5DV008 Computer Architecture Fall 2010

1 Course Staff

Main Instructor:	Name:	Stephen J. Hegner
	Office:	C444 MIT-huset
	Telephone:	090 786 5760
	E-mail:	`hegner@cs.umu.se`
	URL:	`http://www.cs.umu.se/~hegner`
	Office hours:	1015-1100 Monday and Thursday on days with a lecture
Assistant:	Name:	Marcus Karlsson
	Office:	C420 MIT-huset
	Telephone:	090 786 6372
	E-mail:	`marcusk@cs.umu.se`
	URL:	`http://www.cs.umu.se/~marcusk`
	Office hours:	1300-1430 Monday and Thursday

- Stephen J. Hegner is available to answer questions about the lectures and the course material in general, but cannot answer detailed questions about what is or is not acceptable as an answer to an obligatory exercise.
- Questions regarding the exercises should be directed to Marcus Karlsson.

2 Course Language

All lectures will be given in English. However, questions may be asked during the lectures in either English or Swedish, and written work may be submitted in either English or Swedish. The questions on final examination will be written in English; answers may be in either English or Swedish. For the final examination, it will be permitted to use an XX-English / English-XX dictionary, where XX is a natural language of the student's choice.

3 Course Literature

The official textbook for this offering of the course is the following.

1. David Patterson and John Hennessy, *Computer Organization and Design, The Hardware/Software Interface*, Fourth Edition, 2008, Morgan Kaufmann, ISBN-13: 978-0-12-374493-7.

The third edition differs substantially from the fourth and is not recommended as a substitute.

Version: February 04, 2011

In addition to the course text, there will be relatively detailed overhead slides, based largely upon those used in courses given by Professor Mary Jane Irwin at Pennsylvania State University, who has graciously granted permission to use those materials. They will be available for download on the course web page.

4 Course Content and Outline

The official kursplan (in Swedish) and an abbreviated course description (in English) is available by searching for `5DV008` on this link. A more offering-specific outline is shown below. The numbers shown in the single rectangular brackets (i.e., [..]) identify chapters in the textbook. The numbers in angle brackets ⟨..⟩ indicate the approximate number of 45-minute lecture periods which will be devoted to the topic.

- Reasonably detailed overhead slides will be available for many topics. The authoritative source for relevant (i.e., possible examination) material is the course lectures and these slides. In many cases, material not covered in the textbook may nonetheless be covered in lecture presentations.
- The number of 45-minute lecture "hours" to be devoted to each topic is approximate, and in particular is rounded to the nearest integer. Adjustments will be made as the course progresses, and so the table below should not be used a definitive guide to which topics will be covered on which days.

1 Introduction [1] ⟨2⟩

2 The MIPS Architecture [2] ⟨7⟩

3 Arithmetic [3] ⟨4⟩

4 Processor Architecture [4.1-4.13] ⟨8⟩

5 Memory [5] ⟨4⟩

6 Secondary Storage [6] ⟨2⟩

7 Multi-Stuff and Parallelism [7] ⟨1⟩

8 Review ⟨2⟩

4.1 Online Materials

There web site for the course is located at

http://www.cs.umu.se/kurser/5DV008/H10/index.html.

The following materials may be found on these pages.

1. This syllabus, in both PDF and HTML.
2. The lecture slides for the course.
3. Descriptions of the obligatory exercises.
4. Information on auxiliary software, such as the Spim assembler and emulator.
5. Sample programs.
6. Miscellaneous links to architecture-related things.
7. Some official documents required by the Department of Computing Science.

5 Course Materials Outline

The following is a list of those chapters and sections which will be covered in the course. For each chapter or section, a symbol is given which indicates the nature of coverage in the course. The meaning of these symbols is provided in the table below.

✔	Material will be covered in the course.
➲	Overview material, covered in concept but not detail
✘	Material will not be covered in the course.
✙	Review material, prior knowledge is expected.
✣	Material will be covered partially or selectively.
??	Coverage extent to be announced later.

Note the following:

- If an entire chapter is covered, no section-by-section breakdown is given.
- Entries have not been provided for sections entitled "Summary" or the like.
- In general, omitted items will not be covered. However, the possibility that some covered material may appear in an omitted chapter or section remains. In all cases, the lectures and course slides should be taken to be the definitive guides to the course material.

- Sections entitled "Concluding Remarks" and "Historical Perspective and Further Reading" are not listed. They are all classified as overview material.
- Sections entitled "Exercises" are not listed.

1 Computer Abstractions and Technology

- 1.1 Introduction ➲
- 1.2 Below Your Program ➲
- 1.3 Under the Covers ➲
- 1.4 Performance ✔
- 1.5 The Power Wall ➲
- 1.6 The Sea Change ✔
- 1.7 Real Stuff ➲
- 1.8 Fallacies and Pitfalls ✔
- 1.9 Concluding Remarks ➲

2 Instructions: Language of the Computer

- 2.1 Introduction ➲
- 2.2 Operations of the Computer Hardware ✔
- 2.3 Operands of the Computer Hardware ✔
- 2.4 Signed and Unsigned Numbers ✔
- 2.5 Representing Instructions in the Computer ✔
- 2.6 Logical Operations ✔
- 2.7 Instructions for Making Decisions ✔
- 2.8 Supporting Procedures in Computer Hardware ✔
- 2.9 Communicating with People ✔
- 2.10 MIPS Addressing for 32-Bit Immediates and Addresses ✔
- 2.11 Parallelism and Instructions: Synchronization ✔
- 2.12 Translating and Starting a Program ✔
- 2.13 A C Sort Example to Put It All Together ✔
- 2.14 Arrays vs. Pointers ✔
- 2.15 Advanced Material: Compiling C and Interpreting Java ✘
- 2.16 Real Stuff: ARM Instructions ➲

- 2.17 Real Stuff: ARM Instructions ✘
- 2.18 Fallacies and Pitfalls ➲

3 Arithmetic for Computers

- 3.1 Introduction ➲
- 3.2 Addition and Subtraction ✔
- 3.3 Multiplication ✔
- 3.4 Division ✔
- 3.5 Floating Point ✣
- 3.6 Parallelism and Computer Arithmetic: Associativity ✔
- 3.7 Real Stuff: Floating Point in the x86 ✘
- 3.8 Fallacies and Pitfalls ➲
- 3.9 Concluding Remarks ➲

4 The Processor

- 4.1 Introduction ✔
- 4.2 Logic Design Conventions ✔
- 4.3 Building a Datapath ✔
- 4.4 A Simple Implementation Scheme ✔
- 4.5 An Overview of Pipelining ✔
- 4.6 Pipelined Datapath and Control ✔
- 4.7 Data Hazards: Forwarding versus Stalling ✔
- 4.8 Control Hazards ✔
- 4.9 Exceptions ➲
- 4.10 Parallelism and Advanced Instruction-Level Parallelism ✣
- 4.11 Real Stuff ✘
- 4.12 Advanced Topic ✘
- 4.13 Fallacies ✔

5 Large and Fast: Exploiting Memory Hierarchy

- 5.1 Introduction ✔
- 5.2 The Basics of Caches ✔

- 5.3 Measuring and Improving Cache Performance ✔
- 5.4 Virtual Memory ✔
- 5.5 A Common Framework for Memory Hierarchies ✔
- 5.6 Virtual Machines ➲
- 5.7 Using a Finite-State Machine to Control a Simple Cache ✣
- 5.8 Parallelism and Memory Hierarchies: Cache Coherence ✣
- 5.9 Advanced Material: Implementing Cache Controllers ✘
- 5.10 Real Stuff ➲
- 5.11 Fallacies and Pitfalls ✔

6 Storage and Other I/O Topics

- 6.1 Introduction ✔
- 6.2 Dependability, Reliability, and Availability ✔
- 6.3 Disk Storage ✔
- 6.4 Flash Storage ✔
- 6.5 Connecting Processors, Memory, and I/0 Devices ✔
- 6.6 Interfacing I/O Devices to the Processor, Memory and Operating System ✔
- 6.7 I/O Performance Measures: Examples from Disk and File Systems ➲
- 6.8 Designing an I/O System ➲
- 6.9 Parallelism and I/O: Redundant Arrays of Independent Disks ✔
- 6.10 Real Stuff ✘
- 6.11 Advanced Topics: Networks ✘
- 6.12 Fallacies and Pitfalls ✔

7 Multicores, Multiprocessors, and Clusters

- 7.1 Introduction ✔
- 7.2 The Difficulty of Creating Parallel Processing Programs ✔
- 7.3 Shared Memory Multiprocessors ➲
- 7.4 Clusters and Other Message-Passing Multiprocessors ➲
- 7.5 Hardware Multithreading ➲
- 7.6 SISD, MIMD, SIMD, SPMD, and Vector ✔
- 7.7 Introduction to Graphics Processing Units ➲

7.8 Introduction to Multiprocessor Network Topologies ➲

7.9 Multiprocessor Benchmarks ➲

7.10 Roofline: A Simple Performance Model ➲

7.11 Real Stuff ✘

7.12 Fallacies and Pitfalls ✔

A Graphics and Computing CPUs ✘

B Assemblers, Linkers, and the SPIM Simulator ✣

C The Basics of Logic Design (on CD) ✙

D Mapping Control to Hardware (on CD) ✘

E A Survey or RISC Architectures for Desktop, Server, and Embedded Computers (on CD) ✘

6 Laboratory Schedule and Computer Resources

There is no official laboratory booking for the course, nor any in-laboratory instruction. In general, when not reserved by a course, the computer laboratories of the department are open for use by students for their coursework.

The assembler and simulator *Spim* (command-line version), *XSpim* (X-windows version), and *PCSpim* (Microsoft Windows version) is available on the departmental computers for the purposes of learning the MIPS instruction set and testing the correctness of the programming exercises. However, the programming exercises themselves will be carried out using standard programming languages such as `C` or `Java`, and the final, submitted versions must run under the Unix/Linux environment of the department.

7 Course Schedule

The table below identifies the course meeting times and places, together with the nature of the meeting. The key "L" denotes a lecture, "R" denotes a review session, while "E" denotes an examination booking.

For each lecture, the topics to be covered are identified via the outline header number of Section 4 of this syllabus. So, for example, on November 30 the topics of 5, Pipelining, will be covered. This is only an approximate assignment of meeting times to topics, and it may be altered as the course progresses.

Rooms whose identifiers begin with the letter *M* are located in MIT-huset; room B203 is in Humanisthuset.

Week	Type	Date	Time	Room	Topics
45	L	Nov 08	0815-1000	MC313	1
45	L	Nov 10	0815-1000	MC313	2
45	L	Nov 11	0815-1000	MC313	2
46	L	Nov 15	0815-1000	MC313	2
46	L	Nov 17	1015-1200	MC313	2,3
46	L	Nov 18	0815-1000	MC313	3
47	L	Nov 22	0815-1000	MC313	3,4
47	L	Nov 24	0815-1000	MC313	4
47	L	Nov 25	0815-1000	MC313	4
48	L	Nov 29	0815-1000	MC313	4
48	L	Dec 01	1015-1200	B203	4,5
48	L	Dec 02	0815-1000	MC313	5
49	L	Dec 06	0815-1000	MC313	5,6
49	L	Dec 08	0815-1000	MC413	6,7
49	R	Dec 09	0815-1000	MC313	8
02	E	Jan 14	0900-1500	IKSU	Final examination
17	E	Apr 29	0900-1500	Skrivsal 1	Final examination
23	E	Jun 09	0900-1500	Skrivsal 7	Final examination

8 Prerequisites

The formal requirements are listed in the course plan, which may be found by searching for 5DV008 at the following at the following link. They include the following.

1. A knowledge of programming in C in the Unix/Linux environment. This requirement is met by the formal prerequisite of the course *Systemprogrammering* (Systems Programming).

2. A thorough knowledge of data structures and algorithms, as presented in the course *Datastrukturer och algoritmer* (Data Structures and Algorithms), which is a prerequisite for the course in systems programming.

3. A knowledge of discrete mathematics and the formal foundations of computer science. This requirement is met by the course *Diskret matematik* (Discrete Mathematics).

4. One of the courses *Envariabelanalys 1* (Calculus of a single variable) or the course *Grundläggande analys* (Fundamentals of analysis) is also required formally, but is not really essential for an understanding of the course material.

5. A broad background in computer science consisting of at least 60 credits.

These requirements should be met by students who are following a normal path of study in one of the programs of the Department of Computing Science. However, students from other disciplines who are considering this course should understand that the level of sophistication required in these topics is relatively high, and often not met by persons who work in other disciplines, even if they have a fair amount of practical programming experience.

Some background in digital design (as presented in the course *Digitalteknik*) will also be very helpful. Students who have not taken this course should read Appendix C of the book (which is on the accompanying CD).

9 Grading System

This course has two parts (*moment* in Swedish), a conceptual part (*teoridelen* in Swedish and an exercise part (*laborationsmoment* in Swedish).

The only possible grades for the exercise part are S (Satisfactory; G=*Godkänd* in Swedish) and U (Unsatisfactory, *Underkänd* in Swedish). The grade on this part will be determined entirely by two obligatory programming projects. Each project will be graded as S (Satisfactory) or U (Unsatisfactory). To obtain the grade of S for the exercise part of the course, it is both necessary and sufficient to obtain the grade of S on both projects. In addition, for each obligatory project it will be possible to earn a maximum of 75 quality points. These points will be assigned based upon the overall quality and correctness of the work. Fifty of these points will be based upon the quality of the basic parts of the project, with an additional 25 bonus points possible for implementing extra features. Thus, a maximum of 150 points may be earned on the two obligatory projects.

There will also be two non-obligatory problem exercises, which will provide practice in solving typical problems on such topics as pipeline and cache analysis. Each will have a maximum point total of 50, so a maximum of 100 points may be eraned on the two non-obligtory exercises.

The examination will have a total of 1000 points.

The final point total F for the course is computed as

$$F = \max(E, \quad 0.8 \times E + (L + P))$$

with E the number points earned on the examination, L the number of points earned on the laboratory exercises, and P the number of points on the non-obligatory exercises. The final grade on the conceptual part of the course is computed as follows.

Number of points	Grade
$F \geq 800$	5 (med beröm godkänd – excellent)
$650 \leq F < 800$	4 (icke utan beröm godkänd – very good)
$500 \leq F < 650$	3 (godkänd – satisfactory)
$F < 500$	U (underkänd – unsatisfactory)

In addition, to pass the course, a minimum of 500 points on the examination is necessary, regardless of how many points are earned on the exercises. Thus, exercise points can only be used to increase the grade from 3 to 4, or from 4 to 5. They cannot be used to rescue a performance of less than 50% on the examination.

Students who completed the exercise part of the course in a previous year may nevertheless submit exercises for points.

10 Obligatory Work

10.1 General Remarks on the Obligatory Projects

- The exercises may be completed in groups, and collaboration is permitted on the software exercises, roughly as described in the documents *Riktlinjer vid labgenomförande (Policy for Obligatory Exercises)* and *Hederskodex (Honor Code)*. More details will be provided later, when the descriptions of these exercises are distributed.

- The exercises may be submitted individually, or two or three persons may submit one solution. However, once a solution is submitted, only those named on the submission will receive credit for it. Partners in solution may not be added after the initial submission.

- Each exercise will have a submission deadline. For late submissions, the number of points awarded will be $(1-0.1\times p)$, where p is the number of working days which the submission is late. Thus, a submission which is more than two weeks late cannot receive any points at all.

- All obligatory exercises must be submitted for grading on or before the third and final examination (during June, 2011). No exercises will be accepted after that date. After that date, the student must re-register for the next offering of the course and complete the exercise part according to the rules for that part.

10.2 Obligatory Work Completed in Previous Years

- Credit for individual exercises may not be carried over from previous years. A student who does not already have a satisfactory grade recorded for the exercise part of the course must complete all requirements for that part as defined by this offering of the course.

11 Non-Obligatory Exercises

More information on these exercises will be provided when they are distributed. However, the the deadlines will be reasonably strict and late submissions risk not be accepted.