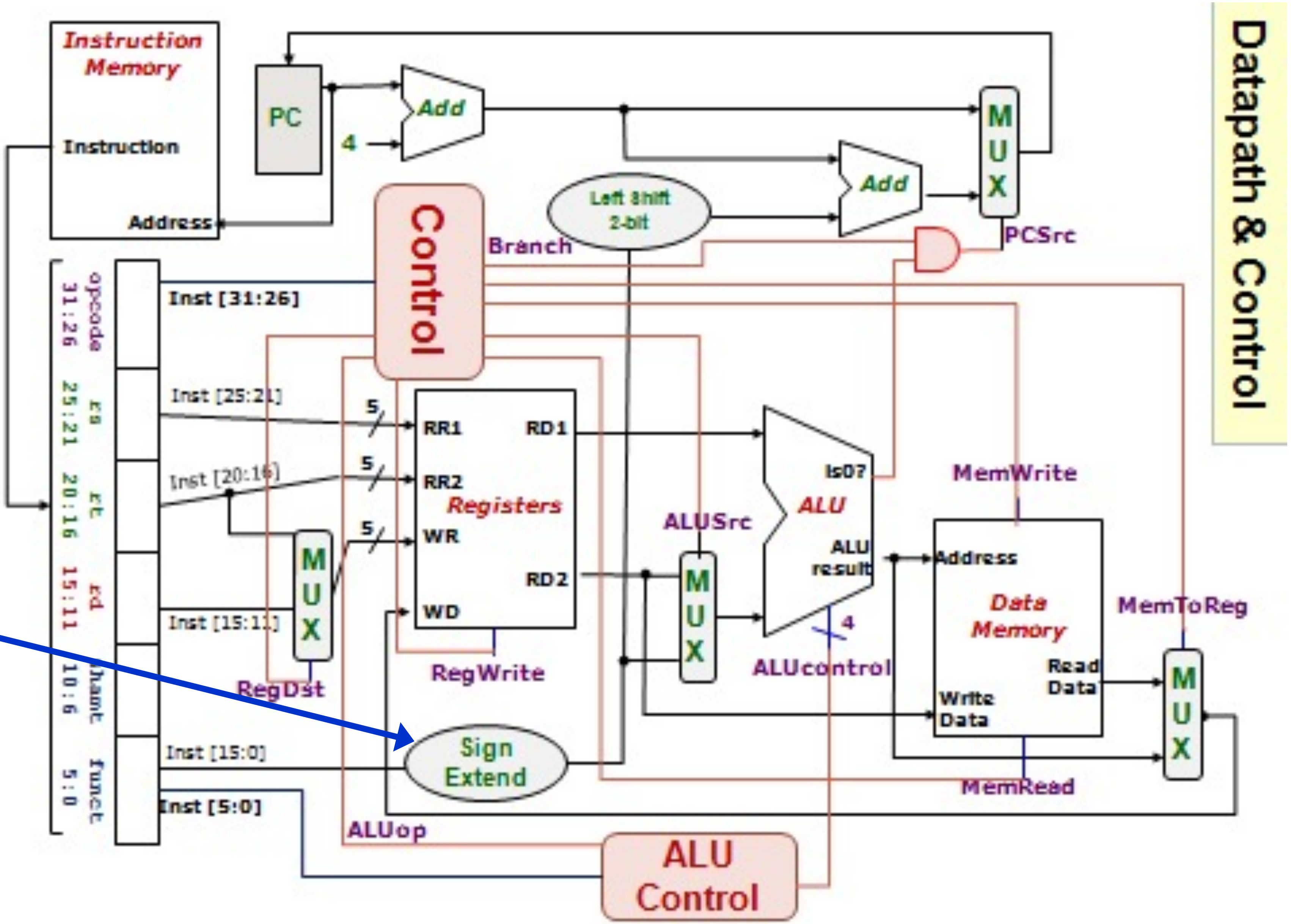
C AND NUMBER SYSTEMS

(PREPARED BY: AARON TAN)

Q1.

You will learn this in week 5.

What is sign extension?



Q1. Sign extension – extending the sign bit to fill in the bit representation.

Example: from 4 bits to 8 bits.

$$(5)_{10} = (0101)_{2s} = (00000101)_{2s}$$

$$(-3)_{10} = (1101)_{2s} = (11111101)_{2s}$$

Note: Sign extension works for complement systems (1's complement and 2's complement.)

Does it work for sign-and-magnitude system?

Q1. Positive values – straightforward: $(5)_{10} = (0101)_{2s} = (00000101)_{2s}$

Negative values? Why does padding 1s in front work?

$$(-3)_{10} = (1101)_{2s} = (\underbrace{11111}101)_{2s}$$

Value: $-2^3 = -8$

Value: $-2^7 + 2^6 + 2^5 + 2^4 + 2^3$

$= -128+64+32+16+8 = -8$

-2^k

$$\begin{aligned} &-2^m + (2^{m-1} + \ldots + 2^{k+1} + 2^k) \\ &= -2^m + 2^k\left(2^{m-k} - 1\right) \\ &= -2^m + 2^m - 2^k \\ &= -2^k \end{aligned}$$

Recall that sum of a GP = $\frac{a(r^n - 1)}{r-1}$.

Here, $a = 2^k, n = m - 1 - k + 1 = m - k$, and $r = 2$.

Q2. Performing subtraction in 1's complement.

Strategy: Convert $A - B$ to $A + (-B)$.

Why not perform subtraction directly?

(a) 0101.11 – 010.0101

Q2. Performing subtraction in 1's complement.

Strategy: Convert $A - B$ to $A + (-B)$.

(b) 010111.101 – 0111010.11

Q3. Convert the following decimal numbers to fixed-point binary in 2's complement, with 4 bits for the integer portion and 3 bits for the fraction portion.

(a) 1.75

$$0.75 = 0.5 + 0.25 = \frac{1}{2} + \frac{1}{2^2} = (0.1)_2 + (0.01)_2 = (0.11)_2$$

Therefore $1.75 = (0001.110)_2 =$ **$(0001.110)_{2s}$**

or

$0.75 \times 2 = 1.5$

$0.5 \times 2 = 1.0$

end

Q3. Convert the following decimal numbers to fixed-point binary in 2's complement, with 4 bits for the integer portion and 3 bits for the fraction portion.

(b) -2.5

$2.5 = (0010.100)_2$

$-2.5 = -(0010.100)_2$

Invert bits in 0010.100 and add 0.001

$$\begin{array}{r} 1101.011 \\ +\quad\quad\quad\ 1 \\ \hline 1101.100 \end{array}$$

$= \mathbf{(1101.100)_{2s}}$

Q3. Convert the following decimal numbers to fixed-point binary in 2's complement, with 4 bits for the integer portion and 3 bits for the fraction portion.

(c) 3.876

$0.876 = (0.1110)_2 \approx (0.111)_2$

Therefore $3.876 = (0011.111)_2$

$= \mathbf{(0011.111)_{2s}}$

$$
\boxed{\begin{aligned}
0.876\times 2 &= 1.752\\
0.752\times 2 &= 1.504\\
0.504\times 2 &= 1.008\\
0.008\times 2 &= 0.016
\end{aligned}}
$$

Q3. Convert the following decimal numbers to fixed-point binary in 2's complement, with 4 bits for the integer portion and 3 bits for the fraction portion.

(d) 2.1

$0.1 = (0.0001)_2 \approx (0.001)_2$

Therefore $2.1 = (0010.001)_2$

$= \mathbf{(0010.001)_{2s}}$

$0.1\times2 = 0.2$
$0.2\times2 = 0.4$
$0.4\times2 = 0.8$
$0.8\times2 = 1.6$

Q3. Using the binary representations you have just derived, convert them back into decimal.

(a) 1.75

= $\mathbf{(0001.110)_{2s}}$

= $(0001.110)_2$

$(0001.110)_2$

$= 2^0 + 2^{-1} + 2^{-2}$

$= 1.75$

(b) -2.5

= $\mathbf{(1101.100)_{2s}}$

= $-(0010.100)_2$

$-(0010.100)_2$

$= -(2^1 + 2^{-1})$

$= -2.5$

(c) 3.876

= $\mathbf{(0011.111)_{2s}}$

= $(0011.111)_2$

$(0011.111)_2$

$= 2^1 + 2^0$

$+ 2^{-1} + 2^{-2} + 2^{-3}$

$= 3.875$

(d) 2.1

= $\mathbf{(0010.001)_{2s}}$

= $(0010.001)_2$

$(0010.001)_2$

$= 2^1 + 2^{-3}$

$= 2.125$

Q4. How would you represent the decimal value -0.078125 in the IEEE 754 single-precision representation? Express your answer in hexadecimal.

$$-0.078125 = -(0.000101)_2 = -(1.01)_2 \times 2^{-4}$$

Exponent: $-4 + 127$

$= 123$

$= (01111011)_2$

Sign-bit	8-bit exponent	23-bit mantissa
1	01111011	01000000000000000000000

$$\underbrace{1\,011}_{B}\,\underbrace{1101}_{D}\,\underbrace{1\,010}_{A}\,\underbrace{0000}_{0}\,\underbrace{0000}_{0}\,\underbrace{0000}_{0}\,\underbrace{0000}_{0}\,\underbrace{0000}_{0}$$

1 01111011 01000000000000000000000

B D A 0 0 0 0 0

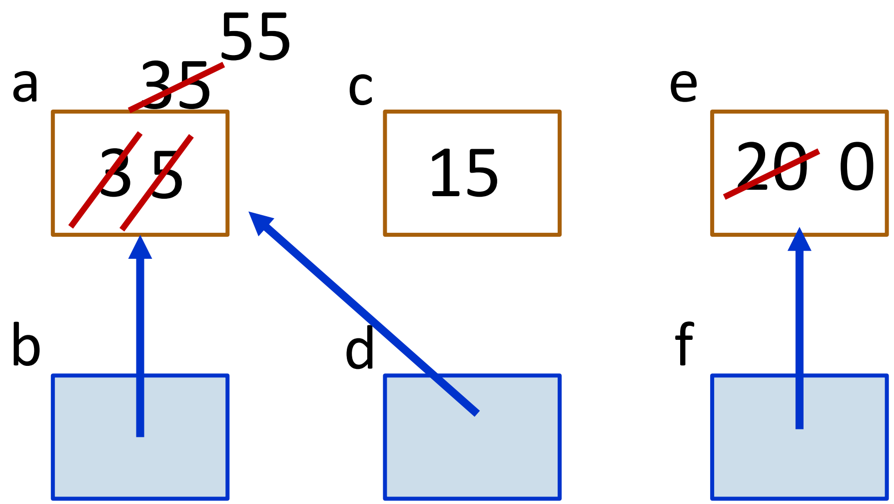
Q6. Trace the program manually and write out its output.

```
int a = 3, *b, c, *d, e, *f;

b = &a;
*b = 5;
c = *b * 3;
d = b;
e = *b + c;
*d = c + e;
f = &e;
a = *f + *b;
*f = *d - *b;

printf("a = %d, c = %d, e = %d\n", a, c, e);
printf("*b = %d, *d = %d, *f = %d\n", *b, *d, *f);
```

This does NOT make d point to b!
This copies the content of b into d.



Output:

a = 55, c = 15, e = 0

***b = 55, *d = 55, *f = 0**

END OF FILE

For reference.

1s complement

- Given a number ***x*** which can be expressed as an *n*-bit binary number, its negated value can be obtained in **2s-complement** representation using: $\mathbf{-x = 2^n - x - 1}$

- Example: x=12; Assume 8 bits, $-x = 2^8 - 12 - 1 = 243 = (11110011)_{1s}$

- Technique: invert the bits. $12 = (00001100)_{1s} \rightarrow -12 = (11110011)_{1s}$

2s complement

- Given a number ***x*** which can be expressed as an *n*-bit binary number, its negated value can be obtained in **2s-complement** representation using: $\mathbf{-x = 2^n - x}$

- Example: x=12; Assume 8 bits, $-x = 2^8 - 12 = 244 = (11110100)_{2s}$

- Technique: invert the bits then plus 1. $12 = (00001100)_{2s} \rightarrow -12 = (11110100)_{2s}$

For reference.

10.4 Comparisons

Important!

4-bit system

Positive values

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
+7	0111	0111	0111
+6	0110	0110	0110
+5	0101	0101	0101
+4	0100	0100	0100
+3	0011	0011	0011
+2	0010	0010	0010
+1	0001	0001	0001
+0	0000	0000	0000

Negative values

Value	Sign-and-Magnitude	1s Comp.	2s Comp.
-0	1000	1111	-
-1	1001	1110	1111
-2	1010	1101	1110
-3	1011	1100	1101
-4	1100	1011	1100
-5	1101	1010	1011
-6	1110	1001	1010
-7	1111	1000	1001
-8	-	-	1000