Numbering Systems

- **Numbering systems are characterized by their base number.**
- **In general a numbering system with a base *r* will have *r* different digits (including the 0) in its number set. These digits will range from 0 to *r*-1**

Numbering System	Base	Digits Set
Binary	2	1 0
Octal	8	7 6 5 4 3 2 1 0
Decimal	10	9 8 7 6 5 4 3 2 1 0
Hexadecimal	16	F E D C B A 9 8 7 6 5 4 3 2 1 0

- **Binary Numbers**
  - Each digit (bit) is either 1 or 0
  - Each bit represents a power of 2
  - Every binary number is a sum of powers of 2

2^n	Decimal Value	2^n	Decimal Value
2^0	1	2^8	256
2^1	2	2^9	512
2^2	4	2^{10}	1024
2^3	8	2^{11}	2048
2^4	16	2^{12}	4096
2^5	32	2^{13}	8192
2^6	64	2^{14}	16384
2^7	128	2^{15}	32768

Ref: Bryant and O'Hallaron, Computer Systems: A Programmer's , Third Edition

Converting Binary to Decimal

binary 10101001 = decimal 169:
$(1 \times 2^7) + (1 \times 2^5) + (1 \times 2^3) + (1 \times 2^0) = 128+32+8+1=169$

Convert Unsigned Decimal to Binary

Repeatedly divide the decimal integer by 2. Each remainder is a binary digit in the translated value:

Division	Quotient	Remainder	
37 / 2	18	1	least significant b
18 / 2	9	0	
9 / 2	4	1	
4 / 2	2	0	
2 / 2	1	0	
1 / 2	0	1	most significant bit

37 = 100101

Hexadecimal Integers

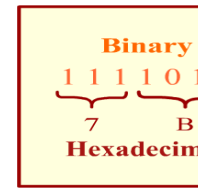
Table 1-5 Binary, Decimal, and Hexadecimal Equivalents.

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

Ref: Bryant and O'Hallaron, Computer Systems: A Programmer's , Third Edition

Translate the binary integer 000101101010011110010100 to hexadecimal

1	6	A	7	9	4
0001	0110	1010	0111	1001	0100



Converting Hexadecimal to Binary

F08AB5

Converting Hexadecimal to Decimal:

Hex 1234 = $(1 \times 16^3) + (2 \times 16^2) + (3 \times 16^1) + (4 \times 16^0)$ =

Decimal 4,660

Converting Decimal to Hexadecimal

Division	Quotient	Remainder	
422 / 16	26	6	LSB
26 / 16	1	A	
1 / 16	0	1	MSB

Decimal 422 = 1A6 hexadecimal

Boolean Algebra

- **Developed by George Boole in 19th Century**
  - Algebraic representation of logic
    - Encode "True" as 1 and "False" as 0

And

- A&B = 1 when both A=1 and B=1

&	0	1
0	0	0
1	0	1

Or

- A|B = 1 when either A=1 or B=1

|	0	1
0	0	1
1	1	1

Not

- ~A = 1 when A=0

~	
0	1
1	0

Exclusive-Or (Xor)

- A^B = 1 when either A=1 or B=1, but not both

^	0	1
0	0	1
1	1	0

Ref: Bryant and O'Hallaron, Computer Systems: A Programmer's , Third Edition

General Boolean Algebras

All of the Properties of Boolean Algebra Apply

- **&** – AND
  - Result is **1** if both operand bits are **1**
- | – OR
  - Result is **1** if either operand bit is **1**

^ – Exclusive OR

Result is **1** if operand bits are different

- ~ – Complement
  - Each bit is reversed
- << – Shift left
  - Multiply by 2
- >> – Shift right
  - Divide by 2

Operate on Bit Vectors

- Operations applied bitwise

```
  01101001      01101001      01101001
& 01010101    | 01010101    ^ 01010101    ~ 01010101
  01000001      01111101      00111100      10101010
```

C expression	Binary expression	Binary result	Hexadecimal result
~0x41	~[0100 0001]	[1011 1110]	0xBE
~0x00	~[0000 0000]	[1111 1111]	0xFF
0x69 & 0x55	[0110 1001] & [0101 0101]	[0100 0001]	0x41
0x69 | 0x55	[0110 1001] | [0101 0101]	[0111 1101]	0x7D

Ref: Bryant and O’Hallaron, Computer Systems: A Programmer’s , Third Edition

- **Operate on Bit Vectors**
  - Operations applied bitwise

```
  01101001    01101001    01101001
& 01010101  | 01010101  ^ 01010101  ~ 01010101
  01000001    01111101    00111100    10101010
```

- **All of the Properties of Boolean Algebra Apply**


```
  01101001    01101001    01101001
& 01010101  | 01010101  ^ 01010101  ~ 01010101
  01000001    01111101    00111100    10101010
```


Ref: Bryant and O'Hallaron, Computer Systems: A Programmer's , Third Edition

- **Data Size:**
  - Word Size
  - Max. Size of Virtual address Range.
  - w-bit word size has virtual address range : 0 to 2^w-1
  -

Multiple data formats:

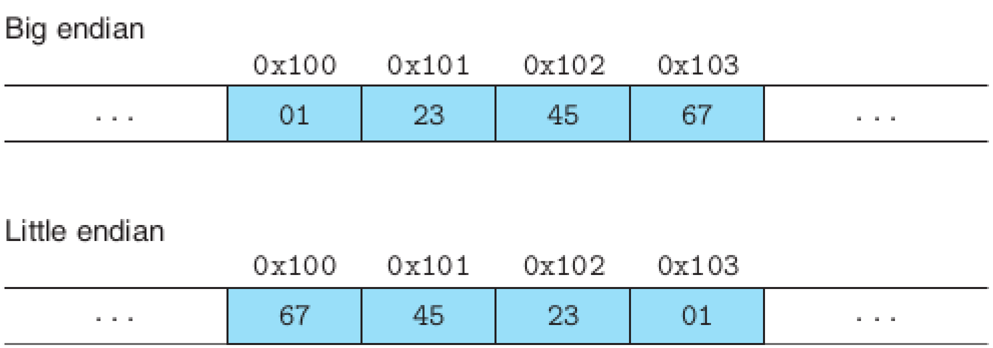
C declaration		Bytes	
Signed	Unsigned	32-bit	64-bit
[signed] char	unsigned char	1	1
short	unsigned short	2	2
int	unsigned	4	4
long	unsigned long	4	8
int32_t	uint32_t	4	4
int64_t	uint64_t	8	8
char *		4	8
float		4	4
double		8	8

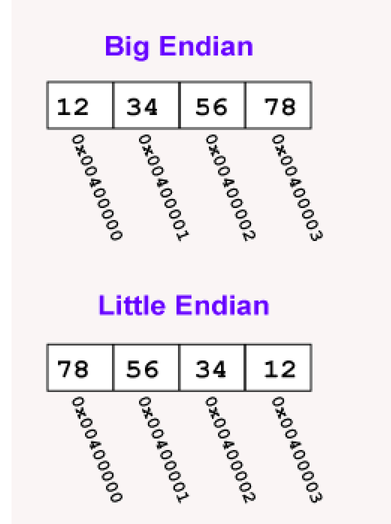
Typical sizes (in bytes) of basic C data types

Ref: Bryant and O'Hallaron, Computer Systems: A Programmer's , Third Edition

Addressing and Byte Ordering

- Some machines choose to store the object in memory ordered from least significant byte to most. while other machines store them from most to least
- **Little endian: L**east significant byte comes first
- **Big endian:** Where the most significant byte comes first.
    - byte orderings used by their machines are totally invisible;
    - byte ordering becomes an issue when
        - binary data are communicated over a network between different machines.
        - when inspecting machine-level programs.
-





Ref: Bryant and O'Hallaron, Computer Systems: A Programmer's , Third Edition

```c
#include <stdio.h>

typedef unsigned char *byte_pointer;

void show_bytes(byte_pointer start, size_t len) {
    int i;
    for (i = 0; i < len; i++)
        printf(" %.2x", start[i]);
    printf("\n");
}

void show_int(int x) {
    show_bytes((byte_pointer) &x, sizeof(int));
}

void show_float(float x) {
    show_bytes((byte_pointer) &x, sizeof(float));
}

void show_pointer(void *x) {
    show_bytes((byte_pointer) &x, sizeof(void *));
}
```

ure 2.4 **Code to print the byte representation of program objects.** This code

Ref: Bryant and O'Hallaron, Computer Systems: A Programmer's , Third Edition