EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF
SCIENCE
Communication Networks

Informatik der Systeme –
Chapter 1: Units and Number Formats

Prof. Dr. Michael Menth

http://kn.inf.uni-tuebingen.de

Principal Metric Prefixes

Exp.	Explicit	Prefix	Exp.	Explicit	Prefix
10^{-3}	0.001	milli	10^{3}	1,000	Kilo
10^{-6}	0.000001	micro	10^{6}	1,000,000	Mega
10^{-9}	0.000000001	nano	10^{9}	1,000,000,000	Giga
10^{-12}	0.000000000001	pico	10^{12}	1,000,000,000,000	Tera
10^{-15}	0.000000000000001	femto	10^{15}	1,000,000,000,000,000	Peta
10^{-18}	0.000000000000000001	atto	10^{18}	1,000,000,000,000,000,000	Exa
10^{-21}	0.000000000000000000001	zepto	10^{21}	1,000,000,000,000,000,000,000	Zetta
10^{-24}	0.000000000000000000000001	yocto	10^{24}	1,000,000,000,000,000,000,000,000	Yotta

Source: A. Tanenbaum, Structured Computer Organization, 5/E, © Pearson

Information Units

- Storage mostly as 2^X bytes (8 bits)
  - Sometimes "rounded" to $10^{X/10*3}$ bytes
  - Kilobyte – 2^{10} or 10^3 bytes
  - Megabyte – 2^{20} or 10^6 bytes
  - Gigabyte – 2^{30} or 10^9 bytes
  - Terabyte – 2^{40} or 10^{12} bytes
  - Petabyte – 2^{50} or 10^{15} bytes
  - Exabyte – 2^{60} or 10^{18} bytes
- Transmission speeds always come in 10^X bits per second
- Convention in MIPS architecture (specific to ISA)
  - Word: 32 bits
  - Halfword: 16 bits

Hexadecimal Encoding of Bit Strings

- Base 16
  - Compact representation of bit strings
  - 4 bits per hex digit

0	0000	4	0100	8	1000	c	1100
1	0001	5	0101	9	1001	d	1101
2	0010	6	0110	a	1010	e	1110
3	0011	7	0111	b	1011	f	1111

- Notation of hex numbers
  - 72_{16}, 72_{hex}, 72h, 72_{H}, 0x72, "72, $72 and X'72,
  - Prefix 0x most widely used in programming
- Example: 0x ECA8 6420
  - 1110 1100 1010 1000 0110 0100 0010 0000

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Unsigned Binary Integers

- Unsigned binary integer
  $$x = x_{n-1}x_{n-2}\dots x_1x_0$$
- Interpretation as decimal
  $$X = x_{n-1}\cdot 2^{n-1} + \cdots + x_0 \cdot 2^0$$
- Range: 0 to $+2^n - 1$
- Example using 32 bits
  - $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1011_2$
    $= 0 + \ldots + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 1\times 2^0$
    $= 0 + \ldots + 8 + 0 + 2 + 1 = 11_{10}$
  - Range: 0 to +4,294,967,295

Hex	Binary	Decimal
0x00000000	0…0000	0
0x00000001	0…0001	1
0x00000002	0…0010	2
0x00000003	0…0011	3
0x00000004	0…0100	4
0x00000005	0…0101	5
0x00000006	0…0110	6
0x00000007	0…0111	7
0x00000008	0…1000	8
0x00000009	0…1001	9
	…	
0xFFFFFFFC	1…1100	
0xFFFFFFFD	1…1101	
0xFFFFFFFE	1…1110	
0xFFFFFFFF	1…1111	

2s-Complement Signed Integers

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

- Range: -2^{n-1} to $+2^{n-1} - 1$
  - $-(-2^{n-1})$ can't be represented
- Example using 32 bits
  - $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_2$
    $= -1\times2^{31} + 1\times2^{30} + \ldots + 1\times2^2 + 0\times2^1 + 0\times2^0$
    $= -2{,}147{,}483{,}648 + 2{,}147{,}483{,}644$
    $= -4_{10}$
  - Range: –2,147,483,648 to +2,147,483,647
- Bit 31 is sign bit
  - 1 for negative numbers
  - 0 for non-negative numbers
  - Non-negative numbers have the same unsigned and 2s-complement representation

- Some specific numbers
  - 0: 0000 0000 … 0000
  - –1: 1111 1111 … 1111
  - Most-negative: 1000 0000 … 0000
  - Most-positive: 0111 1111 … 1111
- Complement ($\bar{x}$) means 1 → 0, 0 → 1
- "Signed negation"
  - $x + \bar{x} = 1111\ldots111_2 = -1_{10} \Rightarrow \bar{x} + 1_{10} = -x$
  - Complement and add 1 to obtain negated value

Negation of 2'sc Signed Integers

2'sc binary	decimal
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7

$-2^3 =$ 1000

$-(2^3 - 1) =$ 1001

$2^3 - 1 =$ 0111

1010 → Complement all the bits → 0101 → and add a 1 → 0110

0101 → Complement all the bits → 1010 → and add a 1 → 1011

Sign Extension

- Representing a number using more bits
  - Preserve the numeric value
  - Convert from byte to halfword or word

- Unsigned binary integers
  - Extend with 0s to the left (zero extension)

- 2'sc signed integers
  - Extend with the sign bit to the left (sign extension)
  - Examples: 8-bit to 16-bit
    - +2: 0000 0010 ⇒ 0000 0000 0000 0010
    - –2: 1111 1110 ⇒ 1111 1111 1111 1110

Binary Floating Point Numbers

- What's 18 in binary?
  - 18/2 = 9 remainder 0 (least significant bit)
  - 9/2 = 4 remainder 1
  - 4/2 = 2 remainder 0
  - 2/2 = 1 remainder 0
  - 1/2 = 0 remainder 1 (most significant bit)

  = 10010

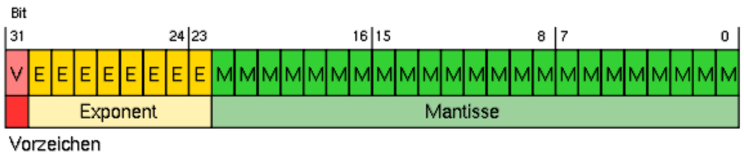
- What's 0.4 in binary?
  - 0.4·2 = 0.8 = 0 remainder 0.8 (most significant bit)
  - 0.8·2 = 1.6 = 1 remainder 0.6
  - 0.6·2 = 1.2 = 1 remainder 0.2
  - 0.2·2 = 0.4 = 0 remainder 0.4
  - 0.4·2 = 0.8 = 0 remainder 0.8
  - 0.8·2 = 1.6 = 1 remainder 0.6 (least significant bit)
  - …

  = 0.011001100110011...

- What's 18.4 in binary?
  - = 10010.011001100110011...

IEEE 754 Format

- Definition



  - Sign V, exponent E, significand M
    - r : number of E-bits
    - p : number of M-bits
  - Precision
    - Single: 32 bit, r=8, p=23
    - Double: 64 bit, r=11, p=52
  - Interpretation: $x = s \cdot f \cdot 2^e$
    - $s = (-1)^V$
    - $f = 1 + \mathrm{M}/2^p$
    - $e = E - B$
      - $B = 2^{r-1} - 1$ (exponential offset)
  - Exceptions
    - E=0, M=0: +/-Null
    - $E = 2^r - 1, M = 0$: +/-infinity
    - $E = 2^r - 1, M > 0$: not a number (NaN)
    - E=0, M>0: denormalized number, interpretation: $(-1)^V \times M / 2^p \times 2^{1-B}$

- Conversion from decimals to IEEE 754
- Example
  - $18.4 = 10010.0110011001100110011\ _2$
  - Normalized binary float
    - $1{,}00100110011001100110011 \ldots \cdot 2^{(4_{10})}$
  - IEEE 754 representation (single precision)
    - $B = 2^{r-1} - 1 = 2^7 - 1 = 127$
    - $\mathrm{V} = 0$
    - $E = \mathrm{e} + \mathrm{B} = 4 + 127 = 131 = 10000011_2$
    - $M = 00100110011001100110011_2$