CS330 - Computer Organization and Assembly Language Programming

Lecture 1

- Overview -

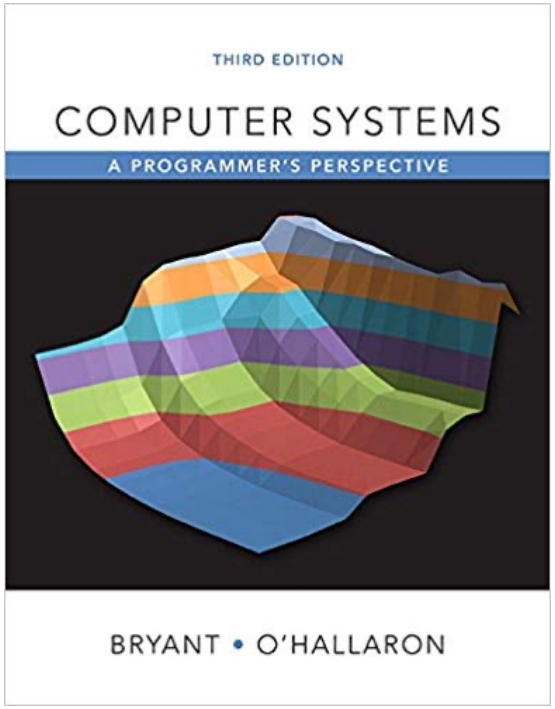
Professor : Mahmut Unan – UAB CS

Agenda

- Syllabus
- Course Overview

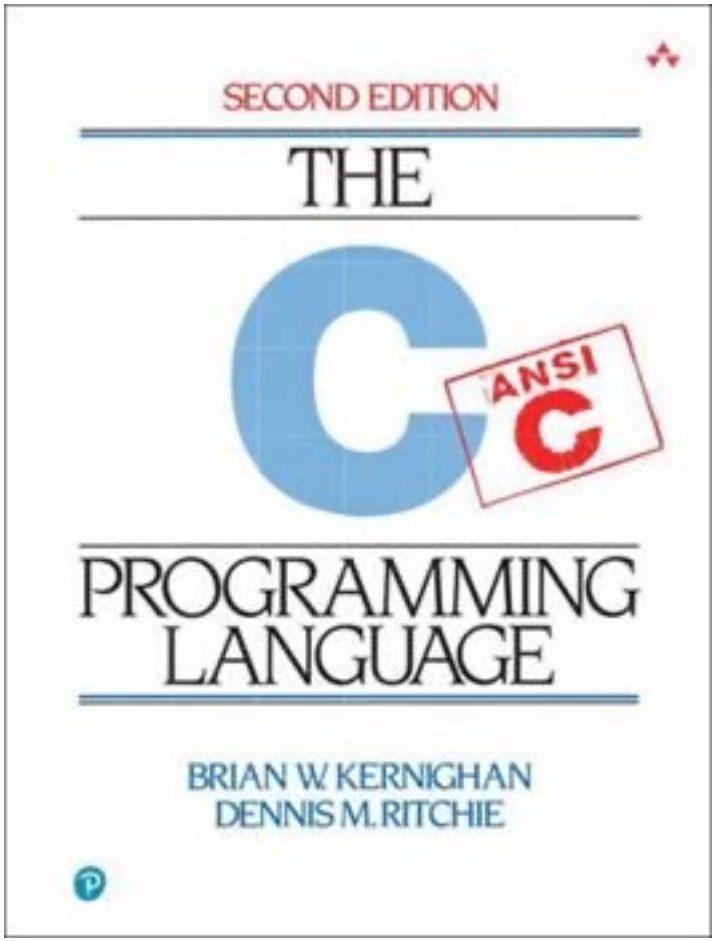
CS 330 - 1B - Computer Organization and Assembly Language Programming

Required Text Book: *Computer Systems: A Programmer's Perspective, 3rd Edition, Randal E. Bryant and David R. O'Hallaron, Carnegie Mellon University*

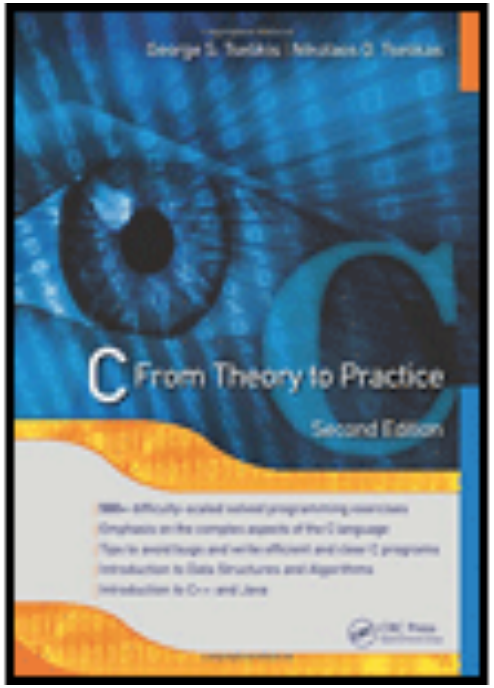


*** Lecture Slides are borrowed from this textbook

Recommended Textbooks



- C Programming Language, 2nd Edition Brian W. Kernighan , Dennis M. Ritchie
- C From Theory to Practice - 2nd edition, Nikolaos D. Tselikas and George S. Tselikis



Hardware or Software ?

- Powerful programmer ?
- For example, consider the following C function to compute the squares of 5, 50, 500, 5000, 50000, 500000, and 5000000:

```
void show_squares()
{
 int x;
 for (x = 5; x <= 5000000; x*=10){
    printf("x = %d x^2 = %d\n", x, x*x);
 }
}
```

- When run on a typical **(32-bit)** processor, this program produces the following values:
- x = 5 x^2 = 25
- x = 50 x^2 = 2500
- x = 500 x^2 = 250000
- x = 5000 x^2 = 25000000
- x = 50000 x^2 = -1794967296
- x = 500000 x^2 = 891896832
- x = 5000000 x^2 = -1004630016

The first four values are exactly what one would expect, but the last three seem quite peculiar. We even see that the “square” of a number can be negative!

Ints are not Integers, Floats are not Reals

- **Example 1: is $x^2 \geq 0$?**
  - Float's: Yes!
  - Int's:
    - 40000 * 40000 = 1600000000
    - 50000 * 50000 = ??
- **Example 2: is (x + y) + z = x + (y + z)?**
  - Unsigned & Signed Int's: Yes!
  - Float's:
    - (1e20 + -1e20) + 3.14 --> 3.14
    - 1e20 + (-1e20 + 3.14) --> ??

Should I care about it?

- https://www.youtube.com/watch?v=PK_yguLapgA&t=4s

- The cause of the failure was a software error in the inertial reference system. Specifically a 64-bit floating-point number relating to the horizontal velocity of the rocket with respect to the platform was converted to a 16 bit signed integer.

- The number was larger than 32,767, the largest integer storeable in a 16 bit signed integer, and thus the conversion failed.

Memory Matters

- **Random Access Memory Is an Unphysical Abstraction**
- **Memory is not unbounded**
  - It must be allocated and managed
  - Many applications are memory dominated
- **Memory referencing bugs especially pernicious**
  - Effects are distant in both time and space
- **Memory performance is not uniform**
  - Cache and virtual memory effects can greatly affect program performance
  - Adapting program to characteristics of memory system can lead to major speed improvements

Memory Referencing Errors

- **C and C++ do not provide any memory protection**
  - Out of bounds array references
  - Invalid pointer values
  - Abuses of malloc/free
- **Can lead to nasty bugs**
  - Whether or not bug has any effect depends on system and compiler
  - Action at a distance
    - Corrupted object logically unrelated to one being accessed
    - Effect of bug may be first observed long after it is generated
- **How can I deal with this?**
  - Program in Java, Ruby, Python, ML, …
  - Understand what possible interactions may occur
  - Use or develop tools to detect referencing errors (e.g. Valgrind)

Memory System Performance Example

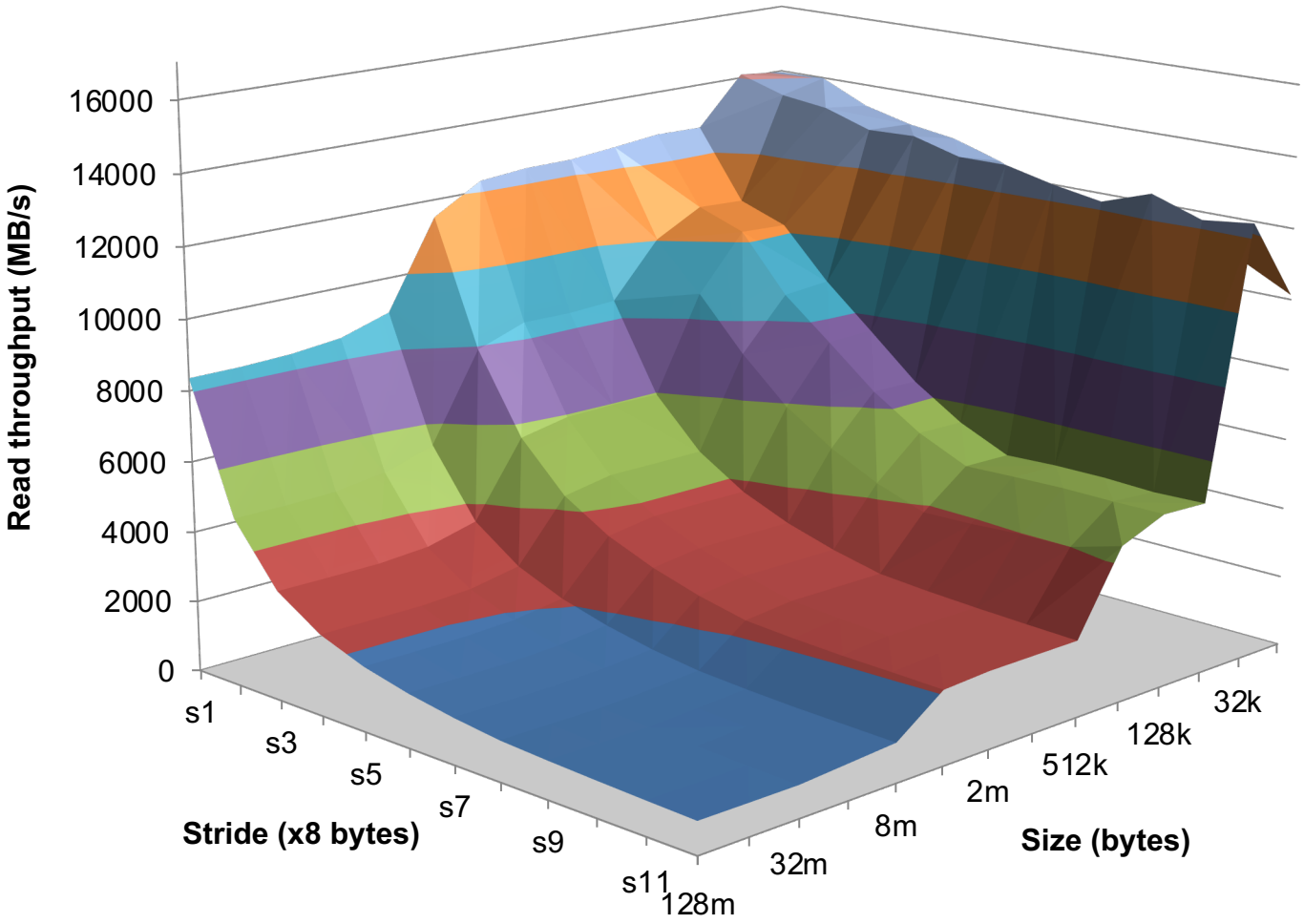
```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

4.3ms 2.0 GHz Intel Core i7 Haswell 81.8ms

- Hierarchical memory organization
- Performance depends on access patterns
  - Including how step through multi-dimensional array

Why The Performance Differs



There's more to performance than asymptotic complexity

- **Constant factors matter too! (**How about CS303 ☺**)**
- **And even exact op count does not predict performance**
  - Easily see 10:1 performance range depending on how code written
  - Must optimize at multiple levels: algorithm, data representations, procedures, and loops
- **Must understand system to optimize performance**
  - How programs compiled and executed
  - How to measure program performance and identify bottlenecks
  - How to improve performance without destroying code modularity and generality

Computers do more than execute programs

- **They need to get data in and out**
  - I/O system critical to program reliability and performance

- **They communicate with each other over networks**
  - Many system-level issues arise in presence of network
    - Concurrent operations by autonomous processes
    - Coping with unreliable media
    - Cross platform compatibility
    - Complex performance issues

Why to learn C Programming?

- Powerful
- C is the most commonly used programming language for writing operating systems.
- Coding close to wire
- Know register, stack, heap…
- Simple, elegant, wicked **fast**…
- C Family of Languages…
- Embedded programming
- ……
- Learn more:

https://www.pluralsight.com/blog/software-development/why-every-programmer-should-learn-c

https://en.wikibooks.org/wiki/C_Programming/Why_learn_C%3F

https://www.topcoder.com/blog/5-reasons-keep-learning-c/

C Programming Tutorials

- https://www.learn-c.org/
- https://www.cprogramming.com/tutorial/c-tutorial.html
- Udemy
  - C Programming Tutorial – Complete Tutorial for beginners
  - The Complete C Programming Tutorial
- Lynda
  - Learning C
  - Advanced C Programming
- https://www.tutorialspoint.com/cprogramming/
- https://www.geeksforgeeks.org/c-programming-language/
- http://www.zentut.com/c-tutorial/
- Youtube
  - https://www.youtube.com/watch?v=2NWeucMKrLI&list=PL6gx4Cwl9DGAKIXv8Yr6nhGJ9Vlcjyymq

Why to learn Assembly ?

- Low-level programming language
- Manipulate hardware directly
- Understand how things really works
- Performance
- Real-time systems
- Transparent

Lab 1 and Lab 2

- Lab 1
  - Install environment, start working on C
- Lab 2
  - C coding exercises

Next Class

We will learn the major ideas and themes in computer systems by tracing the life cycle of a simple “hello world” program.

- Please read **Chapter 1**