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)

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

Number System (Conversion to Decimal)

Binary: $(101110)_2$ =

$1\times 2^5 + 0\times 2^4 + 1\times 2^3 + 1\times 2^2 + 1\times 2^1 + 0\times 2^0 =$

$= 32 + 0 + 8 + 4 + 2 + 0 = (46)_{10}$

Octal: $(325)_8$ =

$3\times 8^2 + 2\times 8^1 + 5\times 8^0 =$

$= 192 + 16 + 5 = 213_{10}$

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

$= 14\times 16^2 + 3\times 16^1 + 2\times 16^0$

$= 14\times 256 + 48 + 2$

$= 3584 + 48 + 2 = 3634_{10}$

Conversion from Base R to 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 (11)

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

$$2578_{10} - 2048_{10} = 2^{11}$$
$$530_{10}$$

$$530 - 512 = 2^9$$
$$18$$

$$18 - 16 = 2^4$$
$$2$$

$$2 - 2 = 2^1$$
$$0$$

$$101000010010_2$$

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

$$289 - 256 = 2^8$$
$$33$$
$$33 - 32 = 2^5$$
$$1$$
$$1 - 1 = 2^0$$

$$(100100001)_2$$

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: Number N in base R

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

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

$$Q_0 = \frac{N}{R} = a_{n-1} R^{n-2} + a_{n-2} R^{n-3} + \cdots a_1 R^0 \quad \text{remainder } a_0$$

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

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

Convert $(723642)_8$ to binary

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

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

$5\ \cancel{7}\ {}^{1}8\ \cancel{9}\ {}^{1}2$

\- 3 2 9 4 6

2 4 9 4 6

Binary:

0 1 10 1 1 10 10

$\cancel{1}\ \cancel{0}\ \cancel{1}\ \cancel{0}\ \cancel{0}\ \cancel{1}\ \cancel{1}\ {}^{1}0$

\- 0 0 1 1 0 1 1 1

0 1 1 0 1 1 1 1

Arithmetic Operations (cont.)

Binary:

$$
\begin{array}{rccccccc}
 & & & & 1 & 0 & 0 & 1 \\
* & & & & 1 & 0 & 1 & 1 \\
\hline
 & & & & 1 & 0 & 0 & 1 \\
 & & & 1 & 0 & 0 & 1 & \\
 & & 0 & 0 & 0 & 0 & & \\
+ & 1 & 0 & 0 & 1 & & & \\
\hline
 & (1 & 1 & 0 & 0 & 0 & 1 & 1)_2
\end{array}
$$

Negative Numbers

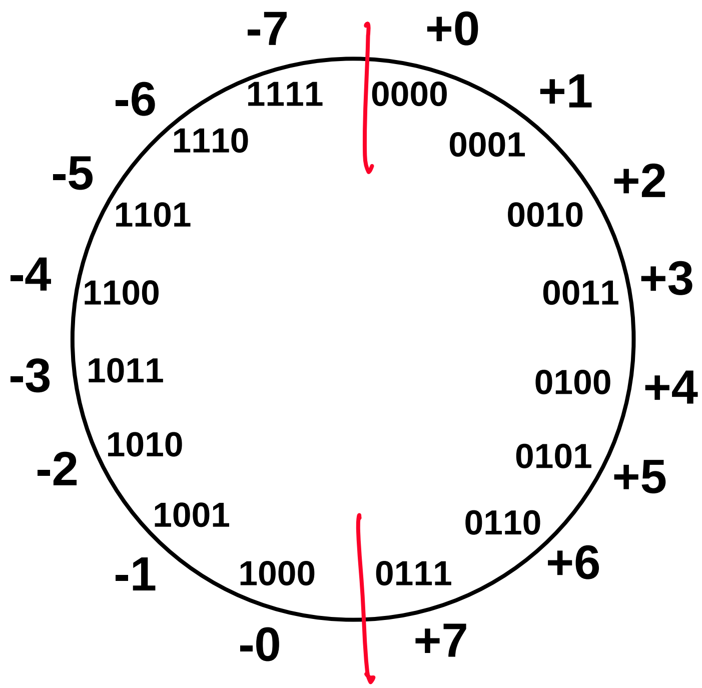
Need an efficient way to represent negative numbers in binary

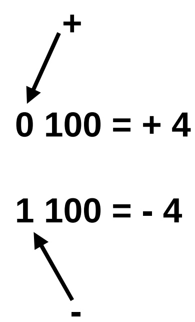
- Both positive & negative numbers will be strings of bits
- Use fixed-width formats (4-bit, 16-bit, etc.)

Must provide efficient mathematical operations

- Addition & subtraction with potentially mixed signs
- Negation (multiply by -1)

Sign/Magnitude Representation





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 = $\pm 2^{n-1} - 1$

Representations for 0:

Sign/Magnitude Addition

Idea: Pick negatives so that addition/subtraction works

$$
\begin{array}{rl}
0\ 0\ 1\ 0 & (+2)_{10} \\
+\ \underline{0\ 1\ 0\ 0} & (+4)_{10} \\
0\ 1\ 1\ 0 &
\end{array}
\qquad
\begin{array}{rl}
1\ 0\ 1\ 0 & (-2) \\
+\ \underline{1\ 1\ 0\ 0} & (-4) \\
(1\ 0\ 1\ 1\ 0) & = -6 \\
 & = +6
\end{array}
$$

$$
\begin{array}{rl}
0\ 0\ 1\ 0 & (+2) \\
+\ \underline{1\ 1\ 0\ 0} & (\ -4) \\
1\ 1\ 1\ 0 & (-6)_{10}
\end{array}
\qquad
\begin{array}{rl}
1\ 0\ 1\ 0 & (\ -2) \\
+\ \underline{0\ 1\ 0\ 0} & (+4) \\
(1\ 1\ 1\ 0)_2 & (-6)_{10}
\end{array}
$$

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

Idea: Pick negatives so that addition works

Let -1 = 0 - (+1):

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

Does addition work?

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

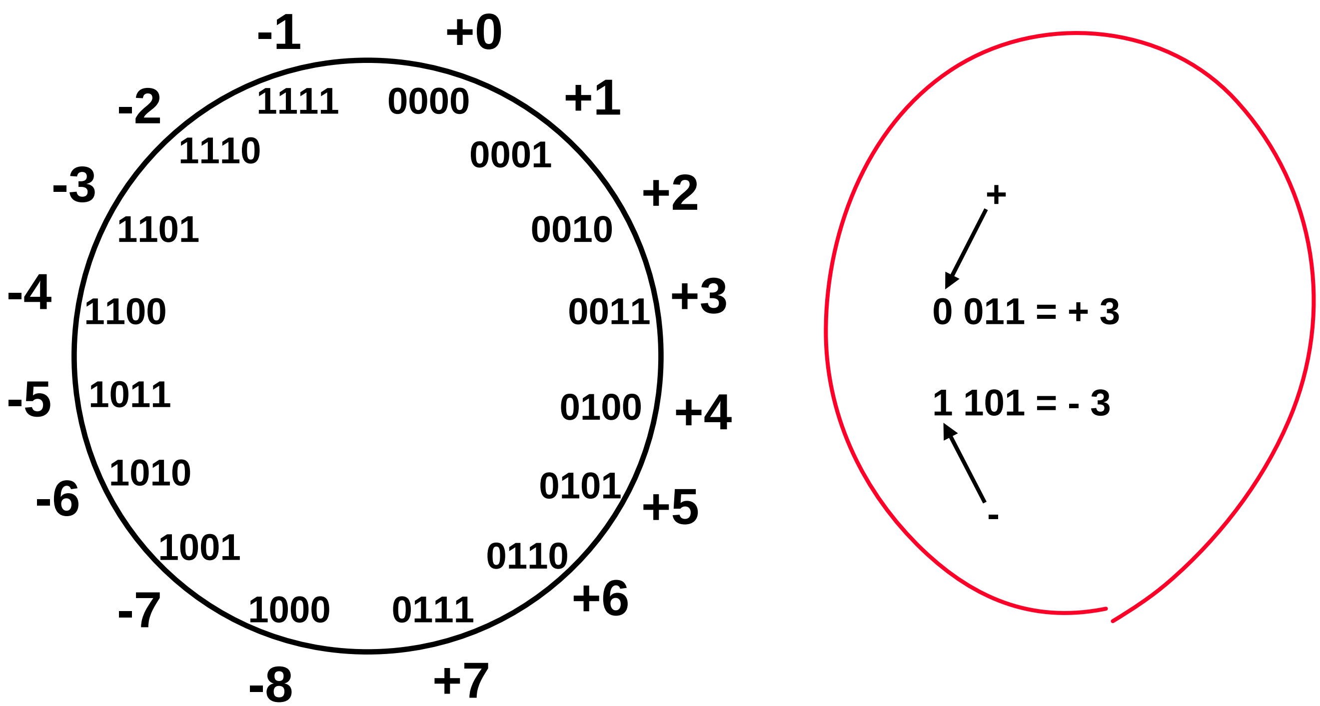
Result: Two's Complement Numbers

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

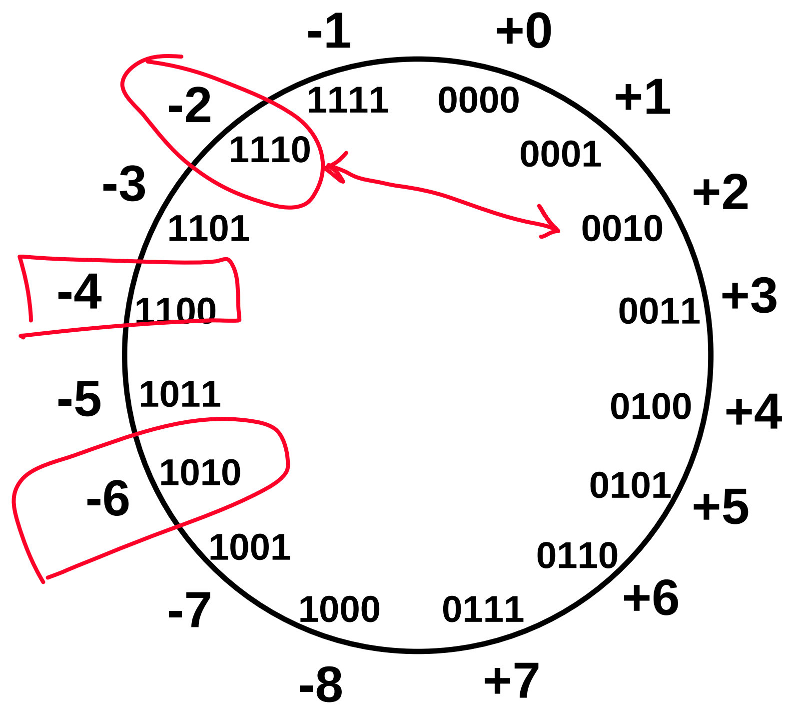
Flip bits & Add 1

Negate $(0010)_2$ (+2)

$$
\begin{array}{r}
1101 \\
+\quad 1 \\
\hline
1110
\end{array}
= (-2)_{10}
$$

Negate $(1110)_2$ (-2)

$$
\begin{array}{r}
0001 \\
+\quad 1 \\
\hline
0010
\end{array}
= (2)_{10}
$$



Addition in Two's Complement

$$
\begin{array}{rccccl}
 & 0 & 0 & 1 & 0 & (+2) \\
+ & 0 & 1 & 0 & 0 & (+4) \\
\hline
 & 0 & 1 & 1 & 0 & = (6)_{10}
\end{array}
\qquad\qquad
\begin{array}{rcccccl}
 & & 1 & & & & \\
 & & 1 & 1 & 1 & 0 & (-2) \\
+ & & 1 & 1 & 0 & 0 & (-4) \\
\hline
 & 1 & \multicolumn{4}{c}{\underbrace{1\ \ 0\ \ 1\ \ 0}} & = (-6)_{10}
\end{array}
$$

$$
\begin{array}{rccccl}
 & 0 & 0 & 1 & 0 & (+2) \\
+ & 1 & 1 & 0 & 0 & (-4) \\
\hline
 & 1 & 1 & 1 & 0 & = (-2)_{10}
\end{array}
\qquad\qquad
\begin{array}{rcccccl}
 & & 1 & & & & \\
 & & 1 & 1 & 1 & 0 & (-2) \\
+ & & 0 & 1 & 0 & 0 & (+4) \\
\hline
 & 1 & \multicolumn{4}{c}{\underbrace{0\ \ 0\ \ 1\ \ 0}} & = (2)_{10}
\end{array}
$$

Subtraction in Two's Complement

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

0010 - 0110

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

$$\begin{array}{r} 0010 \\ -\ 0110 \end{array} \xrightarrow{\text{flip}} \begin{array}{r} 0010 \\ 1001 \\ +1 \end{array} \rightarrow \begin{array}{r} 0010 \\ 1010 \\ \hline 1100 \end{array} = (-4)_{10}$$

1011 - 1001

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

$(+2)_{10}$

$$\begin{array}{r} 1011 \\ -\ 1001 \end{array} = \begin{array}{r} 1011 \\ +\ 0111 \\ \hline 1\,0010 \end{array} = (+2)_{10}$$

1011 - 0001

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

$(-6)_{10}$

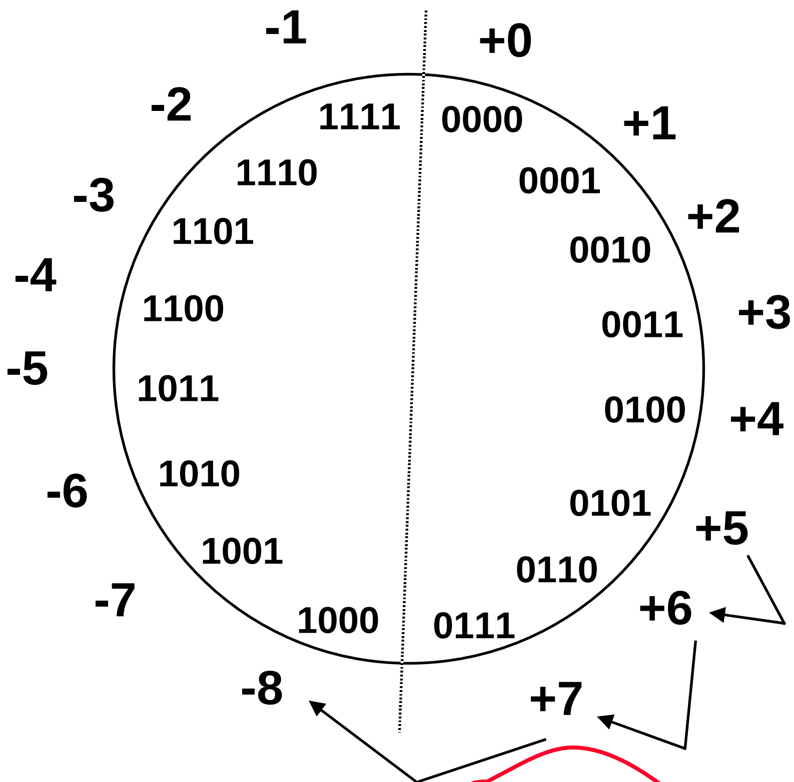
$$= \begin{array}{r} 1011 \\ -\ 0001 \end{array} = \begin{array}{r} 1011 \\ +\ 1111 \\ \hline 1\,1010 \end{array} = (-6)_{10}$$

4

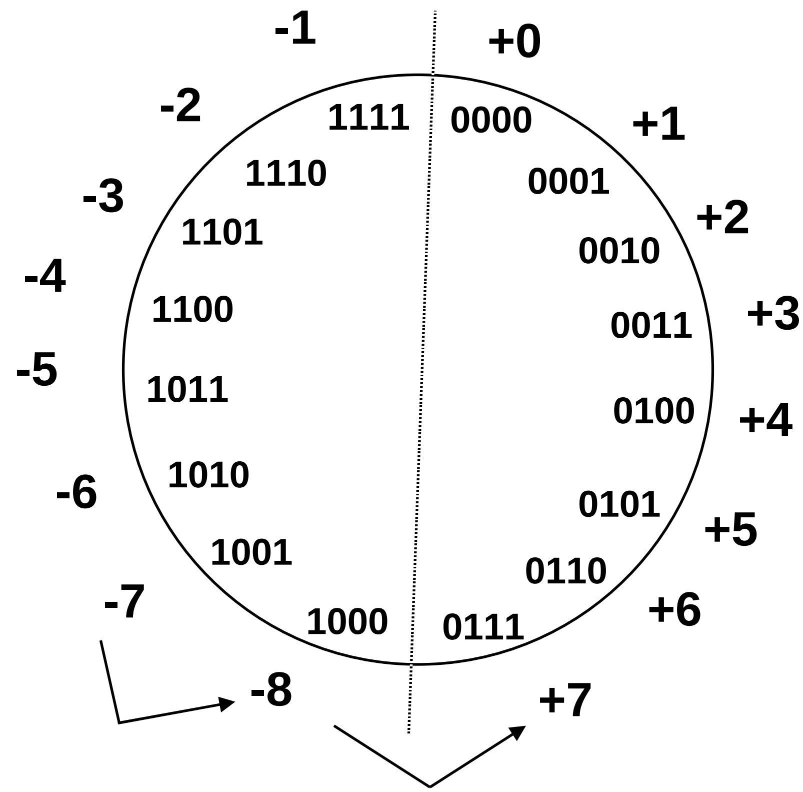
Overflows in Two's Complement

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

Overflow

	0 0 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

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

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

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

Convert $(01011)_2$ to Decimal

$$01011_2 = 11_{10}$$

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

$(1011)_2 \rightarrow (1111\ 1011)_2$

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

$(0010)_2 \rightarrow (0000\ 0010)_2$