EE 213 Computer Organization and Assembly Language

Week # 1, Lecture # 1

12th Dhu'l-Hijjah, 1439 A.H

27th August 2018

These slides contains materials taken from various sources. I fully acknowledge all copyrights.

Today’s Topics

- COAL course mechanics (Theory + Lab)

Course Objective (1)

This course allows students to understand how high-level code is executed by the processor (a complex digital circuit) by explaining organization of digital architecture (micro-architecture) and its working and usage (through assembly language).

It also highlights important issues when the software-hardware boundary is crossed, specially discussing the implications related to performance and compatibility for language, compilers, and operating systems.

Course Objective (2)

Students will be able to explain the following concepts with proficiency:

- Micro-architecture of processors (computer organization)
  - Introducing basics of data path and control path including related concepts. Coverage of CISC and RISC architectures.
  - Overview of micro-architecture of Intel x86 family and MIPS family of processors.
- Assembly Language of x86 and MIPS.
  - Direct coding in assembly, converting assembly to high-level language and visa versa. Debugging, using latest IDE and simulator for assembly coding.
- Foundational concepts relating to OS and CA courses:
  - ISA. Instruction encoding (generating machine code for each assembly instruction)
  - How to access computer hardware directly. I/O Ports, Interrupts, Device handlers.
  - How HLL and Assembly programs interact with the operating system for various services
  - APIs including memory management and input/output services.
  - How is it possible to interface high-level language and low-level language modules?
  - Polling and Interrupts, basic concepts of ISRs, etc.

Topics (To be covered in lectures & labs)

- Programs are translated by other programs into different forms
- Understand how compilation systems works
- Processors read and interpret instructions stored in memory
- Caches matters
- Storage devices form a hierarchy
- The operating systems manages the hardware
- System communicate with other systems using networks
- Representing and manipulating information: information storage, integer representations, integer arithmetic, floating point
- Machine-level representation of programs: a historical perspective, programs control, procedures, array allocation and access, heterogeneous data structures
- Stack frames and Recursion
- Putting it together: understanding pointers
- Life in real-world: using gdb debugger, out-of-bound memory references and buffer overflows
- X86-64: extending IA32 to 64 bits, machine-level representations of floating point programs
- Processor micro-architecture: The X86 instruction set and implementation, MIPS 2000 instructions set and implementation, CISC vs RISC
- Logic design and Hardware control language (AHDL)
- General principles of pipelining and pipelined implementation implementation of instruction set.
- Polling and Interrupts
- Machine Language Translation
- MIPS simulator for MIPS assembly
- X86 assembly and assembler
- Using assembly within C/C++ program

How course topics are delivered?

Coverage up to Mid # 1

- Fundamental concepts
- Von Neumann architecture
- Caches basics
- Micro-Architecture
- x86 Micro-architecture
- x86 Assembly programming

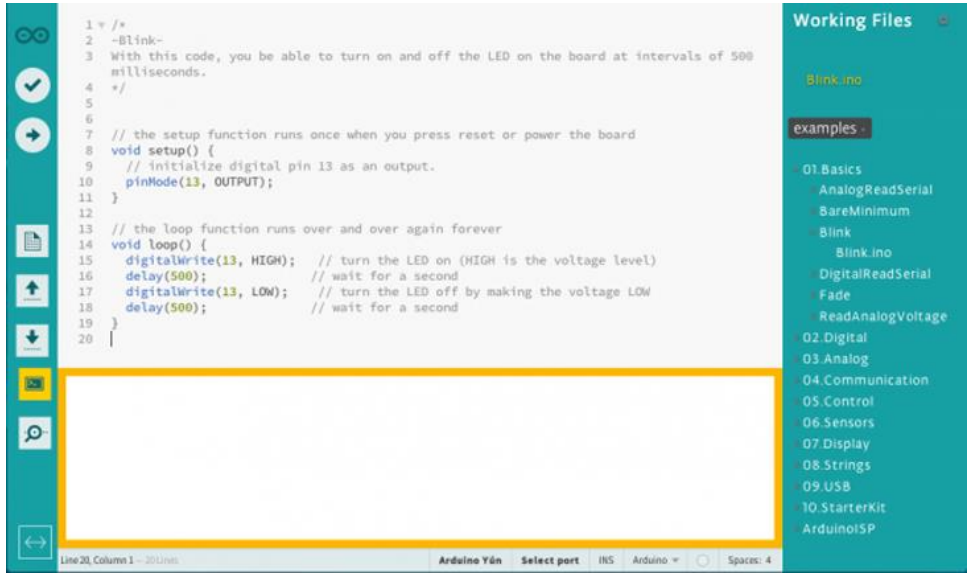
Coverage up to Mid # 2

- x86 Assembly programming (Contd.)
- CISC vs RISC
- MIPS micro-architecture
- MIPS Assembly programming

Coverage up to Final Exams

- Logic design and Hardware control language (AHDL)
- Polling and Interrupts
- Machine Language Translation
- Using assembly within C/C++ program
- Stack frames and Recursion (for x86)

Semester Projects



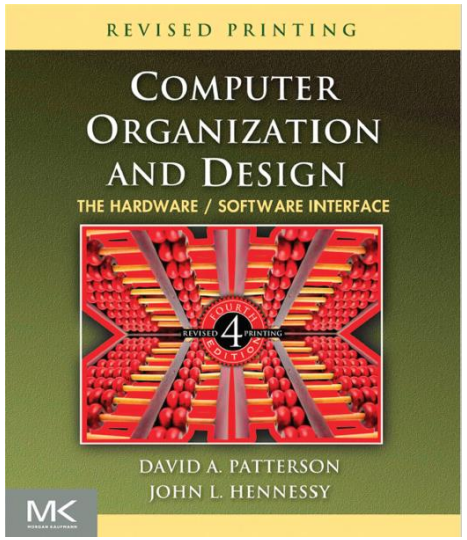
COAL group (two students) projects will be based on Arduino board (having MIPS based assembly language). The hardware will be programmed in C/C++ language (programming in assembly is also possible). Project ideas and support will be given by the course and lab instructors. The key goal of semester project is to enable student to use off-the-self low-cost hardware platforms for various hardware interfacing tasks on-their-own in a high-level language. A secondary goal is to understanding how real-word (small-scale) processors using course theory and lab skills.

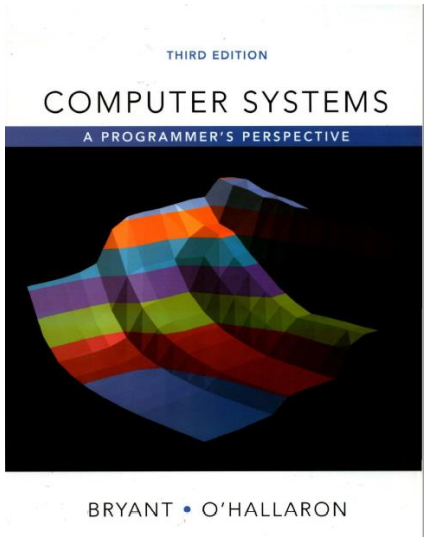
Course and Lab Mechanics

- Passing cut-off is 50 marks.
- Grading will those secured >= 50 marks.
- Attendance will be take in the first 5 minutes. After that you will be marked Late in the class.
- Assignment submitted after the due date will get only 50% marks. ZERO marks due + 2 days.
- 10 marks will be deducted for first plagiarism case from both students. Semester work will be marked ZERO in other cases.
- Class participation will get 10%
  - Notebook as per instructor
  - Question (verbal) with answers on notebooks.
  - Class demonstrations
- Semester project based on self-study and group of two students. Use of Arduino platform programmed using AVR assembly.
- COAL has little theory hence partial marks.

- Lab format:
  - Pre-lab (quiz of previous lab)
  - New Lab
  - Post lab (per lab assignment)
- Labs are for acquiring practical skills using tools and techniques ONLY.
- Q&As and Explanations should be performed during Office Hours of Lab Instructors.
- “A” grade in Lab will only be given to students who maintained a lab file and submitted 100% pre and post labs.
- Labs constitute 60% weightage. Midterm and Final Exam are 40%. If you miss a lab it will cost you a grade point fraction.
- Post-Lab deliverables must be submitted on Slate. Lab instructors may ask you to submit hand written document as well.
- Attendance will be take in the first 5 minutes. After that you will be marked Late in the Lab.
- Same rules for plagiarism/late submissions.

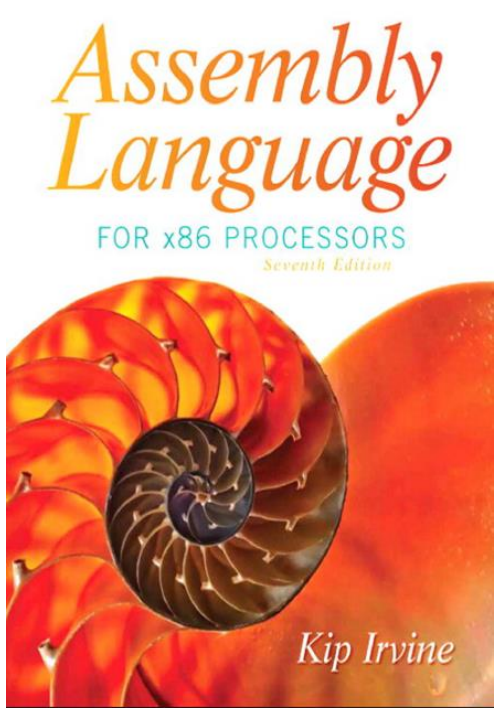
Text books

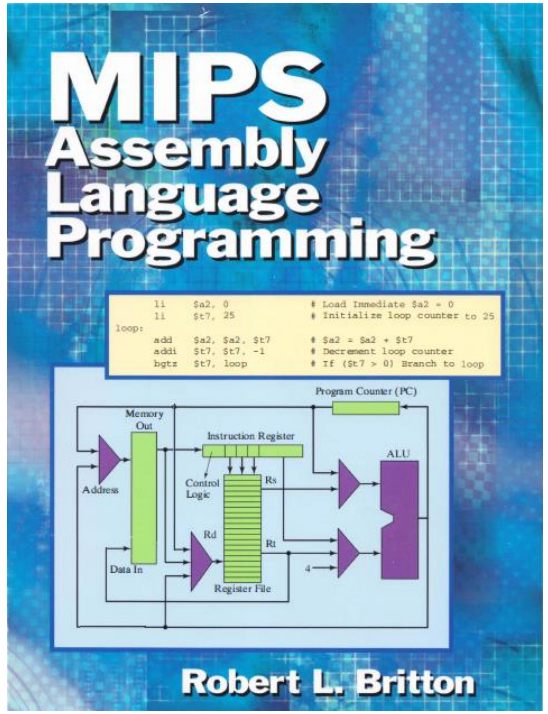




40% Course Coverage

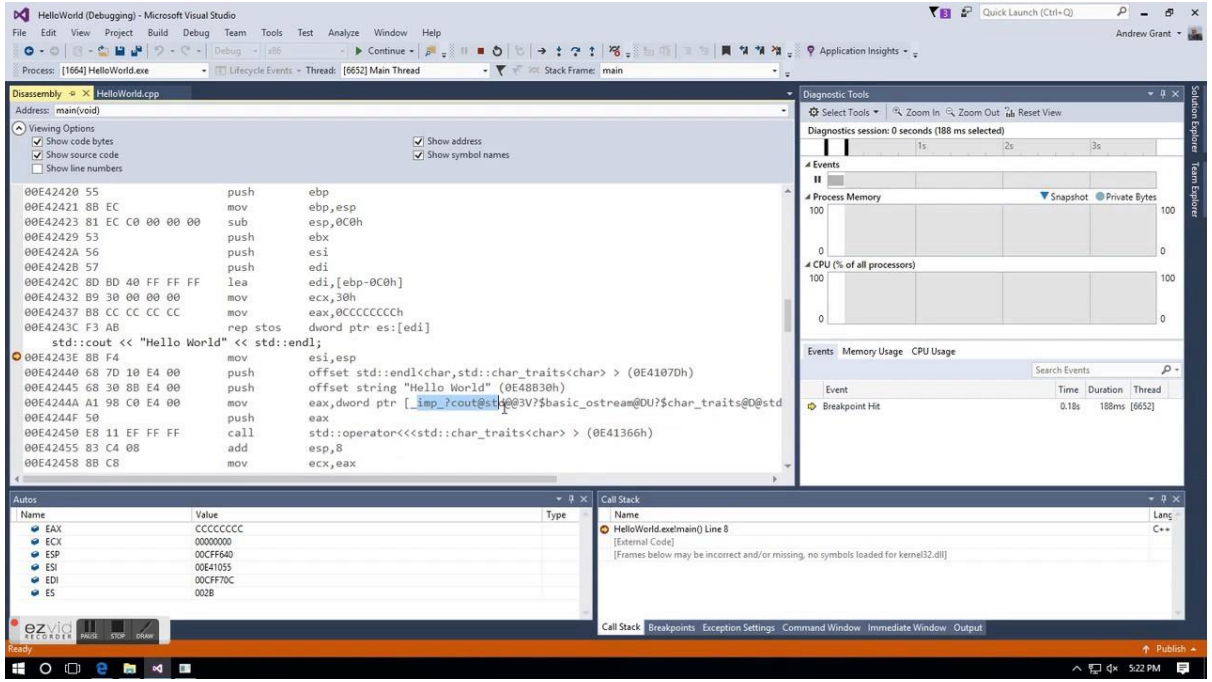
60% Course Coverage



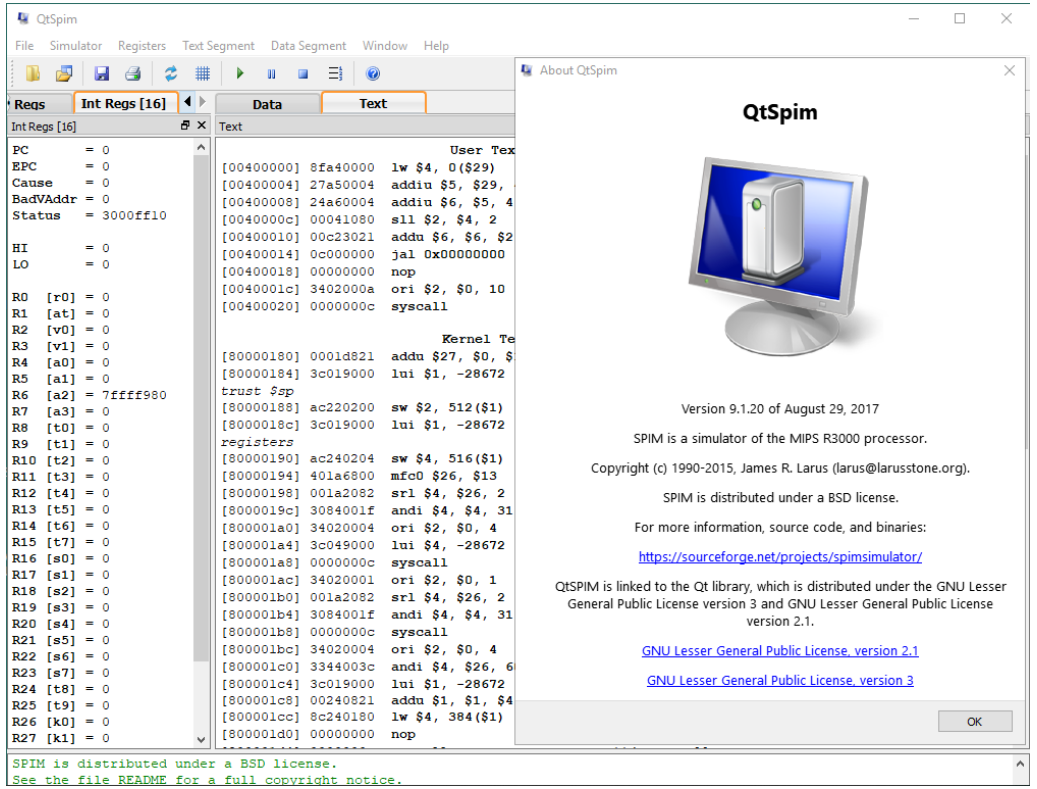


Tools for Assembly Language (1)

X86 MASM in Visual Studio



SPIM MIPS 3000 Simulator



Assembly Programming (x86 & MIPS) using tools including debugger and step-by-step execution

Tools for Assembly Language (2)