SER 232

Computer Systems Fundamentals I

Dr. Heinrichs

Topics

- Numbers in General
- Analog vs. Digital
- Sampling
- Number Representations

Numbers

- Numbers in computer area in general?
  - Pixel: Brightness, RGB colors, …
  - Sound: Amplitude, frequency, time, …
  - Games: Character health, damage, speed, …
  - 3D geometry (models, levels): X/Y/Z coordinates, …
  - …
- Numbers in programming?
  - Int, bool, char, string…

Numbers

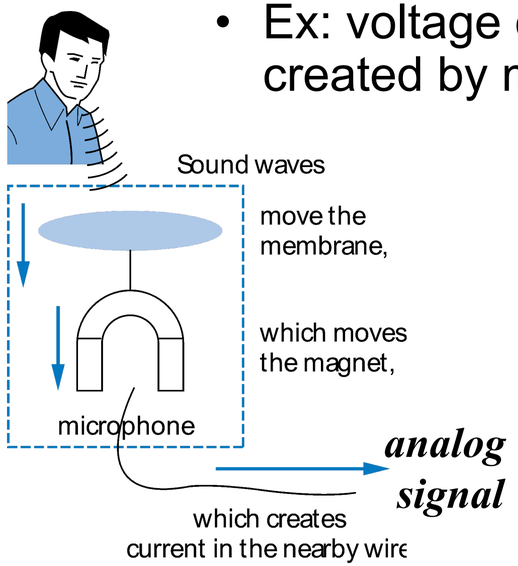
- Even characters/strings are numbers
- ASCII
  - American Standard Code for Information Interchange

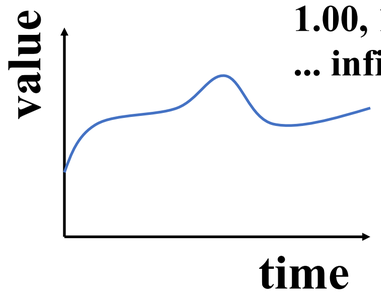
Binary	Oct	Dec	Hex	Glyph		
				'63	'65	'67
010 0000	040	32	20	space		
010 0001	041	33	21	!		
011 0000	060	48	30	0		
011 0001	061	49	31	1		
011 0010	062	50	32	2		
100 0100	104	68	44	D		
110 0100	144	100	64		d	

Figure: https://en.wikipedia.org/wiki/ASCII

Analog & Digital

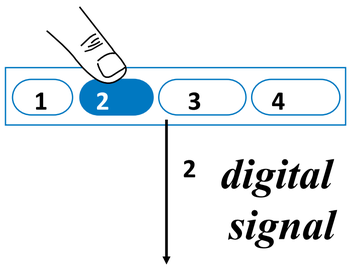
- Analog signal
  - Infinite possible values
    - Ex: voltage on a wire created by microphone

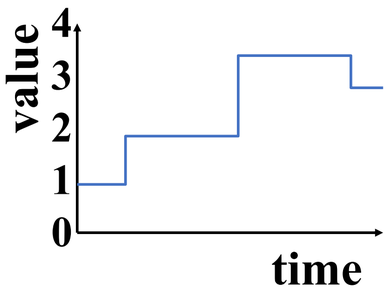




Possible values:
1.00, 1.01, 2.0000009,
... infinite possibilities

- Digital signal
  - Finite possible values
    - Ex: button pressed on a keypad





Possible values:
0, 1, 2, 3, or 4.
That's it.

Source: Frank Vahid, Digital Design

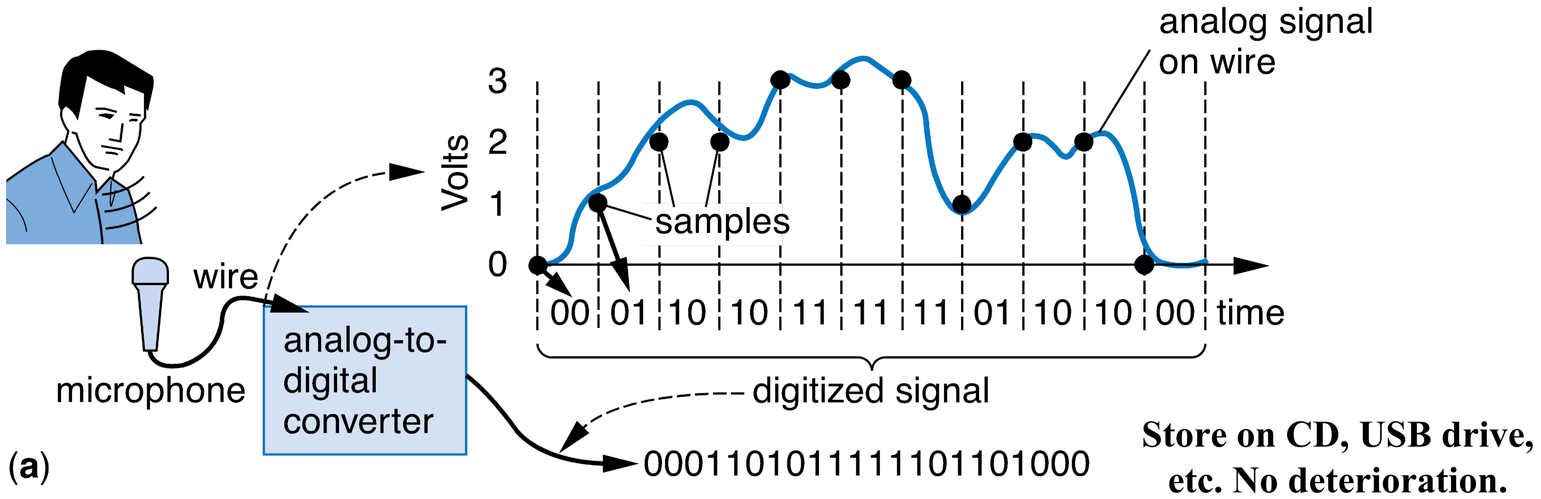
Computers: Exclusively Digital

- Computers are digital systems
- Everything is stored and handled as a binary number

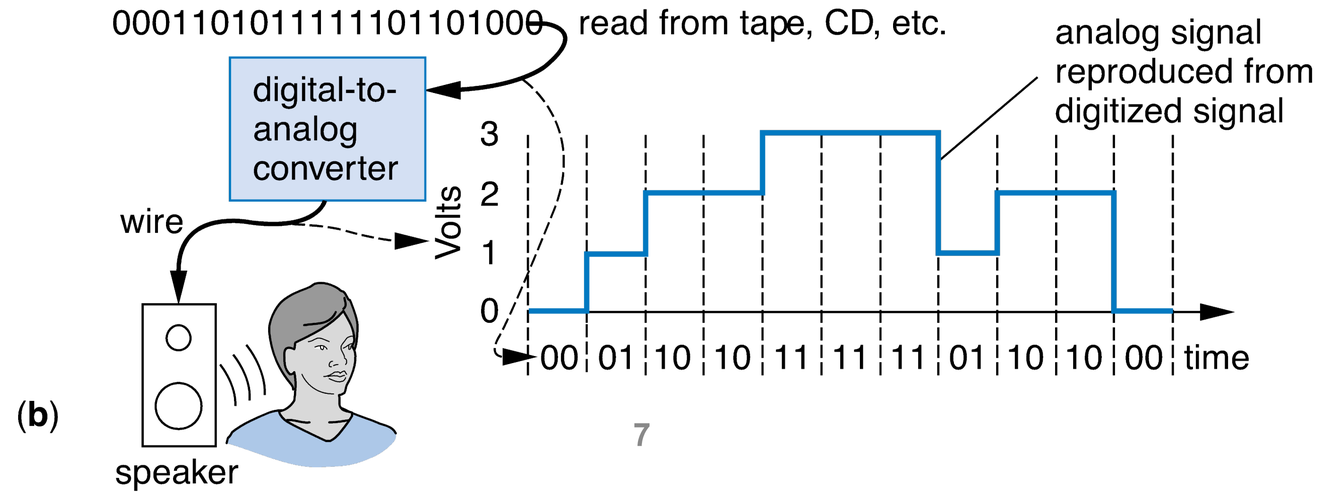
- How do we store analog signals, like sound?

Sampling Analog Signals

- Convert to a digital signal:



Store on CD, USB drive, etc. No deterioration.

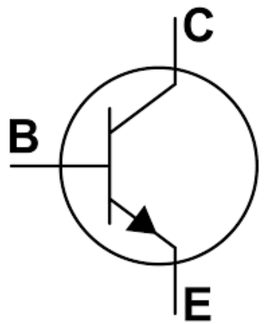


Source: Frank Vahid, Digital Design

Why Are Different Bases Used?

- Comes down to counting
  - Decimal (base 10)
    - We have 10 fingers (or digits)
  - Binary (base 2)
    - Transistors can be on or off
  - Hex(adecimal) (base 16) and octal (base 8)
    - Used to represent groups of 4 or 3 binary digits

- Transistor acts like a light switch (on/off)
- Actuated by current instead manually



Binary

- We only have two numbers in a single **b**inary dig**it** (bit): 1 and 0
- This aligns with the state of a transistor: On (1) or off (0)
- Everything stored and handled in a computer is represented as a binary number
- E.g.: “01001000 01100101 01101100 01101100 01101111” interpreted as a String reads: “Hello”
  - Each group of 8 bits represents one character
  - Knowledge what a binary number represents is necessary

Hexadecimal

- One hexadecimal (or “hex”) digit has 16 different values:
  - 0 through 9 are the same as in decimal
  - 10-15 are letters: A (10), B (11), C (12), D (13), E (14,) F (15)

Number Representations

Decimal

Binary

Hexadecimal

SER 232

Computer Systems Fundamentals I

Dr. Heinrichs

Topics

- Counting in Binary
- Converting: Decimal ←→ Binary

Counting in Binary

Number Representations

Decimal

Binary

Hexadecimal

Binary to Decimal

- First: How does decimal presentation work in detail?
- In decimal each digit position represents a power of 10
  - 7538

    $= 7(10^3) + 5(10^2) + 3(10^1) + 8(10^0)$

    $= 7000 + 500 + 30 + 8$

Binary to Decimal

- Binary numbers work the same, but each digit position represents a power of 2
  - 1101

    $= 1(2^3) + 1(2^2) + 0(2^1) + 1(2^0)$

    $= 8 + 4 + 0 + 1$

    $= 13$

Binary to Decimal

- Binary numbers work the same, but each digit position represents a power of 2
  - 1101

    $= 1(2^3) + 1(2^2) + 0(2^1) + 1(2^0)$

    $= 8 + 4 + 0 + 1$

    $= 13$
  - 1111 1111

$= 1(2^7) + 1(2^6) + 1(2^5) + 1(2^4) + 1(2^3) + 1(2^2) + 1(2^1) + 1(2^0)$

$= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1$

$= 255$

Key Binary Values

- 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, …
- Multiplies by two every time
- Values for different positions of bits:

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

Convert:

a) 11001

b) 10100111

Binary to Decimal

Convert:

a) 11001

b) 10100111

a. 11001

$= 1(2^4) + 1(2^3) + 0(2^2) + 0(2^1) + 1(2^0)$

$= 16 + 8 + 0 + 0 + 1$

$= 25$

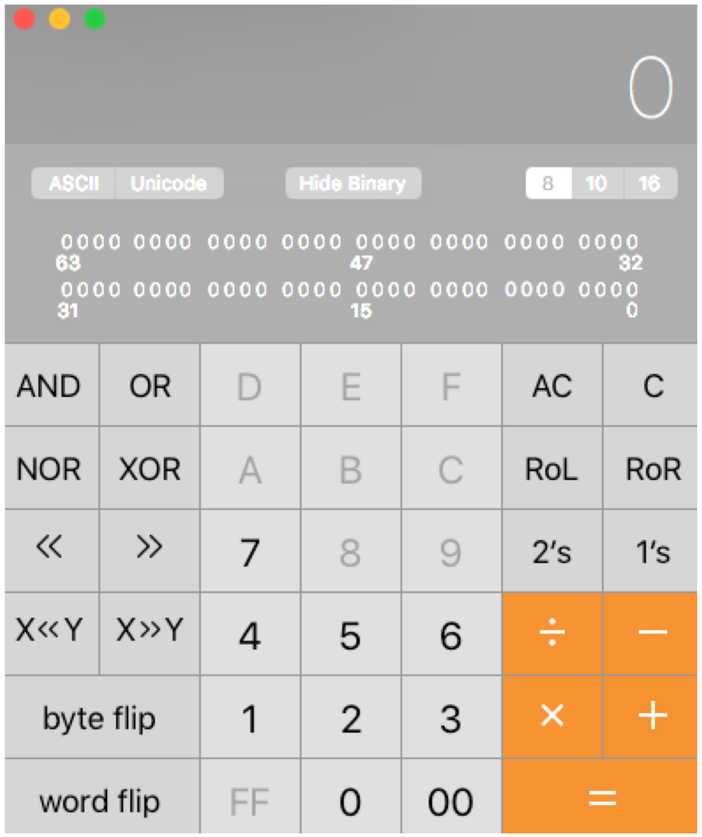
b. 10100111

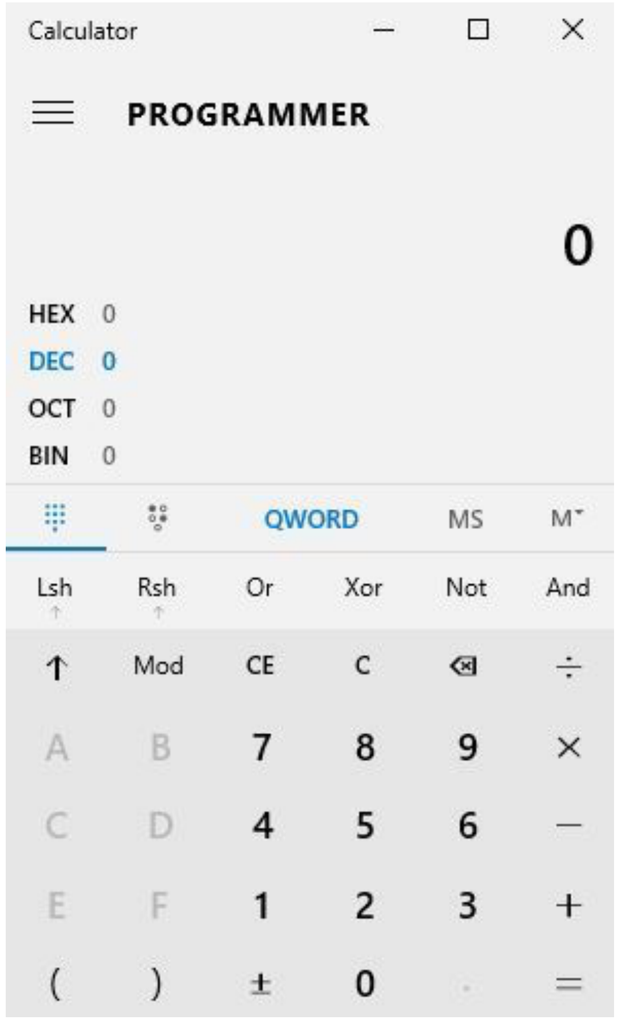
$= 1(2^7) + 0(2^6) + 1(2^5) + 0(2^4) + 0(2^3) + 1(2^2) + 1(2^1) + 1(2^0)$

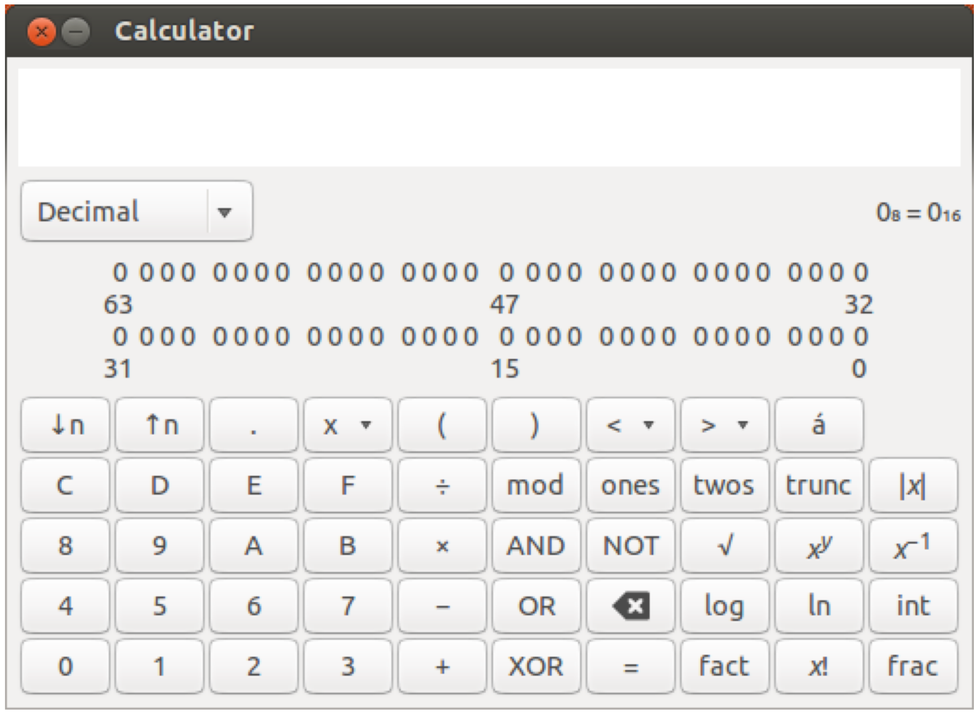
$= 128 + 0 + 32 + 0 + 0 + 4 + 2 + 1$

$= 167$

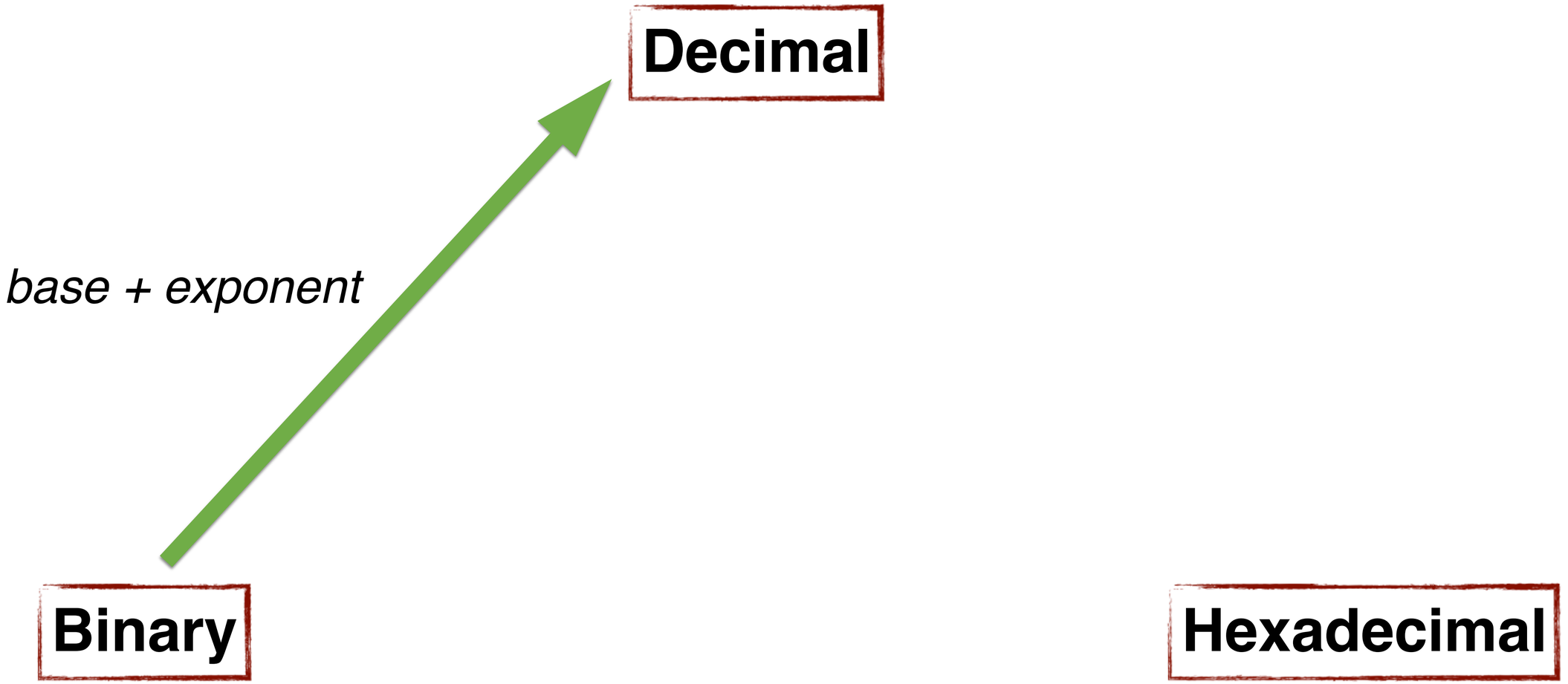
OS Calculator







Number Representations



Decimal to Binary

- To convert a decimal number into binary the “Divide-by-2” method can be used
- “Divide-by-2” method:
  - Step 1) Divide decimal number by 2
  - Step 2) Write down if remainder exists (0: no remainder, 1: remainder)
  - Step 3) Discard remainder
  - Step 4) Repeat until result is smaller than 1
  - Binary number will be the remainder indicators read backwards (last to first)
- Divide-by-two tracks of remainders from repeatedly dividing decimal value by 2

Divide-by-2 Method

Decimal	Remainder
20	

Divide-by-2 Method

Decimal	Remainder
20	0
10	0
5	1
2	0
1	1

$20_{10} = 10100_2$

Divide-by-2 Method

Decimal	Remainder
20	0
10	0
5	1
2	0
1	1

Apply to:

a) 13

b) 32

c) 31

d) 63

$20_{10} = 10100_2$

Divide-by-2 Method

Decimal	Remainder
13	

Apply to:

a) 13

b) 32

c) 31

d) 63

Divide-by-2 Method

Decimal	Remainder
13	1
6	0
3	1
1	1

Apply to:

a) 13

b) 32

c) 31

d) 63

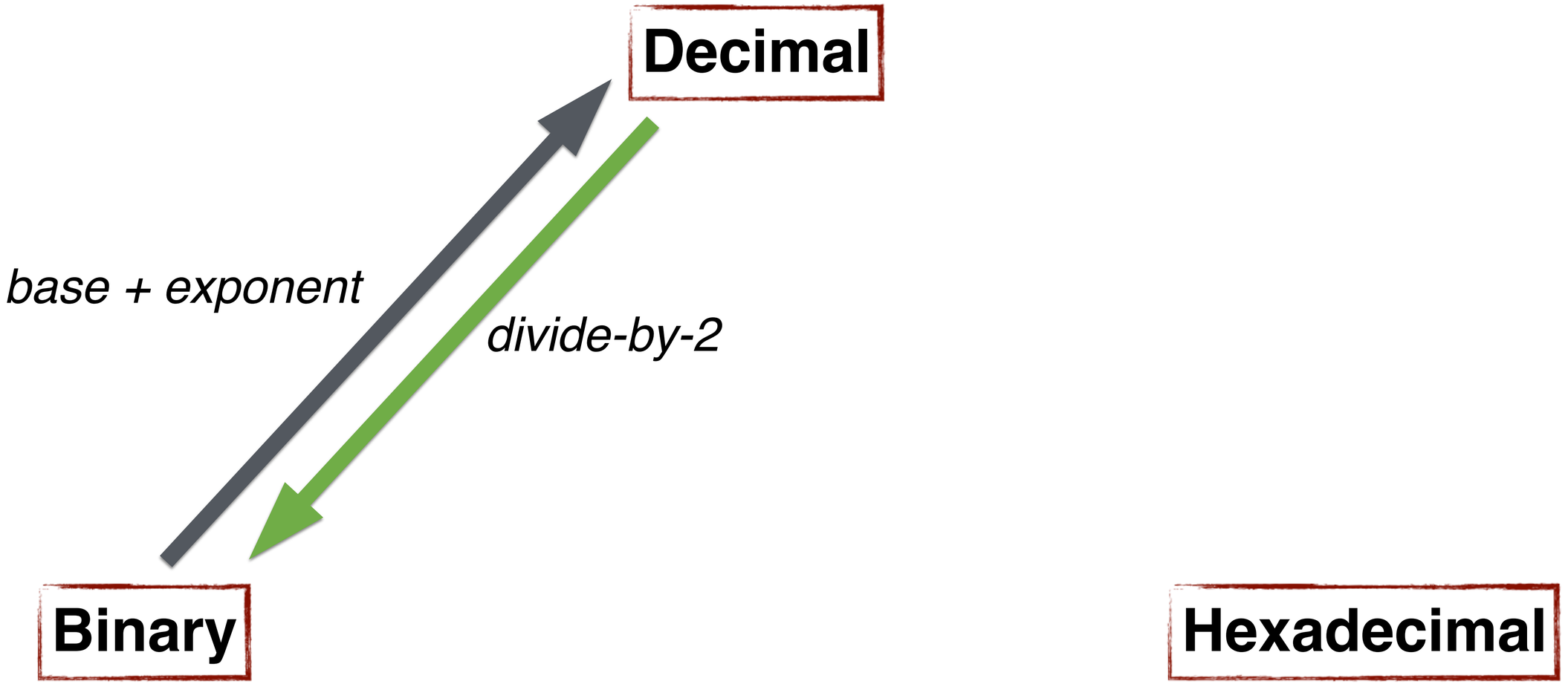
$13_{10} = 1101_2$

$32_{10} = 100000_2$

$31_{10} = 11111_2$

$63_{10} = 111111_2$

Number Representations



SER 232

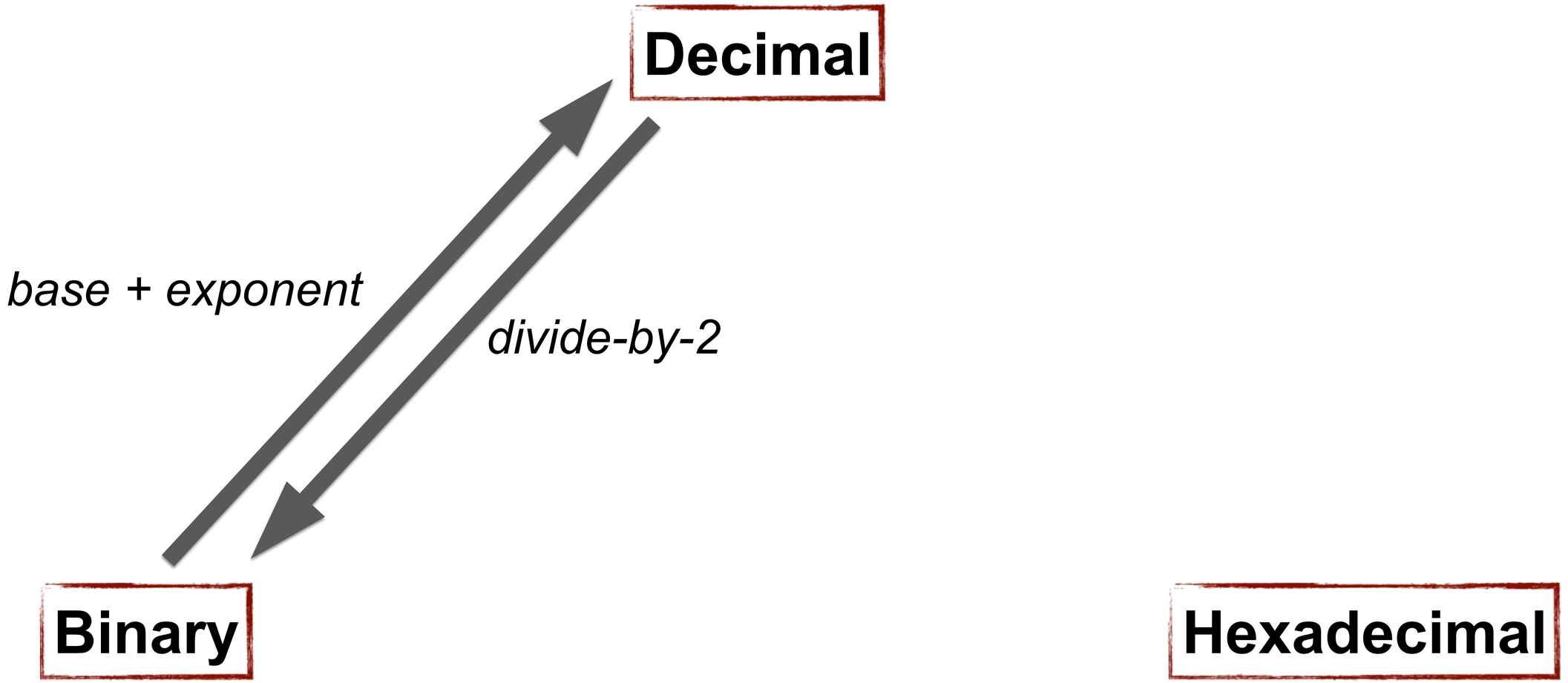
Computer Systems Fundamentals I

Dr. Heinrichs

Topics

- Converting: Decimal ←→ Hex

Number Representations



Hex to Decimal

Apply to:
a) 10
b) 101
c) C
d) 1F

- Each place of hexadecimal numbers represents a power of 16
  - 115_{16}
    $= 1(16^2) + 1(16^1) + 5(16^0)$
    $= 256 + 16 + 5$
    $= 277$
  - $2F_{16}$
    $= 2(16^1) + F(16^0)$
    $= 2(16^1) + 15(16^0)$
    $= 32 + 15$
    $= 47$

Decimal	Hex
10	A
11	B
12	C
13	D
14	E
15	F

Hex to Decimal

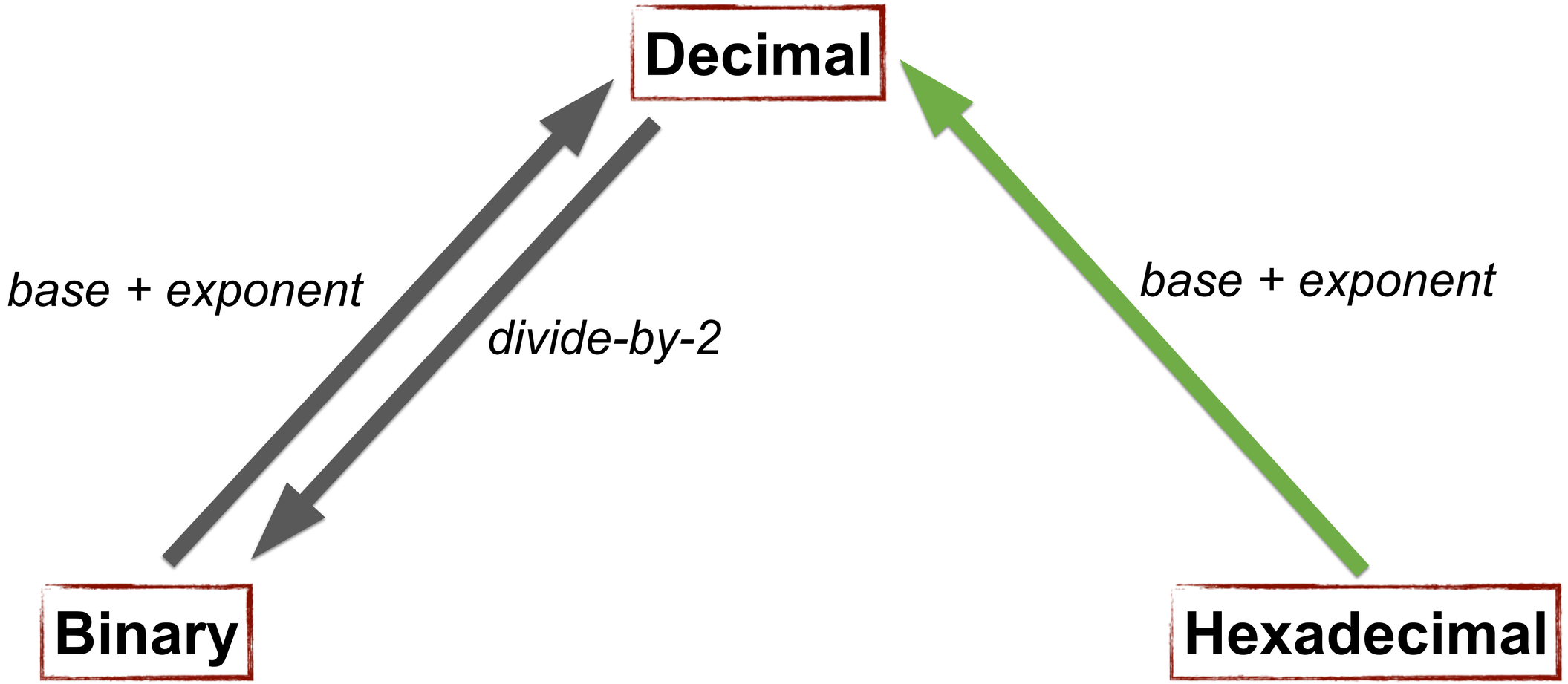
Apply to:
a) 10
b) 101
c) C
d) 1F

- Each place of hexadecimal numbers represents a power of 16
  - a) 10_{16}
    $= 1(16^1) + 0(16^0)$
    $= 16 + 0$
    $= 16$
  - d) $1F_{16}$
    $= 1(16^1) + F(16^0)$
    $= 1(16^1) + 15(16^0)$
    $= 16 + 15$
    $= 31$

Decimal	Hex
10	A
11	B
12	C
13	D
14	E
15	F

b) $101_{16} = 257_{10}$
c) $C_{16} = 12_{10}$

Number Representations



Decimal to Hex

- Similar approach as converting decimal to binary with divide-by-2 method
- “Divide-by-16” method:
  - Step 1): Divide decimal number by 16
  - Step 2): Multiply fractional part by 16 & write down as hex
  - Step 3): Continue with whole part
  - Step 4): Repeat until whole part is 0
  - Hexadecimal value is read from last to first

Decimal to Hex

Decimal	Remainder
516	

Decimal to Hex

Decimal	Remainder
516	4
32	0
2	2

$516_{10} = 204_{16}$

Decimal to Hex

Decimal	Remainder
875	

Decimal to Hex

Decimal	Remainder
875	B
54	6
3	3

$875_{10} = 36B_{16}$

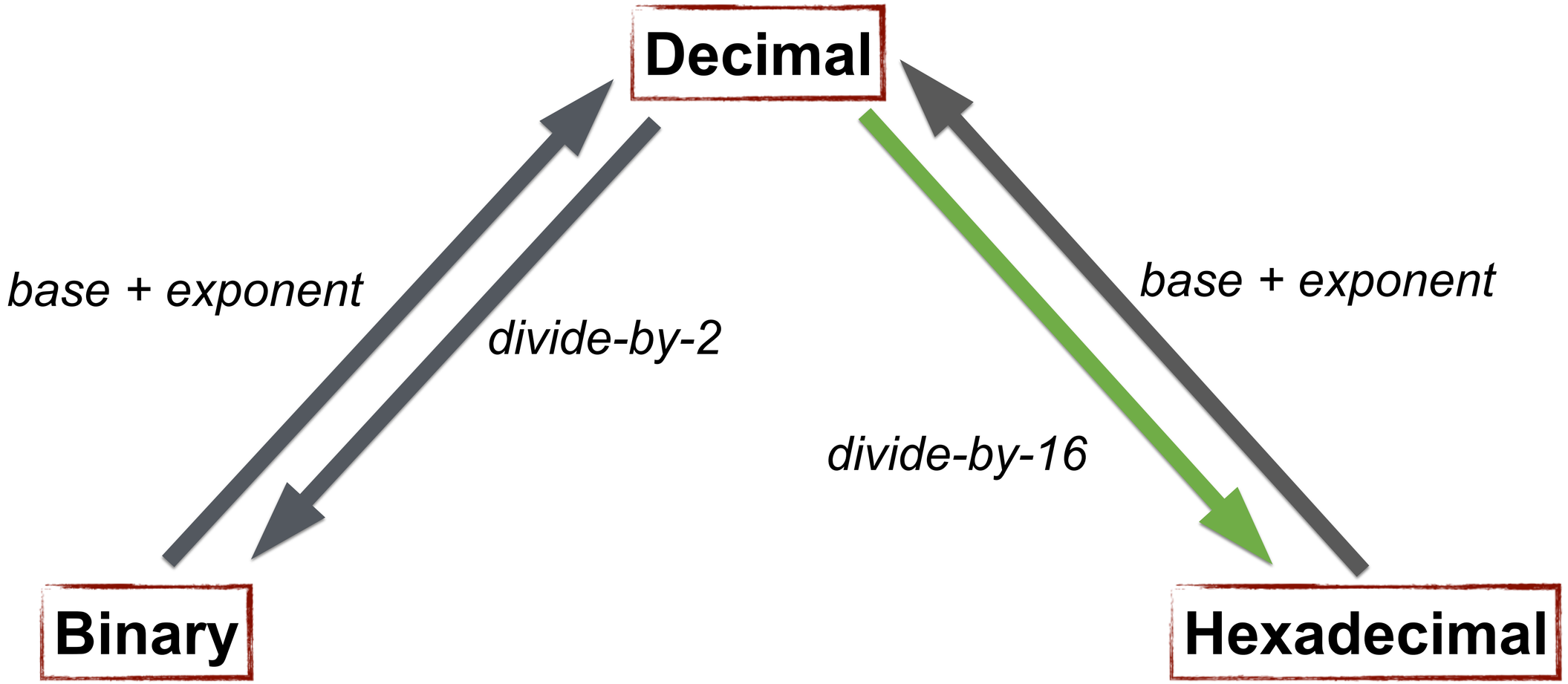
Decimal to Hex

- “Divide-by-16” method:
  - Step 1): Divide decimal number by 16
  - Step 2): Multiply fractional part by 16 & write down as hex
  - Step 3): Continue with whole part
  - Step 4): Repeat until whole part is 0
  - Hexadecimal value is read from last to first

Apply to:
a) 122_{10}
b) 377_{10}

a) $122_{10} = 7A_{16}$
b) $377_{10} = 179_{16}$

Number Representations



SER 232

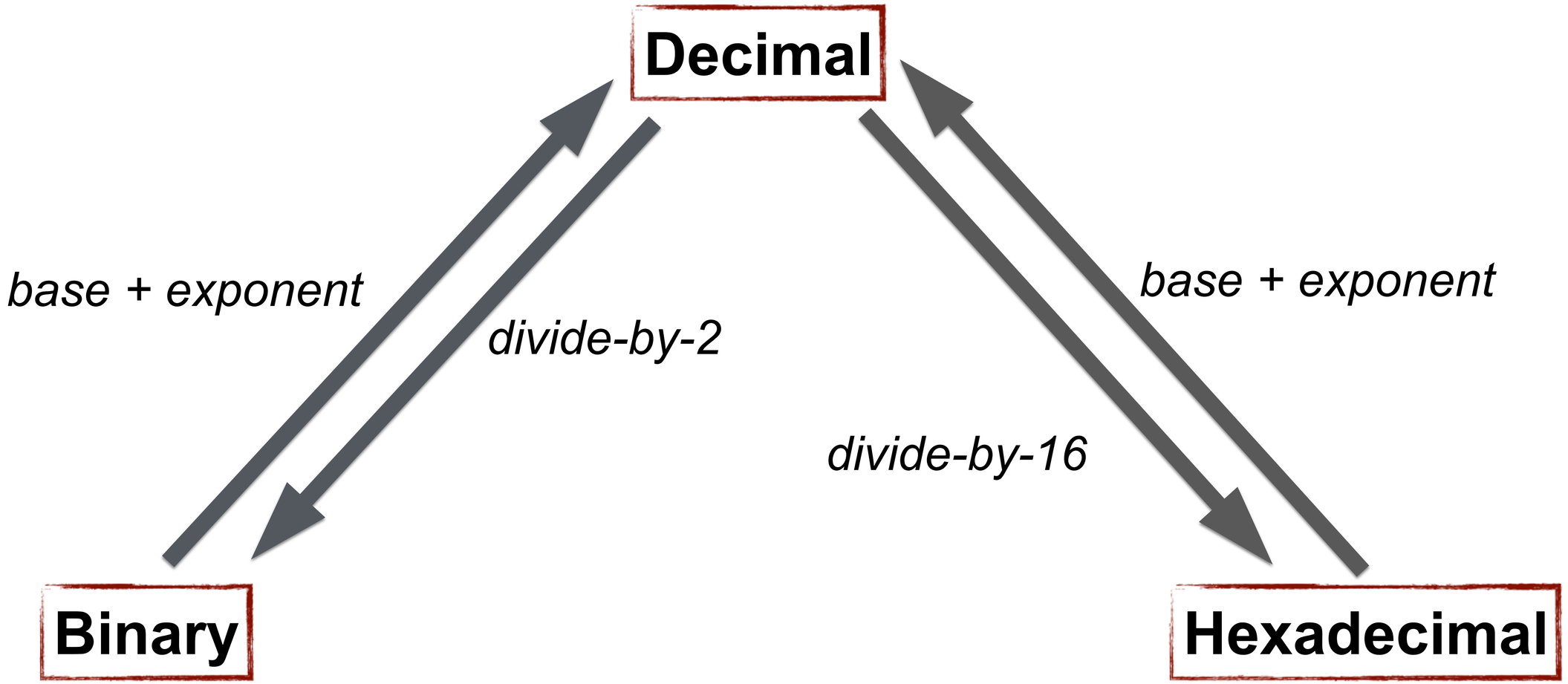
Computer Systems Fundamentals I

Dr. Heinrichs

Topics

- Converting: Hex ←→ Binary

Number Representations



Hex to Binary

- Each hex digit represents 4 bits
- Conversion:
  - Convert each hex digit separately using table
  - String together bits in correct order
- E.g.: $7FA_{16}$
  - $7_{16} = 0111_2$
  - $F_{16} = 1111_2$
  - $A_{16} = 1010_2$
  - $7FA_{16} = 0111\ 1111\ 1010_2$

Apply to:
a) $A1C_{16}$
b) $CAFE_{16}$

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Hex to Binary

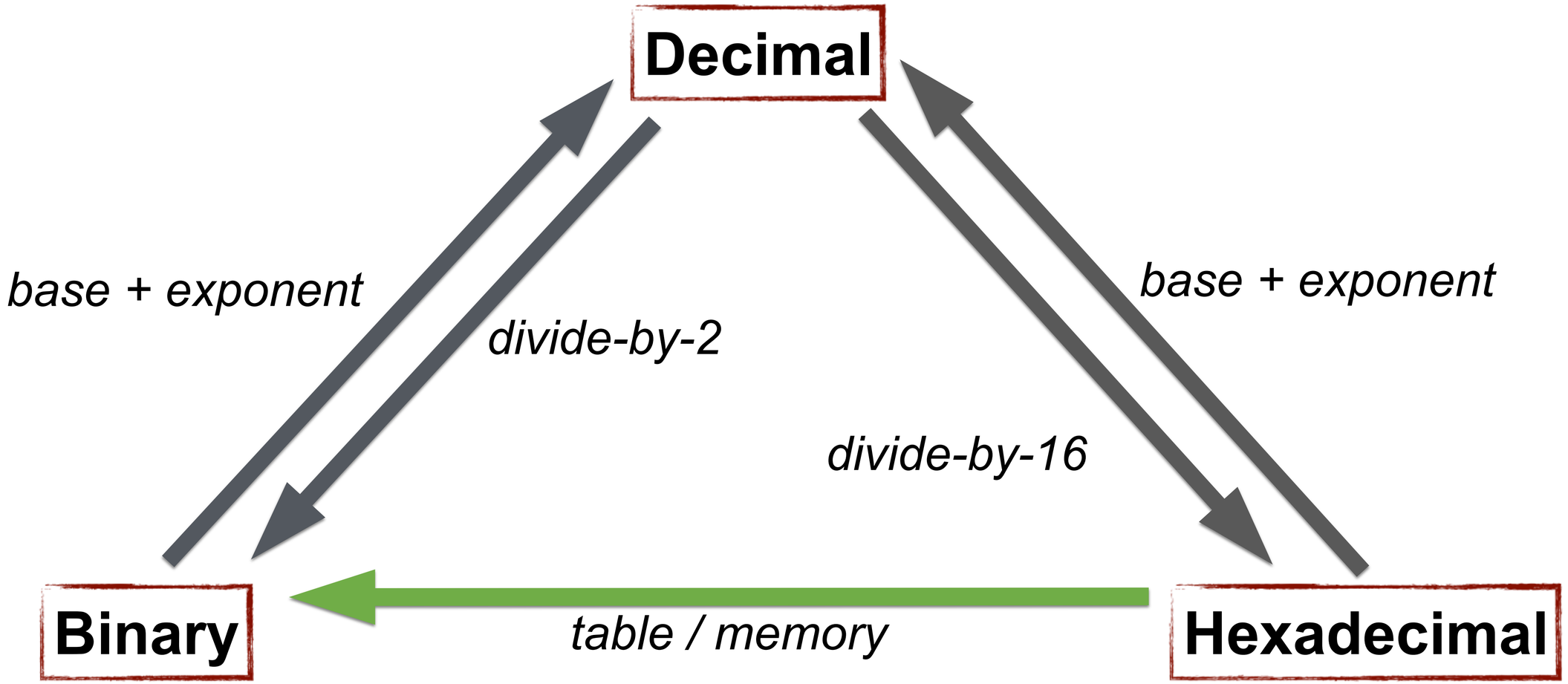
a) $A1C_{16}$
- $A_{16} = 1010_2$
- $1_{16} = 0001_2$
- $C_{16} = 1100_2$
  - $A1C_{16} = 1010\ 0001\ 1100_2$

b) $CAFE_{16}$
- $C_{16} = 1100_2$
- $A_{16} = 1010_2$
- $F_{16} = 1111_2$
- $E_{16} = 1110_2$
  - $CAFE_{16} = 1100\ 1010\ 1111\ 1110_2$

Apply to:
a) $A1C_{16}$
b) $CAFE_{16}$

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Number Representations



Binary to Hex

- Each group of 4 bits converts into one hex digit
- Conversion:
  - Start with rightmost group of 4 bits and convert to hex using table
  - Fill up last group with leading zeros if less than 4 bits
- E.g.: 100011_2

Apply to:

a) 101

b) 1100 0000

c) 1100 1010 1111 1110

d) 1011 1110 1110 1111

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Binary to Hex

- a) $0101_2 = 5_{16}$
- b) $1100\ 0000_2 = C0_{16}$
- c) $1100\ 1010\ 1111\ 1110_2 = CAFE_{16}$
- d) $1011\ 1110\ 1110\ 1111_2 = BEEF_{16}$

Apply to:

a) 101

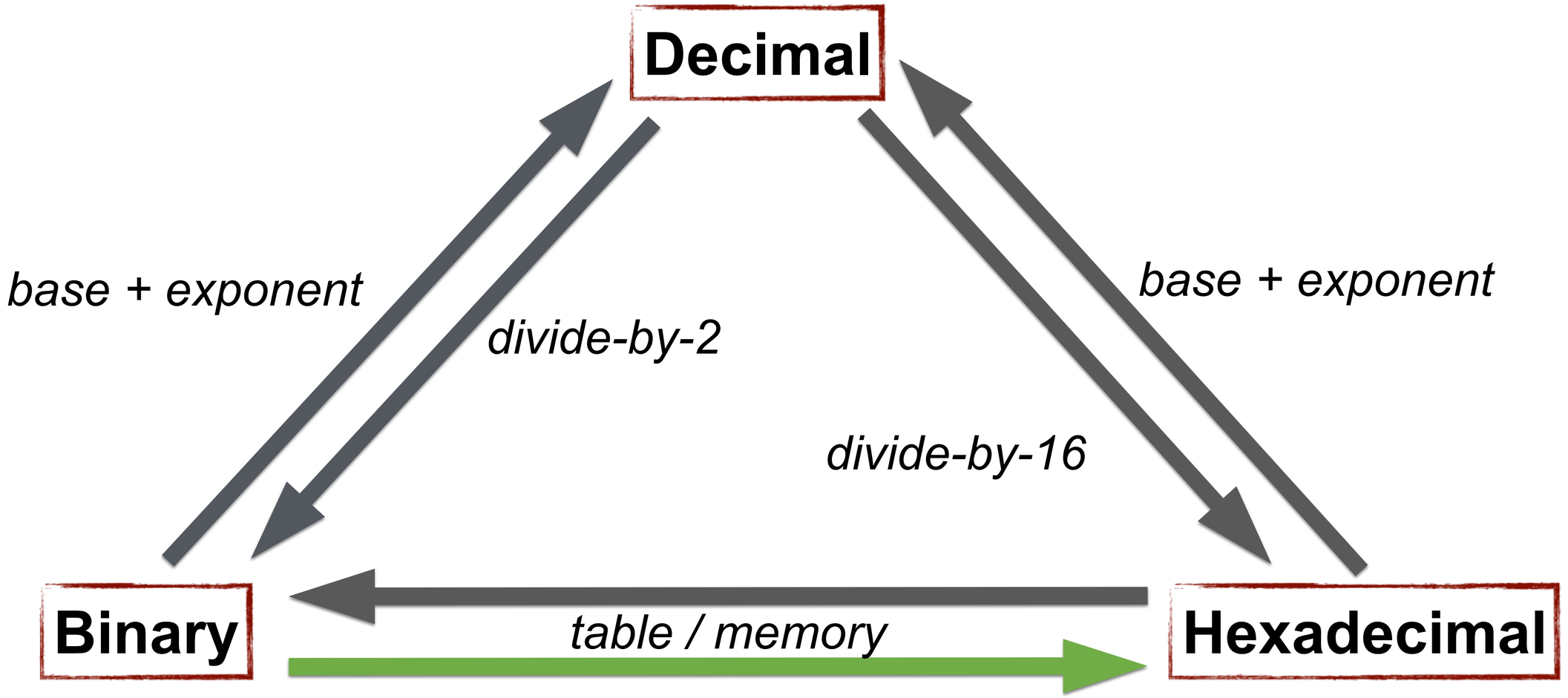
b) 1100 0000

c) 1100 1010 1111 1110

d) 1011 1110 1110 1111

Decimal	Binary	Hex
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Number Representations



SER 232

Computer Systems Fundamentals I

Dr. Heinrichs

Topics

- Number Notation
- Bits & Bytes
- Fractions in Binary
- Floating Point

Mathematical Notation

- Subscript with base
  - 15_{16} for 15 in base 16 / hex
  - 101_2 for 101 in base 2 / binary
  - 63 without base usually implies base 10 / decimal

Programming Notation

- Text editors/IDE's usually do not have rich text (enabling e.g. subscript)
- Different notation used to indicate used number representation / base:
  - Decimal:
    - Number written normally
  - Binary: Prefix '`0b`'

    1101_2 → 0b1101
  - Hex: Prefix '`0x`'

    $FE2_{16}$ → 0xFE2

Bit vs. Byte

- Byte is similar to a dozen
  - 8 bits == 1 byte
  - 4 bits == 1 nibble (half-byte)
- Ever wondered how to check if your download speed is using the whole bandwidth of your internet connection?
  - Download speed usually in MByte per second
  - Connection speed usually in MBit per second
    - Conversion: MBit / 8 = MByte
    - E.g.: 100 MBit / s = 12.5 MByte / s

Bits and Range

- Range of numbers represented by *n* bits:
  - 2^n numbers ranging from 0 to 2^n-1
  - Example:
    - With 3 bits we can represent $2^3 = 2 * 2 * 2 = 8$ numbers
    - Values: 0 to 7

Fractions in Binary

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

2^2	4
2^1	2
2^0	1
2^{-1}	0.5
2^{-2}	0.25
2^{-3}	0.125
2^{-4}	0.0625
2^{-5}	0.03125

Decimal to Binary

- Separate number into whole and fractional part
  1. Use divide-by-2 method on whole part
  2. Use equivalent method on fractional part, but with multiplication

“Divide-by-2” Method (Fraction)

- Step 1) Multiply fractional part by 2
- Step 2) The whole part of each product is appended to the current fractional part of the binary result (after the binary point)
- Step 3) Discard whole part of result
- Step 4) Continue until fractional part is 0

“Divide-by-2” Method (Fraction)

Decimal	Binary Fraction
.375	

“Divide-by-2” Method (Fraction)

Decimal	Binary Fraction
.375	0
.75	1
.5	1
.0	Done

$0.375_{10} = 0.011_2$

whole part

fractional part

Convert:

a) 8.125

b) 19.375

Result:

a) 0b1000.001

b) 0b10011.011

Limitations of Binary Fractions

- Some decimal fractions require an infinite number of bits to represent them
  - With a finite number of bits we can only get close to these fractions
  - Example: 0.1_{10}
    - In binary: 0.000110011001100…

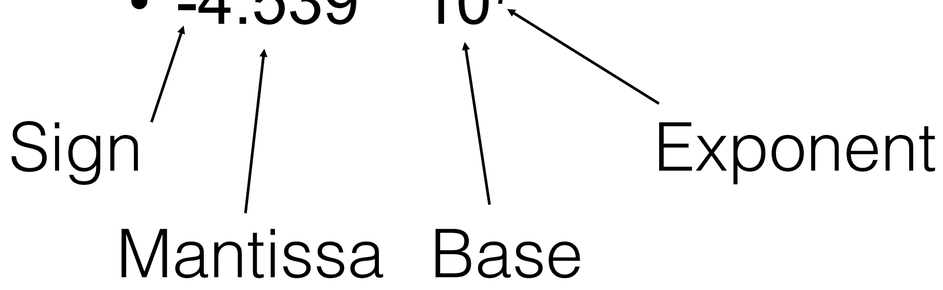
Limitations of Binary Fractions

- Table shows how close we can get to 0.1 with certain amount of bits
  - First three bits do not contribute to fraction because too big
  - Bit 6 and 7 would bring the value above 0.1, thus not used and no change

Number of Bits	Binary Value
1	0
2	0
3	0
4	0.0625
5	0.09375
6	0.09375
7	0.09375
8	0.09765625
9	0.099609375

Floating Point

- IEEE 754
  - Scientific notation with specific number of bits for each part
    - $-4.539 * 10^{7}$



Sign [31]	Exponent [30:23]	Mantissa [22:0]

Floating Point

- IEEE 754 defines both single precision (32 bit) and double precision (64 bit) floating point
- A set number of bits to represent fractional part instead of exponent is referred to as **fixed point**

Limitations of Floating Point

- Very large numbers (large exponent) have limited accuracy
  - Result of $1.0*2^{30} + 1$ will be represented in floating point as $1.0*2^{30}$
  - For addition and subtraction, results become more meaningful the closer exponents