An Educational Environment for Teaching a Course in Computer Architecture and Organization

Jovan Djordjevic, Aleksandar Milenkovic, Nenad Grbanovic, Miroslav Bojovic
Faculty of Electrical Engineering, University of Belgrade
P.O.Box 3554, 11120 Belgrade, Yugoslavia
e-mail: jdjordjevic@kiklop.etf.bg.ac.yu

Abstract: *The paper presents an educational environment for teaching a course in Computer architecture and organization. It is made up of an educational computer system, a reference manual, a software package and a set of laboratory experiments. The educational computer system is devised in such a way that it covers the basic structure of a computer system: processor, memory, input/output subsystem and bus. The reference manual provides all implementation details with the appropriate circuits drawings and detailed descriptions. For the devised educational computer system a software package is developed which includes the program development tools and the graphic simulator. They make it possible to develop programs for it and execute them under the graphic simulator. The simulator allows to execute programs at the clock, instruction and program levels and to examine, at any time, the values of all signals of the educational computer system down to the register transfer level. In the paper is also given a set of laboratory experiments that the students must carry out successfully using the reference manual and the software package as a prerequisite for taking an exam in Computer architecture and organization.*

1. Introduction

The course in Computer architecture and organization is a second year undergraduate course at the Faculty of Electrical Engineering, University of Belgrade attended by the students at the Department of Communications, Automation and Electronics and the Department of Computer Science. The students of both departments acquire the knowledge necessarry for following such a course through the first year course in Fundamentals of Computer Science and the second year course in Programming methodology and languages. In them they study, first, the binary arithmetic, the standard combinational and sequential circuits and the design of digital computers and, then, the programming methodology, the assembly language and high level programming languages such as Fortran, Pascal and C. The course in Computer architecture and organization 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. Such approach of the course is taken under the assumption that the students from the Department of Communications, Automation and Electronics will primarily use relatively simple configurations of computer systems as controllers in communication pieces of equipment, automated control systems and other computer controlled electronic devices. For the students at the Department of Computer Science this is just an introductory course. Their knowledge in this area they further widen during the remaining three years of studies through the courses such as Advanced computer architecture and organization, Microprocessor systems, Design of VLSI circuits, Distributed computer systems, Parallel computer systems, Multoprocessor systems etc.

The course in Computer architecture and organization is made up of four self contained entities within which are considered the basic concepts of the processor architecture and organization, the memory, the input/output subsystem and the bus. The processor architecture is considered through the study of the processor program controlled registers, data types, instruction formats, addressing modes, instruction set and interrupt mechanism. The processor organization study begins by pointing out that the processor of a particular architecture can be realized with different processor organizations. This follows by the presentation of the digital system design methodology, also applied to the processor design, whereby a digital system is made up of a processing

unit and a control unit. Possible implementations of processing units and control units (hardwired and microprogrammed) are considered in detail. The memory considerations are directed to the problems of the memory modules organizations and their interconnections with the processor and the input/output subsystem through the bus. The input/output subsystem is presented through the description of programming techniques used for input/output and the structure of peripheral device controller and the direct memory access controller (DMA). The bus considerations are directed towards the problems arising when the processor, the memory and the input/output subsystem are being interconnected with common address, data and control lines. The read, write and interrupt vector number acquisition bus cycles are presented as well as the specifics of the asynchronous and synchronous busses.

Generally, a great problem in teaching a course in 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 theoretic knowledge with practical experience. This problem is usually solved by providing some kind of computer simulation demonstrating the behavior of the computer system during the execution of an instruction. Therefore, the authors looked at the papers which describe the concept of using the computer emulation or simulation for teaching the computer architecture and organization with the aim to find something suitable for the above course. A software educational tool, called ET, which simulate a wide variety of computers on a fixed MOH (Microprogram Organized Hardware) is described in [1]. A computer-aided teaching (CAT) package used in a microprocessor system course based on the Z80 CPU is presented in [2]. A computer simulator of a simple educational processor called ASP (Animated Simple Processor) which demonstrates the program execution on the macro- and microcode and electronic level is shown in [3]. However, since none of these covers the outlined course in computer architecture and organization, the authors decided to devise and develop their own educational environment.

The educational environment for teaching the course in Computer architecture and organization includes the Educational Computer System (ECS), the reference manual for it, the Software Package of the ECS (SPECS) and a set of laboratory experiments. The ECS is devised to cover all concepts relevant for the course in Computer architecture and organization and its detailed description is given in the accompanying manual [4]. The ECS is described in Section 2. The SPECS includes: (a) tools for program development which allow writing programs in assembly language, editing, translating, linking and loading programs into the memory, and (b) tools which provide graphical animation of all implementation details of the ECS down to the register level during a program execution. A brief description of the SPECS is given in Section 3. A set of laboratory experiments is devised to demonstrate to the students the situations of interest during their practical work with ECS. The set of laboratory experiments is presented in Section 4.

2. The educational computer system

The educational computer system (ECS) is made up of the processor, the memory and the input/output subsystem while the communication between them is carried out through the asynchronous bus. A detailed description of all its parts is given in the accompanying manual written specifically for the course in Computer architecture and organization.

2.1. The processor architecture and organization

The processor architecture is the load/store type. Thus, only the Load and Store instructions can access operands in the memory, while all remaining instructions work with operands in one of the general purpose registers. The processor program controlled registers include, besides 16 general purpose registers, the program counter (PC), the stack pointer (SP), the program status word (PSW), the interrupt mask register (IMR) and the interrupt vector table pointer (IVTP). Data types supported are 16-bit signed and unsigned integers. The length of the instructions is 16 or 32 bits. The instruction format is the variable one, so that, depending on the operation specified by a particular instruction, the three address, the two address, the one address and

the zero address instruction formats are used. The Load and Store instructions can only explicitly specify the use of one of the following addressing modes: the immediate (only for Load), the memory direct, the register indirect and the register indirect with displacement. All the remaining instructions implicitly use the register direct addressing mode. The instruction set includes the transfer, arithmetic, logic, shift, rotate and control instructions. There are internal and external interrupts. The external interrupts are maskable and have assigned priorities. The **intr$_0$** till **intr$_3$** and **inta$_0$** till **inta$_3$** lines are used to exchange the interrupt request and interrupt acknowledgment signals between the processor and input/output units, respectively.

The processor organization follows the design approach by which the processor is made up of the processing unit and the controlling unit. The processing unit is made up of the register file unit (RFU), the execution unit (EXU), the interrupt service unit (ISU) and the bus interface unit (BIU) interconnected with the 16-bit internal bus. The register file unit contains the program controlled and auxiliary registers. The execution unit is made of the ALUSHIFT block, where the basic arithmetic, logic and shift operations are performed, the auxiliary X and Y registers, where the input data for the ALUSHIFT block are kept, and the combinational circuit, where the values to be set into the condition code bits of the program status register are generated. The bus interface unit contains the circuitry necessarry to perform the read, write and interrupt vector number acquisition bus cycles on the bus and arbitrate between the processor and the DMA controller requests for accessing the bus. The **hold** and **hlda** lines are used to exchange the bus request and bus acknowledgment signals between the processor and the DMA controller. The interrupt service unit contains the circuitry necessarry to accept all internal and external interrupt requests and establish the address of the appropriate interrupt routine. The adopted processor design approach, where the processor is made up of a separate processing and controlling unit, made it possible to design four types of controlling units for the same processing unit. One of them use the hardwired technique, while the remaining three the microprogramming technique with the mixed and vertical formats of microinstructions and nanoprogramming.

2.2. The memory

The memory of the educational computer system has the capacity of 64KWords, the memory word length is 16 bits and the processor generated addresses are for 16 bit words. The memory module has the usual input address lines, the bidirectional data lines and the input control lines to start the memory read or write operations and the output control line to indicate that the requested memory cycle has been finished.

2.3. The input/output subsystem

The input/output subsystem is made up of three input/output units and one direct memory access controller. Each input/output unit is made up of a peripheral device controller and the peripheral device itself. The control part of the peripheral device controller is responsible to accept data from the input peripheral device, store into its data register, set the ready bit in its status register and generate an interrupt request if the interrupt enable bit in its control register is set. In addition to that, it reset the ready bit in its status register when the contents of the data register is read by the processor. The control part of the peripheral device controller also reset the ready bit in its status register when the contents of the data register is written by the processor. In addition to that, it sends the contents of its data register to the output peripheral device, set the ready bit in its status register and generate an interrupt request if the interrupt enable bit in its control register is set. The peripheral device controller has not only the usual control, status and data registers, but, also, the interrupt vector register where the interrupt vector table entry number of this input/output unit is being kept.

The DMA controller is assigned to one of the input/output units. In addition to the usual control, status and data registers, it, also, has the source and destination address registers, the block count register and the interrupt vector register. It supports, not only transfers between the peripheral device and memory and vice versa, but, also, the memory to memory transfers. The transfers can be carried out in the cycle stealing and burst modes. The direct memory access controller is a slave when its registers are being initialized by the processor and a

master during the transfer of data. The required mode of operation is specified by writing the appropriate value into the control register of the direct memory address controller.

The input/output address space is memory mapped, so that the highest 8Kwords addresses of the memory address space are reserved for the purpose of addressing the registers in the peripheral device controllers and the direct memory access controller.

2.4. The bus

The components of the educational computer system are interconnected through the asynchronous bus. The bus is made up of the 16 address lines, 16 data lines and three control lines. By generating the active values on the rd (read) or wr (write) control lines, the processor or the direct memory access controller, as the bus masters, specify that the read or write cycle should be performed on the bus. By generating the active values on the fc (function completed) control line, the memory, the peripheral device controller or the direct memory access controller, as the bus slaves, specify that the initiated read or write cycle has been completed.

3. The Software Package of the ECS

The Software Package of the ECS (SPECS) is presented by giving, firstly, the rational behind the design of the SPECS, and, secondly, the facilities offered by the SPECS.

3.1. The rational behind the design of the SPECS

The laboratory experiments are organized in such a way that the students should at home, before coming to the laboratory, study a particular part of the ECS from the manual and, then, in the laboratory carry out the appropriate experiments that would demonstrate some typical situations concerning this particular part of the ECS. Thus, such organization of laboratory experiments and the use of the ECS in laboratory experiments was possible only if some kind of simulators of the ECS was to be developed. The first choice was to do it by using high level hardware description programming language ISP and programming package ENDOT. This was a natural choice for at least two reasons. Firstly, it allows to describe a piece of hardware down to the register transfer level and carry out the simulation at the clock level. Secondly, it was used in some other courses at the Faculty and there was significant experience in using it which made it possible to develop the simulator of the ECS in a reasonable period of time [5]. The simulator of the ECS was implemented in such a way that the simulation was carried out at the register transfer level. The simulator, used together with the reference manual of the ECS, made it possible to follow the values of all signals inside the ECS at the level of standard combinational and sequential modules while executing the carefully prepared set of laboratory experiments. The simulator, together with the manual, was a useful aid in teaching the inner workings of a computer system and fulfilled its design objectives. However, the textual instead of graphical presentation of all relevant signals and the lack of ability to interactively write, modify, translate and execute programs were obvious shortcomings of the simulator. It was, therefore, decided to start the development of a new graphically oriented simulator and the appropriate user friendly environment named the SPECS.

The SPECS contains the simulator that provide the graphical presentation of all implementation details of the given computer system down to the register transfer level and makes it possible to follow the functioning of all parts of the computer system at the clock, instruction and program levels, and the current values of all signals at the inputs and outputs of all combinational and sequential modules. The SPECS offers various facilities such as the interactive setting and examination of the contents of memory locations, registers etc., the drawing of timing diagrams of any of the selected signals of the computer system during the complete duration of the simulation, etc. It provides program development tools to write programs in its assembly language, edit, translate, link and load them into the memory of the computer system, etc. All these features are provided in a user friendly manner by extensive use of modern tools for the development of Windows applications.

The SPECS is developed using the MS Visual BASIC 3.00 programming language. The simulator works with the Windows95 or WindowsNT operating systems and the minimal configuration required to run it is the 486 PC with 8MB RAM memory.

3.2. The facilities offered by the SPECS

The facilities offered by the SPECS make it possible to (a) specify the configuration and initialize the ECS, and (b) run the simulator.

3.2.1. Specifying the configuration and initialization of the ECS

The specification of the configuration of the ECS is the step which allows to select one of four implemented control units (section 2.1) to be used with the processor processing unit. If this step is not performed then as the default control unit the microprogramming one with the mixed format of control signal encoding is used.

The initialization of the ECS is the step required in order to have the system ready for running. It consists of the initialization of the processor, the memory and the input/output units. The initialization of the processor involves the loading of the PC register with the starting address of the program, the IVTP register with the starting address of the interrupt vector table, the IMR register with the value specifying the input/output units from which interrupts will be accepted, the SP register denoting the top of the stack, the general purpose registers etc. The initialization of the memory includes the loading of the appropriate memory locations with the binary values of the programs, the data to be used during the execution of the program, the addresses of interrupt routines in the interrupt vector table, the data to be sent to the output units, etc. The initialization of the input/output units means the loading of the simulated input peripheral device with the sequence of data that this device will generate and the time when the data should be generated.

The initialization of the ECS can be done in two ways. In the first case the user can interactively write his own programs in a symbolic manner, translate, link and load them into the memory. For this purpose the software tools such as the editor, the translator, the linker and the loader are devised. There are also programs which can be used to write and read into and from memory locations, processor registers and simulated peripheral devices, and to automatically initialize the processor, memory, and simulated peripheral devices. There is, also, the possibility to save the current state of the educational computer system in a file. In the second case the user can select one of the prepared test examples carefully devised to illustrate the typical situations of the above enumerated topics lectured in the course in Computer architecture and organization or recall one of the files with previously saved state of the ECS. This will cause the initialization of ECS with the values necessarry to successfully run the simulator.

3.2.2. Running the simulator

The running of the simulator is the step performed in the same way regardless of which of two possible ways of the initialization of the ECS is used. It allows to follow the values of signals of the ECS at various levels during the execution of a program after a clock, an instruction or a complete program. Due to the fact that a limited number of elements can be displayed on a screen, a hierarchical scheme of the ECS, as shown in Fig. 1, is developed. 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.

Each screen, in general, is made up of two windows. The larger window in the upper part of the screen, named the Block diagram window, contains either only a composition of combinational and sequential circuits, if this is a leaf block in the hierarchical scheme, or a composition of subblocks, that can be further selected, and combinational and sequential circuits, if this is not a leaf block in the hierarchical scheme. The smaller window in the lower part of the screen, named the Info and Command window, is divided into the Info window at its left hand side and Command window at its right hand side. The Status buttons PC and Tclk in the Info window display the current values of program counter and the number of clock periods executed, respectively. The Info window Sequence gives either the value of the microprogram counter mPC, in case that one of the microprogram Control units are used, or the value of the step counter, in case that the hardwired Control unit is used, then the control signals generated for that clock period and, finally, a brief explanation of the actions that are going to take place during that clock period. The Command window contains three groups of command buttons: Navigation, Simulation and Miscellaneous. The Navigation command buttons are UP and Hierarchy. Button UP allows to move from the current screen to the screen one level up in

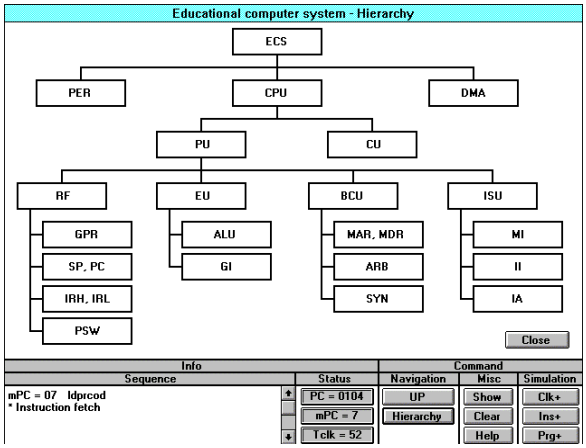

Figure 1. The ECS Hierarchy screen

the hierarchy, while button Hierarchy allows to move directly from the current screen to the ECS Hierarchy screen (Fig.1). Only from the ECS Hierarchy screen one can go directly to the screen of any of the blocks down in the hierarchy. This can be achieved by positioning the cursor at the appropriate block and clicking the mouse button. The Simulation command buttons Clk+, Ins+ and Prg+ allow to continue with the simulation just with one clock period or with the number of clock periods required to complete the current instruction or the complete program, respectively. The Miscellaneous command buttons are Show, Clear and Help. Button Show opens the window which allows to select one of three screens. They further facilitate to show and set the values of memory locations, processor registers and draw the timing diagrams of selected set of signals. Button Clear clears the current state of simulation and returns it to the beginning. Button Help activates the help system where all details concerning the functioning of the educational computer system and its simulator are available.

The running of the simulator with the hierarchy of screens of the ECS is described briefly in the following. The first screen with which the simulation begins is the one with the block structure of the educational computer system (Fig. 2). One can also arrive to that screen from the ECS Hierarchy screen (Fig. 1) by positioning the cursor at the ECS block and clicking the mouse button. This screen shows at the block level how the processor (CPU), memory (MEM), DMA controller (DMA) and peripheral devices (PER1, PER2 and PER3) are interconnected through data (DBUS), address (ABUS) and control (CBUS (RDBUS, WRBUS and FCBUS)) lines of the system bus. The simulation can be carried out at this hierarchical level by activating either Clk+, Ins+ or Prg+ button. What one can follow, then, are the values of signals exchanged at the block level. The values for groups of lines, such as for data (DBUS) and address bus (ABUS) lines, are given in the hexadecimal form, while single lines are colored either in blue or red depending on whether the signal on that line has logical value zero or one, respectively. If one needs more detailed structure of any of the blocks from Fig. 2., he can move one level down in the hierarchy by positioning the cursor at that block and clicking the mouse button. As an example of this one can assume that the cursor is positioned at block CPU on the screen given in Fig. 2 and the mouse button clicked. What appears is the screen giving the block structure of the educational processor (Fig. 3). From this screen one can go one level down and get more detailed structure of any of four units of the Processing Unit (Register File Unit, Executing Unit, Bus Control Unit or Interrupt Service Unit) and the Control Unit. By positioning the cursor at block Register File Unit and clicking the mouse button, one goes one level down in the hierarchy and gets the block structure of the Register File Unit (Fig. 4). The same actions applied this time to the GPR block of screen from Fig. 4., brings the screen of last level in the hierarchy (Fig. 5). Here one can see the design of this block at the level of standard sequential (registers, flip-flops, etc,) and combinational (decoders, logical circuits, etc.) elements.

Such organization of the simulator with the hierarchical structure of screens, makes it possible carry out the simulation of the working of the ECS at various hierarchical levels. At higher levels, one can follow the simulation at the level of signal exchanged between blocks considered here as black blocks. At the lowest level, if it is deemed interesting, one can follow the simulation at the level of registers, flip-flops, logical circuits etc.

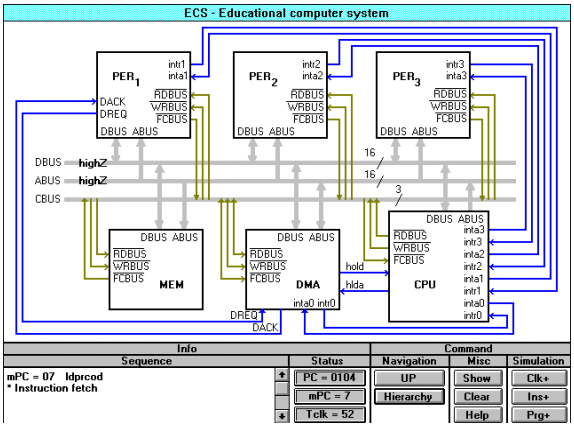

Figure 2. The Educational Computer System screen

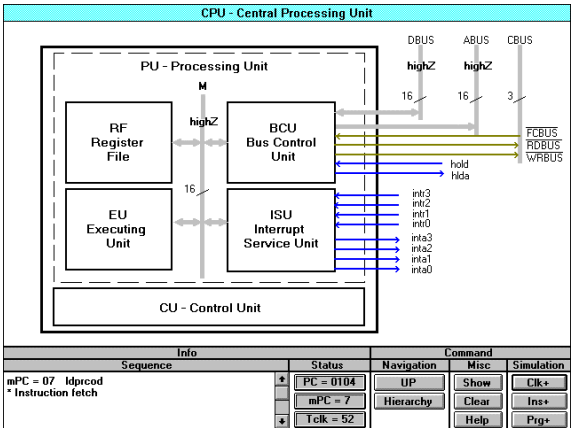

Figure 3. The central processing unit screen

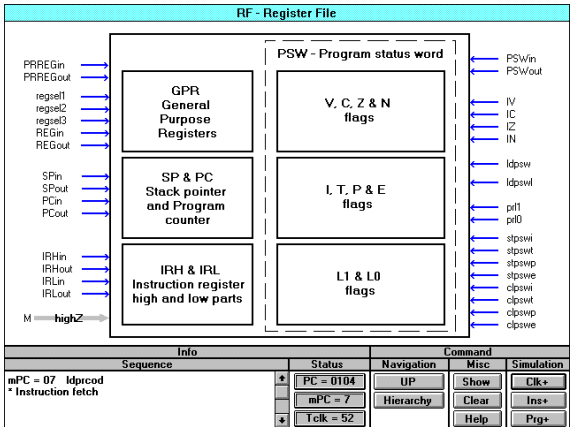

Figure 4. The Register File screen

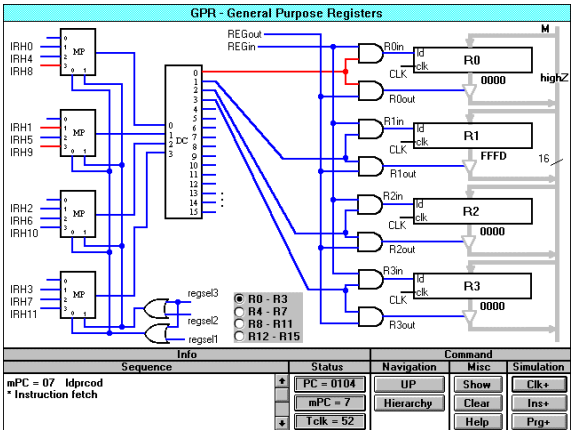

Figure 5. The General Purpose Registers screen

If one find useful to examine and set registers, he can use button Show at any moment during the simulation and get the screen as shown in Fig. 6. By using the same button one can get the timing diagram of selected signals from the beginning of the simulation until the current clock period in the form shown in Fig. 7. for the signals relevant for the realization of the read cycle on the system bus.

4. The organization of laboratory experiments

The practical work with the simulator of the educational computer system is organized through five laboratory experiments the duration of which is two hours. In all five laboratory experiments the students have, by using the educational computer system hierarchical structure of screens, to follow, either at the clock or the instruction level, its working during the execution of earlier prepared example programs that demonstrate the situations of interest and to answer questions concerning some typical situations of the subject covered by the particular laboratory experiment. In addition to that, in the last two laboratory experiments each student is also given a problem which he is supposed to solve independently, write the appropriate piece of program, by using software tools such as the editor, the translator, the linker and the loader, and test it using the simulator. In some cases the students are, also, requested to draw the timing diagrams of some interesting signals and compare them with those obtained by using the show signals option of the educational computer system for drawing the timing diagrams of selected signals (Fig. 7). A brief description of the laboratory experiments carried out by the students is given in this section.

4.1. LAB1: The execution of instructions

The experiment is aimed to demonstrate how the instruction fetch, operand address calculation and operation execution phases for some typical instructions and addressing modes are performed,

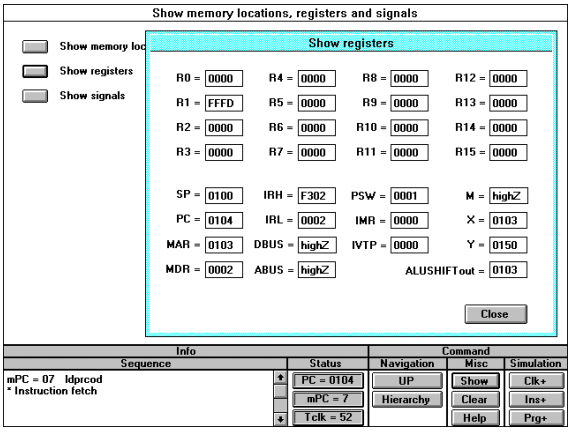

Figure 6. The show registers screen (foreground) and the show memory locations, registers and signals screen (background)

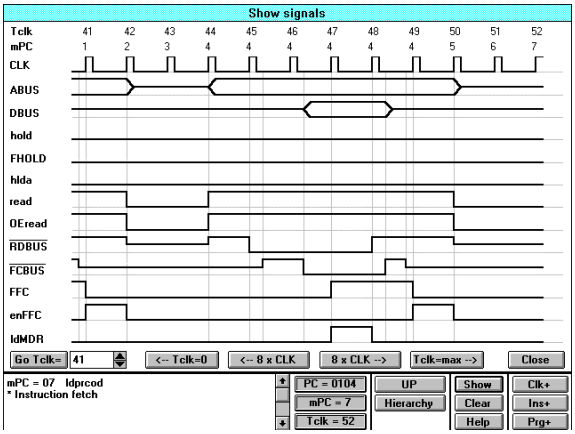

Figure 7. The show signals screen

while the interrupt handling phase is considered separately in LAB 3. To achieve this, the students are requested to run, clock by clock, an example program, which includes the transfer, arithmetic and branch instructions and the register indirect with displacement, immediate and memory direct addressing modes. The program is executed four times each time with the same processing unit and three types of microprogrammed control units and one hardwired control unit. By executing instructions clock by clock the students go through the sequence of control signals generated for the same processing unit by four control units for the instruction phases considered.

4.2. LAB2: The bus arbitration and data transfers

The aim of this experiment is to demonstrate how, first, the arbitration between the processor and the DMA controller, then, the memory read and write cycles between the processor and a memory module, and, finally, the interrupt vector number acquisition cycle between the processor and a peripheral device, are carried out on the asynchronous bus. To achieve this single instruction **add** R1, R2, R1 is executed. It is, also, taken that during its execution an interrupt request from a peripheral device arrives which causes the interrupt handling phase of the instruction to be, also, carried out. As the result, the following data transfers initiated by the processor take place on the bus: one data read cycle to get the instruction, two data write cycles to save the contents of the PC and PSW registers on stack, and, finally, one interrupt vector number acquisition cycle and one data read cycle to get the starting address of the interrupt routine. It is, also, taken that during this time period the DMA controller sends to the processor three requests for accessing the bus. They appear at such moments in time, relative to the requests for accessing the bus generated by the processor, that five typical situations, concerning the synchronization between the processor and the DMA controller in accessing the bus, are demonstrated.

4.3. LAB3: The interrupt mechanism

The aim of this experiment is to demonstrate some typical situations concerning the interrupt mechanism both at the clock and instruction levels.

At the clock level, the sequences of control signals for the interrupt handling phase of an instruction and the execution phase of the return from interrupt (**rti**) instruction are demonstrated. To make it possible the following scenario is prepared. During the execution of the main program, made up of single instruction **add** R2, R2, R3, an interrupt request is generated by a peripheral device. The execution of the instruction continues until its interrupt handling phase is reached, and then, after the checks for whether this interrupt request can be accepted are carried out, the values of the PC and PSW registers are saved on stack, the interrupt acknowledge signal sent to the peripheral device, the interrupt vector table number acquired, and the interrupt routine starting address read from the interrupt vector table and written into the PC register. This causes a jump to the interrupt routine, made up of single instruction RTI. Its execution restores the values of the PSW and PC registers which results in the return to the main program.

At the instruction level some of the situations demonstrated are: the selective and complete masking of interrupt requests coming from the peripheral devices using the interrupt mask register bits and the PSW interrupt enable bit, respectively, the servicing of multiple interrupt requests coming from the peripheral devices according to their priorities, the trap mechanism, the interrupt nesting and the execution of the interrupt instruction (**int**). For each of these situations a simple scenario is prepared with very simple main program and the appropriate interrupt routines.

4.4. LAB4: The program controlled input/output

This laboratory experiment demonstrates the internal structure of a peripheral device controller, the way it is initialized and started, and how the transfer of data takes place using the program controlled input/output. This is done for transfers between both a peripheral device and the memory and the other way around and using both the checking of the status register of the peripheral device controller and the interrupt mechanism. The experiment is organized in two parts. In the first part the students are requested to follow at the instruction level two example programs that transfer a block of data between a peripheral device and the memory using the peripheral device controller status register and between the memory and a peripheral device using the interrupt mechanism. In the second part the students are requested to write, test and successfully run two programs that transfer a block of data between a peripheral device and the memory using the interrupt mechanism and between the memory and a peripheral device using the peripheral device controller status register.

4.5. LAB5: The input/output with the direct memory access controller

This laboratory experiment demonstrates the internal structure of a direct memory access (DMA) controller, the way it is initialized and started, and how the transfer of data takes place using the DMA controller. The transfers of data are illustrated for transfers between a peripheral device and the memory, between the memory and a peripheral device and between two parts of memory with both the cycle stealing and burst modes of operation. As in LAB4, the laboratory experiment is organized in two parts. In the first part the students are requested to follow at the instruction level three example programs that transfer a block of data between a peripheral device and the memory using the cycle stealing mode of operation, between the memory and a peripheral device using the burst mode of operation and between two parts of the memory using the cycle stealing mode of operation. In the second part the students are requested to write, test and successfully run three programs that transfer a block of data between a peripheral device and the memory using the burst mode of operation, between the memory and a peripheral device using the cycle stealing mode of operation, between two parts of the memory using the burst mode of operation.

5. Conclusion

The developed educational environment for teaching a course in Computer architecture and organization, made up of (a) an educational computer system (ECS), (b) a reference manual for the ECS, a software package for the ECS (SPECS), and (d) a set of laboratory experiments, is presented. The ECS is devised to cover the basic concepts included in the course in Computer architecture and organization and its implementation details are described in the accompanying manual. The SPECS provides a user friendly environment for the graphic simulation of the ECS during the execution of a program at the clock, instruction and program levels, with a number of options. The simulation can be carried out either for the programs from the set of devised laboratory experiments or the new ones that one can create using the other tools provided by the SPECS.

The presented educational environment has been used for three years and as a prerequisite for taking an exam in Computer architecture and organization the students have been obliged to carry out successfully the set of laboratory experiments. A lot of students have been taking advantage from the fact that the SPECS runs on personal computers (PCs) and is available free of charge with its reference manual, by using it at home. Based on the number of discussions conducted with the students and the results of the exams, the authors are convinced that the educational environment has been a power aid in teaching the course in Computer architecture and organization and has fulfilled its design objectives.

This resulted in the development of similar environments as a help in teaching topics such memory interleaving, cache memories, virtual memories, translation look aside buffers etc. for some other courses in the field of Computer architecture and organization.

As the result of the work reported in this paper, the research which is under way is directed towards the development of such an user friendly graphic environment that would allow the students to show a great deal of creativity by designing their own computer systems using the library of modules, such as gates, flip-flops, decoders, registers, buses, cache memories, etc.

References

[1] M. Cutler, R. Eckert, "A Microprogrammed Computer Simulator", IEEE Transactions on Education, vol. E-30, no. 3, pp. 135-141, August 1987.

[2] H. Diab, I. Demaskieh, "A Computer Aided Teaching Package for Microprocessor Systems Education", IEEE Transactions on Education, vol. 34, no. 2, pp. 179-183, May 1991.

[3] W. Henderson, "Animated Models for Teaching Aspects of Computer Systems Organization", IEEE Transactions on Education, vol. 37, no. 3, pp. 247-255, August 1994.

[4] J. Djordjevic, "Computer Architecture and Organization –The reference manual for the Educational Computer System", Faculty of Electrical Engineering, University of Belgrade, 1995.

[5] V. Milutinovic, "Surviving the Design of a 200 MHz RISC Microprocessor: Lessons Learned", IEEE Computer Society Press, 1997.