NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

DESIGNING AN ELECTRONIC ORGAN USING 8085 MICROPROCESSOR

Graduation Project
COM- 400

Student : Süleyman Kerimoğlu (20010522)

Supervisor : Prof. Dr Doğan İbrahim

Nicosia – 2006

ACKNOWLEDGEMENT

My gratitude for the following people cannot be express in this short acknowledgement.

First, I wish to thank to my supervisor, Prof. Dr Doğan İbrahim , whose encouragement and enthusiasms have been constant sources of inspiration to me. Also his invaluable advices and belief in my work and myself over the course of this Graduation Project.

Second, I would like to express our gratitude to Near East University (NEU) for the scholarship that made the work possible. Also special thanks starting from our Dean Prof. Dr Fakhreddin Mamedov and all other lecturers for the advices and helps.

I dedicate this project to my home friends Hüseyin Tulunay,Sinan Özerenler ,Ozan Akkoça and my best friends Mesut Arık , Göktuğ Ataç .

Finally, I would like to thank all our friends for their advices , supports and being with me with a big respect and patience.

ABSTRACT

Microprocessors are electronic equipments that is used in most of place which requires electronic control mechanism. They include from hundred thousands to millions of transistors. Microprocessors are used in traffic control systems, music systems, electronic devices at home, computers to meet the need of global world.

Most of music devices used to produce as mechanical devices in the past. But, as a reason of development in new technologies, they have been produced by using electronic equipment. Such an electronic organ, lots of electronic music device are able to hold songs (also notes) in their storage part, and whenever it is needed, they can play it automatically. This happens via microcontrollers and microchips. They are playing very important role in music equipment like in our lives.

As a graduation project, I decided to prepare a very simple electronic organ, using the 8085 Microprocessor and assembly language to code it. So, this project book presents electronic structure of microprocessors, music notes definitions in binary system and also assembly language code examples briefly.

TABLE OF CONTENTS

LIST OF ABREVATIONS

ALU	Arithmetic Logic Unit
ALE	Address Latch Enable
CPU	Central Procesing Unit
ALU	Arithmetic Logic Unit
INTR	Interrupt
I/O	Input/ Output
LSB	Least Signifigant Bit
µp	Microprocesor
RAM	Random Access Memory
R/W	Read/Write
RP	Register Pair
RST	Restart
PC	Program Counter
SP	Stack Pointer
SID	Serial Input Data
SOD	Serial Output Data
A/D	Anolog to Digital
D/A	Digital to Analog
PDS	Personal Development System

INTRODUCTION

In response of development in microprocessors, electronic devices are used in most of music equipments. As a result of interest in music instruments and hardware devices, I decided to prepare a simple electronic organ.

After some research, it becomes definitive to use a 8085 microcontrollers and assembly language. Some requirements such as a speaker and buttons are investigated. I used a PRIMER 8085 Microprocessor Trainer System because, it contains an internal speaker to get voice at specific frequencies and internal numpad to enter assembly numerical codes.

Chapter 1 presents definition of microprocessor, historical development and evoluation. Chapter also includes the attributes of general purpose CPUs of some companies.

Chapter 2 presents attributes of 8085 Microprocessor, internal architecture and bus systems of this microprocessor. It also provides explanation about pin structure, system interface and programming methods of 8085 microprocessor. A mnemonics opcode instruction set table is also provided in chapter.

Chapter 3 presents the PRIMER Training System properties. Chapter also investigates hardware and programming features of the device, bus system and the internal equipments such as seven segment display, numpad, speaker, timer, counter, power supply and communication ports.

Chapter 4 provides a hexadecimal values of notes table. The included part is very important for the project, because representation of music notes in digital binary system is the evoluational idea of the electronical music equipments.

Chapter 5 describes assembly codes that is used in project. An algorithm shcheme to play seven music notes and ordered code structure of the assembly program is included in the chapter.

CHAPTER 1
MICROPROCESSOR

Microprocessor is a multipurpose, programmable, clock-driven, register based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions, and provides as output.

A common way of categorizing microprocessors is by the number of bits that their ALU can work time. In other words a microprocessor with a 4 bit will be referred to as a 4-bit microprocessor, regardless of the number of address lines or the number of data bus lines that it has.

The first commercially available microprocessor was the Intel 4004 produced in 1971. It contained 2300 PMOS transistors. The 4004~was a 4 bit device intended to be used with some other devices in making a calculator. Some logic designers however saw that this device could be used to replace PC boards full of combinational and sequential logic devices. Also the ability to change the function of a system by just changing the programming, rather than redesigning the hardware is very appealing.

In 1972 Intel came out with the 8008, which was capable of working with 8-bit words. The 8008, however required 20 or more additional devices to form a functional CPU In 1974 Intel announced the 8080, which had a much larger instruction set than the 8008 and required only two additional devices to form a functional CPU. Also the 8080 used NMOS transistors so it operated much faster than the 8008. The 8080 is referred to as a second-generation microprocessor. It requires +12 V power supply.

Soon after Intel produced the 8080, Motorola came out with the MC6800 another 8-bit general-purpose CPU. The 6800 had the advantage that it required only a +5-V supply rather than the -5-V. +5-V. and + 12V supplies required by the 8080.

For several years the 8080 and the 6800 were the top-selling 8-bit microprocessors. Some of their competitors were the MOS Technology 6502. used as the CPU in the Apple II microcomputer and the Zilog

Z80 used as the CPU in the Radio Shack TRS~O microcomputer.

1-1- The Major Evolution of Microprocessor

Three major directions of microprocessor Evolutions are

(i) Dedicated or Embedded Controllers

(ii) Bit Slice Processors

(iii) General purpose CPUs

1-1-1- Dedicated or Embedded Controllers

One direction has been dedicated or embedded controllers. These devices are used to control "smart" machines such as microwave ovens Clothes washers. Sewing machines auto ignition systems and metal lathes.

Texas Instruments has produced millions of their TMS-1000 family of 4-bit microprocessors for this type of application.

In 1976 Intel introduced the 8048, which contains an 8-bit CPU, RAM, ROM. and some I/O ports all in one 40-pin package. Other manufacturers have followed with similar products. These devices are often referred to as microcontrollers.

Some currently available devices in this category-the Intel 8051 and the Motorola MC6801 a more recently introduced single chip microcontroller, the Intel 8096, contains a 16-bit CPU,ROM, RAM, a UART, ports, timers, and a 10-bit analog-to-digital converter.

1-1-2- Bit Slice Processors

A second direction of microprocessor evolution has been bit-slice processors. For some applications general purpose CPUs such as the 8080 and 6800 are not fast enough or do not have suitable instruction sets. For these applications, several manufacturers produce devices which can be used to build a custom CPU.

An example is the Advanced Micro Devices 2900 family of devices. This family includes 4-bit ALU's, Multiplexers, sequencers and other parts needed for custom-building a CPU. The term *slice* comes from the fact that these parts can be connected in parallel to work with 8-bit words, 16-bit words. or 32-bit words. In other words a designer can add as many slices as needed for a particular application.

The designer not only custom-designs the hardware of the CPU, but also custom-

makes the instruction set for it using "microcode".

1-1-3- General Purpose CPUs

The third major direction of microprocessor evolution has been toward general-purpose CPUs which give a microcomputer most or all of the computing power of earlier minicomputers.

Intel 8085:

After Motorola came out with the MC6800 Intel produced the 8085. an upgrade of the 8080 that required only a +5-V supply.

Motorola MC 6809:

Motorola then produced the MC6809 which has a few 16-bit instructions. but is still basically an 8-bit processor.

Intel 8086:

In 1978 Intel came out with the 8086 which is a full 16bit processor. Some 16-bit microprocessors such as the National PACE and the Texas Instruments 9900 family of devices had been available previously. but the market apparently was not ready.

Motorola MC68000:

Soon after Intel came out with the 8086, Motorola came out with the 16-bit MC68000.

The 8086 and the 68000 work directly with 16-bit words instead of with 8-bit words they can address a million or more bytes of memory instead of the 64 Kbytes addressable by the 8-bit processors and they execute instructions much faster than the 8-bit processors. Also these 16bit processors have single instructions for functions such as multiply and divide. This required a lengthy sequence of instructions on the 8-bit processors.

The evolution along this last path has continued on to 32-bit processors that work with gigabytes (109 bytes) or terabytes (1012 bytes) of memory. Examples of these devices are the Intel 80386. the Motorola MC68020, and the National 32032.

CHAPTER 2

Intel 8085 Microprocessor

The Intel 8085 was an 8-bit microprocessor made by Intel in the mid-1970s. It was binary compatible with the more-famous Intel 8080 but required less supporting hardware, thus allowing simpler and less expensive microcomputer systems to be built.

The "5" in the model number came from the fact that the 8085 required only a +5-volt (V) power supply rather than the +5V, -5V and +12V supplies the 8080 needed. Both processors were sometimes used in computers running the CP/M operating system, and the 8085 later saw use as a microcontroller (much by virtue of its component count reducing feature). Both designs were later eclipsed by the compatible but more capable Zilog Z80, which took over most of the CP/M computer market as well as taking a large share of the booming home computer market in the early-to-mid-1980s.

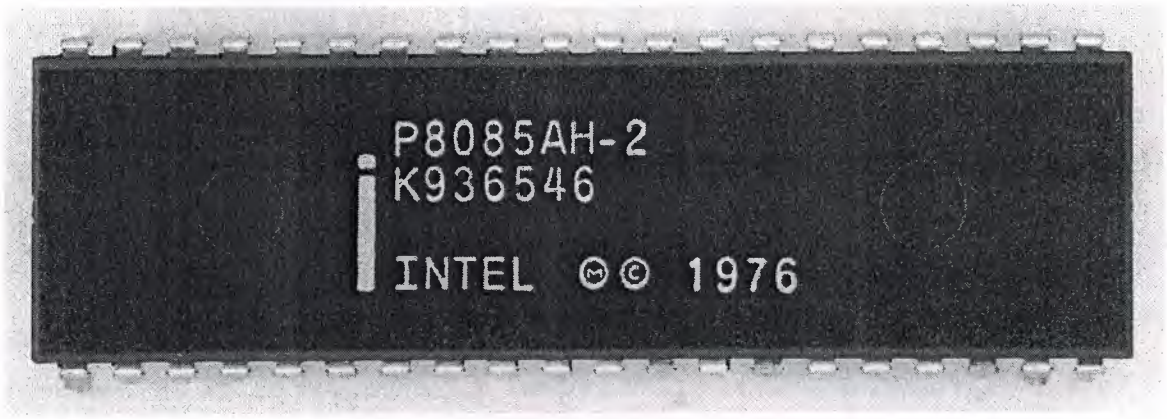


Figure 2-1- 8085A µp

Intel produced a series of development systems for the 8080 and 8085, known as the Personal Development System. The original PDS was a large box (in the Intel corporate blue color) which included a CPU and monitor, and used 8 inch floppy disks. It ran the ISIS operating system and could also operate an emulator pod and EPROM programmer. The

later PDS was a much more portable unit featuring a small green screen and a 5 1/4 inch floppy disk drive, and ran the ISIS-II operating system. It could also accept a second 8085 processor, allowing a limited form of multi-processor operation where both CPUs shared the screen, keyboard and floppy disk drive. In addition to an 8080/8085 assembler, Intel produced a number of compilers including PL/M-80 and Pascal languages, and a set of tools for linking and statically locating programs to enable them to be burnt into EPROM's and used in embedded systems.

The 8085 can access 2^{16} (= 65,536) individual 8-bit memory locations, or in other words, its address space is 64k bytes. Unlike some other microprocessors of its era, it has a separate address space for up to 2^8 (=256) I/O ports. It also has a built in register array which are usually labeled A(Accumulator), B, C, D, E, H, and L. Further special-purpose registers are the 16-bit Program Counter (PC), Stack Pointer (SP), and 8-bit flag register F. The microprocessor has three hardware based interrupt operations which are found in pins 7 through 9, these are called RST 7.5, RST 6.5, and RST 5.5 respectively. The 8085 has a TRAP interrupt which cannot be disabled (that is, TRAP is a Non-Maskable interrupt or NMI) and an INTR interrupt. Use of the INTR requires an external Programmable Interrupt Controller such as an Intel 8259.

The 8085 can accommodate slower memories through externally generated Wait states (pin 35, READY), and also has provisions for Direct Memory Access (DMA) using HOLD and HLDA signals (pins 39 and 38).

The 8085 processor has found marginal use in small scale computers up to the 21st century. The CMOS version 80C85 of the NMOS/HMOS 8085 processor has/had several manufacturers, and some versions (e.g. Tundra Semiconductor Corporation's CA80C85B) have additional functionality, e.g. extra machine code instructions.

2-1- Internal Architecture of 8085 Microprocessor

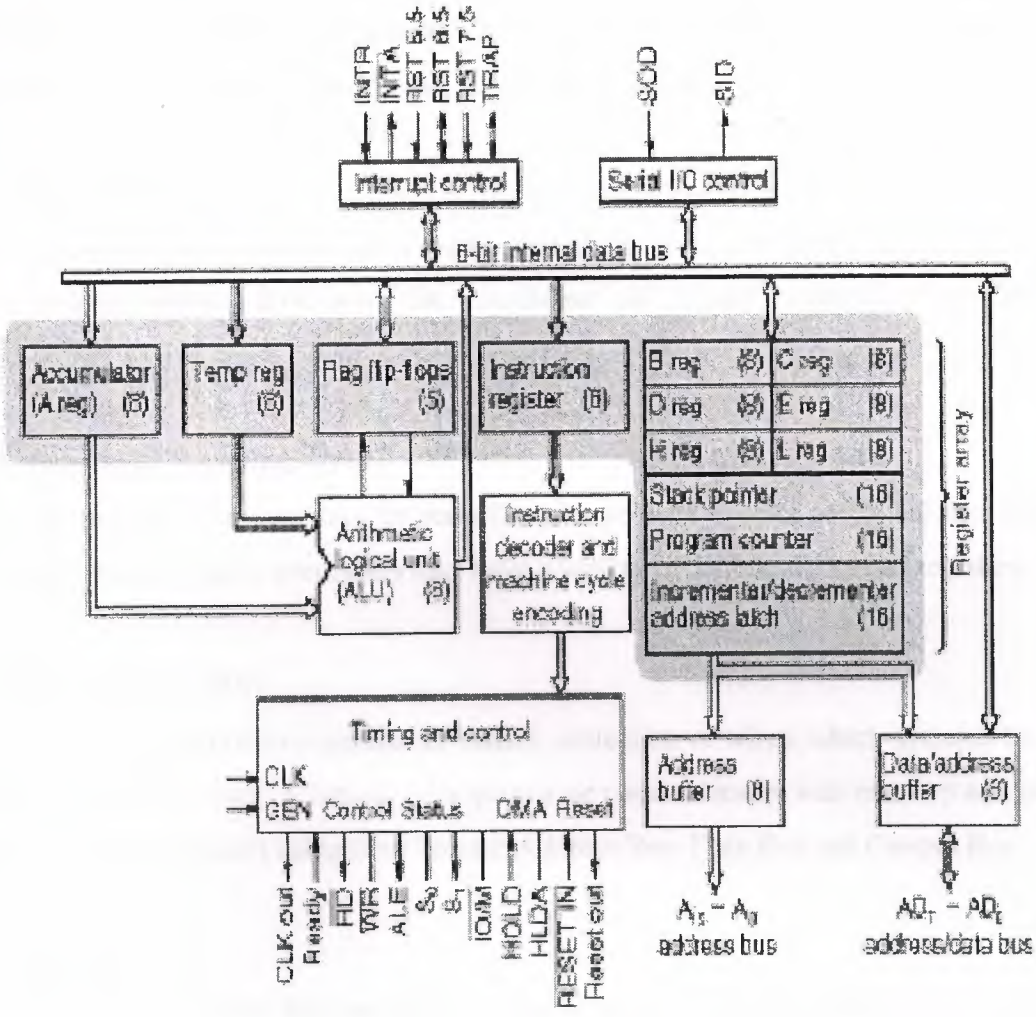


Figure 2-2-Internal Architecture 8085A μp

2-1-1- Memory Address Register

Holds address, received from PC, of next program instruction, feeds the address bus with addresses of location of the program under execution.

2-1-2-Control Generator

Generates signals within μP to carry out the instruction which has been decoded. In reality causes certain connections between blocks of the μP to be opened or closed, so that data goes where it is required, and so that ALU operations occur.

2-1-3-Register Selector

This block controls the use of the register stack in the example. Just a logic circuit which switches between different registers in the set will receive instructions from Control Unit.

2-1-4- General Purpose Registers

μP requires extra registers for versatility can be used to store additional data during a program. More complex processors may have a variety of differently named registers.

2-2- 8085 System Bus

Typical system uses a number of busses, collection of wires, which transmit binary numbers, one bit per wire. A typical microprocessor communicates with memory and other devices (input and output) using three busses: Address Bus, Data Bus and Control Bus.

2-2-1- Address Bus

One wire for each bit, therefore 16 bits = 16 wires. Binary number carried alerts memory to 'open' the designated box. Data (binary) can then be put in or taken out. The Address Bus consists of 16 wires, therefore 16 bits. Its "width" is 16 bits. A 16 bit binary number allows 216 different numbers, or 32000 different numbers, i.e. 0000000000000000 up to 1111111111111111. Because memory consists of boxes, each with a unique address, the size of the address bus determines the size of memory, which can be used. To communicate with memory the microprocessor sends an address on the address bus, e.g. 0000000000000011 (3 in decimal), to the memory. The memory the selects box number 3 for reading or writing data Address bus is unidirectional, i.e. numbers only sent from microprocessor to memory, not other way.

2-2-2- Data Bus

Data Bus: carries 'data', in binary form, between μP and other external units, such as memory. Typical size is 8 or 16 bits. Size determined by size of boxes in memory and μP size helps determine performance of μP. The Data Bus typically consists of 8 wires Therefore 28 combinations of binary digits. Data bus used to transmit "data", i.e. information, results of arithmetic, etc, between memory and the microprocessor. Bus is bi-directional. Size of the data bus determines what arithmetic can be done. If only 8 bits wide then largest number is 11111111 (255 in decimal). Therefore, larger number has to be broken down into chunks of 255. This slows microprocessor. Data Bus also carries instructions from memory to the microprocessor. Size of the bus therefore limits the number of possible instructions to 256, each specified by a separate number.

2-2-3- Control Bus

Control Bus are various lines which have specific functions for coordinating and controlling μP perations. E.g.: Read/NotWrite line, single binary digit. Control whether memory is being 'written to' (data stored in memory) or 'read from' (data taken out of memory) 1 = Read, 0 = Write. May also include clock line(s) for Timing/synchronizing, 'interrupts', 'reset' etc. Typically μP has 10 control lines. Cannot function correctly without these vital control signals.

The Control Bus carries control signals partly unidirectional, partly bi-directional. Controls signals are things like "read or write". This tells memory that we are either reading from a location, specified on the address bus, or writing to a location specified Various other signals to control and coordinate the operation of the system. Modern day microprocessors, like 80386, 80486 have much larger busses. Typically 16 or 32 bit busses which allow larger number of instructions, more memory location and faster arithmetic. Microcontrollers organized along same lines, except: because microcontrollers have memory etc inside the chip, the busses may all be internal. In the microprocessor the three busses are external to the chip (except for the internal data bus). In case of external busses, the chip connects to the busses via buffers, which are simply an electronic connection between external bus and the internal data bus.

2-3- 8085 Pin description

The Intel 8085A is a new generation, complete 8 bit parallel central processing unit (CPU). The 8085A uses a multiplexed data bus. The address is split between the 8bit address bus and the 8bit data bus. Figures are at the end of the document.

Single + 5V Supply

4 Vectored Interrupts (One is Non Maskable)

Serial In/Serial Out Port

Decimal, Binary, and Double Precision Arithmetic

Direct Addressing Capability to 64K bytes of memory

Pin Description

The following describes the function of each pin:

A6 - A1s (Output 3 State)

Address Bus; The most significant 8 bits of the memory address or the 8 bits of the I/0 address,3 stated during Hold and Halt modes.

AD0 - 7 (Input/Output 3state)

Multiplexed Address/Data Bus; Lower 8 bits of the memory address (or I/0 addresses) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles. 3 stated during Hold and Halt modes.

ALE (Output)

Address Latch Enable: It occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. ALE can also be used to strobe the status information. ALE is never 3stated.

SO, S1 (Output)

Data Bus Status Encoded status of the bus cycle:

S1 S0

0 0 HALT

0 1 WRITE

1 0 READ

1 1 FETCH

S1 can be used as an advanced R/W status.

RD (Output 3state)

READ; indicates the selected memory or 1/0 device is to be read and that the Data Bus is available for the data transfer.

WR (Output 3state)

WRITE; indicates the data on the Data Bus is to be written into the selected memory or 1/0 location. Data is set up at the trailing edge of WR. 3 stated during Hold and Halt modes.

READY (Input)

If Ready is high during a read or write cycle, it indicates that the memory or Peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

HOLD (Input)

HOLD; indicates that another Master is requesting the use of the Address and Data Buses. The CPU upon receiving the Hold request will relinquish the use of buses as soon as the completion of the current machine cycle. Internal processing can continue. The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.

HLDA (Output)

HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

INTR (Input)

INTERRUPT REQUEST; is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

INTA (Output)

INTERRUPT ACKNOWLEDGE; is used instead of (and has the same timing as) RD during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

RST 5.5

RST 6.5 - (Inputs)

RST 7.5

RESTART INTERRUPTS; These three inputs have the same timing as I NTR except they cause an internal RESTART to be automatically inserted.

RST 7.5 Highest Priority

RST 6.5

RST 5.5 Lowest Priority

The priority of these interrupts is ordered as shown above. These interrupts have a higher priority than the INTR.

TRAP (Input)

Trap interrupt is a nonmaskable restart interrupt. It is recognized at the same time as INTR. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.

RESET IN (Input)

Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. None of the other flags or registers (except the instruction register) are affected The CPU is held in the reset condition as long as Reset is applied.

RESET OUT (Output)

Indicates CPU is being reset can be used as a system RESET. The signal is synchronized to the processor clock.

X1, X2 (Input)

Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.

CLK (Output)

Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU. The period of CLK is twice the X1, X2 input period.

IO/M (Output)

IO/M indicates whether the Read/Write is to memory or I/O Tristated during Hold and Halt modes.

SID (Input)

Serial input data line The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (output)

Serial output data line. The output SOD is set or reset as specified by the SIM instruction.

Vcc

+5 volt supply.

Vss

Ground Reference.

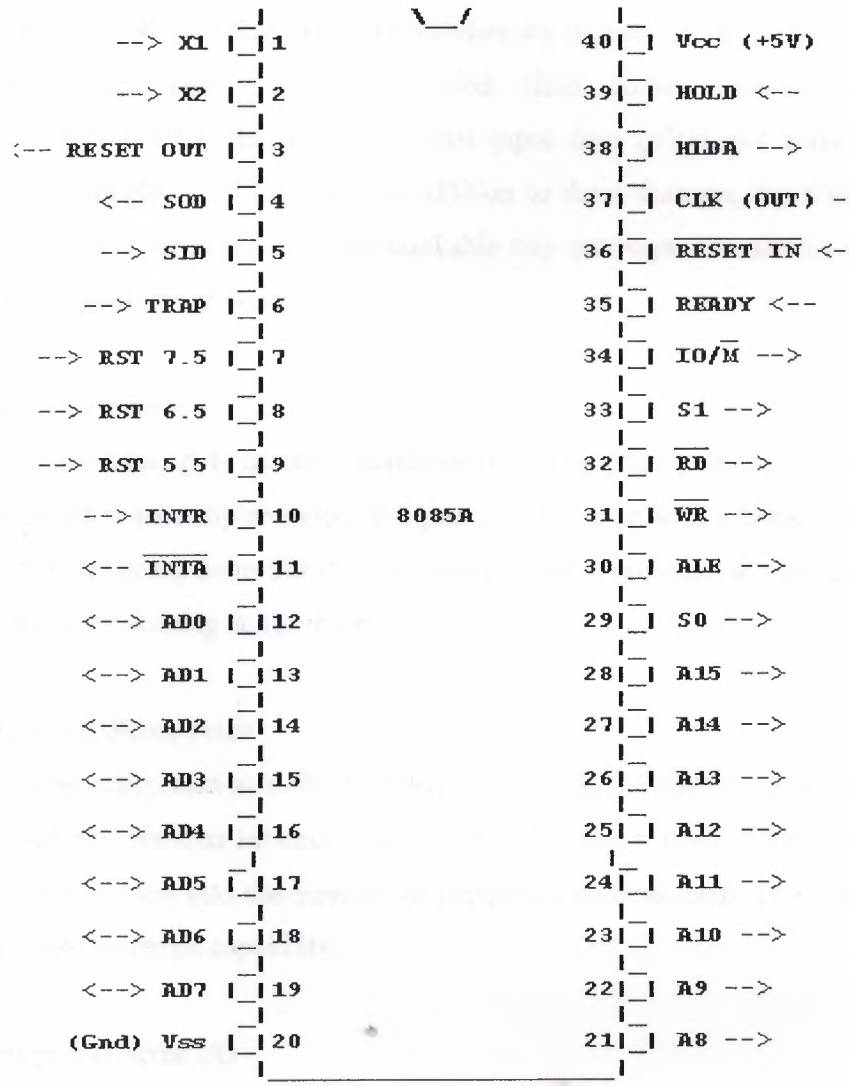


Figure 2-3-8085A Pin Diagram

2-4- 8085 Functional Description

The 8085A is a complete 8 bit parallel central processor. It requires a single +5 volt supply. Its basic clock speed is 3 MHz thus improving on the present 8080's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The CPU, a RAM/ IO, and a ROM or PROM/IO chip.

The 8085A uses a multiplexed Data Bus. The address is split between the higher 8bit Address Bus and the lower 8bit Address/Data Bus. During the first cycle the address is sent out. The lower 8bits are latched into the peripherals by the Address Latch Enable (ALE). During the rest of the machine cycle the Data Bus is used for memory or I/O data.

The 8085A provides RD, WR, and IO/Memory signals for bus control. An Interrupt Acknowledge signal (INTA) is also provided. Hold, Ready, and all Interrupts are synchronized. The 8085A also provides serial input data (SID) and serial output data (SOD) lines for simple serial interface. In addition to these features, the 8085A has three maskable, restart interrupts and one non-maskable trap interrupt. The 8085A provides RD, WR and IO/M signals for Bus control.

2-4-1- Status Information

Status information is directly available from the 8085A. ALE serves as a status strobe. The status is partially encoded, and provides the user with advanced timing of the type of bus transfer being done. IO/M cycle status signal is provided directly also. Decoded So, S1 Carries the following status information:

2-4-2- Halt, Write, Read, Fetch

S1 can be interpreted as R/W in all bus transfers. In the 8085A the 8 LSB of address are multiplexed with the data instead of status. The ALE line is used as a strobe to enter the lower half of the address into the memory or peripheral address latch. This also frees extra pins for expanded interrupt capability.

2-4-3- Interrupt and Serial I/O

The8085A has5 interrupt inputs: INTR, RST5.5, RST6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080 INT. Each of the three RESTART inputs, 5.5, 6.5, 7.5, has a programmable mask. TRAP is also a RESTART interrupt except it is nonmaskable. The three RESTART interrupts cause the internal execution of RST (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The non-maskable TRAP causes the internal execution of a RST independent of the state of the interrupt enable or masks.

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP highest priority, RST 7.5, RST 6.5, RST 5.5, INTR lowest priority This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST5.5 can interrupt a RST 7.5 routine if the interrupts were re-enabled before the end of the RST 7.5 routine. The TRAP interrupt is useful for catastrophic errors such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both edge and level sensitive.

2-4-4-Basic System Timing

The 8085A has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8-bits of address on the Data Bus. Figure shows an instruction fetch, memory read and I/O write cycle (as would occur during processing of the OUT instruction). Note that during the I/O write and read cycle that the I/O port address is copied on both the upper and lower half of the address.

There are seven possible types of machine cycles. Which of these seven takes place is defined by the status of the three status lines (IO/M, S1, S0) and the three control signals (RD, WR and INTA). (See Table 2.1) The status line can be used as advanced controls (for device selection, for example), since they become active at the T1 state, at the outset of each machine cycle. Control lines RD and WR become active later, at the time when the transfer of data is to take place, so are used as command lines. A machine cycle normally consists of three T states, with the exception of OPCODE FETCH, which normally has either four or six T states (unless WAIT or HOLD states are forced by the receipt of READY or HOLD inputs). Any T state must be one of ten possible states, shown in Table.

Machine Cycle		Status			Control		
		IO/$\overline{M}$	S_1	S_0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
Opcode Fetch	(OF)	0	1	1	0	1	1
Memory Read	(MR)	0	1	0	0	1	1
Memory Write	(MW)	0	0	1	1	0	1
I/O Read	(IOR)	1	1	0	0	1	1
I/O Write	(IOW)	1	0	1	1	0	1
Acknowledge of INTR	(INA)	1	1	1	1	1	0
Bus Idle	(BI): DAD	0	1	0	1	1	1
	ACK. OF						
	RST, TRAP	1	1	1	1	1	1
	HALT	TS	0	0	TS	TS	1

Table 2-1- MSM80C85AH Machine Cycle Chart

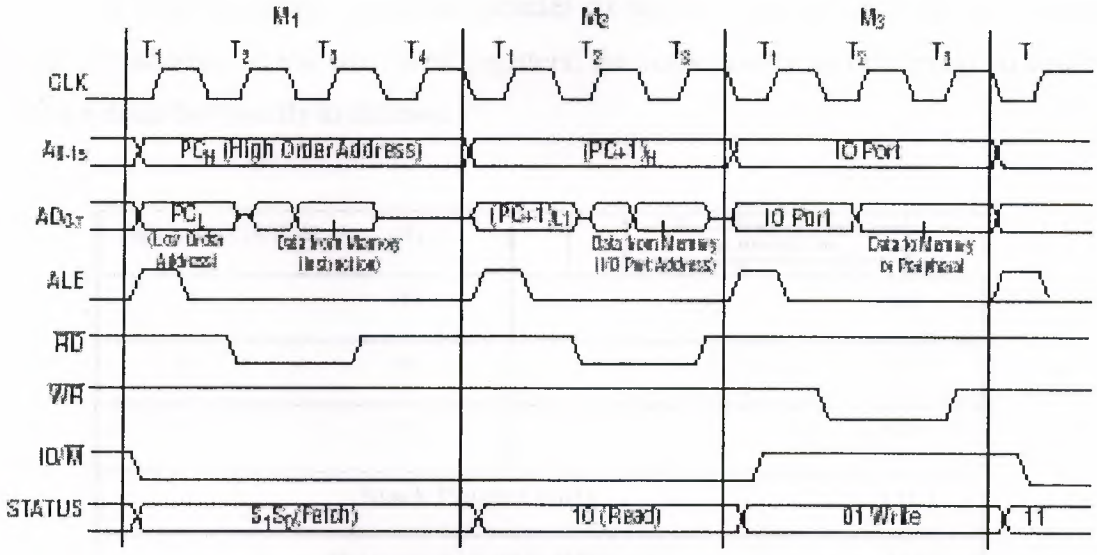


Figure 2-4-MSM80C85AH Basic System Timing

2-4-5- System Interface

8085A family includes memory components, which are directly compatible to the 8085A CPU. For example, a system consisting of the three chips, 8085A, 8156, and 8355 will have the following features:

- 2K Bytes ROM
- 256 Bytes RAM
- 1 Timer/Counter
- 4 8bit I/O Ports

· 1 6bit I/O Port

· 4 Interrupt Levels

· Serial In/Serial Out Ports

In addition to standard I/O, the memory mapped I/O offers an efficient I/O addressing technique. With this technique, an area of memory address space is assigned for I/O address, thereby, using the memory address for I/O manipulation. The 8085A CPU can also interface with the standard memory that does not have the multiplexed address/data bus.

2-5- The 8085 Programming Model

The 8085 programming model includes six registers, one accumulator, and one flag register, In addition, it has two 16-bit registers: the stack pointer and the program counter. They are described briefly as follows.

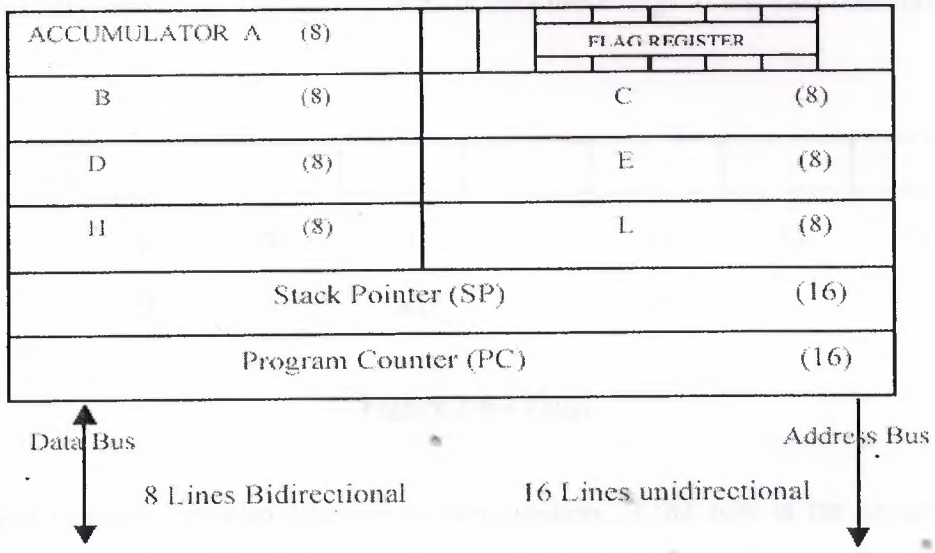


Figure 2-5- 8085 programming model

2-5- 1- Registers

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L as shown in the figure. They can be combined as register pairs - BC,

DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

2-5- 2- Accumulator

The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

2-5- 3- Flags

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero(Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags; their bit positions in the flag register are shown in the Figure below. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.

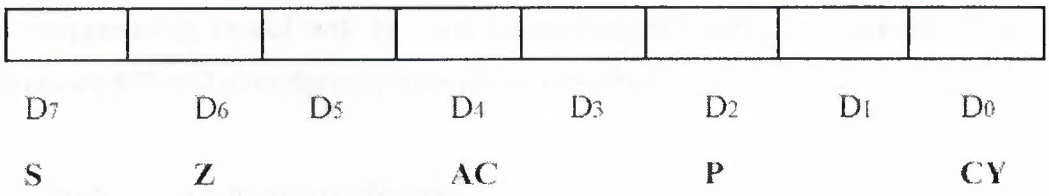


Figure 2-6-- Flags

For example, after an addition of two numbers, if the sum in the accumulator id larger than eight bits, the flip-flop uses to indicate a carry called the Carry flag (CY) is set to one. When an arithmetic operation results in zero, the flip-flop called the Zero(Z) flag is set to one. The Figure shows an 8-bit register, called the flag register, adjacent to the accumulator. However, it is not used as a register; five bit positions out of eight are used to store the outputs of the five flip-flops. The flags are stored in the 8-bit register so that the programmer can examine these flags (data conditions) by accessing the register through an instruction. These flags have critical importance in the decision-making process of the

micro-processor. The conditions (set or reset) of the flags are tested through the software instructions. For example, the instruction JC (Jump on Carry) is implemented to change the sequence of a program when CY flag is set. The thorough understanding of flag is essential in writing assembly language programs.

2-5-4- Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.

2-5-5- Stack Pointer (SP)

The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

This programming model will be used in subsequent tutorials to examine how these registers are affected after the execution of an instruction.

2-6- The 8085 Addressing Modes

The instructions MOV B, A or MVI A, 82H are to copy data from a source into a destination. In these instructions the source can be a register, an input port, or an 8-bit number (00H to FFH). Similarly, a destination can be a register or an output port. The sources and destination are operands. The various formats for specifying operands are called the ADDRESSING MODES. For 8085, they are:

1. Immediate addressing.
2. Register addressing.
3. Direct addressing.
4. Indirect addressing.

Immediate addressing Data is present in the instruction. Load the immediate data to the destination provided.

Example: MVI R, data

Register addressing

Data is provided through the registers.

Example: MOV Rd, Rs

Direct addressing Used to accept data from outside devices to store in the accumulator or send the data stored in the accumulator to the outside device. Accept the data from the port 00H and store them into the accumulator or Send the data from the accumulator to the port 01H.

Example: IN 00H or OUT 01H

Indirect Addressing this means that the Effective Address is calculated by the processor. And the contents of the address (and the one following) are used to form a second address. The second address is where the data is stored. Note that this requires several memory accesses; two accesses to retrieve the 16-bit address and a further access (or accesses) to retrieve the data which is to be loaded into the register.

2-7- Instruction Set Classification

An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the **instruction set**, determines what functions the microprocessor can perform. These instructions can be classified into the following five functional categories: data transfer (copy) operations, arithmetic operations, logical operations, branching operations, and machine-control operations.

2-7- 1- Data Transfer (Copy) Operations

This group of instructions copy data from a location called a source to another location called a destination, without modifying the contents of the source. In technical manuals, the term data transfer is used for this copying function. However, the term transfer is misleading; it creates the impression that the contents of the source are destroyed when, in fact, the contents are retained without any modification.

The various types of data transfer (copy) are listed below together with examples of each type:

Types	Examples
1. Between Registers.	1. Copy the contents of the register B into register D.
2. Specific data byte to a register or a memory location.	2. Load register B with the data byte 32H.
3. Between a memory location and a register.	3. From a memory location 2000H to register B.
4. Between an I/O device and the accumulator.	4.From an input keyboard to the accumulator.

Table 2-3- Data Types

2-7- 2- Arithmetic Operations

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.

Addition: Any 8-bit number, or the contents of a register or the contents of a memory location can be added to the contents of the accumulator and the sum is stored in the accumulator. No two other 8-bit registers can be added directly (e.g., the contents of register B cannot be added directly to the contents of the register C). The instruction DAD is an exception; it adds 16-bit data directly in register pairs.

Subtraction: Any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator. The subtraction is performed in 2's compliment, and the results if negative, are expressed in 2's complement. No two other registers can be subtracted directly.

Increment/Decrement : The 8-bit contents of a register or a memory location can be incremented or decrement by 1. Similarly, the 16-bit contents of a register pair (such- as BC) can be incremented or decrement by 1. These increment and decrement operations

differ from addition and subtraction in an important way; i.e., they can be performed in any one of the registers or in a memory location.

2-7- 3- Logical Operations

These instructions perform various logical operations with the contents of the accumulator.

AND, OR Exclusive-OR : Any 8-bit number, or the contents of a register, or of a memory location can be logically ANDed, Ored, or Exclusive-Ores with the contents of the accumulator. The results are stored in the accumulator.

Rotate: Each bit in the accumulator can be shifted either left or right to the next position.

Compare: Any 8-bit number or the contents of a register, or a memory location can be compared for equality, greater than, or less than, with the contents of the accumulator.

Complement : The contents of the accumulator can be complemented. All 0s are replaced by 1s and all 1s are replaced by 0s.

2-7- 4- Branching Operations

This group of instructions alters the sequence of program execution either conditionally or unconditionally.

Jump : Conditional jumps are an important aspect of the decision-making process in the programming. These instructions test for a certain conditions (e.g., Zero or Carry flag) and alter the program sequence when the condition is met. In addition, the instruction set includes an instruction called *unconditional jump.*

Call, Return, and Restart : These instructions change the sequence of a program either by calling a subroutine or returning from a subroutine. The conditional Call and Return instructions also can test condition flags.

2-7- 5- Machine Control Operations

These instructions control machine functions such as Halt, Interrupt, or do nothing. The microprocessor operations related to data manipulation can be summarized in four functions:

- Copying data
-.Performing arithmetic operations
- Performing logical operations
- Testing for a given condition and alerting the program sequence

Some important aspects of the instruction set are noted below:

In data transfer, the contents of the source are not destroyed; only the contents of the destination are changed. The data copy instructions do not affect the flags.

Arithmetic and Logical operations are performed with the contents of the accumulator, and the results are stored in the accumulator (with some expectations). The flags are affected according to the results.

Any register including the memory can be used for increment and decrement.

A program sequence can be changed either conditionally or by testing for a given data condition.

2-8- Instruction Format

An **instruction** is a command to the microprocessor to perform a given task on a specified data. Each instruction has two parts: one is task to be performed, called the **operation code** (opcode), and the second is the data to be operated on, called the **operand.** The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit. Instruction word size The 8085 instruction set is classified into the following three groups according to word size:

One-word or 1-byte instructions

Two-word or 2-byte instructions

Three-word or 3-byte instructions

In the 8085, "byte" and "word" are synonymous because it is an 8-bit microprocessor. However, instructions are commonly referred to in terms of bytes rather than words.

2-8- 1- One-Byte Instructions

A 1-byte instruction includes the opcode and operand in the same byte. Operand(s) are internal register and are coded into the instruction.

For example:

Task	Op code	Operand	Binary Code	Hex Code
Copy the contents of the accumulator in the register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (compliment) each bit in the accumulator.	CMA		0010 1111	2FH

Table 2-4- One Byte Instruction Example

These instructions are 1-byte instructions performing three different tasks. In the first instruction, both operand registers are specified. In the second instruction, the operand B is specified and the accumulator is assumed. Similarly, in the third instruction, the accumulator is assumed to be the implicit operand. These instructions are stored in 8- bit binary format in memory; each requires one memory location.

MOV rd, rs

rd <-- rs copies contents of rs into rd.

Coded as 01 ddd sss where ddd is a code for one of the 7 general registers which is the destination of the data, sss is the code of the source register.

Example: MOV A,B

Coded as 01111000 = 78H = 170 octal (octal was used extensively in instruction design of such processors).

ADD r

A <-- A + r

2-8- 2- Two-Byte Instructions

In a two-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Source operand is a data byte immediately following the opcode. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A, Data	0011 1110	3E	First Byte
			DATA	Data	Second Byte

Table 2-5-Two Byte Instruction Example

Assume that the data byte is 32H. The assembly language instruction is written as

Mnemonics	Hex code
MVI A, 32H	3E 32H

The instruction would require two memory locations to store in memory.

MVI r, data

r <-- data

Example: MVI A, 30H coded as 3EH 30H as two contiguous bytes. This is an Example of immediate addressing;

ADI data

A <-- A + data

OUT port: Where port is an 8-bit device address (Port) <-- A. Since the byte is not the data but points directly to where it is located this is called direct addressing.

2-8- 3-Three-Byte Instructions

In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address.

opcode + data byte + data byte

For example:

Task	Opcode	Operand	Binary code	Hex Code	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011 1000 0101 0010 0000	C3 85 20	First byte Second Byte Third Byte

Table 2-6-Three Byte Instruction Example

This instruction would require three memory locations to store in memory.

Three byte instructions - opcode + data byte + data byte LXI rp, data16 rp is one of the pairs of registers BC, DE, HL used as 16-bit registers. The two data bytes are 16-bit data in L H order of significance.

rp <-- data16

Example:

LXI H, 0520H coded as 21H 20H 50H in three bytes. This is also immediate addressing. LDA addr A <-- (addr) Addr is a 16-bit address in L H order. Example: LDA 2134H coded as 3AH 34H 21H. This is also an example of direct addressing.

CHAPTER 3

8085 MICROPROCESSOR TRAINING SYSTEM

The Primer is a low cost 8085 based training tool developed specifically for learning the operation of today's microprocessor based systems. Microwave ovens, stereos, TVs, and almost every other electronic product utilizes embedded microprocessor technology. The Primer Trainer demonstrates the principles used by those products, providing you the opportunity to program, interface, and control virtually any device. Whether you're exploring climate control, data logging or just experimenting on your own, the Primer Trainer has the features needed to learn the necessary programming and interfacing skills required for today's world. The Primer Trainer has the ability to process analog signals (like temperature and voltage) as well as digital signals (like switches and relays). Add the capability to process those signals at precisely timed, interrupt driven intervals, and now you're talking real time embedded control. Maybe now is the time to replace those aging SDK85 trainers.

The Primer Trainer's 8085 microprocessor is an ideal platform for learning microprocessor theory. The straightforward 8085 architecture is easy to understand and the instruction set is powerful allowing the use of programming techniques similar to those used for the PC, but much simpler to learn. Since the system may be purchased as a kit, circuit assembly techniques can also be learned.

The Primer Trainer's low cost helps it fit into almost any curriculum. Its rugged construction allows it to be used by schools year after year. The Primer is affordable, allowing students to purchase it like a textbook to use and keep. For schools or individuals, the cost/performance ratio makes the Primer the most economical full-featured training system we know of.

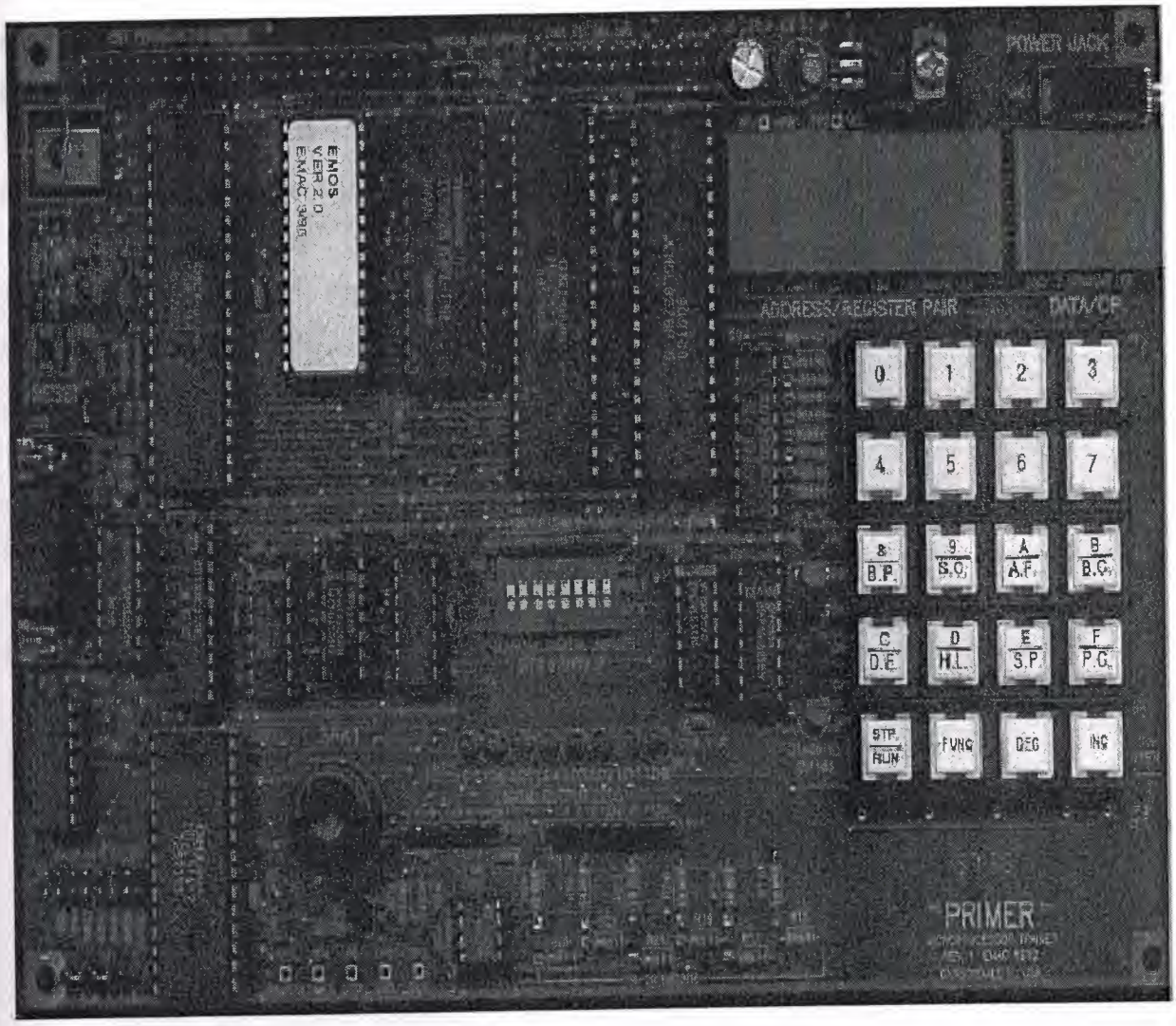


Figure 3-1- Primer Trainer

3-1- Hardware Features

Knowledge gained on the Primer applies directly to computers that are widely used in engineering and business applications. ICs used include:

- 8085 Microprocessor.
- 8155 Programmable peripheral interface with timer and RAM.
- 8279 Keypad and display controller.
- 8251 Optional UART serial controller.

20 Key, Keypad.

6 digit, 7 segment, led display.

8 position dipswitch input port with I/O connector.

8 bit output port with status leds and I/O connector.

Analog to digital converter.

Digital to analog converter.

Timer output with speaker.

14 bit timer with interrupt support.

System reset button.

50 pin bus expansion connector.

Allows interfacing of most devices.

Available as a kit or assembled.

Easy to assemble.

Rugged construction to stand up to hard use.

Ideal replacement for aging SDK85 trainers.

3-2- Programming Features

* 8K EPROM containing monitor operating system (mos) allows the user to:

- Display and edit memory.
- Display and edit registers.
- Display and edit top of stack.
- Single step by instruction.
- Run full speed with breakpoint.
- Utilize MOS internal subroutines for each I/O device as well as for multiply, divide, getkey, display number, and display ASCII.

The Monitor Operating System (the standard monitor software in EPROM) included will cause the keyboard controller chip to scan the keypad buttons, interpret the entries, and allow storage of data into the 256 byte RAM included in the PRIMER. The operating system also displays related data on the six (6) digit LED display. The operating system will allow examining the data stored at various memory locations, examining the contents of the microprocessor's registers and other functions, via the display and keypad.

The operating system software will permit to run (execute) the program entered, and to execute the program in steps, instruction by instruction, to permit debugging of programs.

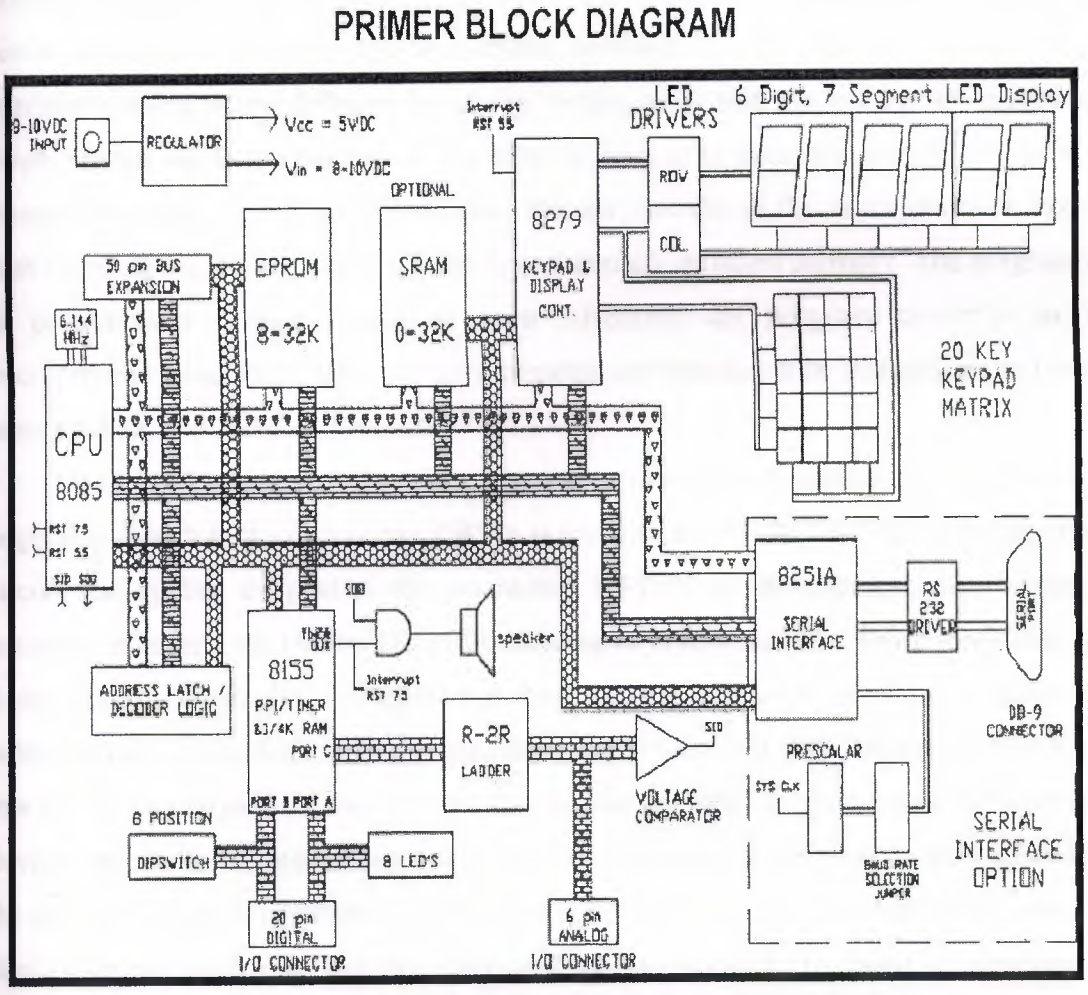


Figure 3-2- Primer Block Diagram

3-3- Microprocessor and Expansion Bus

3-3-1- The Microprocessor

The 8085 microprocessor chip is the brain of the PRIMER system. It coordinates almost all activity in the PRIMER. The 8085 microprocessor is a collection of counters, gates, registers, decoders, (etc.) that sequentially fetches instructions from memory, finds the purposes of each instruction, and executes the purpose of each instruction. The instructions are placed in memory in the order they are expected to be used. The types of instructions are widely variant, but very concise. They operations consist of moving data,

logical operations, branching and conditional branching, some mathematical operations, and input/output. When correctly assembled into a logical order, these primitive instructions form what is called a program. Programs can simply move data from an input port to an output port, using a few instructions, or they can perform complex control operations using many different input and output ports and the number of instructions would stretch out to the thousands. The 8085 is built to be able to access 65,536 possible memory locations, and 256 I/O locations. The architecture of the microprocessor expects both the program and the data to reside in sequentially arranged memory. The programmer (a person who creates programs) must assemble the program correctly so the microprocessor will know what memory contents are intended to be instructions and which ones are data.

The 8085 and most other microprocessors must start executing the program at its beginning, which in the case of the 8085 is at memory address 0. This is accomplished by a circuit that applies a signal to the pin named RST-IN on the microprocessor when the computer is turned on. On the PRIMER, this signal comes from the Power-on/Pushbutton Reset circuit . When power is applied to the PRIMER, capacitor C1, initially discharged, holds the microprocessor's RST-IN* pin, low (the "*" in "RST-IN*" means signal is active low). The low signal on the RST-IN pin causes the 8085 to reset some of its internal devices and make an internal register called the program counter point to the beginning of the program. Another event that occurs is that RST-OUT signal is asserted, which resets the display/keypad controller and PPI chip used in the PRIMER. If EMAC peripherals are connected to the expansion port they are also affected by the RST-OUT signal. This reset condition can also be brought about by pressing the reset button (PB1) which merely shorts out C1 and brings the RST-IN pin low. As the capacitor charges through R1, the RST-IN* line voltage rises until a logic high de-asserts this signal. Diode D1 serves to quickly discharge capacitor C1 through the power supply only when power is off. With RST-IN* de-asserted, the 8085 begins operation.

The system clock oscillator, whose frequency is set to 6.144 MHZ by crystal Y1, drives all timing functions within the 8085 and it runs as long as power is applied to the PRIMER. It is possible to run the 8085 under a wide range of frequencies, but in the PRIMER use 6.144 MHZ to provide compatibility with circuits used elsewhere in the

system. The oscillator signal is divided by two within the 8085, and this signal is now the main system clock, referred to as SYSCLK. All timing of the microprocessor's operation is driven by this SYSCLK, which in our case is 3.072 MHZ. Even asynchronous input signals like RST-IN*, and interrupts are internally synchronized to this oscillator. Each pulse of this clock signal, is called a "T state", when referring to operations internal to the 8085 and its busses.

The first thing the microprocessor will do after RST-IN* de-asserts, is to fetch an instruction at memory address 0. Before the instruction can be fetched, the memory address of the instruction must be output from the microprocessor. To reduce the number of pins on the 8085's package, a scheme called multiplexing is employed. Multiplexing allows the AD0-AD7 pins to be used as the data bus and also to output the lower 8 bits of the 16 bit address needed to select the first instruction. These pins must be demultiplexed before they can be used properly.

The 8085 outputs the low byte of the address on signal lines AD0-AD7, and the high byte directly on signal lines A8-A15. A signal called Address Latch Enable (ALE) strobes high, then low, causing the address latch (chip U8) to trap the address byte that is on pins AD0-AD7. The output of this latch then joins up with address lines A8-A15, to provide a 16 bit, filtered, stable, linear address to the memory chips and decoders .

The 16 bit linear address is applied to the memory block (memories and decoders) to select the address that the 8085 wants to read from or write to. In the case of an instruction fetch, the 8085 signal RD* will be asserted, to read the byte from the memory selected. This byte will then go into the instruction decoder, which will determine what instruction it is, and then the 8085 will execute the instruction. Depending on what type of instruction it is, the 8085 will either fetch additional data or read/write data to other memory locations or perform I/O, (etc.). Each time a bus transaction occurs these operations will occur: the low byte of the address will be sent to AD0-AD7, which will then be held by the address latch. A complete cycle of instruction fetch, decode, and execution, forms a "bus cycle", using anywhere from 3 to 14 T-States, (SYSCLK pulses) depending on the type of instruction.

The 8085 has five pins dedicated to "interrupting" the microprocessor. These pins are named TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. When a signal is applied to one

of these pins an "interrupt" is generated. An interrupt is a special function of the 8085 to suspend the program it is running then execute a program related to the interrupt that requested it. Once the interrupt is completed, the 8085 must RESTORE the original program's data and status, and RETURN to it where it left off from. Sometimes the interrupt is deliberately ignored by MASKING the interrupt, so interrupts can be turned on or off as needed, should a section of code be so important to require that it not be interrupted until it is finished.

An interrupt may be likened to answering the telephone while reading a book. When the telephone rings, you put the book down on it's face to mark your place, then you answer the phone. When you are done, you pick up the book and begin reading where you left off. Should you be too engrossed in the story, you can ignore the ringing telephone (you can mask it). But if you ignore it too long, you'll miss the call. Sometimes interrupts can be missed as well. Some interrupts can be made to latch, usually by hardware. These latched interrupt requests will persist, just as a person on the phone might continue ringing (interrupting) until you answer it.

3-3-2- Bus Expansion Port

The Primer's Bus Expansion PORT is actually a simple connector with leads running to power sources and microprocessor pins. The 8085's AD0-AD7 lines run out to the connector, as well as address lines A0-A15, control signal IO/M* (which the 8085 asserts to tell whether it is doing a memory or I/O transaction), RD*, WR*, RST-OUT, SYSCLK, GROUND, and PWR (this is directly from the power jack, 5V is not provided on the bus connector). Also included are INTRQ, and INTA*, which allow the use of a multi-level interrupt controller; S0, and S1, which are part of the 8085 machine status output; RDY/WT* which is used to put the 8085 into a wait state; HOLDRQ and HLDACK which are used to allow another bus master to assume control of the bus; and finally, ID* and EXTIOCS*, which are used by external I/O devices of the EPAC 2000 family. The Bus Expansion Port footprint permits direct connection of the PRIMER to most EPAC peripheral cards, via a 50 pin ribbon cable. The type of module port (shape of the module) sometimes gives helpful information about the kind of connections made. For example, module port D[0..7] is a bi-directional port, and all occurrences of this port on

other pages denote this is a bi-directional 8 line bus interconnection. Individual lines to module ports also connect these lines to corresponding points on other sheets of the schematic, such as lines RD*, and WR*.

3-4- I/O Decoding and Memory

The decoder section of a microprocessor system sorts out the address bus signals from the 8085, and picks out (selects) the correct chip(s) in the memory or I/O sections that the 8085 wants to access. The decoder must evaluate the address applied to it, and determine whether the address is for a memory access, or an I/O map access. Most decoders are custom hardware collections of gates, using various levels of integration. As every system has a different configuration of memory and I/O, every decoder is unique to that design.

In the PRIMER, two types of memory maps may be decoded. In the standard map, a 16K (K=1024 bytes) byte space is reserved for program EPROM, starting at address 0000 hex ending at 3FFF hex. (Remember the 8085 begins execution at 0000 hex after reset) This socket may hold 8K, 16K or 32K EPROM chips. The monitor operating system program is actually very small, requiring less than 4K, but 8K EPROM chips are the smallest memory chip of compatible footprint that will fit there.

Almost all programs (the Primer's monitor is no exception) will require some memory that can be *written to* as well as read. The PRIMER provides this type of memory, without additional discrete RAM chips. The 8155 PPI/MEMORY/TIMER I.C. is a multi-function chip, with digital I/O ports, a timer/counter, and 256 bytes of RAM in it. This memory is used for storage of variables, pointers, executable code and various other functions. Because the 8155 has both I/O and memory mapped elements, the decoder section provides an additional specially encoded line called 8155CS for this device.

The optional 32 K byte memory socket allows a 32 K static RAM chip to be installed onto the PRIMER. Option selection jumpers, OJ2 and OJ3 resolve the hardware differences between the 8 K, 16 K and 32 K EPROM chips, and allow the addition of the 32 K byte memory and by rearranging the memory map to accommodate the different chips. This is necessary when software options such as E-FORTH and MTBASIC are

installed. The memory can be altered to use the two memory maps described. In the case of the EPROM chip, an 8 K device (2764) or a 16 K device (27128) may be used interchangeably, bearing in mind the smaller sized devices do not fill the entire memory map. When the alternate memory map is used, a 32 K device, (27256) may be installed in the EPROM socket. The difference of the pinout in the 32 K device is the reason for the option jumpers. Jumper OJ2 switches pin 27 between A14 or Vcc, as required by the 32 K device. The 8155 PPI's memory space begins at C000 hex and ends at the top of the 8085's address range at FFFF hex. Note that this range is 16 K, but the actual memory is only 256 bytes. What happens is the same 256 bytes repeat themselves over and over again, in this case 64 times over. In other words, hexadecimal addresses C000, C100, F300 and FD00 all refer to the first memory address within the 8155 and C0FF, C1FF, F3FF and FDFF all refer to the last memory address within the 8155. Programs that are written must use only the amount of memory that is available.

The alternate memory map reserves 32 K of EPROM at addresses 0000 hex to 7FFF hex, and 32 K static Ram from 8000 hex to FFFF hex. The 256 bytes of RAM in the 8155 PPI are ignored, so in this map, the entire 64 K of memory that the 8085 is capable of accessing does not contain memory cells with duplicate addresses.

The PRIMER uses a very simple decoder. Two, "2 to 4 line" decoders decode the appropriate 8085 address lines to select the proper chips. The memory decoder observes the code on address lines A14 and A15. The state of these lines represents 16 K intervals of address. To change the decoder interval to 32 K bytes boundaries, the decoder pin that is connected to A14 can instead be connected to ground. The I/O decoder observes the state of address lines A6 and A7. The I/O map is only 256 addresses wide. The I/O decoder is maximally used at select rates of 64 addresses per output.

The 8085 signal line IO/M* tells the decoders that the address is an I/O MAP transaction, if it is high, or that it is a memory map transaction, if it is low. A simple inverter changes the polarity of IO/M* to IO*/M. Since each decoder has an active low enable input, connection of the appropriate polarity signal of IO/M* to the appropriate decoder, will enable the right decoder for the job at hand. Also all decoder outputs are active low signals, and will be de-asserted unless the right address in the right map selects it.

The EPROM chip select will be valid for memory addresses 0000 hex to 3FFF hex, (or to 7FFF hex if non-standard EPROM) and simply goes directly to the EPROM's chip enable. The optional static RAM chip select must be valid for addresses 4000 hex to BFFF hex, (or 8000 to FFFF hex if nonstandard) which is an interval of 32 K. As the memory decoder outputs at intervals of 16 K, (standard map) two chip selects are logically negative OR'ed by AND gate U10A. This produces a single 32 K wide chip select for this memory.

The 8155 chip has elements of both the memory map and the I/O map, so a similar AND gate logically negative OR's the memory and I/O map selects into a single dual mapped chip select for the 8155. The memory portion of this chip select is valid throughout C000 hex to FFFF hex. The I/O portion is I/O addresses 00 hex to 3F hex.

The 8279 keypad and display controller is I/O mapped from 40 hex to 7F hex, and the optional 8251A communication controller from I/O addresses 80 hex to BF hex. A special I/O chip select, used primarily for EPAC expansion peripherals, via the bus expansion connector, uses the remaining I/O space of C0 hex to FF hex. As in the memory chip select description, many of the addresses repeat in the I/O map, since these chips only uses a few of the available addresses.

The rest of the address bus lines not used by the chip select decoder described, go directly to the memory chip(s) other I/O chips, and bus expansion connector. Inside these chips are decoders which select the precise internal memory cells or device registers denoted by the address pins. In more complex systems, the chip select decoders can be more precise, but this increases the complexity and cost of the system. The simple device select decoder used here in the PRIMER will serve quite well, as long as programs are written carefully.

3-5- Display and Keypad Circuitry

The PRIMER has a key pad matrix of pushbuttons, for entering commands, loading data or programs, and testing programs. It also has a 6 digit LED matrix display, to view data. The 8279 is a specialized controller chip, which removes the chores of keyboard scan, switch de-bounces, and display refresh, from the microprocessor. The 8279 contains its own memory which stores key press data and data to be displayed. Through the I/O map, the 8085 sends commands to the 8279 that configure the operational characteristics of the

display/keypad system, then writes the display characters themselves to the 8279's internal display RAM. The 8279 scans this display RAM, and drives the outputs to the display LED's continuously thereafter. No further microprocessor action need be taken until a new display pattern is needed.

The 8279 has 8 data bus lines that connect directly to the 8085's AD0 to AD7 lines. These bi-directional bus leads transfer data and control information between the 8085 and the 8279 Display/Keypad controller. Signals RD*, and WR* determine whether data is read from or written to the 8279. Address line A0 is used by the 8279 to determine whether the data byte is a command byte, or a data byte for the internal RAM and the 8279's chip select pin determines whether it will perform bus transactions or not. Because address line A0 is used to select the 8279 function, two I/O map addresses exist for the Display/Keypad controller. A program that operates the controller merely writes to or reads from two consecutive addresses to control all 8279 functions. With A0 being low, the 8279 memory can receive data from, or send data to the data bus. If A0 is high, the bus transaction is a command or status byte. The base address of the 8279 in the PRIMER is 40 hex. The I/O decoder's chip select will be valid for all I/O map transactions up to 7F hex. Therefore, the two addresses for the 8279 repeat themselves throughout this range, just as the memory addresses repeat in the 8155. This repetition is common to all I/O devices on the PRIMER, due to the broad nature of the chip select decoder used.

The signals RST-OUT, SYSCLK, and KEYINT also terminate at the 8279. RST-IN will hardware clear the 8279's internal control registers to their default state, which at a later time can be changed to other states by the Primer's program in the EPROM. However, the data in the internal RAM is not cleared by hardware. The SYSCLK signal (the microprocessor's 3.072 MHz master clock signal), is divided internally by the 8279 to provide the keypad scan and LED display refresh clocks. Finally, the KEYINT signal is a special control line that is set active high by the 8279 when a valid key entry is made. This line goes through a jumper that (in its default connection) connects to the RST 5.5 interrupt pin on the 8085 microprocessor.

The 8279 scans the keypad buttons and debounces the key inputs to avoid erroneous entries due to contact bounce or key rollover. After this, it stores the key inputs in the internal keypad RAM. When a valid key entry is made, the 8279 generates an interrupt

NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

DESIGNING AN ELECTRONIC ORGAN USING 8085 MICROPROCESSOR

Graduation Project
COM- 400

Student : Süleyman Kerimoğlu (20010522)

Supervisor : Prof. Dr Doğan İbrahim

Nicosia – 2006

ACKNOWLEDGEMENT

My gratitude for the following people cannot be express in this short acknowledgement.

First, I wish to thank to my supervisor, Prof. Dr Doğan İbrahim , whose encouragement and enthusiasms have been constant sources of inspiration to me. Also his invaluable advices and belief in my work and myself over the course of this Graduation Project.

Second, I would like to express our gratitude to Near East University (NEU) for the scholarship that made the work possible. Also special thanks starting from our Dean Prof. Dr Fakhreddin Mamedov and all other lecturers for the advices and helps.

I dedicate this project to my home friends Hüseyin Tulunay,Sinan Özerenler ,Ozan Akkoça and my best friends Mesut Arık , Göktuğ Ataç .

Finally, I would like to thank all our friends for their advices , supports and being with me with a big respect and patience.

ABSTRACT

Microprocessors are electronic equipments that is used in most of place which requires electronic control mechanism. They include from hundred thousands to millions of transistors. Microprocessors are used in traffic control systems, music systems, electronic devices at home, computers to meet the need of global world.

Most of music devices used to produce as mechanical devices in the past. But, as a reason of development in new technologies, they have been produced by using electronic equipment. Such an electronic organ, lots of electronic music device are able to hold songs (also notes) in their storage part, and whenever it is needed, they can play it automatically. This happens via microcontrollers and microchips. They are playing very important role in music equipment like in our lives.

As a graduation project, I decided to prepare a very simple electronic organ, using the 8085 Microprocessor and assembly language to code it. So, this project book presents electronic structure of microprocessors, music notes definitions in binary system and also assembly language code examples briefly.

TABLE OF CONTENTS

LIST OF ABREVATIONS

ALU	Arithmetic Logic Unit
ALE	Address Latch Enable
CPU	Central Procesing Unit
ALU	Arithmetic Logic Unit
INTR	Interrupt
I/O	Input/ Output
LSB	Least Signifigant Bit
µp	Microprocesor
RAM	Random Access Memory
R/W	Read/Write
RP	Register Pair
RST	Restart
PC	Program Counter
SP	Stack Pointer
SID	Serial Input Data
SOD	Serial Output Data
A/D	Anolog to Digital
D/A	Digital to Analog
PDS	Personal Development System

INTRODUCTION

In response of development in microprocessors, electronic devices are used in most of music equipments. As a result of interest in music instruments and hardware devices, I decided to prepare a simple electronic organ.

After some research, it becomes definitive to use a 8085 microcontrollers and assembly language. Some requirements such as a speaker and buttons are investigated. I used a PRIMER 8085 Microprocessor Trainer System because, it contains an internal speaker to get voice at specific frequencies and internal numpad to enter assembly numerical codes.

Chapter 1 presents definition of microprocessor, historical development and evoluation. Chapter also includes the attributes of general purpose CPUs of some companies.

Chapter 2 presents attributes of 8085 Microprocessor, internal architecture and bus systems of this microprocessor. It also provides explanation about pin structure, system interface and programming methods of 8085 microprocessor. A mnemonics opcode instruction set table is also provided in chapter.

Chapter 3 presents the PRIMER Training System properties. Chapter also investigates hardware and programming features of the device, bus system and the internal equipments such as seven segment display, numpad, speaker, timer, counter, power supply and communication ports.

Chapter 4 provides a hexadecimal values of notes table. The included part is very important for the project, because representation of music notes in digital binary system is the evoluational idea of the electronical music equipments.

Chapter 5 describes assembly codes that is used in project. An algorithm shcheme to play seven music notes and ordered code structure of the assembly program is included in the chapter.

CHAPTER 1
MICROPROCESSOR

Microprocessor is a multipurpose, programmable, clock-driven, register based electronic device that reads binary instructions from a storage device called memory, accepts binary data as input and processes data according to those instructions, and provides as output.

A common way of categorizing microprocessors is by the number of bits that their ALU can work time. In other words a microprocessor with a 4 bit will be referred to as a 4-bit microprocessor, regardless of the number of address lines or the number of data bus lines that it has.

The first commercially available microprocessor was the Intel 4004 produced in 1971. It contained 2300 PMOS transistors. The 4004~was a 4 bit device intended to be used with some other devices in making a calculator. Some logic designers however saw that this device could be used to replace PC boards full of combinational and sequential logic devices. Also the ability to change the function of a system by just changing the programming, rather than redesigning the hardware is very appealing.

In 1972 Intel came out with the 8008, which was capable of working with 8-bit words. The 8008, however required 20 or more additional devices to form a functional CPU In 1974 Intel announced the 8080, which had a much larger instruction set than the 8008 and required only two additional devices to form a functional CPU. Also the 8080 used NMOS transistors so it operated much faster than the 8008. The 8080 is referred to as a second-generation microprocessor. It requires +12 V power supply.

Soon after Intel produced the 8080, Motorola came out with the MC6800 another 8-bit general-purpose CPU. The 6800 had the advantage that it required only a +5-V supply rather than the -5-V. +5-V. and + 12V supplies required by the 8080.

For several years the 8080 and the 6800 were the top-selling 8-bit microprocessors. Some of their competitors were the MOS Technology 6502. used as the CPU in the Apple II microcomputer and the Zilog

Z80 used as the CPU in the Radio Shack TRS~O microcomputer.

1-1- The Major Evolution of Microprocessor

Three major directions of microprocessor Evolutions are

(i) Dedicated or Embedded Controllers

(ii) Bit Slice Processors

(iii) General purpose CPUs

1-1-1- Dedicated or Embedded Controllers

One direction has been dedicated or embedded controllers. These devices are used to control "smart" machines such as microwave ovens Clothes washers. Sewing machines auto ignition systems and metal lathes.

Texas Instruments has produced millions of their TMS-1000 family of 4-bit microprocessors for this type of application.

In 1976 Intel introduced the 8048, which contains an 8-bit CPU, RAM, ROM. and some I/O ports all in one 40-pin package. Other manufacturers have followed with similar products. These devices are often referred to as microcontrollers.

Some currently available devices in this category-the Intel 8051 and the Motorola MC6801 a more recently introduced single chip microcontroller, the Intel 8096, contains a 16-bit CPU,ROM, RAM, a UART, ports, timers, and a 10-bit analog-to-digital converter.

1-1-2- Bit Slice Processors

A second direction of microprocessor evolution has been bit-slice processors. For some applications general purpose CPUs such as the 8080 and 6800 are not fast enough or do not have suitable instruction sets. For these applications, several manufacturers produce devices which can be used to build a custom CPU.

An example is the Advanced Micro Devices 2900 family of devices. This family includes 4-bit ALU's, Multiplexers, sequencers and other parts needed for custom-building a CPU. The term *slice* comes from the fact that these parts can be connected in parallel to work with 8-bit words, 16-bit words. or 32-bit words. In other words a designer can add as many slices as needed for a particular application.

The designer not only custom-designs the hardware of the CPU, but also custom-

makes the instruction set for it using "microcode".

1-1-3- General Purpose CPUs

The third major direction of microprocessor evolution has been toward general-purpose CPUs which give a microcomputer most or all of the computing power of earlier minicomputers.

Intel 8085:

After Motorola came out with the MC6800 Intel produced the 8085. an upgrade of the 8080 that required only a +5-V supply.

Motorola MC 6809:

Motorola then produced the MC6809 which has a few 16-bit instructions. but is still basically an 8-bit processor.

Intel 8086:

In 1978 Intel came out with the 8086 which is a full 16bit processor. Some 16-bit microprocessors such as the National PACE and the Texas Instruments 9900 family of devices had been available previously. but the market apparently was not ready.

Motorola MC68000:

Soon after Intel came out with the 8086, Motorola came out with the 16-bit MC68000.

The 8086 and the 68000 work directly with 16-bit words instead of with 8-bit words they can address a million or more bytes of memory instead of the 64 Kbytes addressable by the 8-bit processors and they execute instructions much faster than the 8-bit processors. Also these 16bit processors have single instructions for functions such as multiply and divide. This required a lengthy sequence of instructions on the 8-bit processors.

The evolution along this last path has continued on to 32-bit processors that work with gigabytes (109 bytes) or terabytes (1012 bytes) of memory. Examples of these devices are the Intel 80386. the Motorola MC68020, and the National 32032.

CHAPTER 2

Intel 8085 Microprocessor

The Intel 8085 was an 8-bit microprocessor made by Intel in the mid-1970s. It was binary compatible with the more-famous Intel 8080 but required less supporting hardware, thus allowing simpler and less expensive microcomputer systems to be built.

The "5" in the model number came from the fact that the 8085 required only a +5-volt (V) power supply rather than the +5V, -5V and +12V supplies the 8080 needed. Both processors were sometimes used in computers running the CP/M operating system, and the 8085 later saw use as a microcontroller (much by virtue of its component count reducing feature). Both designs were later eclipsed by the compatible but more capable Zilog Z80, which took over most of the CP/M computer market as well as taking a large share of the booming home computer market in the early-to-mid-1980s.

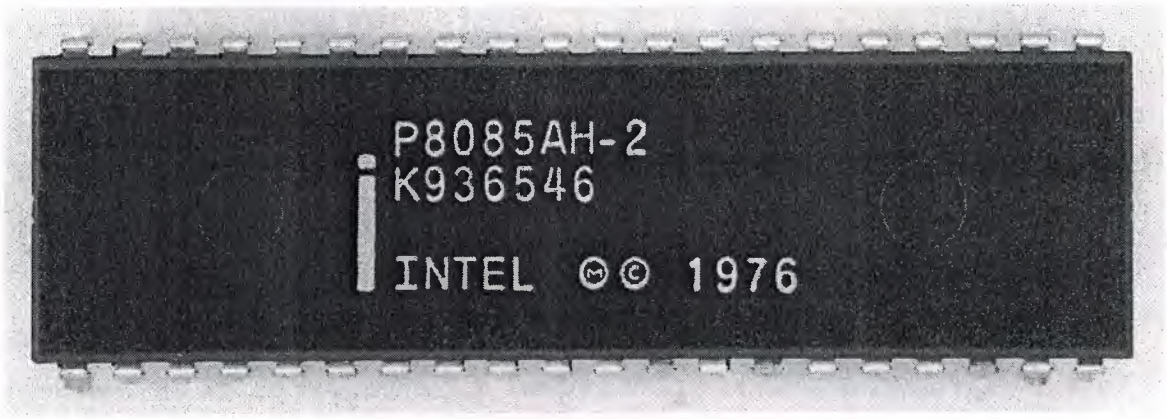


Figure 2-1- 8085A µp

Intel produced a series of development systems for the 8080 and 8085, known as the Personal Development System. The original PDS was a large box (in the Intel corporate blue color) which included a CPU and monitor, and used 8 inch floppy disks. It ran the ISIS operating system and could also operate an emulator pod and EPROM programmer. The

later PDS was a much more portable unit featuring a small green screen and a 5 1/4 inch floppy disk drive, and ran the ISIS-II operating system. It could also accept a second 8085 processor, allowing a limited form of multi-processor operation where both CPUs shared the screen, keyboard and floppy disk drive. In addition to an 8080/8085 assembler, Intel produced a number of compilers including PL/M-80 and Pascal languages, and a set of tools for linking and statically locating programs to enable them to be burnt into EPROM's and used in embedded systems.

The 8085 can access 2^{16} (= 65,536) individual 8-bit memory locations, or in other words, its address space is 64k bytes. Unlike some other microprocessors of its era, it has a separate address space for up to 2^8 (=256) I/O ports. It also has a built in register array which are usually labeled A(Accumulator), B, C, D, E, H, and L. Further special-purpose registers are the 16-bit Program Counter (PC), Stack Pointer (SP), and 8-bit flag register F. The microprocessor has three hardware based interrupt operations which are found in pins 7 through 9, these are called RST 7.5, RST 6.5, and RST 5.5 respectively. The 8085 has a TRAP interrupt which cannot be disabled (that is, TRAP is a Non-Maskable interrupt or NMI) and an INTR interrupt. Use of the INTR requires an external Programmable Interrupt Controller such as an Intel 8259.

The 8085 can accommodate slower memories through externally generated Wait states (pin 35, READY), and also has provisions for Direct Memory Access (DMA) using HOLD and HLDA signals (pins 39 and 38).

The 8085 processor has found marginal use in small scale computers up to the 21st century. The CMOS version 80C85 of the NMOS/HMOS 8085 processor has/had several manufacturers, and some versions (e.g. Tundra Semiconductor Corporation's CA80C85B) have additional functionality, e.g. extra machine code instructions.

2-1- Internal Architecture of 8085 Microprocessor

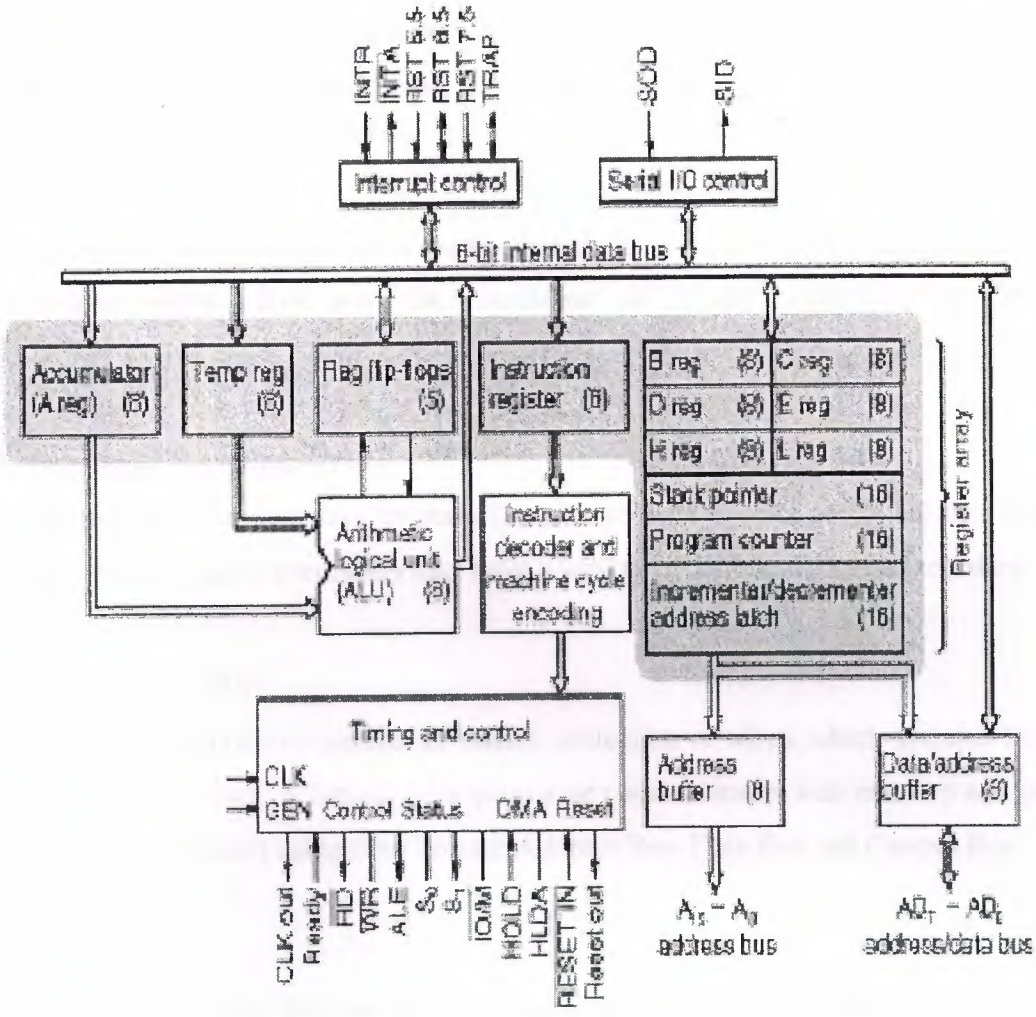


Figure 2-2-Internal Architecture 8085A μp

2-1-1- Memory Address Register

Holds address, received from PC, of next program instruction, feeds the address bus with addresses of location of the program under execution.

2-1-2-Control Generator

Generates signals within μP to carry out the instruction which has been decoded. In reality causes certain connections between blocks of the μP to be opened or closed, so that data goes where it is required, and so that ALU operations occur.

2-1-3-Register Selector

This block controls the use of the register stack in the example. Just a logic circuit which switches between different registers in the set will receive instructions from Control Unit.

2-1-4- General Purpose Registers

μP requires extra registers for versatility can be used to store additional data during a program. More complex processors may have a variety of differently named registers.

2-2- 8085 System Bus

Typical system uses a number of busses, collection of wires, which transmit binary numbers, one bit per wire. A typical microprocessor communicates with memory and other devices (input and output) using three busses: Address Bus, Data Bus and Control Bus.

2-2-1- Address Bus

One wire for each bit, therefore 16 bits = 16 wires. Binary number carried alerts memory to 'open' the designated box. Data (binary) can then be put in or taken out. The Address Bus consists of 16 wires, therefore 16 bits. Its "width" is 16 bits. A 16 bit binary number allows 216 different numbers, or 32000 different numbers, i.e. 0000000000000000 up to 1111111111111111. Because memory consists of boxes, each with a unique address, the size of the address bus determines the size of memory, which can be used. To communicate with memory the microprocessor sends an address on the address bus, e.g. 0000000000000011 (3 in decimal), to the memory. The memory the selects box number 3 for reading or writing data Address bus is unidirectional, i.e. numbers only sent from microprocessor to memory, not other way.

2-2-2- Data Bus

Data Bus: carries 'data', in binary form, between μP and other external units, such as memory. Typical size is 8 or 16 bits. Size determined by size of boxes in memory and μP size helps determine performance of μP. The Data Bus typically consists of 8 wires Therefore 28 combinations of binary digits. Data bus used to transmit "data", i.e. information, results of arithmetic, etc, between memory and the microprocessor. Bus is bi-directional. Size of the data bus determines what arithmetic can be done. If only 8 bits wide then largest number is 11111111 (255 in decimal). Therefore, larger number has to be broken down into chunks of 255. This slows microprocessor. Data Bus also carries instructions from memory to the microprocessor. Size of the bus therefore limits the number of possible instructions to 256, each specified by a separate number.

2-2-3- Control Bus

Control Bus are various lines which have specific functions for coordinating and controlling μP perations. E.g.: Read/NotWrite line, single binary digit. Control whether memory is being 'written to' (data stored in memory) or 'read from' (data taken out of memory) 1 = Read, 0 = Write. May also include clock line(s) for Timing/synchronizing, 'interrupts', 'reset' etc. Typically μP has 10 control lines. Cannot function correctly without these vital control signals.

The Control Bus carries control signals partly unidirectional, partly bi-directional. Controls signals are things like "read or write". This tells memory that we are either reading from a location, specified on the address bus, or writing to a location specified Various other signals to control and coordinate the operation of the system. Modern day microprocessors, like 80386, 80486 have much larger busses. Typically 16 or 32 bit busses which allow larger number of instructions, more memory location and faster arithmetic. Microcontrollers organized along same lines, except: because microcontrollers have memory etc inside the chip, the busses may all be internal. In the microprocessor the three busses are external to the chip (except for the internal data bus). In case of external busses, the chip connects to the busses via buffers, which are simply an electronic connection between external bus and the internal data bus.

2-3- 8085 Pin description

The Intel 8085A is a new generation, complete 8 bit parallel central processing unit (CPU). The 8085A uses a multiplexed data bus. The address is split between the 8bit address bus and the 8bit data bus. Figures are at the end of the document.

Single + 5V Supply

4 Vectored Interrupts (One is Non Maskable)

Serial In/Serial Out Port

Decimal, Binary, and Double Precision Arithmetic

Direct Addressing Capability to 64K bytes of memory

Pin Description

The following describes the function of each pin:

A6 - A1s (Output 3 State)

Address Bus; The most significant 8 bits of the memory address or the 8 bits of the I/0 address,3 stated during Hold and Halt modes.

AD0 - 7 (Input/Output 3state)

Multiplexed Address/Data Bus; Lower 8 bits of the memory address (or I/0 addresses) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles. 3 stated during Hold and Halt modes.

ALE (Output)

Address Latch Enable: It occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. ALE can also be used to strobe the status information. ALE is never 3stated.

SO, S1 (Output)

Data Bus Status Encoded status of the bus cycle:

S1 S0

0 0 HALT

0 1 WRITE

1 0 READ

1 1 FETCH

S1 can be used as an advanced R/W status.

RD (Output 3state)

READ; indicates the selected memory or 1/0 device is to be read and that the Data Bus is available for the data transfer.

WR (Output 3state)

WRITE; indicates the data on the Data Bus is to be written into the selected memory or 1/0 location. Data is set up at the trailing edge of WR. 3 stated during Hold and Halt modes.

READY (Input)

If Ready is high during a read or write cycle, it indicates that the memory or Peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

HOLD (Input)

HOLD; indicates that another Master is requesting the use of the Address and Data Buses. The CPU upon receiving the Hold request will relinquish the use of buses as soon as the completion of the current machine cycle. Internal processing can continue. The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.

HLDA (Output)

HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

INTR (Input)

INTERRUPT REQUEST; is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

INTA (Output)

INTERRUPT ACKNOWLEDGE; is used instead of (and has the same timing as) RD during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

RST 5.5

RST 6.5 - (Inputs)

RST 7.5

RESTART INTERRUPTS; These three inputs have the same timing as I NTR except they cause an internal RESTART to be automatically inserted.

RST 7.5 Highest Priority

RST 6.5

RST 5.5 Lowest Priority

The priority of these interrupts is ordered as shown above. These interrupts have a higher priority than the INTR.

TRAP (Input)

Trap interrupt is a nonmaskable restart interrupt. It is recognized at the same time as INTR. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.

RESET IN (Input)

Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. None of the other flags or registers (except the instruction register) are affected The CPU is held in the reset condition as long as Reset is applied.

RESET OUT (Output)

Indicates CPU is being reset can be used as a system RESET. The signal is synchronized to the processor clock.

X1, X2 (Input)

Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.

CLK (Output)

Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU. The period of CLK is twice the X1, X2 input period.

IO/M (Output)

IO/M indicates whether the Read/Write is to memory or I/O Tristated during Hold and Halt modes.

SID (Input)

Serial input data line The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (output)

Serial output data line. The output SOD is set or reset as specified by the SIM instruction.

Vcc

+5 volt supply.

Vss

Ground Reference.

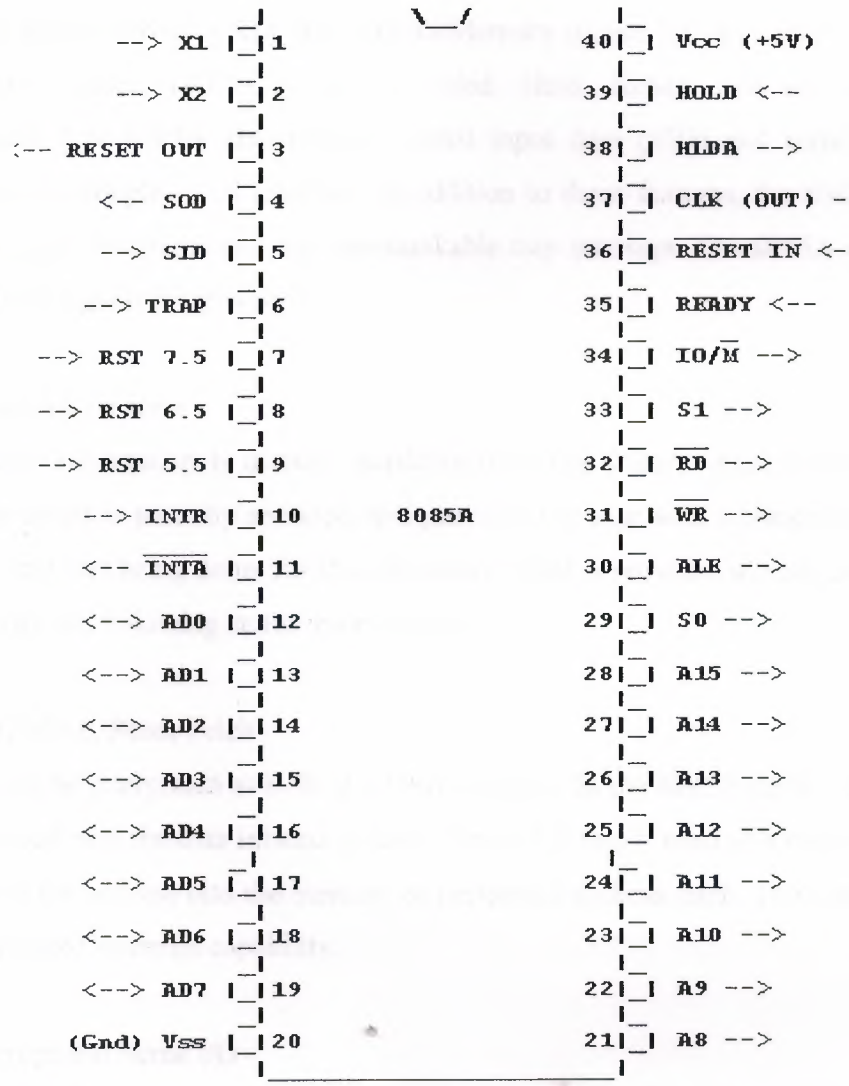


Figure 2-3-8085A Pin Diagram

2-4- 8085 Functional Description

The 8085A is a complete 8 bit parallel central processor. It requires a single +5 volt supply. Its basic clock speed is 3 MHz thus improving on the present 8080's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The CPU, a RAM/ IO, and a ROM or PROM/IO chip.

The 8085A uses a multiplexed Data Bus. The address is split between the higher 8bit Address Bus and the lower 8bit Address/Data Bus. During the first cycle the address is sent out. The lower 8bits are latched into the peripherals by the Address Latch Enable (ALE). During the rest of the machine cycle the Data Bus is used for memory or I/O data.

The 8085A provides RD, WR, and IO/Memory signals for bus control. An Interrupt Acknowledge signal (INTA) is also provided. Hold, Ready, and all Interrupts are synchronized. The 8085A also provides serial input data (SID) and serial output data (SOD) lines for simple serial interface. In addition to these features, the 8085A has three maskable, restart interrupts and one non-maskable trap interrupt. The 8085A provides RD, WR and IO/M signals for Bus control.

2-4-1- Status Information

Status information is directly available from the 8085A. ALE serves as a status strobe. The status is partially encoded, and provides the user with advanced timing of the type of bus transfer being done. IO/M cycle status signal is provided directly also. Decoded So, S1 Carries the following status information:

2-4-2- Halt, Write, Read, Fetch

S1 can be interpreted as R/W in all bus transfers. In the 8085A the 8 LSB of address are multiplexed with the data instead of status. The ALE line is used as a strobe to enter the lower half of the address into the memory or peripheral address latch. This also frees extra pins for expanded interrupt capability.

2-4-3- Interrupt and Serial I/O

The8085A has5 interrupt inputs: INTR, RST5.5, RST6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080 INT. Each of the three RESTART inputs, 5.5, 6.5, 7.5, has a programmable mask. TRAP is also a RESTART interrupt except it is nonmaskable. The three RESTART interrupts cause the internal execution of RST (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The non-maskable TRAP causes the internal execution of a RST independent of the state of the interrupt enable or masks.

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP highest priority, RST 7.5, RST 6.5, RST 5.5, INTR lowest priority This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST5.5 can interrupt a RST 7.5 routine if the interrupts were re-enabled before the end of the RST 7.5 routine. The TRAP interrupt is useful for catastrophic errors such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both edge and level sensitive.

2-4-4-Basic System Timing

The 8085A has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8-bits of address on the Data Bus. Figure shows an instruction fetch, memory read and I/O write cycle (as would occur during processing of the OUT instruction). Note that during the I/O write and read cycle that the I/O port address is copied on both the upper and lower half of the address.

There are seven possible types of machine cycles. Which of these seven takes place is defined by the status of the three status lines (IO/M, S1, S0) and the three control signals (RD, WR and INTA). (See Table 2.1) The status line can be used as advanced controls (for device selection, for example), since they become active at the T1 state, at the outset of each machine cycle. Control lines RD and WR become active later, at the time when the transfer of data is to take place, so are used as command lines. A machine cycle normally consists of three T states, with the exception of OPCODE FETCH, which normally has either four or six T states (unless WAIT or HOLD states are forced by the receipt of READY or HOLD inputs). Any T state must be one of ten possible states, shown in Table.

Machine Cycle		Status			Control		
		IO/$\overline{M}$	S_1	S_0	$\overline{RD}$	$\overline{WR}$	$\overline{INTA}$
Opcode Fetch	(OF)	0	1	1	0	1	1
Memory Read	(MR)	0	1	0	0	1	1
Memory Write	(MW)	0	0	1	1	0	1
I/O Read	(IOR)	1	1	0	0	1	1
I/O Write	(IOW)	1	0	1	1	0	1
Acknowledge of INTR	(INA)	1	1	1	1	1	0
Bus Idle	(BI): DAD	0	1	0	1	1	1
	ACK. OF						
	RST, TRAP	1	1	1	1	1	1
	HALT	TS	0	0	TS	TS	1

Table 2-1- MSM80C85AH Machine Cycle Chart

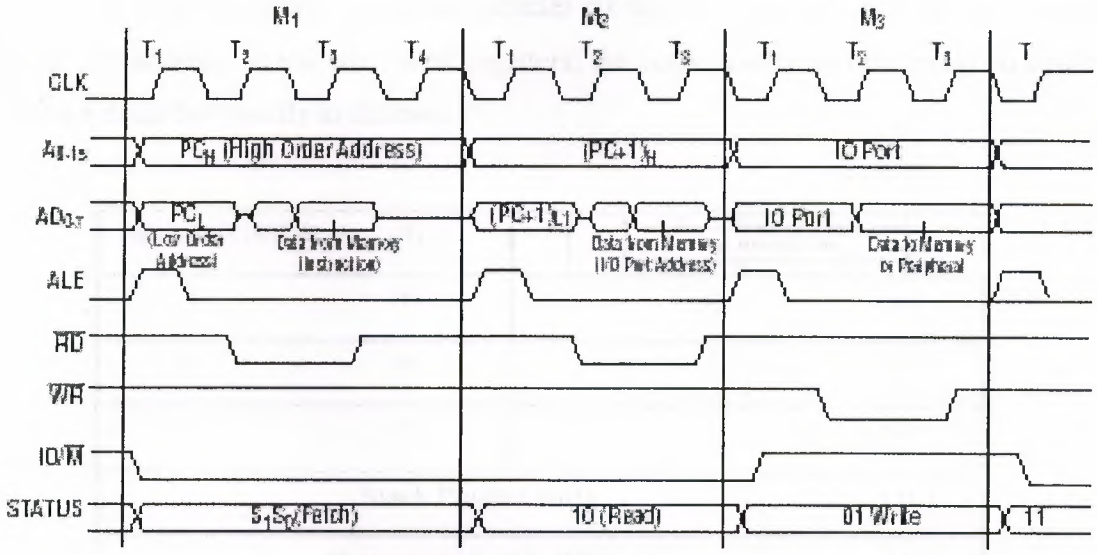


Figure 2-4-MSM80C85AH Basic System Timing

2-4-5- System Interface

8085A family includes memory components, which are directly compatible to the 8085A CPU. For example, a system consisting of the three chips, 8085A, 8156, and 8355 will have the following features:

- 2K Bytes ROM
- 256 Bytes RAM
- 1 Timer/Counter
- 4 8bit I/O Ports

· 1 6bit I/O Port

· 4 Interrupt Levels

· Serial In/Serial Out Ports

In addition to standard I/O, the memory mapped I/O offers an efficient I/O addressing technique. With this technique, an area of memory address space is assigned for I/O address, thereby, using the memory address for I/O manipulation. The 8085A CPU can also interface with the standard memory that does not have the multiplexed address/data bus.

2-5- The 8085 Programming Model

The 8085 programming model includes six registers, one accumulator, and one flag register, In addition, it has two 16-bit registers: the stack pointer and the program counter. They are described briefly as follows.

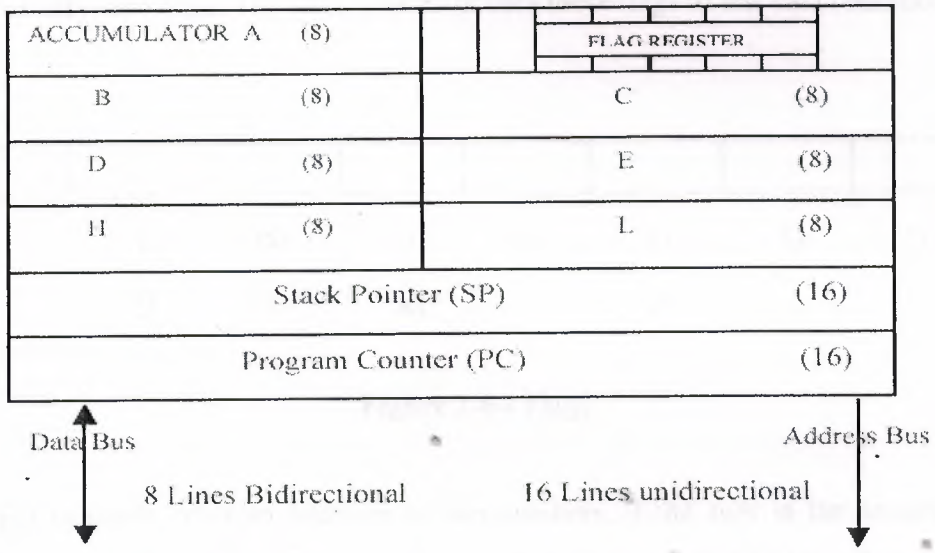


Figure 2-5- 8085 programming model

2-5- 1- Registers

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L as shown in the figure. They can be combined as register pairs - BC,

DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

2-5- 2- Accumulator

The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

2-5- 3- Flags

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero(Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags; their bit positions in the flag register are shown in the Figure below. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.

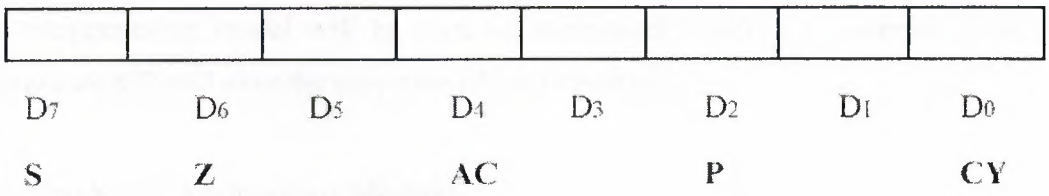


Figure 2-6-- Flags

For example, after an addition of two numbers, if the sum in the accumulator id larger than eight bits, the flip-flop uses to indicate a carry called the Carry flag (CY) is set to one. When an arithmetic operation results in zero, the flip-flop called the Zero(Z) flag is set to one. The Figure shows an 8-bit register, called the flag register, adjacent to the accumulator. However, it is not used as a register; five bit positions out of eight are used to store the outputs of the five flip-flops. The flags are stored in the 8-bit register so that the programmer can examine these flags (data conditions) by accessing the register through an instruction. These flags have critical importance in the decision-making process of the

micro-processor. The conditions (set or reset) of the flags are tested through the software instructions. For example, the instruction JC (Jump on Carry) is implemented to change the sequence of a program when CY flag is set. The thorough understanding of flag is essential in writing assembly language programs.

2-5-4- Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location.

2-5-5- Stack Pointer (SP)

The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

This programming model will be used in subsequent tutorials to examine how these registers are affected after the execution of an instruction.

2-6- The 8085 Addressing Modes

The instructions MOV B, A or MVI A, 82H are to copy data from a source into a destination. In these instructions the source can be a register, an input port, or an 8-bit number (00H to FFH). Similarly, a destination can be a register or an output port. The sources and destination are operands. The various formats for specifying operands are called the ADDRESSING MODES. For 8085, they are:

1. Immediate addressing.
2. Register addressing.
3. Direct addressing.
4. Indirect addressing.

Immediate addressing Data is present in the instruction. Load the immediate data to the destination provided.

Example: MVI R, data

Register addressing

Data is provided through the registers.

Example: MOV Rd, Rs

Direct addressing Used to accept data from outside devices to store in the accumulator or send the data stored in the accumulator to the outside device. Accept the data from the port 00H and store them into the accumulator or Send the data from the accumulator to the port 01H.

Example: IN 00H or OUT 01H

Indirect Addressing this means that the Effective Address is calculated by the processor. And the contents of the address (and the one following) are used to form a second address. The second address is where the data is stored. Note that this requires several memory accesses; two accesses to retrieve the 16-bit address and a further access (or accesses) to retrieve the data which is to be loaded into the register.

2-7- Instruction Set Classification

An **instruction** is a binary pattern designed inside a microprocessor to perform a specific function. The entire group of instructions, called the **instruction set**, determines what functions the microprocessor can perform. These instructions can be classified into the following five functional categories: data transfer (copy) operations, arithmetic operations, logical operations, branching operations, and machine-control operations.

2-7- 1- Data Transfer (Copy) Operations

This group of instructions copy data from a location called a source to another location called a destination, without modifying the contents of the source. In technical manuals, the term data transfer is used for this copying function. However, the term transfer is misleading; it creates the impression that the contents of the source are destroyed when, in fact, the contents are retained without any modification.

The various types of data transfer (copy) are listed below together with examples of each type:

Types	Examples
1. Between Registers.	1. Copy the contents of the register B into register D.
2. Specific data byte to a register or a memory location.	2. Load register B with the data byte 32H.
3. Between a memory location and a register.	3. From a memory location 2000H to register B.
4. Between an I/O device and the accumulator.	4.From an input keyboard to the accumulator.

Table 2-3- Data Types

2-7- 2- Arithmetic Operations

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement.

Addition: Any 8-bit number, or the contents of a register or the contents of a memory location can be added to the contents of the accumulator and the sum is stored in the accumulator. No two other 8-bit registers can be added directly (e.g., the contents of register B cannot be added directly to the contents of the register C). The instruction DAD is an exception; it adds 16-bit data directly in register pairs.

Subtraction: Any 8-bit number, or the contents of a register, or the contents of a memory location can be subtracted from the contents of the accumulator and the results stored in the accumulator. The subtraction is performed in 2's compliment, and the results if negative, are expressed in 2's complement. No two other registers can be subtracted directly.

Increment/Decrement : The 8-bit contents of a register or a memory location can be incremented or decrement by 1. Similarly, the 16-bit contents of a register pair (such- as BC) can be incremented or decrement by 1. These increment and decrement operations

differ from addition and subtraction in an important way; i.e., they can be performed in any one of the registers or in a memory location.

2-7- 3- Logical Operations

These instructions perform various logical operations with the contents of the accumulator.

AND, OR Exclusive-OR : Any 8-bit number, or the contents of a register, or of a memory location can be logically ANDed, Ored, or Exclusive-Ores with the contents of the accumulator. The results are stored in the accumulator.

Rotate: Each bit in the accumulator can be shifted either left or right to the next position.

Compare: Any 8-bit number or the contents of a register, or a memory location can be compared for equality, greater than, or less than, with the contents of the accumulator.

Complement : The contents of the accumulator can be complemented. All 0s are replaced by 1s and all 1s are replaced by 0s.

2-7- 4- Branching Operations

This group of instructions alters the sequence of program execution either conditionally or unconditionally.

Jump : Conditional jumps are an important aspect of the decision-making process in the programming. These instructions test for a certain conditions (e.g., Zero or Carry flag) and alter the program sequence when the condition is met. In addition, the instruction set includes an instruction called *unconditional jump.*

Call, Return, and Restart : These instructions change the sequence of a program either by calling a subroutine or returning from a subroutine. The conditional Call and Return instructions also can test condition flags.

2-7- 5- Machine Control Operations

These instructions control machine functions such as Halt, Interrupt, or do nothing. The microprocessor operations related to data manipulation can be summarized in four functions:

- Copying data
-.Performing arithmetic operations
- Performing logical operations
- Testing for a given condition and alerting the program sequence

Some important aspects of the instruction set are noted below:

In data transfer, the contents of the source are not destroyed; only the contents of the destination are changed. The data copy instructions do not affect the flags.

Arithmetic and Logical operations are performed with the contents of the accumulator, and the results are stored in the accumulator (with some expectations). The flags are affected according to the results.

Any register including the memory can be used for increment and decrement.

A program sequence can be changed either conditionally or by testing for a given data condition.

2-8- Instruction Format

An **instruction** is a command to the microprocessor to perform a given task on a specified data. Each instruction has two parts: one is task to be performed, called the **operation code** (opcode), and the second is the data to be operated on, called the **operand.** The operand (or data) can be specified in various ways. It may include 8-bit (or 16-bit) data, an internal register, a memory location, or 8-bit (or 16-bit) address. In some instructions, the operand is implicit. Instruction word size The 8085 instruction set is classified into the following three groups according to word size:

One-word or 1-byte instructions

Two-word or 2-byte instructions

Three-word or 3-byte instructions

In the 8085, "byte" and "word" are synonymous because it is an 8-bit microprocessor. However, instructions are commonly referred to in terms of bytes rather than words.

2-8- 1- One-Byte Instructions

A 1-byte instruction includes the opcode and operand in the same byte. Operand(s) are internal register and are coded into the instruction.

For example:

Task	Op code	Operand	Binary Code	Hex Code
Copy the contents of the accumulator in the register C.	MOV	C,A	0100 1111	4FH
Add the contents of register B to the contents of the accumulator.	ADD	B	1000 0000	80H
Invert (compliment) each bit in the accumulator.	CMA		0010 1111	2FH

Table 2-4- One Byte Instruction Example

These instructions are 1-byte instructions performing three different tasks. In the first instruction, both operand registers are specified. In the second instruction, the operand B is specified and the accumulator is assumed. Similarly, in the third instruction, the accumulator is assumed to be the implicit operand. These instructions are stored in 8- bit binary format in memory; each requires one memory location.

MOV rd, rs

rd <-- rs copies contents of rs into rd.

Coded as 01 ddd sss where ddd is a code for one of the 7 general registers which is the destination of the data, sss is the code of the source register.

Example: MOV A,B

Coded as 01111000 = 78H = 170 octal (octal was used extensively in instruction design of such processors).

ADD r

A <-- A + r

2-8- 2- Two-Byte Instructions

In a two-byte instruction, the first byte specifies the operation code and the second byte specifies the operand. Source operand is a data byte immediately following the opcode. For example:

Task	Opcode	Operand	Binary Code	Hex Code	
Load an 8-bit data byte in the accumulator.	MVI	A, Data	0011 1110	3E	First Byte
			DATA	Data	Second Byte

Table 2-5-Two Byte Instruction Example

Assume that the data byte is 32H. The assembly language instruction is written as

Mnemonics	Hex code
MVI A, 32H	3E 32H

The instruction would require two memory locations to store in memory.

MVI r, data

r <-- data

Example: MVI A, 30H coded as 3EH 30H as two contiguous bytes. This is an Example of immediate addressing;

ADI data

A <-- A + data

OUT port: Where port is an 8-bit device address (Port) <-- A. Since the byte is not the data but points directly to where it is located this is called direct addressing.

2-8- 3-Three-Byte Instructions

In a three-byte instruction, the first byte specifies the opcode, and the following two bytes specify the 16-bit address. Note that the second byte is the low-order address and the third byte is the high-order address.

opcode + data byte + data byte

For example:

Task	Opcode	Operand	Binary code	Hex Code	
Transfer the program sequence to the memory location 2085H.	JMP	2085H	1100 0011 1000 0101 0010 0000	C3 85 20	First byte Second Byte Third Byte

Table 2-6-Three Byte Instruction Example

This instruction would require three memory locations to store in memory.

Three byte instructions - opcode + data byte + data byte LXI rp, data16 rp is one of the pairs of registers BC, DE, HL used as 16-bit registers. The two data bytes are 16-bit data in L H order of significance.

rp <-- data16

Example:

LXI H, 0520H coded as 21H 20H 50H in three bytes. This is also immediate addressing. LDA addr A <-- (addr) Addr is a 16-bit address in L H order. Example: LDA 2134H coded as 3AH 34H 21H. This is also an example of direct addressing.

CHAPTER 3

8085 MICROPROCESSOR TRAINING SYSTEM

The Primer is a low cost 8085 based training tool developed specifically for learning the operation of today's microprocessor based systems. Microwave ovens, stereos, TVs, and almost every other electronic product utilizes embedded microprocessor technology. The Primer Trainer demonstrates the principles used by those products, providing you the opportunity to program, interface, and control virtually any device. Whether you're exploring climate control, data logging or just experimenting on your own, the Primer Trainer has the features needed to learn the necessary programming and interfacing skills required for today's world. The Primer Trainer has the ability to process analog signals (like temperature and voltage) as well as digital signals (like switches and relays). Add the capability to process those signals at precisely timed, interrupt driven intervals, and now you're talking real time embedded control. Maybe now is the time to replace those aging SDK85 trainers.

The Primer Trainer's 8085 microprocessor is an ideal platform for learning microprocessor theory. The straightforward 8085 architecture is easy to understand and the instruction set is powerful allowing the use of programming techniques similar to those used for the PC, but much simpler to learn. Since the system may be purchased as a kit, circuit assembly techniques can also be learned.

The Primer Trainer's low cost helps it fit into almost any curriculum. Its rugged construction allows it to be used by schools year after year. The Primer is affordable, allowing students to purchase it like a textbook to use and keep. For schools or individuals, the cost/performance ratio makes the Primer the most economical full-featured training system we know of.

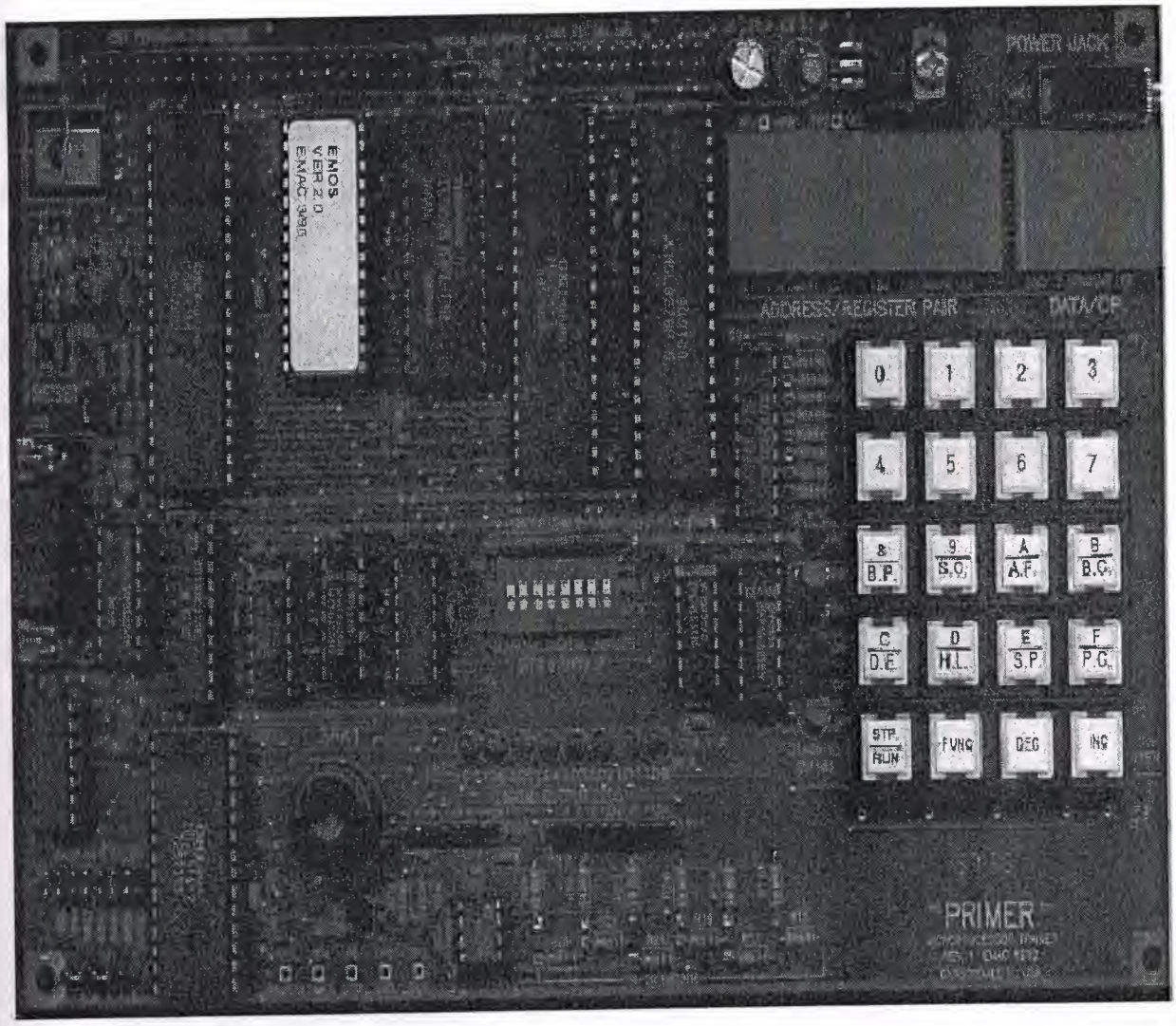


Figure 3-1- Primer Trainer

3-1- Hardware Features

Knowledge gained on the Primer applies directly to computers that are widely used in engineering and business applications. ICs used include:

- 8085 Microprocessor.
- 8155 Programmable peripheral interface with timer and RAM.
- 8279 Keypad and display controller.
- 8251 Optional UART serial controller.

20 Key, Keypad.

6 digit, 7 segment, led display.

8 position dipswitch input port with I/O connector.

8 bit output port with status leds and I/O connector.

Analog to digital converter.

Digital to analog converter.

Timer output with speaker.

14 bit timer with interrupt support.

System reset button.

50 pin bus expansion connector.

Allows interfacing of most devices.

Available as a kit or assembled.

Easy to assemble.

Rugged construction to stand up to hard use.

Ideal replacement for aging SDK85 trainers.

3-2- Programming Features

* 8K EPROM containing monitor operating system (mos) allows the user to:

- Display and edit memory.
- Display and edit registers.
- Display and edit top of stack.
- Single step by instruction.
- Run full speed with breakpoint.
- Utilize MOS internal subroutines for each I/O device as well as for multiply, divide, getkey, display number, and display ASCII.

The Monitor Operating System (the standard monitor software in EPROM) included will cause the keyboard controller chip to scan the keypad buttons, interpret the entries, and allow storage of data into the 256 byte RAM included in the PRIMER. The operating system also displays related data on the six (6) digit LED display. The operating system will allow examining the data stored at various memory locations, examining the contents of the microprocessor's registers and other functions, via the display and keypad.

The operating system software will permit to run (execute) the program entered, and to execute the program in steps, instruction by instruction, to permit debugging of programs.

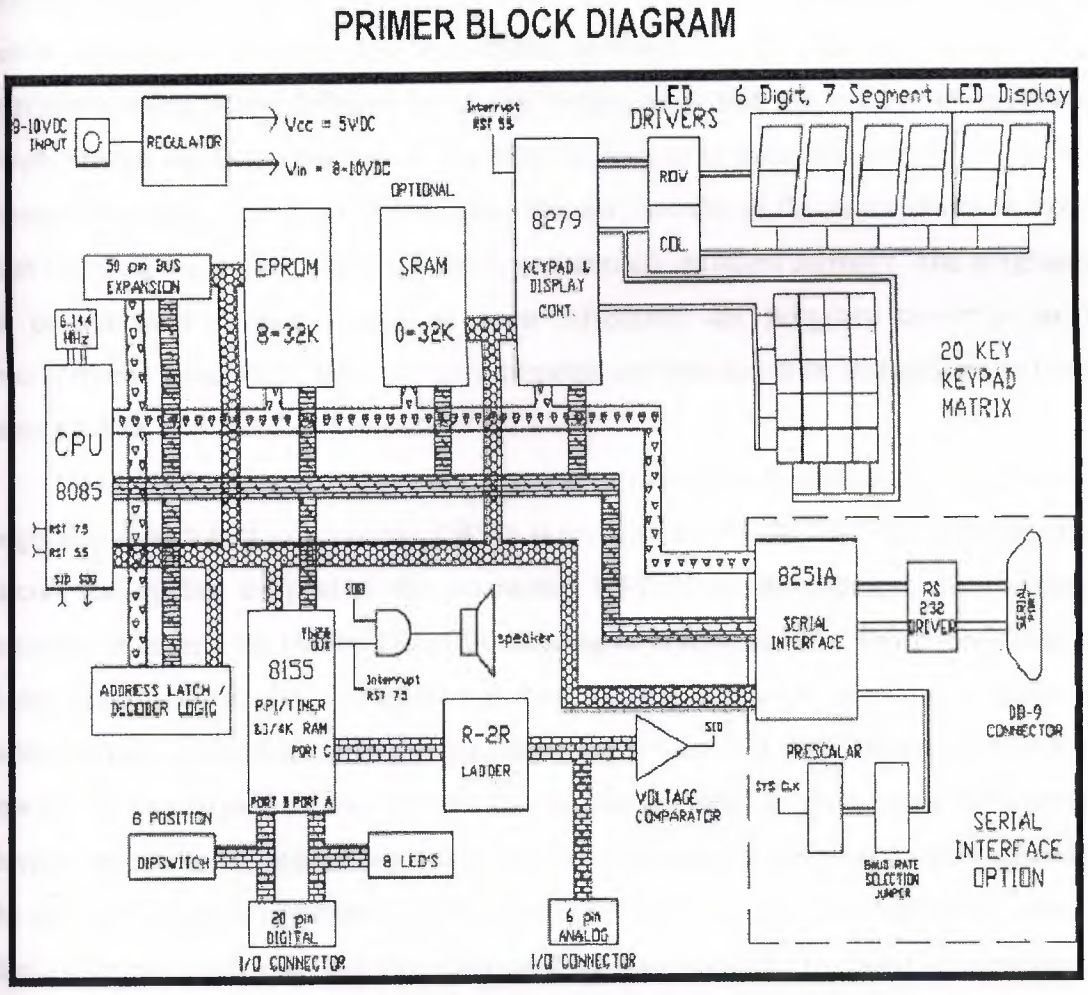


Figure 3-2- Primer Block Diagram

3-3- Microprocessor and Expansion Bus

3-3-1- The Microprocessor

The 8085 microprocessor chip is the brain of the PRIMER system. It coordinates almost all activity in the PRIMER. The 8085 microprocessor is a collection of counters, gates, registers, decoders, (etc.) that sequentially fetches instructions from memory, finds the purposes of each instruction, and executes the purpose of each instruction. The instructions are placed in memory in the order they are expected to be used. The types of instructions are widely variant, but very concise. They operations consist of moving data,

logical operations, branching and conditional branching, some mathematical operations, and input/output. When correctly assembled into a logical order, these primitive instructions form what is called a program. Programs can simply move data from an input port to an output port, using a few instructions, or they can perform complex control operations using many different input and output ports and the number of instructions would stretch out to the thousands. The 8085 is built to be able to access 65,536 possible memory locations, and 256 I/O locations. The architecture of the microprocessor expects both the program and the data to reside in sequentially arranged memory. The programmer (a person who creates programs) must assemble the program correctly so the microprocessor will know what memory contents are intended to be instructions and which ones are data.

The 8085 and most other microprocessors must start executing the program at its beginning, which in the case of the 8085 is at memory address 0. This is accomplished by a circuit that applies a signal to the pin named RST-IN on the microprocessor when the computer is turned on. On the PRIMER, this signal comes from the Power-on/Pushbutton Reset circuit . When power is applied to the PRIMER, capacitor C1, initially discharged, holds the microprocessor's RST-IN* pin, low (the "*" in "RST-IN*" means signal is active low). The low signal on the RST-IN pin causes the 8085 to reset some of its internal devices and make an internal register called the program counter point to the beginning of the program. Another event that occurs is that RST-OUT signal is asserted, which resets the display/keypad controller and PPI chip used in the PRIMER. If EMAC peripherals are connected to the expansion port they are also affected by the RST-OUT signal. This reset condition can also be brought about by pressing the reset button (PB1) which merely shorts out C1 and brings the RST-IN pin low. As the capacitor charges through R1, the RST-IN* line voltage rises until a logic high de-asserts this signal. Diode D1 serves to quickly discharge capacitor C1 through the power supply only when power is off. With RST-IN* de-asserted, the 8085 begins operation.

The system clock oscillator, whose frequency is set to 6.144 MHZ by crystal Y1, drives all timing functions within the 8085 and it runs as long as power is applied to the PRIMER. It is possible to run the 8085 under a wide range of frequencies, but in the PRIMER use 6.144 MHZ to provide compatibility with circuits used elsewhere in the

system. The oscillator signal is divided by two within the 8085, and this signal is now the main system clock, referred to as SYSCLK. All timing of the microprocessor's operation is driven by this SYSCLK, which in our case is 3.072 MHZ. Even asynchronous input signals like RST-IN*, and interrupts are internally synchronized to this oscillator. Each pulse of this clock signal, is called a "T state", when referring to operations internal to the 8085 and its busses.

The first thing the microprocessor will do after RST-IN* de-asserts, is to fetch an instruction at memory address 0. Before the instruction can be fetched, the memory address of the instruction must be output from the microprocessor. To reduce the number of pins on the 8085's package, a scheme called multiplexing is employed. Multiplexing allows the AD0-AD7 pins to be used as the data bus and also to output the lower 8 bits of the 16 bit address needed to select the first instruction. These pins must be demultiplexed before they can be used properly.

The 8085 outputs the low byte of the address on signal lines AD0-AD7, and the high byte directly on signal lines A8-A15. A signal called Address Latch Enable (ALE) strobes high, then low, causing the address latch (chip U8) to trap the address byte that is on pins AD0-AD7. The output of this latch then joins up with address lines A8-A15, to provide a 16 bit, filtered, stable, linear address to the memory chips and decoders .

The 16 bit linear address is applied to the memory block (memories and decoders) to select the address that the 8085 wants to read from or write to. In the case of an instruction fetch, the 8085 signal RD* will be asserted, to read the byte from the memory selected. This byte will then go into the instruction decoder, which will determine what instruction it is, and then the 8085 will execute the instruction. Depending on what type of instruction it is, the 8085 will either fetch additional data or read/write data to other memory locations or perform I/O, (etc.). Each time a bus transaction occurs these operations will occur: the low byte of the address will be sent to AD0-AD7, which will then be held by the address latch. A complete cycle of instruction fetch, decode, and execution, forms a "bus cycle", using anywhere from 3 to 14 T-States, (SYSCLK pulses) depending on the type of instruction.

The 8085 has five pins dedicated to "interrupting" the microprocessor. These pins are named TRAP, RST 7.5, RST 6.5, RST 5.5 and INTR. When a signal is applied to one

of these pins an "interrupt" is generated. An interrupt is a special function of the 8085 to suspend the program it is running then execute a program related to the interrupt that requested it. Once the interrupt is completed, the 8085 must RESTORE the original program's data and status, and RETURN to it where it left off from. Sometimes the interrupt is deliberately ignored by MASKING the interrupt, so interrupts can be turned on or off as needed, should a section of code be so important to require that it not be interrupted until it is finished.

An interrupt may be likened to answering the telephone while reading a book. When the telephone rings, you put the book down on it's face to mark your place, then you answer the phone. When you are done, you pick up the book and begin reading where you left off. Should you be too engrossed in the story, you can ignore the ringing telephone (you can mask it). But if you ignore it too long, you'll miss the call. Sometimes interrupts can be missed as well. Some interrupts can be made to latch, usually by hardware. These latched interrupt requests will persist, just as a person on the phone might continue ringing (interrupting) until you answer it.

3-3-2- Bus Expansion Port

The Primer's Bus Expansion PORT is actually a simple connector with leads running to power sources and microprocessor pins. The 8085's AD0-AD7 lines run out to the connector, as well as address lines A0-A15, control signal IO/M* (which the 8085 asserts to tell whether it is doing a memory or I/O transaction), RD*, WR*, RST-OUT, SYSCLK, GROUND, and PWR (this is directly from the power jack, 5V is not provided on the bus connector). Also included are INTRQ, and INTA*, which allow the use of a multi-level interrupt controller; S0, and S1, which are part of the 8085 machine status output; RDY/WT* which is used to put the 8085 into a wait state; HOLDRQ and HLDACK which are used to allow another bus master to assume control of the bus; and finally, ID* and EXTIOCS*, which are used by external I/O devices of the EPAC 2000 family. The Bus Expansion Port footprint permits direct connection of the PRIMER to most EPAC peripheral cards, via a 50 pin ribbon cable. The type of module port (shape of the module) sometimes gives helpful information about the kind of connections made. For example, module port D[0..7] is a bi-directional port, and all occurrences of this port on

other pages denote this is a bi-directional 8 line bus interconnection. Individual lines to module ports also connect these lines to corresponding points on other sheets of the schematic, such as lines RD*, and WR*.

3-4- I/O Decoding and Memory

The decoder section of a microprocessor system sorts out the address bus signals from the 8085, and picks out (selects) the correct chip(s) in the memory or I/O sections that the 8085 wants to access. The decoder must evaluate the address applied to it, and determine whether the address is for a memory access, or an I/O map access. Most decoders are custom hardware collections of gates, using various levels of integration. As every system has a different configuration of memory and I/O, every decoder is unique to that design.

In the PRIMER, two types of memory maps may be decoded. In the standard map, a 16K (K=1024 bytes) byte space is reserved for program EPROM, starting at address 0000 hex ending at 3FFF hex. (Remember the 8085 begins execution at 0000 hex after reset) This socket may hold 8K, 16K or 32K EPROM chips. The monitor operating system program is actually very small, requiring less than 4K, but 8K EPROM chips are the smallest memory chip of compatible footprint that will fit there.

Almost all programs (the Primer's monitor is no exception) will require some memory that can be *written to* as well as read. The PRIMER provides this type of memory, without additional discrete RAM chips. The 8155 PPI/MEMORY/TIMER I.C. is a multi-function chip, with digital I/O ports, a timer/counter, and 256 bytes of RAM in it. This memory is used for storage of variables, pointers, executable code and various other functions. Because the 8155 has both I/O and memory mapped elements, the decoder section provides an additional specially encoded line called 8155CS for this device.

The optional 32 K byte memory socket allows a 32 K static RAM chip to be installed onto the PRIMER. Option selection jumpers, OJ2 and OJ3 resolve the hardware differences between the 8 K, 16 K and 32 K EPROM chips, and allow the addition of the 32 K byte memory and by rearranging the memory map to accommodate the different chips. This is necessary when software options such as E-FORTH and MTBASIC are

installed. The memory can be altered to use the two memory maps described. In the case of the EPROM chip, an 8 K device (2764) or a 16 K device (27128) may be used interchangeably, bearing in mind the smaller sized devices do not fill the entire memory map. When the alternate memory map is used, a 32 K device, (27256) may be installed in the EPROM socket. The difference of the pinout in the 32 K device is the reason for the option jumpers. Jumper OJ2 switches pin 27 between A14 or Vcc, as required by the 32 K device. The 8155 PPI's memory space begins at C000 hex and ends at the top of the 8085's address range at FFFF hex. Note that this range is 16 K, but the actual memory is only 256 bytes. What happens is the same 256 bytes repeat themselves over and over again, in this case 64 times over. In other words, hexadecimal addresses C000, C100, F300 and FD00 all refer to the first memory address within the 8155 and C0FF, C1FF, F3FF and FDFF all refer to the last memory address within the 8155. Programs that are written must use only the amount of memory that is available.

The alternate memory map reserves 32 K of EPROM at addresses 0000 hex to 7FFF hex, and 32 K static Ram from 8000 hex to FFFF hex. The 256 bytes of RAM in the 8155 PPI are ignored, so in this map, the entire 64 K of memory that the 8085 is capable of accessing does not contain memory cells with duplicate addresses.

The PRIMER uses a very simple decoder. Two, "2 to 4 line" decoders decode the appropriate 8085 address lines to select the proper chips. The memory decoder observes the code on address lines A14 and A15. The state of these lines represents 16 K intervals of address. To change the decoder interval to 32 K bytes boundaries, the decoder pin that is connected to A14 can instead be connected to ground. The I/O decoder observes the state of address lines A6 and A7. The I/O map is only 256 addresses wide. The I/O decoder is maximally used at select rates of 64 addresses per output.

The 8085 signal line IO/M* tells the decoders that the address is an I/O MAP transaction, if it is high, or that it is a memory map transaction, if it is low. A simple inverter changes the polarity of IO/M* to IO*/M. Since each decoder has an active low enable input, connection of the appropriate polarity signal of IO/M* to the appropriate decoder, will enable the right decoder for the job at hand. Also all decoder outputs are active low signals, and will be de-asserted unless the right address in the right map selects it.

The EPROM chip select will be valid for memory addresses 0000 hex to 3FFF hex, (or to 7FFF hex if non-standard EPROM) and simply goes directly to the EPROM's chip enable. The optional static RAM chip select must be valid for addresses 4000 hex to BFFF hex, (or 8000 to FFFF hex if nonstandard) which is an interval of 32 K. As the memory decoder outputs at intervals of 16 K, (standard map) two chip selects are logically negative OR'ed by AND gate U10A. This produces a single 32 K wide chip select for this memory.

The 8155 chip has elements of both the memory map and the I/O map, so a similar AND gate logically negative OR's the memory and I/O map selects into a single dual mapped chip select for the 8155. The memory portion of this chip select is valid throughout C000 hex to FFFF hex. The I/O portion is I/O addresses 00 hex to 3F hex.

The 8279 keypad and display controller is I/O mapped from 40 hex to 7F hex, and the optional 8251A communication controller from I/O addresses 80 hex to BF hex. A special I/O chip select, used primarily for EPAC expansion peripherals, via the bus expansion connector, uses the remaining I/O space of C0 hex to FF hex. As in the memory chip select description, many of the addresses repeat in the I/O map, since these chips only uses a few of the available addresses.

The rest of the address bus lines not used by the chip select decoder described, go directly to the memory chip(s) other I/O chips, and bus expansion connector. Inside these chips are decoders which select the precise internal memory cells or device registers denoted by the address pins. In more complex systems, the chip select decoders can be more precise, but this increases the complexity and cost of the system. The simple device select decoder used here in the PRIMER will serve quite well, as long as programs are written carefully.

3-5- Display and Keypad Circuitry

The PRIMER has a key pad matrix of pushbuttons, for entering commands, loading data or programs, and testing programs. It also has a 6 digit LED matrix display, to view data. The 8279 is a specialized controller chip, which removes the chores of keyboard scan, switch de-bounces, and display refresh, from the microprocessor. The 8279 contains its own memory which stores key press data and data to be displayed. Through the I/O map, the 8085 sends commands to the 8279 that configure the operational characteristics of the

display/keypad system, then writes the display characters themselves to the 8279's internal display RAM. The 8279 scans this display RAM, and drives the outputs to the display LED's continuously thereafter. No further microprocessor action need be taken until a new display pattern is needed.

The 8279 has 8 data bus lines that connect directly to the 8085's AD0 to AD7 lines. These bi-directional bus leads transfer data and control information between the 8085 and the 8279 Display/Keypad controller. Signals RD*, and WR* determine whether data is read from or written to the 8279. Address line A0 is used by the 8279 to determine whether the data byte is a command byte, or a data byte for the internal RAM and the 8279's chip select pin determines whether it will perform bus transactions or not. Because address line A0 is used to select the 8279 function, two I/O map addresses exist for the Display/Keypad controller. A program that operates the controller merely writes to or reads from two consecutive addresses to control all 8279 functions. With A0 being low, the 8279 memory can receive data from, or send data to the data bus. If A0 is high, the bus transaction is a command or status byte. The base address of the 8279 in the PRIMER is 40 hex. The I/O decoder's chip select will be valid for all I/O map transactions up to 7F hex. Therefore, the two addresses for the 8279 repeat themselves throughout this range, just as the memory addresses repeat in the 8155. This repetition is common to all I/O devices on the PRIMER, due to the broad nature of the chip select decoder used.

The signals RST-OUT, SYSCLK, and KEYINT also terminate at the 8279. RST-IN will hardware clear the 8279's internal control registers to their default state, which at a later time can be changed to other states by the Primer's program in the EPROM. However, the data in the internal RAM is not cleared by hardware. The SYSCLK signal (the microprocessor's 3.072 MHz master clock signal), is divided internally by the 8279 to provide the keypad scan and LED display refresh clocks. Finally, the KEYINT signal is a special control line that is set active high by the 8279 when a valid key entry is made. This line goes through a jumper that (in its default connection) connects to the RST 5.5 interrupt pin on the 8085 microprocessor.

The 8279 scans the keypad buttons and debounces the key inputs to avoid erroneous entries due to contact bounce or key rollover. After this, it stores the key inputs in the internal keypad RAM. When a valid key entry is made, the 8279 generates an interrupt

display/keypad system, then writes the display characters themselves to the 8279's internal display RAM. The 8279 scans this display RAM, and drives the outputs to the display LED's continuously thereafter. No further microprocessor action need be taken until a new display pattern is needed.

The 8279 has 8 data bus lines that connect directly to the 8085's AD0 to AD7 lines. These bi-directional bus leads transfer data and control information between the 8085 and the 8279 Display/Keypad controller. Signals RD*, and WR* determine whether data is read from or written to the 8279. Address line A0 is used by the 8279 to determine whether the data byte is a command byte, or a data byte for the internal RAM and the 8279's chip select pin determines whether it will perform bus transactions or not. Because address line A0 is used to select the 8279 function, two I/O map addresses exist for the Display/Keypad controller. A program that operates the controller merely writes to or reads from two consecutive addresses to control all 8279 functions. With A0 being low, the 8279 memory can receive data from, or send data to the data bus. If A0 is high, the bus transaction is a command or status byte. The base address of the 8279 in the PRIMER is 40 hex. The I/O decoder's chip select will be valid for all I/O map transactions up to 7F hex. Therefore, the two addresses for the 8279 repeat themselves throughout this range, just as the memory addresses repeat in the 8155. This repetition is common to all I/O devices on the PRIMER, due to the broad nature of the chip select decoder used.

The signals RST-OUT, SYSCLK, and KEYINT also terminate at the 8279. RST-IN will hardware clear the 8279's internal control registers to their default state, which at a later time can be changed to other states by the Primer's program in the EPROM. However, the data in the internal RAM is not cleared by hardware. The SYSCLK signal (the microprocessor's 3.072 MHz master clock signal), is divided internally by the 8279 to provide the keypad scan and LED display refresh clocks. Finally, the KEYINT signal is a special control line that is set active high by the 8279 when a valid key entry is made. This line goes through a jumper that (in its default connection) connects to the RST 5.5 interrupt pin on the 8085 microprocessor.

The 8279 scans the keypad buttons and debounces the key inputs to avoid erroneous entries due to contact bounce or key rollover. After this, it stores the key inputs in the internal keypad RAM. When a valid key entry is made, the 8279 generates an interrupt

request to the 8085 via the KEYINT line, and holds the key data in the keypad RAM until the 8085 actually reads the key memory. Should the microprocessor not respond immediately, the 8279 will store in the internal key RAM up to 8 key entries. When the PRIMER is turned on, the 8279 is configured by the PRIMER software to produce what is referred to as an "encoded scan" on its SL0-SL3 lines. This encoded scan is decoded by a "3 to 8" decoder, to produce 1 of 8 active low row scan outputs. Six of these outputs go to upside down biased PNP transistor followers, to provide high current sinks for the common cathode LED's, since the 74HC138 (3 to 8 decoder) can only sink a few milliamps. The scan outputs from the scan decoder also go to the keypad buttons. When a key is pressed, a low signal will be presented to one of these corresponding return lines: RL0-RL3. These return lines are already connected to pull up resistors that are conveniently built into the 8279. The key must be closed and be scanned several times in order to be recognized by the 8279. It is then entered into key RAM and an interrupt request is generated which tells the microprocessor to read the 8279's keypad RAM and interpret the key code to perform the function that is dedicated to the key. The key must be released and stay that way for several more scans before that key or another key will be accepted. This is the key pad debounce feature. The 8279 will also ignore additional key closures while if another key is pressed.

3-6- Digital I/O, Timer and Speaker

The 8155 combination Digital I/O, Timer, and Memory Chip provides the PRIMER with capability to experiment with Digital input and output programmable timing, and provides the read/write memory function for the system. Its bus interface is very similar to the interface for the 8279 Display/Keypad controller. The 8155 also has 8 data bus lines, RD*, and WR* control signals, and RESET input. There are additional bus control lines, but no individual address inputs are required since the lower byte of the address and the data are demultiplexed within the 8155 from the AD0-AD7 inputs. The 8155 is missing the address line(s) the 8279 has, instead, it uses two new signals, IO/M* and ALE. These signal lines are the same as described in the previous microprocessor description. The 8155 chip is a special function chip made specifically to work along with the 8085. It has an internal address latch, and an internal address decoder, which operates in parallel with the Primer's address latch. The internal address latch and decoder uses the 8085 ALE signal, to trap the

low order 8 bits of multiplexed address. Since the internal RAM is only 256 bytes and there are only 4 I/O addresses, only the low 8 multiplexed address/data bits are needed anyway. The chip select signal for the 8155 resolves whether bus transactions will occur to the address internally latched and selected during microprocessor read or write cycles, to either the I/O or memory sections of the 8155.

The Digital I/O ports are two "byte wide" (8 bits) ports, and one six bit port. The ports can be independently set to be OUTPUT or INPUT ports.

On the PRIMER unit, Port A is set by monitor software to function as an output port and eight LEDs serve as status indicators of the bit state for each output line. Each LED lights when the bit for it is set LOW (0), and it is off when the bit is set HIGH (1). Therefore, negative logic levels are assumed.

Port B of the 8155 is set by software to input. An eight station DIP switch is used to set each bit LOW (0) when each station is turned on. Negative logic is also assumed here. When a switch station is switched OFF, a resistor pullup brings the line HIGH. Port C of the 8155 goes to the digital to analog converter which is described later. The 8155 chip is totally flexible as to what kind and data and direction is present on each port, however, for hardware purposes on the PRIMER, the default configuration should be maintained.

The 8155 has a programmable timer/counter in it that is software programmable to count from 1 to 16,383. The counter becomes a timer if it is fed a regular pulse train, as it is in the PRIMER. The SYSCLK signal is first divided by ten, and fed to the 8155 timer input. The timer's output goes to the 8085's RST 7.5 interrupt input, and to an AND gate that drives a piezo-electric speaker element.

The piezo-electric speaker has a limited frequency range, but many useful sounds can still be produced by appropriate setting of the timer's divide ratio, and turning the output on and off through the 8085's SOD signal. The SYSCLK frequency of 3.072 MHZ is divided down by U12 to 307.2 KHZ, then fed to the timer input of the 8155. By setting the timer's divide ratio from 2 to 16383 (in software), the range of frequencies can be from 153.6 KHZ down to 18.75 Hz.

3-7- Analog I/O and Power Supply

Many Microprocessor systems read analog signals, such as temperatures, pressures, and amperage. Because microprocessors are digital devices, some sort of interface is required to change an analog voltage into a digital code the microprocessor can deal with. In many cases, the microprocessor system will also output a digital signal, after processing the analog and digital inputs it has reviewed. The PRIMER has a 1 channel analog output, and a 1 channel analog input with the range of both at approximately 0 to 5 volts. The digital to analog (D/A) output comes from a R-2R ladder network of resistors, which is then buffered by op-amp U14A. The D/A ladder is driven by port C of the 8155. The D/A converter has 6 bit resolution, so programs may be written which write a value in the range of 0-63 decimal to the D/A port, resulting in an analog voltage proportional to the number written to the port.

The analog to digital (A/D) input is a voltage comparator, made up of U14B, and some digital level shifting circuitry. The comparator has an input pin on connector CN3, where the voltage to be measured is applied. The comparator then compares this input voltage to the D/A ladder output voltage, and sends a digital signal to the SID input of the 8085. This digital signal is high (or 1) when the D/A voltage is higher than the input voltage and the digital signal is low (or 0) when the D/A voltage is lower than the input voltage. The Primer's monitor operating system contains a built in program which converts an input voltage to a decimal value in the range of 0-63. Using this built in program within other programs and attaching the appropriate hardware will allow the PRIMER to function as a voltmeter, temperature gauge, pressure gauge or any other kind of analog measurement device. The PRIMER can then be used as what is often referred to as an "embedded controller". The final area of the PRIMER schematics is the power supply section. In the PRIMER only a very simple power supply is required. The D.C. input is fed from power jack J1, and is pre-filtered by C8, then applied to an LM7805 regulator. The LM7805 regulator is an industry standard device found in almost all small microprocessor systems, because its application is very simple. The LM7805 regulator steps the 7 to 10 VDC power supply input down to the +5 volts required by the microprocessor and all the other digital logic chips. Some of the input power bypasses the LM7805, and goes directly to the LM358 dual op-amp chip, U14. The LM358 needs " voltage overhead " to work correctly

for the analog voltages it operates with. Capacitor C9, and seven other capacitors spread about the board provide D.C. power supply bypass. These bypass capacitors stabilize the power supply voltage fed to the logic devices on the PRIMER.

3-8- Hardware Reference

3-8-1-Hardware Reset

The PRIMER board can be reset through the reset button provided on the board. There is a reset output pin on the expansion connector which allows the resetting of the PRIMER to reset any devices that may be connected to the expansion connector.

3-8-2- Serial Communication PORT

The PRIMER uses an RS232 standard serial communication port. The port interfaces to a PC or terminal through a DB-9 shell connector (communication cables are available as an accessory from EMAC). The serial communication rate (or baud rate) must be set to the rate used by the terminal or PC. Placement of jumper JP1 can set the baud rate from 300 baud up to 19,200 baud. MOS services are available which access the 8251 serial communication port

3-8-3- Dip Switch

The DIP switch has 8 switches which may be used for applications such as selection of program options. The DIP switch is connected to the system data bus and is accessed through I/O, address 012H. A MOS service is available which accesses the DIP switch.

3-8-4- Digital Outputs

The PRIMER has 8 outputs and each output can be independently programmed to an ON (+5v or binary 1) or OFF (0v or binary 0) state. The outputs are connected directly to the digital output LEDs and an LED can be turned on by the output of a binary 0, and turned off by the output of a binary 1. The outputs are also connected to the digital I/O connector CN3.

The output is driven by the 8155 I/O I.C. which, in the standard configuration, uses PORT A (11H) as the output port, PORT B (1 2H) as the input port and PORT C (13H) as an analog output port.

The standard configuration of the 8155 which is set up by MOS should not be changed. Also, since PORT A outputs have limited drive capabilities, buffering should be considered if these outputs are to be used.

3-8-5- Digital Inputs

The PRIMER allows 8 inputs through 8155 port B. These inputs are connected directly to the 8 station DIP switch. The inputs are also connected to digital I/O connector CN3, so if the DIP switches are all turned off, you may connect external TTL level input devices through this connector.

3-9- Timer/Counter

The PRIMER comes equipped with a 14 bit timer/event counter. This timer/counter is resident in the 8155 I/O I.C. By loading a user programmable termination count, time intervals from 3.25 microseconds to 53.3 milliseconds. When the termination count is reached, a RST 7.5 interrupt can then be issued to the CPU. If interrupts are not desirable the timer can be read directly or the interrupt line can be polled. The timer can also be set up to reload itself or to stop counting upon reaching the termination count. In either case, an interrupt can be issued. The timer/counter is a 14 bit down counter that counts the 'timer input' pulses and provides a pulse or square wave to the 8085 RST 7.5 interrupt when the terminal pulse is reached. The user can reprogram the length of the count before the termination pulse is reached if so desired. The user can also determine the timer interval by programming the counter register from values 2H to 3FFFH.

The timer/counter has four operating modes which are:

Mode 0 No operation mode (NOP) does not affect the timer.

Mode 1 Stop mode stops the timer/counter if it is running otherwise NOP.

Mode 2 Stops the timer if running immediately after the terminal count has been reached otherwise NOP.

Mode 3 The start mode loads the output mode and count length and starts the timer/counter immediately if timer is not running, otherwise it waits
for the terminal count then starts the timer/counter.

The timer/counter operating modes are programmed through the 8155 control register (I/O address 10H). The lower 8 bits of the count length is written to I/O address 14H. The upper 6 bits of the count length along with the 2 bit output mode is written to I/O address 15H. The timer/counter can be used as an external program interval timer. If you wish to perform a software operation at a specific time interval, then the timer/counter can be programmed to that interval. Upon the resulting timer interrupt your program can execute the desired software.

The timer/counter is also used to drive the Primer's speaker. Different frequencies can be output from the speaker using a MOS service.

The PRIMER has a 40 pin expansion connector (CN1) on board which provides additional expansion capabilities. Primarily, this port gives access to the Data and Low Address Busses and Control Lines. See Appendix D "EXPANSION CONNECTOR CN1" drawing for a detailed description.

3-10-The PRIMER Keypad Description

The PRIMER Trainer has twenty keys for input, with some keys having a second function which can be invoked using the "FUNC." key. Each key produces an audible tone when pressed, providing feedback for the user.

KEY DESCRIPTION

0-F : Numeric keys when a numeric key (0-F HEX) is pressed, the numeric value appears at the right side of a data field display and the number that was there before will be shifted left. To correct an entry error just repeat, using the correct number key and the error will be overwritten.

Step : This key causes the microprocessor to execute one instruction (single step) at the current program counter address (the value of the P.C. register). Single stepping is a

valuable debugging tool that allows the user to examine registers and memory after each instruction executes. The Step key executes the instruction at the program counter address and then returns to the Monitor Operating System and waits for another user command. When you single step a CALL instruction that is calling a subroutine in ROM (such as a service call), the processor will execute the subroutine at full speed and stop at the instruction immediately following the CALL instruction.

Dec: This key decrements the program counter and shows the program counter in the left four displays and the data at the program counter address in the right two displays.

Enter & Inc: When the monitor is in the data entry mode, pressing this key stores the data that is on the two far right displays into the program counter address then increments the program counter and displays the new program counter address in the left four displays and the data at that address in the two far right displays. This key is also used to store new values for the registers, the
breakpoint and the stack content (see the section below about second function keys). If the register, breakpoint or stack content information displayed has been changed but you don't want to enter it, then simply press the "Func." key twice and the original information is maintained. This key can also be used to just view data in memory.

Func: This selects the second function of the keys that have two functions. When this key is pressed "Func." appears on the display, and if a key is pressed which has a second function, that function will be performed. This key may be pressed two times in a row to exit the current mode and return to the data entry mode.
(Below are the second functions of the keys with two functions)

B.P: This key displays the current software breakpoint address. If 0000 is displayed, then no breakpoint has been established. The displayed value may be changed by entering the desired breakpoint address in hex using the numeric keys and then pressing the "Enter & Inc." key. When the program reaches the breakpoint the software breakpoint address is automatically reset to

0000. NOTE: Obviously, if the program execution never reaches the breakpoint address, the program will not stop at the breakpoint address. If the breakpoint address points to a byte other than the op code in a two or three byte instruction, the program will not stop at the breakpoint address. Also any address specified that references addresses in EPROM (addresses below 8000 hex) will not stop execution, even if the op code at that address is executed.

S.C : This displays the 16 bits that are at the top of the stack, or in other words, the data that will be removed from the stack with the next POP, RET or XTHL instruction. The two displays on the far left represent the data at SP + 1 (stack pointer address + 1) and next pair to the right represent the data at SP. The displayed value may be changed by entering the desired hex number using the numeric keys and then pressing the "Enter & Inc." key.

A.F: This key displays the contents of the A register (Accumulator) and the condition Flags. The A register and the flags are displayed as four hex digits in the four displays starting at the left. The digit pair on the left of the four displays show the value of the A register and the pair on the right show the value of the condition Flags. The displayed value may be changed by entering the desired hex number using the numeric keys and then pressing the "Enter & Inc." key. The condition Flags are defined bit by bit as follows:

B-2 BIT 0 CARRY FLAG (if this bit is 1 it indicates a carry)

BIT 1 not used

BIT 2 PARITY FLAG (if this bit is 1 it indicates an even number of bits, odd parity)

BIT 3 not used

BIT 4 AUX. CARRY FLAG (if this bit is 1 it indicates a carry from bit 3 to bit 4 in the A register)

BIT 5 not used

BIT 6 ZERO FLAG (if this bit is 1 it indicates a zero result)

BIT 7 SIGN FLAG (if this bit is 1 it indicates bit 7 of the A register is a 1)

For example, if the display reads: 0044 A.F.

This indicates the contents of the A register is 0 and the ZERO and PARITY Flags are both set.

B.C : This key displays the contents of the BC register pair. The BC register pair is displayed as four hex digits. Starting from the left display, the first pair of displays

represents the contents of the B register and the second pair represents the contents of the C register. The displayed value may be changed by entering the desired hex number using the numeric keys and then pressing the "Enter & Inc." key.

D.E :This key displays the contents of the DE register pair. The DE register pair is displayed as four hex digits. Starting from the left display, the first pair of displays represents the contents of the D register and the second pair represent the contents of the E register. The displayed value may be changed by entering the desired hex number using the numeric keys and then pressing the "Enter & Inc." key.

H.L :This key displays the contents of the HL register pair. The HL register pair is displayed as four hex digits. Starting from the left display, the first pair of displays represent the contents of the H register and the second pair represents the contents of the L register. The displayed value may be changed by entering the desired hex number using the numeric keys and then pressing the "Enter & Inc." key.

S.P: This key displays the contents of the stack pointer. The stack pointer is a 16 bit register, displayed as four hex digits. The displayed value may be changed by entering the desired hex number using the numeric keys and then pressing the "Enter & Inc." key.

P.C : This key displays the contents of the program counter. The program counter is a 16 bit register, displayed as four hex digits. The displayed value may be changed by entering the desired hex number using the numeric keys and then pressing the "Enter & Inc." key.

Run :This key causes the microprocessor to execute a program at full speed starting at the current program counter (PC) address.

The program will continue to execute until an optional software breakpoint is encountered or until a RST 7 (FF hex) instruction is executed.

1 This key invokes the PRIMER diagnostics.

2 This executes the MOS service selected by the value in the C register, only affecting the registers that pass data back from the service. No instructions need to be

stored in memory and the PC register is not affected. For more details see "Using MOS Services".

3 This enables a PRIMER with a Deluxe or Standard Upgrade Option to receive an Intel hex file through its serial port.

4 This invokes the EPROM programmer menu driven interface. This requires the optional EPROM PROGRAMMER BOARD, the standard or deluxe upgrade option and a connection to a dumb terminal or terminal emulation package running on a PC.

3-11- MOS Services

Service B Keyin

Read the keypad; this service waits for a key to be pressed and returns the value in the L register.

INPUT REGISTER C: 0B

OUTPUT REGISTER L: Key value. Keys "0-F" return 00-0F respectively and the "Step", "Func.", "Dec.", and "Enter" keys return 14-17 respectively.

Service 10 Pitch

Pitch output; this service sends the 14 bit count (the upper two bits are ignored) in the DE register to the speaker timer. The larger the number the lower the pitch. If the DE register pair = 0, then the speaker tone
is turned off.

INPUT REGISTER C: 10

REGISTER PAIR DE: 14 bit pitch value.

OUTPUT NONE

Service 14 Delay

Delay according to the value of the HL register pair. The larger the value, the longer the delay

INPUT REGISTER C: 14

REGISTER PAIR HL: Amount of delay.

OUTPUT NONE

CHAPTER 4

Note frequency (hertz)

In music, a **note** is either a unit of fixed pitch that has been given a name, or the graphic representation of that pitch in a notation system (and sometimes its duration) or a specific instance of either, so one can speak of "the second note of Happy Birthday to You" for example. The general and specific meanings are freely mixed by musicians, although they can be initially confusing: "the first two notes of Happy Birthday to You are the same note", meaning, "the first two sounds of Happy Birthday to You have the same pitch." A note is a discretization of musical or sound phenomena and thus facilitates musical analysis

In all technicality, music can be composed of notes at any arbitrary frequency. Since the physical causes of music are vibrations of mechanical systems, they are often measured in hertz (Hz), with 1 Hz = 1 complete vibration per second. For historical and other reasons especially in Western music, only twelve notes of fixed frequencies are used. These fixed frequencies are mathematically related to each other, and are defined around the central note, A4. The current "standard pitch" or "concert pitch" for this note is 440 Hz. Actual practice may vary. In the past there has been a rising tendency.

Name	prime		second		third	fourth		fifth		sixth		seventh
Natural	C		D		E	F		G		A		B
Sharp (symbol)		C♯		D♯			F♯		G♯		A♯	
Flat (symbol)		D♭		E♭			G♭		A♭		B♭	
Sharp (text)		Cis		Dis			Fis		Gis		Ais	
Flat (text)		Des		Es			Ges		As		Bes	
French/Italian (only in C major)	Do		Re		Mi	Fa		Sol		La		Si
Variants	Ut		-		-	-		So		-		Ti
German	C		D		E	F		G		A	B	H
Frequency [Hz]	262	277	294	311	330	349	370	392	415	440	466	494

Figure 4-1- Note frequency Table

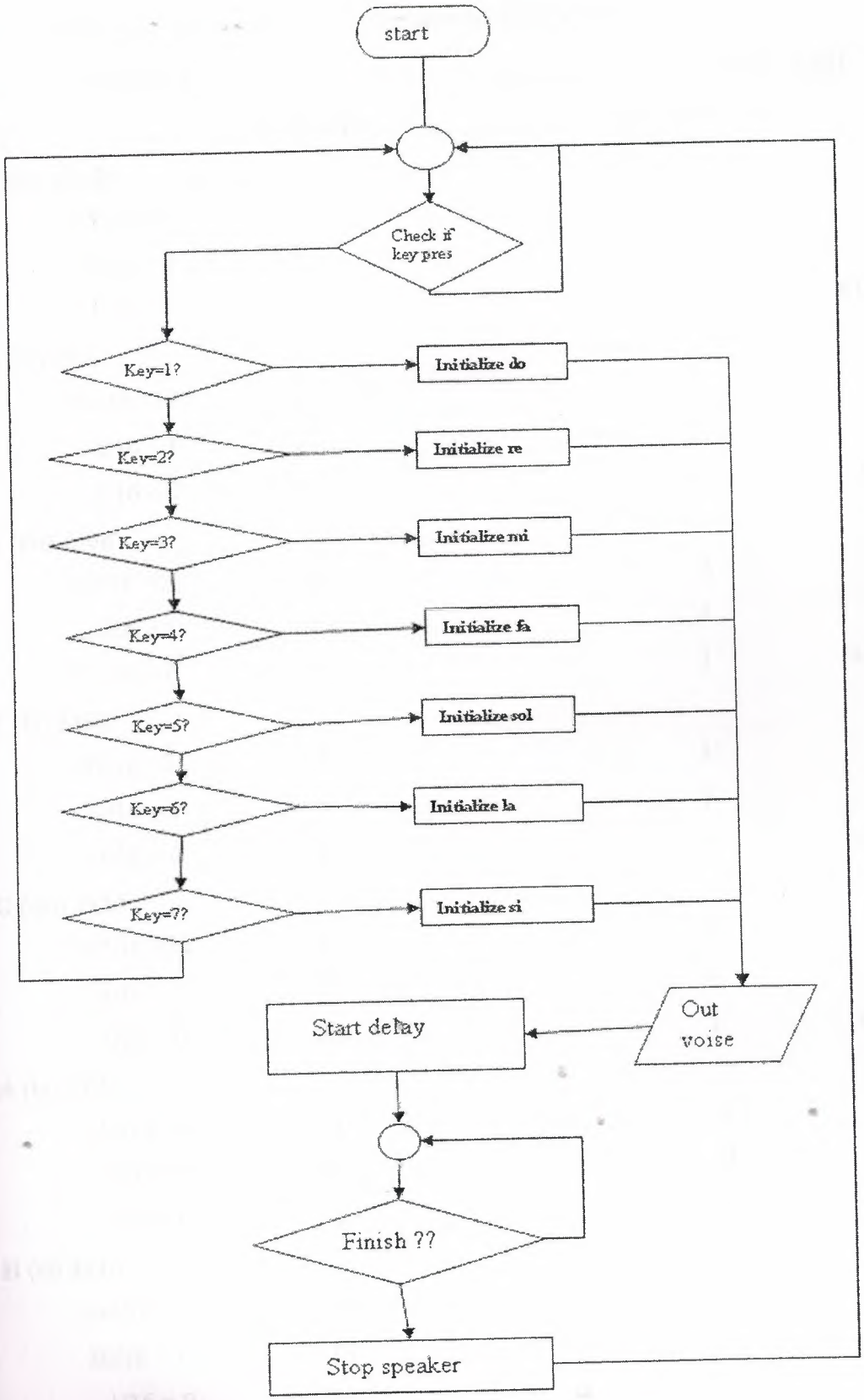
CHAPTER 5

Electronic Organ Design Using 8085 Microprocessor

The program read the key value from keypad and generates sound according the key pressed. In the program used 7 key (1 to 7) and 0 octave 7 note between 262 to 494 hertz. when the key pressed the note frequency load in to register pair according the key and generating sound After generating sound in a specific time stop the speaker and wait to pressing key again.

To use a 8085 microcontrollers and assembly language; some requirements such as a speaker and buttons are investigated. I used a PRIMER 8085 Microprocessor Trainer System because, it contains an internal speaker to get voice at specific frequencies and internal numpad to enter assembly numerical codes. The trainer has the mos services in the memory which make easy to reach the speaker , keypad and enter the opcode of the program.

5-1-Algorithm schema of 7 note electronic organ



5-2-Converting decimal note frequency to Hexadecimal

Decimal		Hexadecimal	
Note	Remainder	Remainder	Note
C (do) 262d			
262/16 =16	6	6	
16/16 = 1	0	0	
1/16 = 0	1	1	**0106h**
D (re) 294d			
294/16 =18	6	6	
18/16 = 1	2	2	
1/16 = 0	1	1	**0126h**
E (mi) 330d			
330/16 =20	10	A	
20/16 = 1	4	4	
1//16 = 0	1	1	**014Ah**
F (fa) 349d			
349/16 =21	13	D	
21/16 = 1	5	5	
1/16 = 0	1	1	**015Dh**
G (sol) 392d			
392/16 =24	8	8	
24/16 = 1	8	8	
1/16 = 0	1	1	**0188h**
A (la) 440d			
440/16 =27	8	8	
27/16 = 1	11	B	
1/16 = 0	1	1	**01B8h**
H (si) 494d			
494/16=30	14	E	
30/16 = 1	14	E	
1/16 = 0	1	1	**01EEh**

5-3- Assembly Program Subroutine of the Electronic Organ

```
mos     equ     1000h
keyin   equ      0Bh
pitch   equ      10h
delay   equ      14h
```

LABEL	MNEMONIC	OPERAND	COMMENT
	ORG	0FF01h	; program starts at this address
	MVI	C,keyin	; keyin sevice routine
	CALL	MOS	; call mos for keyin service
	MOV	L,A	; mov key value to acumulator
	CPI	01	; compare if key presed is 01
	JZ	DO	; jump if key 01
	CPI	02	; compare if key presed is 02
	JZ	RE	; jump if key 02
	CPI	03	; compare if key presed is 03
	JZ	MI	; jump if key 03
	CPI	04	; compare if key presed is 04
	JZ	FA	; jump if key 04
	CPI	05	; compare if key presed is 05
	JZ	SOL	; jump if key 05
	CPI	06	; compare if key presed is 06
	JZ	LA	; jump if key 06
	CPI	07	; compare if key presed is 01
	JZ	SI	; jump if key 01
	JMP	START	; else jmp to start
DO:	LXI	D, 0106h	; load d do note frequency
	JMP	SOUND	; jump to sound

LABEL	MNEMONIC	OPERAND	COMMENT
RE:	LXI	D, 0126h	; load d re note frequency
	JMP	SOUND	; jump to sound
MI:	LXI	D, 014Ah	; load d mi note frequency
	JMP	SOUND	; jump to sound
FA:	LXI	D, 015Dh	; load d FA note frequency
	JMP	SOUND	; jump to sound
SOL:	LXI	D, 0188h	; load sol note frequency
	JMP	SOUND	; jump to sound
LA:	LXI	D, 01B8h	; load d la note frequency
	JMP	SOUND	; jump to sound
SI:	LXI	D, 01EEh	; load d si note frequency
	JMP	SOUND	; jump to sound
SOUND:	MVI	C,pitch	; pitch sevice routine
	CALL	MOS	; call mos for pitch service
DELAY:	LXI	H, XXXX	; load delay value
	MVI	C,delay	; delay sevice routine
	CALL	MOS	; call mos for delay service
	LXI	D,0000h	; load d 0 for stop speaker
	MVI	C,pitch	; pitch sevice routine
	CALL	MOS	; call mos for pitch service
	JMP	START	; do it again

5-4- Machine Language of the Program

ADDRESS	DATA	INSTRUCTION
FF01	0E	MVI C, OB
FF02	0B	
FF03	CD	CALL MOSS
FF04	00	
FF05	10	
FF06	7D	MOV L, A
FF07	FE	CPI 01
FF08	01	
FF09	CA	JZ DO
FF0A	33	
FF0B	FF	
FF0C	FE	CPI 02
FF0D	02	
FF0E	CA	JZ RE
FF0F	39	
FF10	FF	
FF11	FE	CPI 03
FF12	03	
FF13	CA	JZ MI
FF14	3F	
FF15	FF	
FF16	FE	CPI 04
FF17	04	
FF18	CA	JZ FA
FF19	45	
FF1A	FF	
FF1B	FE	CPI 05

ADDRESS	DATA	INSTRUCTION
FF1C	05	
FF1D	CA	JZ MI
FF1E	4C	
FF1F	FF	
FF20	FE	CPI 06
FF21	06	
FF22	CA	JZ FA
FF23	53	
FF24	FF	
FF25	FE	CPI 07
FF26	07	
FF27	CA	JZ SOL
FF28	59	
FF29	FF	
FF2A	C3	JMP 01FF
FF2B	01	
FF2C	FF	
FF2D	00	NOP
FF2E	00	NOP
FF2F	00	NOP
FF30	00	NOP
FF31	00	NOP
FF32	00	NOP
FF33	11	LXI D, 0106
FF34	06	
FF35	01	
FF36	C3	JMP FF6A
FF37	6A	

ADDRESS	DATA	INSTRUCTION
FF38	FF	
FF39	11	LXI D, 0126h
FF3A	26	
FF3B	01	
FF3C	C3	JMP FF6A
FF3D	6A	
FF3E	FF	
FF3F	11	LXI D, 014Ah
FF40	4A	
FF41	01	
FF42	C3	JMP FF6A
FF43	6A	
FF44	FF	
FF45	11	LXI D, 015Dh
FF46	5D	
FF47	01	
FF48	00	NOP
FF49	C3	JMP FF6A
FF4A	6A	
FF4B	FF	
FF4C	00	NOP
FF4D	11	LXI D, 0188h
FF4E	88	
FF4F	01	
FF50	C3	JMP FF6A
FF51	6A	
FF52	FF	

ADDRESS	DATA	INSTRUCTION
FF53	11	LXI D, 01B8h
FF54	B8	
FF55	01	
FF56	C3	JMP FF6A
FF57	6A	
FF58	FF	
FF59	11	LXI D, 01EEh
FF5A	EE	
FF5B	01	
FF5C	C3	JMP FF6A
FF5D	6A	
FF5E	FF	
FF5F	11	LXI D, 01EEh
FF60	EE	
FF61	01	
FF62	C3	JMP FF6A
FF63	6A	
FF64	FF	
FF65	00	NOP
FF66	00	NOP
FF67	00	NOP
FF68	00	NOP
FF69	00	NOP
FF6A	0E	MVI C, 1O
FF6B	10	
FF6C	CD	CALL MOS
FF6D	00	
FF6E	10	

ADDRESS	DATA	INSTRUCTION
FF6F	21	LXI H, DELAY
FF70	5F	
FF71	3F	
FF72	0E	MVI C, 14
FF73	14	
FF74	CD	CALL MOS
FF75	00	
FF76	10	
FF77	11	LXI D, 0000
FF78	00	
FF79	00	
FF7A	0E	MVI C, 1O
FF7B	10	
FF7C	CD	CALL MOS
FF7D	00	
FF7E	10	
FF7F	00	NOP
FF80	00	NOP
FF81	C3	JMP 01FF
FF82	01	
FF83	FF	

CONCLUSION

In practice, functionality of a microprocessor on electronic devices is observed during the preparation of the project. Like in any other control mechanism, using a microprocessor for controlling input values to determine outputs is very efficient way for an electronic music instrument.

Although developed computer system needs complicated microprocessor (including millions of transistors), we need the simple ones in our simple electronic devices used at home and offices. We can combine hardware components to make basic devices which help us in daily works. They can be any smart devices including input and output components, storage mechanicsm to hold any needed data such as passwords and a programmable microprocessor. Devices such as burglar alarms, remote control is designed by these components.

I have learned about 8085 microprocessor architecture and programming methods in assembly through working on project. The project is just playing seven music notes on a prepared electronic set including a microprocessor and internal speaker and a numpad. But the idea behind construction of this electronic organ is very important. It is the main concept of creating new smart devices which can produce works in response of the given input values.

This project is the first step to creating this type of devices and I try to develop myself for working at a company producing electronic systems.

REFERENCES

[1] Ibrahim D.," 8085/Z80 Microprocessor Problems", Near East University Press, Nicosia 2001.

[2] Ibrahim D., "8085/Z80 Microprocessor Answers", Near East University Press, Nicosia 2001.

[3] Ibrahim D., Uyar K.," 8085 Laboratory Manual", Near East University Press, Nicosia 2006.

[4] Uyar K.,"Microprocessors", Lecture notes", Near East University Press, Nicosia 2005.

[5] Notes and Note frequency , web site: http://en.wikipedia.org/wiki/Note ,Retrieved May 2006.

[6] ComputerArchitecture and Operating Systems, web site : http://csnet.otago.ac.nz/cosc243/lectures.html,Retrieved March 2006.

[7] The Primer Trainer Self Instruction Manual Revision 2.5, web site : http://www.emacinc.com/trainers/primer_textbooks.htm.

[8] The Primer Trainer Self Application Manual Revision 2.0 ,web site : http://www.emacinc.com/trainers/primer_textbooks.htm.

[9] Electronic Circuit & Tutorials, web site: http://www.hobbyprojects.com/microprocessor_tutorials/8085_data_transfer_group.html, Retrieved May 2006.

APPENDIX
Mnemonics Opcode Instruction Set Table

MOVE, LOAD and STORE

40	MOV	B,B
41	MOV	B,C
42	MOV	B,D
43	MOV	B,E
44	MOV	B,H
45	MOV	B,L
46	MOV	B,M
47	MOV	B,A
48	MOV	C,B
49	MOV	C,C
4A	MOV	C,D
4B	MOV	C,E
4C	MOV	C,H
4D	MOV	C,L
4E	MOV	C,M
4F	MOV	C,A
50	MOV	D,B
51	MOV	D,C
52	MOV	D,D
53	MOV	D,E
54	MOV	D,H
55	MOV	D,L
56	MOV	D,M
57	MOV	D,A
58	MOV	E,B
59	MOV	E,C
5A	MOV	E,D
5B	MOV	E,E
5C	MOV	E,H
5D	MOV	E,L
5E	MOV	E,M
5F	MOV	E,A
60	MOV	H,B
61	MOV	H,C
62	MOV	H,D
63	MOV	H,E
64	MOV	H,H
65	MOV	H,L
66	MOV	H,M
67	MOV	H,A
68	MOV	L,B
69	MOV	L,C
6A	MOV	L,D
6B	MOV	L,E
6C	MOV	L,H
6D	MOV	L,L
6E	MOV	L,M
6F	MOV	L,A
70	MOV	M,B
71	MOV	M,C
72	MOV	M,D
73	MOV	M,E
74	MOV	M,H
75	MOV	M,L
77	MOV	M,A
78	MOV	A,B
79	MOV	A,C
7A	MOV	A,D
7B	MOV	A,E
7C	MOV	A,H
7D	MOV	A,L
7E	MOV	A,M
7F	MOV	A,A

3E nn	MVI A,byte
06 nn	MVI B,byte
0E nn	MVI C,byte
16 nn	MVI D,byte
1E nn	MVI E,byte
26 nn	MVI H,byte
2E nn	MVI L,byte
36 nn	MVI M,byte

01 nnnn	LXI B,dble
11 nnnn	LXI D,dble
21 nnnn	LXI H,dble
31 nnnn	LXI SP,dble

02	STAX	B
12	STAX	D
0A	LDAX	B
1A	LDAX	D
32 nnnn	STA	adr
3A nnnn	LDA	adr
22 nnnn	SHLD	adr
2A nnnn	LHLD	adr
EB	XCHG	

STACK

C5	PUSH	B
D5	PUSH	D
E5	PUSH	H
F5	PUSH	PSW

C1	POP	B
D1	POP	D
E1	POP	H
F1	POP	PSW
E3	XTHL	
F9	SPHL	
33	INX	SP
3B	DCX	SP

ARITHEMATICS

C6 nn	ADI	byte
CE nn	ACI	byte
80	ADD	B
81	ADD	C
82	ADD	D
83	ADD	E
84	ADD	H
85	ADD	L
86	ADD	M
87	ADD	A
88	ADC	B
89	ADC	C
8A	ADC	D
8B	ADC	E
8C	ADC	H
8D	ADC	L
8E	ADC	M
8F	ADC	A
D6 nn	SUI	byte
DE nn	SBI	byte
90	SUB	B
91	SUB	C
92	SUB	D
93	SUB	E
94	SUB	H
95	SUB	L
96	SUB	M
97	SUB	A
98	SBB	B
99	SBB	C
9A	SBB	D
9B	SBB	E
9C	SBB	H
9D	SBB	L
9E	SBB	M
9F	SBB	A

09	DAD	B
19	DAD	D
29	DAD	H
39	DAD	SP

LOGICAL

E6 nn	ANI	byte
EE nn	XRI	byte
F6 nn	ORI	byte
A0	ANA	B
A1	ANA	C
A2	ANA	D
A3	ANA	E
A4	ANA	H
A5	ANA	L
A6	ANA	M
A7	ANA	A
A8	XRA	B
A9	XRA	C
AA	XRA	D
AB	XRA	E
AC	XRA	H
AD	XRA	L
AE	XRA	M
AF	XRA	A
B0	ORA	B
B1	ORA	C
B2	ORA	D
B3	ORA	E
B4	ORA	H
B5	ORA	L
B6	ORA	M
B7	ORA	A

COMPARE

FE nn	CPI	byte
B8	CMP	B
B9	CMP	C
BA	CMP	D
BB	CMP	E
BC	CMP	H
BD	CMP	L
BE	CMP	M
BF	CMP	A

ROTATE

07	RLC	
17	RAL	
0F	RRC	
1F	RAR	

JUMP

C3 nnnn	JMP	adr
DA nnnn	JC	adr
D2 nnnn	JNC	adr
CA nnnn	JZ	adr
C2 nnnn	JNZ	adr
F2 nnnn	JP	adr
FA nnnn	JM	adr
EA nnnn	JPE	adr
E2 nnnn	JPO	adr
E9	PCHL	

CALL

CD nnnn	CALL	adr
DC nnnn	CC	adr
D4 nnnn	CNC	adr
CC nnnn	CZ	adr
C4 nnnn	CNZ	adr
F4 nnnn	CP	adr
FC nnnn	CM	adr
EC nnnn	CPE	adr
E4 nnnn	CPO	adr

RETURN

C9	RET
D8	RC
D0	RNC
C8	RZ
C0	RNZ
F0	RP
F8	RM
E8	RPE
E0	RPO

RESTART

C7	RST	0
CF	RST	1
D7	RST	2
DF	RST	3
E7	RST	4
EF	RST	5
F7	RST	6
FF	RST	7

INPUT/OUTPUT

DB nn	IN	byte
D3 nn	OUT	byte

INCREMENT/DECREMENT

04	INR	B
0C	INR	C
14	INR	D
1C	INR	E
24	INR	H
2C	INR	L
34	INR	M
3C	INR	A
03	INX	B
13	INX	D
23	INX	H
05	DCR	B
0D	DCR	C
15	DCR	D
1D	DCR	E
25	DCR	H
2D	DCR	L
35	DCR	M
3D	DCR	A
0B	DCX	B
1B	DCX	D
2B	DCX	H

SPECIALS

2F	CMA
37	STC
3F	CMC
27	DAA

CONTROL

00	NOP
F3	DI
FB	EI
76	HLT
20	RIM
30	SIM