BLM5207
Computer Organization

Prof. Dr. Nizamettin AYDIN
naydin@yildiz.edu.tr
http://www3.yildiz.edu.tr/~naydin

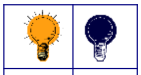
Number Systems

1

1

Why Binary?

- Early computer design was decimal
  - Mark I and ENIAC
- John von Neumann proposed binary data processing (1945)
  - Simplified computer design
  - Used for both instructions and data
- Natural relationship between on/off switches and calculation using Boolean logic



On	Off
True	False
Yes	No
1	0

2

2

Counting and Arithmetic

- Decimal or base 10 number system
  - Origin: counting on the fingers
  - Digit from the Latin word digitus meaning finger
- Base: the number of different digits including zero in the number system
  - Example: Base 10 has 10 digits, 0 through 9
- Binary or base 2
- Bit (binary digit): 2 digits, 0 and 1
- Octal or base 8: 8 digits, 0 through 7
- Hexadecimal or base 16: 16 digits, 0 through F
  - Examples: $10_{10} = A_{16}$; $11_{10} = B_{16}$

3

3

Keeping Track of the Bits

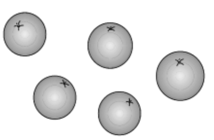
- Bits commonly stored and manipulated in groups
  - 8 bits = 1 byte
  - 4 bytes = 1 word (in many systems)
- Number of bits used in calculations
  - Affects accuracy of results
  - Limits size of numbers manipulated by the computer

4

4

Numbers: Physical Representation

- Different numerals, same number of oranges
  - Cave dweller: IIIII
  - Roman: V
  - Arabic: 5
- Different bases, same number of oranges
  - 5_{10}
  - 101_2
  - 12_3



5

5

Number System

- Roman: position independent
- Modern: based on positional notation (place value)
  - Decimal system: system of positional notation based on powers of 10.
  - Binary system: system of positional notation based powers of 2
  - Octal system: system of positional notation based on powers of 8
  - Hexadecimal system: system of positional notation based powers of 16

6

6

Positional Notation: Base 10

- $43 = 4 \times 10^1 + 3 \times 10^0$

10's place   1's place

Place	10^1	10^0
Value	10	1
Evaluate	4 x 10	3 x1
Sum	40	3

7

Positional Notation: Base 10

- $527 = 5 \times 10^2 + 2 \times 10^1 + 7 \times 10^0$

100's place   10's place   1's place

Place	10^2	10^1	10^0
Value	100	10	1
Evaluate	5 x 100	2 x 10	7 x1
Sum	500	20	7

8

Positional Notation: Octal

- $624_8 = 404_{10}$

64's place   8's place   1's place

Place	8^2	8^1	8^0
Value	64	8	1
Evaluate	6 x 64	2 x 8	4 x 1
Sum for Base 10	384	16	4

9

Positional Notation: Hexadecimal

- $6704_{16} = 26372_{10}$

4,096's place   256's place   16's place   1's place

Place	16^3	16^2	16^1	16^0
Value	4,096	256	16	1
Evaluate	6 x 4,096	7 x 256	0 x 16	4 x 1
Sum for Base 10	24,576	1,792	0	4

10

Positional Notation: Binary

- $1101\ 0110_2 = 214_{10}$

Place	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Value	128	64	32	16	8	4	2	1
Evaluate	1 x 128	1 x 64	0 x 32	1 x16	0 x 8	1 x 4	1 x 2	0 x 1
Sum for Base 10	128	64	0	16	0	4	2	0

11

Estimating Magnitude: Binary

- $1101\ 0110_2 = 214_{10}$
  - $1101\ 0110_2 > 192_{10}$ (128 + 64 + additional bits to the right)

Place	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Value	128	64	32	16	8	4	2	1
Evaluate	1 x 128	1 x 64	0 x 32	1 x16	0 x 8	1 x 4	1 x 2	0 x 1
Sum for Base 10	128	64	0	16	0	4	2	0

12

Range of Possible Numbers

- $R = B^K$ where
  - R = range
  - B = base
  - K = number of digits
- Example #1: Base 10, 2 digits
  - $R = 10^2$ = 100 different numbers (0…99)
- Example #2: Base 2, 16 digits
  - $R = 2^{16}$ = 65536 or 64K
  - 16-bit PC can store 65536 different number values

13

Decimal Range for Bit Widths

Bits	Digits	Range
1	0+	2 (0 and 1)
4	1+	16 (0 to 15)
8	2+	256
10	3	1,024 (1K)
16	4+	65,536 (64K)
20	6	1,048,576 (1M)
32	9+	4,294,967,296 (4G)
64	19+	Approx. 1.6×10^{19}
128	38+	Approx. 2.6×10^{38}

14

Base or Radix

- Base:
  - The number of different symbols required to represent any given number
- The larger the base, the more numerals are required
  - Base 10: 0,1, 2,3,4,5,6,7,8,9
  - Base 2: 0,1
  - Base 8: 0,1,2, 3,4,5,6,7
  - Base 16: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F

15

Number of Symbols vs. Number of Digits

- For a given number, the larger the base
  - the more symbols required
  - but the fewer digits needed
- Example #1:
  - 65_{16} 101_{10} 145_8 $110\ 0101_2$
- Example #2:
  - $11C_{16}$ 28410434_8 $1\ 0001\ 1100_2$

16

Counting in Base 2

Binary Number	Equivalent 8's (2^3)	4's (2^2)	2's (2^1)	1's (2^0)	Decimal Number
0				0×2^0	0
1				1×2^0	1
10			1×2^1	0×2^0	2
11			1×2^1	1×2^0	3
100		1×2^2			4
101		1×2^2		1×2^0	5
110		1×2^2	1×2^1		6
111		1×2^2	1×2^1	1×2^0	7
1000	1×2^3				8
1001	1×2^3			1×2^0	9
1010	1×2^3		1×2^1		10

17

Base 10 Addition Table

- $3_{10} + 6_{10} = 9_{10}$

+	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13

etc

18

Base 8 Addition Table

- $3_8 + 6_8 = 11_8$

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

– Because the base is 8, there is no 8 or 9.

19

19

Base 10 Multiplication Table

- $3_{10} \times 6_{10} = 18_{10}$

x	0	1	2	3	4	5	6	7	8	9
0				← 0 →						
1		1	2	3	4	5	6	7	8	9
2		2	4	6	8	10	12	14	16	18
3		3	6	9	12	15	18	21	24	27
4	0	4	8	12	16	20	24	28	32	36
5		5	10	15	20	25	30	35	40	45
6		6	12	18	24	30	36	42	48	54
7		7	14	21	28	35	42	49	56	63

etc.

20

20

Base 8 Multiplication Table

- $3_8 \times 6_8 = 22_8$

x	0	1	2	3	4	5	6	7
0				← 0 →				
1		1	2	3	4	5	6	7
2		2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4		4	10	14	20	24	30	34
5		5	12	17	24	31	36	43
6		6	14	22	30	36	44	52
7		7	16	25	34	43	52	61

21

21

Addition

Base	Problem	Largest Single Digit
Decimal	6 +3	9
Octal	6 +1	7
Hexadecimal	6 +9	F
Binary	1 +0	1

22

22

Addition

Base	Problem	Carry	Answer
Decimal	6 +4	Carry the 10	10
Octal	6 +2	Carry the 8	10
Hexadecimal	6 +A	Carry the 16	10
Binary	1 +1	Carry the 2	10

23

23

Binary Arithmetic

1	1	1	1	1			
	1	1	0	1	1	0	1
+			1	0	1	1	0
1	0	0	0	0	0	1	1

- Addition
  – Boolean using XOR and AND
- Multiplication
  – AND
  – Shift
- Division

+	0	1
0	0	1
1	1	10

x	0	1
0	0	0
1	0	1

24

24

Binary Arithmetic: Boolean Logic

- Boolean logic without performing arithmetic
  - EXCLUSIVE-OR
    - Output is “1” only if either input, but not both inputs, is a “1”
  - AND (carry bit)
    - Output is “1” if and only both inputs are a “1”

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

25

25

Binary Multiplication

- Boolean logic without performing arithmetic
  - AND (carry bit)
    - Output is 1 if and only both inputs are a 1
  - Shift
    - Shifting a number in any base left one digit multiplies its value by the base
  - Shifting a number in any base right one digit divides its value by the base
  - Examples:
    - 10_{10} shift left $= 100_{10}$
    - 10_{10} shift right $= 1_{10}$
    - 10_2 shift left $= 100_2$
    - 10_2 shift right $= 1_2$

26

26

Binary Multiplication

```
      1 1 0 1
        1 0 1
      -------
      1 1 0 1    1's place
    0 0 0 0      2's place
  1 1 0 1        4's place (bits shifted to line up with 4's place of multiplier)
-------------
1 0 0 0 0 0 1    Result (AND)
```

27

27

Binary Multiplication

```
              1 1 0 1 1 0 1
          x   1 0 0 1 1 0
          -----------------
            1 1 0 1 1 0 1      2's place (bits shifted to line up with 2's place of multiplier)
          1 1 0 1 1 0 1        4's place
    1 1 0 1 1 0 1              32's place
---------------------------
1 0 0 0 0 0 0 1 0 1 1 1 0      Result (AND)
                               Note the 0 at the end, since the 1's place is not brought down.
```

Note: multiple carries are possible.

28

28

Converting from Base 10

- Powers Table

Power / Base	8	7	6	5	4	3	2	1	0
2	256	128	64	32	16	8	4	2	1
8				32,768	4,096	512	64	8	1
16					65,536	4,096	256	16	1

29

29

From Base 10 to Base 2

- $42_{10} = 101010_2$

Power / Base	6	5	4	3	2	1	0
2	64	32	16	8	4	2	1
		1	0	1	0	1	0
Integer		42/32 = 1	10/16 = 0	10/8 = 1	2/4 = 0	2/2 = 1	0/1 = 0
Remainder		10	10	2	2	0	0

30

30

From Base 10 to Base 2

Base 10 42

Remainder

Quotient

2) 42 (**0** Least significant bit
2) 21 (**1**
2) 10 (**0**
2) 5 (**1**
2) 2 (**0**
2) **1** Most significant bit

Base 2 **1 0 1 0 1 0**

31

31

From Base 10 to Base 16

- $5735_{10} = 1667_{16}$

Power / Base	4	3	2	1	0
16	65,536	4,096	256	16	1
		1	**6**	**6**	**7**
Integer		5,735 /4,096 = 1	1,639 / 256 = 6	103 /16 = 6	7
Remainder		5,735 - 4,096 = 1,639	1,639 –1,536 = 103	103 – 96 = 7	

32

32

From Base 10 to Base 16

Base 10 5,735

Remainder

Quotient

16) 5,735 (7 Least significant bit
16) 358 (6
16) 22 (6
16) 1 (1 Most significant bit
16) 0

Base 16 1667

33

33

From Base 10 to Base 16

Base 10 8,039

Remainder

Quotient

16) 8,039 (7 Least significant bit
16) 502 (6
16) 31 (15
16) 1 (1 Most significant bit
16) 0

Base 16 1F67

34

34

From Base 8 to Base 10

- $7263_8 = 3763_{10}$

$7263_8 = 3,763_{10}$

Power	8^3	8^2	8^1	8^0
	512	64	8	1
	x 7	x 2	x 6	x 3
Sum for Base 10	3,584	128	48	3

35

35

From Base 8 to Base 10

- $7263_8 = 3763_{10}$

7
x 8
56 + **2** = 58
x 8
464 + **6** = 470
x 8
3760 + **3** = 3,763

36

36

From Base 16 to Base 2

- The nibble approach
  - Hex easier to read and write than binary

Base 16	1	F	6	7
Base 2	0001	1111	0110	0111

- Why hexadecimal?
  - Modern computer operating systems and networks present variety of troubleshooting data in hex format

37

37

Fractions

- Number point or radix point
  - Decimal point in base 10
  - Binary point in base 2
- No exact relationship between fractional numbers in different number bases
  - Exact conversion may be impossible

38

38

Decimal Fractions

- Move the number point one place to the right
  - Effect: multiplies the number by the base number
  - Example: 139.0_{10} → 1390_{10}
- Move the number point one place to the left
  - Effect: divides the number by the base number
  - Example: 139.0_{10} → 13.9_{10}

39

39

Fractions: Base 10 and Base 2

- 0.2589_{10}

Place	10^{-1}	10^{-2}	10^{-3}	10^{-4}
Value	1/10	1/100	1/1000	1/10000
Evaluate	2 x 1/10	5 x 1/100	8 x 1/1000	9 x1/1000
Sum	0.2	0.05	0.008	0.0009

- $0.101011_2 = 0.671875_{10}$

Place	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
Value	1/2	1/4	1/8	1/16	1/32	1/64
Evaluate	1 x 1/2	0 x 1/4	1x 1/8	0 x 1/16	1 x 1/32	1 x 1/64
Sum	0.5		0.125		0.03125	0.015625

40

40

Fractions: Base 10 and Base 2

- No general relationship between fractions of types $1/10^k$ and $1/2^k$
  - Therefore, a number representable in base 10 may not be representable in base 2
  - But: the converse is true: all fractions of the form $1/2^k$ can be represented in base 10
- Fractional conversions from one base to another are stopped
  - If there is a rational solution or
  - When the desired accuracy is attained

41

41

Mixed Number Conversion

- Integer and fraction parts must be converted separately
- Radix point: fixed reference for the conversion
  - Digit to the left is a unit digit in every base
  - B^0 is always 1 regardless of the base

42

42