Learning to Design a Computer

Overview of Hardware Design Subjects and Educational Software at Faculty of Electrical Engineering in Belgrade

Časlav Božić
Department of Computer Engineering , Faculty of Electrical Engineering , University of Belgrade

Abstract

The objective of this paper and the presentation is to guide the audience through the process of teaching hardware related subjects on Department of Computer Engineering on Faculty of Electrical Engineering at University of Belgrade and to present special educational software developed for this purpose.

The overview of curriculum is given at the beginning, stating that one half of all classes are computer related and about one third of these are hardware related.

The educational software specially developed for teaching in subjects: "Introduction to Computer Engineering", "Computer Architecture" and "Computer Architecture and Organization" is presented in addition. Interesting is option of self-testing over the Internet before taking a test in the laboratory.

The next step in education is designing a microcomputer system and designing a VLSI device using commercial software ("Protel", "VHDL").

For conclusion some personal experiences from real-life project carried out for international CSIDC competition are presented, considering it needed synthesis of all acquired knowledge.

1. Introduction

Faculty of Electrical Engineering has been founded in 1948, and until now has issued over 13000 graduate, 1400 master and 400 PhD degrees. Department of Computer Engineering was established in 1993 and is the youngest department on the Faculty of Electrical Engineering.

The tradition of high-quality teaching at the Faculty of Electrical Engineering and the need for making the adoption of knowledge in the field of computer design easier, induced teachers of the Faculty to organize classes, laboratory exercises and compulsory projects in the manner that will help students understand matter and grasp the essence of the knowledge, necessary for coping with real-life problems they would meet in their professional life.

This paper, together with corresponding presentation has the goal to guide the audience through the process of teaching hardware related subjects on Department of Computer Engineering on Faculty of Electrical Engineering at University of Belgrade. It was written from the student's perspective, and incorporates experience collected while following lectures in this subjects, finishing laboratory exercises, doing compulsory projects and preparing and taking exams.

The paper is organized as follows. Section 2 gives the overview of the current curriculum for the Department of Computer Engineering and some approaching changes. Section 3 presents subjects and educational software used in lower division courses, while Section 4 is dedicated to higher division courses. Section 5 describes some personal experience in using adopted knowledge in solving a real-life problem. Section 6 concludes the paper.

2. Curriculum

Studying at the Faculty of Electrical Engineering is carried out in 10 semesters, of which first 4 are common for all courses of study. At the beginning of the 3rd year students choose their major. After finishing all 5 years, students are awarded diploma of the high education, and the title graduate engineer.

For the analysis of the school curriculum, considering the purpose of this paper, from the total set of subjects taught at the Department of Computer Engineering, one subset of computer-related subjects is distinguished. Outside this group stayed common engineering educational classes (mathematics, physics), common electrical engineering educational classes (fundamentals of electrical engineering, electrical circuit theory) and classes that introduce students to the majors of all other courses of study (electronics, telecommunications, automation, physics of materials).

Further, from the total number of computer-related subjects, another subset is noticed, which is formed by subjects that consider hardware design.

The total number of exams is 42, and the average number of hours that students weekly spend at the University throughout all five years is 28.55. Because the curriculum of the fifth year consists of some compulsory and a number of elective subjects, and also of completing 2 projects and final exam, it is solely on student to choose the final course of his/hers studies. That is why the data is given separately for the first 8 semesters and separately for the last two.

In Table 1 column 1 denotes the number of classes that students weekly spend at the University, the columns 2 and 3 represents the number of computer-related and hardware-related classes per week, respectively. The last two columns are showing the share that computer-related classes have in all classes and the share that hardware-related classes have in total number of computer classes. These data is given for the first 8 semesters separately, and then the average value for these 8 semesters.

Semester	1	2	3	2/1	3/2
	Total [hour/week]	Computer [hour/week]	HW [hour/week]	Computer [%]	HW [%]
1	30	4	4	13.33	100.00
2	30	4	4	13.33	100.00
3	30	9	5	30.00	55.56
4	30	5	0	16.67	0.00
5	30	14	10	46.67	71.43
6	30	19	10	63.33	52.63
7	30	30	6	100.00	20.00
8	25.5	25.5	0	100.00	0.00
Average	29.44	13.81	4.88	46.92	35.29

TABLE 1: NUMBER OF CLASSES PER WEEK (FIRST 8 SEMESTERS)

The graphic representation of these data is given in Figure 1 in the form of bar chart. Figure 1 shows the increase in number of computer classes during the studies, but also states that the number of hardware classes has its maximum in 5th and 6th semester.

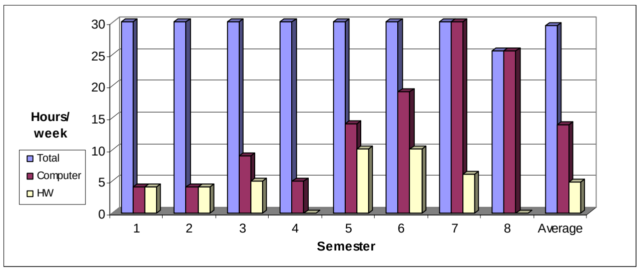


Figure 1: Number of classes per week (first 8 semesters) - graphic representation

Semester	1	2	3	2/1	3/2
	Total	Computer	HW	Computer [%]	HW [%]
1	1	0	0	0.00	
2	5	1	1	20.00	100.00
3	4	1	1	25.00	100.00
4	7	1	0	14.29	0.00
5	2	0	0	0.00	
6	6	4	2	66.67	50.00
7	4	4	1	100.00	25.00
8	6	6	0	100.00	0.00
Sum	35	17	5	48.57	45.83

TABLE 2: NUMBER OF EXAMS (FIRST 8 SEMESTERS)

In Table 2 column 1 denotes the total number of exams, the columns 2 and 3 represents the number of computer-related and hardware-related exams, respectively. The last two columns are showing the share that computer-related exams have in all exams and the share that hardware-related exams have in total number of computer exams. These data is given for the first 8 semesters separately, and then the total sum for these 8 semesters.

Figure 2 is the graphic representation of the Table 1.

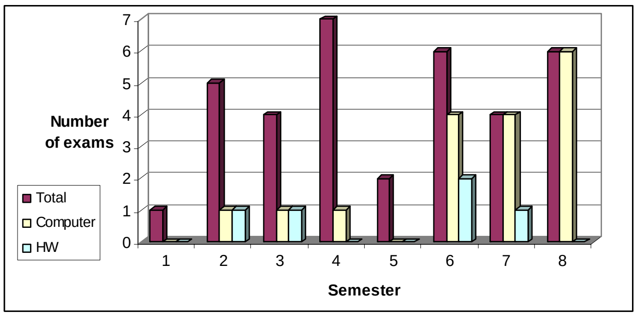


Figure 2: Number of exams (first 8 semesters) - graphic representation

The number of classes per week and number of exams in 9th and 10th semester are given in Table 3 and Table 4, respectively. Because of the adjustable curriculum, some of the ratios are individual and different for each student thus could not be given.

Semester	1	2	3	2/1
	Total [hour/week]	Computer [hour/week]	HW [hour/week]	Computer [%]
9	26	22	>=6	84.62
10	24	24		100.00

TABLE 3: NUMBER OF CLASSES PER WEEK (LAST 2 SEMESTERS)

Semester	1	2	3	2/1
	Total	Computer	HW	Computer [%]
9	6	5	>=1	83.33
10	1	1		100.00

TABLE 4: NUMBER OF EXAMS (LAST 2 SEMESTERS)

In the last year of studies, students have only 2 compulsory subjects, with possibility to choose 3 subjects from the group of 9 elective subjects. The list of elective subjects is updated almost every year, depending on the appearing of the new technologies and the interest of students. Currently there are 3 subjects that consider advanced architecture and organization of computers (Controlling computer systems, Parallel computer systems and Multiprocessing systems) That is one third of all subject, thus fitting into average values for other years.

Last year authorities of Serbia and Montenegro signed the Bologna Declaration, and the text of the new Law of Education and Universities that corresponds to that agreement is proposed, and it is still in public discussion. When the law is accepted, the Faculties are going to be obligated to change their organization and curriculums to correspond with the law and the Bologna Declaration. Some movements in that directions are already made at Belgrade University and the Faculty of Electrical Engineering.

It is easily noticeable that only about one half of all classes are computer related classes, and that is mainly because of the fact that the firs two years are common to all courses of study. There are some subjects in the lover division that broaden the knowledge of the student, but most likely the student would not at all or would rarely need that knowledge in his/her future job. That is why the changes would incorporate students' decision of the major subject and course of studies earlier, maybe even at enrollment. That would allow reducing of the number of common engineering educational classes and classes that introduce students to the majors of other courses of study, in that way changing the ratio in the favor of the computer-related classes.

Other changes would be of organizational nature, considering dividing multi-semester exams into more single-semester, and introducing the system of ECTS scoring. These changes are already influencing the studies of the students enrolled during this year. Also awarding a masters degree after finished 5 years and diploma after only 3 years is planed, while the curriculums of individual subjects would stay more or less unchanged. The complete curricula of the subjects connected with computer design, with topics taught, are given in Appendix.

3. First, Second and Third Year

In this section an overview of three educational software systems will be given, one for each subject taught. At the end the system for on-line testing of students using the Internet is described.

The teaching of the hardware-related subjects starts in very first year of studies with subject "Introduction to Computer Engineering". The first-year course covers the switching functions, the combinational and sequential switching circuits, the analyses and syntheses of switching circuits and the standard modules such as multiplexers, decoders, arithmetic and logic units, registers, counters, etc. The second-year course in Computer Architecture covers the basic concepts related to the most commonly found structure of a computer, which includes the processor, the memory, the input/output subsystem and the bus. The course in Computer Architecture and Organization, that is taught at the third year, goes one step further covering the topics such as the architecture and organization of CISC and RISC processors, the organization of pipelined processors, the storage system, the interconnection networks, and the memory system. These courses have been lectured for a few years and cover the core topics in computer architecture and organization identified by the joint IEEE Computer Society and ACM Computer Engineering Task Force [15].

Generally, a great problem in teaching any course in the field of computer architecture and organization is to provide means which would facilitate the students to make a cognitive leap from the blackboard description of the architecture and organization of a computer to its utilization as a programmable device and connect their theoretical knowledge with practical experience. The laboratory where students get hands-on experience is crucial in helping them to bridge this gap between theoretical knowledge and the practical problems architects face in designing various computer system modules. The commonest approach in narrowing this gap relies on software simulators of computer systems. Software simulators have several advantages over "real" microcomputer platforms: they are less expensive, more flexible and more appropriate for lower division courses, which typically have a large number of students. Additionally, graphical presentation and animation help students to "experience" computer system functioning and better understand various design issues. Several specialized educational simulators of computer systems have been developed at the Department of Computer Engineering.

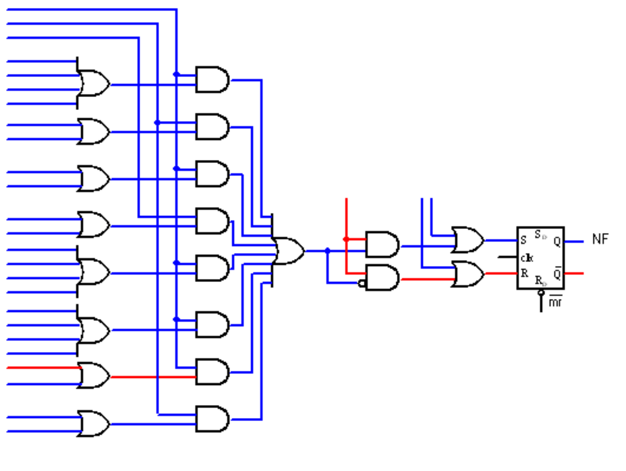


Figure 3: Screenshot from the educational software used in the first year

Students of the first year have the possibility to use digital system design software, developed at our Department.

During the lab exercises or at home, students can interconnect logical elements and standard modules into combinational and sequential circuits, set their inputs, and watch the output values. The logical states of the signal lines are visually presented by color, where blue represents low state or logical "0", while red denotes high state or logical "1". This helps students to understand the matter taught at the course Introduction to Computer Engineering by creating a switching circuits by themselves and interactively setting different input values and noticing changes at the output. One example of digital circuit designed using this software is given in Figure 3.

For the students in the second year another more complicated software is developed. It covers the topics taught in the course in "Computer Architecture" and it is used in compulsory laboratory exercises, but also students can use it at their home. One educational computer system design called EDCOMP is used as a base for the simulation. The structure of EDCOMP is given in Figure 4.

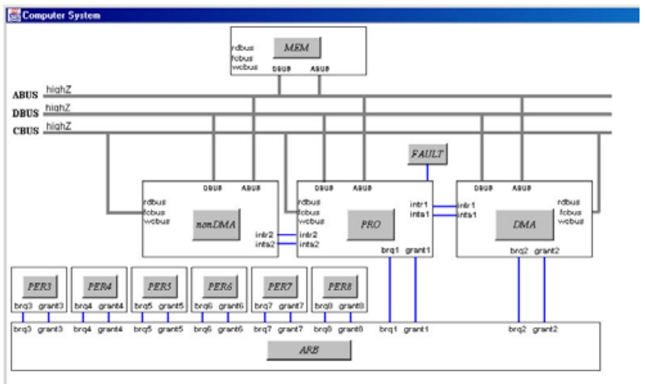


Figure 4: Structure of EDCOMP

The Web-based graphical simulator supports animation of instruction execution and allows students to write their own assembly programs, translate them, interactively set and examine values of memory locations, registers, and input/output units, and run simulation. It gives a visual presentation of all parts of the computer system both at the level of standard system modules and at the level of combinational and sequential circuits. It also displays the values of signals. Simulation can be performed at the level of a clock cycle, an instruction, and a complete program. Further, the timing diagrams of selected signals can be displayed.

The simulator graphically presents parts of the computer system and signal values, simulates the behavior of EDCOMP, and displays simulation results in a user-friendly manner. During a simulation run, two windows are present on the screen. The larger window in the upper part of the screen, named the Block Diagram Window, shows parts of the computer system. Because a limited number of elements can be displayed on a screen, a two-level hierarchical scheme of screens is developed. The first level screen gives the structure of the computer system at the level of modules (shown in Figure 4), and the simulation run begins with this screen. For a more detailed structure of any of the modules, the student needs to point and click to select a module and go to the second level of screens. The second level screen gives the structure of the module's processing unit at the level of combinational and sequential circuits, as presented in Figure 5. The exception is the processor module for which the user goes from the first to the second level through an intermediate level screen, which gives the structure of the processor module at the level of units. For each signal coming from another part of the computer system, there is a link button with the name of the module where the signal is generated; this provides an easy and fast navigation through screens. A single line changes its color depending on the current value and a bus is accompanied by its current value.

The Main Window in the lower part of the screen (see Figure 5) shows the status of simulation (PC – program counter, T – step counter, Tclk – processor clock cycles executed), the control signals generated for that clock period, and a brief explanation of the actions to take place during that clock period in the Sequence box. The command buttons support navigation (UP, Hierarchy, and Back), stop of simulation (Exit), simulation (Clk, Ins, Prg, and Clear) and examination of simulation results (Show, Clock, Signals), and saving the simulation context (Save).

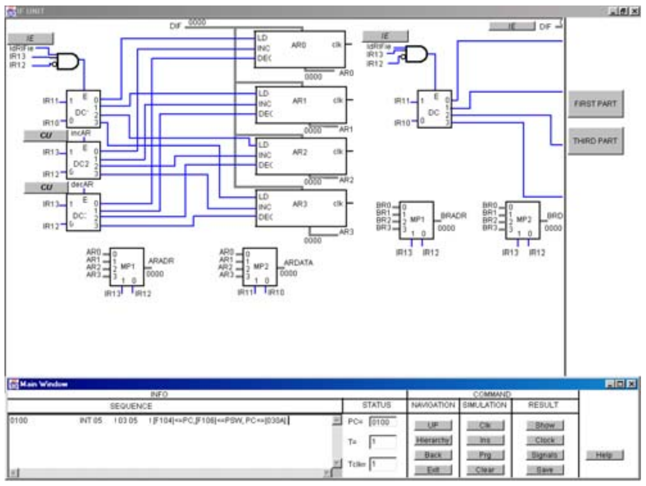


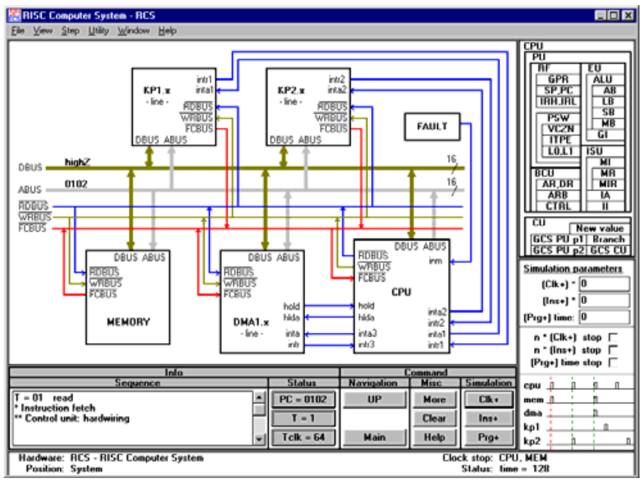
Figure 5: Screenshot from the educational software used in the second year

The simulator flexibility allows students to focus on a specific topic. For example, when studying integer multiply instruction execution, students can run a simulation on a clock-by-clock cycle basis, closely following the changes in the relevant registers and ALU. On the other hand, when learning I/O fundamentals, students can follow simulation at the system level, where the processor is considered as a black box, executing a program on the instruction-by-instruction level.

The initial version of the simulator was developed as a standalone application in Java. After several years of use, it was decided to pursue a transition to a Web-based environment due to following reasons. The number of students enrolled in the course exceeds 400, thus posing a tremendous pressure to laboratory equipment and staff. Web-based environment allows students to prepare for lab at home, at their own pace, thus reducing time needed for successful completion of lab exercises. The Web-based technology also offers seamless integration with knowledge assessment and administrative tasks and cost reductions for installations, updates, and maintenance.

More details regarding simulator capabilities can be found at http://electra.etf.bg.ac.yu:8080/SIMCISCO/SIMCISCO.html.

As a complexity of the mater taught increases, more and more complicated structures of software tools are needed. For the purpose of teaching third-year course in "Computer Architecture and Organization", and to help the students take next step into advanced computer architecture, another software package is developed by professor Dr. Jovan Djordjevic and his associates. It consists of three self contained systems: the RISC processor based educational computer system (the RCS system), the CISC processor based educational computer system (the CCS system), and the educational hierarchical memory system (the HMS system). It helps in understanding topics like pipelined design, the difference between CISC and RISC architectures, hierarchical cache memories and virtual memories. The simulator based on RCS system is described here, while the other two have similar user interface.


T = 01 read
* Instruction fetch
** Control unit: hardwiring

PC = 0102
T = 1
Tclk = 64

Hardware: RCS - RISC Computer System
Position: System
Clock stop: CPU, MEM
Status: time = 128

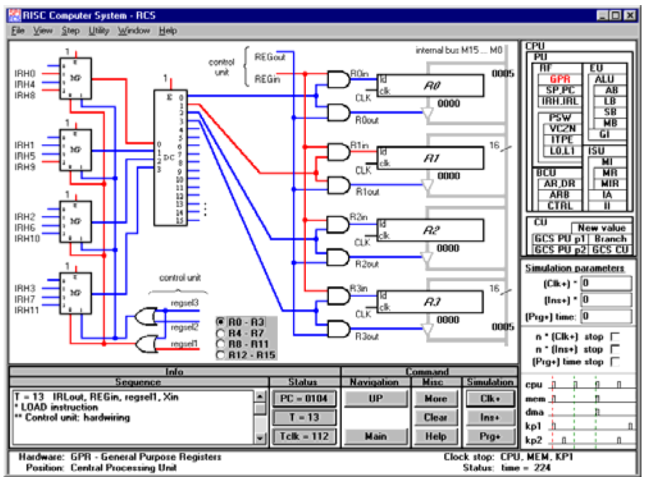
Figure 6: Units of the RCS system

Figure 6 shows the root of the hierarchical scheme of the RCS with all relevant units. Each screen, in general, is made up of the title bar, menu, and five windows.

The largest window in the upper left part of the screen, named the Block diagram window, contains either only a composition of combinatorial and sequential circuits or a composition of subblocks, that can be further selected, and combinatorial and sequential circuits. The upper right window, named the Hierarchy window, shows the hierarchical scheme of the CPU. Each block from the hierarchical scheme is further presented with one or more screens. The traversing through the blocks of the hierarchical scheme can be achieved by selecting the appropriate block, at any moment.

The lower left window, named the Info and Command window contains the status of the simulation, a brief explanation of the actions that are going to take place during that clock period; command buttons for executing the simulation to the next clock, or to complete the current instruction, or the complete program; command buttons for moving through the levels of hierarchy (Up, Main); and some more buttons (Help, Clear button that returns the simulation to the beginning)

The lower right window, named the Signal graph window, shows the simulation control parameters and clock timing diagrams. The bottom window, named the Status window, shows the status information such as the description of the Block diagram window, the current position inside the RCS and the total number of system clocks elapsed.


T = 13 IRLout, REGin, regsel1, Xin
* LOAD instruction
** Control unit: hardwiring

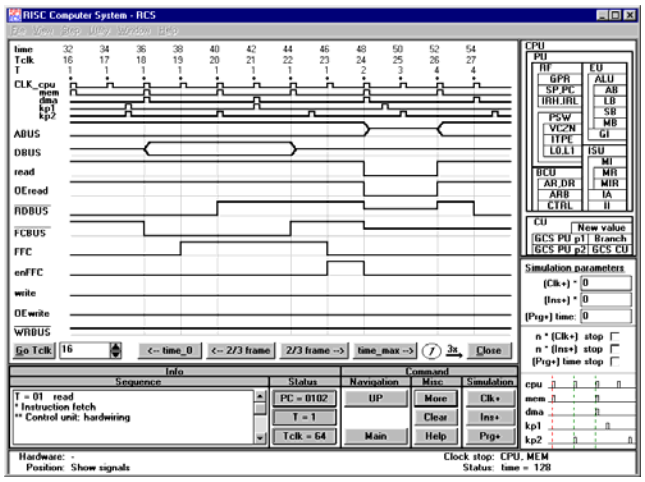
PC = 0104
T = 13
Tclk = 112

Hardware: GPR - General Purpose Registers
Position: Central Processing Unit
Clock stop: CPU, MEM, KP1
Status: time = 224

Figure 7: Screenshot from the RCS simulator used in the third year

The running of the simulator with the hierarchy of screens of the RCS is described briefly in the following. The first screen with which the simulation begins is the root hierarchical screen, which gives the block structure of the RCS (shown in Figure 6). If student needs more detailed structure of any of the blocks from Figure 6, he/she can move one level down in the hierarchy by clicking at that block. For example, by positioning the cursor at block Register File, one goes one level down in the hierarchy and gets the block structure of the Register File Unit. The same actions applied this time to the GPR block, brings the screen of the last level in the hierarchy (Figure 7). This screen can be obtained directly at any time by clicking on the GPR box in the Hierarchy window. Here one can see the design of this block at the level of standard sequential elements (registers, flip-flops, etc.) and combinatorial elements (decoders, logical circuits, etc.) Groups of lines, such as data and address bus are given either in green if they are in the state of high impedance or in grey with the valid values given in the hexadecimal form, while the control bus lines and other signal lines are coloured either in blue or red depending on whether the signal on that line has logical value "0" or "1", respectively.

At any time during the simulation by activating button More student can examine and set the memory, examine and set registers, and get the timing diagram of selected signals from the beginning of the simulation until the current clock period, as shown in Figure 8. Buttons at this screen allow students to move the time frame to the desired clock, to the beginning, 8 clocks backward, 8 clocks forward, or to the end. Any combination of signals can be selected during the simulator set up.



T = 01 read
* Instruction fetch
** Control unit: hardwiring

PC = 0102
T = 1
Tclk = 64

Simulation parameters
(Clk+) * 0
(Ins+) * 0
(Prg+) time: 0

Hardware: -
Position: Show signals
Clock stop: CPU, MEM
Status: time = 128

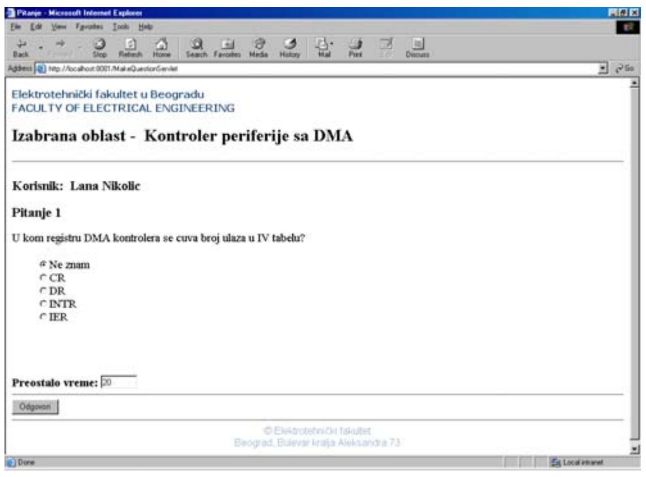
Figure 8: Timing diagrams of the RCS

The organization of the simulator with the hierarchical structure of screens makes it possible to carry out the simulation of the functioning of the RCS at various hierarchical levels. At higher levels, student can follow the simulation at the level of signal exchanged between blocks considered here as black boxes. At the lowest level, if it is deemed interesting, student can follow the simulation at the level of registers, flip-flops, logical circuits etc. This provide abstracting the unnecessary details at the stage of grasping the essence of the matters, but allow detailed insight to the lowest levels of design, when the basics are understood. This was proved as a good way of adopting the knowledge.

As a part of classes for all of these three subjects compulsory lab exercises are made. Each exercise has four components: prelab preparation, in-lab knowledge assessment, in-lab assignment, and written report. To prepare for a particular lab, the students must review related material from lectures and the textbook, and read the related sections from the lab manual. They can also access simulators from home and use it for self-study. Each lab assignment is preceded by a short computer-based test aimed to verify whether the students understand the topic covered in the assignment. After passing the test, the students select a predefined experiment, which results in the initialization of the computer system from a file. Then, they execute the programs, either at the instruction level or at the clock level. Before, during, and after the simulation, the students are requested to examine the values of relevant memory locations and registers in the processor and the input/output units, follow the values of selected signals at combinational and sequential circuits, and draw their timing diagrams. In some of the experiments the students are also asked to develop their own assembly programs using the Editor, the Translator, and the Loader and to verify their correctness and determine performance. Based on the observations made during the experiment, they answer questions relevant to the topic and turn in a written report.

Knowledge assessment for all of these three subjects is conducted in a lab by a testing software which is since last year available on-line, through the Internet. This means that student can take trial tests from any place at any time, from any personal computer provided with Internet connection and a web browser. In the addition, student can choose the complexity of the questions, also the topics of interest and the number of questions and then the questions matching that criteria are randomly chosen from the database. It is possible to define the time-limit for answering the questions, so that allow students at home to make the conditions similar to these in the laboratory. The statistical overview tells the student in which topics he/she needs improvement, and explanations given with the each answer make possible for students to learn in this way.

When a student opens and log in at the web address: http://electra.etf.bg.ac.yu:8080/Test, he/she will be offered to configure the test parameters, and then to take the test. Then he/she will be given a screen with the question and offered answers, like in Figure 9. After a completion, the results will be presented to the student, with correct answer and explanation for each incorrectly given answer.


Elektrotehnički fakultet u Beogradu
FACULTY OF ELECTRICAL ENGINEERING

Izabrana oblast - Kontroler periferije sa DMA

Korisnik: Lana Nikolic

Pitanje 1

U kom registru DMA kontrolera se cuva broj ulaza u IV tabelu?

- [✓ Ne znam]
- CR
- DR
- INTR
- IER

Preostalo vreme: 20

© Elektrotehnički fakultet
Beograd, Bulevar kralja Aleksandra 73

Pitanje - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address http://localhost:8081/MakeQuestionServlet
Go
Odgovori
Done
Local intranet

Figure 9: Screenshot of on-line testing software

This web tool facilitate the students with a way to continually estimate their knowledge and the quick way of spotting the lacks in it. The tool's flexibility is important because of the different level of pre-knowledge of the students entering the courses. In that manner, one can choose to answer to questions only of lower complexity at the beginning and then proceed with medium or higher complexity, or can gradually decrease time-limit for answering the questions. This allows everyone to learn at his/hers own pace.

4. Fourth and Fifth Year

Both courses in the higher division years have the goal to prepare the students for the real-life projects, similar to that the student is supposed to meet at his/hers future professional life. This is usually achieved by teaching the existing examples of the devices or systems, and commenting implementation details, together with demand on students to complete at least two projects individually. Other part of the classes are practical, and are dedicated to acquainting students with commercial tools needed for the completing such a project.

The first of these constructive courses is "Microprocessor Systems". The curriculum of this subject consists of several case studies, in different topics starting at assembly level programming, through theory and practice of pipelining, branch prediction and multithreading to the theory and practice of Pentium of newer generation. The home project in this subject comprises development of microprocessor system using Intel 8086 and 8051/52 microprocessors, development of interfaces and needed software for special purpose computing. This year home assignment required a design of elementary operating system for the designed microprocessor system. The commercial tools that are generally used are Protel 99 SE for circuit design, different assemblers, and Keil C extension of ANSI C.

The other course "VLSI Systems" is taught at the final year of studies. The objective of this subject is to introduce students to designing hardware VLSI device. The curriculum consists of different topics, among them are hardware design based on geometrical, logical and functional symbols, design based on the method of standard cells and FPGA chips. As a case study is used design of microprocessor DARPA RISC MIPS on 200MHz carried out by our teacher professor Dr. Veljko Milutinović. In the addition the overview of newest commercial FPGA chips is given; Max series by Altera and Spartan series by Xilinx are described in detail and compared. The practical part of the classes encompasses details of the VHDL and Verilog hardware description languages.

Beside craft skills gain on these courses, there is another component of the adopted knowledge: the designer's point of view. Good designer has to be able to choose between different possibilities for implementation, to weigh the different tradeoffs, correctly estimate advantages and disadvantages of chosen design. This is usually achieved by experience, but for the students at the University this process is accelerated by studying different case studies, their implementation details, tradeoffs and problems emerged.

The creators of the commercial tools used in these subjects usually pay attention to usability and shortening of the production process, rather than didactic and learning aspect of the tool. However, the whole process of learning of this topics can be made easier by allowing students to easily exchange their thoughts and experience considering tool usage or some design issue. This is attained by mailing lists, and starting this year "Future Learning Environment".

Two mailing lists are created, one for each course. Any student is free to subscribe or unsubscribe at any time he/she likes. The majority of information considering the classes, exams and individual projects go through this mailing lists. That is why almost entire number of students enrolled in these courses are subscribed to mailing lists. Usually students use this lists to exchange experience or ask different questions, and other students or the assisting teacher answer these questions. These mailing lists have very lively traffic with up to 200 messages per semester, from which about 20% are posted by assisting teacher as a various information or an answers to questions.

The FLE system started with test use this year, and it is offered to students to freely join this computer supported collaborative learning system placed on server with address http://titan.etf.bg.ac.yu/titanium/FLE. This environment consists of a personal Webtops. Webtops can be used to store different items (documents, files, links to web etc.) related to studies, organize them into folders and share them with others. With Knowledge Building tool groups may carry out knowledge building dialogues, theory building and debates by storing their thoughts into a shared database. The knowledge building discussions are structured by knowledge types. Another part of FLE is Jamming tool. Jamming is a tool for collaborative construction of digital artifacts. Using jamming tool student can explore the possibilities of changing a file by making new versions of the starting artifact together with others [16].

Only small number of students utilized this environment during its first semester of test use. Plans are to increase this number and to induce students to collaborate, but some of the basic assumptions of the environment of this kind seem to be in contradiction with the demand for the individual completion of the projects.

5. EXPERIENCES

As a member of team that represented Belgrade University at international CSIDC 2004 competition, the author together with three colleague students encountered several challenges. During the work on the project "Small Sonar Device" (see http://titan.etf.bg.ac.yu/csidc) the synthesis of knowledge and learning some new things were necessary. The knowledge gathered from the courses at the University proved itself as very useful; accepted methodology of learning and good basics made quick learning possible.

The project consisted of designing and implementing handheld sonar device, that could be used in emergency cases when the visibility is reduced. It had two parts: hardware device, designed with DSP and ultrasound sensors; and software for the handheld device to which sonar would be attached (PDA, Java enabled mobile phone etc.) The production of hardware device, besides topics learned at classes, needed some special knowledge about DSP and ultrasound sensor and some theory on sonar functioning for developing the algorithm for recognizing the objects. Software development considered assembly level programming with the use of specific instructions of DSP, and the use of the unique features of the handheld devices in higher programming languages.

The communication between team members was satisfactory. The mailing list was formed at the beginning of the project, and the most of the information went through it. Weekly meeting were used for defining the next week assignments for each member; for clearing the dilemmas and finding the answer to current questions. Content management system (http://titan.etf.bg.ac.yu/titanium/plone) named "Plone" was used for exchanging the documents considering user manuals for different software tools, design notes and working versions of programs. Each member had password protected on-line storage space, and could choose which of the uploaded documents are going to be published and thus shared with other users of CMS. For tracking the changes between different versions of programs, and for allowing cooperation in writing these programs, CVS system for version control was used (http://titan.etf.bg.ac.yu/cvs/viewcvs.cgi). It works by checking out the source program files from the on-line database, changing them locally and then checking them back in. All these services are available through the Internet.

At the end, things were complicated by lack of understanding for educational affairs. Namely, the only firm in Serbia that had the equipment enough sophisticated for achieving resolution high enough for printing the printed circuit board, refused the job. They had a big order for PCBs going on from an international company, and did not have time to deal with students. But that is the kind of real-life problems we will have to cope with in future everyday professional life. The final prototype was late, and some lessons learned.

6. CONCLUSION

It is not the only goal of learning about computer hardware to finally have enough skills to design one, but having insight in computer architecture and organization is a way of writing good optimized low-level software, operating systems and it is a basics for understanding other, more complicated architectures like distributed and multiprocessor systems.

At the job fair "Career Days" held this April in Belgrade 38 companies offered jobs. Only 8 of them (about 20%) searched for computer engineers. Further 10 companies (25%) needed graduate students regardless of the course of their studies. After three months about 100 graduate students that visited this fair found job. There is no data of how many computer engineers were among them.

The offer of hardware-related jobs is somewhat narrower than the offer in software industry, especially in our country, with its stage of development in technology. On the other hand, though there are some more schools of Belgrade University that give education in software engineering, only engineers from Faculty of Electrical Engineering have enough both software and hardware knowledge to cope with jobs that require some hardware design. So while others learn how to use the computer, we learn how to design one. That is where the idea for the title comes from.

APPENDIX

The official curricula of the subjects that consider hardware design are given. The numbers in parentheses denotes the semester and the number of lectures, practical classes and lab exercises per week, respectively.

Introduction to computer engineering (I: 2+2+0 II: 2+2+0)
Boolean algebra. Switching functions and circuits. Minimization of switching functions (Karnaugh maps). Combinational and sequential systems (coders, decoders, multiplexers, demultiplexers, adders, arithmetic and logic units). Sequential systems (types, flip-flops, static and shift registers, counters). Memory elements (ROM, RAM). Analysis and design of combinational and sequential systems.
Computer architecture (III: 3+1+1)
Types of computer architectures. Machine code and Assembler. CPU structure (ALU, registers, stack, operation modes). Control unit (design using delay elements, counters and decoders; programmable units). Microcomputing. Interrupts (hierarchy, masking). Input and output (controllers, programmable IO). DMA. Memory (associative, LIFO and FIFO memory, operating memory, cache). Data, address and control buses. Fundamentals of operating systems (multitasking, process synchronization, semaphores, queues). Data bases.
Pulse and digital electronics (V: 3+2+2 VI: 3+2+1)
Pulse electronics: Switching properties of diode, BJT and MOSFET. Logic gates in bipolar technology (RTL, DTL, TTL, STTL, ASTTL, ALSTTL, ECL, three stage output). Logic gates in NMOS and CMOS technology. Flip-Flops (positive feedback concept, SR, T, JK, D and MS flip-flops; TTL, ECL, NMOS and CMOS logic). Anstable and monostable circuits. Comparators (differential, Schmidt trigger). Time base generators (Miller, Bootstrap). Digital electronics: Registers. Shift registers. Combinational digital systems (decoders, coders, multiplexers, demultiplexers, PAL, PLA). Sequential digital systems (parallel and serial binary counters, bi-directional counters). Memories (static, dynamic, ROM, PROM, EPROM, EEPROM, NVRAM). Arithmetic and logic units (adders, multipliers). A/D converters. D/A converters.
Computer architecture and organization (V: 2+1+0 VI: 2+2+1)
Introduction. Computer structure. Functional units. Processor. I/O devices. Memory. Bus. Computer architecture. Registers. Instruction format. Types of data. Addressing. Instruction set. Standard and special instructions. Interrupt. Interrupt routine. Internal and external interrupts. Masking. Priorities. Processor organization. Instruction execution phases. Operational unit. Registers. ALU. Internal bus. Interrupt unit. Control unit. Instruction flowchart. Control unit realization. Hardwired realization. Microprogram realization. Horizontal, vertical, mixed and nano-programming. I/O. I/O device controllers. Organization of I/O. Programmed I/O using status register and interrupt. I/O using DMA controller. I/O using I/O processor. Memory. Organization of operative memory. Overlay. Distribution of the addresses. Cache. Associative, direct, and set-associative copying. Word and block level copying. Substitution algorithms. Data consistency. Virtual memory. Paged, segmented and segment-paged organization. Associative, direct, and set-associative copying. Bus. Asynchronous bus. Arbitration. Parallel and serial arbitration. bus transfers. Read and write cycles. Multi-master busses. Local busses. Synchronous bus. Pipeline. Instruction phases. Pipeline stages. Conflicts and their solving. Read-ahead, alternative buffers, continual loops, discontinual loops, jump prediction.
Microprocessor systems (VII: 3+1+2)
Theory and practice of microprocessor assembly level programming. Examples. Case study: ix86. Class project: Ten assembly level programs of interest for the students. Theory and practice of input/output programming. Interrupt i/o, peripherals i/o, and DMA programming in microprocessor environment. Examples. Case study: ix86. Class project: One large i/o program which changes each semester. Theory and practice of microprocessor interface hardware. Examples. Case study: ix86. Class project: Assembly/disassembly or analysis/synthesis of a hardware/software system for general purpose computing. Theory and practice of microprocessor assembly level programming. Examples. Case study: ix86. Class project: Assembly/disassembly or analysis/synthesis of a hardware/software system for special purpose computing. Theory and practice of Pentium, Pentium Pro, Pentium MMX, and Pentium II. Architectural, organizational, and design issues. Pipeline, caching, branch prediction, interrupt, support for SMP and DSM. Engineering issues. Theory and practice of Merced and Beyond. Architectural, organizational, and design issues. Instruction level parallelism, caching, branch prediction strategies, i/o, multithreading, issues of interest for SMP and DSM.
VLSI systems (IX: 3+1+2)
Introduction to VLSI. Design based on geometrical, logical and functional symbols. Hardware and software details of DARPA RISC MIPS microprocessor on 200MHz. Description of internal resources in architectural, organizational and design level. ISP language and ENDOT simulation package. Design of the microprocessor by the method of standard cells. Design using FPGA chips. Silicon compilation. Details of VHDL and Verilog languages. Testing. Examples of implementations in VHDL and Verilog. Examples of applications implemented on leading world universities. Two compulsory projects.

REFERENCES

[1] Prof. Dr Borivoj Ž. Lazić: Logičko projektovanje računara, Nauka Beograd 1998
[2] Dr Jovan Djordjević: Priručnik iz arhitekture računara, ETF 1998
[3] Dr Jovan Djordjević: Priručnik iz arhitekture i organizacije računara, ETF 1999
[4] Milutinovic, V., Microprocessor and Multimicroprocessor Systems, Copyright by Veljko Milutinovic, 1998
[5] Douglas V. Hall, Microprocessors and Interfacing - Programming and Hardware, 2nd Edition, Macmilan/McGraw-Hill, USA
[6] I. Scott MacKenzie, The 8051 Microcontroller, Prentice Hall, USA
[7] Veljko Milutinović, Projektovanje Telekomunikacionih uređaja pomo-ću mikroračunara, Institut "Mihajlo Pupin", Beograd
[8] Veljko Milutinovic, Dragan Božanic, Dejan Polomcic, Milivoje Aleksic, Uvod u projektovanje racunarskih sistema, Beograd
[9] Veljko Milutinović, Projektovanje i arhitektura RISC procesora za VLSI, Nauka, Beograd
[10] Peter J. Ashanden, The Designer's Guide to VHDL, Morgan Koufmann Publishers, Inc, USA
[11] John L. Henessy, David A.Patterson, Computer Architecture a Quantitative Approach, Second Edition, Morgan Koufmann Publishers, Inc, USA
[12] Jovan Djordjević, Bosko Nikolić, Aleksandar Milenković: Flexible Web-based Educational System for Teaching Computer Architecture and Organization
[13] Jovan Djordjevic, Aleksandar Milenkovic, Nenad Grbanovic: An Integrated Educational Environment for Teaching Computer Architecture and Organisation
[14] J. Djordjevic, B. Nikolic, M. Mitrovic: A memory system for education
[15] "Computer Engineering: Computing Curricula 2001," IEEE Computer Society and ACM February 2003
[16] http://fle3.uiah.fi
[17] http://www.etf.bg.ac.yu/