Introduction to Digital Design

Week 2: Number System

Yao Zheng
Assistant Professor
University of Hawai'i at Mānoa
Department of Electrical Engineering

Converting from Binary to Decimal

- Just add weights
  - 1_2 is just $1*2^0$, or 1_{10}.
  - 110_2 is $1*2^2 + 1*2^1 + 0*2^0$, or 6_{10}. We might think of this using base ten weights: 1*4 + 1*2 + 0*1, or 6.
  - 10000_2 is 1*16 + 0*8 + 0*4 + 0*2 + 0*1, or 16_{10}.
  - 10000111_2 is 1*128 + 1*4 + 1*2 + 1*1 = 135_{10}. Notice this time that we didn't bother to write the weights having a 0 bit.
  - 00110_2 is the same as 110_2 above — the leading 0's don't change the value.

Useful to know powers of 2:

2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
512	256	128	64	32	16	8	4	2	1

Practice counting up by powers of 2: 512 256 128 64 32 16 8 4 2 1

Converting from Decimal to Binary

- Put 1 in leftmost place without sum exceeding number
- Track sum

	Desired decimal number: **12**	Current sum	Binary number (16 8 4 2 1)
(a)	16 > 12, too big; Put 0 in 16's place	0	0 _ _ _ _
(b)	8 <= 12, so put 1 in 8's place, current sum is 8	8	0 1 _ _ _
(c)	8+4=12 <= 12, so put 1 in 4's place, current sum is 12	12	0 1 1 _ _
(d)	Reached desired 12, so put 0s in remaining places	*done*	0 1 1 0 0

Converting from Decimal to Binary

- Example using a more compact notation

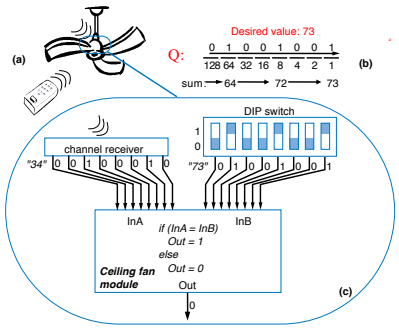
Desired decimal number: **23**

Binary number (16 8 4 2 1)

sum: 0 → 16 **(a)** → 16 **(b)** → 20 **(c)** → 22 **(d)** → 23 **(e)**

Example: DIP-Switch Controlled Channel

- Ceiling fan receiver should be set in factory to respond to channel "73"
- Convert 73 to binary, set DIP switch accordingly



Base Sixteen: Another Base Used by Designers

16^4	16^3	16^2	16^1	16^0
		8	A	F
		1000	1010	1111

hex	bina ry	hex	bina ry
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

- Nice because each position represents four base-two positions
  - Compact way to write binary numbers
- Known as ***hexadecimal***, or just ***hex***

Q: Write 11110000 in hex → F 0

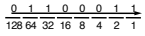
Q: Convert hex A01 to binary → 1010 0000 0001

Decimal to Hex

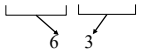
- Easy method: convert to binary first, then binary to hex

Convert 99 base 10 to hex

First convert to binary:



Then binary to hex:

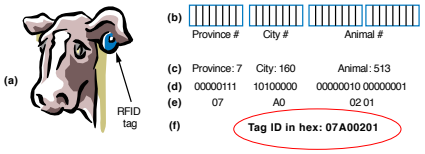


(Quick check: 6*16 + 3*1 = 96+3 = 99)

7

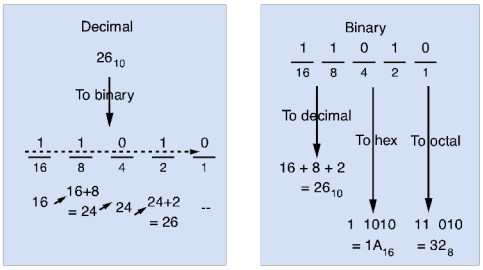
Hex Example: RFID Tag

- Batteryless tag powered by radio field
  - Transmits unique identification number
  - Example: 32 bit id
    - 8-bit province number, 8-bit country number, 16-bit animal number
    - Tag contents are in binary
    - But programmers use hex when writing/reading



8

Converting To/From Binary by Hand: Summary



9

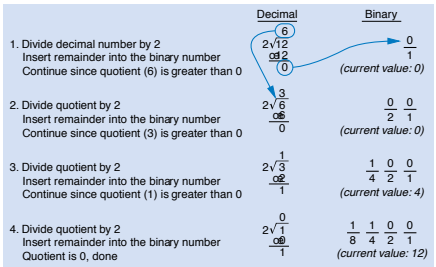
Example Video: Convert from Hex to Binary

- Video link.

10

Divide-By-2 Method Common in Automatic Conversion

- Repeatedly divide decimal number by 2, place remainder in current binary digit (starting from 1s column)



Note: Works for any base N—just divide by N instead

11

Example Video: Convert from Decimal to Binary

- Video link.

12

Bytes, Kilobytes, Megabytes, and More

- Byte: 8 bits 
- Common metric prefixes:
  - kilo (thousand, or 10^3), mega (million, or 10^6), giga (billion, or 10^9), and tera (trillion, or 10^{12}), e.g., kilobyte, or KByte
- BUT, metric prefixes also commonly used inaccurately
  - 2^{16} = 65536 commonly written as "64 Kbyte"
  - Typical when describing memory sizes
- Also watch out for "KB" for kilobyte vs. "Kb" for kilobit

13

Addition and Subtraction of Nondecimal Numbers

- Addition

```
            101111000
X     190   10111110
Y     141   10001101
___+_____________
X+Y   331 101001011
```

C_{in}/b_{in}	X	Y	C_{out}	S	B_{out}	D
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	1	0	1
0	1	1	1	0	0	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

14

Binary Arithmetic Operations: Addition

- Follow same rules as in decimal addition, with the difference that when sum is 2 indicates a carry (not a 10)
- Learn new carry rules
  - 0+0 = 0c0 (sum 0 with carry 0)
  - 0+1 = 1+0 = 1c0
  - 1+1 = 0c1
  - 1+1+1 = 1c1

Carry	1	1	1	1	1	0
Augend	0	0	1	0	0	1
Addend	0	1	1	1	1	1
Result	1	0	1	0	0	0

15

Binary Arithmetic Operations: Addition (cont.)

- "Half addition" (rightmost bit position, aka LSB): only 2 bits are added, yielding a sum and a carry
- "Full addition" (remaining positions): three bits are added, yielding a sum and a carry
- In Chapter 8, we'll see many different hardware implementations of half-adders and full-adders

16

Binary Arithmetic Operations: Subtraction

- Learn new borrow rules
  - 0-0 = 1-1 = 0b0 (result 0 with borrow 0)
  - 1-0 = 1b0
  - 0-1 = 1b1
  - …

Borrow	1	1	0	0	
Minuend	1	1	0	1	1
Subtrahend	0	1	1	0	1
Result	0	1	1	1	0

17

- Binary subtraction: (borrow, difference bits)

```
Bout                  001111100
Minuend     X   229    11100101
Subtrahend  Y    46    00101110
_____________-_________________
Difference  X-Y 183    10110111
```

  - Use binary subtraction to compare numbers. If X-Y produces a borrow out at the most significant bit, then X is less than Y.

18

Representation of Negative Numbers

- Signed-Magnitude System/Representation:
  - MSB: sign bit, 0: plus, 1: minus

  $01010101_2 = +85_{10}$ $11010101_2 = -85_{10}$

  $01111111_2 = +127_{10}$ $11111111_2 = -127_{10}$

  $00000000_2 = +0_{10}$ $10000000_2 = -0_{10}$
  - n-bit signed integer lies within $-(2^{n-1}-1)$ through $+(2^{n-1}-1)$ with two representations of zero.

19

- Signed-magnitude adder
  - If signs are same
    add magnitudes, sign is same

    else
    compare magnitudes,
    subtract smaller from the larger,
    sign is sign of larger
- Signed-magnitude subtractor
  - Change the sign of subtrahend, perform addition

20

Complement Number Systems

- Taking the complement is more difficult than changing the sign, but in complement system add/subt are easier.
- Radix complement of D: $(-D) = r^n - D$
  where D is an n-digit number
  - If D is between 1 and r^n-1
    then D complement is between r^n-1 and $r^n-(r^n-1) = 1$
  - When D=0, D complement is r^n, which is (n+1) digits, hence D complement is also 0.

21

Two's Complement

- Radix complement for binary numbers
  D: n bit binary number
  $(-D) = 2^n - D = (2^n-1) - D + 1$

  $1710 = 00010001_2$ 0000_2

  11101110 1111

  $+\quad\quad 1$ $+\quad 1$

  $11101111_2 = -17_{10}$ 10000_2
  - The range of representable numbers: $-(2^{n-1}) - (2^{n-1}-1)$
  - A number is negative iff MSB is 1.
  - Zero (0) is positive, one extra negative number.
  - Decimal equivalent of two's complement number is computed in the same way for unsigned except MSB is (-2^{n-1}), not (2^{n-1}).
  - Sign extension property: extend MSB for (n+1) bit number from n bit number: 0110 = 00110 (5 bit), 1010 = 11010 (5 bit)

22

One's Complement Representation

- One's complement of D
  $-D = 2^n - 1 - D$

  $17_{10} = 00010001_2$

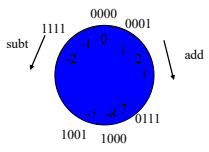
  $11101110_2 = -17_{10}$

23

Two's complement addition

-3	1101	-2	1110	-1	1111
+1	0001	+1	0001	+1	0001
-2	1110	-1	1111	0	10000

- Subtract n (add –n)
  - Add 16-n
  - 16-n is 4-bit two's complement of n, that is (-n).



24

Two's complement subtraction

```
                1 cin                          1
  +4   0100   0100        +3   0011   0011
- +3   0011   1100      - -4   1100   0011
  +1          10001        +7          0111
```

- Overflow
  - If the result exceeds the range of the number system

```
  +5   0101
+ +6   0110
  +11  1011 = -5
```

Overflow rule:
Add^n: if the sign of the addends' are same and the sign of the sum is different from addends' sign.

25

Binary Multiplication

- Unsigned binary multiplication
  - Shift and add multiplication

```
   11   1011
x 13   1101
      01011
     00000
    01011
   1011
  10001111
```

26

- Two's complement multiplication
  - Shift and two's complement addition except for the last step. Remember MSB represent (-2^{n-1})

```
   -5   1011
x –3   1101
       0000  initial partial product, which is zero.
      1011
      11011 partial product
     0000
     111011 partial product
    1011
    11100111
     0101     shifted-and-negated
   1 00001111
```

27

Binary division

- Shift and subtract with unsigned numbers
- No other easy way.
- So for signed numbers, take care of signs and perform shift-and-subtract

28

BCD: Binary-Coded Decimal

- 0-9 encoded with their 4-bit unsigned binary representation (0000 – 1001). The codewords (1010 – 1111) are not used.
- 8-bit byte represent values from 0 to 99.
- BCD Addition:

```
                Carry          1      1
  448                     0100   0100   1000
+489                      0100   1000   1001
  937   Sum               1001   1101   10001
        Add 6                    + 0110 + 0110
        BCD sum                  1 0011 1 0111
        BCD result        1001   0011   0111
```

29

GRAY CODES

- As we count up and down using binary codes, the number of bits that change from one to another varies.
- For Gray Codes, only 1 bit changes.

Decimal	8,4,2,1	Gray
0	0000	0000
1	0001	0100
2	0010	0101
3	0011	0111
4	0100	0110
5	0101	0010
6	0110	0011
7	0111	0001
8	1000	1001
9	1001	1000

30

Summary

- Number systems use 0s and 1s
  - Conversion between different number system
  - Binary Arithmetic

31