10
Number Systems

ENGR 3410 – Computer Architecture

Mark L. Chang

Fall 2009

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: $(101110)_2$

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

- Octal: $(325)_8$

$$3\times8^2+2\times8^1+5\times8^0$$
$$3\times64+16+5 = (213)_{10}$$

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

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

Conversion Decimal

5 4 3 2 1 0

- Binary: $(110101)_2$

$$2^5 + 2^4 + 2^2 + 2^0 =$$

$$32 + 16 + 4 + 1 = 53_{10}$$

- Octal: $(524)_8$

$$5 \times 8^2 + 2 \times 8^1 + 4 \times 8^0$$

$$5 \times 64 + 16 + 4 = 340_{10}$$

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

$$10 \times 16^1 + 6 \times 16^0 = 166_{10}$$

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

$$2578 - 2048 = 2^{11} \quad 530 - 512 = 2^{9} \quad 18 - 16 = 2^{4} \quad 2 - 2 = 2^{1}$$

$$530 \quad 18 \quad 2 \quad 0$$

$$(101000010010)_2 = \emptyset b\ 101000010010$$

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

$$289 - 256 = 2^{8}$$

$$33 - 32 = 2^{5}$$

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

$$0$$

$\emptyset b\ 100100001$

$(\qquad\qquad)_2$

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:

$$N = (a_{n-1}\, a_{n-2}\, a_{n-3} \cdots a_0)_R$$

$$= a_{n-1} R^{n-1} + a_{n-2} R^{n-2} + \cdots + a_0 R^0$$

$$Q = \frac{N}{R}$$

remainder

Decimal to Binary Method 2

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

289	
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	r 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 $(11100011010111010011)_2$ 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 $(1010010100110101001 1)_2$ to octal

$(2\ 4\ 5\ 1\ 5\ 2\ 3)_8$

- Convert $(723642)_8$ to binary

7	2	3	6	4	2
111	010	011	110	100	010

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

Converting Decimal to Octal/Hex

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

$(11000110)_2 = (198)_{10}$

$198 - 128 = 70$ 2^7

$70 - 64 = 6$ 2^6

$6 - 4 = 2$ 2^2

$2 - 2 = 0$ 2^1

$(C\ 6)_{16}$

- 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
```

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
```

Decimal:

```
  6   8
  5 7 8 9 2
- 3 2 9 4 6
-----------
  2 4 9 4 6
```

Binary:

```
  0 1 1 0 1 1 0 1 0
  1 0 1 0 0 1 1 0
- 0 0 1 1 0 1 1 1
-----------------
+ 0 1 1 0 1 1 1 1
```

Arithmetic Operations (cont.)

Binary:

```
      1 0 0 1
    * 1 0 1 1
    ---------
      1 0 0 1
    1 0 0 1
  0 0 0 0
1 0 0 1
---------------
1 1 0 0 0 1 1
```

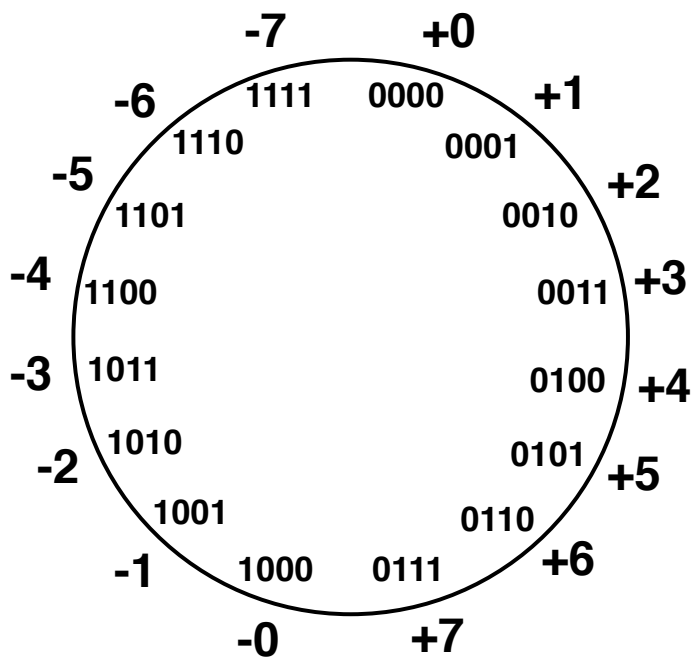
partial product "away"

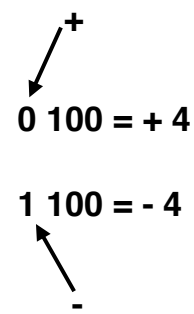
Negative Numbers

- 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





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

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

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

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

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

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

Idea: Pick negatives so that addition works

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

$$
\begin{array}{rrrrrl}
 & 0 & 0 & 0 & 0 & (\ 0) \\
- & 0 & 0 & 0 & 1 & (+1) \\
\hline
 & 1 & 1 & 1 & 1 & (-1)_{10}
\end{array}
$$

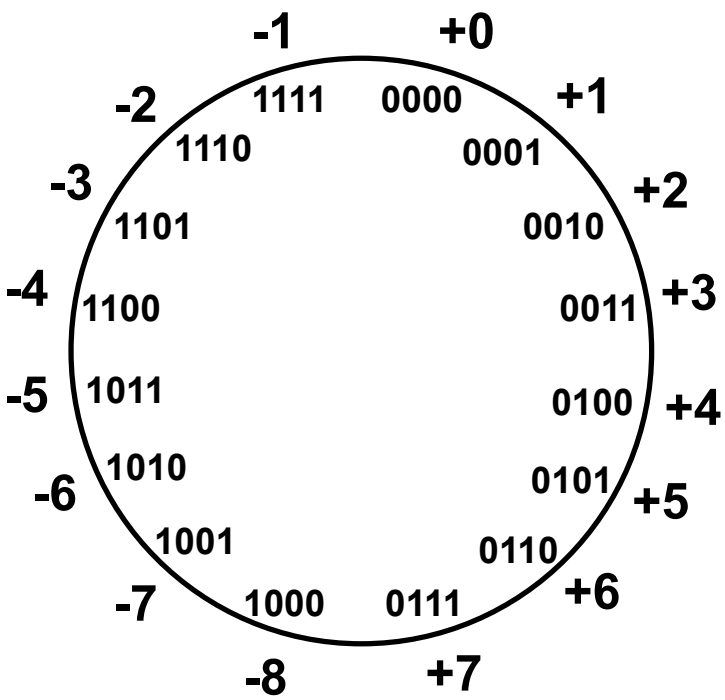
- Does addition work?

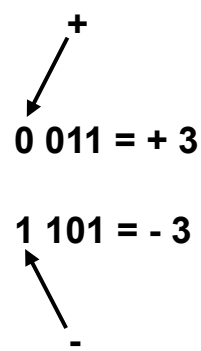
$$
\begin{array}{rrrrrl}
 & & 1 & 1 & & \\
 & 0 & 0 & 1 & 0 & (+2) \\
+ & 1 & 1 & 1 & 1 & (\ -1) \\
\hline
1 & 0 & 0 & 0 & 1 & (+1)_{10}
\end{array}
$$

- 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)

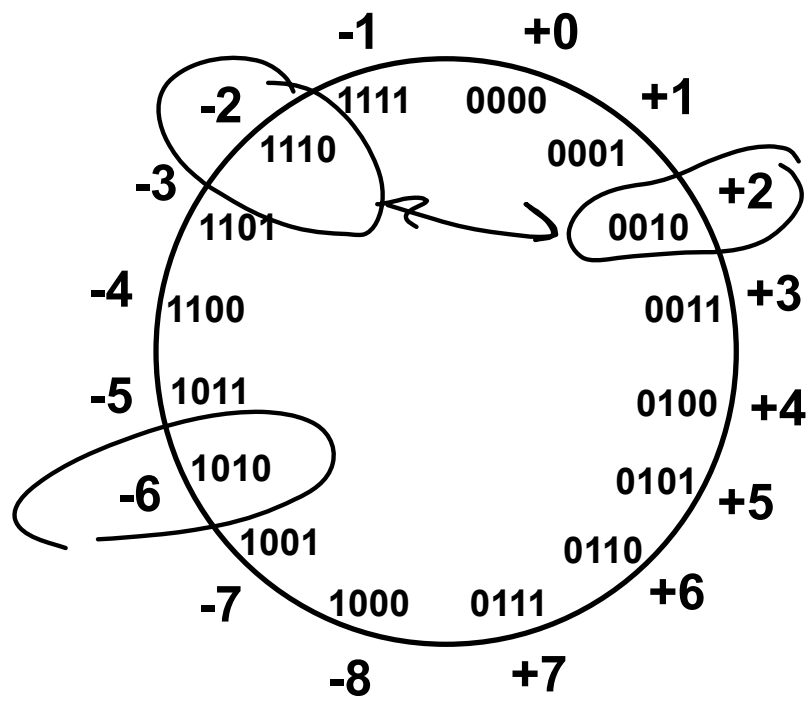
1101

1110

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

0001

0010



Addition in Two's Complement

$$\begin{array}{rrrrrl} & 0 & 0 & 1 & 0 & (+2) \\ + & 0 & 1 & 0 & 0 & (+4) \\ \hline & 0 & 1 & 1 & 0 & \end{array}$$

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

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

$$\begin{array}{rrrrrl} & 1 & 1 & 1 & 0 & (-2) \\ + & 0 & 1 & 0 & 0 & (+4) \\ \hline 1 & 0 & 0 & 1 & 0 & (+2) \end{array}$$

Subtraction in Two's Complement

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

- 0010 - 0110

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

$$\begin{array}{r} 0010 \\ -\,0110 \\ \hline \end{array} \quad \text{- - f -} \quad \begin{array}{r} 0010 \\ 1001 \end{array} \quad \begin{array}{r} 0010 \\ +\,1010 \\ \hline (1100)_2 = (-4)_{10} \end{array}$$

- 1011 - 1001

$-5 - -7 = +2$

$$\begin{array}{r} 1011 \\ -\,1001 \\ \hline \end{array} \quad \text{flip} \quad \begin{array}{r} 1011 \\ 0110 \;+1 \end{array} \quad \begin{array}{r} 111 \\ 1011 \\ 0111 \\ \hline 0010 \end{array} = (2)_2$$

- 1011 - 0001

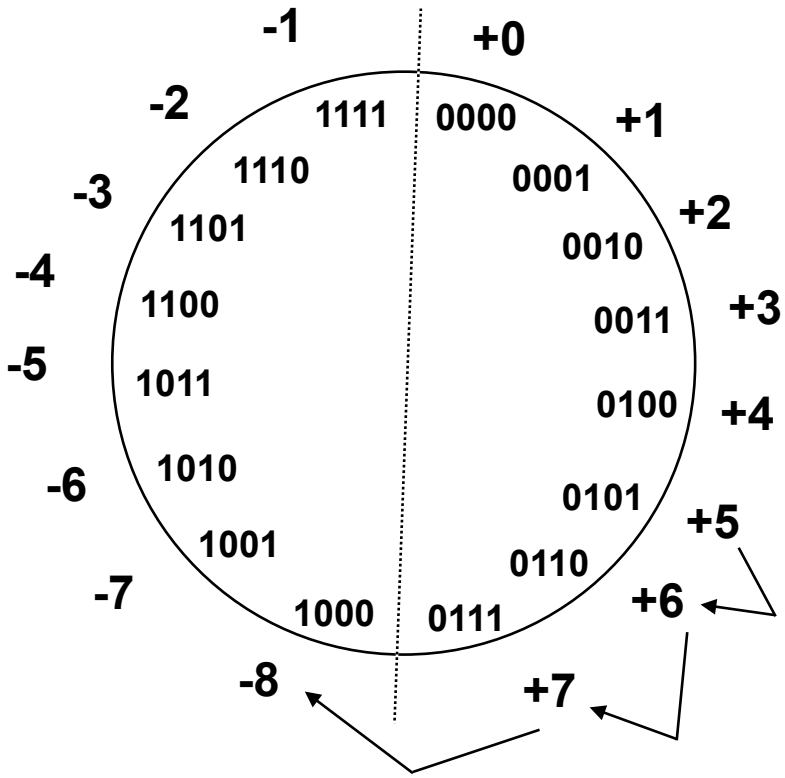
$-5 - 1$

$$\begin{array}{r} 1011 \\ -\,0001 \\ \hline \end{array} \quad \text{flip} \quad \begin{array}{r} 1011 \\ 1110 \;+1 \end{array} \quad \begin{array}{r} 111 \\ 1011 \\ 1111 \\ \hline 1\;\underbrace{1010} \end{array} = (-6)_{10}$$

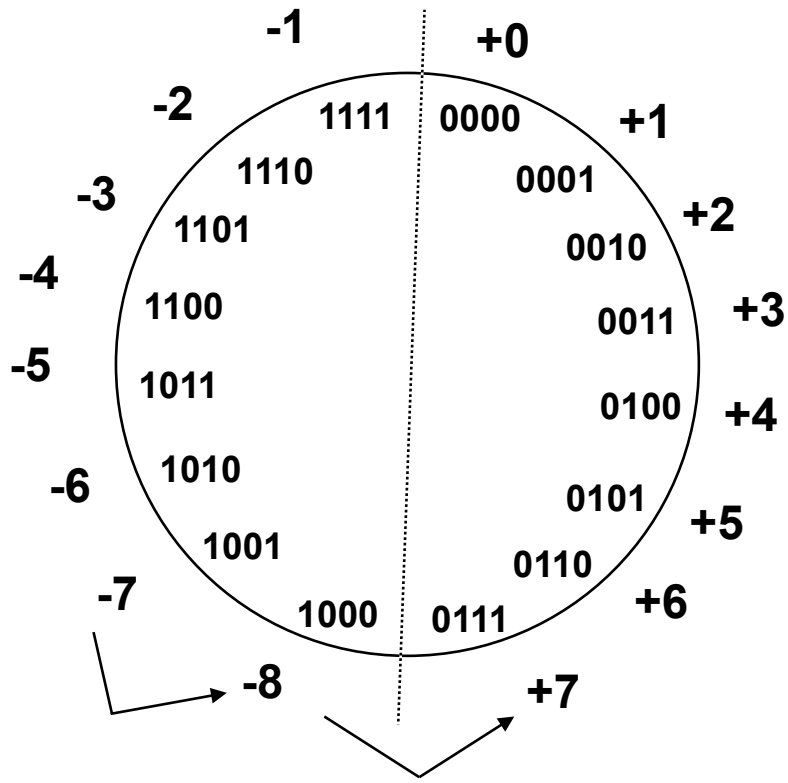
Overflows in Two's Complement

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		0 1 0 1
+ 3		0 0 1 1
-8		0 1 0 0 0

Overflow

		0
-7		1 0 0 1
+ -2		1 1 1 0
7		1 0 1 1

Overflow

A_n
B_n
S_n

Cin_n
$Cout_n$
XOR
Oflow

5	0 1 0 1
+ 2	0 0 1 0
7	

No overflow

-3	1 1 0 1
+ -5	1 0 1 1
-8	

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

$$001001 \overset{f}{=} 110110 = (110111)_2$$

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

001001

Converting Two's Complement to Decimal

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

- Convert $(11010)_2$ to Decimal

$$00101 + 1 \Rightarrow 00110 = (-6_{10})$$

- Convert $(01011)_2$ to Decimal

$$01011 = (11)_{10}$$

Sign Extension

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

-5_{10}

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

(bad) 0000,1011 = $(11)_{10}$

(Good) 1111,1011 = 00000100 + 1 = 0000,0101

$5 - (5)_{10}$

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

0000,0010 → 2_{10}