A Training Software Model of von Neumann Register Machine with Two-Operand Instructions

Orlin Tomov, Angel Smrikarov

Abstract: *This paper describes a software simulator for distance and e-Learning purposes. Its purpose is introducing the students to the details in the General Purpose Register architecture, with accent of the self-education. The model is also useful, when evaluation of the students' knowledge is needed. The focus is on the real-time execution and debugging of simple programs, written in a very basic assembler language. The hardware is decomposed to register level, and the instructions are executed by sets of microoperations.*

Key Words: *Computer Systems and Technologies, Model, Assembler, Microoperation, General Purpose Register, Computer Architecture.*

INTRODUCTION

Computer based simulators often replace the traditional real objects in the areas of scientific research, testing, evaluation, and education. The advantages of this approach are obvious:

- reducing the costs of the research;
- applicable in almost every environment at every time and every place [4];
- representing abstract levels of the examined object's decomposition - which is practically impossible in most cases when real objects are used.

It is a well known fact to educators that successful engineering education is achieved through involving the students in practical activities such as design projects and laboratory assignments [5]. The next question is whether or not distance education can provide virtual students with the same opportunities and equal access to lab facilities when compared to their standard classmates. The concept of the virtual lab arises in this context trying to offer a solution for this problem. Through virtual labs, students will have access to educational and research facilities in order to practice their skills executing practical experiments in virtual environments using simulation software.

The virtual laboratory on "Computer organization", developed at the University of Rousse [6] includes training software models of the basic components of the processor – ALU, Control Unit, CACHE and Interrupt System. In addition it includes a model of a hypothetical single address von Neumann processor. In order to be extended and deepened the knowledge of the students it is necessary to be developed training models of different types of processors – a stack machine, an accumulator machine, a register machine. This paper describes an approach to development of register machine simulator.

GPR ARCHITECTURE AND SIMULATOR REQUIREMENTS

The model represents a hypothetical von Neumann machine with two-operand instructions at a register level. After specifying the requirements to the training model an exemplary architecture should be selected as a basic prototype of the simulator. The GPR architecture, described in [2] is the most appropriate solution because of its simplicity and detailed description in the literature.

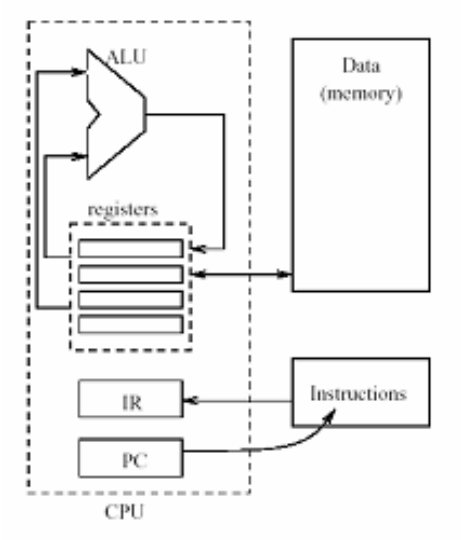

Figure1 – GPR Architecture

This architecture, shown on fig. 2 is typical and includes a set of several registers (GPR file), for general purposes. Two types of this architecture are considered in [2]. The first one is Register-Register type, or so-called "Load - Store" architecture, because the access to the memory is implemented only with Load and Store instructions. The second type is Register – Memory, because here the arithmetic and logic instructions can use operands from the memory. [2]. The developed simulator should be built as a "Load-Store" machine. It has to be comprehensive, easy for work, imposing minimal system requirements for hardware and software environment.

The application size has to be small and to enable fast downloading from Internet, that will make the model applicable for E-learning and will facilitate its embedding in web based courses on the disciplines "Computer Organization" and "Computer Architectures".

The simulator should be helpful tool for understanding the basic concepts of the register architecture, and to be an easy to use debugger for programs, written in a simple assembler.

The model should illustrate in details the idea of the selected architecture, being at the same time not too complicated for the students. It means that the student should not learn how the model works but contrariwise – the model has to show to the student how the represented architecture operates. In other words the model is intended to "explain" the lesson to the student.

SIMULATOR DESCRIPTION

The simulator was implemented with "Borland Delphi 5.0" and works under Win9x, NT, and newer versions. It has also been tested under Linux with the Wine emulator.

When the application starts, the users have access to the program-entering window. Since this is a hypothetical processor, the assembler is very simple, and contains only the most important instructions. (fig. 2)

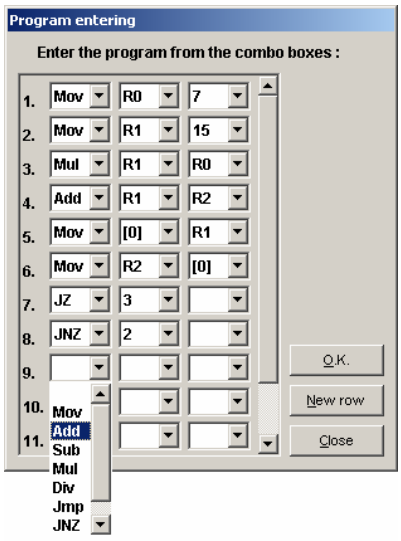

Figure 2. Program entering window

The program entering is very intuitive. It is not necessary to learn the assembler instructions and syntax in advance. Each instruction is selected from a combo box. If the user has ever used any assembler language, or even just saw a manual, he (she) will be able to find out the meaning of the instructions. The content of the combo box for the first address is dynamically updated, when an instruction is selected. After that, the combo box for the second address is updated. The update values are the only possible and correct for the current instruction. With this approach, even if the user is not familiar with the assembler programming, it will be impossible for him (her) to enter a wrong syntax construction, or a wrong address.

After this step, the user has an access to the main window. (fig. 3)

There is a small window, called "Program", that shows the program code and highlights the instruction that is currently being executed, and the instruction, pointed by the program counter.

The right side of the main window is used for selection of the current action. Each action refers to an instruction, a group of instructions or initializes the system. When selected, the window in the corner shows a short description of the instruction and the microoperartions included in it.

At the centre are placed the CPU modules. At the right side is placed the primary storage (RAM memory).

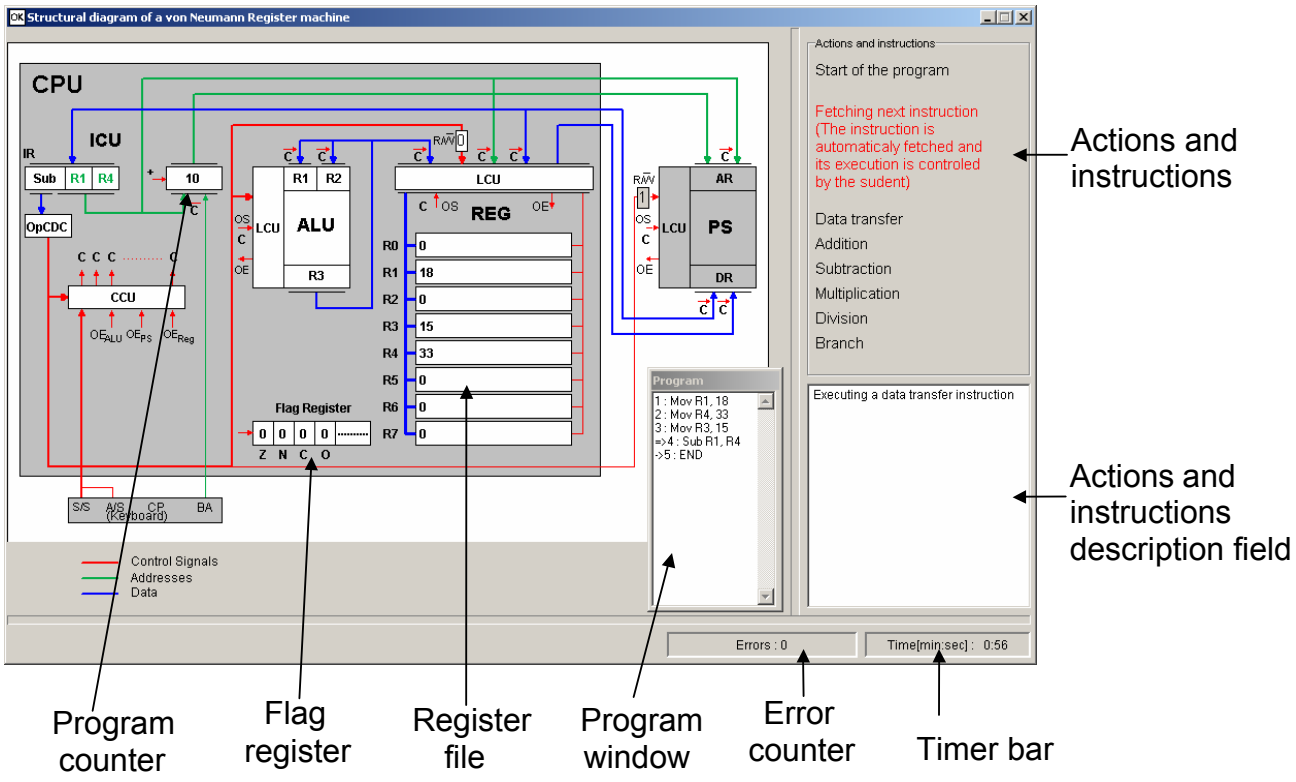

Figure 3. Main application window

The represented modules are :

- ICU – Instruction Control Unit – includes the Instruction Register (IR), Program counter, OpCode decoder (OpCDC) and Command Control Unit (CCU);
- ALU;
- Flag Register;
- Register Block (REG) – includes the register file (R0 – R7) and the Local Control Unit (LCU) of the register block;
- Keyboard.

At the bottom of the window are placed timer and error counter, which are reset, when a new simulation is started.

Execution of the source code is interactive. The student has to initialize the machine at first. To do this, he (she) has to send signals from the keyboard respectively for setting the start address, setting the mode (automatic/step-by-step) and to send the start signal. Sending each correct signal is simulated with an animated blinking line between the modules. If an error code was sent, a message informs the user for the error. At each third error on a given action, the error message containing information about the next correct step appears. This allows the usage of the simulator as a virtual guide.

During the execution, the user can "debug" the program. The contents of the registers in the register file, the registers of the ALU, the Flag Register, Program counter and the Instruction register are viewable in real-time.

As an example, here is considered the execution of the instruction "Add R1, R2". In this case the user has to submit the following actions, after the simulator is initialized:

- Selecting the instruction "Addition" from the "Action and Instructions" panel;
- Sending a signal for transfer the address of the first operand from the instruction register (IR) to the LCU of the register block;
- Setting read mode, and sending a start-operation signal in LCU of the register file;
- Sending a signal for receiving the first operand in the input register (RI1) of the ALU;
- Sending a signal for transfer the address of the second operand from the instruction register (IR) to the LCU of the register block;
- Setting read mode, and sending a start-operation signal in LCU of the register file;
- Sending a signal for receiving the first operand in the input register (RI2) of the ALU;
- Sending an operation-start signal in the ALU;
- Sending signal for transfer from the result register (RO) of the ALU to the LCU of register block;

Setting write mode and sending a start-operation signal in LCU of the register file.

Sending all of these signals requires only clicking on the "C"-fields on the main form, corresponding to each signal. The correct signals are followed by an animated line, between the modules. After each microoperation, the user can see the changes of the registers' content – for example GPR registers, Flag register, ALU registers, Program counter and Instruction register.

CONCLUSIONS AND FUTURE WORK

As the experience shows, the interactive training models are very easy to understand from the students' perspective, which makes them useful for creation of virtual laboratories and e-learning courses.

Work in the future:

- Building a set of models of different architectures;
- Making the models more flexible with enhanced set of assembler instructions.

REFERENCES

[1] Смрикаров,А.С., Ст.Р.Смрикарова, Х.М.Авакян, А.С.Василева. Организация на компютъра. Русе, Авангард принт ООД, 2002 г.

[2] Hennesy & Paterson – Computer Architecture. A quantitive approach. 3-rd edition

[3] http://www.marcocantu.com/

[4] Hristov T., S. Smrikarova, A. Vasileva, A. Smrikarov. An Approach to Development of an e-Learning Software Platform, International Conference on Computer Systems and Technologies – CompSysTech, 2002.

[5] Licks,V., M. Quiroga, R. Jordan, J. S. Correa, Building Virtual Laboratories - A Web Based Experience Proceedings of the Internatioonal conference ITHET2001,Kumamoto, JAPAN, July 4-6, 2001.

[6] Vasileva,A., A.Smrikarov, T.Hristov. A Conceptual Model of a Virtual Laboratory on "Computer Organization". Proceedings of the ***CompSysTech'2002***, Sofia, 20-21 June 2002.

ABOUT THE AUTHORS

Orlin Tomov, Department of Computer Systems, University of Rousse, Phone: +359 82 888 276, E-mail: OTomov@ecs.ru.acad.bg.

Assoc.Prof. Angel Smrikarov, PhD, Department of Computer Systems, University of Rousse, Phone: +359 82 888 743, E-mail: ASmrikarov@ecs.ru.acad.bg.