M1 – Computers and Data

Data Representation

Module Outline

- Architecture vs. Organization.
- Computer system and its submodules.
- Concept of frequency.
- Processor performance equation.
- **Representation of information – characters, signed and unsigned integers.**
  - **IEEE 754 floating point standard.**

Objective

- How are program data (integers, strings, floats, addresses) represented (encoded) in an object file and during program execution?

Character Data

- American Standard Code for Information Interchange (ASCII) 7-bit format

Character Data

- American Standard Code for Information Interchange (ASCII) 7-bit format
- Unicode 8/16/32-bit formats (UTF-8, UTF-16, UTF-32)
  - Repertoire of >110,000 characters covering 100 scripts and various symbols
  - All ASCII characters included in UTF-8.

Integer Data

- Signed and Unsigned
  - int x;
  - unsigned int y;

Signed Integer Data

- Sign Magnitude Representation

Signed Integer Data

- Sign Magnitude Representation

$$x_{n-1}x_{n-2}\ldots x_2x_1x_0$$

Signed Integer Data

- Sign Magnitude Representation

$$\overbrace{x_{n-1}}^{\textbf{Sign bit}}\ \overbrace{\underbrace{x_{n-2}}_{\textbf{MSB}} \ldots x_2\, x_1\, \underbrace{x_0}_{\textbf{LSB}}}^{\textbf{Magnitude bits}}$$

Signed Integer Data

- Sign Magnitude Representation

$$\overbrace{x_{n-1}}^{\textbf{Sign bit}}\ \overbrace{\underbrace{x_{n-2}}_{\textbf{MSB}}\ldots x_2\, x_1\, \underbrace{x_0}_{\textbf{LSB}}}^{\textbf{Magnitude bits}}$$

$$(-1)^{x_{n-1}} \times [x_{n-2}\cdot 2^{n-2} + \ldots + x_1 \cdot 2^1 + x_0 \cdot 2^0]$$

Signed Integer Data

- Sign Magnitude Representation

$$\overbrace{x_{n-1}}^{\text{Sign bit}}\ \overbrace{\underbrace{x_{n-2}}_{\text{MSB}} \ldots x_2\, x_1\, \underbrace{x_0}_{\text{LSB}}}^{\text{Magnitude bits}}$$

$$(-1)^{x_{n-1}} \times [x_{n-2} \cdot 2^{n-2} + \ldots + x_1 \cdot 2^1 + x_0 \cdot 2^0]$$

Binary to/from Decimal conversion (8-bit values):

(a) 0x4C (b) –27

Signed Integer Data – Problems

- Sign bit uses up one bit space.

Signed Integer Data – Problems

- Sign bit uses up one bit space.
- Arithmetic circuits need extra steps to identify sign of the result.

Signed Integer Data – Problems

- Sign bit uses up one bit space.
- Arithmetic circuits need extra steps to identify sign of the result.
- +0 and -0 exist.

Two's Complement Representation

- Leading zeros mean positive and leading ones mean negative

Two's Complement Representation

- Leading zeros mean positive and leading ones mean negative

0000 → 0
0001 → 1
0010 → 2
.....
0111 → 7

Two's Complement Representation

- Leading zeros mean positive and leading ones mean negative

$$\begin{aligned} 0000 &\rightarrow 0 \\ 0001 &\rightarrow 1 \\ 0010 &\rightarrow 2 \\ &\ldots\ldots \\ 0111 &\rightarrow 7 \\ 1000 &\rightarrow -8 \end{aligned}$$

Two's Complement Representation

- Leading zeros mean positive and leading ones mean negative

0000 → 0
0001 → 1
0010 → 2
.....
0111 → 7
1000 → -8
1001 → -7
.....
1110 → -2
1111 → -1

Two's Complement Representation

- Leading zeros mean positive and leading ones mean negative

0000	→ 0
0001	→ 1
0010	→ 2
.....	
0111	→ 7
1000	→ -8
1001	→ -7
.....	
1110	→ -2
1111	→ -1

0000 0000	→ 0
0000 0001	→ 1
0000 0010	→ 2
.....	
0111 1110	→ 126
0111 1111	→ 127
1000 0000	→ -128
1000 0001	→ -127
1000 0010	→ -126
.....	
1111 1110	→ -2
1111 1111	→ -1

Two's Complement Representation

$$\boxed{x_b = x_{n-1}\, x_{n-2} \ldots x_2\, x_1\, x_0}$$

$$\boxed{x_{decimal} = -x_{n-1} \cdot 2^{n-1} + x_{n-2} \cdot 2^{n-2} + \ldots + x_1 \cdot 2^1 + x_0 \cdot 2^0}$$

Two's Complement Representation

$$x_b = x_{n-1}\, x_{n-2} \ldots x_2\, x_1\, x_0$$

$$x_{decimal} = -x_{n-1} \cdot 2^{n-1} + x_{n-2} \cdot 2^{n-2} + \ldots + x_1 \cdot 2^1 + x_0 \cdot 2^0$$

Example

Convert 17, -21, 32 into their 2's complement notation.

Two's Complement Representation

$$\boxed{x_b = x_{n-1}\, x_{n-2} \ldots x_2\, x_1\, x_0}$$

$$\boxed{x_{decimal} = -x_{n-1} \cdot 2^{n-1} + x_{n-2} \cdot 2^{n-2} + \ldots + x_1 \cdot 2^1 + x_0 \cdot 2^0}$$

- Observe: Pick an 8-bit, 2's complement binary number. Invert its bits: 0 → 1 and 1 → 0. Add both these numbers in binary. What is the result?

Two's Complement Representation

x	**1111 1111 1111 1111 1111 1111 1111 1100**

Two's Complement Representation

x	**1111 1111 1111 1111 1111 1111 1111 1100**
$\bar{x}$	**0000 0000 0000 0000 0000 0000 0000 0011**

Two's Complement Representation

x	**1111 1111 1111 1111 1111 1111 1111 1100**
$\bar{x}$	**0000 0000 0000 0000 0000 0000 0000 0011**

$$x + \bar{x} = -1 \longrightarrow (\bar{x} + 1) = -x$$

$\bar{x}$ +1	**0000 0000 0000 0000 0000 0000 0000 0100** $\longrightarrow$	**x = -4**

Two's Complement Representation

x	1111 1111 1111 1111 1111 1111 1111 1100
$\bar{x}$	0000 0000 0000 0000 0000 0000 0000 0011

$x + \bar{x} = -1$ → $(\bar{x} + 1) = -x$

$\bar{x}$ +1 | 0000 0000 0000 0000 0000 0000 0000 0100 → x = -4

Binary to/from Decimal conversion:

0xFFFFFBD5

Sign Extension and Zero Extension

Binary to Decimal conversion:

(a) 1 1011 (b) 11 1011 (c) 111 1011 (d) 1111 1011

Sign Extension and Zero Extension

Binary to Decimal conversion:

(a) 1 1011 (b) 11 1011 (c) 111 1011 (d) 1111 1011

- Replicate the Most Significant Bit to the left.

Sign Extension and Zero Extension

Binary to Decimal conversion:

(a) 1 1011 (b) 11 1011 (c) 111 1011 (d) 1111 1011

- Replicate the Most Significant Bit to the left.
- Sign extension
  - 1001, 11001, 111001 ... = -7
  - 11......1001 = - 7

Sign Extension and Zero Extension

Binary to Decimal conversion:

(a) 1 1011 (b) 11 1011 (c) 111 1011 (d) 1111 1011

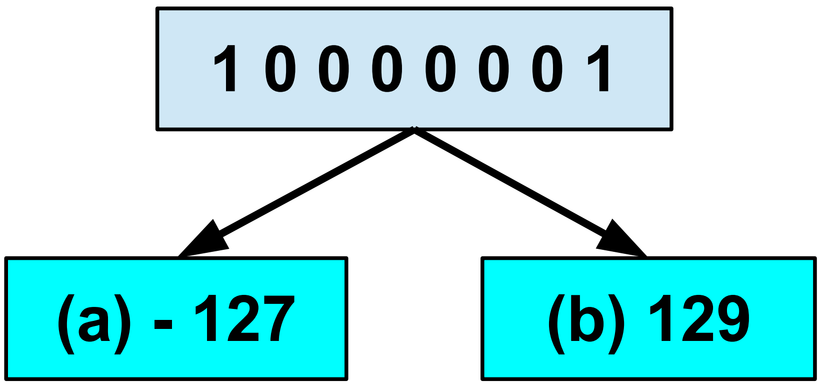
- Replicate the Most Significant Bit to the left.
- Sign extension
  - 1001, 11001, 111001 ... = -7
  - 11......1001 = - 7
- Zero extension
  - 0111 = 7, 00111 = 7, 000111 = 7, ...

Unsigned Integer Data

- Interpret the 8-bit quantity 0x81 as (a) a signed integer, and (b) an unsigned integer.

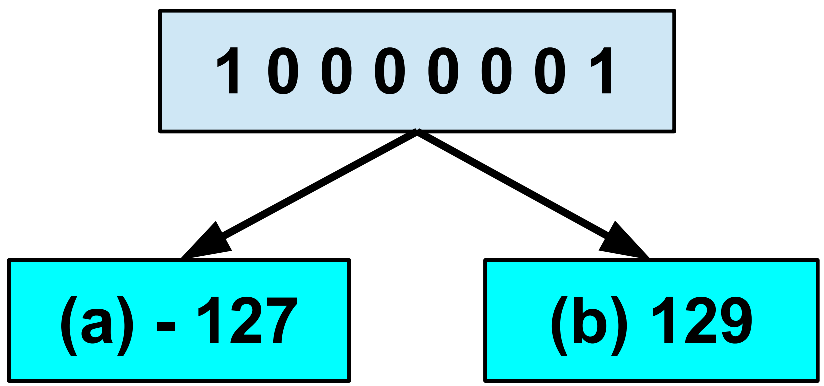
Unsigned Integer Data

- Interpret the 8-bit quantity 0x81 as (a) a signed integer, and (b) an unsigned integer.



Unsigned Integer Data

- Interpret the 8-bit quantity 0x81 as (a) a signed integer, and (b) an unsigned integer.



- What are the max and min unsigned values of 16bit unsigned data?

Byte Ordering

...		0 0	0 0	4 7	A 0		...
	99	100	101	102	103	104	

- How does one read the 4 bytes at address 100?

Byte Ordering

...		0 0	0 0	4 7	A 0		...
	99	100	101	102	103	104	

- How does one read the 4 bytes at address 100?

0 0 0 0 4 7 A 0 vs. **A 0 4 7 0 0 0 0**

Byte Ordering

...		0 0	0 0	4 7	A 0		...
	99	100	101	102	103	104	

- How does one read the 4 bytes at address 100?

0 0 0 0 4 7 A 0	**vs.**	**A 0 4 7 0 0 0 0**
MSB is at 100		LSB is at 100
27566	**vs.**	**−1605959680**

Byte Ordering

...		0 0	0 0	4 7	A 0		...
	99	100	101	102	103	104	

- How does one read the 4 bytes at address 100?

0 0 0 0 4 7 A 0 **vs.** **A 0 4 7 0 0 0 0**

MSB is at 100 LSB is at 100

27566 **vs.** **−1605959680**

Big Endian notation **vs.** **Little Endian notation**

Biased Representation

- The most negative value is represented by 0.

Two's Complement

0000 0000 →	**0**
0000 0001 →	**1**
0000 0010 →	**2**
.....	
0111 1110 →	**126**
0111 1111 →	**127**
1000 0000 →	**-128**
1000 0001 →	**-127**
1000 0010 →	**-126**
.....	
1111 1110 →	**-2**
1111 1111 →	**-1**

Biased Representation

- The most negative value is represented by 0.

Two's Complement

0000 0000	→ 0
0000 0001	→ 1
0000 0010	→ 2
.....	
0111 1110	→ 126
0111 1111	→ 127
1000 0000	→ -128
1000 0001	→ -127
1000 0010	→ -126
.....	
1111 1110	→ -2
1111 1111	→ -1

Biased Notation (Biased by 128)

1000 0000	→ 0
1000 0001	→ 1
1000 0010	→ 2
.....	
1111 1110	→ 126
1111 1111	→ 127
0000 0000	→ -128
0000 0001	→ -127
0000 0010	→ -126
.....	
0111 1110	→ -2
0111 1111	→ -1

(Excess 128)

Real Data

- 3.14159265 ... (pi), 2.71828 ... (e)
- 0.00001 = 1.0×10^{-5}
- 3,155,760,000 = 3.15576×10^{-9}

Real Data

- 3.14159265 ... (pi), 2.71828 ... (e)
- $0.00001 = 1.0 \times 10^{-5}$
- $3{,}155{,}760{,}000 = 3.15576 \times 10^{-9}$
- Scientific form of a real number
  - Single digit before the decimal point
  - 1.0×10^{-5}, 0.1×10^{-4}, 0.01×10^{-3}, ...

Real Data

- 3.14159265 ... (pi), 2.71828 ... (e)
- $0.00001 = 1.0 \times 10^{-5}$
- $3,155,760,000 = 3.15576 \times 10^{-9}$
- Scientific form of a real number
  - Single digit before the decimal point
  - 1.0×10^{-5}, 0.1×10^{-4}, 0.01×10^{-3}, ...
- Normalized representation
  - No leading zeros
  - 1.0×10^{-5}
  - Not 0.1×10^{-4}, 0.01×10^{-3}, ...

Real Data – Binary Notation

- $7.5_{10} = (\_\_\_\_\_\_\_\_\_)_2$
- $7.25_{10} = (\_\_\_\_\_\_\_\_\_)_2$
- $7.125_{10} = (\_\_\_\_\_\_\_\_\_)_2$
- $7.75_{10} = (\_\_\_\_\_\_\_\_\_)_2$
- $7.375_{10} = (\_\_\_\_\_\_\_\_\_)_2$
- $7.625_{10} = (\_\_\_\_\_\_\_\_\_)_2$
- $7.875_{10} = (\_\_\_\_\_\_\_\_\_)_2$

Real Data – Binary Notation

- $7.625_{10} = 111.101_2$

Real Data – Binary Notation

- $7.625_{10} = 111.101_2$
- 111.101×2^0, 1111.01×2^{-1}, 11.1101×2^1, ...

Real Data – Binary Notation

- $7.625_{10} = 111.101_2$
- 111.101×2^0, 1111.01×2^{-1}, 11.1101×2^1, *...*
- Scientific form
  - 1.11101×2^2, 0.111101×2^3, 0.0111101×2^4, ...

Real Data – Binary Notation

- $7.625_{10} = 111.101_2$
- 111.101×2^0, 1111.01×2^{-1}, 11.1101×2^1, ...
- Scientific form
  - 1.11101×2^2, 0.111101×2^3, 0.0111101×2^4, ...
- Normalized representation
  - 1.11101×2^2

Real Data – Binary Notation

- $7.625_{10} = 111.101_2$
- 111.101×2^0, 1111.01×2^{-1}, 11.1101×2^1, ...
- Scientific form
  - 1.11101×2^2, 0.111101×2^3, 0.0111101×2^4, ...
- Normalized representation
  - 1.11101×2^2
- Computers use Normalized representation to store floats.
  - $1.f \times 2^e$
  - f: fraction, e: exponent.

Floating Point Representation

31	30 23	22 21 .... 1 0
s	**exponent**	**fraction**

$$(-1)^{s} \times 1.f \times 2^{e-127}$$

Floating Point Representation

31	30 23	22 21 .... 1 0
s	**exponent**	**fraction**

$$(-1)^s \times 1.f \times 2^{e-127}$$

- IEEE 754 single precision format
- Exponent is biased by 127

IEEE 754 FP Representation

- Convert 0.5 into binary. Represent in IEEE 754 single precision floating point representation.

IEEE 754 FP Representation

- Convert 0.5 into binary. Represent in IEEE 754 single precision floating point representation.
- Binary = 0.1 = 1.0 x 2^{-1}.
- s=?, f = ?, e = ?

IEEE 754 FP Representation

- Convert 0.5 into binary. Represent in IEEE 754 single precision floating point representation.
- Binary = 0.1 = 1.0 x 2^{-1}.
- s=?, f = ?, e = ?
- s=0, f=0, e = (-1) + 127 = 126

IEEE 754 FP Representation

- Convert 0.5 into binary. Represent in IEEE 754 single precision floating point representation.
- Binary = 0.1 = 1.0 x 2^{-1}.
- s=?, f = ?, e = ?
- s=0, f=0, e = (-1) + 127 = 126

0	**0 1 1 1 1 1 1 0**	**0 0 ... 0 0**

0x3F000000

IEEE 754 FP Representation

- Convert 0.75 into IEEE 754 SP FP.
- Convert 0xBDCCCCCC into decimal.

IEEE 754 FP Representation

- Convert 0.75 into IEEE 754 SP FP.
- Convert 0xBDCCCCCC into decimal.
- Four special values

Exponent	Fraction	Value
0	0	Zero (± 0)
255	0	Infinity (± ∞)
0	≠ 0	Denormal numbers (± 0.M x 2^{-126})
255	≠ 0	NaN (0/0 or √-1)

IEEE 754 Double Precision

63	62 52	51 50 .... 1 0
s	**exponent**	**fraction**

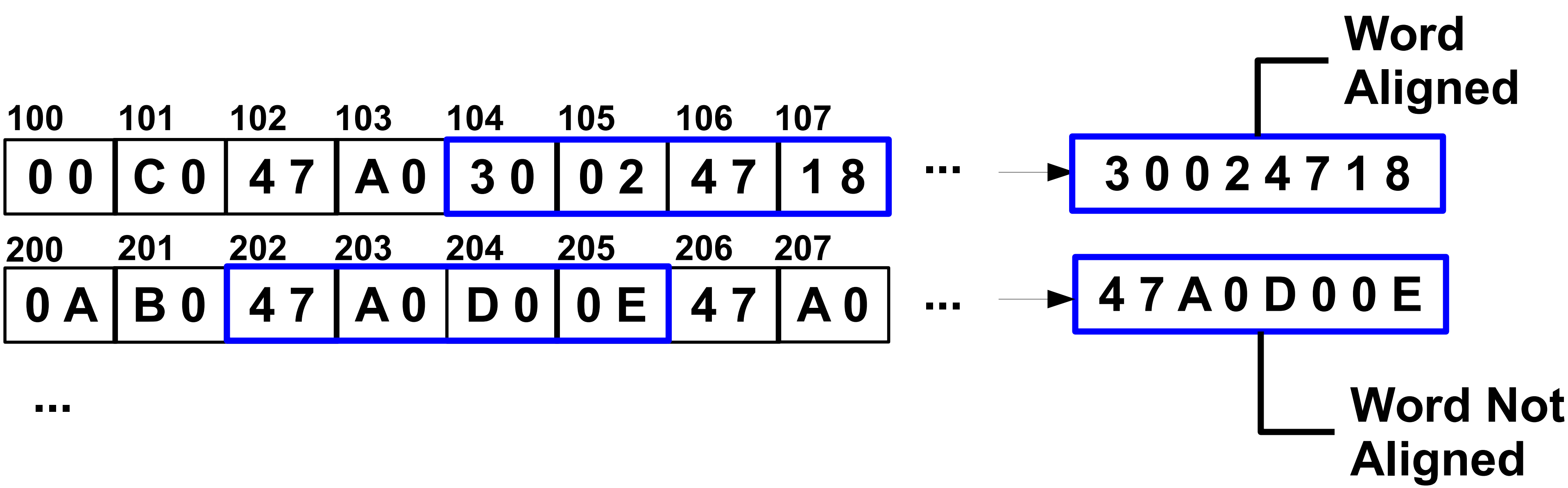
$$(-1)^s \times 1.f \times 2^{e-1023}$$

Module Outline

- Architecture vs. Organization.
- Computer system and its submodules.
- Concept of frequency.
- Processor performance equation.
- Representation of information – characters, signed and unsigned integers.
  - IEEE 754 floating point standard.

Word Alignment

- Word size = 32 bits
- Words can be stored linearly in addresses 0x00, 0x04, 0x08, 0x0C, 0x10, ...
- These addresses are 'word boundaries'



Real Data Representation – Recap

- What are the place values of 7 positions after the decimal point in a binary real number?

2^n	2^{n-1}	2^{n-2}	...	2^0	.	2^{-1}	2^{-2}	2^{-3}	...

decimal point

Binary to Decimal conversion:

(a) 1.10 (b) 11.11 (c) 111.1011 (d) 111.100110011001
(e) 111.110111011011101 ...

Real Data Representation – Recap

- Convert into IEEE 754 SP FP:
  - 13.1, 12.2, 11.3, 10.4, 9.5, 8.6, 7.7, 6.8, 5.9