

**Computer Science 251**  
**Computer Organization and Assembly Language**  
**Fall 2008**

**Course: Computer Organization and Assembly Language** (3 units)

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

**Time & Place:** 8:00 a.m. to 9:30 a.m. Tuesdays and Thursdays HS 208

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

**Office Hours:** Tuesdays Thursdays: 9:30 a.m. to 11:00 a.m. and 1:00 p.m. to 2:30 p.m.  
**or by appointment.**

**Required Text:** J. Yackel and A. Perrie, Computer Organization and Assembly Language.

**Course material:**

- Lecture notes and homework assignments are posted on the Student drive Q:\Shared\Huen\251.
- MIPS examples are on Linux /shared/huen/251
- Your completed homework should be submitted to /DROPBOXES/251/*lastnfd* where *lastnfd* is your login name

**Reference:**

- Instructor's lecture notes
- 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 Learning Outcomes

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

1. Data Representation

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

The student is expected to:

- (a) express characters and integers in binary, hexadecimal, signed and unsigned representations (Review of Homework/quiz/exam/project)
- (b) determine whether overflows occur in signed or unsigned additions and subtractions of integers (Review of Homework/quiz/exam/project)
- (c) write normalized and denormalized floating point numbers in single and double precision using the IEEE 754 Floating Point Standard (Review of Homework/quiz/exam/project)
- (d) analyze the IEEE 754 Floating Point Standard and determine what integers that cannot be represented exactly by the Floating Point Standard (Review of Homework/quiz/exam/project)

## 2. Instruction Set Architecture of a RISC computer (the MIPS R2000 processor)

- 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

The student is expected to:

- (a) organize the memory layout of global integers and characters assuming the Little-Endian notation. (Review of Homework/quiz/exam/project)
- (b) edit an assembly language program, assemble the program and print output on console using Linux (Review of Homework/quiz/exam/project)
- (c) design assembly language program given high-level source code. (Review of Homework/quiz/exam/project)
- (d) implement assembly language programs that read in integers from console, process the input and print results on the console (Review of Homework/quiz/exam/project)
- (e) implement high-level language control structures in assembly language (Review of Homework/quiz/exam/project).
- (f) implement one and two-dimensional arrays and control structures in assembly language (do-while, if-else, and for loop) (Review of Homework/quiz/exam/project)

## 3. System stack and the return address, Implementation of functions in a high-level language

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

The student is expected to:

- (a) write nested function calls using stack frames and local variables (Review of Homework/quiz/exam/project)
- (b) write an assembly language program to call recursive functions (in-class activity/exam)

(c) write an assembly language function to perform numerical analysis e.g. finding the square root of a double precision floating point number (Review of Homework/quiz/exam/project)

4. Computer architecture below the instruction level
  - a. Fetch-execute cycle
  - b. Instruction encoding
  - c. Introduction to direct-mapped cache
  - d. Introduction to pipelining

The student is expected to:

- (a) design and implement assembly language programs to interpret the instruction encoding of MIPS instructions. (Review of Homework/quiz/exam/project)
- (b) implement high-level language switch statements with jump tables (Review of Homework/quiz/exam/project)
- (c) analyze the instruction-encoding of control instructions such as branch-not-equal and jump instruction in the MIPS computer (quiz/class exercise)
- (d) 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. determine whether data hazards exist in the a section of code, assuming the five-stage MIPS pipeline. (Review of Homework/quiz/exam/project)
- (e) map an instruction or a data word from main memory to a direct mapped cache. (quiz/class exercise)
- (f) given a word in a direct mapped cache, determine the memory address in the main memory. (quiz/class exercise)
- (g) reduce data hazards (quiz/class exercise)
- (h) draw a timeline showing the instructions moving through a five-stage MIPS pipeline without any stalls. (quiz/class exercise)
- (i) identify data dependencies where the result is needed before it is computed (quiz/class exercise)
- (j) remove all data hazards by inserting the minimum number of nop instructions (quiz)

### Laboratory Projects

- 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

### Course Outline:

1. Number systems
2. Data representation
3. The MIPS machine

4. Variables and expressions
5. Control structures
6. Arrays
7. Functions
8. Bitwise operations
9. Object oriented programming
10. Floating-Point Unit
11. Instruction representation
12. Cache memory
13. Pipelining

### Course Schedule:

	Dates	Tuesday		Thursday	
1	- , 9/4			Linux and editors	Linux, VI, emacs
2	9/9, 9/11	Ch. 1	Number systems	Ch. 2	Data Representation
3	9/16, 9/18	Ch. 2	Data Representation,	Ch. 3	MIPS RISC machine
4	9/23, 9/25	Ch. 3	MIPS, RISC	Ch. 4	Variables and Expressions
5	9/30, 10/2	Ch. 4	Variables and expressions	Ch. 4; Review for Exam 1	Variables and expressions
6	10/7, 10/9	Ch. 5, Exam 1	Control structures; Exam 1 in 10/7 evening	Ch. 5	Control structures
7	10/14, 10/16	Ch. 5	Control structures	Ch. 6	Arrays
8	10/21, 10/23	Ch. 6	Arrays	Conference – no class on 10/23	Arrays
9	10/28, 11/30	Ch. 7	Bitwise operations;	Ch. 8	Functions
10	11/4 11/6	Ch. 8	Functions	Ch. 8	Functions
11	11/11, 11/13	Ch. 8	Functions; Review for Exam 2	Ch. 9; Exam 2 in 11/13 evening	Objects; Exam 2 in 11/13 evening
12	11/18, 11/20	Ch. 10	Floating point	Ch. 11;	Instruction encoding
13	11/25, 11/27	Ch. 11; 12	Instruction encoding; Cache	Thanksgiving - No class	Thanksgiving
14	12/2, 12/4	Ch. 12	Cache	Ch. 13	pipelining
15	12/9, 12/11	Ch. 13	Pipelining ; Review	Exam 3	In class exam

Exams 1 and 2 are 1.5 -hour exams to be held in the evenings on the scheduled dates.  
Exam 3 is an 1.5-hour in-class exam.

**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, weighted equally, worth 30%

- 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 ( 1.5-hour exam) will be given out of class on Tuesday October 7, 2008 evening. Exam 2 (1.5-hour exam) will be given out of class on Thursday Nov. 13 evening. Exam 3 is planned to be held in class on Thursday December 11, 2008. 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. All programming assignments are to be done individually (no 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** will be given regularly, except for exam weeks. There are no make-ups for

missed quizzes. Your lowest quiz score will be dropped.

### **Frequently Asked Questions (FAQ)**

#### **How can I get good grades in this course?**

Read the course notes 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 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/*lastnfd* 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. 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?**

Do well on the written quizzes and oral quizzes.

"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?**

There are no team assignments in this course. All the programming assignments are to be the sole work of the individual student. 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. 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 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 two ways:

1. Standalone Linux: You install Linux on your own computer system but you still need to submit your assignment to the /DROPBOXES, or
2. Online approach: access the UWO Linux system remotely via putty.

Option 2 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.**