CSC 411

Computer Organization (Spring 2024)
Lecture 5: Integers (casting) and Byte ordering

Prof. Marco Alvarez, University of Rhode Island

Casting in C

- Constants
  - considered as signed integers by default, unless the U suffix is included, e.g. `502123U`
- Casting
  - changes the way the data is interpreted, the bit sequence is **maintained**
    - this IS NOT the same as converting a positive value d into its negative $-d$
    - with two's complement, conversions between signed and unsigned basically add or subtract 2^w
  - **explicit casting**
    - requires the specification of the data type — parenthesized cast
  - **implicit casting**
    - occurs automatically in assignments and function calls
    - e.g. assigning an unsigned integer into a signed integer

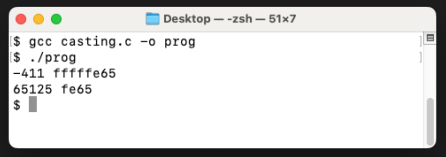
What is the output?

```c
#include <stdio.h>
#include <stdint.h>

int main() {
    // 411 is 0x019B
    int16_t var1 = -411;
    uint16_t var2 = (uint16_t) var1;

    printf("%d %x\n", var1, var1);
    printf("%d %x\n", var2, var2);

    return 0;
}
```



Casting in C

- Expressions (comparisons)
  - if an expression contains signed and unsigned integers, all signed values are **implicitly casted** to **unsigned**

Expression	Type	Evaluation
0 == 0U	unsigned	true
-1 < 0	signed	true
-1 < 0U	unsigned	false
2147483647 > -2147483647 - 1	signed	true
2147483647U > -2147483647 - 1	unsigned	false
2147483647 > (int)2147483648U	signed	true
-1 > -2	signed	true
(unsigned)-1 > -2	unsigned	true

What is the output?

```c
#include <stdio.h>

int main() {
    char a = 254;
    unsigned char b = 254;
    unsigned int c = 0;

    printf("%d %d\n", a, b);

    if (-1 < c) {
        printf("yay\n");
    } else {
        printf("!!!???\n");
    }
}
```

```
Desktop — -zsh — 46×12
$ gcc expr.c -o prog
expr.c:4:14: warning: implicit conversion from
 'int' to 'char' changes value from 254 to -2
[-Wconstant-conversion]
    char a = 254;
           ~   ^~~
1 warning generated.
$ ./prog
-2 254
!!!???
$
```

Casting into "bigger" data types

- Sign extension
  - transform a w-bit integer into an integer with a **larger** bit-width $w + d$, preserving the same value
  - how? just make d copies of the MSB (extension)

−128	64	32	16	8	4	2	1	
				0	1	1	0	6
0	0	0	0	0	1	1	0	6
				1	1	1	0	−2
1	1	1	1	1	1	1	0	−2

Practice

- What is the decimal value of this 64-bit number represented using two's complement?

0xFFFFFFFFFFFFFFF8

11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111000

Casting into "smaller" data types

- Truncation
  - transform a $w + d$-bit integer into an integer with a **smaller** bit-width w, preserving the rightmost w bits
  - how? just drop top d bits, $\mod 2^w$ for unsigned integers

−128	64	32	16	8	4	2	1	
0	1	0	1	0	1	1	0	86
				0	1	1	0	6
1	0	0	1	0	1	1	0	−106
				0	1	1	0	6
1	1	1	1	1	1	1	0	−2
				1	1	1	0	−2
0	0	1	1	1	0	1	0	58
				1	0	1	0	−6

no sign change

sign change

Memory organization

Memory organization

- Memory as a byte array
  - hardware component that stores data and instructions for computer programs
  - memory is a contiguous sequence of bytes, where each byte can be individually accessed using its unique address
- Memory addresses
  - unique numerical identifier assigned to each byte in memory, enabling direct access to its contents
  - a pointer variable stores a memory addresses, providing indirect access to the data stored at that location
- Data Representation
  - different data types (integers, floating-point numbers, characters, etc.) are stored in memory as sequences of bytes, interpreted according to their type
- Operating system provides a private address space to each “process”
  - a process is a program being executed
  - an address space is one of those enormous arrays of bytes
  - each program can see only its own code and data within its enormous array

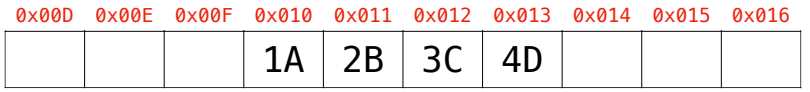
Machine words

- Computers have a “Word Size”
  - usually the size of integer-valued data and of memory addresses
    - word size = 32 provides an address range of $0 \ldots 2^{32} - 1$ ~4GB
      - 4294967295 bytes or 4GB
    - word size = 64 provides an address range of $0 \ldots 2^{64} - 1$ ~16EB
      - 18446744073709551615 bytes or 16EB
- Machines support multiple data formats
  - fractions or multiples of word size

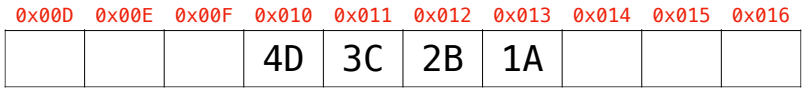
Byte ordering

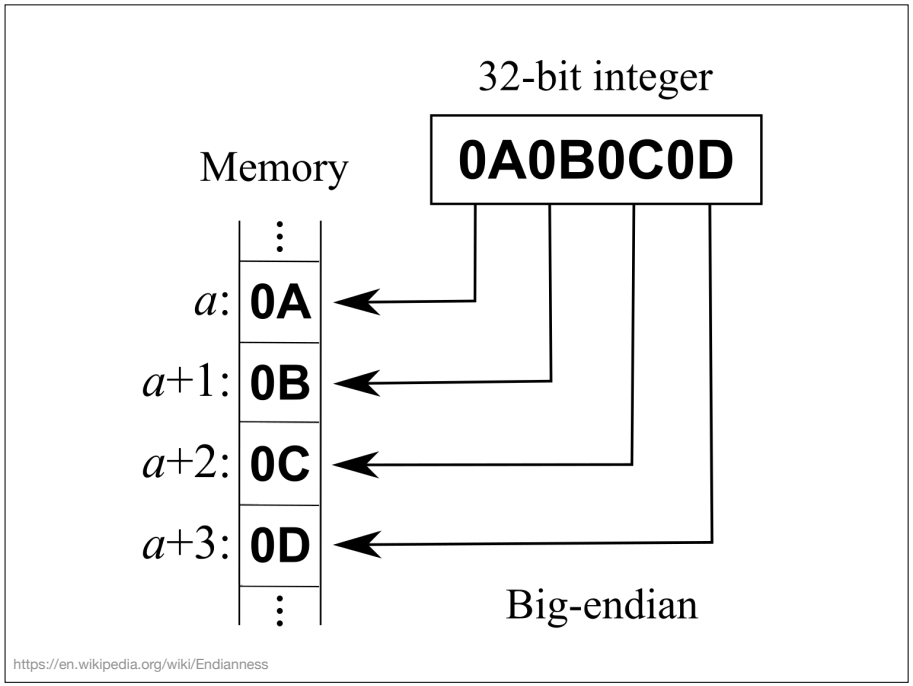
```
x = 0x1A2B3C4D
assume &x is 0x010
```

- Big endian
  - stores the most significant byte at the lowest memory address
  - IBM PowerPC, Motorola 68000, SPARC, network byte order

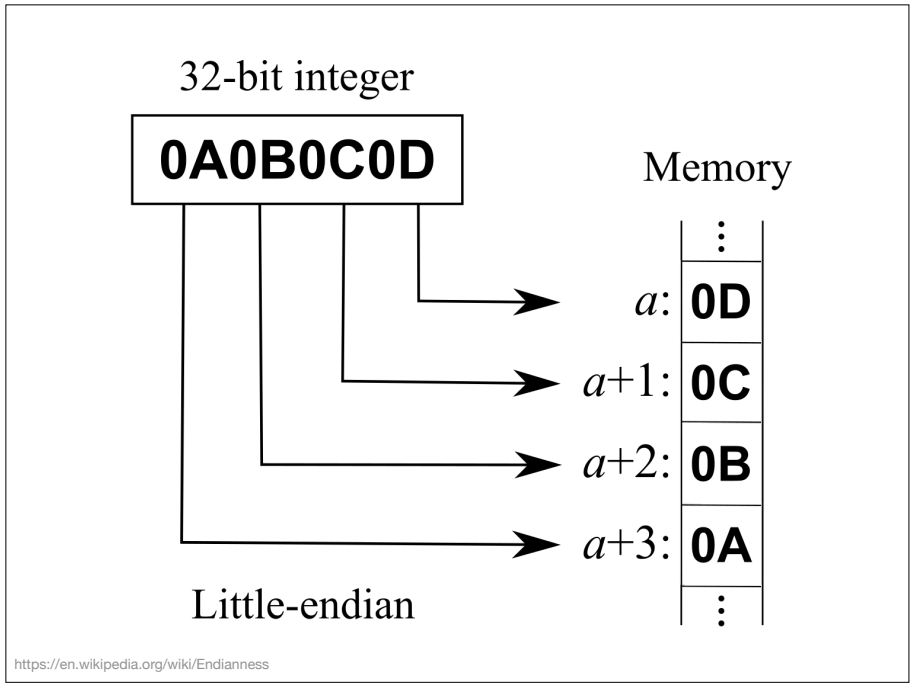


- Little endian
  - stores the least significant byte at the lowest memory address
  - intel x86, ARM, RISC-V, MIPS





https://en.wikipedia.org/wiki/Endianness



https://en.wikipedia.org/wiki/Endianness