10
Number Systems

ENGR 3410 – Computer Architecture

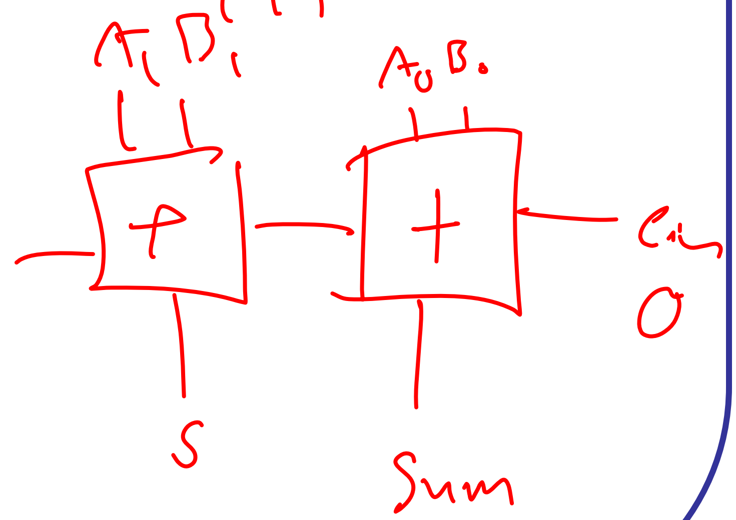
Mark L. Chang

Fall 2006

Number Systems

- Problem: Implement simple pocket calculator
- Need: Display, adders & subtractors, inputs
  - Display: Seven segment displays
  - Inputs: Switches
- Missing: Way to implement numbers in binary
- Approach: From decimal to binary numbers
  (and back)

A	B	C	Sum	Carry
0	0	0	0	0
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



Decimal (Base 10) Numbers

- Positional system - each digit position has a value

  $2534 = 2*1000 + 5*100 + 3*10 + 4*1$

- Alternate view: Digit position i from the right = Digit * 10^i (rightmost is position 0)

  $2534 = 2*10^3 + 5*10^2 + 3*10^1 + 4*10^0$

Base R Numbers

- Each digit in range 0..(R-1)

  0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F ...

  A = 10

  B = 11

  C = 12

  D = 13

  E = 14

  F = 15

- Digit position i = Digit * R^i

  $D_3 D_2 D_1 D_0$ (base R) = $D_3*R^3+D_2*R^2+D_1*R^1+D_0*R^0$

Conversion to Decimal

- Binary: $(101110)_2$

$$1\times2^5 + 0\times2^4 + 1\times2^3 + 1\times2^2 + 1\cdot2^1 + 0\times2^0 = (46)_{10}$$
$$32 + 0 + 8 + 4 + 2 + 0$$

- Octal: $(325)_8$

$$3\times8^2 + 2\times8^1 + 5\times8^0 = (213)_{10}$$
$$192 + 16 + 5 =$$

- Hexadecimal: $(E32)_{16}$

$$14\times16^2 + 3\times16^1 + 2\times16^0$$
$$= 14\times256 + 48 + 2$$
$$= 3584 + 48 + 2 = (3634)_{10}$$

Conversion Decimal

- Binary: $(110101)_2$

- Octal: $(524)_8$

- Hexadecimal: $(A6)_{16}$

Conversion of Decimal to Binary (Method 1)

- For positive, unsigned numbers
- Successively subtract the greatest power of two less than the number from the value. Put a 1 in the corresponding digit position

- $2^0=1$ $2^4=16$ $2^8=256$ $2^{12}=4096$ (4K)
- $2^1=2$ $2^5=32$ $2^9=512$ $2^{13}=8192$ (8K)
- $2^2=4$ $2^6=64$ $2^{10}=1024$ (1K)
- $2^3=8$ $2^7=128$ $2^{11}=2048$ (2K)

Decimal to Binary Method 1

- Convert $(2578)_{10}$ to binary

$$\begin{array}{r} 2578_{10} \\ -\ 2048_{10} \\ \hline 530_{10} \end{array} = 2^{11} \qquad \begin{array}{r} 530 \\ -\ 512 \\ \hline 18 \end{array} = 2^{9} \qquad \begin{array}{r} 18 \\ -\ 16 \\ \hline 2 \end{array} = 2^{4} \qquad \begin{array}{r} 2 \\ -\ 2 \\ \hline 0 \end{array} = 2^{1}$$

$$(101000010010)_2$$

- Convert $(289)_{10}$ to binary

$$\begin{array}{r} 289_{10} \\ -\ 256_{10} \\ \hline 33 \\ -\ 32 \\ \hline 1 \\ -\ 1 \\ \hline \end{array} \quad \begin{array}{l} 2^{8} \\ \\ = 2^{5} \\ \\ = 2^{0} \end{array}$$

$$(100100001)_2$$

Conversion of Decimal to Binary (Method 2)

- For positive, unsigned numbers
- Repeatedly divide number by 2. Remainder becomes the binary digits (right to left)
- Explanation:

Decimal to Binary Method 2

- Convert $(289)_{10}$ to binary

289_{10}

144	1
72	0
36	0
18	0
9	0
4	1
2	0
1	0
0	1

$(100100001)_2$

Decimal to Binary Method 2

- Convert $(85)_{10}$ to binary

85	
42	1
21	0
10	1
5	0
2	1
1	0
0	1

$(1010101)_2$

Converting Binary to Hexadecimal

- 1 hex digit = 4 binary digits
- Convert $(1110001101011101 0011)_2$ to hex

$(E\ 3\ 5\ D\ 3)_{16}$

- Convert $(A3FF2A)_{16}$ to binary

$(1010,\ 0011,\ 1111,\ 1111,\ 0010,\ 1010)_2$

Converting Binary to Octal

- 1 octal digit = 3 binary digits
- Convert $(10100101001101010011)_2$ to octal

$(2451523)_8$

- Convert $(723642)_8$ to binary

$(111, 010, 011, 110, 100, 010)_2$

Converting Decimal to Octal/Hex

- Convert to binary, then to other base
- Convert $(198)_{10}$ to Hexadecimal

- Convert $(1983020)_{10}$ to Octal

Arithmetic Operations

Decimal:

```
   1  1  1
   5  7  8  9  2
+  7  8  9  5  6
----------------
1  3  6  8  4  8   (10)
```

Binary:

```
          1  1  1
   1  0  1  0  1  1  1
+  0  1  0  0  1  0  1
----------------------
(  1  1  1  1  1  0  0 )2
```

Decimal:

```
      6     8
   5  7  18  9  12
-  3  2  9   4  6
----------------
   2  4  9   4  6   (10)
```

Binary:

```
   0   1  10   1   1  10  10
   1   0   1   0   0   1   1   10
-  0   0   1   1   0   1   1   1
-------------------------------
(  0   1   1   0   1   1   1   1 )2
```

Arithmetic Operations (cont.)

Binary:

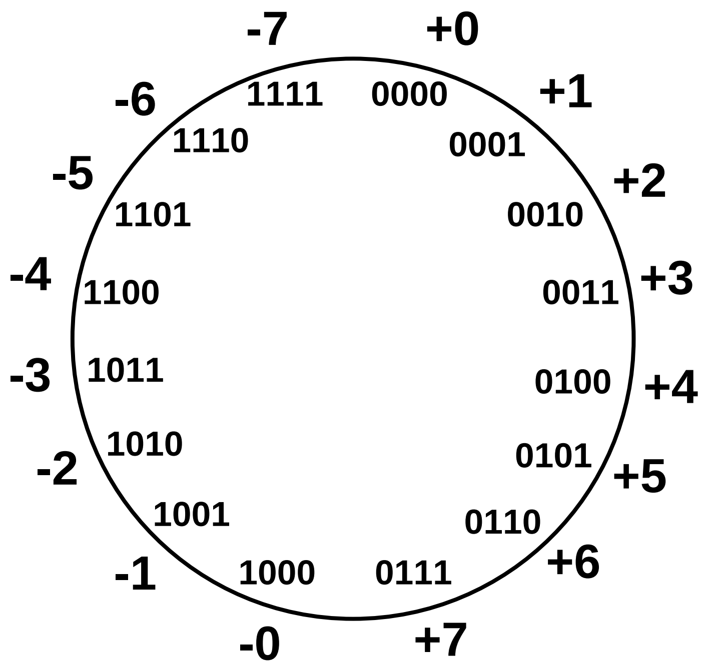
$$
\begin{array}{r}
1\ 0\ 0\ 1 \\
*\ \ 1\ 0\ 1\ 1 \\
\hline
1\ 1\ 0\ 0\ 1 \\
1\ 1\ 0\ 0\ 1\ \ \ \\
0\ 0\ 0\ 0\ \ \ \ \ \ \\
+\ \ 1\ 0\ 0\ 1\ \ \ \ \ \ \ \ \ \\
\hline
(1100011)_2
\end{array}
$$

Negative Numbers

- Need an efficient way to represent negative numbers in binary
  - Both positive & negative numbers will be strings of bits
  - Use fixed-width formats (4-bit, 16-bit, etc.)

- Must provide efficient mathematical operations
  - Addition & subtraction with potentially mixed signs
  - Negation (multiply by -1)

Sign/Magnitude Representation



+

0 100 = + 4

1 100 = - 4

-

High order bit is sign: 0 = positive (or zero), 1 = negative

Three low order bits is the magnitude: 0 (000) thru 7 (111)

Number range for n bits = +/- 2^{n-1} -1

Representations for 0:

Sign/Magnitude Addition

Idea: Pick negatives so that addition/subtraction works

$$\begin{array}{rcccccl} & 0 & 0 & 1 & 0 & (+2)_{10} \\ + & 0 & 1 & 0 & 0 & (+4)_{10} \\ \hline & 0 & 1 & 1 & 0 & (6)_{10} \end{array}$$

$$\begin{array}{rccccccl} & & 1 & 0 & 1 & 0 & (-2) \\ + & & 1 & 1 & 0 & 0 & (-4) \\ \hline & 1 & 0 & 1 & 1 & 0 & \rightarrow = +6_{10} \end{array}$$

$= -6_{10}$

$$\begin{array}{rcccccl} & 0 & 0 & 1 & 0 & (+2) \\ + & 1 & 1 & 0 & 0 & (\ -4) \\ \hline & 1 & 1 & 1 & 0 & (-6)_{10} \end{array}$$

$$\begin{array}{rcccccl} & 1 & 0 & 1 & 0 & (\ -2) \\ + & 0 & 1 & 0 & 0 & (+4) \\ \hline & 1 & 1 & 1 & 0 & (-6)_{10} \end{array}$$

Bottom line: Basic mathematics are too complex in Sign/Magnitude

Idea: Pick negatives so that addition works

- Let -1 = 0 - (+1):

```
   0 0 0 1 0 ( 0)
 - 0 0 0 1   (+1)
   ---------
   1 1 1 1   (-1)10
```

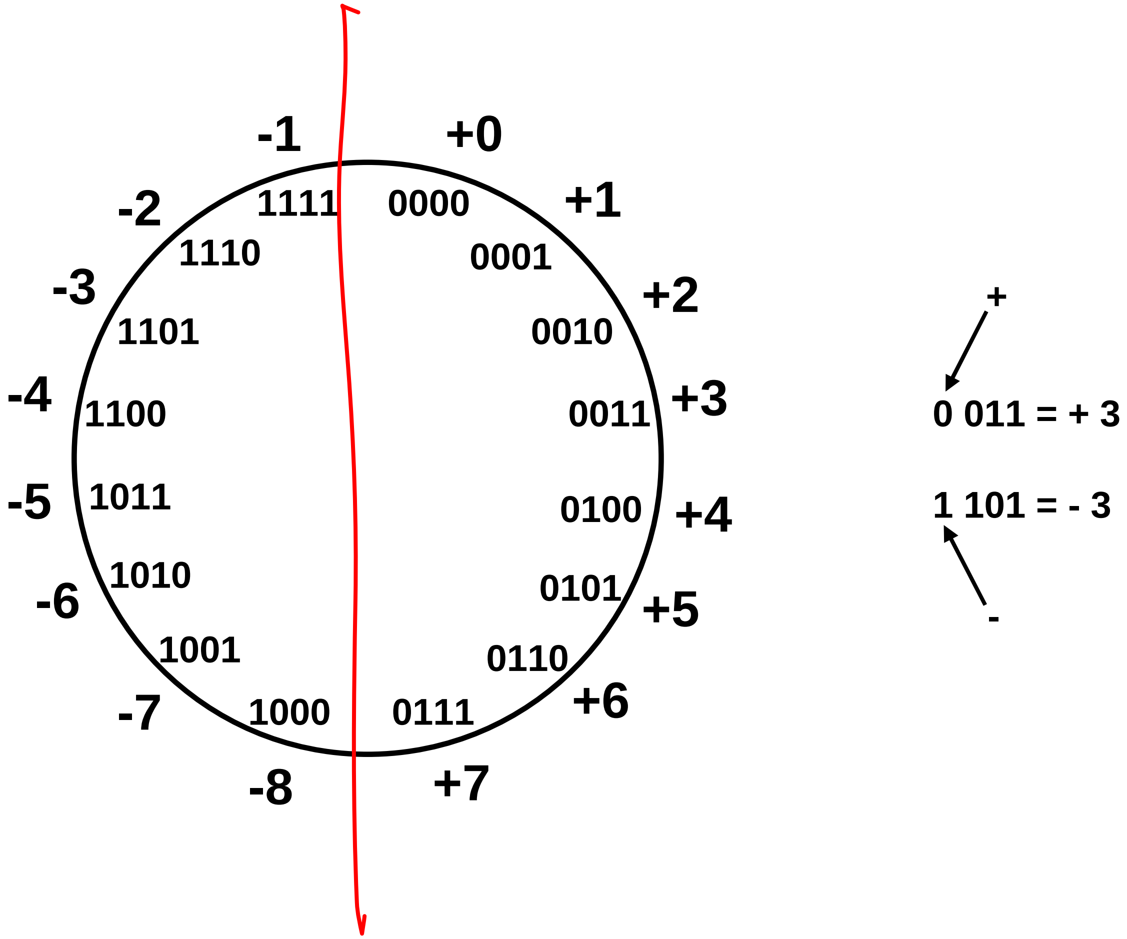
- Does addition work?

```
      1   1
   0 0 1 0   (+2)
 + 1 1 1 1   ( -1)
   ---------
 1 0 0 0 1   = (1)10
```

- Result: Two's Complement Numbers

Two's Complement

- Only one representation for 0
- One more negative number than positive number
- Fixed width format for both pos. & neg. numbers



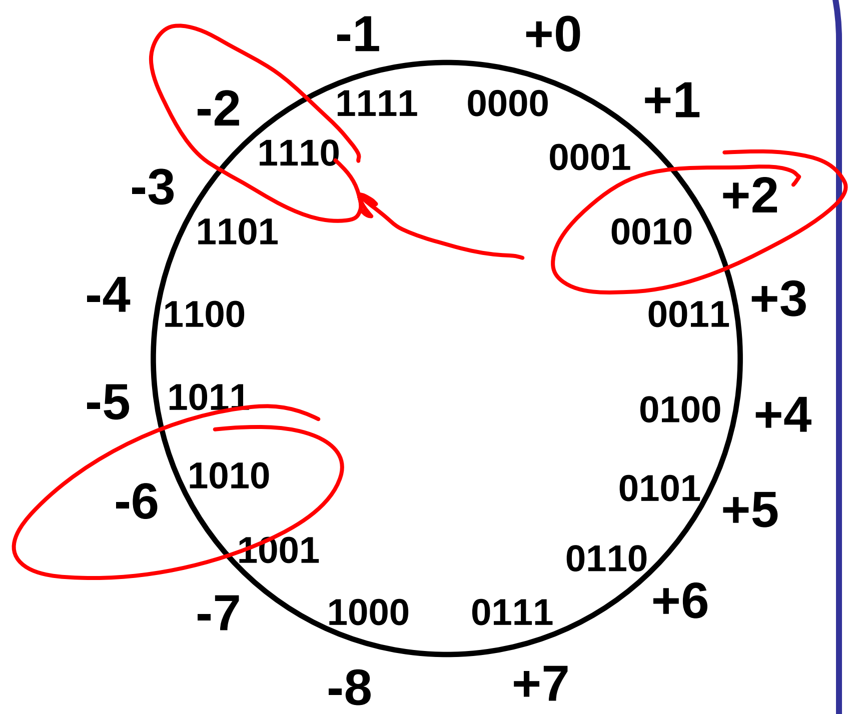
Negating in Two's Complement

- Flip bits & Add 1
- Negate $(0010)_2$ (+2)

1101
+ 1
1110

- Negate $(1110)_2$ (-2)

0001
+ 1
$0010 = (2)_{10}$



Addition in Two's Complement

```
  0 0 1 0  (+2)
+ 0 1 0 0  (+4)
  0 1 1 0 = 6
```

```
  1 1 1 0  (-2)
+ 1 1 0 0  (-4)
1 1 0 1 0 = (-6)10
```

```
  0 0 1 0  (+2)
+ 1 1 0 0  ( -4)
  1 1 1 0 = (-2)10
```

```
  1 1 1 0  ( -2)
+ 0 1 0 0  (+4)
1 0 0 1 0 = (2)10
```

Subtraction in Two's Complement

- A - B = A + (-B) = A + $\overline{\text{B}}$ + 1

- 0010 - 0110

$(2)_{10} - (6)_{10} = (-4)_{10}$

```
  0010        0010        0010
- 0110  -->   1001  -->  +1010
              +  1       -----
              ----        1100
```

$= (-4)_{10}$

- 1011 - 1001

$(-5)_{10} - (-7)_{10}$

$= (+2)_{10}$

```
 1011           111
 0110  -->      1011
 +  1         + 0111
 ----         ------
               10010
```

$= (+2)_{10}$

- 1011 - 0001

$(-5)_{10} - (1)_{10}$

$(-6)_{10}$

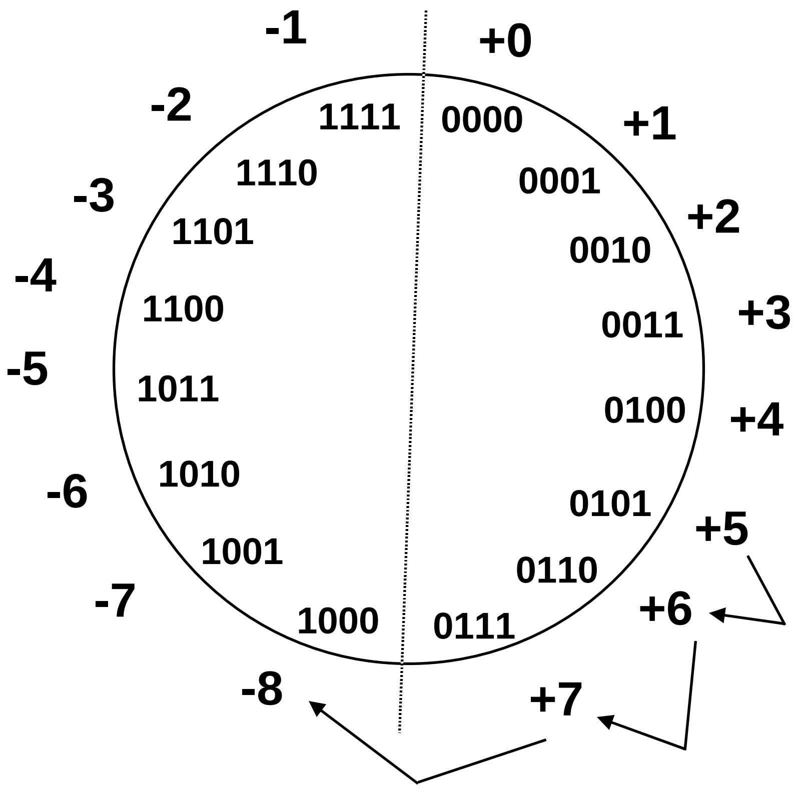
```
 1011          111
 1110          1011
 +  1         +1111
 ----         -----
              11010
```

$(-6)_{10}$

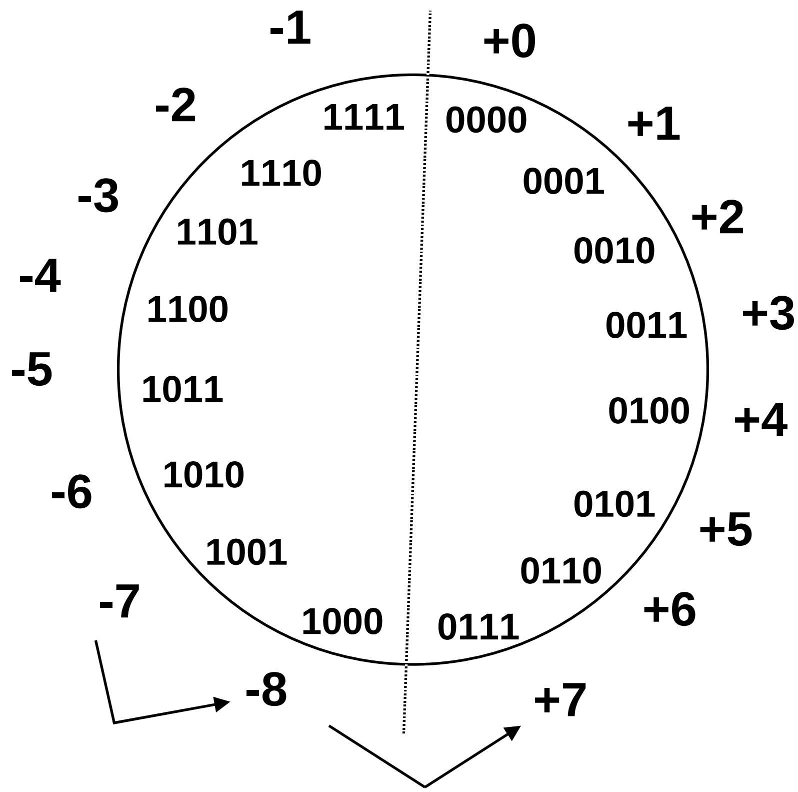
Overflows in Two's Complement

Add two positive numbers to get a negative number

or two negative numbers to get a positive number



5 + 3 = -8



-7 - 2 = +7

Overflow Detection in Two's Complement

		1 1 1
5	XOR	0 1 0 1
+ 3		0 0 1 1
-8		0 1 0 0 0
Overflow		yes

	0
-7	1 0 0 1
+ -2	1 1 1 0
7	1 0 1 1 1
Overflow	yes

	0
5	0 1 0 1
2	0 0 1 0
7	0 0 1 1 1
No overflow	

	1 1 1
-3	1 1 0 1
-5	1 0 1 1
-8	1 1 0 0 0
No overflow	

Overflow when carry in to sign does not equal carry out

Converting Decimal to Two's Complement

- Convert absolute value to binary, then negate if necessary

- Convert $(-9)_{10}$ to 6-bit Two's Complement

2^0

$$|-9|_{10} = 9_{10} \longrightarrow 001001_2 \xrightarrow{\text{flip}} \begin{array}{r} 110110 \\ +\quad\quad 1 \\ \hline (110111)_2 \end{array} = -9_{10}$$

2^3

- Convert $(9)_{10}$ to 6-bit Two's Complement

Converting Two's Complement to Decimal

- If Positive, convert as normal;
  If Negative, negate then convert.

- Convert $(11010)_2$ to Decimal

$$
\begin{array}{r}
00101 \\
+ \quad\quad 1 \\
\hline
(00110)_2
\end{array}
= (6)_{10} \leadsto (-6)_{10}
$$

- Convert $(01011)_2$ to Decimal

Sign Extension

- To convert from N-bit to M-bit Two’s Complement (N>M), simply duplicate sign bit:

- Convert $(1011)_2$ to 8-bit Two’s Complement

$$(1011)_2 \longrightarrow (1111,1011)_2$$

- Convert $(0010)_2$ to 8-bit Two’s Complement

$$(0010)_2 \longrightarrow (0000,0010)_2$$