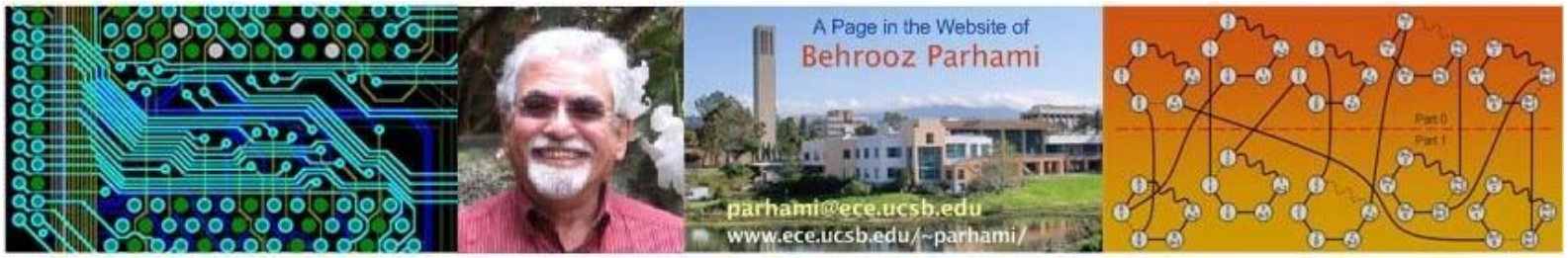
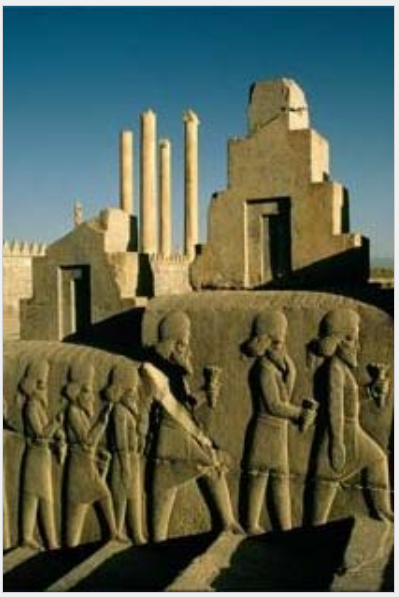


- Home & Contact
- Curriculum Vitae
- Research
  - Computer arithmetic
  - Parallel processing
  - Fault tolerance
  - Broader research
  - Research history
  - List of publications
- Teaching
  - ECE1 Freshman sem
  - ECE154 Comp arch
  - ECE252B Comp arith
  - ECE252C Adv dig des
  - ECE254B Par proc
  - ECE257A Fault toler
  - Student supervision
  - Math + Fun!
- Textbooks
  - Computer arithmetic
  - Parallel processing
  - Dependable comp
  - Comp architecture
  - Other books
- Service
  - Professional activities
  - Academic service
  - Community service
  - Industrial consulting
- Files & Documents
- Useful Links
- Personal

Behrooz Parhami's ECE 154A Course Page for fall 2014

Introductory Computer Architecture



Page last updated on 2014 December 22

Enrollment code: Depends on the discussion session chosen
Prerequisite: ECE 152A or equivalent
Class meetings: TR 2:00-3:15, Psych 1902
Discussion option 1: F 09:00-09:50 (TA1, code 13698), Phelps 1508
Discussion option 2: F 10:00-10:50 (TA3, code 13714), Phelps 3523
Discussion option 3: F 11:00-11:50 (TA2, code 13706), NH 1109
Instructor: Professor Behrooz Parhami
Teaching assistant 1: Nicole Lesperance, nlesperance at umail.ucsb.edu
Teaching assistant 2: David McCarthy, davidmccarthy at umail.ucsb.edu
Teaching assistant 3: Zachary Wells, zawells at umail.ucsb.edu
Intructor's office hours: M 3:00-4:30, R 3:30-5:00, HFH 5155
TA office hours: TA1 T 3:30-4:30, TA2 W 3:00-4:00, TA3 R 5:30-6:30
Location of TA office hours: Trailer 699 Room 103
Course announcements: Listed in reverse chronological order
Course calendar: Schedule of lectures, assignments, and exams
Homework assignments: Three assignments, worth a total of 15%
Practical projects: Two projects, worth a total of 20%
Exams: Midterm (25%) and final (40%), both closed-book
Policy on academic integrity: Please read very carefully
Grade statistics: Range, mean, etc. for assignment/exam grades
References: Textbook and other info sources (**Textbook's web page**)
Lecture slides: Available on the textbook's web page
Miscellaneous information: Motivation, catalog entry, history

Course Announcements

2014/12/22: The fall 2014 offering of ECE 154A is officially over and course grades have been submitted to the Registrar. I take this opportunity to thank the course TAs (Nicole, David, Zach) for their hard work and to apologize for the mishaps with course projects, particularly Project 2. This was my first time teaching ECE 154A after the lecture-only ECE 154 was revamped, incorporating material from the defunct ECE 15B and split into two courses with projects.
If I teach this course again, I will integrate the two projects, making Project 2 a continuation of Project 1. In this way, students won't have to start from scratch in Project 2, which comes at a time during the quarter when much else is going on. With best wishes for a joyous holiday season and a bright new year!
2014/12/17: The final exam is being graded and course grades will be reported to the Registrar by the end of this week. Questions about HW3 and Project 2 grading should be submitted to the course TAs (please copy all three) by midnight on Thursday 12/18. They will review submitted claims and will report proposed adjustments to me for final decision. No adjustments to HW1, HW2, and Project 1 grades are possible at this juncture.
2014/12/02: Homework 3 (the last course assignment, due on 12/12) has been posted to the homework area below. Please note the response format restrictions for HW3 that would allow grading to be completed by the final exam time. Updated slides for Part 5 of the textbook (Chapters 17-20) will be posted to the textbook's Web page no later than Thursday 12/04.
2014/11/27: Please see the hint about modular program development and testing added at the end of the description for Project 2.

Blog & books
Favorite quotations
Poetry
Pet peeve
Virtual retirement
CE Program
ECE Department
UCSB Engineering
UC Santa Barbara

2014/11/21: With regard to Project 2, it was brought to my attention that not allowing shift instructions will unnecessarily complicate the programming task. Therefore, you are now allowed to use sll and srl (left and right logical shifts, with constant shift amounts) in addition to the 23 previously allowed instructions. The project specification has been modified accordingly.
2014/11/16: Project 2, due on 12/05, has been posted to the projects area below. Although the problem defined is different, project deliverables will be very similar to those of Project 1. Updated slides for Part 4 of the textbook (Chapters 13-16) will be posted to the textbook's Web page tomorrow night.
2014/11/07: Sample midterm, final, and study guide have been updated for fall 2014. You can use a simple scientific or graphing caclulator for the closed-book midterm, but no device that stores information or connects to the Internet will be allowed.
2014/10/28: Homework 2, due on 11/07, has been posted to the homework area below. Updated slides for Part 3 of the textbook (Chapters 9-12) will be posted to the textbook's Web page tonight. Due to an important meeting, I will not have an office hour on M 11/3. Note that in the description of Project 1, ".global" was a typo; it should be ".globl" (I know, it's weird to say that "globl" is the correct form of "global"). Note that for the MIPS simulator to run properly, your program must end with an exit system call.
2014/10/15: Project 1, due on 10/31, has been posted to the projects area below. Updated slides for Part 2 of the textbook (Chapters 5-8) have been posted to the textbook's Web page.
2014/10/10: In today's 9:00 AM discussion session, Nicole (the TA for that session) found a set of keys. If the keys are yours, please contact her via e-mail.
2014/10/07: Homework 1, due on 10/17, has been posted to the homework area below. Also, two copies of the textbook have been placed on 2-hour reserve at the library.
2014/10/02: The information on this page is now final for fall 2014. Updated lecture slides for Part 1 of the textbook (Chapters 1-4) have been posted to the textbook's Web page.
2014/07/14: Welcome to the ECE 154A Web site for fall 2014. The following information is tentative and is provided for planning purposes only. The course lecture schedule and requirements will be finalized in late September 2014, with updates appearing at least weekly thereafter. Major changes and additions to the page content will be outlined in this announcements area.

Course Calendar

Course lectures, homework assignments, practical projects, and exams are scheduled as follows. This schedule will be strictly observed. While not mandatory, class attendance helps in identifying important topics and compiling a study guide, particularly if the material covered in each lecture are studied or at least browsed beforehand. PowerPoint and pdf files of course lectures can be found on the **Textbook's web page**.

Day & Date (book chapters) Lecture/discussion topic [Homework posted/due] {Special notes}
R 10/02 (ch. 1) Course introduction, review of combinational logic circuits
F 10/03 (ch. 1-2) Discussion on logic circuits

T 10/07 (ch. 2) Review of sequential logic circuits and registers [Homework 1 posted, ch. 1-4]
R 10/09 (ch. 3) Computer technology overview
F 10/10 (ch. 1-3) Discussion on computer hardware; Project 1 preview

T 10/14 (ch. 4) Introduction to computer performance
R 10/16 (ch. 5) MiniMIPS instructions and addressing modes [Project 1 posted, ch. 5-8]
F 10/17 (ch. 4) Discussion on computer performance and HW1 [Homework 1 due]

T 10/21 (ch. 6) MiniMIPS additional instructions
R 10/23 (ch. 7-8) Assembly programs and ISA variations
F 10/24 (ch. 5-8) Discussion on instruction-set architecture and Project 1

T 10/28 (ch. 9) Number representation [Homework 2 posted, ch. 9-12]
R 10/30 (ch. 10) Addition circuits and simple ALUs
F 10/31 (ch. 9-10) Discussion on number representation, addition, and ALUs [Project 1 due]

T 11/04 (ch. 11) Multiplication and division
R 11/06 (ch. 12) Floating-point arithmetic
F 11/07 (ch. 11-12) Discussion on computer arithmetic and HW2 [Homework 2 due]

T 11/11 Veterans Day Holiday: No lecture
R 11/13 (ch. 1-12) Midterm exam, in our regular classroom (closed-book)
F 11/14 Discussion on midterm exam

T 11/18 (ch. 13) Stages of instruction execution [Project 2 posted, ch. 13-15]
R 11/20 (ch. 14) Control unit synthesis
F 11/21 (ch. 13-14) Discussion on data path and control

T 11/25 (ch. 15) Basics of pipelined data paths
R 11/27 Thanksgiving holiday: no lecture
F 11/28 Thanksgiving weekend: no discussion sessions

T 12/02 (ch. 16) Advanced pipelining and performance limit [Homework 3 posted, ch. 16-19]
R 12/04 (ch. 17, 19) Main and mass memory concepts
F 12/05 (ch. 15-16) Pipelined data paths and their performance implications [Project 2 due]

T 12/09 (ch. 18) Cache memories
R 12/11 (ch. 4, 16, 18) Computer performance revisited
F 12/12 (ch. 4, 16-19) Discussion on memory, performance, and HW3 [Homework 3 due]

T 12/16 (ch. 1-19) Final exam, 4:00-7:00 PM, in our regular classroom (closed-book)
T 12/23 {Grades to be submitted by midnight}

Homework Assignments

-Deposit solutions in ECE 154 homework box (3120 HFH) before 9:00 AM on the due date.
-Because solutions will be handed out on the due date, no extension can be granted.
-Use a cover page that includes your name, course name, and assignment number.
-Staple the sheets and write your name on top of each sheet in case they are separated.
-Some cooperation is permitted, but direct copying of work will have severe consequences.

Homework 1: Computer technology and performance (ch. 1-4, due F 10/17, 9:00 AM)
Do these problems from the textbook: 1.9c, 2.6, 3.16, 4.6, 4.12

Homework 2: Computer arithmetic (ch. 9-12, due F 11/07, 9:00 AM)
Do these problems from the textbook: 9.14ac (correction: in part c, all three numbers should be negative), 10.6, 10.21, 11.2ab, 12.8

Homework 3: Pipelining and memory (ch. 16-19, due F 12/12, 9:00 AM)
Do these problems from the textbook (see format requirements below): 16.10, 17.3, 18.2, 18.4ae, 19.9
Problem 17.3: Please provide your answer in the form of a table with the following columns: Total MB of memory, #256-Mb chips, Possible data widths for 256-Mb chips, #1-Gb chips, Possible data widths for 1-Gb chips
Problem 18.4 (only parts a and e have been assigned): Classify each memory access using the following abbreviations: cache hit = h, compulsory miss = f (for "forced"), capacity miss = s (for "size"), conflict miss = m (for "mapping") and write the letter code after the memory address as follows: 0 = x, 1 = x, 2 = x, 3 = x, 4 = x, 15 = x, 14 = x, etc. (where x is h, f, s, or m in your answer)
Problem 19.9: Please box final answers for each part

Practical Projects

Project specifications, documentation requirements, and submission procedure will appear here when available. Projects are assigned to each student and must be completed individually and independently. Rules of academic honesty apply to projects just like homework assignments and exams. Please begin work on assigned project early, as no extension to the deadline will be granted.

Project 1: Assembly language programming (ch. 5-8, due F 10/31, 9:00 AM)

Project description
"Identifying the Largest and Smallest Among *n* Signed Integers with at Most 3*n*/2 Data Comparisons"
Given a list *x* of *n* signed integers, design, run, and submit a MIPS assembly language program that finds the largest and smallest elements in the list (two answers, *max* and *min*) while performing no more than 3*n*/2 data comparisons (comparisons used for program control flow such as loops don't count towards this total) and using no memory locations other than those for the input list *x*, the length *n*, and the answers *max* and *min*.
Remember that *n* can be even or odd; the 3*n*/2 upper bound on the number of data comparisons should be honored in both cases.
Note that performing the task in 2*n* – 2 data comparisons is conceptually simpler: we either do two passes through the list *x* (one for *max* and one for *min*) or else we do one pass, keeping track of the largest and smallest values found thus far, which requires two comparisons for each of the remaining *n* – 1 elements after setting both *min* and *max* to *x*[0].

Project logistics
Deliverables: Project report (PDF FILE ONLY) and assembly code (.s file)
Submission: Attach report and project code to an e-mail and send to ece154aprojects@gmail.com
Files (.pdf and .s) must be named [last name]_[first name]_[perm no].[file extension]
The project report must contain the following:
Name and Perm Number
Pseudocode for the algorithm used to find *max* and *min*
English description of your algorithm, explaining how no more than 3*n*/2 comparisons are used and the memory constraints are met
An appendix with the assembly language code (in addition to attaching the actual .s file in the e-mail)
Report must be in PDF format or points will be deducted!

Assembly code requirements
Your program must:
Run in QtSPIM without parsing errors.
Use only the MicroMIPS instructions in Table 13.1 (p. 244) of the textbook and the la (load address) instruction.
Assume input array in the .data section with label X, and the number of elements in list X given by the word stored at label N. An example is provided below. We will replace these lines of code when testing your program, so please use these exact label names (including keeping the same lower/upper cases). Your algorithm should be able to handle lists of different sizes.
Use the print int system call to print min, and then print max on a new line to the console
For the MIPS simulator to run properly, your program must end with an exit system call.

Sample assembly file
```
.data
N: .word 5 # number of elements in the array
X: .word 1,-4,7,6,2 # array of numbers in which to find min and max
MIN: .word 0 # storage for min value
MAX: .word 0 # storage for max value
.globl main
.text
main: # start writing your code here
...
```

Project 2: Program instrumentation and performance (ch. 13-15, due F 12/05, 9:00 AM)

Important note
Please start working on the project right away, as it is more involved than Project 1. Only the last two parts are directly related to Chapters 13-15, which we have not covered yet.

Project description
"Modular exponentiation with *k*-bit operands using no more than 2*k* multiplications or divisions"
This project involves writing a MicroMIPS program for modular exponentiation, *c* = (*a*^*b*) mod *p* (where all variables are 16-bit unsigned integers that are represented in 32-bit unsigned format), and instrumenting the program to derive the instruction frequencies and the associated CPIs for multicycle and simple-pipelined

MicroMIPS implementations.
a. [10%] Research and describe in pseudocode an exponentiation algorithm that requires at most 30 multiplications, where 30 = 2k – 2 and k = 16 in our case. Present a written argument for why it does not need more than 30 multiplications. For this part, do not worry about the size of the integers obtained and whether they would fit in a machine word. [*Hint*: The process is identical to Horner's method, with doublings and additions replaced by squarings and multiplications.]
b. [10%] Modify the algorithm of part a so that it performs modulo-p exponentiation, without ever handling a value that does not fit in a 32-bit word. You are allowed to use up to 30 remainder calculations, which are essentially division operations. Again, provide pseudocode and argue that the number of divisions is in fact no more than 30.
c. [20%] Modify the multiplication procedure of Example 11.4 in the textbook so that it uses only the 22 MicroMIPS instructions of Table 13.1, plus "la" (load address), sll (logical left shift), and srl (logical right shift), and performs modular multiplication $z = xy \bmod p$, with the result ending up in $v1 (which is used to hold the value in the MiniMIPS register Lo). The parameter p is passed to the procedure in register $a2. Call the new procedure ummu, for unsigned modular multiplication [*Hint*: Perform modular reduction after each accumulation step; leaving the modular reduction to the end will necessitate the use of division, as in part b, which we don't want.]
d. [20%] Write a MicroMIPS assembly language program to implement the computation of part b, using the procedure of part c and limiting yourself to the same instructions specified in part c. Run your program on at least 10 different test triples (a, b, p), the first five of which are specified under "Sample assembly file" below. The complete program file for this part will be one of your project's deliverables. Using the naming convention of Project 1, call the .s file [last name]_[first name]_[perm no].s
e. [20%] By inserting monitoring instructions (the fewer, the better) at appropriate locations, count how many instructions of each kind are executed during your runs. Report the counts in a table within your report, with rows corresponding to the 22 instructions of Table 13.1 and columns representing your test runs.
f. [10%] From the results of part e, calculate the average CPI for each column of the table, and overall, given a multicycle MicroMIPS implementation, using the cycle counts of Section 14.4. The monitoring instructions themselves do not count in the tally.
g. [10%] From the results of part e, calculate the average CPI for each column of the table, and overall, given a pipelined MicroMIPS implementation as in Fig. 15.9, assuming data forwarding and one cycle waste after every branch instructions (ignore all other waste due to pipeline bubbles).

Project logistics
Deliverables: Project report (PDF FILE ONLY) and assembly code (.s file)
Submission: Attach report and project code to an e-mail and send to ece154aprojects@gmail.com
Files (.pdf and .s) must be named [last name]_[first name]_[perm no].[file extension]
The project report must contain the following:
Name and Perm Number
Sections corresponding to the 7 project parts, with each containing algorithm description in English, pseudocode, and other details, as appropriate.
An appendix with the assembly language code (in addition to attaching the actual .s file in the e-mail)
Report must be in PDF format or points will be deducted!

Assembly code requirements
Your program must:
Run in QtSPIM without parsing errors.
Use only the MicroMIPS instructions in Table 13.1 (p. 244) of the textbook and the la (load address) instruction. [*Important note*: Please do not use lw instructions of the form "lw $t1,N($zero)"; these proved a source of compatibility-related problems and program crashes in Project 1. Instead, place N's address in a register using "la $t2,N" and then complete the load operation with "lw $t1,0($t2)"; similarly, instead of "sw $t0, MAX" use "la $t3, MAX" followed by "sw $t0, 0($t3)".]
Assume arrays in the .data section with labels A, B, P, and C, and the number of elements in the lists given by the word stored at label N. An example is provided below. We will replace these lines of code when testing your program, so please use these exact label names (including keeping the same lower/upper cases). Your algorithm should be able to handle lists of different sizes.
Use the print int system call to print A, B, P, C (comma-separated) on a new line to the console for each test case.
For the MIPS simulator to run properly, your program must end with an exit system call.

Sample assembly file

```
.data
N: .word ... # number of test cases supplied; at least 10
A: .word 5,25,9,12345,54321,... # parameter A in (A^B) mod P; add 5 or more test cases
B: .word 0,16,8,54321,10000,... # parameter B in (A^B) mod P; add 5 or more test cases
P: .word 8,13,2,10000,65535,... # parameter P in (A^B) mod P; add 5 or more test cases
C: .word 1,1,1,1,1,... # results of test cases
.globl main
.text
main: # start writing your code here
...
```

Hints on modular program development and testing

When faced with the development of a program that has multiple functional parts, you should try to develop and test each part separately to reduce complexity and to allow easier pinpointing of problem areas. For example, if in this project you face difficulties with developing the modular multiplication procedure according to the specs, develop a simpler procedure that does the same job (say use multu and divu instructions) so that you can focus on the rest of the parts. Eventually, you will be able to replace the placeholder procedure with your final version that meets the project specs.

Sample Exams and Study Guide

The following sample exams are meant to indicate the types and levels of problems, rather than the coverage (which is outlined in the course calendar). Students must study any section or topic that is not specifically excluded in the study guide that follows the sample exams, even if the material was not covered in class lectures. The final exam will also cover the midterm material, but to a lesser degree.

Sample Midterm 1

Problem 1 [15 points]: Defining concepts and terms -- Define each of the following concepts/terms precisely and concisely within the space provided (about 1.5 vertical inches per term) [3 points each]: Decoder; Shift register; PC-relative addressing; Pseudoinstruction; Carry-skip adder.

Problem 2 [20 points]: Problem 4.4 in the textbook.

Problem 3 [20 points]: Can you replace the following sequence of MiniMIPS instructions with fewer instructions, without changing the functionality? Explain your answer fully. Assume that the values computed in $t0 and $t1 are temporaries that are not needed elsewhere in the program (Table 6.2 will be provided as reference with problems such as this one and the next one):

```
and $t0,$s0,$s1
or $t1,$s1,$s0
beq $t0,$t1,label
```

Problem 4 [20 points]: Problem 7.3ad in the textbook.

Problem 5 [25 points]: In Fig. 10.19 (provided), explain each of the following:

a. Why the overflow signal is formed by an XOR gate.

b. The role of the *k* XOR gates on the input side of the adder.

c. The total number of control signals supplied to the ALU (show how computed).

d. Why the shifter and logic unit are likely to be faster than the adder.

Sample Midterm 2 (not appliable to fall 2014)

Problem 1 [20 points]: Defining concepts and terms -- Define each of the following concepts/terms precisely and concisely within the space provided (about 1.5 vertical inches per term) [4 points each]: Data forwarding; Loop unrolling; Microprogramming, Optimal pipelining; Pipeline data dependency.

Problem 2 [20 points]: Problem 13.1a in the textbook.

Problem 3 [20 points]: Problem 14.12a in the textbook.

Problem 4 [20 points]: Problem 15.2 in the textbook.

Problem 5 [20 points]: Problem 16.2 in the textbook.

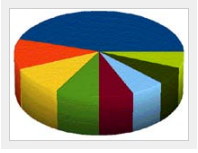
Sample Final Exam

Problem 1 [10 points]: Defining concepts and terms -- Define each of the following concepts/terms precisely and

concisely within the space provided (about 1.5 inches per term) [2 points each]: Pipelined control; Conflict miss; Delayed branch; Set-associative cache; TLB
Problem 2 [15 points]: Problem 4.7 in the textbook.
Problem 3 [15 points]: Problem 6.16ab in the textbook (instruction tables are provided).
Problem 4 [15 points]: Problem 11.5 in the textbook.
Problem 5 [15 points]: Problem 14.2c in the textbook.
Problem 6 [15 points]: Problem 16.9ab in the textbook.
Problem 7 [15 points]: Problem 18.3c in the textbook.

Midterm and Final Exam Study Guide
The following includes topics that will be emphasized in the course exams, as well as excluded topics.
[Chapters 1-3] No direct problem or question, but you need to know (and be able to define) concepts such as tristate buffers, multiplexers, register files, and so on, used to explain the topics that follow.
[Chapter 4] Computer performance: Problem likely on CPI calculation, performance enhancement (Amdahl's law), instruction mix, and/or benchmarks.
[Chapters 5-8] Instruction-set architecture: You do not need to memorize instruction codes or formats. Any problem in this area will be accompanied by a reference table providing a list of codes and formats if required. Ignore Sections 7.5, 7.6, 8.4, and 8.6.
[Chapters 9-12] Computer arithmetic: Problem likely on 2's-complement numbers, number radix conversion, floating-point number formats, shift/logical operations (including distinction between arithmetic and logical shifts), adders and ALUs, basics of multipliers. Ignore Section 12.6.
[Chapters 13-14] Data path and control: Problem very likely on control unit structure, control signal generation, multicycle instruction execution, and control state machine.
[Chapter 15-16] Pipelining: Problem very likely on pipeline bubbles (how to insert or avoid them), pipeline control, data hazards, data forwarding, control hazards, delayed branch, and/or branch prediction.
[Chapters 17-19] Memory system: Problem very likely on the need for memory hierarchy, cache memory concepts (levels 1 and 2), miss/hit rate, average memory access time, compulsory/capacity/conflict misses, mapping schemes. Sections 17.5, 19.5, and 19.6 are excluded.
[Chapters 20-28] Virtual memory, I/O, interfacing, Advanced architectures: No problem or question.

Grade Statistics



Grades listed are in percent, unless otherwise noted.
Homework 1 grades: Range = [53, 100], Mean = 89, SD = 9, Median = 92
Homework 2 grades: Range = [46, 100], Mean = 75, SD = 15, Median = 76
Homework 3 grades: Range = [22, 100], Mean = 79, SD = 16, Median = 83
Project 1 grades: Range = [30, 100], Mean = 90, SD = 16, Median = 96
Project 2 grades: Range = [5, 96], Mean = 41, SD = 27, Median = 31
Midterm exam grades: Range = [33, 93], Mean = 66, SD = 13, Median = 70
Final exam grades: Range = [18, 87], Mean = 50, SD = 13, Median = 50

References

Required text: B. Parhami, *Computer Architecture: From Microprocessors to Supercomputers*, Oxford University Press, 2005. **Textbook's web page** contains an errata and lecture slides.
Recommended book (not required): R. L. Britton, *MIPS Assembly Language Programming*, Pearson, 2004. [There exist free on-line MIPS reference books that are adequate.]
Useful references (not required): D. A. Patterson and J. L. Hennessy, *Computer Organization & Design: The Hardware/Software Interface*, Morgan Kaufmann, 4th ed., 2008. W. Stallings, *Computer Organization and Architecture*, Prentice Hall, 8th ed., 2010.

Miscellaneous Information

Motivation: Computer architecture is the study/specification of computer systems at the interface of hardware and software. Computer architecture is driven from the software side by user needs in terms of functions and speed and from the hardware side by technological innovations and constraints. ECE 154A introduces you to this exciting field and makes you an informed computer user who understands basic architectural features as well as their cost/performance implications. The programmer's view of the instruction set and user interface are considered along with memory organization, addressing methods, and performance parameters. By taking ECE 154B, you will learn additional topics such as input/output, buses, interfacing, and a multitude of performance

issues and computation speedup methods. ECE 154A/B prepare you for participation in computer design efforts and for learning the advanced implementation methods and technologies used in high-performance uniprocessors (ECE 254A), parallel processors (ECE 254B), and distributed systems (ECE 254C).

Catalog entry: **ECE 154A. Introduction to Computer Architecture.** (4) PARHAMI.
Prerequisite: *ECE 152A with a minimum grade of C-; open to EE and CMPEN majors only.*
Enrollment Comments: *Not open for credit to students who have completed Computer Science 154. ECE 154A is the formerly numbered ECE 154. Students who have taken ECE 154 and have received a grade of C- or lower may take ECE 154A for a better grade.*
Repeat Comments: *Students who have taken ECE 154 and have received a grade of C- or lower may take ECE 154A for a better grade.*
Instruction-set architecture (ISA) and computer performance; Machine instructions, assembly, addressing modes; Memory map, arrays, pointers; Procedure calls; Number formats; Simple ALUs; Data path, control, microprogram; Buses, I/O programming, interrupts; Pipelined data paths and control schemes.

History: Computer architecture is a required subject in the Computer Engineering Program, and it can be taken as an optional subject by students in certain other majors. In 2013, the course ECE 154 (Introduction to Computer Architecture) was merged with the discontinued ECE 15B to form the ECE 154A/B sequence, dubbed introductory and advanced computer architecture. The computer architecture course offered by the Computer Science Department, CMPSC 154, was equivalent to ECE 154 many years ago, but the two courses have diverged in content to serve the needs of different curriula and student populations. The following record of previous offerings includes only those taught by Professor Parhami.
Offering of ECE 154 in winter 2011 (PDF file)
Offering of ECE 154 in winter 2010 (PDF file)
Offering of ECE 154 in winter 2009 (PDF file)
Offerings of ECE 154 from 2000 to 2008 (PDF file)