A Proposal of Computer Architecture Courses for the Computer Engineering Curricula of the Polytechnic University of Catalonia[1]

Miguel Valero-García
Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya
c/ Jordi Girona 1-3, Campus Nord - Edifici D6
E-08034 Barcelona (Spain)
Phone: +34-3-401 6995
E-mail: {miguel}@ac.upc.es

1 Introduction

The Computer Architecture Department (DAC) has been teaching courses on computer architecture at the School of Informatics (FIB) since 1976. In 1991, in the framework of the University Reform Law, the FIB opened a new degree in Informatics Engineering and two Diplomas in Computer Software and Computer Systems, which replaced the old Computer Science Degree. Important changes were introduced in the three new degrees, not only in the contents but also in the organization and course development.

Now, the FIB is asked to review the new degrees and to introduce some minor technical changes to fulfil the current national laws. It was clear very soon that this was a good opportunity to undertake a deeper review of contents and teaching methodologies, in the light of the first six years of life of the new degrees. Therefore, the different departments are currently involved in such a review. The discussions and debates should conclude in the near future with a proposal of changes. The main objective of this paper is to present the current state of the debate in the computer architecture group, first providing the context for the debate (section 2), and then describing the most important ideas (section 3) and emphasizing the points that are originating the most interesting discussions (section 4).

1. A slightly version of this paper has been published in IEEE CATC Newsletter, february 1999

2 The Context

2.1 The old degree

The Computer Science Degree (the old degree) was opened in 1976. This 5-year degree had an annual structure. Each academic term had a duration of 25 weeks, from October to June. The degree had 56 different courses covering: algorithms and data structures, software engineering, computer architecture, mathematics, statistics, simulation, operation research, digital systems, computer control, economics and management of organizations. Students had to take 11 compulsory courses and 18 optional courses.

The computer architecture material was covered in three courses:

Computers (first year). The basics of digital systems. Introduction to machine/assembly language.

Computer Structure (second year). The machine language level. Input/output. Microprogramming. Cache memories.

Computer Architecture (fourth year). Arithmetic processors. Pipelined processors. Multiprocessor systems.

Each course had 4 hours of lectures per week (mostly theory lectures). Most of the courses had an open laboratory where students had to develop project practicals, in groups, under the guidance of the lecturer, but usually without a clearly stated timetable (except for final delivering dates). As an example, in the *Computer Structure* course, students had to develop some code in assembly language to control the input/output devices of a PC computer.

2.2 The new degrees

The FIB was pioneering in opening, in 1991, the new degree in Informatics Engineering (EI) and the two Diplomas in Computer Software (ETG) and Computer Systems (ETS), in the framework of the Spanish University Reform Law.

One of the objectives of the reform was to update the contents of the curricula. In addition, the curricula adopted a semester structure (which is the standard in the european context) and some changes in the teaching methodology were introduced. In particular, the courses in the new degrees have four types of activities: theory lectures, applicative seminars, laboratory practicals and project practicals. The groups of students for theory lectures may be large (up to 100 students). The groups for applicative seminars and laboratory practicals are smaller, since the objective is to facilitate the interaction between the student and the lecturer. Finally, project practicals in open labs follow the same methodology as in the old degree.

The three new degrees share most of the courses. In particular, the first year (which is called the Selection Phase) is common to EI, ETG and ETS. When the students pass the Selection Phase then they chose the curriculum they want to follow. Many students (close to 50%) are unable to pass the Selection Phase and cannot continue their studies in the FIB.

After the Selection Phase, there are still many courses that are shared among the three curricula. In fact, the Diplomas ETG and ETS are two different subsets of EI plus a small number of specific courses.

The courses of the new degrees dealing with computer architecture are:

> *Introduction to Computers (semester 1, 9 credits*[1]*).* Basics of logic design. Basic processor organization and machine language.
>
> *Computer Structure I (semester 2, 7.5 credits).* i80X86 assembly programming. Basics of input/output. Hardware support for memory management.
>
> *Computer Structure II (semester 3, 4.5 credits).* Processor-memory interconnection. Microprogramming. Cache memories. Machine language level design.
>
> *Computer Architecture (semester 5, 9 credits).* Arithmetic algorithms and processors. Pipelined processors.
>
> *Vector Computers (6 credits).* Structure of a vector computer. Machine language. Vectorization. Programming and performance.
>
> *Multiprocessors (6 credits).* Types of multiprocessors. Programming languages. Loop scheduling. Small scale multiprocessors. Compilation. Scalable multiprocessors.

1. One credit corresponds to 10 class hours

Supercomputing (6 credits). Supercomputers. ILP processors. Instruction scheduling. Cache memories. Prefetch. Programming models. Parallelising compilers.

Specialized Architectures (4.5 credits). Systolic architectures. Communication processors. Data flow architectures. Neural networks. Graphic processors.

Architectures for Symbolic Processing (4.5 credits). Sequential and parallel LISP machines. Sequential and parallel Prolog machines.

Algorithms for Parallel Architectures (6 credits). Basic numeric algorithms. Shared memory algorithms. Distributed memory algorithms. Parallel benchmarks.

The first four courses are compulsory for the EI curriculum. The rest are optional and students can take them in any semester, once they have passed the *Computer Architecture* course. *Introduction to Computers* and *Computer Structure I* are also compulsory for ETG and ETS currucula while *Computer Structure II* is compulsory for ETS but only optional for ETG. The optional courses for ETS and ETG are a subset of the remaining courses in EI.

2.3 The review of the new degrees

The reviewing process for the new degrees was started in 1995. A first step was to identify the problems of the new curricula, in the light of the six first years of experience. There was an agreement that the major problems are the following:

The personal workload for students is too large, mainly due to an excessive amount of project practicals, which tend to keep busy students more time than expected.

Students must take a too large number of courses per semester (6 or 7) if they want to advance in the curriculum at the expected speed.

There is a large number of optional courses (close to 100), which makes very difficult the planning and organization of the academic term.

There is a lack of specificity in the ETG and ETS diplomas, which seem to be just half the way to the EI degree.

After identifying the problems, the second step was to establish a few general strategies to solve these problems. The proposals from the different departments should adhere as much as possible to these strategies to guarantee the coherence of the reviewed curricula. The most important strategies are:

Increase the number of credits of the courses (for instance, grouping current courses) so that students can advance at the expected speed by taking only 4 or 5 courses per semester (courses should have 7.5 or 9 credits).

To reduce and concentrate the project practicals in a limited number of courses (open lab courses). A regular course has theory lectures, applicative seminars and lab practicals, and the student workload is under control. In a open lab course students put into practice the concepts learned in one or several regular courses. There is still the danger that students spend more time than expected in project practicals, but this would only happen in a limited number of courses.

Reduce the amount of optional courses. In particular, eliminate those courses whose contents may be more appropriate for post graduate studies.

Reorganize the subjects so that students that finish the ETG and ETS diplomas have a more specific training for professional work on some aspects of computer engineering.

We have agreed that some of the general problems stated above clearly affect to the courses dealing to computer architecture. In particular, we have a large number of “small” optional courses, some of them too specific and possibly more adequate for post graduate studies. Moreover, our optional courses are not oriented enough to provide professional skills. On the other hand, the majority of our courses do not have project practicals. Therefore, excessive workload should not be a problems for students in our courses.

3 The proposal

Our group must present in the near future a proposal of reorganization for the courses on computer architecture, according to the strategies described above. This proposal is under development and this section describes its current state. Some of the aspects have not been discussed enough yet, so the proposal may suffer important changes.

The main idea is to organize the material into three blocs: *Basic Computer Architecture*, *Modern Computer Architecture* and *Supercomputing*. The block *Basic Computer Architecture* covers the material that must be well known by all graduates in any of the degrees. The block *Modern Computer Architecture* covers the material related to the organization and performance of the computers that the graduated students will likely use in their future jobs (workstations based on advanced microprocessors and possibly in a small scale multiprocessor organization). It is expected that graduated students understand how this type of machines work and are able to write codes that get reasonable performances. Finally, the block *Supercomputing* covers the aspects related to organization and programming of supercomputers. A small number of graduated students will use this class of machines in the future and should be able to understand and use them efficiently.

In the following section we give some details of the different courses that are included in each block. The level of description of the different courses is not uniform since, while some of them are close to the courses that we are currently teaching (particularly those in block *Basic Computer Architecture),* others are new and their contents are not clearly defined yet. All courses have 7.5 or 9 credits.

3.1 Basic Computer Architecture

The objective of this block is to provide a solid foundation of the basis of computer organization. This block has three courses: *Introduction to Computers, Computer Structure I* and *Computer Structure II*.

Introduction to Computers (semester 1)

The first part of this course covers the basics of logic design: gates, combinational logic, finite state machines, etc. The second part of the course shows how these blocks are used to build a simple processor, including the design of the control unit. This simple processor is used to describe how the computer executes a program stored in memory.

In the lab sessions students use a simulator which runs in a PC to design and test simple logic circuits.

Computer Structure I (semester 2)

The first part of this course describes the TASM assembler for the i80x86 processor. We consider only a small subset of the instructions (no more than 15), the complete set of addressing modes and the basic data types (natural, integer, character and floating point). The emphasis here is in presenting the machine/assembly language as a support for high level programming languages so that students understand what is happening in the computer when their programs are being executed. Therefore, most of the time is spent in describing the correspondence between typical high level contructions and data types, and their representation in machine/assembly language.

The second part of the course covers the basics of input/output: controllers, polling and interrupt mechanisms, and direct memory access. The input/output system is described from the machine language level. Students should learn to write simple assembly programs to read controller registers, to capture interrupts or to initiate a direct memory access transfer and wait for its completion. The hardware concepts related to input/output are not covered in this course but in the following *(Computer Structure II)*.

In the lab sessions students build simple programs in TASM assembler in a PC. Some sessions are devoted to the practice of addressing modes to access data structures and the subroutine mechanism. Some other sessions are devoted to the practice of polling and interrupt mechanisms.

Computer Structure II (semester 3)

This course focuses on the hardware aspects of basic computer organization. The first part covers the material related to the computer bus. The key concepts are: bus signals, bus types, bus addressing and data transfer, timing protocols, arbitration and interrupt mechanisms.

The second part of the course covers the memory subsystem. The key concepts are: cache memories and hardware support for memory management.

The course has a small number of lab sessions to show the impact of cache and TLB misses on the performance of programs.

3.2 Modern Computer Architecture

Workstations of today are very different from those of just one or two years ago because they use techniques that could be find in the past only in supercomputers or in mainframes. The majority of the graduated students will have to work with such a class of machines in their jobs. Therefore, they need a solid foundation on how the machine works and how the programs must be tuned to squeeze more performance from the hardware. The objective of this block is to provide such a solid foundation. The block has to courses: *Computer Architecture* and *Performance Tuning of Modern Workstations.*

Computer Architecture

This course focuses in the architecture of modern workstations. The course must cover the following topics:

RISC architecture: machine language level and implementation

Pipelined processors: data path, control, data hazards, branch hazards and exceptions

Superscalar organization

Small scale multiprocessor organization: cache coherence, bus snooping, synchronization

The course is developed mainly through theory lectures and applicative seminars. The books [3] and [5] contain material that is adequate for this course.

Performance Tuning of Modern Workstations

Students that take this course should learn to use efficiently a modern workstation. Therefore it should be a course with a large amount of lab sessions. The first part of the course covers single processor tuning. Students learn compiler techniques for instruction scheduling (loop unrolling, software pipelining) and cache performance (blocking, copying, prefetching), and also learn their limitations. Then they learn how and when these limitations can be overcome by an adequate reorganization of data structures and code. The book [1] may be useful for this part of the course.

The second part of the course covers multiprocessor tuning. It describes a basic shared memory programming model (loop parallelism, loop scheduling and synchronization primitives) and basic compiler techniques for automatic parallelization. Again the limitation of these techniques are shown and students learn to look for and exploit coarse grain parallelism in the codes and to reduce the parallel overheads (load imbalance, synchronization, false sharing, etc).

3.3 Supercomputing

A limited number of graduated students may have to work with supercomputers. The objective of this block is to cover the material related to the architecture and use of supercomputers. This material is organized into two courses: *Architectures for Supercomputing* and *Algorithms, Languages and Tools for Supercomputing.*

Architectures for Supercomputing

This course describes the architecture of current high performance computers. It should include the following topics:

Advanced processor architecture

Vector computing

Scalable multiprocessors

The course is developed mainly through theory lectures and applicative seminars. The book [3] may be useful here.

Algorithms, Languages and Tools for Supercomputing

Students that take this course should learn to use supercomputers efficiently. This is also a course with a large number of lab sessions. The course should cover:

Tools (programming models and languages, compilers, profiling tools, benchmarks, etc)

Steps in the design of parallel algorithms

Typical algorithmic schemes and data structures for parallel computation

The book [2] contains interesting material for this course.

4 Discussion

Our proposal tries to solve the problems that were identified in section 2. On the one hand, the number of courses has been reduced. On the other hand, the courses have been oriented to provide professional skills to students, both in the area of widely extended workstations and in the area of supercomputing. The block *Basic Computer Architecture* would be compulsory for EI and ETS. It would be also compulsory for ETG excepting the course *Computer Structure II*, that would be only optional since it covers hardware concepts. The block *Modern Computer Architecture* would be optional for ETS. The course *Computer Architecture* in this block would be compulsory for EI while the course *Performance Tuning of Modern Workstations* would be optional. Finally, the block *Supercomputing* would be optional for EI. To finish, we point out in the following the aspects of the proposal that have generated more controversy.

Regarding the course *Introduction to Computers*, the major controversy is on the level of complexity of the pedagogical processor to be used. Some time ago we used an extremely simple processor, whose complete design could be described in two pages. It was enough to introduce the concepts of machine instruction, machine program, etc. However the gap between such a simple processor and the i80x86 used in the next course was so large that students were not able to imagine how the i80x86 processor executes its instructions. Currently, we are discussing what elements should be included in the processor and what elements should be omitted in order not to complicate it more than necessary. Remember that *Introduction to Computers* is taken in the first semester. For this reason, we believe that a simple processor such the one described in [5] would be too complex for this course.

A second controversy point is the selection of the i80x86 processor to introduce the machine-assembly level in the course *Computer Structure I*. There are two features that make the i80x86 unsuitable as a pedagogical tool:

Its segmented model of memory organization.

The lack of orthogonality in the instruction set architecture.

From this point of view other real processors (MIPS, Alpha, etc) or a purpose-designed teaching processor (with its simulator) would be a better choice (see [4][6] for discussions on this topic).

However, the advantages of i80x86 are also clear:

> A PC is cheap and it its the equipment currently available in our labs because it is widely used in other courses.
>
> The PCs allow to write programs that access directly to the controller register, and that control the interrupt mechanism.
>
> The PCs are used in the labs of courses on operating systems and computer networks. Although most of the code developed in these labs is written in high level language, students should understand perfectly the i80x86 segmented model of memory organization.

The third controversy point is the separation of the material related to input/output into two parts (software and hardware viewpoints) to be covered in two different courses (*Computer Structure I* and *Computer Structure II)*. Some people argue that the software viewpoint cannot be assimilated without its hardware counterpart. However, we believe that it is good to concentrate the hardware aspects of basic computer architecture in a course that is only optional to the ETG degree. In addition, it is good to force the students to understand and use the machine at a certain level of abstraction without the knowledge of the implementation details. This is frequent and necessary in many disciplines of computer science.

Finally, another controversy point is on the contents of the course *Computer Architecture*. Some people argue that there are too many new concepts in the current proposal for this course and propose to move the material related to multiprocessor organization to the course *Architectures for Supercomputing.* In this way, *Computer Architecture* would be a course on single processor advanced architecture while *Architectures for Supercomputing* would be basically a course on parallel machines. In our opinion, small scale multiprocessor organization is something to be found in near future workstations. Therefore, the right place for this material is the *Modern Computer Architecture* block. Moreover, the material can be easily built upon the concepts of bus and cache organization, to be covered in detail in the previous course *Computer Structure II*.

References

[1] K. Dowd, High Performance Computing. *O'Reilly & Associates, Inc. 1993*

[2] I. Foster, Designing and Building Parallel Programs, *Addison -Wesley Publishing Company 1995*

[3] K. Hwang, Advanced Computer Architecture: Parallelism, Scalability, Programmability. *New York: McGraw-Hill, 1993.*

[4] M. C. Loui, The Case for Assembly Language Programming, *IEEE Trans. on Education*, *Vol. 31*, *No. 3, August 1988,* pp. 160-164.

[5] D.A. Patterson and J.L Hennessy, Computer Organization & Design: The Hardware/Software Interface, *Morgan Kaufmann Publishers, Inc., 1998*

[6] P.P. Silvester, Introducing Computer Structure by Machine Simulation, *IEEE Trans. on Education, Vol. 33, No. 4, November 1990*, pp. 326-332.