Introduction to Digital Design

Week 2: Number System

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

Overview

- Common number systems
  - Common number systems include: Decimal, Binary, Octal, Hexadecimal.
- Base conversion
  - For convenience, people use other bases (like decimal, hexadecimal) and we need to know how to convert from one to another.
- Arithmetic
  - How to add, subtract, multiply, etc.
- Negative numbers
  - There are more than one way to express a number in binary. So 1010 could be -2, -5 or -6 and need to know which one.

Common Number Systems

System	Base	Symbols	Used by humans?	Used in computers?
Decimal	10	0, 1, … 9	Yes	No
Binary	2	0, 1	No	Yes
Octal	8	0, 1, … 7	No	No
Hexa-decimal	16	0, 1, … 9, A, B, … F	No	No

Quantities/Counting (1 of 3)

Decimal	Binary	Octal	Hexa-decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7

Quantities/Counting (2 of 3)

Decimal	Binary	Octal	Hexa-decimal
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Quantities/Counting (3 of 3)

Decimal	Binary	Octal	Hexa-decimal
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14
21	10101	25	15
22	10110	26	16
23	10111	27	17

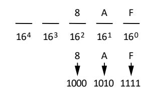
Etc.

Oct Example: Linux Permissions

- Linux default permissions
  - File (7), directory (6).
  - Each bit in the octal umasks "takes away" a permission.
- Octal umask values
  - **0** : read, write and execute
  - **1** : read and write
  - **2** : read and execute
  - **3** : read only
  - **4** : write and execute
  - **5** : write only
  - **6** : execute only
  - **7** : no permissions

Octal	Perms
0	rwx
1	rw-
2	r-x
3	r--
4	-wx
5	-w-
6	--x
7	---

Base Sixteen: Another Base Used by Designers



hex	binary	hex	binary
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

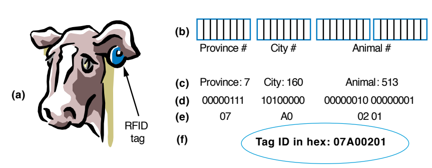
F 0

Q: Convert hex A01 to binary

1010 0000 0001

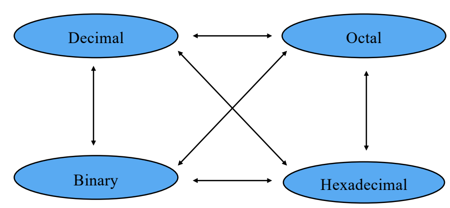
Hex Example: RFID Tag

- 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



Conversion Among Bases

- The possibilities:

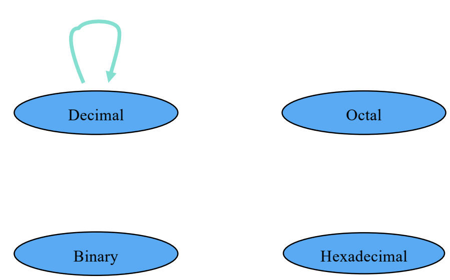


Quick Example

$$25_{10} = 11001_2 = 31_8 = 19_{16}$$

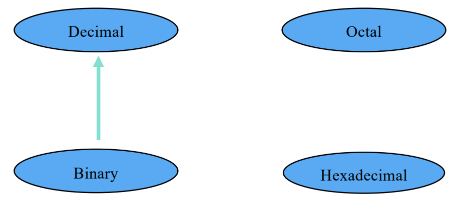
Base

Decimal to Decimal (just for fun)



```
                Weight
125₁₀ =>    5 x 10⁰     =    5
            2 x 10¹     =   20
            1 x 10²     =  100
                           ---
                           125
          Base
```


Binary to Decimal



Binary to Decimal

- Technique
  - Multiply each bit by 2^n, where *n* is the "weight" of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

Technique Explained

- 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

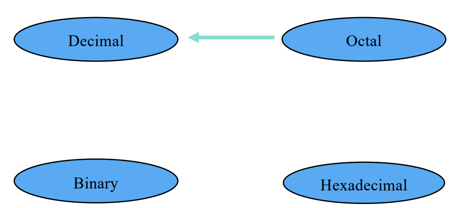
512 256 128 64 32 16 8 4 2 1

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

Example

```
Bit "0"
101011₂ =>   1 x 2⁰ =    1
             1 x 2¹ =    2
             0 x 2² =    0
             1 x 2³ =    8
             0 x 2⁴ =    0
             1 x 2⁵ =   32
                        ---
                        43₁₀
```


Octal to Decimal



Octal to Decimal

- Technique
  - Multiply each bit by 8^n, where n is the "weight" of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

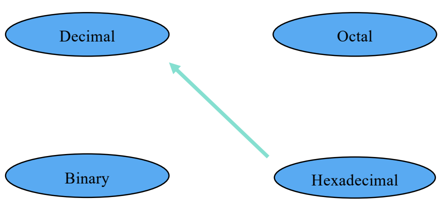
19

Example

$$
724_8 \Rightarrow \begin{array}{lr} 4 \times 8^0 = & 4 \\ 2 \times 8^1 = & 16 \\ 7 \times 8^2 = & \underline{448} \\ & 468_{10} \end{array}
$$

20

Hexadecimal to Decimal



21

Hexadecimal to Decimal

- Technique
  - Multiply each bit by 16^n, where n is the "weight" of the bit
  - The weight is the position of the bit, starting from 0 on the right
  - Add the results

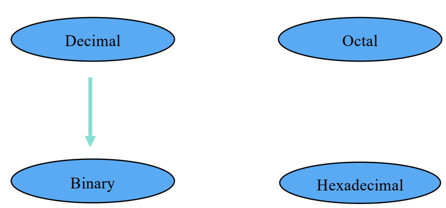
22

Example

$$
ABC_{16} \Rightarrow \begin{array}{lr} C \times 16^0 = 12 \times 1 = & 12 \\ B \times 16^1 = 11 \times 16 = & 176 \\ A \times 16^2 = 10 \times 256 = & \underline{2560} \\ & 2748_{10} \end{array}
$$

23

Decimal to Binary



24

Using Sum

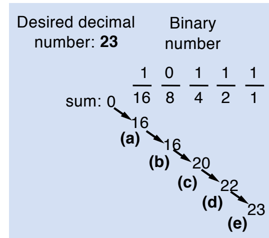
- Put 1 in leftmost place without sum exceeding number
- Track sum

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

25

Using Sum

- Example using a more compact notation



26

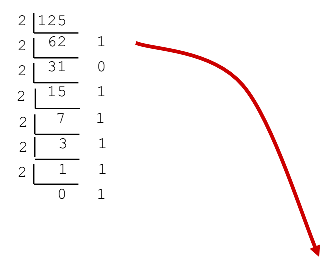
Using Long Division

- Technique
  - Divide by two, keep track of the remainder
  - First remainder is bit 0 (LSB, least-significant bit)
  - Second remainder is bit 1
  - Etc.

27

Example

$125_{10} = ?_2$

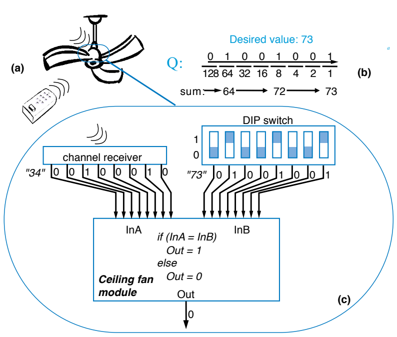


$125_{10} = 1111101_2$

28

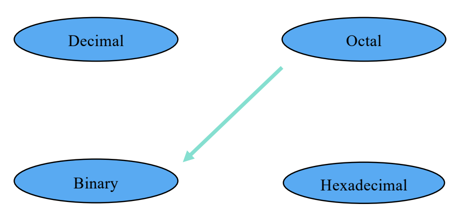
Example: DIP-Switch Controlled Channel

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



29

Octal to Binary



30

Octal to Binary

- Technique
  - Convert each octal digit to a 3-bit equivalent binary representation

31

Example

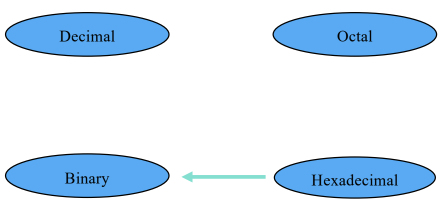
$705_8 = ?_2$

```
 7    0    5
 ↓    ↓    ↓
111  000  101
```

$705_8 = 111000101_2$

32

Hexadecimal to Binary



33

Hexadecimal to Binary

- Technique
  - Convert each hexadecimal digit to a 4-bit equivalent binary representation

34

Example

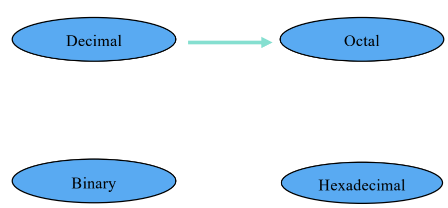
$10AF_{16} = ?_2$

```
  1     0     A     F
  ↓     ↓     ↓     ↓
0001  0000  1010  1111
```

$10AF_{16} = 0001000010101111_2$

35

Decimal to Octal



36

Decimal to Octal

- Technique
  - Divide by 8
  - Keep track of the remainder

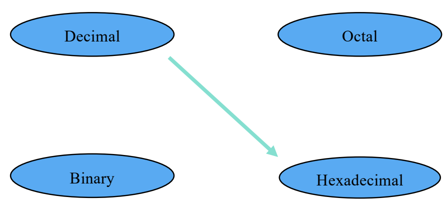
Example

$1234_{10} = ?_8$

```
8 |1234
8 | 154   2
8 |  19   2
8 |   2   3
      0   2
```

$1234_{10} = 2322_8$

Decimal to Hexadecimal



Decimal to Hexadecimal

- Technique
  - Divide by 16
  - Keep track of the remainder

Example

$1234_{10} = ?_{16}$

```
16 |1234
16 |  77   2
16 |   4   13 = D
       0   4
```

$1234_{10} = 4D2_{16}$

Decimal to Hexadecimal

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

Convert 99 base 10 to hex

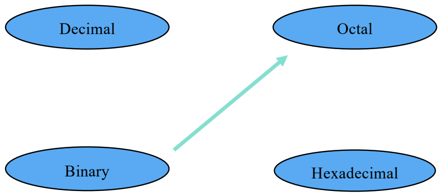
First convert to binary:

0	1	1	0	0	0	1	1
128	64	32	16	8	4	2	1

Then binary to hex: 6 3

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

Binary to Octal



43

Binary to Octal

- Technique
  - Group bits in threes, starting on right
  - Convert to octal digits

44

Example

$1011010111_2 = ?_8$

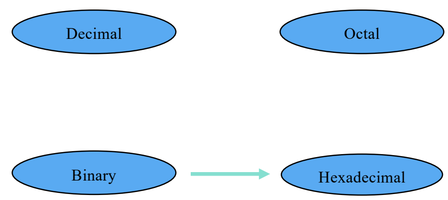
1 011 010 111

1 3 2 7

$1011010111_2 = 1327_8$

45

Binary to Hexadecimal



46

Binary to Hexadecimal

- Technique
  - Group bits in fours, starting on right
  - Convert to hexadecimal digits

47

Example

$1010111011_2 = ?_{16}$

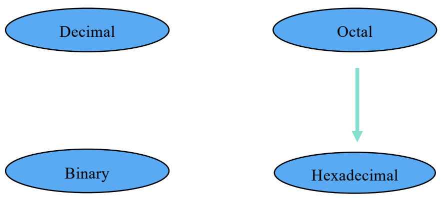
10 1011 1011

2 B B

$1010111011_2 = 2BB_{16}$

48

Octal to Hexadecimal



Octal to Hexadecimal

- Technique
  - Use binary as an intermediary

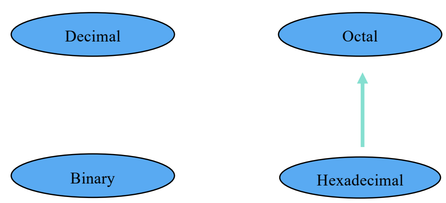
Example

$1076_8 = ?_{16}$

```
  1     0     7     6
 001   0|00  11|1  110
   2  |  3  |   E
```

$1076_8 = 23E_{16}$

Hexadecimal to Octal



Hexadecimal to Octal

- Technique
  - Use binary as an intermediary

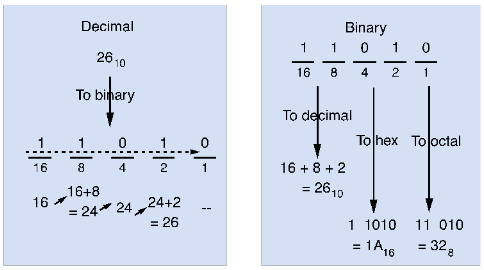
Example

$1F0C_{16} = ?_8$

```
   1      F      0      C
 0|001 | 111|1   00|00  1|100
 |  1  |  7 |  4   | 1  |  4
```

$1F0C_{16} = 17414_8$

Converting To/From Binary by Hand: Summary



55

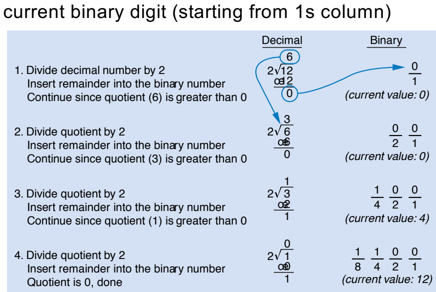
Example Video: Convert from Hex to Binary

- Video link.

56

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

57

Example Video: Convert from Decimal to Binary

- Video link.

58

Exercise – Convert …

Decimal	Binary	Octal	Hexa-decimal
33			
	1110101		
		703	
			1AF

Don't use a calculator!

59

Exercise – Convert …

Decimal	Binary	Octal	Hexa-decimal
33	100001	41	21
117	1110101	165	75
451	111000011	703	1C3
431	110101111	657	1AF

60

Common Powers (1 of 2)

- Base 10

Power	Preface	Symbol	Value
10^{-12}	pico	p	.000000000001
10^{-9}	nano	n	.000000001
10^{-6}	micro	µ	.000001
10^{-3}	milli	m	.001
10^{3}	kilo	k	1000
10^{6}	mega	M	1000000
10^{9}	giga	G	1000000000
10^{12}	tera	T	1000000000000

Common Powers (2 of 2)

- Base 2

Power	Preface	Symbol	Value
2^{10}	kilo	k	1024
2^{20}	mega	M	1048576
2^{30}	Giga	G	1073741824

- What is the value of "k", "M", and "G"?
- In computing, particularly w.r.t. memory, the base-2 interpretation generally applies

Review – multiplying powers

- For common bases, add powers

$$a^b \times a^c = a^{b+c}$$

$$2^6 \times 2^{10} = 2^{16} = 65{,}536$$

or…

$$2^6 \times 2^{10} = 64 \times 2^{10} = 64k$$

Binary Addition (1 of 2)

- Two 1-bit values

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	10

"two"

Binary Addition (2 of 2)

- Two *n*-bit values
  - Add individual bits
  - Propagate carries
  - E.g.,

```
 1   1
  10101      21
+ 11001    + 25
 101110      46
```


Multiplication (1 of 3)

- Decimal (just for fun)

```
   35
x 105
  175
 000
 35
 3675
```

Multiplication (2 of 3)

- Binary, two 1-bit values

A	B	A × B
0	0	0
0	1	0
1	0	0
1	1	1

67

Multiplication (3 of 3)

- Binary, two *n*-bit values
  - As with decimal values
  - E.g.,

```
      1110
    x 1011
      1110
     1110
    0000
   1110
  10011010
```

68

Fractions

- Decimal to decimal (just for fun)

$$
3.14 \Rightarrow \quad
\begin{aligned}
4 \times 10^{-2} &= 0.04\\
1 \times 10^{-1} &= 0.1\\
3 \times 10^{0} &= 3\\
&\quad 3.14
\end{aligned}
$$

69

Fractions

- Binary to decimal

$$
10.1011 \Rightarrow \quad
\begin{aligned}
1 \times 2^{-4} &= 0.0625\\
1 \times 2^{-3} &= 0.125\\
0 \times 2^{-2} &= 0.0\\
1 \times 2^{-1} &= 0.5\\
0 \times 2^{0} &= 0.0\\
1 \times 2^{1} &= 2.0\\
&\quad 2.6875
\end{aligned}
$$

70

Fractions

- Decimal to binary

```
3.14579             .14579
                   x      2
                    0.29158
                   x      2
                    0.58316
                   x      2
                    1.16632
                   x      2
                    0.33264
                   x      2
                    0.66528
                   x      2
                    1.33056
11.001001...        etc.
```

71

Exercise – Convert ...

Decimal	Binary	Octal	Hexa-decimal
29.8			
	101.1101		
		3.07	
			C.82

Don't use a calculator!

72

Exercise – Convert …

Decimal	Binary	Octal	Hexa-decimal
29.8	11101.110011…	35.63…	1D.CC…
5.8125	101.1101	5.64	5.D
3.109375	11.000111	3.07	3.1C
12.5078125	1100.10000010	14.404	C.82

73

Negative Numbers

- How do we write negative binary numbers?
- Historically: 3 approaches
  - Sign-and-magnitude
  - Ones-complement
  - Twos-complement
- For all 3, the most-significant bit (MSB) is the sign digit
  - 0 ≡ positive
  - 1 ≡ negative
- twos-complement is the important one
  - Simplifies arithmetic
  - Used almost universally

74

Sign-and-magnitude

- The most-significant bit (MSB) is the sign digit
  - 0 ≡ positive
  - 1 ≡ negative
- The remaining bits are the number's magnitude
- Problem 1: Two representations for zero
  - 0 = 0000 and also –0 = 1000

Add		Subtract			Compare and subtract		
4	0100	4	0100	0100	– 4	1100	1100
+ 3	+ 0011	– 3	+ 1011	– 0011	+ 3	+ 0011	– 0011
= 7	= 0111	= 1	≠ 1111	= 0001	– 1	≠ 1111	= 1001

75

Ones-complement

- Negative number: Bitwise complement positive number
  - 0011 ≡ 3_{10}
  - 1100 ≡ $–3_{10}$
- Solves the arithmetic problem

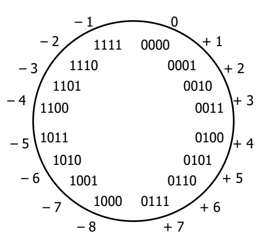
Add		Invert, add, add carry		Invert and add	
4	0100	4	0100	– 4	1011
+ 3	+ 0011	– 3	+ 1100	+ 3	+ 0011
= 7	= 0111	= 1	1 0000	– 1	1110
		add carry:	+1		
			= 0001		

- Remaining problem: Two representations for zero
  - 0 = 0000 and also –0 = 1111

76

Twos-complement

- Negative number: Bitwise complement plus one
  - 0011 ≡ 3_{10}
  - 1101 ≡ $–3_{10}$
- Number wheel



- Only one zero!
- MSB is the sign digit
  - 0 ≡ positive
  - 1 ≡ negative

77

Twos-complement (con't)

- Complementing a complement ➜ the original number
- Arithmetic is easy
  - Subtraction = negation and addition
    - Easy to implement in hardware

Add		Invert and add		Invert and add	
4	0100	4	0100	– 4	1100
+ 3	+ 0011	– 3	+ 1101	+ 3	+ 0011
= 7	= 0111	= 1	1 0001	– 1	1111
		drop carry	= 0001		

78

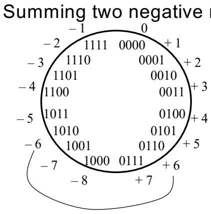
Miscellaneous

- Twos-complement of non-integers
  - $1.6875_{10} = 01.1011_2$
  - $-1.6875_{10} = 10.0101_2$
- Sign extension
  - Write +6 and –6 as twos complement
    - 0110 and 1010
  - Sign extend to 8-bit bytes
    - 00000110 and 11111010
- Can't infer a representation from a number
  - 11001 is 25 (unsigned)
  - 11001 is –9 (sign magnitude)
  - 11001 is –6 (ones complement)
  - 11001 is –7 (twos complement)

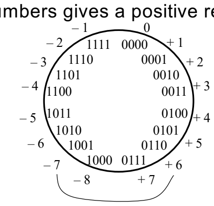
79

Twos-complement overflow

- Summing two positive numbers gives a negative result
- Summing two negative numbers gives a positive result



$6 + 4 \Rightarrow -6$



$-7 - 3 \Rightarrow +6$

- Make sure to have enough bits to handle overflow

80

Gray and BCD codes

Decimal Symbols	Gray Code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100
8	1100
9	1101

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

81

Summary

- Common number systems
  - Common number systems include: Decimal, Binary, Octal, Hexadecimal
- Base conversion
  - For convenience, people use other bases (like decimal, hexdecimal) and we need to know how to convert from one to another.
- Arithmatics
  - How to add, subtract, multiply, etc.
- Negative numbers
  - There are more than one way to express a number in binary. So 1010 could be -2, -5 or -6 and need to know which one.

82