An 8-Bit Scientific Calculator Based Intel 8086 Virtual Machine Emulator

Qasem Abu Al-Haija*, Saleh Al-Abdulatif and Mohaned Al-Ghofaily

Abstract

Microprocessors and their applications course is considered as a significant core course for electrical engineering students due to its potential impact into several real life applications such as complex calculations, interfacing, control and automation technology. In this paper, we propose an eight bit scientific calculator based Intel 8086 assembly language programming. The calculator were designed over the virtual machine for Intel 8086 microprocessor using EMU8086 emulator software. Several arithmetic and logic operations as well as trigonometric functions were implemented in this paper. Also, a plot function and integration of function tools are to be implemented and added as a separate modules for this design. This work was very beneficial in enhancing the student’s skills in mathematics, engineering and computer programming which can be employed in designing a useful applications for users as well as the ability to apply numerical techniques and programming algorithms to design a small microprocessor-based system.

© 2013 The Authors. Published by Elsevier B.V.
Selection and peer-review under responsibility of Elhadi M. Shakshuki

Keywords: Intel 8086; EMU8086; Virtual Machine; Assembly Language; Scientific Calculator.

1. Introduction

In the late of 1978, Intel introduced the 8086 microprocessor as an enhanced product version of previous 8085 microprocessor. The product implementation depended on semiconductor process innovation, improved architecture, better circuit design, and more sophisticated software, yet upward compatibility not envisioned by the first designers was maintained [5].

Intel 8086 Microprocessor was designed to provide an order of magnitude increase in processing throughput over the older 808x. The processor was to be assembly-language-level-compatible with the 8080 so that existing 8080 software could be reassembled and correctly executed on the 8086 [1]. The 8086 processor architecture is described in terms of its memory structure, register structure, instruction

* Corresponding author, Lecturer, ECE Department, KFU. Tel.: +966-3-5895400; fax: +966-3-5871068.
E-mail address: qalhaija@kfu.edu.sa.
set, and external interface. Intel 8086 is a 16-bit microprocessor with 16-bit Data bus/ALU, 20-bit address bus and Maximum clock frequency is 5 MHz [4]. Intel 8086 support up to 1MB of main memory divided into 16 segments with 64KB size each. Intel 8086 contains 14 registers (16-bit) grouped in three main files of registers (Four 16-bit general registers, two 16-bit pointer and two 16-bit index registers, and four 16-bit segment registers) in addition to the status register and instruction pointer. The registers are shown in table 1. Also, Nine flags record the processor state and control its operation: The status register (flag register) [1, 4] is a 16-bit register, 9 out of these 16 bits are active and indicate the current state of the processor. These bits include: Carry flag (CF), Parity flag (PF), Auxiliary flag (AF), Zero flag (ZF), Sign flag (SF), Trap flag (TF), Interrupt flag (IF), Direction flag (DF) and Overflow flag (OF).

Table 1. Intel 8086 Registers

| General Purpose Registers | AH | AL | AX (primary accumulator) |
| Index and base registers | SI | DI | BX (base, accumulator) |
| M | CH | BL | DX (counter, accumulator) |
| DH | CL | CX | DX (accumulator, other functions) |
| Status register | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 (bit position) |
| Flags | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| Segment register | CS | DS | Code Segment |
| ES | SS | Data Segment |
| Instruction pointer | IP | Extra Segment |

8086 has approximately 117 different instructions [1, 3, 4] with about 300 op-codes with three instruction formats: no-operand, single-operand and two-operand instructions as well as the string instructions that involve array operations. Intel 8086 instructions classified into 8 groups: Data transfer instructions, Arithmetic instructions, Bit Manipulation instructions, String instructions, Unconditional Transfer instructions, Conditional Branch instructions, Interrupt instructions, and Processor Control instructions. Intel 8086 provides various 12 different addressing modes to access instruction operands. The operand may be contained in: register, immediate, memory or I/O ports. The addressing Modes are classified into 5 groups: Register and immediate modes (two modes), Memory addressing modes (six modes), Port addressing mode (two modes), Relative addressing mode (one mode) and Implied addressing mode (one mode). The full details about Intel 8086 can be retrieved from [1,3, 4].

In this paper, we are going to utilize the capabilities of 8086, instructions, registers and memory to perform the several operations in the design of an 8-bit scientific calculator.

2. Design Specifications and Modeling

In this paper, we are going to design a program that simulates the operation of an 8-bit calculator system according to the following specifications and assumptions. Figure 1 shows the four basic modules of the calculator system.
- The calculator should perform the following unsigned arithmetic operations: addition (+), subtraction (-), Multiplication (*), division (/), and Power (X^Y).
- The calculator should perform the following Bitwise Logical operations: ANDing (&), ORing (||), XORing (⊕), complementing.
- The calculator should perform the following Trigonometric functions: Sin (x), Cos (x), Tan (x).
- The calculator takes two 8 bit -operands only.
- The user can select one of these operations from a list of items (numbered from 1-11).
- The program should run in the video mode (the black screen – DOS Screen).
- The operands have a maximum size of 3-digits 0-999).
- All operands are entered in decimal form.
- The calculator will Extra two tools are in-process : Function Integration and Function Plot tools.

The paper aims to design the program, implement and test it using the EMU-8086 Emulator available in the lab. The following steps should be considered while designing this paper:
- Design a Flow Chart Diagram for the calculator system.
- Design the program in Assembly language.
- The program should be Assembled, Emulated and Run successfully on Emu_8086 emulator.
- The program should call a subroutine for each operation.

The paper to be designed required the students to have full understanding for the following programming skills:
- Working and understanding Emu_8086 emulator software.
- Running/debugging ALPs and tracing/analysing 8086 registers, memory, ALU and stack.
- Understanding Bit and Arithmetic manipulation using 8086 Intel MP.
- Using and verifying addressing modes for 8086 Intel Microprocessor.
- Memory & Stack access operations for 8086 Intel Microprocessor.
- Understanding String Operations using 8086 Intel MP.
- Conditional and Unconditional Branches.
- Working with subroutines, Loops and Interrupts.
- Ability to design and debug a small microprocessor-based system.
- Understanding the numerical methods and techniques for function integrals and trigonometric functions.

The problem addressed in this paper were divided into several modules, some of these modules are shown in flowchart, figure 2. Each module will be designed as a separate subroutine where the complete calculator system will has separate subroutine for each single operation (arithmetic, logic and others) and subroutine for inputting the data as well as for outputting the data.
The aforementioned flowchart along with the system specifications are considered as the core of the system design for this paper.

3. Simulation Environment and Sample of Results

The proposed work is to design an 8-bit scientific calculator which includes: Arithmetic Operations, Logical operations, Trigonometric functions and some other advanced tools such as function plot for low order polynomials and function integrals. The proposed solution is programmed and implemented in Assembly language programming for 8086 microprocessor using EMU8086 emulator.

EMU8086 [2, 4] is a Microprocessor Emulator with integrated 8086 Assembler and Free Tutorial. Emulator runs programs on a Virtual Machine, it emulates real hardware, such as screen, memory and input/output devices. EMU8086 is considered the right software tool to help in fully understand microprocessors and assembly language. The source code is assembled and executed on emulator step by step. It offers a GUI to control registers, flags and memory while the program is running. Emu8086 pack combines an advanced source editor with automatic syntax-highlight, assembler, dis-assembler, software emulator (Virtual PC) with debugger, and step by step tutorials. Emu8086 is complete 'all in one' solution for coding in Assembly Language. Emulator runs programs on a Virtual PC, this completely blocks the emulated programs from accessing real hardware, such as hard-drives and memory, since the assembly code runs on a virtual machine, this makes debugging much easier.

This work is still under construction, we have done the arithmetic and logical parts while other options of the design will be developed and finished in the near future. Simulation run samples are shown in the figure 3 and 4 below.
4. CONCLUSIONS AND RECOMMENDATIONS

A small microprocessor based system were discussed and designed using the Assembly language programming and EMU8086 virtual machine emulator. The paper will enhance the student ability in applying knowledge of mathematics, engineering and computer programming which can be employed in designing a useful applications for users as well as the ability to apply numerical techniques and arithmetic algorithms to design a small microprocessor-based system. The work in this paper can be improved by several ways such as: implementing more the integration of the functions and add the function plot tool which are under-consideration and extending the capabilities of the calculator to allow a 16-bit calculations as well as add more arithmetic operations such as root square roots, logarithmic functions, other logic functions (XNOR, NOR, NAND), Factorial (X!), Inverse (1/X), Modulus, power of 10 ($10^x$), and exponential ($e^x$).

Acknowledgements

Authors appreciate the publication support of College of Engineering at King Faisal University KFU-AL-AHSA.
References