

Byte Ordering

Jin-Soo Kim (jinsookim@skku.edu)
Computer Systems Laboratory
Sungkyunkwan University
<http://csl.skku.edu>



Memory Model

Physical memory

- DRAM chips can read/write 4, 8, 16 bits.
- DRAM modules can read/write 64 bits.



Programmer's view of memory

- Conceptually very large array of bytes
- Stored-program computers: keeps program codes and data in memory.
- Running programs share the physical memory
- OS handles memory allocation and management



Machine Words

- **Machine has “word size”**
 - Nominal size of integer-valued data
 - Including addresses (= pointer size)
 - Most current machines use 32 bits (4 bytes) words
 - Limits addresses to 4GB
 - Becoming too small for memory-intensive applications
 - High-end systems use 64 bits (8 bytes) words
 - Potential address space $\approx 1.8 \times 10^{19}$ bytes
 - x86-64 machines support 48-bit addresses: 256 Terabytes
 - Machines support multiple data formats
 - Fractions or multiples of word size
 - Always integral number of bytes

Data Representations

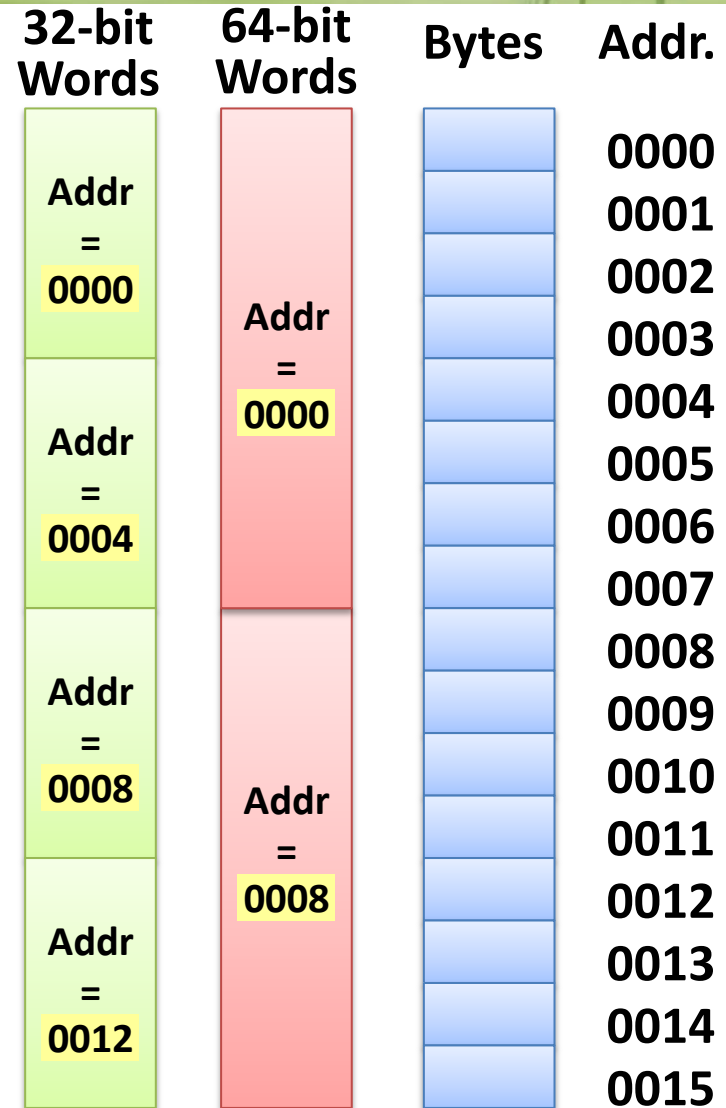
▪ Sizes of C Objects (in bytes)

• C Data Type	Typical 32-bit	Intel IA-32	x86-64
– char	1	1	1
– short	2	2	2
– int	4	4	4
– long	4	4	8
– long long	8	8	8
– float	4	4	4
– double	8	8	8
– long double	8	10/12	10/16
– char *	4	4	8
or any other pointer			

Word-level Memory Access

Addresses specify byte locations

- Address of first byte in word
- Addresses of successive words differ by 4 (32-bit) or 8 (64-bit)
- Usually, addresses should be aligned to the word boundary



Byte Ordering

- How should bytes within multi-byte word be ordered in memory?
- Conventions
 - **Big Endian:** Sun, PowerPC Mac, Internet
 - **Little Endian:** x86
- **Note:**
 - Alpha and PowerPC can run in either mode, with the byte ordering convention determined when the chip is powered up.
 - Problem when the binary data is communicated over a network between different machines.

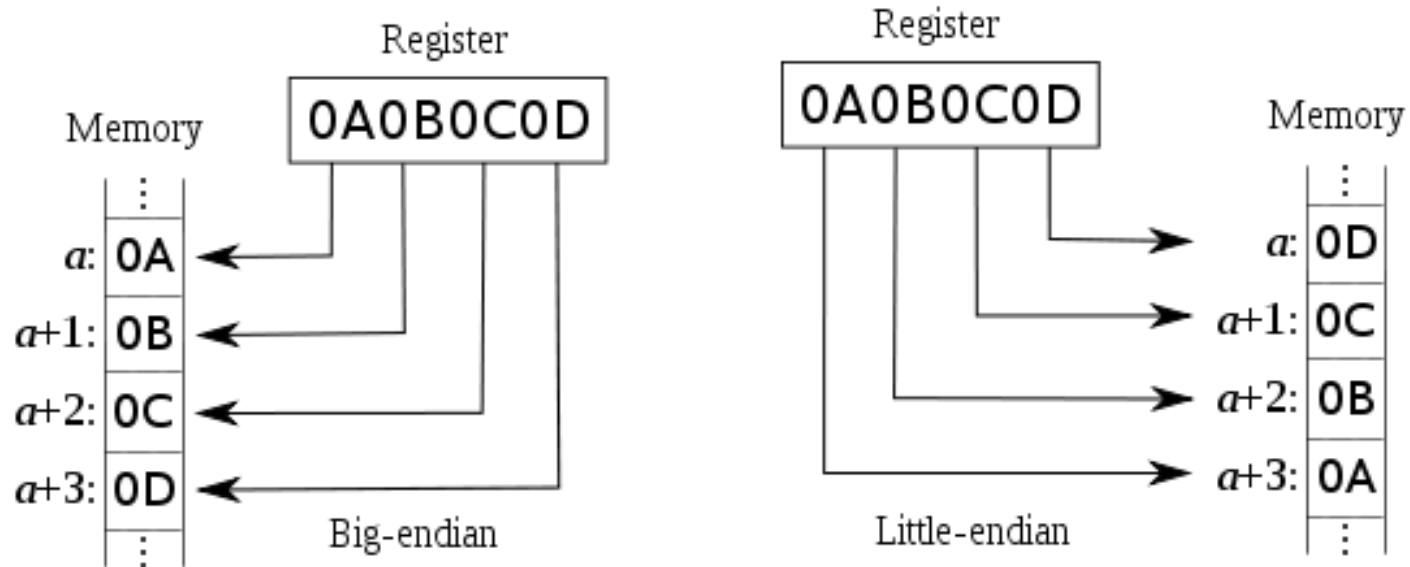
Byte Ordering Example (1)

- **Big endian**

- Least significant byte has highest address

- **Little endian**

- Least significant byte has lowest address



Byte Ordering Example (2)

Disassembly

- Text representation of binary machine code
- Generated by program that reads the machine code

Example fragment

Address	Instruction Code	Assembly Rendition
8048365:	5b	pop %ebx
8048366:	81 c3 ab 12 00 00	add \$0x12ab, %ebx
804836c:	83 bb 28 00 00 00 00	cmpl \$0x0, 0x28(%ebx)

Deciphering numbers

- Value:
- Pad to 32 bits:
- Split into bytes:
- Reverse:

0x12ab
0x000012ab
00 00 12 ab
ab 12 00 00

Byte Ordering Example (3)

- What is the output of this program?
 - Solaris/SPARC: ?
 - Linux/x86: ?

```
#include <stdio.h>

union {
    int i;
    unsigned char c[4];
} u;

int main () {
    u.i = 0x12345678;
    printf ("%x %x %x %x\n", u.c[0], u.c[1], u.c[2], u.c[3]);
}
```

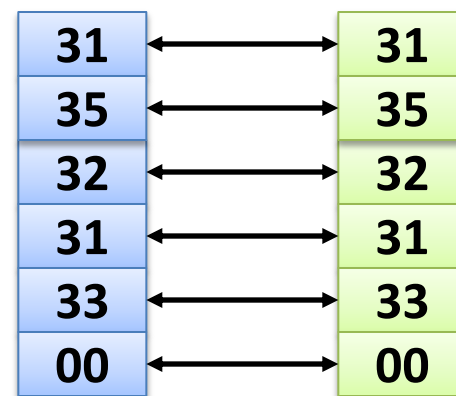
Representing Strings

■ Strings in C

- Represented by array of characters
- Each character encoded in ASCII format
 - Standard 7-bit encoding of character set
 - Character "0" has code 0x30
 - Digit i has code $0x30+i$
- String should be null-terminated
 - Final character = 0x00

`char S[6] = "15213";`

Linux/Alpha S Sun S



■ Compatibility

- Byte ordering not an issue