ViReC PROJECT:
LABORATORY WORK FOR COMPUTER ARCHITECTURE-
THE INSTRUCTION SET OF A SIMPLE PROCESSOR VIP8

Oleg Cernian, Eugen Dumitrascu,
Adrian Neatu, Dumitru Ingeaua, Cornel Mitroi
University of Craiova
Faculty of Automation, Computers and Electronics
5, Lapusului Road, Craiova, Romania

Timothy Hall
ECE Department, University of Limerick
Limerick, Ireland

Abstract: The basic idea developed in the "ViReC e-Initiative" project, which is a component of MINERVA scheme, consists in realizing some virtual on-line labs. Computer Architecture discipline is one of the most important disciplines for students attending the strand of computer specialization. A set of specific laboratory assignments was proposed for this discipline.

One of the designed laboratory works is entitled "The instruction set of a simple processor VIP8". It is based on the structure of a hypothetical simple processor called VIP8. The structure of the computer that uses this kind of processor is similar to the Elementary Digital Computer presented at the course. In this paper it is discussed the functioning model of the processor and the instruction set incorporated.

The students can use a simulator of this processor with which they can write programs in the machine code language, visualize the execution of the instructions including step-by-step mode, display the configurations of internal registers.

The execution of each instruction of the processor was detailed in steps and presented in the animated form of a short movie.

Also, the simulator offers the possibility to run examples of already written programs, exhibiting thus a powerful tool to students for writing their own applications.

Keywords: ViRec project, Minerva Scheme, Processor VIP8, Computer Architecture, Elementary Digital Computer

1. THE DESCRIPTION OF THE ELEMENTARY COMPUTER STRUCTURE

It is considered the structure of a very simple digital computer consisting of only 4 units: Arithmetic Logic Unit (ALU), Control Unit (CU), Memory Unit (MU) and Input/Output Unit (I/O U). The link between these units is realized by two buses: Address Bus on 16 bits and Data Bus on 8 bits.

Arithmetic Logic Unit (ALU)

ALU consists of two distinct areas: the processing device and the group of local registers. Two arithmetic operations are implemented on 8 bits: addition and subtraction. The set of registers in the following: A (accumulator), B, C, H, L (general registers), PSW (flag register), all on 8 bits and SP (stack pointer) on 16 bits. Some of these registers are grouped in pairs, like HL and BC.

The pair HL can be used for implicit addressing of memory, as in this pair of registers is inputted the memory address. The flag register PSW contains the following flags: CY (Carry), Z (Zero), P (Parity), S (Sign).

Beside visible registers, there are provided two additional working registers, designed W and Z, that are used in implementation of arithmetic operations and are not directly accessible to the user.

Control Unit (CU)

CU contains the instruction register (IR) on 8 bits, program counter (PC) on 16 bits, which gives the next instruction address that is to be executed, the control block, which generates the commands to all other units for implementation of the function, and a logical decoder that decodes the opcode of the instruction ensuring the interpretation of the current instruction.

Memory Unit (MU)

MU contains a memory block of 64 Kbytes (2^{16} memory locations, each location of 8 bits). The range for the memory addresses is from 0 up to 2^{16}-1. MU contains also two functional registers: Memory Address Register (MAR) on 16 bits and Memory Buffer or the Data Register of the Memory on 8 bits.

Input/Output Unit (I/O U)

The I/O unit consists of max 256 I/O ports, where the number of the ports is belonging to the interval [0, 255]. Each port is considered a register on 8 bits. The input/output ports are communicating with the processor through the 8 bit Data Bus for data transmission, while addressing is realized through the 16 bit Address Bus.

The operation of this structure is realized sequentially congruous to von Neumann's principles.

2. PRESENTATION OF THE INSTRUCTION SET

The instruction set adopted for the processor is divided into the following five groups:

1) Data Transfer Group
2) Arithmetic Group
3) Logical Processing Group
4) Branch Group
5) Stack, I/O and Machine Control Group

Each group consists of a number of specific instructions realising a particular operation in a determined sequence of steps.

For each instruction it is assigned a particular mnemonic that is very common. The list of all mnemonics is depicted in Fig. 5.

3. DETALIED INSTRUCTION EXECUTION

The execution of instructions of the processor VIP8 are presented in a detailed manner in a graphical mode created in Macromedia Flash. For each instruction the student can set the values of some registers or memory locations and, afterwards, he/she can view all the steps of the execution, which can be framed in subphases.

In the next figure there is presented the instruction cycle for "MOV A, H" instruction. It obviously belongs to the Data Transfer Group. The instruction ensures the transfer of the content of the register H into the register A:

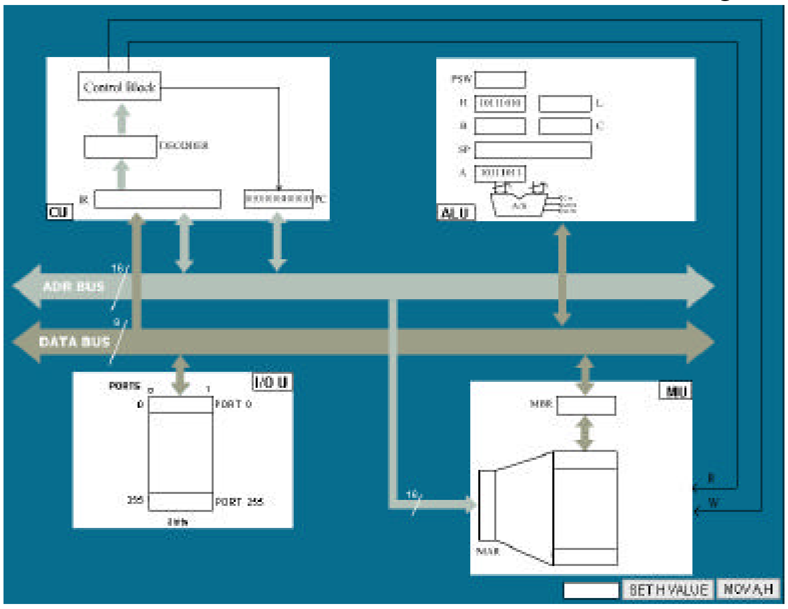


Fig.1. Capture of the execution of instruction MOV A, H

The student can preset the value of the register H (the sender register) before starting the instruction cycle. This value must be formed only of 0-s or 1-s (binary digits). After that running of the entire instruction is initiated comprising the steps of the "fetch phase" and "execute phase".

After the settings are completed, the fetch phase begins. The content of PC (Program Counter) is transferred using the ADR BUS (Address Bus) into the register MAR (Memory Address Register). The Read command is issued by the Control block from the CU and a Read Cycle is started ensuring reading of the memory array at the location pointed by the content of MAR. The read group of 8 bits corresponds to an instruction, which is saved in the register MBR, from which it is sent to the Instruction Register (IR). The content of IR (Instruction Register) is decoded by the "DECODER". The Control Block will increment the PC to point to the next instruction to be fetched and the process switches to the execute phase. In this phase the content of the register H will be transferred into the register A. After this operation the instruction ends, and normally a new fetch phase will be initiated, in the next instruction cycle.

After execution of "MOV A, H" instruction it can be seen that the content of the Accumulator becomes identical to that the register H. The content of H is not lost.

Another instruction from the Data Transfer Group to be described is LXI H, d16, where d16 represents a 16 bit word (data). This instruction loads the register pair HL with d16, which is included in the instruction itself. As it is known, since data is associated to the instruction, such instructions are called immediate. Obviously, the length of this kind of instructions is three bytes.

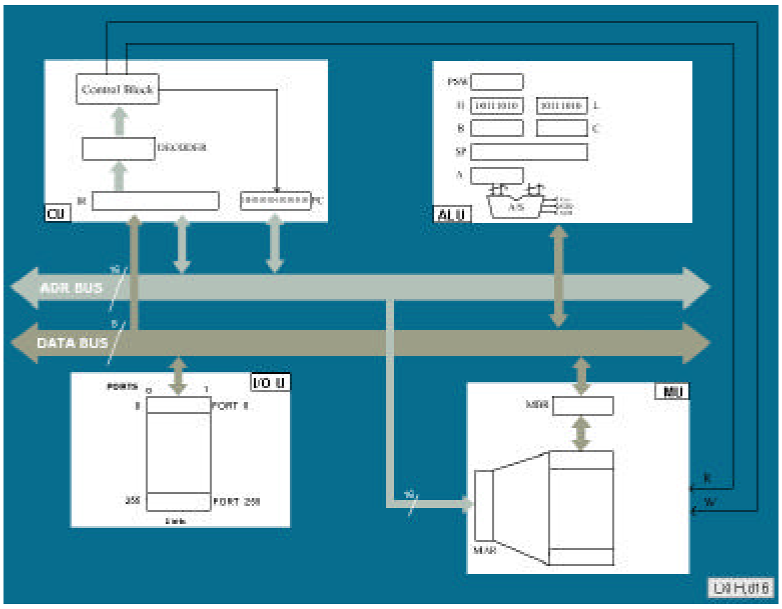


Fig.2. Capture of the execution of instruction LXI H,d16

The evolution of execution of this instruction, as seen by the students, is as follows: after the settings are ready, it is started the fetch phase. The content of PC (Program Counter) is transferred into MAR (Memory Address Register) using the ADR BUS (Address Bus). After reading the content of the addressed location, the first byte of the instruction is transferred into the IR. The content of IR (Instruction Register) is decoded by the “DECODER”. The Control Block is acknowledged that it is a three byte instruction. The Control Block increments the PC and it is continued the fetch phase.

The next two bytes of the instruction are read from the memory and are transferred in the working registers Z and W. After each fetch action the content of PC is incremented.

After that, the process is switched to execute phase. The content of the pair WZ is transferred into the pair HL in two steps (the internal bus is on 8 bits), 8 bits at once.

Thus, eventually the content of the pair HL became equal to the 16 bits specified in the instruction.

Let us consider another instruction from the Data Transfer Group, namely MVI A. This instruction ensures loading of an 8 bit immediate data, given in the instruction, into the Accumulator.

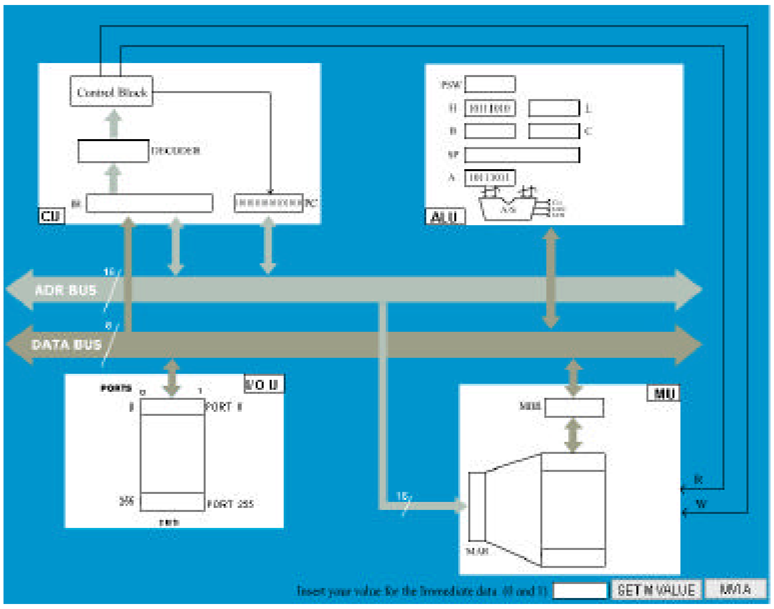


Fig.3. Capture of the execution of instruction MVI A

The student can set the value of the data associated to the instruction (the immediate data) before starting the instruction cycle. This value must be formed only of 0-s or 1-s (binary digits). After that he/she can start the presentation of the instruction respecting all the steps of the ”fetch cycle” and “execute cycle”.

After fetching the first byte from the memory and decoding it the Control block is acknowledged that it is a two-byte instruction. Therefore, a new "read memory" cycle is initiated and the second byte of the instruction representing the immediate data is fetched into MBR from where it is transferred into the working register Z. After this operation, the Control block increments again the PC and the instruction cycle passes into the execute phase. In this phase the 8 bit data from the register Z is transferred into the register A.

From the Arithmetic Group it was selected presentation of the ADD H instruction. This instruction adds the content of the register H with the content of Accumulator and stores the result into the Accumulator, by changing correspondingly by the content of PSW.

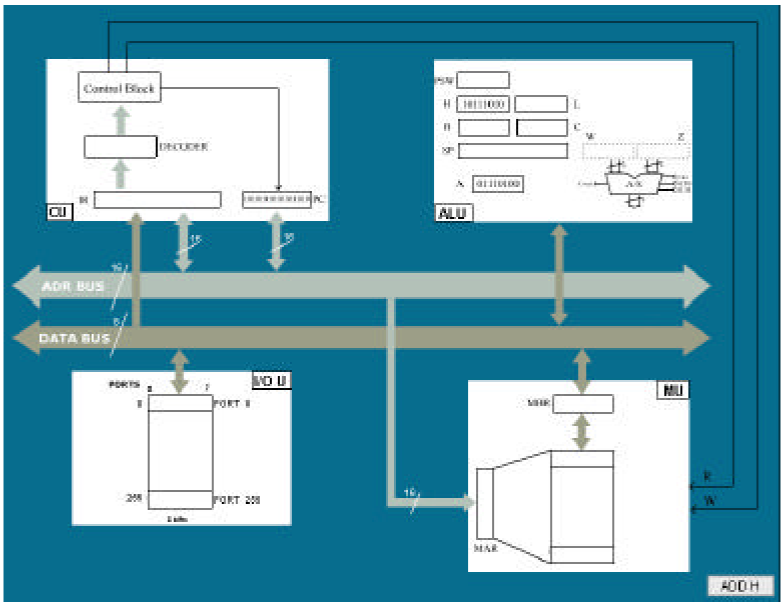


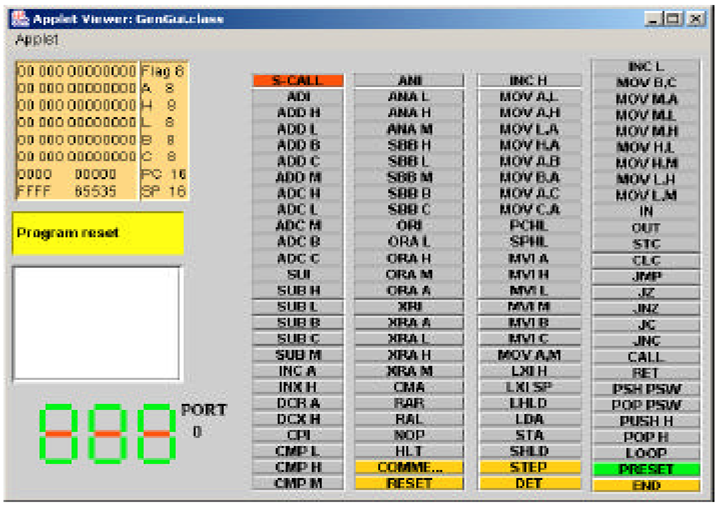
Fig.4. Capture of the execution of the instruction ADD H

Step-by-step execution of the instruction “ADD H”: firstly, it is started the fetch phase, so that the content of PC (Program Counter) is transferred through the ADR BUS (Address Bus) into the Memory Address Register. After that the Read command is issued and the location specified by the content of MAR is read and the corresponding word is placed into MBR, from which it is transferred into IR. The content of IR (Instruction Register) is decoded by the “DECODER”. The Control Block increments the PC and it is initiated the execute phase. In this phase the content of A is transferred into the working register W, while the content of H into the working register Z. After that the Control Block coordinates the addition of the contents of these two registers. When the addition is carried out the sum is stored back into the register A and there are also changed the flags in PSW register (depending on the content of A).

4. THE PROCESSOR SIMULATOR

The simulator was created by using the Java programming language and it was intended to be used inside a browser window, as a Java applet.

Basically, the operation of the simulator is conceived as simple as possible. On the main panel, which occupies the right-most part, the user can choose the instructions to be used to write programs. On the left part of the interface there is the status information. The user can see at any time the values of the internal registers of the processor, the instructions that were used in the program and the value of the active port. The interface is depicted in Fig.5.



```
00 000 00000000 Flag 8
00 000 00000000 A    8
00 000 00000000 H    8
00 000 00000000 L    8
00 000 00000000 B    8
00 000 00000000 C    8
0000     00000  PC  16
FFFF     65535  SP  16
```

Program reset

Fig.5 The simulator interface for VIP8

After writing the program, the user has to press the “END” button and a new interface will be displayed.

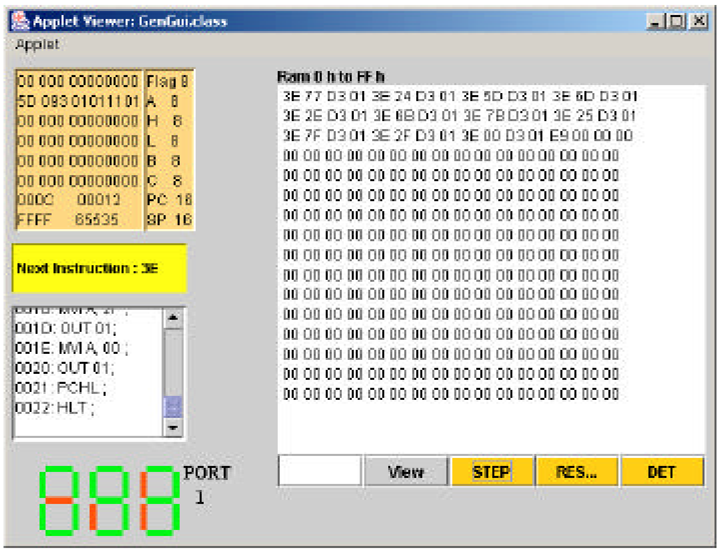
The new interface is a different one; mainly, the button panel from the first disappears and is being replaced by a memory panel. At any instant there can be visualized the values from 256 locations. But the addresses can be modified, such that all memory locations can be inspected (checked).

In order to run the program step by step (that is, instruction by instruction), the user has to press the “STEP” button. The current instruction will be executed and all involved components of the computer will be changed adequately.

At the end of running the program, the user can return to the main interface, with the help of the button “RESET”. All registers will be cleared, as well as the values of the ports and all memory locations.

There is provided an additional button named DET, that allows the user to detail furthermore the execution of an instruction; if the “DET” button is pressed it will open a new browser window with a Flash made simulating environment, which presents in full detail how the instruction is executed internally.

In Fig.6. it is depicted the memory panel inside the interface:



```
00 000 00000000  Flag 8
5D 093 01011101  A    8
00 000 00000000  H    8
00 000 00000000  L    8
00 000 00000000  B    8
00 000 00000000  C    8
000C    00012    PC  16
FFFF    65535    SP  16
```

```
001D: OUT 01;
001E: MVI A, 00 ;
0020: OUT 01;
0021 : PCHL ;
0022: HLT ;
```

```
Ram 0 h to FF h
3E 77 D3 01 3E 24 D3 01 3E 5D D3 01 3E 6D D3 01
3E 2E D3 01 3E 6B D3 01 3E 7B D3 01 3E 25 D3 01
3E 7F D3 01 3E 2F D3 01 3E 00 D3 01 E9 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Fig.6. Debug interface

The application provides also a large variety of ready made examples. The user can very easily run any of these demo programs by using another button called “PRESET”; by pressing this button the following dialog box appears (Fig.7.).

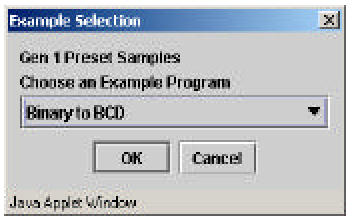


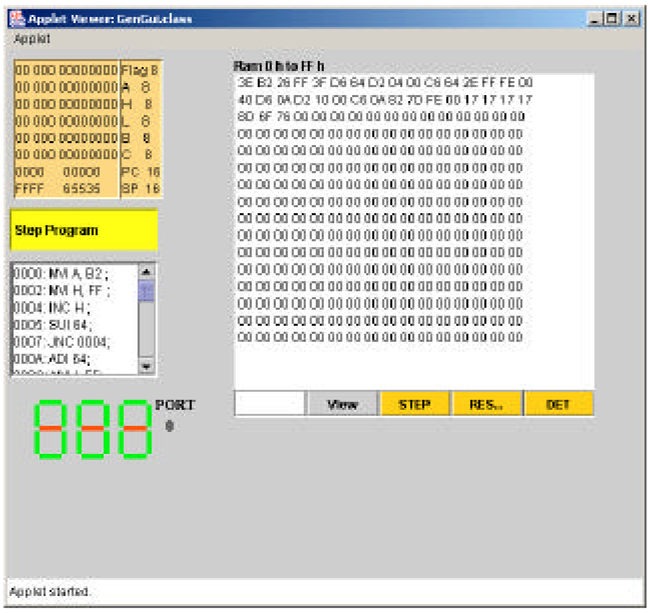
Fig. 7 Loading a preset sample

There is a list box that allows the user to actually choose the program that it is wanted. The samples that can be loaded are listed below:

- Simple Loop - implements a simple conditional loop
- Arith+Logic - implements some arithmetic and logic operation
- Loop+IO Port - implements a loop and writes the result into a port
- I/O Port - implements some operations with ports
- Subcall + Jump - implements a subroutine
- Stack Use - implements simple operations with stack
- Nested Routines - implements a nested routines
- Multiplication - implements a multiplication between two operands on 8 bits
- Division - implements a division between two operands on 8 bits
- Binary to BCD - implements o conversion from Binary to Binary Coded Decimal
- BCD to Binary- implements o conversion from Binary Coded Decimal to Binary
- Interchanging two registers - implements a swap between two registers
- Changing register and memory location contents- implements a swap between a register and a memory location
- 2's complement - calculates 2’s complement of a number
- 9's complement - calculates 9’s complement of a number
- 10's complement - calculates 10’s complement of a number

Let us take, for instance, the “Binary to BCD” conversion sample. After choosing it and after pressing the button “OK”, all instructions of the program will be loaded in the simulator memory.

In the next picture (Fig.8.) it is presented the memory panel showing the memory content for the first 256 locations, the flags and the program itself listed in the scroll box:



```
00 000 00000000 Flag 8
00 000 00000000 A    8
00 000 00000000 H    8
00 000 00000000 L    8
00 000 00000000 B    8
00 000 00000000 C    8
0000      00000 PC  16
FFFF      65535 SP  16
```

```
0000: MVI A, B2 ;
0002: MVI H, FF ;
0004: INC H ;
0005: SUI 64;
0007: JNC 0004;
000A: ADI 64;
```

```
Ram 0 h to FF h
3E B2 26 FF 3F D6 64 D2 04 00 C6 64 2E FF FE 00
40 D6 0A D2 10 00 C6 0A 82 7D FE 00 17 17 17 17
8D 6F 76 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Fig.8. Debug a preset sample

From this point on, the program will be run just like any other user-made program, with no difference at any level.

5. OTHER DIRECTIONS OF THE DESIGNED LABORATORY

Besides the instruction set study the following subjects linked to different chapters of Computer Architecture course are under consideration by the authors:

1) Virtual Memory
2) Cache Memory
3) Memory Addressing Techniques
4) Operation of Stacks
5) Direct Memory Access Block
6) Control of Buses
7) Interrupt handling
8) Pipeline organization

6. CONCLUSIONS

The set of laboratory works envisaged will be used by the students enrolled in the current activities at the Virtual University set up within "ViRec e-Initiative" project. The development is an international endeavor, where two partners, University of Craiova and University of Limerick have geared their efforts to create a set of efficient laboratory works to be remotely approached by the future students in accordance with the planned actions.

REFERENCES

Carpinelli J.D., Computer Systems Organization and Architecture, Addison-Wesley, 2001

Dasgupta S., Computer Architecture – A Modern Synthesis, John Wiley & Sons, 1989

Hayes J., Computer Architecture and Organization, McGraw Hill, 1998

Hennessy J., Patterson D., Computer Architecture: A Quantitative Approach, Morgan Kaufman, 1996

Hill F.J., Peterson G.R., Digital Systems Hardware Organization and Design (3rd Ed), John Wiley & Sons, 1987

Laventhal L., Saville W., Z80 Assembly Language Subroutines, Osborne, Berkeley, Ca, 1983

Mano M.M., Computer Systems Architecture, Prentice Hall, 1992

Murdocca M.J., Heuring V.P., Principle of Computer Architecture, Prentice Hall, 2000

Pollard L.H., Computer Design and Architecture, Prentice Hall, 1990

Stallings W., Computer Organization and Architecture, Prentice Hall, 2001 (6th Ed)

Stone H., High Performance Computer Architecture, Addison-Wesley, 1993