Representations in memory, pointers, strings
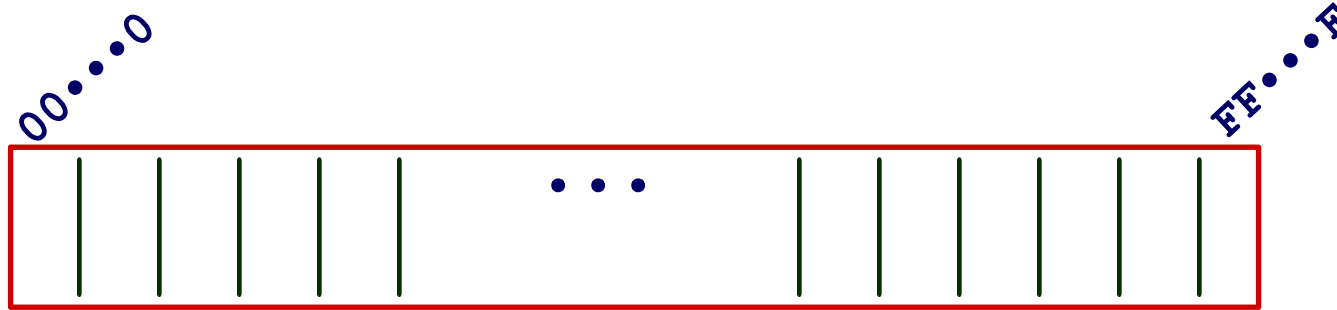
CENG331 - Computer Organization

Instructor:

Murat Manguoglu   (Section 1)

Byte-Oriented Memory Organization

00•••0   FF•••F

- **Programs refer to data by address**

  - Conceptually, envision it as a very large array of bytes

    - In reality, it's not, but can think of it that way

  - An address is like an index into that array

    - and, a pointer variable stores an address

- **Note: system provides private address spaces to each "process"**

  - Think of a process as a program being executed

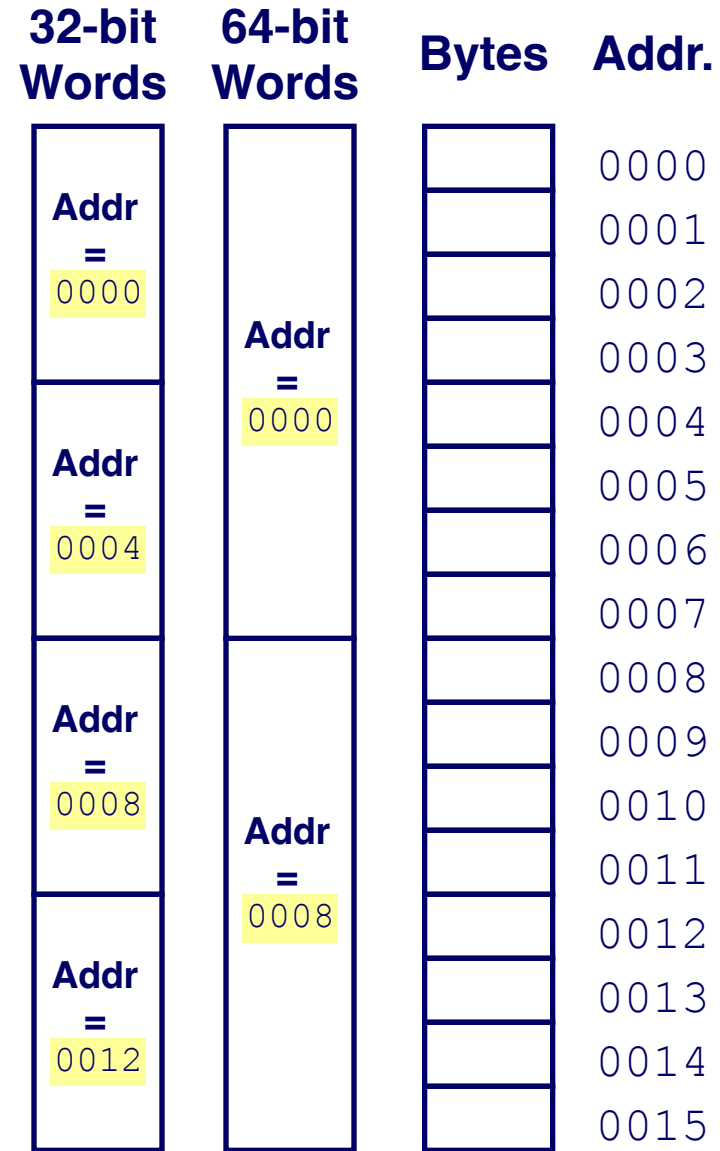 - So, a program can clobber its own data, but not that of others

Machine Words

- **Any given computer has a "Word Size"**
  - Nominal size of integer-valued data
    - and of addresses (i.e. pointers)

  - Until recently, most machines used 32 bits (4 bytes) as word size
    - Limits addresses to 4GB (2^{32} bytes)

  - Increasingly, machines have 64-bit word size
    - Potentially, could have 18 PB (petabytes) of addressable memory
    - That's 18.4×10^{15}

  - Machines still support multiple data formats
    - Fractions or multiples of word size
    - Always integral number of bytes

Word-Oriented Memory Organization

- **Addresses Specify Byte Locations**
  - Address of first byte in word
 - Addresses of successive words differ by 4 (32-bit) or 8 (64-bit)

32-bit Words	64-bit Words	Bytes	Addr.
Addr = 0000	Addr = 0000		0000
			0001
			0002
			0003
Addr = 0004			0004
			0005
			0006
			0007
Addr = 0008	Addr = 0008		0008
			0009
			0010
			0011
Addr = 0012			0012
			0013
			0014
			0015

Example Data Representations

C Data Type	Typical 32-bit	Typical 64-bit	x86-64
`char`	1	1	1
`short`	2	2	2
`int`	4	4	4
`long`	4	8	8
`float`	4	4	4
`double`	8	8	8
`long double`	–	–	10/16
pointer	4	8	8

Byte Ordering

- **So, how are the bytes within a multi-byte word ordered in memory?**

- **Conventions**

  - Big Endian: Sun, PPC Mac, Internet

    - Least significant byte has highest address

  - Little Endian: x86, ARM processors running Android, iOS, and Windows

    - Least significant byte has lowest address

Little/Big Endian: Jonathan Swift's book "Gulliver's Travels" two kinds of religious groups (one prefer to eat their egg starting from the little end, others from the big end)
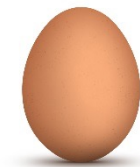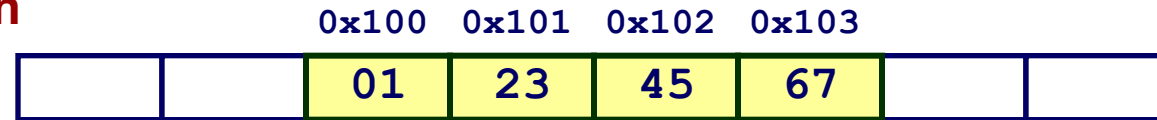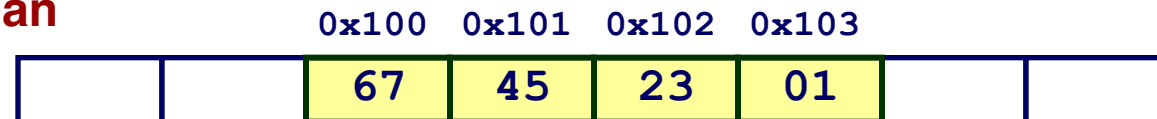
Byte Ordering Example

- **Example**
  - Variable x has 4-byte value of 0x01234567
  - Address given by &x is 0x100

Big Endian

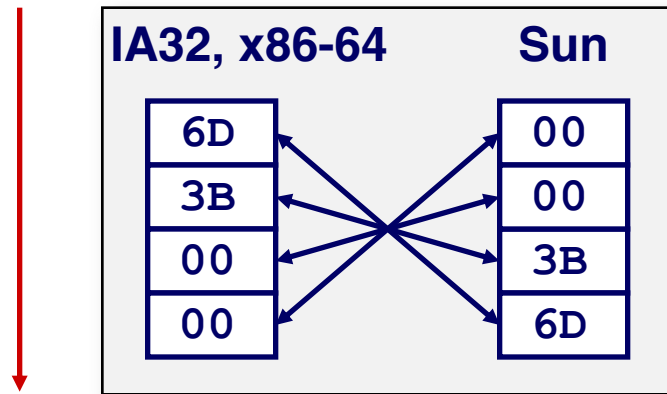|		0x100	0x101	0x102	0x103		
		01	23	45	67		

Little Endian

		0x100	0x101	0x102	0x103		
		67	45	23	01		

Representing Integers

Decimal: 15213

Binary: 0011 1011 0110 1101

Hex: 3   B   6   D

`int A = 15213;`

IA32, x86-64 | Sun
6D → 00
3B → 00
00 → 3B
00 → 6D

`int B = -15213;`

IA32, x86-64 | Sun
93 → FF
C4 → FF
FF → C4
FF → 93

`long int C = 15213;`

IA32 | x86-64 | Sun
6D ↔ 6D → 00
3B ↔ 3B → 00
00 ↔ 00 → 3B
00 ↔ 00 → 6D
         00
         00
         00
         00
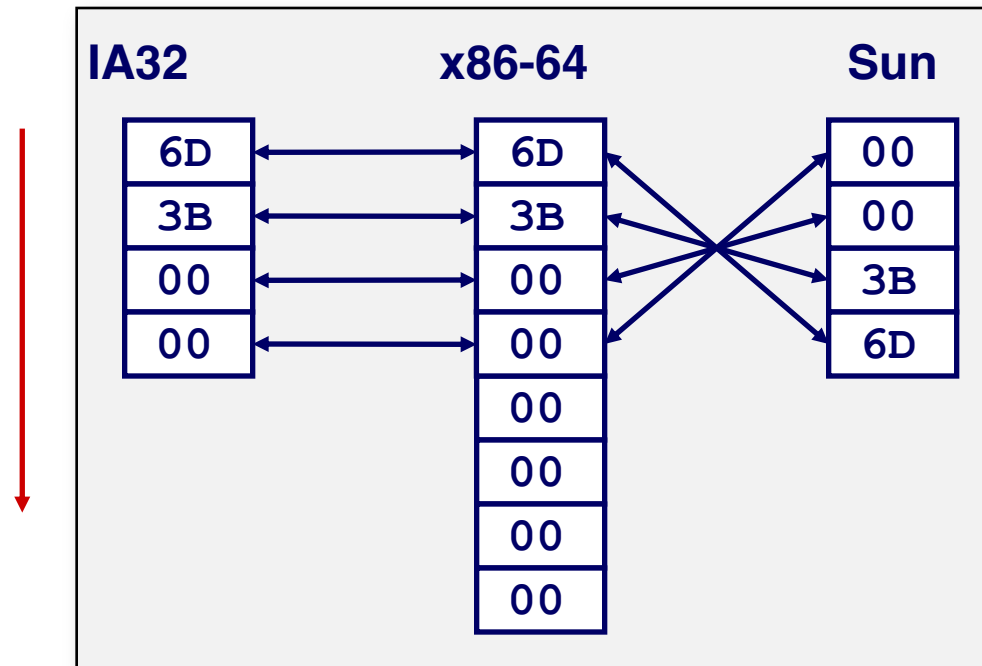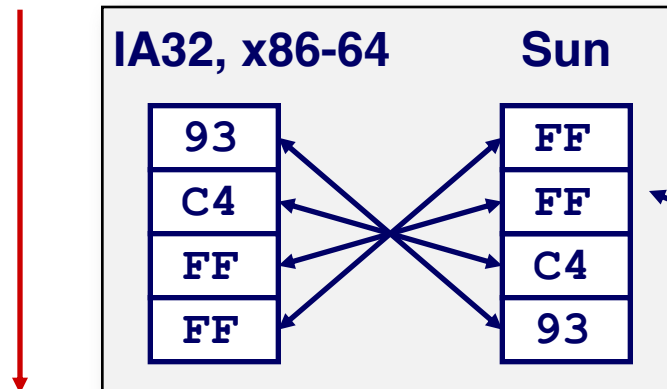
Two's complement representation

Examining Data Representations

- **Code to Print Byte Representation of Data**
  - Casting pointer to unsigned char * allows treatment as a byte array

```c
typedef unsigned char *pointer;

void show_bytes(pointer start, size_t len){
  size_t i;
  for (i = 0; i < len; i++)
    printf("%p\t0x%.2x\n",start+i, start[i]);
  printf("\n");
}
```

Printf directives:
%p: Print pointer
%x: Print Hexadecimal

show_bytes Execution Example

```
int a = 15213;
printf("int a = 15213;\n");
show_bytes((pointer) &a, sizeof(int));
```

Result (Linux x86-64):

```
int a = 15213;
0x7fffb7f71dbc    6d
0x7fffb7f71dbd    3b
0x7fffb7f71dbe    00
0x7fffb7f71dbf    00
```

Representing Pointers

```
int B = -15213;
int *P = &B;
```

	Sun		IA32		x86-64
	EF		AC		3C
	FF		28		1B
	FB		F5		FE
	2C		FF		82
					FD
					7F
					00
					00

Different compilers & machines assign different locations to objects

Even get different results each time run program

Representing Strings
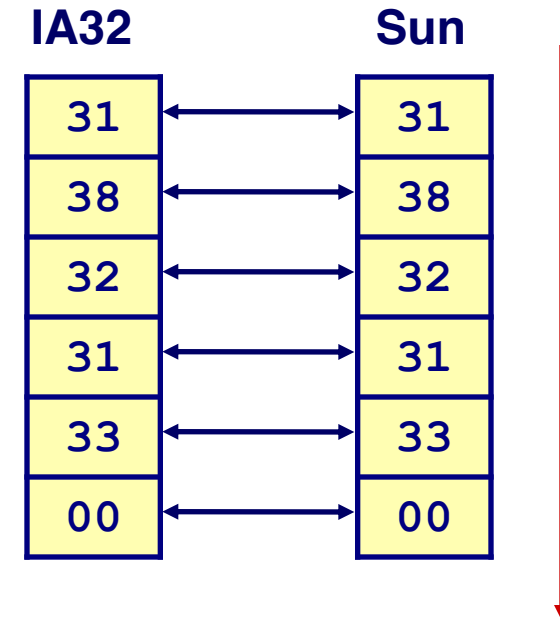
```
char S[6] = "18213";
```

- **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 = 0
- **Compatibility**
  - Byte ordering not an issue

IA32		Sun
31	↔	31
38	↔	38
32	↔	32
31	↔	31
33	↔	33
00	↔	00

Thank you!