CSE220
System Fundamentals I

Prof. Jennifer Wong-Ma
jwong@cs.stonybrook.edu
Office: New CS 212
Phone: 631-632-1728

Office Hours:
- Weds 10am-11am - COURSE-ONLY
- Weds 11am-12pm, Thurs 1-3pm OPEN HOURS
Lectures: Mon & Weds 8:30-9:50am in Javits 102

Course Homepage: [http://www.cs.stonybrook.edu/~cse220](http://www.cs.stonybrook.edu/~cse220)
Course Requirements

**Prerequisite(s):** CSE major, and completed CSE 114 or 160 with a C or higher.

**Note(s):** A minimum grade of C is required in this course for the CSE major or minor.

Grading Breakdown

Pass/No Credit (P/NC) option is not available for this course.

- **Programming Assignments:** 25%
- **Quizzes:** 15%
- **Midterm Exam:** 25%
- **Final Exam:** 30%
- **Recitation Attendance:** 5%
Course Policies: Programming Assignments

- **NO Extensions or late assignments**
  - Emergency? Send email with SUITABLE documentation

- **Plan ahead!!!!**
  - Bugs take TIME!
  - Internet isn’t perfect!
  - Computers are machines, they fail/crash/etc.

- If your assignment is incomplete or is not fully working by the due date, turn in whatever you have.
- **Does NOT compile or run??** Grade = 0.
Course Policies: Quizzes

- ~ 6 quizzes during the semester
- Closed book, closed notes, no calculator
- No electronic devices
- In Lecture on material in lecture, textbook and recitation

- NO MAKEUP QUizzes!

- Grading Issues? 1 week from receipt of graded quiz to email Professor
  - Must identify Problem # and reason for request in email
Course Policies: Examinations

- Closed Book, closed note, no calculator
- No electronic devices

**Common Midterm Exam:** Monday, October 16, 2017 8 - 10:15 pm

**Common Final Exam:** Monday, December 18, 2017 8 - 10:45 am

- **No Makeup Exams!!** Except for PRIOR documented excused absences or documented emergencies
Course Policies: Attendance

- **Recitation Attendance is MANDATORY.** 5% of your grade!
  - Can miss 1 recitation without penalty

- You are **responsible for all missed work**, regardless of the reason for absence. It is also your responsibility to get/review all missing notes or materials.
  - They are posted on the course webpage!
Course Policies: Lectures & Recitation

- **Laptops:** permitted, but
  - Please be aware of their distracting potential
  - Please do not: IM, watch movies (!), browse the web, …
  - Failing to pay attention is not good grounds to ask me to repeat material

- **Electronic communications:** *forbidden*
  - No email, instant messaging, cell phone calls, etc

- Presence in lectures: voluntary, but highly recommended

- No recordings of ANY KIND without MY explicit permission
Course Policies: PIAZZA & Etiquette

- Learning environment!

- Any and all non-personal course-related communications!
  - General Programming Questions
  - General Homework Questions
  - Practice Problem Questions
  - Technical Problems
  - Lecture Questions

- Code??
  - General snippets or course examples only!
Course Policies: PIAZZA & Etiquette


- No TOLERANCE for:
  - Cyber-bullying
  - Memes
  - Grade complaints
  - Course concerns
  - Course comments/criticisms
Course Policies: Email & Etiquette

- All **PERSONAL** course related communication should be sent by email.

- Your Professors are all very busy!

Help them respond quicker to your emails!

- use your **official @stonybrook.edu** email account
- use a **descriptive subject line** that includes “CSE 220” and a brief note on the topic (eg. “CSE220: Appointment”)
- begin with a **proper greeting**, such as “Hi Prof. McDonnell” or “Hi Prof. Wong”
- briefly explain your question or concern or request
- end with a proper salutation that includes your **full name, netid, and SBU ID number**
Course Policies: Academic Dishonesty

- Students are expected to work independently and complete their own work!

- Offering and accepting solutions from others is an act of PLAGIARISM!
- Copying code from a friend or the INTERNET is PLAGIARISM!
- Paying others to do your HW is PLAGIARISM!

When in doubt, ASK your Prof!
Course Policies: Weather & Critical Incidents

Snow, it happens.....Check PIAZZA & Univ Website for class cancellations.

We do our best to keep it FUN! But, disruptions in Lecture will not be tolerated.

“Stony Brook University expects students to respect the rights, privileges, and property of other people. Faculty are required to report to the Office of Judicial Affairs any disruptive behavior that interrupts their ability to teach, compromises the safety of the learning environment, or inhibits students ability to learn. Faculty in the HSC Schools and the School of Medicine are required to follow their school-specific procedures.”
Course Policies: Disabilities

If you have a physical, psychological, medical or learning disability that may impact on your ability to carry out assigned course work, I would urge that you contact the staff in the Disability Support Services office (DSS), ECC Building (behind SAC), 632-6748/TDD. DSS will review your concerns and determine, with you, what accommodations are necessary and appropriate. All information and documentation of disability is confidential. Students who require assistance during emergency evacuation are encouraged to discuss their needs with their professors and Disability Support Services. For procedures and information go to their and search Fire Safety and Evacuation and Disabilities.
# Course Outline

<table>
<thead>
<tr>
<th>Week</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week 1</td>
<td>• Introduction to Digital Systems and von Neumann Architecture</td>
</tr>
<tr>
<td></td>
<td>• Number Systems: SM, 1’s complement, 2’s complement, floating-point</td>
</tr>
<tr>
<td>Week 2</td>
<td>• MIPS Assembly: Basic Instructions, System Calls, Endianness, Machine Language</td>
</tr>
<tr>
<td>Week 3</td>
<td>• MIPS Assembly: Branches and Loops, Function Calls, Stack, 1-D Arrays</td>
</tr>
<tr>
<td>Week 4</td>
<td>• Digital Logic Design: Logic Gates, Boolean Equations and Algebra</td>
</tr>
<tr>
<td>Week 5</td>
<td>• Digital Logic Design: Multilevel Combinational Logic, K-maps</td>
</tr>
<tr>
<td>Week 6</td>
<td>• Digital Logic Design: Combinational Building Blocks and Timing</td>
</tr>
<tr>
<td>Week 7</td>
<td>• Digital Logic Design: Arithmetic Circuits</td>
</tr>
<tr>
<td>Week 8</td>
<td>• Digital Logic Design: Arithmetic Circuits</td>
</tr>
<tr>
<td></td>
<td>• MIPS Assembly: Memory Alignment, Multi-dimensional Arrays</td>
</tr>
<tr>
<td>Week 9</td>
<td>• MIPS Architecture: Single-Cycle Processors</td>
</tr>
<tr>
<td>Week 10</td>
<td>• MIPS Architecture: Single-Cycle Processors</td>
</tr>
<tr>
<td>Week 11</td>
<td>• MIPS Assembly: Recursive Functions</td>
</tr>
<tr>
<td></td>
<td>• MIPS Architecture: Multicycle Processors</td>
</tr>
<tr>
<td>Week 12</td>
<td>• MIPS Architecture: Multicycle Processors</td>
</tr>
<tr>
<td>Week 13</td>
<td>• MIPS Architecture: Pipelined Processors</td>
</tr>
<tr>
<td>Week 14</td>
<td>• MIPS Architecture: Pipelined Processors</td>
</tr>
<tr>
<td></td>
<td>• System Performance</td>
</tr>
</tbody>
</table>

Computers Today

- Current computing landscape contains three main types of devices
  1. Personal computers – general purpose, low computation machines. Mainly focused on input-output (I/O) for user.
  2. Servers and Supercomputers – Built for more specific purposes: storage, computation (IE. Amazon Cloud, IBM BlueGene)
  3. Embedded Computers – special purpose computers which are embedded into a larger system. (IE. Smart TV, DVR, anti-lock brakes in car, network router)
- There are hard to categorize devices, like smart phones and tablets – These tend to fall between personal computer and embedded system depending on the device.

Computer Architecture vs. Computer Organization

- **Computer Architecture** refers to those aspects of the hardware that are visible to the “programmer” e.g., instructions the computer is capable of executing, word size (native unit of data of CPU), data formats
- **Computer Organization** (also called microarchitecture) refers to how the physical components of the machine interact to implement the architecture
- A particular computer architecture could be implemented by several different organizations.

8 Great Ideas in Architecture

1. **Moore’s Law**
   - Circuit complexity/speed doubles every 18-24 months
   - We have reached the limits of Moore’s Law. To “keep up” manufactures have found other mechanisms – ie. multi-core
2. Use abstraction to simplify design
   - Abstraction refers to ignoring irrelevant details and focusing on higher-level design/implementation issues
   - Part of software development too
3. Make the common case fast
   - Enhance the performance of those operations that occur most frequently
4. Increase performance via parallelism
   - Perform operations in parallel (simultaneously) when possible
5. Increase performance via pipelining
   - A form of parallelism in which single instructions are broken into multiple stages and executed in parallel
6. Increase performance via prediction
   - The computer will “guess” which operation will be executed next and start executing it
7. Implement a hierarchy of memories
Fastest, smallest and expensive memory at the top; slowest, largest and cheapest at the bottom

8. Increase dependability via redundancy
   ▪ Include redundant components that can take over when a failure occurs
   ▪ Of particular importance in cloud computing systems and other server technologies

**von Neumann Architecture**

- a particular computer hardware design model for a stored-program digital computer (e.g., PCs)
- Named for Hungarian-American mathematician John von Neumann, but others participated in the original design
- Separate central processing unit (CPU) and random-access memory (RAM)
- Both instructions and data stored in RAM
- Data to be processed is transferred from RAM to CPU, and results are transferred back to RAM

**Stored-Program Computer**

- The program to be executed is stored in RAM along with the data to be processed
• A program consists of binary instructions stored in RAM
• Each instruction or small piece of data in RAM has an associated memory address to indicate its location
• A program counter (or instruction pointer) register in the CPU stores the memory address of the next instruction to be executed

Fetch/Decode/Execute Cycle
• The basic cycle of operation of a von Neumann-style computer:
  o Fetch: the next instruction is retrieved from RAM
  o Decode: the instruction is examined to determine what the CPU should do:
    ▪ Opcode: field that determines the type of instruction
    ▪ Operand(s): fields that determine the source and destination of data to be operated on
  o Execute: the operation specified by the instruction is performed, which may involve one or more memory references

Instruction Set
• The instruction set of a computer is the repertoire of instructions that the CPU can perform
  o Determined by the computer architects/designers
  o “Hard-wired” as part of the computer design
  o Different for each type of CPU

RISC vs CISC
• In RISC (reduced instruction set computer) CPUs, the instruction set is small and consists of simple instructions
• In CISC (complex instruction set computer) CPUs, the instruction set is larger and consists of instructions that vary in length and complexity
• Historically, machines were RISC machines. Instructions were simple, therefore the hardware design as simple. At the time, RAM was very limited and it would take many instructions to execute tasks. As time went on designers added more instructions, which would do more complex tasks to reduce the amount of RAM required to store the program. This resulted in increased complexity in the instructions and the hardware required to implement these instructions.
• Modern processors are a hybrid of these two categories. However, modern embedded devices are still or have reverted to RISC style.

Typical RISC Instructions
• **Load** data from memory to CPU register
• Copy (“**Store**”) data from CPU register to memory
• **Add, subtract, multiply, divide**, etc. data in CPU registers
• **AND, OR, XOR, NOT**, etc. data in CPU registers
• **Shift** and **rotate** data in CPU registers
• **Jump** based on CPU state flags (“condition codes”)
• **Call** a subroutine (function) and **return** to caller

• A **high-level programming language** statement like
  \( X = A + B \) would be translated to assembly language instructions as follows:
  • **LOAD** A
  • **LOAD** B
  • **ADD** C, A, B
  • **STORE** C, X
• Real assembly instructions are a little more sophisticated than this, but the beauty of RISC machines is that the instructions are quite simple
• The assembly language instructions are then translated into machine language (1s and 0s) for execution by the CPU