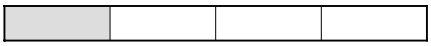
Bit Operations

Stanley Yao
Computer Science Department
University of Arizona

Numbers in Computers

- Analog .vs. digital
- Digital numbers break-downs
  - Number 
  - Byte 
  - Bit 
- Word: length is decided by the register size
  - On lectura (32-bit Machine): 1 word= 4 bytes

Positional Numbering System

- The position of the digit in the number indicates the magnitude of its value
  - $238 = 2*10^2+3*10^1+8*10^0$
  - $d_{n-1}...d_1d_0 = d_{n-1}*r^{n-1}+...+d_1*r^1+d_0*r^0$
  - r: radix or base
- Binary: r = 2
- Octal: r = 8
- Decimal: r = 10
- Hex: r = 16

Binary, Octal and Hex

Bin	000	001	010	011	100	101	110	111	1000
Oct	0	1	2	3	4	5	6	7	10

Bin	0000	0001	0010	0011	0100	0101	0110	0111	
Hex	0	1	2	3	4	5	6	7	
Bin	1000	1001	1010	1011	1100	1101	1110	1111	10000
Hex	8	9	A	B	C	D	E	F	10

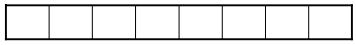
Binary Values

- $2^0 = 1$
- $2^{10} = 1024$ (K - Kilo)
- $2^{20} = 1048576$ (M - Mega)
- $2^{30} =$ ... ... (G - Giga)
- $2^{40} =$ ... ... (T - Tera)
- $2^{50} =$ ... ... (P - Peta)

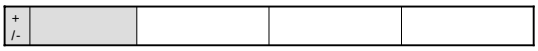
Signed & Unsigned Numbers

- char: -128 ~ 127

- unsigned char: 0 ~ 255



- int:



Bitwise Operators

A	B	A & B	A | B	A ^ B	~A
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

Bitwise Operators (cont.)

- a=10110010 and b=01100100

	1	0	1	1	0	0	1	0
	0	1	1	0	0	1	0	0
a & b	0	0	1	0	0	0	0	0
a | b	1	1	1	1	0	1	1	0
a ^ b	1	1	0	1	0	1	1	0
~a	0	1	0	0	1	1	0	1

Shift

- a << b
  - Shift “a” bitwise left for “b” positions.
  - “b” must be positive
  - Vacated bits are filled with 0
- a >> b
  - Shift “a” bitwise left for “b” positions.
  - “b” must be positive
  - For unsigned “a”, vacated bits are filled with 0
  - For signed “a”, vacated bits are filled with sign bit (arithmetic shift) or 0 (logical shift), depending on the machine

Applications

- Turn on a bit: |
- Turn off a bit: &
- Test a bit: &
- Short forms:
  - a = a & b ➔ a &= b
  - a = a | b ➔ a |= b
  - a = a ^ b ➔ a ^= b

Bit Fields

```
struct {
    unsigned int x:3;
    unsigned int y:2;
    unsigned int z:1;
} foo;
```

- Almost everything about fields is implementation-dependent
  - Alignment
  - Compact
  - Field orders in the memory
- Fields have no addresses. & can’t be used to fields

Acknowledgement

- John H. Hartman, *Classnotes for Csc352-Spring03*, CS Dept., University of Arizona, 2003
- Brian W. Kernighan, Dennis M. Ritchie, *The C Programming Language (2nd Ed.),* Prentice Hall, 1988