13th International Research/Expert Conference
”Trends in the Development of Machinery and Associated Technology”
TMT 2009, Hammamet, Tunisia, 16-21 October 2009

SIMULATION OF THE TFACO PROCESSOR

Nebojša Lj. Stanković, Dr. Siniša S. Ranđić
Technical Faculty, Čačak
65, Svetog Save, Čačak
Serbia

ABSTRACT
This paper describes an application which provides the simulation of the functioning of the central processor (CPU). The simulation provides visual representation of anything occurring both at the level of arithmetic logic units (ALU) for different arithmetic and logical operations and at the level of bits. Visualization is a very efficient teaching method which makes it possible for students to acquire new knowledge successfully, and that is the reason why this method has been increasingly applied in the process of education in recent time. The application is realized using the Microsoft Visual Basic 6.0. programming language.
Keywords: central processor, arithmetic logical unit, simulation, visualization, education.

1. INTRODUCTION
As computer systems have been developed, it is necessary to acquaint as many users as possible with complex operations that occur in the computer. Consequently, along with the development of computers, the simulation techniques whose goal was to illustrate 'how a computer works' were also being developed. The processes occurring in a computer, particularly the functioning of the central processor unit, memory, bus, and input/output units are visually represented. Computer system simulation had two main tasks: to educate users to understand what processes occur in the system and to enable research aimed at further development, using the simulation of specific situations which are important for the computers system [1, 2].
This paper describes an application which provides the simulation of the arithmetic logic unit (ALU) as an integral part of processor TFaCo [1]. The simulation provides visual representation of anything occurring both at the level of arithmetic logic units (ALU) for different arithmetic and logical operations and at the level of bits.

2. PROCESS SIMULATION
Simulation has been developed as a group of five applications which visualize how the central processor works, starting from simpler processes and than moving on to more complex ones (Figure 1).
The first two applications are simulators intended for users with elementary knowledge, and they visualize the synthesis of the computer architecture. The third and the fourth applications are demonstrations of how the program sequence works with and without memory. The fifth application is a simulator intended for advanced students. It represents the solution to processor TFaCo architecture created by the authors of this paper. All these applications offer a graphic interface for the users and provide plenty of commentary and visual effects.

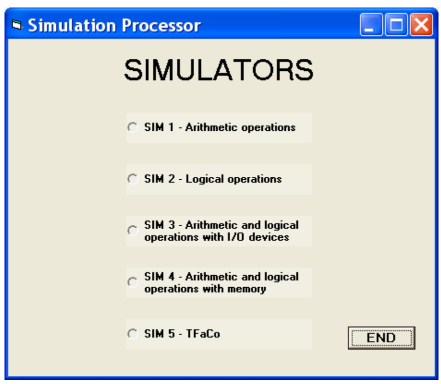

Figure 1. Selection of Simulation

The program is created in the Microsoft Windows XP Professional operating system and it uses the MS Visual Basic 6.0 [3] programming language. The hardware used for executing and testing the simulation is Pentium 4 with RAM of 512 MB and an integrated graphic card.

2.1. SIM1 - Arithmetic operations

This is a simulator that does not require previous knowledge and its task is to explain it to the user how CPU executes arithmetic operations at the level of bits. It should familiarize users with the binary language of computers. The user enters the selected whole 16.bit binary numbers, selects one of three mathematical operations (addition ADD, subtraction SUB or multiplication MUL), confirms the button WRITE and then selects an arithmetic operation. In order to execute these arithmetic operations, the numbers are first moved to the first complement and then to the second one. Figure 2 presents the layout of the window after the selection of arithmetic operation of addition (ADD) and moving the numbers A and B into the first, or the second complement.

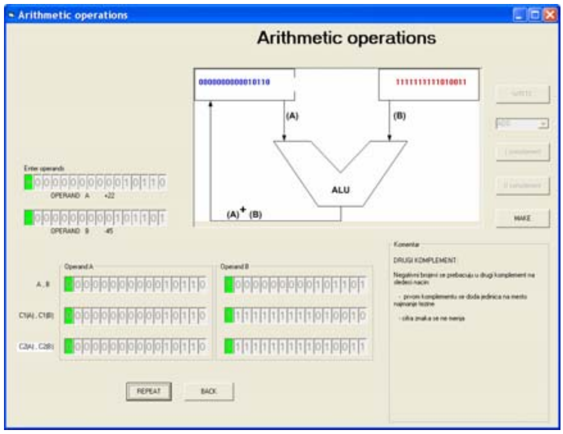

Figure 2. SIM 1 - Arithmetic Operation ADD (Complementing)

Now the operands are ready to execute the operation of addition. After the confirmation of the button MAKE, the window ADD NUMBERS is opened. Addition of the two numbers written in the second operand is simulated in three steps. First (STEP1), numbers are written in the registers, then, (STEP2) the digits of the same weight are added, including the transfer. At the beginning of the simulation the bit of the first operand is illuminated, as well as the bit of the same value of the second operand whereupon the result is written in the bit of the same value of the result. The realized transfer is written down as well. After the completion of the second step the button STEP3 is confirmed, which gives the layout of the final result. The final result is obtained by removing the transfer bit from the result whereupon it is checked whether any exceeding occurred (Figure 3).

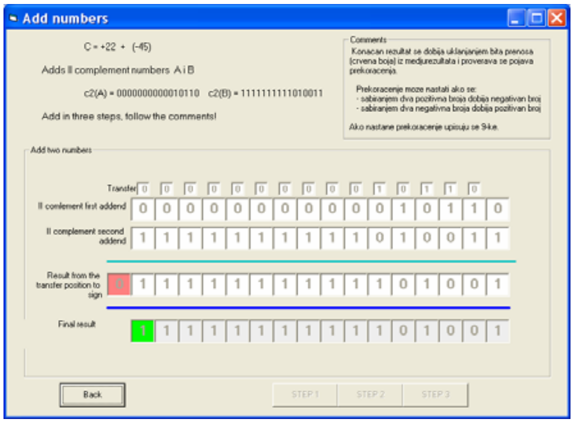

Figure 3. SIM 1 - Arithmetic Operation ADD (Final Result)

2.2. SIM2 - Logical operations

After starting this simulator, the user must select one of the offered logical operations. The table of truthfulness appears for the selected operation, along with its graphic symbol, and the process of simulation begins. The bit of the number (operand) A is illuminated whereupon the corresponding result of the selected logical operation from the table of truthfulness (column Z) is illuminated. Finally, the value equal to the previously illuminated result is written in the register Z. Figure 4 shows the final result of the logical operation OR, which is applied to operands A and B.

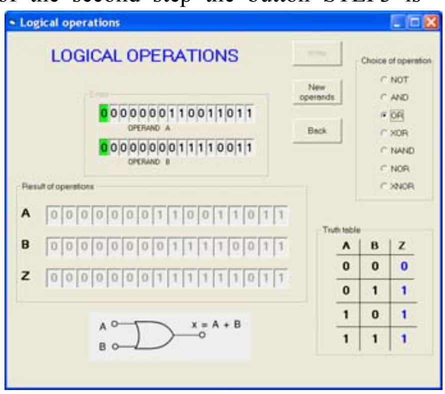

Figure 4. SIM2 - Logical Operation OR

2.3. SIM3 - Program sequence without memory

Using the example of addition of two numbers, this simulator demonstrates to users stepwise transfer of data from the input devices, via ALU, to the output devices, but without using memory (Figure 5). To demonstrate the addition of two numbers, numbers 23 and 25 are used. Their binary symbols are 10111 and 11001. After the confirmation of the button START, demonstration starts and proceeds in eight steps. The action occurring within each step is monitored in "STEPS", while in the demonstration picture every step is labelled by numbers 1, 2 …. For example, during the step 6 numbers are entered in ALU, so that during the following step 7 numbers are added and the result is written in register A.

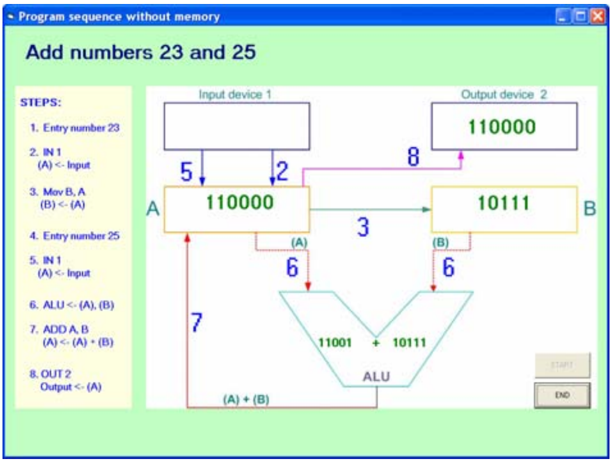

Figure 5. SIM 3 - Program sequence without memory

2.4. SIM4 - Program sequence from memory

Using the example of the addition of the same numbers, as in the case of the third application (demonstrator), this is a demonstration which gives a detailed visualization of how CPU functions, introducing memory, instruction register (IR), programming counter (PC) and memory address register (MAR).

After the confirmation of the button START, the execution of the simulation begins. The beginning value of the programming counter is the address of the first instruction (LDAC 0). MAR accepts the address from the PC and finds the corresponding memory location. The content of the memory location is sent to IR,it is then decoded, and, depending on the instruction, the action is executed. As the first instruction is LDAC 0, the memory content from the location 0 should be forwarded to the accumulator. The instruction is executed in two memory cycles. Upon the completion of this, instruction number (23) is in the accumulator. In order to add the two numbers, the accumulator ACC and the register R are used. The first number, which is in the accumulator, should be saved and added to another number. It is for this reason that the first number is forwarded to the register R, using the instruction MVAC. The accumulator is then loaded with another number (25), using the instruction LDAC 1. In this manner, the first number is placed in the register R, while the other one is in the accumulator. The following instruction (ADD) is addressed, IR is sends the signal to ALU to add numbers which are in the registers (ACC and R). After the execution of the instruction of addition, the result should be sent to memory at the location 2, using the instruction STAC 2. This is the last instruction of the program written in the memory. Figure 6 gives the final view after the realization of the program.

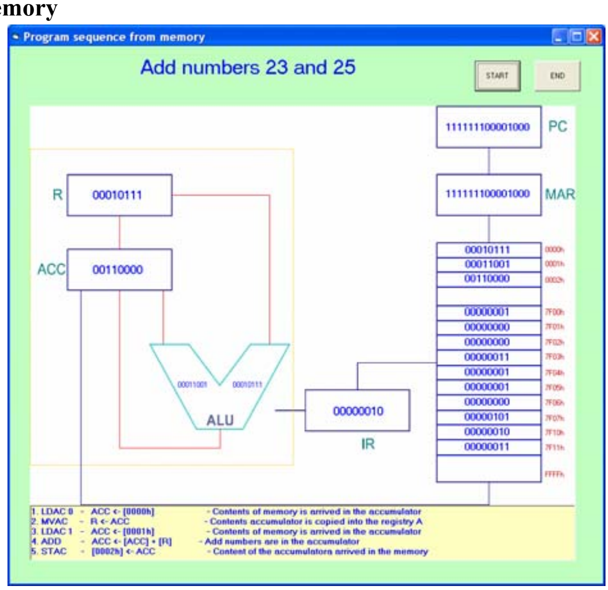

Figure 6. SIM 4 – Program sequence from memory

2.5. SIM5 - Simulator TFaCo

Simulator TFaCo is intended for advanced students, and represents an offered solution to one processor architecture, designed by the authors of this paper. All simulations offer a graphic interface for users and provide plenty of commentary and visual effects. In order to provide proper work, it is necessary to enter instructions (program) which the processor should execute and write in the memory. Two input methods are offered: input instruction by instruction (STEP TO STEP) or loading the program from the textual database (FILE). After the instructions have been written in the memory, they should be obtained and interpreted. The execution of the simulation can be described in several steps: calculating the instruction address, reading the instruction from the memory, decoding the instruction's operational code, executing the operation which is required by the operational code, determining the value of the indicator in the status register. After the operation has been executed and the indicator has been written in the register RS, the procedure is repeated for the following instruction (reading, decoding, executing, and writing). The simulation proceeds according to the sequence of the instructions written in the memory. The layout of the window after the execution of the last instruction is given in the Figure 7.

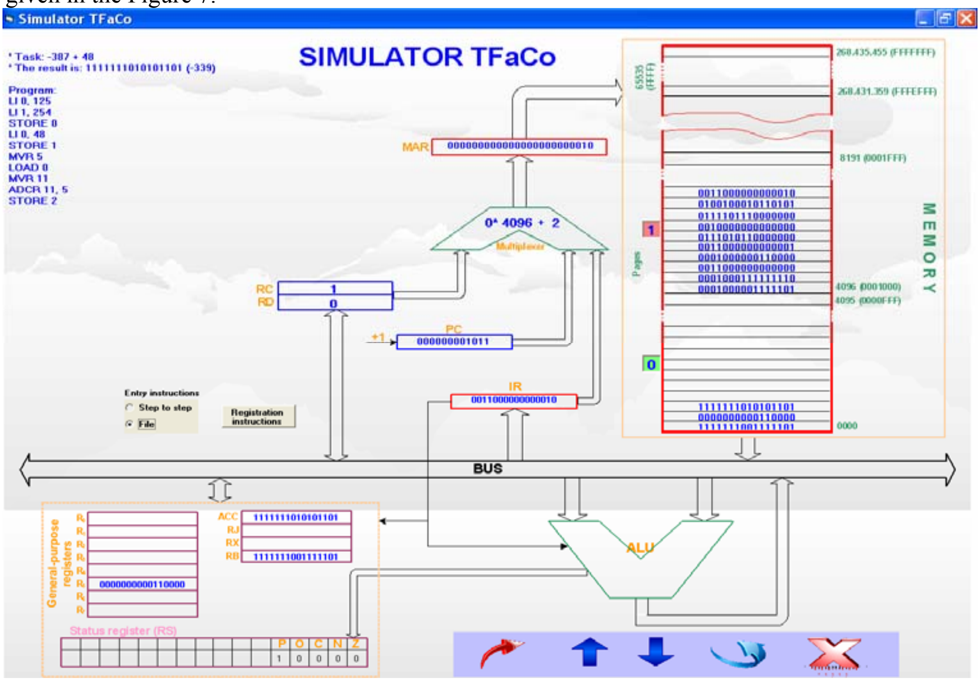

Figure 7. SIM 5 - Simulator TFaCo

3. CONCLUSION

This paper describes computer architecture simulators which, using a16-bit processor, realize different arithmetic and logical operations and also load/execute different programs which are in the form of a sequence of commands in the internal memory (RAM). Developed simulators make it possible for the users to understand how the computer system functions because they start doing certain exercises with the simpler simulators and than move on to the more complex ones. In this manner, users acquire some fundamental knowledge which covers this very otherwise very complex area.

4. REFERENCES

[1] Stanković N.: Prilog simulaciji računarskih arhitektura, Magistarski rad, Tehnički fakultet, Čačak, 2009.,

[2] Đorđević, J.: Arhitektura računara, Edukacioni računarski sistem, Arhitektura i organizacija računarskog sistema, ETF, Beograd, 2003.,

[3] Maxvell, T., Scott, B.: Visual Basic Super Bible, Corte Madera, California, 1992.,