Course Overview & Policies

Computer Systems II: CIT 595
Spring 2010

Staff and Website
Instructor
- Diana Palsetia (palsetia@seas.upenn.edu)
  - Office Location: Moore 174
  - Office hours: TBA

Teaching Assistant
- Paramveer Dhillon
  - Email: pasingh@seas.upenn.edu
  - Location & Office hours: TBA

Course Website:
www.seas.upenn.edu/~cit595

Class Times

- Lecture
  - Days: Monday and Wednesday
  - Time: 1:30 – 3pm
  - Location: Towne 309

- Recitation
  - Day: Friday
  - Time: 3-4pm
  - Location: Towne 309
  - Held for some lectures, quizzes or any computer-lab activities (lab activities will be held in Moore 100A)

Resources

- No required text book
  - Reading handouts will be provided

- Reference Books (on library reserve)
  - Computer Organization and Design (Hardware/Software Interface) by David Patterson and John Hennessy
  - Computer Systems A Programmer's Perspective by Randal E. Bryant and David R. O'Hallaron
  - Modern Operating Systems - 3rd Edition by Andrew Tanenbaum
  - C++: The Complete Reference, 4th Edition

- Online Resources
  - See Resources link on course website
Grading
- Policy for receiving an A
  - 90% or above is not necessarily an A as this depends on the class performance
- Assignments (60% of grade)
  - Programming assignments C/C++
  - Non-programming assignments should be preferably be typed
- Quizzes (40% of grade)
  - 3 quizzes spaced out over the semester
  - They will be held on Friday recitation time
    - 2/26, 3/26, 4/23
  - Quizzes are closed books and notes

Assignment Grading
- 5 total grace days for entire semester
  - Can use only at max. 3 on one assignment
  - After the grace days 20% per day deduction
- Turning assignments
  - Submitted via Digital Dropbox via Blackboard
  - Instructions on how to submit are provided in the syllabus (see course website)

Academic Honesty
- Appropriate
  - Discuss the assignments with one another to clear doubts
  - Help others debug their work
- Inappropriate
  - Work together unless stated otherwise
  - Copy another student’s work
  - Lend your work to someone else, or allow them to copy it
  - Use any material from a book or the web without my permission
- Penalty for first offence:
  - You will be reported to the Office of Student Conduct
  - If you may have accidentally broken a rule, talk to Instructor or TA immediately

CIT595: Computer Systems II
- Focuses on the design and implementation of modern computer systems
- Projects will include significant programming in C
- Introduction to C++ including inheritance, operator overloading and Templates and STL
Prerequisite

- CIT593 or equivalent
  - Data representation (binary system, signed/unsigned integer arithmetic, ASCII)
  - Von Neumann Model, ISA and knowledge of instruction formats, addressing modes, data types
  - Stack/Heap Memory
  - Understanding of system calls
  - Concept of I/O – polling vs. interrupt
  - Any assembly language
  - C language – functions, control structures, file I/O, arrays, pointers, malloc, structs

System Software

- Operating System
  - Process scheduling
    - Multiprogramming: Concept of processes, and threads
    - Synchronization of shared data
    - Communication among heavy/light processes
  - Resource management (Deadlock handling)
  - File System management
  - Protection/Isolation: Virtual Memory

- Other System Software
  - Linking: Object files, static and dynamic linking, libraries, loading in Unix/Linux

C++ Background

- Initially C++ was built on top of C
  - Set of macros and routines for C

- 1979-1983: Bjarne Stroustrup worked on “C with classes”
  - More features (e.g. Templates) came later on

- New C++ is intended to be backward compatible with C

C++ Highlights

- Can pass data by reference
  ```c
  void duplicate (int& a, int& b, int& c){
    a*=2;
    b*=2;
    c*=2;
  }
  ```

  In main:
  ```c
  int x=1, y=3, z=7;
  duplicate (x, y, z);
  //after call x=2, y=6, z=14
  ```
Highlights of C++: Object Oriented (Class Structure)

- Object Oriented
  - String data type vs. char *
  - Unlike Java, Filename does not have to same as classname

```cpp
class Example {
private:
    int y;    //data
public:
    Example(int x) { y=x; }  //Constructor
    int getY() {return y;}    //getter
    void setY(int x) {y = x;} //setter
};
```

Highlights of C++: Object Oriented (Dynamic Memory Allocation)

- Instead of malloc/free, use new/delete
  ```cpp
  Example * ex = new Example(7);
  ....
  delete ex;
  ```
- Also new[]/delete[] for arrays:
  ```cpp
  int * ptr = new int[3];
  ....
  delete[] ptr;
  ```
- Don’t mix and match!

Highlights of C++: Object Oriented (Inheritance)

- Unlike Java, static dispatch is default for methods
- Use virtual keyword to specify dynamic dispatch:
  ```cpp
  virtual int getY(){return y;}
  ```
- Rationale: avoid cost of dynamic dispatch unless programmer wants it
- Good idea? Depends on who you ask....

Highlights of C++: Templates

- Function templates are special functions that can operate with generic types
- Adapt to more than one type or class without repeating the entire code for each type
- E.g. a template function that returns the greater one of two objects
  ```cpp
template <class myType>
myType getMax (myType a, myType b) {
    return (a > b ? a : b);
}
```
- Standard Template Library (STL)
  - Supports data structures such as vectors, lists, queues, sets
  - Class templates allows flexibility in the types supported as elements
Others C++

- Boolean data type: bool
- Function and Operator Overloading
- Supports Exceptions - try catch clauses
- Namespaces - Allow to group entities like classes, objects and functions under a name
  - This way the global scope can be divided in "sub-scopes", each one with its own name
  - In order to access these variables from outside the particular namespace we have to use the scope operator ::
    - E.g. namespace first { int var = 5; }
    - first :: var

Memory Matters

- Memory referencing bugs are especially detrimental
  - Common bugs: buffer overflow, bad references, bad pointer arithmetic, unaligned access, endianness
- Memory is not unbounded
  - It must be allocated and managed. By whom?
- Memory vs. Processor technology speed mismatch affects performance
  - Meet the gap by introducing hierarchy (fast-small vs. slow-large)
  - Cache and Virtual memory effects can greatly affect program performance
    - Adapting program to characteristics of memory system can lead to major performance improvements

E.g. Adapting program to characteristics of memory

- Data stored in cache is data that the processor is likely to use in the very near future
  - E.g. Accessing array, access pattern is sequential & if element at i is requested then element at i+1, i+2 are also brought into the cache
- Suppose Matrix (2D-array) data kept in memory is by rows (known as row-major) i.e. offset = row*NUMCOLS + column
  - Then following is poor code:
    - for (j = 0; j < numcols; j++)
      - for(i = 0; i < numrows; i++)
        - printf("%d", x[i][j]);
  - i.e. access pattern x[0][0], x[1][0], x[2][0] ...
  - The array is being accessed by column and we may not find the access pattern in the cache (depends on the cache size)
  - Solution: switch the for loops
  - C/C++ are row-major, FORTRAN & MATLAB is column-major

Performance

- What defines performance? How do we measure performance?
- CPU execution dependent on
  - Number of instructions executed
  - Clock cycles it takes per instruction (CPI)
  - Clock rate
- Number of Instructions depends on:
  - Algorithm (data representations, procedures, and loops)
  - Compiler
  - ISA
E.g. Loop Inefficiency

- Procedure to Convert String to Lower Case

```c
void lower1(char *s)
{
    int i;
    for (i = 0; i < strlen(s); i++)
        if (s[i] >= 'A' && s[i] <= 'Z')
            s[i] -= ('A' - 'a');
}
```

Performance Results of lower1

- Time quadruples when string length is doubled

<table>
<thead>
<tr>
<th>String Length</th>
<th>CPU Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>256</td>
<td>0.0001</td>
</tr>
<tr>
<td>512</td>
<td>0.001</td>
</tr>
<tr>
<td>1024</td>
<td>0.01</td>
</tr>
<tr>
<td>2048</td>
<td>0.1</td>
</tr>
<tr>
<td>4096</td>
<td>1</td>
</tr>
<tr>
<td>8192</td>
<td>10</td>
</tr>
<tr>
<td>16384</td>
<td>100</td>
</tr>
<tr>
<td>32768</td>
<td>1000</td>
</tr>
<tr>
<td>65536</td>
<td>10000</td>
</tr>
<tr>
<td>131072</td>
<td>100000</td>
</tr>
<tr>
<td>262144</td>
<td>1000000</td>
</tr>
</tbody>
</table>

- `strlen` executed every iteration
- `strlen` linear in length of string
  - Must scan string until finds '\0'

Performance contd..

- CPI depends on
  - Instruction Set Architecture (ISA)
  - Design details (micro-architecture)
    - CPI varies among different implementations of the same ISA
- Clock rate
  - Hardware Technology (transistor technology)
  - Design (Micro-architecture)
    - Simulate the longest propagation path of your circuit

Computer Architecture vs. Organization

**Computer Architecture**

- Logical aspects of system implementation as seen by the programmer
- Such as instruction sets and formats, opcode, data types, addressing modes
  - E.g. LC3/4 ISA studied in CIT 593
- Answers the question: *How do I design a computer?*
  - E.g. Do I want 1, 8, or 32 register?

**Computer Organization**

- Encompasses all physical aspects of computer systems
  - Circuit design, control signals, memory types
- Answers the question: *How does a computer work?*
Hardware Technology and Micro-architecture

- Electronic "switches"
  - Transistor - basis of Logic gates (NOT/OR/AND)
  - Computer components such as ALU, memory are built from logic gates
- Clock
  - Is used to synchronize events in the computer system
  - Events are synchronized on the clock tick or pulse or cycle
  - Clock Speed/Rate = 1/Clock Cycle

Clock Rate

- E.g. Pentium 4 2.0 GHz
  - Refers to clock speed/rate of the Pentium 4
- Hz = Hertz = cycles per second (frequency)
  - E.g. 1GHz = 10^9Hz (Giga = 10^9)
- Clock Speed is crucial for computer performance
  - Assumption: 2 billion instructions can be executed in one second OR 1 instruction takes 0.5 x 10^-9 s or 0.5ns
  - Reality: # instruction executed in a second is only proportional to the speed

Is higher clock rate always a good indicator for performance?
  - Machine A is 900 Mhz and B is 1Ghz – which is faster?

Improving Hardware-level Performance

- Various methods of increasing performance at hardware level
  - Instruction-level parallelism (ILP)
    - Pipelining and superscalar methods like superscalar
  - Thread level parallelism (TLP) methods
  - Multiple independent CPUs (Multiprocessor)

- But there is no free lunch!
  - Adds to design complexity and power consumption
  - Parts of the applications may not be parallelized due to dependencies
    - Amdahl's law: the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used.

Questions