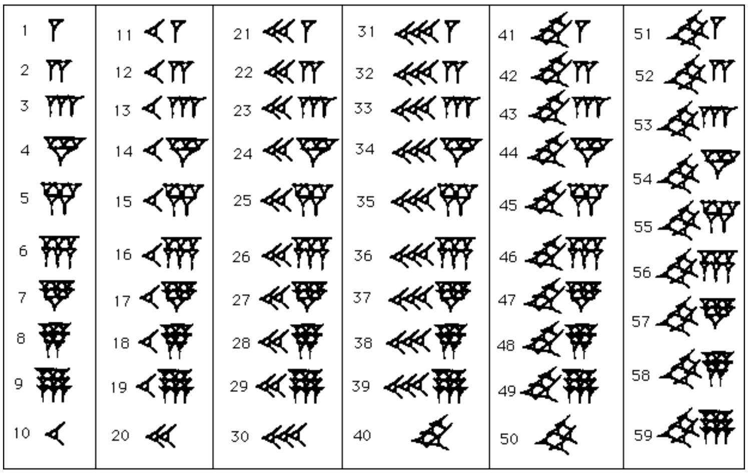
Switching Circuits & Logic Design

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Fall 2013

§1 Number Systems and Conversion



Babylonian number system (3100 B.C.)

Reference: http://www.math.ubc.ca/~cass/courses/m446-03/

Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
  - Negative number representation
    - Signed magnitude
    - 1's complement
    - 2's complement
    - Arithmetic under 1's and 2's complements
- Binary codes

Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
  - Negative number representation
    - Signed magnitude
    - 1's complement
    - 2's complement
    - Arithmetic under 1's and 2's complements
- Binary codes

Digital Systems and Switching Circuits

- Given some inputs, a digital system produces some outputs
  - It may or may not have "memory" (i.e., its outputs may or may not depend on the *history* of the inputs)

x_i and y_j are often binary, i.e., x_i, y_j $\in \{0,1\}$

Digital Systems and Switching Circuits

- Given a specification constraining the input and output relation of a digital system, we'll learn how to design a logic circuit that implements the specification

Digital Systems and Switching Circuits

- A digital system usually contains **datapath** and **control** parts
  - Datapath part
    - performs arithmetic operations, data transfer, etc.
      - Unit 1 introduces digital *data representation* and *arithmetic operation*
  - Control part
    - controls when and where data should be ready in the datapath
      - Later we'll study *combinational* and *sequential circuits* for control logic

Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
  - Negative number representation
    - Signed magnitude
    - 1's complement
    - 2's complement
    - Arithmetic under 1's and 2's complements
- Binary codes

Number Systems and Conversion

- Decimal number (base 10):
  - 953.78_{10}
    $= 9\times10^2 + 5\times10^1 + 3\times10^0 + 7\times10^{-1} + 8\times10^{-2}$
- Binary number (base 2):
  - 1011.11_2
    $= 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0 + 1\times2^{-1} + 1\times2^{-2}$
    $= 11.75_{10}$
- Octal number (base 8):
  - 147.3_8
    $= 1\times8^2 + 4\times8^1 + 7\times8^0 + 3\times8^{-1}$
    $= 103.375_{10}$
- Hexadecimal number (base 16):
  - $A2F_{16}$
    $= 10\times16^2 + 2\times16^1 + 15\times16^0$
    $= 2607_{10}$

The decimal, binary, octal, and hexadecimal points separate positive and negative powers of 10, 2, 8, 16, respectively.

Number Systems and Conversion Generalized Radix System

- Any integer R>1 can be chosen as the **radix** or **base**
  - Base (radix) R: (0, 1, ⋯, R-1), where R>1
    E.g.,
    - Base 5: (0,1,2,3,4)
    - Base 16: (0, 1, 2, ..., 9, A, B, C, D, E, F)

- $N_{(10)}$
  $= (a_4\ a_3\ a_2\ a_1\ a_0\ .\ a_{-1}\ a_{-2}\ a_{-3})_R$
  $= a_4\times R^4 + a_3\times R^3 + a_2\times R^2 + a_1\times R^1 + a_0\times R^0 + a_{-1}\times R^{-1} + a_{-2}\times R^{-2} + a_{-3}\times R^{-3}$
  Note that R is in base 10!

  E.g.,
  - $A2F_{16}$
    $= 10\times16^2 + 2\times16^1 + 15\times16^0$
    $= 2607_{10}$

Number Systems and Conversion
Generalized Radix System

Name	Decimal	Binary	Octal	Hexadecimal
Base	10	2	8	16
Digits	0,1,2,3,4, 5,6,7,8,9	0,1	0,1,2,3,4, 5,6,7	0,1,2,3,4,5,6,7, 8,9,A,B,C,D,E,F
Numbers	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	0 1 10 11 100 101 110 111 1000 1001 1010 1011 1100 1101 1110 1111 10000	0 1 2 3 4 5 6 7 10 11 12 13 14 15 16 17 20	0 1 2 3 4 5 6 7 8 9 A B C D E F 10

Number Systems and Conversion
Integer Conversion by Division

- $N = (a_n a_{n-1} \cdots a_2 a_1 a_0)_R$
  $= a_n \times R^n + a_{n-1} \times R^{n-1} + \cdots + a_2 \times R^2 + a_1 \times R^1 + a_0 \times R^0$
- $N/R = a_n \times R^{n-1} + a_{n-1} \times R^{n-2} + \cdots + a_2 \times R^1 + a_1 = Q_1$
  remainder a_0
- $Q_1/R = a_n \times R^{n-2} + a_{n-1} \times R^{n-3} + \cdots + a_3 \times R^1 + a_2 = Q_2$
  remainder a_1
- $Q_2/R = a_n \times R^{n-3} + a_{n-1} \times R^{n-4} + \cdots + a_4 \times R^1 + a_3 = Q_3$
  remainder a_2

Number Systems and Conversion Exercise

- Convert 53_{10} to binary

$$
\begin{array}{r|l l}
2 & 53 & \\
2 & 26 & \text{... remainder } 1 = a_0 \\
2 & 13 & \text{... remainder } 0 = a_1 \\
2 & 6 & \text{... remainder } 1 = a_2 \\
2 & 3 & \text{... remainder } 0 = a_3 \\
2 & 1 & \text{... remainder } 1 = a_4 \\
 & 0 & \text{... remainder } 1 = a_5
\end{array}
$$

So $53_{10} = (a_5\, a_4\, a_3\, a_2\, a_1\, a_0)_2 = 110101_2$

Number Systems and Conversion Fraction Conversion by Multiplication

- $F = (.a_{-1}a_{-2}a_{-3} \cdots a_{-m})_R$
  $= a_{-1}\times R^{-1} + a_{-2}\times R^{-2} + a_{-3}\times R^{-3} + \cdots + a_{-m}\times R^{-m}$
- $F\cdot R = a_{-1} + a_{-2}\times R^{-1} + a_{-3}\times R^{-2} + \cdots + a_{-m}\times R^{-m+1}$
  $= a_{-1} + F_1$
- $F_1\cdot R = a_{-2} + a_{-3}\times R^{-1} + \cdots + a_{-m}\times R^{-m+2}$
  $= a_{-2} + F_2$
- $F_2\cdot R = a_{-3} + \cdots + a_{-m}\times R^{-m+3}$
  $= a_{-3} + F_3$

Number Systems and Conversion Exercise

- Convert 0.625_{10} to binary

F = .625	F1 = .250	F2 = .500
×2	×2	×2
(1).250	(0).500	(1).000
a_{-1}	a_{-2}	a_{-3}

So $0.625_{10} = (.\,a_{-1}\,a_{-2}\,a_{-3})_2 = 0.101_2$

Number Systems and Conversion Exercise

- Convert 0.7_{10} to binary

	0 .7
	× 2
a_{-1}	(1).4
	× 2
a_{-2}	(0).8
	× 2
a_{-3}	(1).6
	× 2
a_{-4}	(1).2
	× 2
a_{-5}	(0).4
	× 2
a_{-6}	(0).8

repeat

So $0.7_{10} = (0.1\,\underline{0110}\,\underline{0110}\,\underline{0110}\,\ldots)_2$
Not finite conversion!

Number Systems and Conversion Exercise

- Convert 231.3_4 to base 7

$231.3_4 = 2\times4^2 + 3\times4^1 + 1\times4^0 + 3\times4^{-1} = 45.75_{10}$

7 ⌋ 45
7 ⌋ 6 ... remainder 3 = a_0
0 ... remainder 6 = a_1

0 .75
× 7
(5).25 ← a_{-1}
× 7
(1).75 ← a_{-2}

repeat

So $231.3_4 = (63.\underline{51}\ \underline{51}\ \ldots)_7$

Decimal system is useful as an intermediate representation for such conversion (though not necessary)

Number Systems and Conversion Exercise

- Conversion from binary to Octal

101110.0011_2
$= \underline{101}\ \underline{110}.\underline{001}\ \underline{100}_2$
$= 56.14_8$

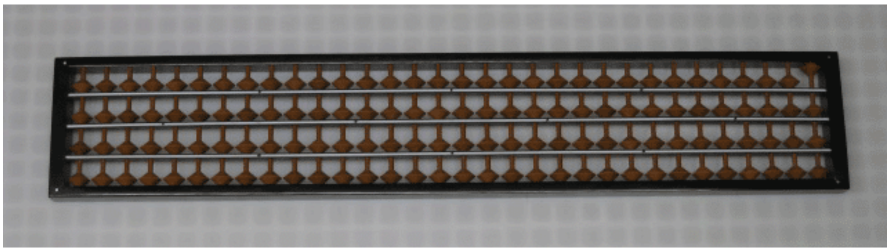
- Conversion from binary to Hexadecimal

1001101.010111_2
$= \underline{0100}\ \underline{1101}.\underline{0101}\ \underline{1100}_2$
$= 4D.5C_{16}$

Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
  - Negative number representation
    - Signed magnitude
    - 1's complement
    - 2's complement
    - Arithmetic under 1's and 2's complements
- Binary codes

Binary Arithmetic



extracting the square root of 1/2 on a binary abacus

Photo: www.binaryabacus.com

Binary Arithmetic
Addition

- Addition
  - 0 + 0 = 0
  - 0 + 1 = 1
  - 1 + 0 = 1
  - 1 + 1 = 0 (carry 1 to next column)

	1 1 1 1	carry
13_{10} =	$1\ 1\ 0\ 1_2$	
11_{10} =	$1\ 0\ 1\ 1_2$	
	$1\ 1\ 0\ 0\ 0_2 = 24_{10}$	

Binary Arithmetic
Subtraction

- Subtraction
  - 0 – 0 = 0
  - 1 – 0 = 1
  - 1 – 1 = 0
  - 0 – 1 = 1 (borrow 1 from next column)

	1	borrow
	$1\ 1\ 1\ 0\ 1_2$	
–	$1\ 0\ 0\ 1\ 1_2$	
	$1\ 0\ 1\ 0_2$	

Binary Arithmetic
Subtraction

- More examples

$$\begin{array}{rcl} & 1\ 1\ 1\ 1 & \text{borrows} \\ 16_{10} = & 1\ 0\ 0\ 0\ 0_2 & \\ -\ 3_{10} = & 1\ 1_2 & \\ \hline & 1\ 1\ 0\ 1_2 = 13_{10} & \end{array}$$

$$\begin{array}{rcl} & 1\ 1\ 1 & \text{borrows} \\ 57_{10} = & 1\ 1\ 1\ 0\ 0\ 1_2 & \\ -\ 11_{10} = & 1\ 0\ 1\ 1_2 & \\ \hline & 1\ 0\ 1\ 1\ 1\ 0_2 = 46_{10} & \end{array}$$

Binary Arithmetic
Multiplication

- Multiplication
  - $0 \times 0 = 0$
  - $0 \times 1 = 0$
  - $1 \times 0 = 0$
  - $1 \times 1 = 1$

$$\begin{array}{rrl} 13_{10} = & 1\ 1\ 0\ 1_2 & \text{multiplicand} \\ \times\ 11_{10} = & 1\ 0\ 1\ 1_2 & \text{multiplier} \\ \hline & 1\ 1\ 0\ 1 & \\ & 1\ 1\ 0\ 1\ \ \ & \\ & 0\ 0\ 0\ 0\ \ \ \ \ \ & \\ & 1\ 1\ 0\ 1\ \ \ \ \ \ \ \ \ & \\ \hline & 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1_2 = 143_{10} & \text{product} \end{array}$$

Binary Arithmetic
Division

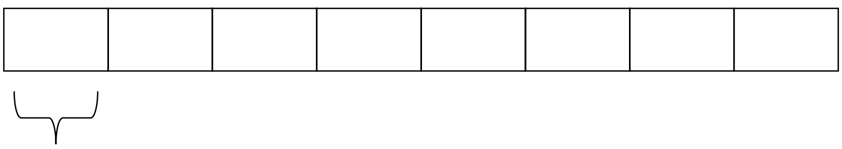
- Division of 145_{10} by 11_{10} in binary

```
                1 1 0 1      quotient
        ___________________
1 0 1 1 /  1 0 0 1 0 0 0 1
           1 0 1 1
           -------
             1 1 1 0
             1 0 1 1
             -------
                 1 1 0 1
                 1 0 1 1
                 -------
                     1 0     remainder
```


Negative Number Representation

- The 1st bit (most significant bit, MSB) is reserved for sign indication
  - Assume base-2 system

An 8-bit **word** example

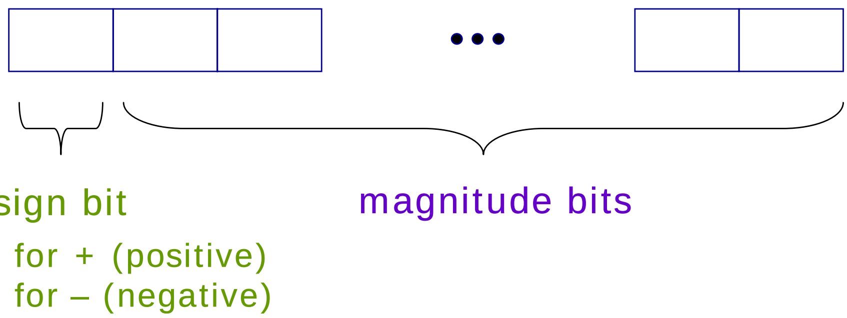


sign bit   0 for + (positive)
            1 for – (negative)

(Modern computers are of word size 64 or 32 bits.)

Negative Number Representation
Signed Magnitude Numbers

- $N^{sm} = (a_{n-1}a_{n-2} \cdots a_1a_0)_2$
  $= (-1)^{a_{n-1}} \cdot (a_{n-2}\times 2^{n-2} + \cdots + a_1\times 2^1 + a_0\times 2^0)$



E.g., -12 = 11100 in a 5-bit word = 101100 in a 6-bit word

Negative Number Representation
1's Complement

- For an n-bit word, the negative number –N of N in the 1's complement is
  $N^{(1)} = (2^n - 1) - N$
  - Both N and –N are in the 1's complement (N can be positive/negative)
  - $N^{(1)}$ is the bitwise complement of N
  - $N = (2^n - 1) - N^{(1)}$

E.g., n = 6 and N = 010101 (= 21)

	$(2^n - 1)$	= 111111
–	N	= 010101
	$N^{(1)}$	= 101010 (= –21)

Negative Number Representation
2's Complement

- For an n-bit word, the negative number –N of N in the 2's complement is
  $N^{(2)} = 2^n - N = N^{(1)} + 1$
  - Both N and –N are in the 2's complement (N can be positive/negative)
    - Mind the marginal case when N = 0 and N = -2^{n-1}
  - $N^{(2)}$ is the bitwise complement of N plus 1
    - Effectively, only complementing every bit of N starting from the MSB until the last bit with value 1 (not applicable when N = 0)

E.g., n = 6, N = 001110 (14)

↑ last bit with value 1

$\Rightarrow N^{(2)}$ = 110001 + 000001 = 110010 (-14)

Negative Number Representation
2's Complement

- The value of $N^{(2)}$ can be obtained from N

E.g., n = 6

$N^{(2)}$ = 101010 (?) $\Rightarrow$ N = 010110 (22) $\Rightarrow$ $N^{(2)}$ represents the negative number –22

$N^{(2)}$ = 100000 (–32) $\Rightarrow$ no corresponding N!

N = 000000 (0) $\Rightarrow$ no corresponding $N^{(2)}$!

Negative Number Representation

- The signed magnitude, 1's complement, and 2's complement systems all have the same representation for any nonnegative number
  - For a negative number, we need to specify which system we use (but not for a positive number)

E.g., n = 6

$N^{(1)}, N^{(2)}, N^{sm} = 011111$ (31)

$\Rightarrow N = 011111$ (31)

Negative Number Representation

Signed binary integers (word length: n= 4)

+N	Positive integers **(same for all systems!)**	-N	Negative integers		
			Sign and magnitude N^{sm}	2's complement $N^*, N^{(2)}$	1's complement $\overline{N}, N^{(1)}$
+0	**0000**	-0	1000	-----	1111
+1	0001	-1	1001	1111	1110
+2	0010	-2	1010	1110	1101
+3	0011	-3	1011	1101	1100
+4	0100	-4	1100	1100	1011
+5	0101	-5	1101	1011	1010
+6	0110	-6	1110	1010	1001
+7	0111	-7	1111	1001	1000
		-8	-----	**1000**	-----

Negative Number Representation
Arithmetic with 2's Complement

- Subtraction is replaced with addition
- Addition is computed as if its two operands were positive numbers
  - Carry from the sign bit is ignored (bit-length preserved)

```
E.g.,   –5      1011
        +6      0110
        ––    –––––––
        +1   (1)0001    discard the carry
```

  - Result is correct when no **overflow** occurs (why?)
    - Overflow occurs when
      1. adding two positive numbers yields a negative number, or
      2. adding two negative numbers yields a positive number.

```
E.g.,   +5    0101           –5      1011
        +6    0110           –6      1010
        ––   ––––––          ––    –––––––
              1011                 (1)0101
  wrong! (overflow)       wrong! (overflow)
```


Negative Number Representation
Arithmetic with 2's Complement

A	B	A+B	overflow
+	+	+	no
+	+	–	yes
+	–	+/–	no
–	+	+/–	no
–	–	+	yes
–	–	–	no

Checking sign-bits for overflow detection

Negative Number Representation Arithmetic with 1's Complement

- Similar to the addition of 2's complement except for having end-around carry (instead of discarding the last carry)
  - Add the last carry to the sum

E.g.,

```
  –5        1010
  +6        0110
  ----    --------
          (1) 0000
           └─────→ 1   (end-around carry)
          --------
              0001

  +5     0101              –5        1010
  +6     0110              –6        1001
  ----   ----              ----   ---------
         1011                     (1) 0011
wrong! (overflow)                  └────→ 1
                                  ---------
                                      0100   wrong! (overflow)
```


Outline

- Digital systems and switching circuits
- Number systems and conversion
- Binary arithmetic
  - Negative number representation
    - Signed magnitude
    - 1's complement
    - 2's complement
    - Arithmetic under 1's and 2's complements
- Binary codes

Binary Codes

Photo: datapeak.net

Binary Codes

- Use binary codes to represent *symbols*
  - Not associated with arithmetic
  - Different from the *binary number system*

E.g.,

3.1415

0011 . 0001 0100 0001 0101

in Binary Coded Decimal (BCD)

Different from converting the number as a whole into binary!

Binary Codes

Common binary codes for decimal digits

Decimal digit	8-4-2-1 code (BCD)	6-3-1-1 code	Excess-3 code (BCD+3)	2-out-of-5 code (good for error checking)	Gray code (good for low power and reliability)
0	0000	0000	0011	00011	0000
1	0001	0001	0100	00101	0001
2	0010	0011	0101	00110	0011
3	0011	0100	0110	01001	0010
4	0100	0101	0111	01010	0110
5	0101	0111	1000	01100	1110
6	0110	1000	1001	10001	1010
7	0111	1001	1010	10010	1011
8	1000	1011	1011	10100	1001
9	1001	1100	1100	11000	1000

$a_3a_2a_1a_0$ in w_3-w_2-w_1-w_0 (4-bit weighted) code represents $N = w_3a_3+w_2a_2+w_1a_1+w_0a_0$
E.g., 1011 in 6-3-1-1 code represents $N = 6{\cdot}1+3{\cdot}0+1{\cdot}1+1{\cdot}1=8$

Binary Codes

- ASCII code
  - 7 bits
  - Used to represent computer characters

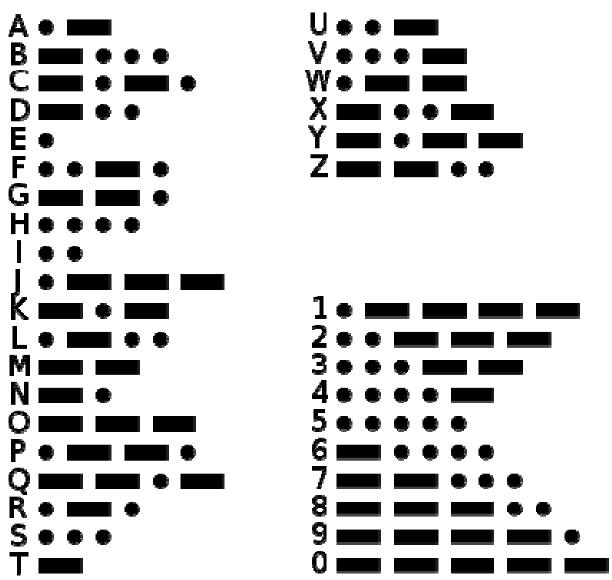
Binary Codes

ASCII printable characters (decimal code, character)					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	delete

Other Binary Codes

International Morse Code

1. A dash is equal to three dots.
2. The space between parts of the same letter is equal to one dot.
3. The space between two letters is equal to three dots.
4. The space between two words is equal to seven dots.



Other Binary Codes