Microprocessor simulator

ASSIGN BUSTER

Microprocesor

When the components of a CPU can be integrated on a single chip then it is called a microprocessor. Our project deals with the 8085 microprocessor.

The computers using the microprocessor and the additional peripherals is called as microcomputer.

Microprocessor Simulator

The 8085 Microprocessor Simulator is a total software solution to replace the microprocessor kit from training and design labs. The Simulator executes the instructions from the user entered program, instruction by instruction showing all the register and flag status at the end of execution of each instruction

Softwares

Visual Basic-6

Microsoft Visual Basic is probably the fastest and easiest way to create applications for Microsoft Windows. Visual Basic provides you with a complete set of tools to simplify rapid application development.

MS Access

A full- featured procedural programming language —preferably a subset of Visual Basic. It is a powerful database-handling tool and acts as a efficient backend to most of the front-end applications which include Visual Basic too.

Methodology

8085 Microprocessor Simulator

Introduction To 8085

The 8085 is a complex IC of sequential circuits. The sequential circuits are designed to do some operation depending on what is the input on their lines. The vital inputs on the lines are what therefore determine what operation will be done by the sequential circuits inside it. The operations can be very complex and therefore this chip is also called a processor. Since we can find a way to put different values of inputs to the input lines of the processor at different times, we can make it execute different operations in a sequence that we desire. Thus, in other words we can make the processor execute a program to do useful things for us. These inputs then could alternately, be called instructions.

The program that we desire to be executed must be loaded into consecutive locations of memory chips. The memory is not part of the 8085 processor. The memory chips are again sequential circuits consisting of flip-flops that are capable of storing digital values. Since we would be interested in storing a huge number of such digital values, a large number of these memories are packed together with a scheme of addresses, so that we can address them individually. Generally the memories are arranged in large numbers of 8 bit bunches each. The 8085 has a address bus which is 16 bit wide. Therefore it can put 2^16 different digital values on it, and therefore it can address a maximum of 2^16 different address locations. This is called the addressing space and it is 64 kilobyte for the 8085, because 2^16= 65535. And then we ask the processor to execute those instructions from a particular memory location onwards. It goes on executing those instructions one after another.

The processor uses certain internal memory locations called Registers in doing all the operations that we ask it to do. The contents of those memory locations can be directly altered by the instructions that we give.

Evolution

INTEL Corp. is generally recognized as the company that introduced the microprocessor successfully in the market. The first microprocessor, the 4004 was introduced in 1971. It was a central component in the chip set, called MCS-4.

The microprocessors introduced between 1971 and 1973 were the first generation systems. They used PMOS technology, which provided low cost, slow speed and low output currents. After 1973-second generation microprocessors such as Motorola 6800 and 6809, Intel 8085 evolved. They were fabricated using NMOS technology. After 1978

The third generation microprocessors were evolved. They were 16 bit wide and included Intel 8086/80186/80286. They were designed using HMOS technology. In 1980 the fourth generation microprocessors evolved. Since 1985, 32 bit microprocessors are fabricated using low power version of HMOS technology called HCMOS and they include an on-chip RAM called cache memory to speed up program execution. So extensive research is being carried out for the implementation of more on-chip functions and for the improvement of the speeds of memory and I/O devices

Programming Languages

Programming is basically running a set of well-defined instructions in a syntactical manner to solve a particular task/problem. Microcomputer programming languages can typically be segmented as:

1. Machine Language

2. Assembly Language

3. High-level Language

Machine Language:

A machine language program consists of either binary or hexadecimal OP(operation) codes. Eight bit microcomputers can be programmed using machine language. A microprocessor has a unique set of machine language instructions defined by the manufacturer. No 2 microprocessors have same machine language instruction set.

Assembly Language:

Assembly Language uses semi-English statements for 8 bit microprocessors. Each instruction in an assembly language comprises:

(a) Label Field

(b) Instruction, Mnemonic or Op-code field

(c) Operand field

(d) Comment field

Assembly language basically consists of programs written with the help of mnemonics

Mnemonic is a combination of letters to suggest the operation of an instruction. In general an instruction has 2 components-operation code (OP-code) field and Address field.

The OP-code field specifies how data is to be manipulated and the purpose of the address field is to indicate the address of a data item.

High- Level Language

Most 16 and 32 bit microprocessors in addition to assembly and machine language use a more understandable human oriented language called high-level language. High level language programs are composed of English language type statements . a no of high-level language.

Regardless of what type of language is used to write a program, the microcomputers understand only binary numbers. So the programs must eventually be translated into their appropriate binary forms. An assembler is one such translator that translates a program written in assembly language to machine language (object code). A compiler/an interpreter converts a high-level language program into a machine language one. A compiler translates the entire source code to object code and then executes it. On the other hand, the interpreter performs line-by-line translation and execution simultaneously like FORTRAN, COBOL, BASIC, C, C++ are widely used these days.

Instruction Set

Data Transfer Groups

These groups include the move, exchange, load, and store operations. Data transfer instructions are among the most widely used of all microprocessor instructions. This group of instructions transfers data to and from registers and memory.

None of the instructions of this group are the flag affecting instructions.

The instructions included in this group are MOV, MVI, LXI, LDA, STA, LHLD, SHLD, LDAX, STAX, and XCHG

For e. g.:

MOV (A, B) will move the contents of register B to register A.

Arithematic Groups:

This group includes the add, add with carry, subtract, subtract with borrow, increment, decrement, and decimal adjust accumulator operations. This group of instructions performs arithmetic operations on data in registers and memory. Unless indicated otherwise all the instructions are flag affecting instructions. All subtraction operations are performed via 2s complement arithmetic and set the carry flag to 1 to indicate a borrow and clear it to indicate no borrow.

The instructions included in this group are ADD, ADC, SUB, SBB, DAD, INR, INX, DCR, DCX, and DAA.

For e. g.:

ADD B will add the contents of register B to the contents of register A and store the result in A.

Logical Groups

This include AND, OR, XOR, compare, rotate and complement instructions. This group of instructions performs

Logical (Boolean) operations o the data in registers and memory and on flags.

The instructions include in the group are ANA, XRA, ORA, CMP, AI, XRI, ORI, CPI, RLC, RRC, RAR, RAL, CMA, STC and CMC.

For e. g.:

ANA C will logically AND the contents of register C with the contents of register A and store the result in A.

Branching Groups:

This include jump, call, return and restart instructions. This group of instructions alters the normal sequential program flow. The to types of branch instructions are:

- -Unconditional
- -Conditional

Unconditional transfers simply perform the specified operation o the program counter.

Conditional transfers examine the status of one of the four MPU flags to determine whether the specified branch is to be executed.

The instructions include in the group are

Unconditional – JMP, RET, CALL, RST 0, RST 1, RST 2, RST 3, RST 4,

RST 5, RST 6, RST 7, PCHL.

Conditional – JNZ, JZ, JNC, JC, JPO, JPE, JP, JM, CNZ, CZ, CNC, CC, CPO, CPE, CP CM, RNZ, RZ, RNC, RC, RPO, RPE, RP, RM.

For e. g.:

JMP 4000 will transfer the program flow to the memory location 4000

The 8085 Simulator

The 8085 Microprocessor Simulator is a total software solution to replace the microprocessor kit from training and design labs. The Simulator executes the instructions from the user entered program, instruction by instruction showing all the register and flag status at the end of execution of each instruction. The input process is fast and efficient, leading to better productivity. The Mnemonic Pad allows the user to enter the code. This makes is impossible to enter a wrong Mnemonic. The Tool tips tell you the syntax along with the operation performed by a particular Mnemonic. The Data Entry Window allows you to enter the Data Segment of your code. This is provided to enable the user to enter the data once and save it along with the code

Need for 8085 Simulator -Difficulties faced by microprocessor kit:

The 8085 Microprocessor is the basic microprocessor taught to students first all over the world. The concepts of how microprocessor can be well

understood in this basic chip. The same concepts can be very easily extended to advanced processors like the 8086, 80286, 80386, 80486 or the Pentium processor.

Today, students write their code in machine code in a Microprocessor kit which is a piece of hardware with keys that very often develop key debounce problems. After an hour of painful machine code entry the program is executed and the student is left clue less about where his problem just went on an infinite loop. Since he can not save his program he goes home for the day and goes over the entire exercise again the next day. It just steals away the fun from the Microprocessors.

In this simulator all that is History. He just clicks on the mnemonics and the machine code is written for him. Then he can debug the program by executing Step by Step. All the register states, flag states, memory dump, I/O Dump, Stack Locations, M/c Cycle and T-States taken, are clearly shown in a tabular fashion at the completion of each instruction.

This package eliminates the need for the expensive hardware setup. Online facility to simulate programs helps students to have hands on experience of programming.

Benefits Of Simulators:

1. Safety

While there are many things that can actually be done without needing a computer simulation, there are also a large number of activities that are best left to computers before physically doing them due to the risks involved.

First of all, there are some things like, surgery, which if extremely difficult, are best left to a computer to attempt the procedure first before actually risking the life of another person.

2. Monetary Issues

Obviously, there is a lot of cost involved when implementing any actual real life scenario, such as flying a plane, or building a new interstate highway system, so by using computers, we are able to drastically undercut prices and costs of the activities

3. Visualisation

Large amounts of people are visual learners. That is important to keep in mind when dealing with the benefits of computers, because they are primarily visual tools. Scientists definitely use the visualization aspect in their research. In fact, scientists are able to look at something as small as strands of DNA using a computer simulation program, which make them big enough to really observe and study.

4. Projection

This aspect of computer simulation actually relates back to the visualization concept in that because a computer simulation can speed up time, the user is able to determine effects of any given variable on the control. The ability to speed up time is especially beneficial because cause and effect relationships are much easier to demonstrate and understand

5. Replication

As any researcher knows, it can be quite difficult to get the exact same situation more than once. So, by using a computer model the user can replicate the exact same conditions time after time. In real life situations, this attempt at replication is not viable, so researchers and users do the best with what they have, but by using a computer, not only is perfect repetition possible, but it is also practical.

Disadvantages Of Simulators

1. Accuracy Of The Model

The problem with computer simulations is that their accuracy in determining what will actually happen next in a real life situation, depends solely on how well he model was made to begin with. If the model actually doesn't follow the real life situation closely, then simulating it will do the user no good, but if the model closely resembles the " real world", then it can be quite valuable. Because many if not all, models are based off of assumptions, they can be very difficult to actually verify or qualify. Within this framework, there is a very high chance for error, either mathematical, or personal, and because of that computers are not perfect predictors. The problem is that it can be hard to find or discover these errors, so many simulators or models could contain them, and no one would know until the real life situation came along and the results are not the same as predicted.

2. Computers Not Free From Error

The biggest problem is that people take computers to be somewhat " God-like" in that whatever their answer is, that must be the truth, because computers don't lie. It is the information given in these computers reports of

simulations that people base so many of their decisions. Decisions and choices that are more than just minor details, but also decisions which affect millions of dollars, lives, or amount of production, to name a few. People trust computers because they believe that anything a computer generates, must be true because it is a machine, and therefore incapable of misinterpreting something, or forgetting about something

Visual Basic 6

Microsoft Visual Basic is probably the fastest and easiest way to create applications for Microsoft Windows. Visual Basic provides you with a complete set of tools to simplify rapid application development.

The “ Visual” part refers to the method used to create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, you simply add pre built objects into place on screen so you can put in all your programming skills in the real application part.

The “ Basic” part refers to the BASIC (Beginners All-Purpose Symbolic Instruction Code) language, a language used by more programmers than any other language in the history of computing. Visual Basic has evolved from the original BASIC language and now contains several hundred statements, functions, and keywords, many of which relate directly to the Windows GUI. Beginners can create useful applications by learning just a few of the keywords, yet the power of the language allows professionals to accomplish anything that can be accomplished using any other Windows programming language

Whether your goal is to create a small utility for yourself or your work group, a large enterprise-wide system, or even distributed applications spanning the globe via the Internet, Visual Basic easily accomplishes using the following tools.

- Data access features allow you to create databases, front-end applications, and scalable server-side components for most popular database formats, including Microsoft SQL Server and other enterprise-level databases.
- ActiveX technologies allow you to use the functionality provided by other applications, such as Microsoft Word word processor, Microsoft Excel spreadsheet, and other Windows applications.
- Internet capabilities make it easy to provide access to documents and applications across the Internet or intranet from within your application, or to create Internet server applications.
- Your finished application is a true . exe (executable) file that uses a Visual Basic Virtual Machine that you can freely distribute.

8085 Assembler

A 8085 assembler creates object code by translating assembly instruction mnemonics into opcodes, and by resolving symbolic names for memory locations and other entities. The use of symbolic references is a key feature of assemblers, saving calculations and manual address updates after program modifications. Most assemblers also include macro facilities for performing textual substitution. e. g., To generate common short sequences of instructions to run inline, instead of in a subroutine.

Assemblers are generally simpler to write than compilers for high-level languages, and have been available since the 1950s. Modern assemblers, especially for RISC based architectures, such as MIPS, Sun SPARC, and HP PA-RISC, as well as x86(-64), optimize instruction scheduling to exploit the CPU pipeline efficiently.

There are two types of assemblers based on how many passes through the source are needed to produce the executable program. One-pass assemblers go through the source code once and assume that all symbols will be defined before any instruction that references them. Two-pass assemblers (and multi-pass assemblers) create a table with all unresolved symbols in the first pass, then use the 2nd pass to again solve these addresses. The advantage in one-pass assemblers is speed, which is not as important as it once was with advances in computer speed and capabilities. The advantage of the two-pass assembler is that symbols can be defined anywhere in the program source. As a result, the program can be defined in a more logical and meaningful way. This makes two-pass assembler programs easier to read and maintain the data.

More sophisticated high-level assemblers provide language abstractions such as:

- Advanced control structures
- High-level procedure/function declarations and invocations
- High-level abstract data types, including structures/records, unions, classes, and sets
- Sophisticated macro processing

- Object-Oriented features such as encapsulation, polymorphism, inheritance, interfaces

in normal professional usage, the term assembler is often used ambiguously: It is frequently used to refer to an assembly language itself, rather than to the assembler utility. Thus: “ CP/CMS was written in S/360 assembler” as opposed to “ ASM-H was a widely-used S/370 assembler