# EASY/VI - A NEW INSTRUCTIONAL COMPUTER

Luisa Koneva
Center for Informatics and Computer Technology
Bulgarian Academy of Sciences
Acad. G. Bonchev Str. bl. 25 - A
1113 Sofia
Bulgaria

Jordan Denev
Faculty of Mathematics and Computer Science
Sofia University
5, Anton Ivanov Str.
1126 Sofia
Bulgaria

## ABSTRACT

In this paper it is suggested a simple hypothetical computer which has been designed to assist teaching of the basic concepts in introductory course in computer organization and assembly language programming. The hypothetical computer and the instructional assembly language presented here have some new features which facilitate learning significantly. They have been designed to be a vehicle in almost all the lectures of the course.

## INTRODUCTION

The authors of this paper are familiar with the difficulties in learning and teaching real computer architectures and real assembly languages. The disadvantages of such an approach in introductory course in computer organization are discussed in [Decker, 85], [Little, 88], [Lees, 88], [Eckert, 87].

Little and Gayler (see [Little, 88], [Gayler, 87]) suggest teaching subsets of the assembly languages of the IBM S/360 and PDP 11/40 systems. Our opinion is that the students should not learn a subset of any real assembly language. We have taken into account that it is not possible to avoid unnecessary details which could make the understanding difficult.

The idea of using a simple hypothetical computer to teach the fundamental principles of computer organization and assembly language programming has been developed by Tannenbaum and Knuth (see [Knuth, 73], [Tannenbaum, 76]) and has been used and developed further by Decker, Miller, Eckert and some others (see [Decker, 85], [Miller, 83], [Eckert, 87] ).

Such projects as Sebesta's and Eckert's hypothetical computers (see [Sebesta, 84], [Eckert, 87]) are too elementary and can be used only in introduction to such a course.

The problem that arose from our practice in the teaching computer organization and assembly language programming was that students have not deep understanding of such concepts as addressing modes, subprogram construct, high - level statements implementation, input/output operations and interrupts. There was not a proper instructional software system for laboratory exercises.

The software system suggested in this paper simulates a hypothetical simple computer called EASY/VI (EASY/VIrtual machine). The system allows students to execute programs written in a simple assembly language, called ALL - Assembly Language for Learning.

The EASY/VI computer is simple and easy to be understood. Nevertheless it can be used for writing too complex and instructive programs (for example programs representation and manipulation with different data structures at assembly level).

The EASY/VI computer has been designed to teach the basic concepts in introductory course in computer organization. They are: memory, registers, machine instructions, fetch-execute cycle, addressing modes, subprograms, implementation of the constructs of high level languages. These concepts comprise about a half of the course contents as it is described in Curriculum' 78 (see [SIGED, 78], p. CS3).

The machine architecture suggested here and the instructional assembly language are described below. The hypothetical computer simulator has been implemented in the C language.

## THE HYPOTHETICAL COMPUTER ARCHITECTURE

The architecture of the suggested instructional computer is illustrated in

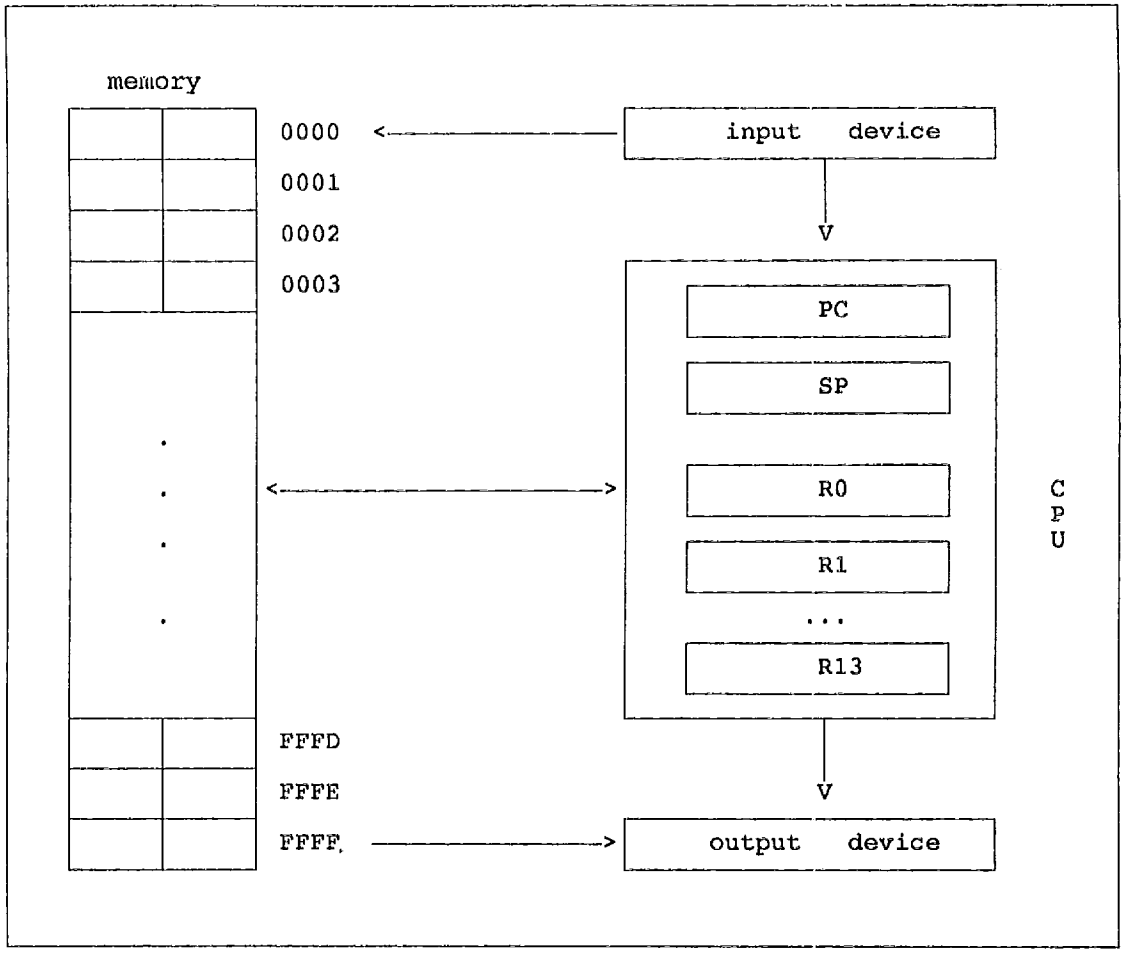


Figure 1. Hypothetical computer architecture

Figure 1. It has a memory divided into cells. Every cell has address which is a hexadecimal number from 0 to FFFF. Two hexadecimal digits can be stored into a memory cell. Two neighbour cells make a word. The hypothetical computer has sixteen registers named R0, R1, ..., R13, SP (Stack Pointer) and PC (Program Counter). They are general purpose registers (special purpose registers are difficult for learning ). The last ones (SP and PC) have specific functions as well. There are four flags used for conditional branches.

## THE INSTRUCTIONAL ASSEMBLY LANGUAGE

There are sixteen assembly language instructions. Among them are: MOVE, INC, DEC, BC, JSB, RSB, STOP which are very often used in real assembly languages. The arithmetic instructions usually have two operands but in our assembly language they have three operands in order to be more clear and comprehensive. The FOR instruction is used to facilitate cycles construction.

There are a sufficient number of addressing modes. The immediate, absolute and register addressing modes are basic ones. They are used often in real assembly languages. The other addressing modes give opportunities to construct and use various data structures: arrays (index, autoincrement and autodecrement modes), dynamical structures (indirect modes), records (direct modes). The ALL - addressing modes are shown in Table 1.

The instructions of the suggested assembly language have the following feature (it is called orthogonality): every addressing mode could be used for every kind of operand and in all the assembly instructions. For example, after learning about index mode (which is one of the basic principles of addressing), that mode could be used to describe every operand of all the instructions. In this way the program writing is significantly facilitated.

| Addressing modes | Description |
|---|---|
| # integer | operand = integer |
| identifier | A equals the content of the memory word defined by the identifier. |
| @ address | A = address |
| i [ Rx ] | A = A ( i ) + ( Rx ) * 2 |
| Rn | A = Rn |
| ( Rn ) | A = ( Rn ) |
| - ( Rn ) | ( Rn ) = ( Rn ) - 2 ; A = ( Rn ) |
| ( Rn ) + | A = ( Rn ) ; ( Rn ) = ( Rn ) + 2 |
| @ ( Rn ) + | A = ( ( Rn ) ) ; ( Rn ) = ( Rn ) + 2 |
| @ D ( Rn ) | operand = ( D + ( Rn ) ) |
| * @ D ( Rn ) | operand = ( ( D + ( Rn ) ) ) |

Table 2.

Remarks:

1. The registers R0, R1, ..., R13, SP and PC are noted by "Rn". The registers R0, R1, ..., R13, SP are noted by "Rx".

3. D is an integer number called offset.

4. The content of the register Rn is noted by "(Rn)".

5. The absolute memory address of an operand is noted by "A".

The existence of a number of addressing modes gives a possibility of wide variety of assembly instructions. For example, a dyadic instruction could be represented in 121 different ways because each operand could be represented in eleven different ways. Hence, it is possible to write quite complex and instructive programs by means of a small number of assembly instructions and eleven different addressing modes.

## COMPARISON TO OTHER PROJECTS

The computer EASY/VI suggested in this paper is not so simple as the instructional machines described in [Eckert, 87] and [Sebesta, 84]. These machines could be used to illustrate only the basic concepts in introduction to a course in computer organization. For example, Eckert's computer has four special purpose registers, seven very simplified assembly instructions and absolute addressing mode. The EASY/VI computer has more complicated architecture and assembly language and therefore could be used for teaching addressing modes, subprograms, implementation of high-level languages constructs and different data structures representation.

It turned out that it is not possible to define the EASY/VI machine language by means of Lees's interactive system for modelling of simple assemblers which has been used to assist the teaching of the concepts in a course in computer organization (see [Lees, 88]). The reason is that the machine instruction length for the EASY/VI machine is not fixed in advance, i. e. machine instructions length depends on the lengths of the operands by contrast with machine commands of the following systems: Motorolla 68000, IBM/370' and PDP-11'.

## CONCLUSIONS

The instructional system described here has been designed to assist learning in introductory course in computer organization and assembly language programming.

There are some other analogical projects but suggested computer architecture and assembly language programming have some new features that facilitate learning too much. Orthogonality of the assembly instructions and the existence of several addressing modes make possible writing more complex and instructive programs without making the language difficult for learning. The fact that it is possible to use the proposed computer to illustrate a lot of the notions in the course makes that project an original one.

That software system has been designed to give us an opportunity to base all the lectures and laboratory exercises on simple hypothetical computer. We consider that such an approach will give us a deep understanding of the concepts in the course.

At present EASY/VI could be used to illustrate only about a half of the concepts in the course. In order to complete our project it is necessary to develop the software so that it could be possible to be demonstrated the input/output operations and interrupts.

Our simulator is interactive. At every step users may choose either to run the next command or to display the contents of registers or individual storage locations. The means of making computer operation visual will be extended further.

REFERENCES


Decker, W.
A Modern Approach to Teaching Computer Organization and Assembly Language Programming, ACM SIGCSE Bulletin, Vol. 17, No. 4, pp. 38 - 45, 1985.

Eckert, R.
Kicking off a Course in Computer Organization and Assembly/Machine Language Programming, ACM SIGCSE Bulletin, Vol. 19, No. 4, pp. 2 - 10, 1987.

Gayler, R.; Beise, C.; Owen, G.
Conversion of PDP - 11/40 Assembler and Simulator from Mainframe Pascal to Ada on IBM PC Microcomputers, ACM SIGCSE Bulletin, Vol. 19, No. 1, pp. 378 - 381, 1987.

Gillet
A Pedagodical Processor Model, ACM SIGCSE Bulletin, Vol. 15, No. 1, pp. 159 - 164, 1983.

Knuth, D.
The Art of Computer Programming, Vol. 1/Fundamental Algorithns , 2nd ed. Addison - Wesley Publishing Co. Reading Mass., pp. 120 - 225, 1973.

Lees, B.
Interactive Modelling of System Software, Proceedings Intern., AMSE Conference " Modelling & Simulation ", Instanbul (Turkey), June 29 - July 1, AMSE Press, Vol. 1D, pp. 83 - 94, 1988.

Little, R.; Smotherman, M.
Assembly Language Courses in Transition, ACM SIGCSE Bulletin, Vol. 20, No. 1, pp. 95 - 99, 1988.

Miller, D.
Computer-1 - A Modern Simple Computer to Introduce Computer Organization and Assembly Language Programming, ACM SIGCSE Bulletin, Vol. 15, No. 1, pp. 271 - 277, 1983.

Sebesta, R.
VAX - 11 Structured Assembly Language Programming, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, U. S. A., 1984.

SIGED
(Special Interesting Group in Education) Curriculum' 78, Communications of the ACM, Vol. 22,. No. 3, pp. 147 - 165, 1979.

Tannenbaum, A.
Structured Computer Organization, Prentice - Hall, Inc., Englewood Cliffs, New Jersey, 1976.


---

' IBM/370 is a registered trademark of International Business Machines Corporation.

' PDP and VAX are registered trademarks of Digital Equipment Corporation.