HIGHER TECHNICAL INSTITUTE

ELECTRICAL ENGINEERING COURSE

DIPLOMA PROJECT

DEVELOPMENT OF AN 8085 VISUAL SIMULATOR

E / 998

BY: CHASAPIS DIMITRIOS

JUNE 1996

HIGHER TECHNICAL INSTITUTE

ELECTRICAL ENGINEERING COURSE

DIPLOMA PROJECT

DEVELOPMENT OF AN 8085 VISUAL SIMULATOR

E.998

CHASAPIS DIMITRIOS

JUNE 1996

HIGHER TECHNICAL INSTITUTE

ELECTRICAL ENGINEERING COURSE

DIPLOMA PROJECT

DEVELOPMENT OF AN 8085 VISUAL SIMULATOR

E.998

CHASAPIS DIMITRIOS

JUNE 1996

DEVELOPMENT OF AN 8085 VISUAL SIMULATOR

BY

DIMITRI CHASAPI

PROJECT REPORT

Submitted to the Department of Electrical Engineering
of the Higher Technical Institute
Nicosia Cyprus
in partial of the fulfilment of the requirements
for the award the diploma of
TECHNICIAN ENGINEER in ELECTRICAL ENGINEERING

E.998

Project Supervisor:
Dr. M. Kassinopoulos

June 1996

CONTENTS

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my Project Supervisor Dr. Mario Kassinopoulo for his guidance and assistance throughout the project period.

I would also like to thank my professor Mr. Ch. Theopemptou for his suggestion, assistance but most of all, for his patience during the project period.

SUMMARY

PROJECT TITLE : DEVELOPMENT OF AN 8085 VISUAL SIMULATOR

STUDENT : CHASAPIS DIMITRIOS

SUPERVISOR : MARIOS KASSINOPOULOS

EXTERNAL ASSESSOR : PANTELIS FRANGOULIS

The objectives of this project are:

- To study fully the 8085 microprocessor Assembly language.
- To write a program in high level language for running program written in 8085 Assembly language program.
- Investigate possibility of executing the program "step by step" displaying on the screen at the same time, the contents of the registers.

Terms and Conditions:

- The program should be written in Basic.
- An instruction manual should be provided, for the use of the program above in Digital laboratory.

INTRODUCTION

"Programmers are the owners of this world."

This is what Bill Gates, owner of Microsoft Corporation and editor of Basic programming language said. And this is true, since today's world is ruled over by computers and having the ability to force the computers to do exactly what you want them to do is a power. In order to help the programmers, some of those wrote several high level languages (Basic, Pascal etc.) which made programming and at the same time control over those stupid machines that rules the world, a piece of cake.

Error detection and most times auto-correction, debugging utilities, program testing and run time program modifications are some of the features offered by these languages. But what happens with the low level languages like Assembly? The programmer is left alone to "take the snake out of its hole", as a Greek proverb says. Using an assembler, the programmer can test his program for syntax errors and other typing mistakes, but what about the operation? Nobody can tell the programmer that his program runs correctly and that it performs the task that it is supposed to. It is left just to the programmers' ability to ensure the operation of the program.

The most common and oldest way to test if the program is written correctly, is to convert it into machine code, using an Assembler as mentioned above and program the memory where the program will be stored (usually an EPROM or a PROM). Then the memory is inserted in the application board, and by operating the application the programmer tests it to see if it works. At the time that the programmer converts the program from mnemonics to machine code, the Assembler informs him about the syntax errors that exists, the exact location of the errors and most of the times, which is the error and how it should be corrected. If there is a programming error, the programmer will see that the application will not operate correctly, and he will try to find out the mistake he made either in the hole program, or in a part of it, if he uses

subroutines and the problem allow him to find out which is the subroutine that has the mistake.

Then, if the program is corrected, he must erase the memory (if it is a PROM, buy a new one), reprogram it, place it back on the application board, and test the application again. In the case that an other mistake exists, the programmer must follow the same steps again. This way of testing a program, is very hard to make, since it takes a lot of time and effort.

So, an other method of testing the program should be used. Here is where the simulator takes the charge. A simulator can run the program, so that the programmer can see whether the program operates correctly or not. He can see all the changes inside the microprocessor, find all the programming mistakes that exist and correct them, before even programming the memory.

What can 8085 Visual Simulator do?

In a few words all the above. It can test the program for syntax mistakes, inform the programmer about their existence and further more, it can run each instruction of the program and simulate its operation, so that the programmer can correct his programming mistakes even before he runs the program in a real application.

Also, the many editing functions that this simulator has, allows the programmer to manipulate his programs as easy as possible, by using all the functions that all the editors have. This will not allow the programmer only to test and correct his program, but write it form the beginning. By this way, less effort is going to be needed, since he can test the program throughout the writing so that he has to find out for mistakes only in a few program lines. This can be achieved, if in every 5 or 10 lines he test the program.

The main advantage of this simulator, is that it can run under any P.C. with Ms Windows, available in almost all existing computers and it can use all the functions of Windows. Its disadvantage is that since Assembly language is a dedicated language, i.e. it runs only in one specific microprocessor, this simulator can test

programs written in i8085 microprocessor Assembly. (Although the i8085 stands for the Intel microprocessor, all 8085 microprocessors use the same Assembly language)

The program is written in a special version of Basic programming language, called Visual Basic. With Visual Basic programming language, full-fledged Windows applications can be written with minimum program instructions. User interfaces are created by directly manipulating on-screen objects such as control buttons and dialogue boxes, as well as fully functioning menu systems added to the application. Event driven procedures, allow the programmer to define what happens when a particular event, such as clicking on a button, occurs. This minimises the programming required, since there is no need to write a code to call a subroutine when a button is pressed etc.

Visual Basic has many debugging capabilities. It allows the programmer to enter a break point in the program so that when the program is tested, at each break point, a window appears that allow the programmer to see the contents of variables, make calculations etc.

All this programming environment, but most of all the fact that windows applications are designed, with small effort compared with the one needed with a language such as Pascal, C etc. makes Visual Basic the programming language of the future. And it is believed that soon, all the high level languages, in order to be used by the programmers will have the same, or a similar environment with the one of Visual Basic.