Four Important Number Systems

System	Why?	Remarks
Decimal	Base 10: (10 fingers)	Most used system
Binary	Base 2: On/Off systems	3-4 times more digits than decimal
Octal	Base 8: Shorthand notation for working with binary	3 times less digits than binary
Hex	Base 16	4 times less digits than binary

Positional Number Systems

- Have a radix *r* (base) associated with them.
- In the decimal system, $r = 10$:
  - Ten symbols: 0, 1, 2, ..., 8, and 9
  - More than 9 move to next position, so each position is power of 10
  - Nothing special about base 10 (used because we have 10 fingers)
- What does 642.391_{10} mean?

$$6 \text{ x } 10^{2} + 4 \text{ x } 10^{1} + 2 \text{ x } 10^{0} \quad . \quad 3 \text{ x } 10^{-1} + 9 \text{ x } 10^{-2} + 1 \text{ x } 10^{-3}$$

← Increasingly +ve powers of radix

↑ Radix point

→ Increasingly -ve powers of radix

Positional Number Systems(2)

- What does 642.391_{10} mean?

Radix point

Base 10 (r)	10^2 (100)	10^1 (10)	10^0 (1)	10^{-1} (0.1)	10^{-2} (0.01)	10^{-3} (0.001)
Coefficient (a_j)	6	4	2	3	9	1
Product: a_j*r^i	600	40	2	0.3	0.09	0.001
Value	= 600 + 40 + 2 + 0.3 + 0.09 + 0.001 = 642.391					

- Multiply each digit by appropriate power of 10 and add them together
- In general: $$\sum_{i=n-1}^{-m} a_i \times r^i$$

Positional Number Systems(3)

Number system	Radix	Symbols
Binary	2	{0,1}
Octal	8	{0,1,2,3,4,5,6,7}
Decimal	10	{0,1,2,3,4,5,6,7,8,9}
Hexadecimal	16	{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f}

Binary Number System

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

Octal Number System

Decimal	Octal	Decimal	Octal
0	**0**	**8**	**10**
1	**1**	**9**	**11**
2	**2**	**10**	**12**
3	**3**	**11**	**13**
4	**4**	**12**	**14**
5	**5**	**13**	**15**
6	**6**	**14**	**16**
7	**7**	**15**	**17**

Hexadecimal Number System

Decimal	Hex	Decimal	Hex
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F

Four Number Systems

Decimal	Binary	Octal	Hex	Decimal	Binary	Octal	Hex
0	0000	0	0	8	1000	10	8
1	0001	1	1	9	1001	11	9
2	0010	2	2	10	1010	12	A
3	0011	3	3	11	1011	13	B
4	0100	4	4	12	1100	14	C
5	0101	5	5	13	1101	15	D
6	0110	6	6	14	1110	16	E
7	0111	7	7	15	1111	17	F

Conversion: Binary to Decimal

Binary → Decimal

1101.011_2 → $(??)_{10}$

r	$2^3(8)$	$2^2(4)$	$2^1(2)$	$2^0(1)$	$2^{-1}(0.5)$	$2^{-2}(0.25)$	$2^{-3}(0.125)$
a_j	1	1	0	1	0	1	1
$a_j{*}r$	8	4	0	1	0	0.25	0.125
	$(1101.011)_2 = 8 + 4 + 1 + 0.25 + 0.125 = 13.375$						

$1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \; . \; 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 13.375_{10}$

↑ Binary point

Conversion: Decimal to Binary

- A decimal number can be converted to binary by repeated division by 2 if it is an integer

number	÷2	Remainder	
155	77	1	Least Significant Bit (LSB)
77	38	1	
38	19	0	
19	9	1	
9	4	1	
4	2	0	
2	1	0	
1	0	1	Most Significant Bit (MSB)

Arrange remainders in reverse order

$155_{10} = 10011011_2$

Conversion: Decimal to Binary

- If the number includes a radix point, it is necessary to separate the number into an integer part and a fraction part, each part must be converted differently.

Decimal → Binary

$(27.375)_{10}$ → $(??)_2$

number	÷2	Remainder
27	13	1
13	6	1
6	3	0
3	1	1
1	0	1

Arrange remainders in reverse order: 11011

number	x2	Integer
0.375	0.75	0
0.75	1.50	1
0.50	1.0	1

Arrange in order: 011

⇒ $27.375_{10}=11011.011_2$

Conversion: Octal to Binary

Octal → Binary

345.5602_8 → $(???)_2$

$$\underbrace{3}_{011}\ \underbrace{4}_{100}\ \underbrace{5}_{101}\ .\ \underbrace{5}_{101}\ \underbrace{6}_{110}\ \underbrace{0}_{000}\ \underbrace{2}_{010}$$

$345.5602_8 = 11100101.101110000010_2$

Conversion: Binary to Octal

Binary → Octal

11001110.0101101_2 → $(??)_8$

Note trailing zeros

011 001 110 . 010 110 100

3 1 6 . 2 6 4

Group by 3's
Add leading zeros if necessary

Group by 3's
Add trailing zeros if necessary

$11001110.0101101_2 = 316.264_8$

Conversion: Binary to Hex

Binary → Hex

11100101101.1111010111_2 → $(??)_{16}$

Note trailing zeros

0111 0010 1101 . 1111 0101 1100

7 2 D . F 5 C

Group by 4's
Add leading zeros if necessary

Group by 4's
Add trailing zeros if necessary

$= 72D.F5C_{16}$

Conversion: Hex to Binary

Hex → Binary

$B9A4.E6C_{16}$ → $(??)_2$

1011	1001	1010	0100	.	1110	0110	1100
B	9	*A*	4	↑	*E*	6	*C*

$1011100110100100.1110011011_2$

Conversion: Hex to Decimal

Hex → Decimal

$B63.4C_{16}$ → $(??)_{10}$

16^2	16^1	16^0	16^{-1}	16^{-2}
B (=11)	6	3	4	C (=12)
= 2816 + 96 + 3 + 0.25 + 0.046875 = 2915.296875				

$$11\times16^2+6\times16^1+3\times16^0\,.\,4\times16^{-1}+12\times16^{-2}=2915.296875_{10}$$

Conversion: Activity 1

- Convert the hexadecimal number A59.FCE to binary
- Convert the decimal number 166.34 into binary

Binary Numbers

- How many distinct numbers can be represented by *n* bits?

No. of bits	**Distinct nos.**
1	2 {0,1}
2	4 {00, 01, 10, 11}
3	8 {000, 001, 010, 011, 100, 101, 110, 111}
n	2^n

- Number of permutations double with every extra bit
- 2^n *unique* numbers can be represented by *n* bits

Number System and Computers

- Some tips
  - Binary numbers often grouped in fours for easy reading
  - 1 byte=8-bit, 1 word = 4-byte (32 bits)
  - In computer programs (e.g. Verilog, C) by default decimal is assumed
  - To represent other number bases use

System	Representation	Example for 20
Hexadecimal	0x...	0x14
Binary	0b...	0b10100
Octal	0o… (zero and ‘O’)	0o24

Number System and Computers(2)

- Addresses often written in Hex
  - Most compact representation
  - Easy to understand given their hardware structure
  - For a range 0x000 – 0xFFF, we can immediately see that 12 bits are needed, 4K locations
  - Tip: 10 bits = 1K

Negative Number Representation

- Three kinds of representations are common:
  1. Signed Magnitude (SM)
  2. One's Complement
  3. Two's Complement

Signed Magnitude Representation

[0,1] {..............................}

Sign bit
(left most)

(*n* -1)
magnitude bits

- 0 indicates +ve
- 1 indicates -ve

8 bit representation for +13 is 0 0001101

8 bit representation for -13 is 1 0001101

1’s Complement Notation

Let N be an n-bit number and $\tilde{N}(1)$ be the 1’s Complement of the number. Then,

$$\tilde{N}(1) = 2^n - 1 - |N|$$

- The idea is to leave positive numbers as is, but to *represent negative numbers by the 1’s Complement of their magnitude*.
- *Example*: Let $n = 4$. What is the 1’s Complement representation for +6 and -6?
  - +6 is represented as 0110 (as usual in binary)
  - -6 is represented by 1’s complement of its magnitude (6)

1' s Complement Notation (2)

- 1' s C representation can be computed in 2 ways:
  - ***Method 1***: 1's C representation of -6 is:

    $2^4 - 1 - |N| = (16 – 1 – 6)_{10} = (9)_{10} = (1001)_2$

  - ***Method 2***: For -6, the magnitude = 6 $= (0110)_2$
    - The 1's C representation is obtained by complementing the bits of the magnitude: $(1001)_2$

2’s Complement Notation

Let N be an *n* bit number and Ñ(2) be the 2’s Complement of the number. Then,

$$\tilde{N}(2) = 2^n - |N|$$

- Again, the idea is to leave positive numbers as is, but to *represent negative numbers by the 2’s C of their magnitude*.
- *Example*: Let $n = 5$. What is 2’s C representation for +11 and -13?
  - +11 is represented as 01011 (as usual in binary)
  - -13 is represented by 2’s complement of its magnitude (13)

2' s Complement Notation (2)

- 2' s C representation can be computed in 2 ways:
  - *Method 1*: 2's C representation of -13 is
    $2^5 - |N| = (32 - 13)_{10} = (19)_{10} = (10011)_2$
  - *Method 2*: For -13, the magnitude is
    $13 = (01101)_2$
    - The 2's C representation is obtained by adding 1 to the 1's C of the magnitude
    - $2^5 - |N| = (2^5 - 1 - |N|) + 1 =$ 1's C + 1

$$01101 \xrightarrow[\text{1's C}]{} 10010 \xrightarrow[\text{add 1}]{} 10011$$

Comparing All Signed Notations

4-bit No.	SM	1's C	2's C
0000	+0	+0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

- In all 3 representations, a –ve number has a 1 in MSB location
- To handle –ve numbers using *n* bits,
  - = 2^{n-1} symbols can be used for positive numbers
  - = 2^{n-1} symbols can be used for negative umbers
- In 2' s C notation, only 1 combination used for 0

Unsigned Binary Integers

- Given an n-bit number

$$x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2^1 + x_0 2^0$$

- Range: 0 to $+2^n - 1$
- Example
  - $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1011_2$
    $= 0 + \ldots + 1\times2^3 + 0\times2^2 + 1\times2^1 + 1\times2^0$
    $= 0 + \ldots + 8 + 0 + 2 + 1 = 11_{10}$
- Using 32 bits
  - 0 to +4,294,967,295

2's-Complement Signed Integers

- Given an n-bit number

$$x = -x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_1 2^1 + x_0 2^0$$

- Range: -2^{n-1} to $+2^{n-1} - 1$
- Example
  - $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_2$
    $= -1\times2^{31} + 1\times2^{30} + \ldots + 1\times2^{2} + 0\times2^{1} + 0\times2^{0}$
    $= -2{,}147{,}483{,}648 + 2{,}147{,}483{,}644 = -4_{10}$
- Using 32 bits
  - –2,147,483,648 to +2,147,483,647

2’s-Complement Signed Integers(2)

- Bit 31 is sign bit
  - 1 for negative numbers
  - 0 for non-negative numbers
- Non-negative numbers have the same unsigned and 2’s-complement representation
- Some specific numbers
  - 0: 0000 0000 … 0000
  - –1: 1111 1111 … 1111
  - Most-negative: 1000 0000 … 0000
  - Most-positive: 0111 1111 … 1111

Signed Negation

- Complement and add 1
  - Complement means 1 → 0, 0 → 1

$$x + \overline{x} = 1111...111_2 = -1$$

$$\overline{x} + 1 = -x$$

- Example: negate +2
  - $+2 = 0000\ 0000 \ldots 0010_2$
  - $-2 = 1111\ 1111 \ldots 1101_2 + 1$
    $= 1111\ 1111 \ldots 1110_2$

Sign Extension

- Representing a number using more bits
  - Preserve the numeric value
- In MIPS instruction set
  - `addi`: extend immediate value
  - `lb`, `lh`: extend loaded byte/halfword
  - `beq`, `bne`: extend the displacement
- Replicate the sign bit to the left
  - c.f. unsigned values: extend with 0s
- Examples: 8-bit to 16-bit
  - +2: 0000 0010 => 0000 0000 0000 0010
  - –2: 1111 1110 => 1111 1111 1111 1110