C S 10: COMPUTER ARCHITECTURE & ORGANIZATION

Foothill College Course Outline of Record

Heading	Value
Effective Term:	Summer 2021
Units:	4.5
Hours:	4 lecture, 2 laboratory per week (72 total per quarter)
Prerequisite:	One of the following: C S 1A, 2A or 3A.
Advisory:	C S 1C or 2C.
Degree & Credit Status:	Degree-Applicable Credit Course
Foothill GE:	Non-GE
Transferable:	CSU/UC
Grade Type:	Letter Grade (Request for Pass/No Pass)
Repeatability:	Not Repeatable

Student Learning Outcomes

- The student will demonstrate the ability to analyze the assembly language instructions generated by a C, C++ or Java program.
- The student will demonstrate knowledge of the architecture of a microprocessor including the use of registers, the program counter, and the arithmetic logic unit.

Description

Introduction to the organization, architecture and machine-level programming of computer systems. Topics include mapping of high-level language constructs into assembly code, internal data representations, numerical computation, virtual memory, pipelines, caching, multitasking, MIPS architecture, MIPA assembly language code, interrupts, input/output, peripheral storage processing, and comparison of CISC (Intel) and RISC (MIPS) instruction sets.

Course Objectives

The student will be able to:
A. Describe the architectural components of a computer system.
B. Discuss and demonstrate the use of compilers, linkers, and loaders.
C. Describe computer representation of numbers and how computer arithmetic is carried out.
D. Describe the representation of nonnumeric data (character codes, graphical data).
E. Demonstrate the knowledge of MIPS assembly language.
F. Compare and contrast MIPS architecture and assembly language with IA32.
G. Write and debug assembly programs that use load/store, arithmetic, logic, branches, call/return and push/pop instructions.
H. Discuss how variable access, arithmetic, function calls, and pointers are translated from a high level language into assembly.
I. Write programs that interface between a high level language and assembly.
J. Write programs that contain system calls.
K. Demonstrate and evaluate the use of efficient programming techniques.

Course Content

A. The architectural components of a computer system (the von Neumann machine: CPU (registers, ALU), memory, buses)
1. CPU
a. Control-unit
b. Instruction fetch
c. Decode
d. Execution
2. Memory
a. Bits
b. Bytes
c. Words
3. Registers
4. Flag bit hardware
5. Relationship of the hardware components to software
6. Instruction sets and architectures
a. Data manipulation
b. Control
c. I/O
7. Addressing modes
a. Immediate
b. Register
c. Memory
8. Buses
B. Prepare high level language programs for execution
1. Invoking the compiler
a. Choosing the optimization level
b. Examine the assembler language program generated by the compiler
2. Use a linker to generate the executable program
3. Use the loader to execute the program
C. The computer representation of numbers and how computer arithmetic is carried out
1. Binary
2. Hexadecimal
3. Packed decimal
4. Integer (signed and twos compliment representations)
5. Floating point
6. Conversions between numeric representations
D. The representation of nonnumeric data (character codes, graphical data)
1. BCD, EBCDIC, ASCII, and Unicode character codes
2. Rendering pixels on displays and printers
E. Representation of arrays and records
F. Instruction sets, assembly and machine language programming
1. Syntax (instruction format)
2. Data transfer instructions
3. Binary integer arithmetic
4. Conditional branching
a. Related hardware instructions
b. Extended mnemonics
c. Unconditional jump
d. Conditional jump
5. Array access
6. String processing instructions
7. Bit-Level instructions
8. Run time stack instructions
9. Packed decimal instruction
10. Extensions 64-bit instructions

G. RISC (MIPS) instruction set architectures
1. Motivation for RISC
a. Chip simplification, cost reduction, performance improvements
2. Current thoughts on CISC vs. RISC
H. Write and debug assembly programs that use load/store, arithmetic, logic, branches, call/return and push/pop instructions
1. Syntax
2. Variables and constants
a. Declarations
b. Redefinition
c. Conversion between types
3. Subroutine call and return
a. Parameter passing
b. Save and restore conventions
c. Local variables
4. Copying instructions
5. Binary integer arithmetic
6. Conditional branching
a. Hardware level
b. Extended mnemonics
c. If and If...Else...
7. Looping
a. While construct
b. For construct
c. Repeat construct
8. Array access
a. Direct addressing
b. Indirect addressing
9. String processing instructions
10. Bit-Level instructions
11. Run time stack instructions
I. Examine and demonstrate how variable access, arithmetic operations, function calls, and pointers are translated from a high level language to assembly language
1. Use the compiler to produce assembly language instructions
2. Determine the assembly language instructions generated by specific high level language instructions
J. System calls
1. Calling from assembly code
K. Evaluate efficiencies in programming
1. Location of variables
2. Exploiting pipelining
3. Exploiting multicore processors

Lab Content

A. Assembler and compiler installation
1. Install a MIPS assembler/simulator and a C compiler
2. Compile a simple C to demonstrate the proper installation of the compiler
3. Use the assembler to generate the corresponding object code
4. Run the linkage editor and execute the program
B. High level language programming and numeric representations
1. Write a program in a high level language that stores and converts between a variety of numeric representations
2. Compile the program
3. Examine the output of the compiler and correlate the high level language instructions with the generated assembler instructions
4. Discuss the various numeric representations and how they interoperate
C. High level language programming and non-numeric representations
1. Write a program in a high level language the stores and converts between a variety of non-numeric representations
2. Use BCD, EBCDIC, ASCII and Unicode character codes and convert among them
3. Compile the program
4. Examine the output of the compiler and correlate the high level language instructions with the generated assembler instructions
E. Basic assembly language programming
1. Write an assembly language program which demonstrates data transfer instructions, binary arithmetic, and conditional branching
2. Assemble and execute the program
3. Evaluate the output
F. Advanced assembly language programming
1. Write an assembly language program which demonstrates array access and manipulation, String processing, bit-level instruction
2. Assemble and execute the program
3. Evaluate the output
G. Interaction between high level and low level languages
1. Write an high-level language program which calls, passes data to, and receives data from another assembly language program
2. Assemble and execute the program the program
3. Examine the output of the program to ensure that it produces the desired results
H. Interaction between assembly language programs and system calls
1. Write an assembly language program which calls, passes data to, and receives data from operating system calls
2. Assemble and execute the program the program
3. Examine the output of the program to ensure that it produces the desired results
I. Writing efficient high level language programs
1. Compile a sample source program and analyze the resulting assembly language program
2. Modify the source program in order the resulting assembly language program execute faster
3. Analyze and explain why the changes made the assembly language program execute faster

Special Facilities and/or Equipment

A. Computer laboratory with MIPS simulator and C compilers.
B. Website or course management system with an assignment posting component (through which all lab assignments are to be submitted) and a forum component (where students can discuss course material and receive help from the instructor). This applies to all sections, including on-campus (i.e., face-to-face) offerings.
C. When taught via Foothill Global Access, a fully functional and maintained course management system through which the instructor and students can interact.
D. When taught via Foothill Global Access, students must have currently existing email accounts and ongoing access to computers with internet capabilities.

Method(s) of Evaluation

Tests and quizzes
Written laboratory assignments which include source code, sample runs and documentation
Final examination

Method(s) of Instruction

Lectures which include motivation for the architecture of computer systems (CPU, RAM, storage), syntax and use of the ISA 32 assembly language, example programs, and analysis of these programs

Online labs (for all sections, including those meeting face-to-face/on-campus), consisting of:
1. A programming assignment webpage located on a college-hosted course management system or other department-approved internet environment. Here, the students will review the specification of each programming assignment and submit their completed lab work
2. A discussion webpage located on a college-hosted course management system or other department-approved internet environment. Here, students can request assistance from the instructor and interact publicly with other class members
Detailed review of programming assignments which includes model solutions and specific comments on the student submissions
In-person or online discussion which engages students and instructor in an ongoing dialog pertaining to all aspects of designing, implementing and analyzing programs
When course is taught fully online:
1. Instructor-authored lecture materials, handouts, syllabus, assignments, tests, and other relevant course material will be delivered through a college-hosted course management system or other department-approved internet environment
2. Additional instructional guidelines for this course are listed in the attached addendum of CS department online practices

Representative Text(s) and Other Materials

Patterson, David. Computer Organization and Design MIPS Edition: The Hardware/Software Interface, 6th ed.. 2020.

Null, Linda. Essentials of Computer Organization and Architecture, 5th ed.. 2018.

Types and/or Examples of Required Reading, Writing, and Outside of Class Assignments

A. Reading
1. Textbook assigned reading averaging 30 pages per week.
2. Supplied handouts and modules averaging 10 pages per week.
3. Online resources as directed by instructor though links pertinent to programming.
4. Library and reference material directed by instructor through course handouts.
B. Writing
1. Technical prose documentation that supports and describes the programs that are submitted for grades.

Discipline(s)

Computer Science