

Computer Science 251
Computer Organization and Assembly Language
Spring 2009

Course: Computer Organization and Assembly Language (3 units)

Current Catalog Description

COMP SCI 251 Comp Org & Assembly Language (3 units)

An introduction to RISC-based instruction set architecture. Topics to be studied include: data representation, assembly language programming, and introduction to system software.

Prerequisite: Computer Science 221 with a grade of C or better. (Fall, Spring)

Time & Place: 8:00 a.m. to 9:30 a.m. Tuesday Thursday HS 212

Instructor: Wing Huen **Office:** HS 221 **Phone:** 424-1324 **Email:** huen@uwosh.edu

Office Hours: Tuesday Thursday: 9:30 a.m. to 11:30 a.m.; 1:00-2:00 p.m.
or by appointment.

Required Textbook

Computer Organization and Assembly Language by Jonathan Yackel and Andrew Perrie, December 2008.

References

Instructors lecture notes available at Q:\Shared\huen\251
MIPS examples are on Linux /shared/huen/251
Computer Organization and Design by David A. Patterson and John L. Hennessy, Morgan Kaufmann, Third Edition
Introduction to RISC assembly Language Programming by John Waldron, Addison-Wesley

Course Outcomes

Topic areas and corresponding Course Outcomes

This course aims to train students on processor and memory hardware, mapping high level language programs onto a Reduced Instruction Set Architecture (RISC). The student will be able to analyze why programs behave the way they do and how inefficiencies arise.

Data Representation is covered in Course Outcomes 1-4

- i. Character
- ii. Integer
- iii. IEEE 754 Floating Point Standard

Instruction Set Architecture of a RISC computer (the MIPS R2000 processor) is covered in Course Outcomes 5-10

- CPU, Memory and the system bus
- Memory layout
- Assembly language instructions
- assembly language implementation of high level language control structures
- assembly language implementation of one and two dimensional arrays
- Pseudo-instructions and their replacement with real instructions

System stack and the return address, Implementation of functions in a high-level language is covered in Course Outcomes 11-13

- Call-by-value, call-by-reference, call-by-array-reference
- Parameter passing methods
- MIPS register conventions
- Stack frames and local variables

Computer architecture below the instruction level is covered in Course Outcomes 14-24

- b. Fetch-execute cycle
- c. Instruction encoding
- d. Introduction to direct-mapped cache
- e. Introduction to pipelining

Communicate effectively, both orally and in writing is covered in Course Outcomes 25-26

Teamwork is covered in Course Outcomes 27-29

At conclusion of this course, students will be able to:

- | |
|--|
| <ol style="list-style-type: none">(1) Express characters and integers in binary, hexadecimal, signed and unsigned representations(2) Determine whether overflows occur in signed or unsigned additions and subtractions of integers(3) Write normalized and denormalized floating point numbers in single and double precision using the IEEE 754 Floating Point Standard(4) Analyze the IEEE 754 Floating Point Standard and determine what integers that cannot be represented exactly by the Floating Point Standard(5) Organize the memory layout of global integers and characters assuming the Little-Endian and Big-Endian notations.(6) Edit an assembly language program, assemble the program and print output on |
|--|

console using Linux

- (7) Design assembly language program given high-level source code.
- (8) Implement assembly language programs that read in integers from console, process the input and print results on the console
- (9) Implement high-level language control structures in assembly language
- (10) Implement one and two-dimensional arrays and control structures in assembly language (do-while, if-else, and for loop)
- (11) Write nested function calls using stack frames and local variables
- (12) Write an assembly language program to call recursive functions
- (13) Write an assembly language function to perform numerical analysis e.g. finding the square root of a double precision floating point number
- (14) Implement high-level language switch statements with jump tables
- (15) Interpret the instruction encoding of MIPS instructions.
- (16) Analyze the instruction-encoding of control instructions such as branch-not-equal and jump instruction in the MIPS computer
- (17) Design a direct-mapped cache unit with a data capacity with a size much smaller than that of main memory, assuming each cache block holds 2^n words of data from main memory. Assume the main memory is byte-addressable, with 32-bit addresses.
- (18) Determine whether data hazards exist in a section of code, assuming the five-stage MIPS pipeline. (Review of Homework/quiz/exam/project)
- (19) Map an instruction or a data word from main memory to a direct mapped cache.
- (20) Given a word in a direct mapped cache, determine the memory address in the main memory.
- (21) Reduce data hazards in pipelining
- (22) Draw a timeline showing the instructions moving through a five-stage MIPS pipeline without any stalls.
- (23) Identify data dependencies where the result is needed before it is computed
- (24) Remove all data hazards by inserting the minimum number of nop instructions
- (25) Create and document program design solutions for MIPS programs
- (26) Present and justify, to a group of peers, the design and implementation of a problem solution
- (27) Plan for and schedule adequate time to complete assembly language

projects no later than the required due date

- (28) Consult various online and independent resources to independently attempt to resolve problems BEFORE requesting assistance from co-workers/co-learners or supervisor/instructor
- (29) Determine whether it is appropriate to seek assistance, from co-workers/co-learners or supervisor/instructor to resolve problems that could not be resolved independently.

Assembly Programming Projects on implementing combinations of the following high level language features:

- Simple Arithmetic
- If statements and loops
- while loop, for loops, do-while loops
- Arrays, jump tables, switch statement
- Bit Manipulations
- Functions and parameters
- Functions using floating-point data

Prerequisites by Topic

- Control statements (CS221)
- Iteration (CS221)
- Arrays (CS221)
- Functions (CS221)
- Object class and methods (CS221)
- College Algebra (MATH 104)

Major Topics Covered in the Course

- Basic Computer Organization and Memory Layout, including Big-Endian vs Little-Endian byte ordering
- Integer Representation: sign-magnitude, 1's Complement, 2's Complement
- Instruction set of the MIPS R2000 processor
- Pseudo-instructions vs. real machine instructions
- The fetch-execute cycle
- The stack, functions, recursive functions, function calls and the MIPS Register Conventions
- Compiler stack frame and memory for local variables on the stack
- Objects, classes, and methods of Object-Oriented Programming
- IEEE 754 Floating Point Standard: Floating Point representation and Floating Point Processing Unit

- Memory Hierarchy and Direct-mapped Cache
- Pipelining

Course Schedule:

	Dates	Tuesday		Thursday	
1	2/3, 2/5	Ch. 1	Overview, Syllabus, Number systems	Ch. 2	Data Representation
2	2/10, 2/12	Ch. 2	Data Representation	Ch. 3	MIPS RISC machine, Linux
3	2/17, 2/19	Ch. 3	MIPS, RISC	Ch. 4	Variables and Expressions
4	2/24, 2/26	Ch. 4	Variables and expressions	Ch. 4; Review for Exam 1	Variables and expressions
5	3/3, 3/5	Ch. 5, Exam 1	Control structures; Exam 1 on 3/3	Ch. 5	Control structures
6	3/10, 3/12	Ch. 5	Control structures	Ch. 6	Arrays
7	3/17, 3/19	Ch. 6	Arrays	Ch. 7	Bitwise operations;
	3/22-3/29		Spring Break		Spring Break
8	3/31, 4/2	Ch. 8	Functions	Ch. 8	Functions
9	4/7, 4/9	Ch. 8	Functions	Ch. 8	Functions
10	4/14, 4/16	Ch. 9	Objects; Review for Exam 2	Exam 2 on 4/16	
11	4/21, 4/23	Ch. 10	Floating point	Ch. 11	Instruction encoding
12	4/28, 4/30	Ch. 11	Instruction encoding	Ch. 12	Cache
13	5/5, 5/7	Ch. 12	Cache	Ch. 13	Pipelining
14	5/12, 5/14	Ch. 13	Pipelining; Review	Exam 3 on 5/14	In class exam

All three exams are 1-hour exams to be held during class time. You will be allowed to take a one-page note to the exams.

Grading: Your course grade will be based on exams, homework, and quizzes as follows:

- three exams, each worth 20% of your grade, for a total of 60% of your grade
- homework assignments worth 30% . Your completed programs should be submitted to the Linux directory /DROBOXES/251/*lastnfdd* where *lastnfdd* is your login name
- weekly quizzes worth 10%

Your letter grade for the course will be based on the following scale.

Score	Grade
92-100	A
89-91	AB
82-88	B
79-81	BC

72-78	C
69-71	CD
57-68	D
< 57	F

Exams: Exam 1: March 3, 2009
Exam 2: April 16, 2009
Exam 3: May 14, 2009

Let the instructor know as far in advance as possible if you can not attend on one of these dates so that makeup arrangements can be made. The exams will be closed-book, however you may bring one 8 ½ × 11 sheet of notes to each exam.

Homework assignments will be given regularly throughout the semester. Some assignments will be written problem sets; others will be assembly language programming assignments. You are encouraged to work on written homework in study groups but you must write up your own answers individually and in your own words. Written homework is due at the beginning of class on the due date. Late written homework is not accepted. Some programming assignments may be done in programming teams. On-time completion of programming assignments is important. If any component of a programming assignment is turned in late, the following penalties apply to the assignment.

Turned in	Penalty
on due date, after deadline	10%
one day late	25%
two days late	50%
three days late	75%

Note that programming assignments more than three days late receive no points. Weekend days and holidays count as “regular days” when computing late penalties. Extensions on homework will be granted at the discretion of the instructor if you contact the instructor and provide a written excuse from a medical doctor **before** the due date.

Quizzes: Written and oral quizzes will be given regularly, except for exam weeks. There are no make-ups for missed quizzes.

Frequently Asked Questions (FAQ)

How can I get good grades in this course?

Read the textbook according to the course schedule shown above in this syllabus before coming to class and ask questions. For the assembly language programming assignments, be sure to write down the pseudo-code or flowchart first before coding.

Start your programming assignments early. If you are stuck with a programming assignment, talk it over with the tutors and/or the instructor. Don't wait until the last minute.

What do I have to hand in for these programming assignments and what if it's late?

Handing in a programming assignment requires that you (1) submit an electronic copy of your program to the /DROPBOXES/251 directory and (2) hand in a neatly stapled report, which will include a hard copy of your program, as well as notes (including pseudo-code or flowcharts) on how you solved the problem. If you have erroneous execution results but could not debug your program in time, indicate the error in your report. Each assignment will carry with it a due date. The electronic copy of your assignment is due "at the bewitching hour" on the due date. The hard copy is due at or before the beginning of the first class meeting following the due date. The hard copy must completely match the electronic copy. Failure to submit either component on time means that your assignment is late. Late programming assignments will be accepted but will be penalized at the rate shown above.

Is there any way I can carelessly lose points in the course?

Be late in handing in your work on assignments.

Don't "participate" in the class and do poorly on the quizzes.

What is this class participation stuff? How does one "participate" in a subject like this?

Read the textbook and the lecture notes before the class. Be prepared to ask questions. Do well on the written quizzes and oral quizzes in class.

"Research has demonstrated that after a lecture, students recall 62% of the information. However, only 45% is recalled by students after 3-4 days and in 8 weeks only 24% of the information is recalled. If a quiz or exam was administered after the lecture, recall was doubled at the 8-week period. It is interesting that many faculty members appear to ignore the potential impact which quizzes and tests can have upon learning." -- Bonwell C.C., Eison J.A.: Active Learning: Creating Excitement in the Classroom. Washington, DC: George Washington University, 1991.

Can I get an extension on work that is due on a specified date?

Only if you're gravely ill. Be sure that you have signed documents from a medical professional to verify the illness.

If I miss an exam, can I make it up?

Only if you were gravely ill at the time of the exam. Be sure that you have signed documents from a medical professional to verify the illness. You cannot make up a quiz.

Can I work with others on programming assignments?

Some assignments, if indicated, may be worked as a team in this course. All the programming assignments are to be the sole work of the individual student or your team. You may help each other by discussing your approach or difficulties with other students but you should not share your program with others or copy the programs of other students. You may discuss the programming assignments at a high level (i.e., work on flow charts) with others, but the lower level work (i.e., all the work on a computer) must be done by you or your team. You are encouraged to seek the help of the instructor and the assigned tutors of this course.

Submitting an assignment that was not entirely done by you/your team is considered academic dishonesty and will result in appropriate disciplinary action.

Can I do programming assignments on my own computer instead of using the computer systems physically in a university lab?

Sure, there are several ways:

1. PCSPIM on a Windows machine. You install PCSPIM on your own computer system but you still need to submit your assignment to the /DROPBOXES on the Linux machine. Be sure to set up the PCSPIM with the notrap option. Also note that there are differences in the DOS and Linux file formats.
2. Standalone Linux: You install Linux on your own computer system but you still need to submit your assignment to the /DROPBOXES, or
3. Online approach: access the UWO Linux system remotely via putty.
4. Online approach: access the UWO Linux system remotely via secure VNC

Option 3 or 4 is recommended since the files you created are directly on the university system and it frees you from administration of your own Linux software.

How can I get a good grade?

Scan the class material before class. Ask the instructor and/or lab assistants on the finer points you don't understand. Work on exercises and the quizzes in the textbook to check your understanding. **Be prepared to spend at least 3 hours for each hour of contact time in class.**