SOFTWARE SIMULATOR FOR A PRESUMPTIVE MICROPROCESSOR

NEDELCU DRAGOŞ-IULIAN[1*], NEDELCU GRIGORAŞ-DORINEL[2]

[1] *"Vasile Alecsandri" University of Bacau, Calea Marasesti 156, Bacau, 600115, Romania*

[2] *Petroleum – Gas University of Ploieşti, Bulevardul Bucureşti 39, Prahova, 100680, Romania*

Abstract: The paper wants to develop a software simulator for a presumptive microprocessor, to allow better understanding of the architecture and general structure of the microprocessor, as well as the functional model of such an integrated circuit with major importance in a computing system, due to its functions.

Keywords: simulator, microprocessor, integrated circuit

1. INTRODUCTION

The degree of complexity and variety for the operations which can be performed by the microprocessor defines the possible computing power and performance for the microprocessor.

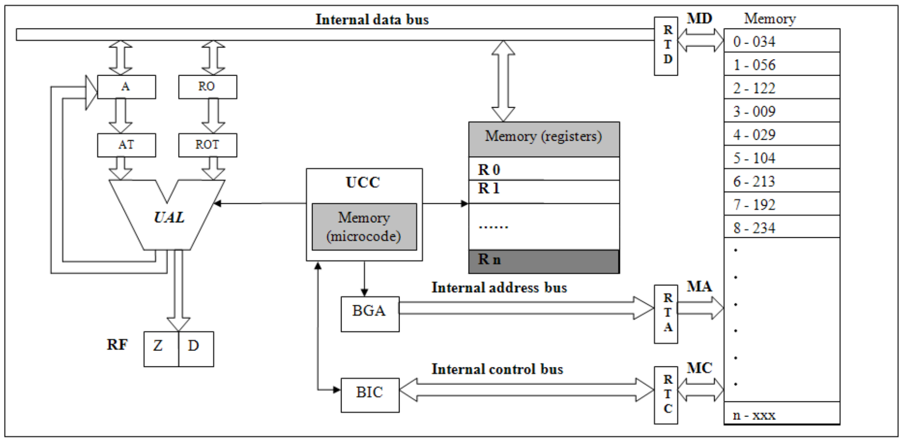

Fig. 1. Architecture for the presumptive microprocessor: A – accumulator register; RO – operand register; AT – temporary accumulator register; ROT – temporary operand register; UAL – arithmetic-logic unit; UCC – command and control unit; Rn – internal general registers; BGA – address generator block; BIC – block interface – control; MA – external address bus; RF – flags register; MD – external data bus; RTD – data buffer register; RTA – buffer address register; RTC – buffer control register; MC – external control bus.

[*] Corresponding author, email: iulian.nedelcu@ub.ro

The sum of all possible operations: between registries, memory operations, arithmetic-logic and other special operations, which activate different constructive parts of the microprocessor, compose the instruction set for the microprocessor. A simple modification of one or more instructions, leads to different functionality from the system [1].

In order to better assess the correlation between the operations which are performed to process some instructions [2], we will schematically present the informational transfer and the operations performed in each clock period, starting from the architecture for the presumptive microprocessor (Figure 1).

2. SOFTWARE SIMULATOR APPLICATION

The application is written using Borland C++ Builder Version 6 programming language. C++ Builder is a C++ development environment produced by Embarcadero Technologies. Besides the standard C++ library it contains the VCL libraries for visual components. According to the architecture and the functional principle of a presumptive microprocessor, a program was created using Borland C++ Builder (V.6) in order to simulate an instruction performed by the chosen microprocessor [3].

Using the developed program we choose the adding instruction from the four instruction set (adding, subtracting, multiplying and dividing) which will be performed in the program. First of all we choose the instruction, the operands and the destination for the result, and then we will perform the selected instruction, and at the end the result will be placed in the specified destination [4].

Therefore, we will present the execution of the ADD instruction (out of the set of four: ADD, SUB, MUL and DIV) for adding operand 1, which is accessed from the memory (Memory n-xxx) and operand 2 from the Rn register. The result from processing the information by the UAL will be placed in the accumulator register and then transferred back into the memory, using memory busses, into the specified memory location. This is just a single instruction out of the 2-3 billion instructions performed by an average microprocessor in a second.

Figure 2 presents the active areas which lead to loading the accumulator register with the selected memory area (3-009). The command and control unit transmits using address and control busses a command signal to the memory unit. From the memory unit, using the internal data bus, the information is transmitted and kept in the accumulator register.

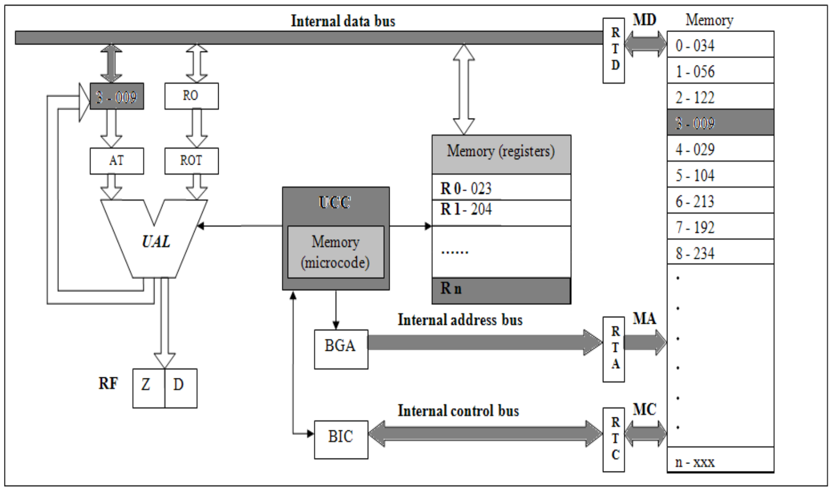

Fig. 2. Loading the accumulator register (A) with operand 1.

In Figure 3, loading the operand register with operand 2 (R1-204), takes place by transmitting the command signal performed by the command and control unit over the general internal registers and transmitting the selected register into the operand register using the internal data bus.

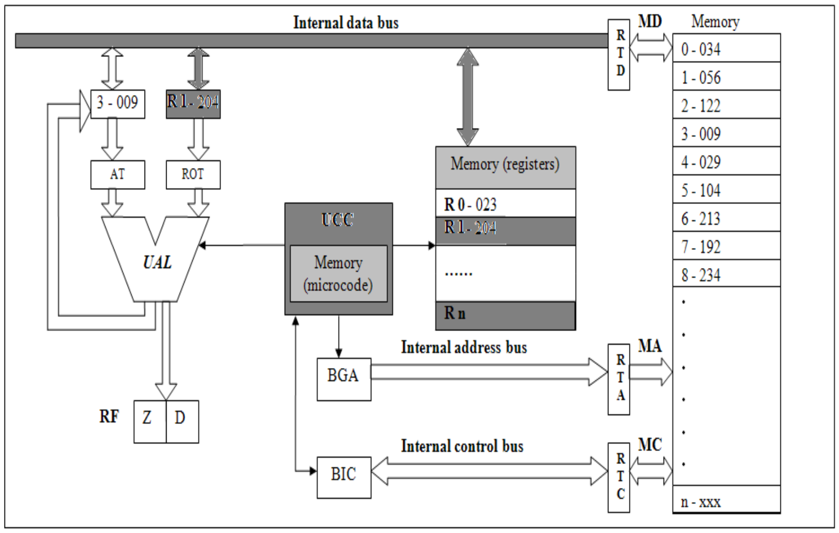

Fig. 3. Loading operand register (RO) with operand 2.

After unloading the accumulator register with the memory address 3-009 and loading the operand register with register R1-204 they are transferred to command by the temporary accumulator register and the temporary operand register (Figure 4).

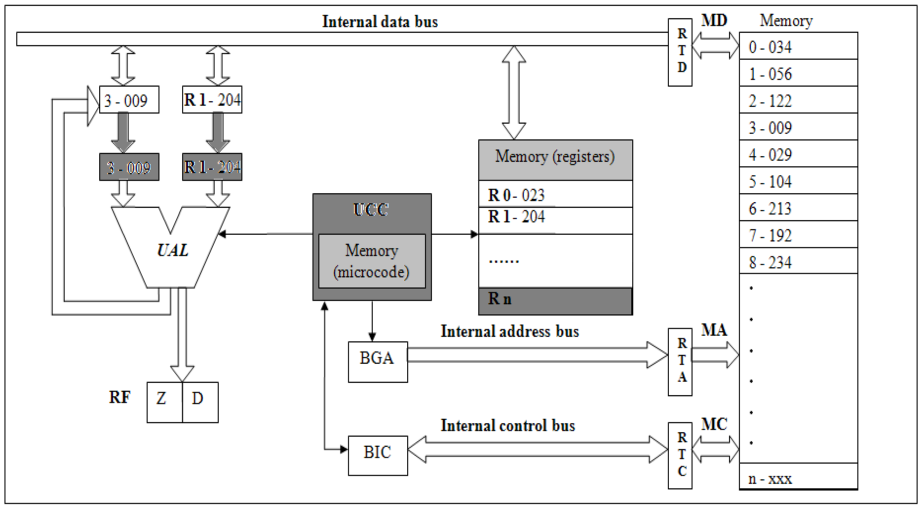

Fig. 4. Transferring to command of the data in the temporary accumulator register (AT) and the temporary operand register (ROT).

The control unit sends a signal to the arithmetic-logic unit which has available operands, and after completing the operation (in the given example the adding operation of the operands takes place) the result is loaded into the accumulator register (Figure 5).

Cases in which the result can suffer modifications:

- if the result of the operation is ≤ 255, the result is transferred to the accumulator register;
- if the result of the operation is =256, the value 0 is transferred in the accumulator register, and in the flags register null and overflow is declared;
- if the result of the operation is > 256, in the flags register (RF) overflow (D) is declared, and the rest from dividing by 256 is transferred into the accumulator register;
- if the result of the operation is = 0 (in any of the above conditions), the result in the accumulator register is null, and null is declared in the flags register (Z) (if the result from dividing by 255 is null, overflow and null is declared in the flags register).

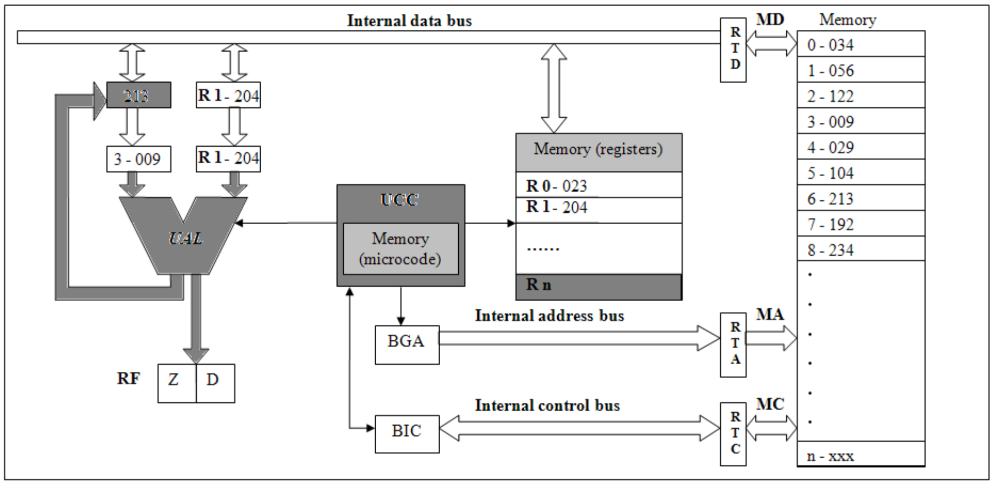

Fig. 5. Performing the operation by the arithmetic-logic unit (UAL) and sending the result to the accumulator register (A).

After performing the operation between the two operands by the arithmetic-logic unit, the result from the accumulator register is transferred into the memory (the given example – Figure 6) using the command and control unit and the internal and external busses into the specified location (example: 6-213).

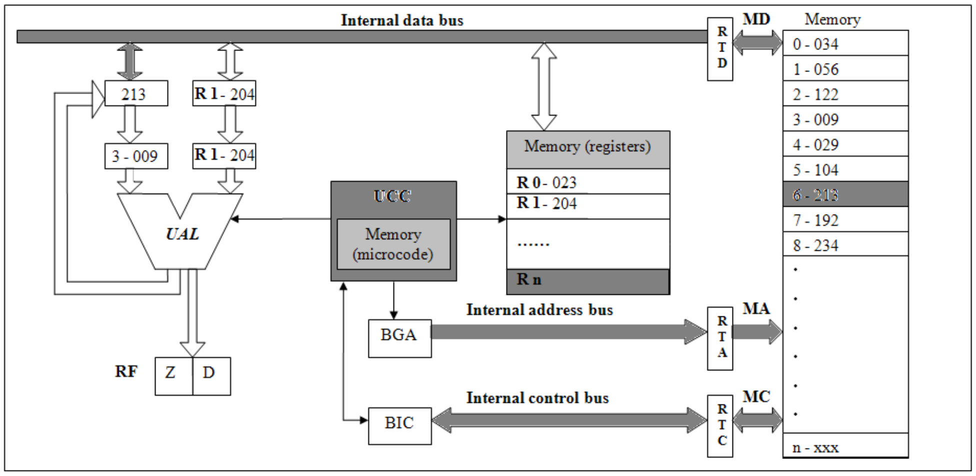

Fig. 6. Transferring the result from the accumulator register (A) into the memory address.

3. CONCLUSIONS

The proposed application represents a completely useful example, from a teaching point of view, by studying the internal functioning of a presumptive microprocessor (standard), by understanding some structure and architecture notions: registers, busses, arithmetic-logic unit, command and control unit, execution cycle for the microprocessor, memory.

The application is easy to use and intuitive due to the simulator structure at graphic interface level. The simulator is designed using an open architecture, which allows later adding of new instructions in the existent set.

The paper can be a solid start point to study, understand and practical use other types of microprocessors widely used in industrial applications.

REFERENCES

[1] Rădulescu, G., Elemente de arhitectură a sistemelor de calcul. Programare în limbaj de asamblare, Editura Matrix Rom, Bucureşti, 2007.
[2] Francisc, I., Sisteme multiprocesor, Editura Victor, Bucureşti, 2000.
[3] http://ro.wikipedia.org/wiki/C%2B%2B_Builder (12.03.2011).
[4] Kreindler, L., Bazele microprocesoarelor, Editura Matrix Rom, Bucureşti, 1997.