March 16, 1976

To the Graduate Council:

I am submitting herewith a thesis written by Jayantilal Govindbhai Patel entitled "A Microcomputer Design and Application." I recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

R. W. Rochelle, Major Professor

We have read this thesis
and recommend its acceptance:

Accepted for the Council:

Vice Chancellor
Graduate Studies and Research

A MICROCOMPUTER DESIGN AND APPLICATION

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Jayantilal Govindbhai Patel

June 1976

1286634

ACKNOWLEDGEMENTS

The author would like to thank Dr. R. W. Rochelle for his support during the preparation of this thesis. Appreciation is extended also to Dr. Robert E. Bodenheimer and Dr. J. M. Bailey for serving on the author's committee. Gratitude is extended to Mr. Alan Metcalf who furnished the frequency counter for the application. Finally, the author would like to thank Dr. J. C. Hung for giving him generous space to set up his experiment and Dr. T. V. Blalock for providing the necessary test instruments.

ABSTRACT

A microcomputer using the Intel-8080 microprocessor has been designed and implemented to provide an educational tool for illustrating microcomputer operation. Photographs of the timing waveforms for various instructions of the microcomputer were taken to show the interrelation of the microprocessor control signals. An application is described to demonstrate the case in which the microcomputer can be used to perform a dedicated task.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
II.	INTEL-8080 MICROPROCESSOR ARCHITECTURE AND OPERATION	5
	The 8080 Architecture	11
	The Processor Cycle	14
	Machine Cycle Identification for Various Instructions	16
III.	DEVELOPMENT OF MICROCOMPUTER SYSTEM	32
	Interfacing the Memory and Input/Output Ports	32
	Status Information Definition	35
	Logic Circuit Development for Input Multiplexer Control Lines	38
	Logic Circuit Development for Memory Control	40
	Logic Circuit for Output Port Latch Control Line	40
	Front Panel Control Logic Operation	42
	Run/Stop Logic	45
	Single-Step Logic	45
	Examine Logic	45
	Examine-Next	48
	Logic Circuit for Load and Load-Next Functions	49
	Clock Circuit for the Microcomputer	49
IV.	FRONT PANEL OPERATION	58
	Examine (EXM)	60
	Examine-Next	60

CHAPTER PAGE

Load 60

Load-Next 61

Reset 61

Single-Step (S.S.) 61

Clear (CLR) 61

Memory Access (MEM. ACCE.) 61

Run/Stop 62

Power Supply Connections 62

Connector 62

Summary 62

V. APPLICATION 64

Autoranging a Frequency Counter 64

Conclusion 71

VI. CONCLUSION 72

LIST OF REFERENCES 73

APPENDICES 75

Appendix A 76

Appendix B 85

VITA 90

LIST OF TABLES

TABLE		PAGE
II-1.	Functional Description of Each Pin of the 8080 Microprocessor	6
III-1.	The 8080 Status Bit Definitions	36

LIST OF FIGURES

FIGURE		PAGE
I-1.	Typical Microcomputer System	2
II-1.	8080 Pin Configuration	10
II-2.	8080 Block Diagram	12
II-3.	8080 Instruction Cycle	15
II-4.	Typical FETCH Machine Cycle	18
II-5.	"ADD B" Instruction Cycle	21
II-6.	"ADD M" Instruction Cycle	23
II-7.	The "SHLD" Instruction Cycle	25
II-8.	"DAD B" Instruction Cycle	27
II-9.	"IN" Instruction Cycle	29
II-10.	"OUT" Instruction Cycle	31
III-1.	Interfacing 8080 with Memory and Input/Output Port	33
III-2.	Buffering the Address and Data Lines	34
III-3.	Input Multiplexer Control Line Logic	39
III-4.	$R/\overline{W}$ Control Logic	41
III-5.	Output Port Latch Control Logic	43
III-6.	Run/Stop Flip-Flop and Ready Logic	44
III-7.	Single-Step Logic	46
III-8.	Examine Logic	47
III-9.	Examine-Next, Load-Next Logic	50
III-10.	Load, Load-Next Logic	51
III-11.	System Clock	53

FIGURE		PAGE
III-12.	Central Processing Unit (CPU)	54
III-13.	Memory and Output Ports	55
III-14.	Front Panel Control	56
III-15.	Light-Emitting Diode Display	57
IV-1.	Front Panel of the Microcomputer	59
IV-2.	Connector Pin Assignments	63
V-1.	Frequency Counter Block Diagram	65
V-2.	Interface Circuit for Autoranging the Frequency Counter	67
V-3.	Input/Output Port Pin Assignments	69

CHAPTER I

INTRODUCTION

Recent advancements in the technology of large-scale integrated (LSI) circuits have tremendously increased the utility of digital electronic circuits. It was clear in the late 1960's that the technology needed to put thousands of transistors on a single chip was rapidly approaching; but it was not clear what electronic applications could usefully employ so many devices. Memory arrays were an obvious application. After the memories were developed, a companion processor was needed to help utilize and sell the memories. Thus, the microprocessor was born. The development of the microprocessor was therefore an incidental by-product of the memory development effort pioneered by the founders of Intel Corporation (1, 2). Following the introduction of the first microprocessor, an impressive number of microprocessor applications quickly appeared. However, the most important application is the microcomputer utilizing an LSI microprocessor as its central processing unit (CPU). Core or semiconductor memory is used to store instructions and data (Figure I-1). The microprocessor interprets and controls instructions and includes the hardware registers, arithmetic logic unit, and input/output control logic.

The advantages of using a microcomputer are: quick design cycles, small physical size, low costs, and flexibility of design. Hard-wired logic is being replaced by the microcomputers; consequently, design board layouts are simplified. The read-only-memory

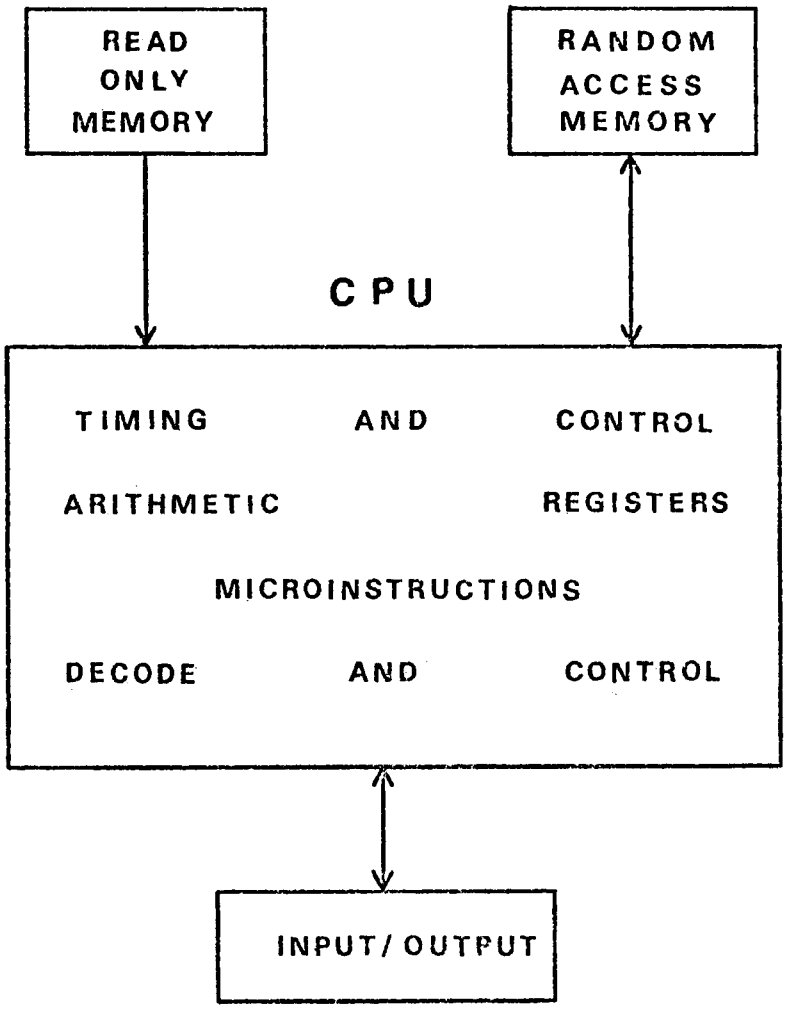


FIGURE I-1. Typical microcomputer system.

(ROM) in the microcomputer replaces a large number of integrated circuits used in hard-wired logic designs. Therefore, the only interconnect wiring on the design board runs from the microprocessor to memory and input/output devices of the microcomputer. The logic system design becomes simply the manipulation of functional blocks. The control sequence is stored and executed as a software program.

Expansion and flexibility within a microcomputer system design is accomplished by simply modifying the control program. In contrast, modifications of a hardwired logic design would require redesigning and laying out the new circuit diagram, utilizing additional components, and rewiring the layout. Flexibility within a microcomputer is described by the fact that random data collecting, complex logic decisions, and arithmetic computations can be performed by software techniques. Generating these functions by hard-wired logic would require many I.C. components (3).

One disadvantage of the microcomputer is the speed at which a microprocessor operates. The speed is limited by the type of metal-oxide-semiconductor (MOS) process used, as these processes improve, so will the speed. Another disadvantage is the long time required in order to develop a software program; consequently, the software costs are presently much greater than the hardware costs for small quantity production.

To understand the usefulness of the microprocessor, a microcomputer using the Intel-8080 microprocessor has been developed. Chapter II includes the architecture of the 8080. Timing waveforms have been studied on an oscilloscope and the photographs of these waveforms

for certain types of instructions have been taken and included also in Chapter II.

Interconnections of the 8080 microprocessor to the other devices to develop a microcomputer are described in Chapter III. Logic circuit diagrams to synchronize the operations of these devices are also explained.

Chapter IV is developed to help the operator to understand the functions of the front panel switches and light-emitting diodes (LED) of the implemented system. This includes the procedure to load and modify the software program.

Microcomputers can be used for any application in which random logic control requires optimization. An application of the computer developed in Chapter III is described in Chapter V to demonstrate the case in which the microcomputer can be used to perform a dedicated task.

CHAPTER II

INTEL-8080 MICROPROCESSOR ARCHITECTURE AND OPERATION

A forty-pin LSI chip contains the Intel-8080 microprocessor. Table II-1 describes the function of all of the 8080 input/output pins. The pin configuration is shown in Figure II-1. All pins, except the clock inputs, are at TTL levels.

The 8080 CPU has 78 instructions with provisions for arithmetic and logical operations, register-register and register-memory transfers, subroutine handling, input/output transactions, and decimal arithmetic. Four internal status flags enable the user to program conditional branches based on carry, sign, zero, and parity.

Using its 16 latched address lines, the 8080 can access 65,536 (64K) memory locations directly. As many as 256 input devices and up to 256 output devices may be addressed during I/O operations, using either A_0-A_7 or A_8-A_{15} address lines.

The 8080 contains six eight-bit index registers. Two of these, the H and the L registers, are designed to double as an address pointer during the execution of memory referencing instructions. A sixteen-bit program counter enables the CPU to address instructions stored in any portion of memory, and a sixteen-bit stack pointer permits the unlimited nesting of subroutines (or multiple-level interrupts).

TABLE II-1

FUNCTIONAL DESCRIPTION OF EACH PIN
OF THE 8080 MICROPROCESSOR

Pin	Symbol	Description of Function
2	Vss	Ground Reference.
3-10	D7-D0 (Input/output) tri-state	DATA BUS: the data bus provides bidirectional communication between memory and I/O devices for instructions and data transfers. D0 is the least significant bit.
11	Vbb	-5 Volts (substrate bias).
12	RESET (input)	RESET: while the RESET signal is activated, the content of the program counter is cleared and the instruction register is set to zero. After RESET, the program will start at location 0 in memory. The INTE and HLDA flip/flops are also reset.
13	HOLD (input)	HOLD: the HOLD signal requests the CPU to enter the HOLD state. The HOLD state allows an external device to gain control of the 8080 address and data bus as soon as the 8080 has completed its use of these buses for the current machine cycle. It is recognized under the following conditions: •the CPU is in the HALT STATE •the CPU is in the T2 or Tw state and the READY signal is active. As a result of entering the HOLD state the CPU ADDRESS BUS (A_{15}-A_0) and DATA BUS (D_7-D_0) will be in their high impedance state. The CPU acknowledges its state with the HOLD ACKNOWLEDGE (HLDA) pin.

TABLE II-1 (continued)

Pin	Symbol	Description of Function
14	INT (input)	INTERRUPT REQUEST: the CPU recognizes an interrupt request on this line at the end of the current instruction or while halted. If the CPU is in the HOLD state or if the Interrupt Enable flip/flop is reset it will not honor the request. The CPU acknowledges acceptance of an interrupt by sending out the INTA (Interrupt Acknowledge) status signal at SYNC time. During the next instruction fetch cycle the program counter is not advanced and a one-byte instruction (usually RESTART) can be inserted.
15	ϕ_2	ϕ_2: clock input.
16	INTE (output)	INTERRUPT ENABLE: indicates the content of the internal interrupt enable flip/flop. This flip/flop may be set or reset by the EI and DI instructions and inhibits interrupts from being accepted by the CPU if it is reset. It is automatically reset (disabling further interrupts) at time T_1 of the instruction fetch cycle (M_1) when an interrupt is accepted and is also reset by the RESET signal.
17	DBIN (output)	DATA BUS IN: the DBIN signal indicates to external circuits that the data bus in in the input mode. This signal should be used to enable the gating of data onto the 8080 data bus from memory or I/O.
18	$\overline{WR}$ (output)	WRITE: the $\overline{WR}$ signal is used for memory WRITE or I/O output control. The data on the data bus are stable while the $\overline{WR}$ signal is active ($\overline{WR}$ = 0).

TABLE II-1 (continued)

Pin	Symbol	Description of Function
19	SYNC (output)	SYNCHRONIZING SIGNAL: the SYNC pin provides a signal to indicate the beginning of each machine cycle. (Instructions can be executed in 1, 2, 3, 4, or 5 machine cycles, and the status information of each machine cycle is sent to external latches at SYNC time.)
20	Vcc	+5 Volts.
21	HLDA (output)	HOLD ACKNOWLEDGE: the HLDA signal appears in response to the HOLD signal and indicates that the data and address bus will go to the high impedance state. The HLDA signal begins at: T_3 for READ memory or input. The clock period following T_3 for WRITE memory or OUTPUT operation. In either case, the HLDA signal appears after the rising edge of ϕ_1 and high impedance occurs after the rising edge of ϕ_2.
22	ϕ_1	Clock input.
23	READY (input)	READY: the READY signal indicates to the 8080 that valid memory or input data are available on the 8080 data bus. This signal is used to synchronize the CPU with slower memory or I/O devices. If after sending an address out the 8080 does not receive a READY input, the 8080 will enter a WAIT state for as long as the READY line is low.
24	WAIT (output)	WAIT: the WAIT signal acknowledges that the CPU is in a wait state.
28	Vdd	+12 Volts.

TABLE II-1 (continued)

Pin	Symbol	Description of Function
1, 25-27 29-40	A_{15}-A_0 (output tristate)	ADDRESS BUS: the address bus provides the address to memory (up to 64 K 8-bit words) or denotes the I/0 device number for up to 256 input and 256 output devices. A0 is the least significant address bit.

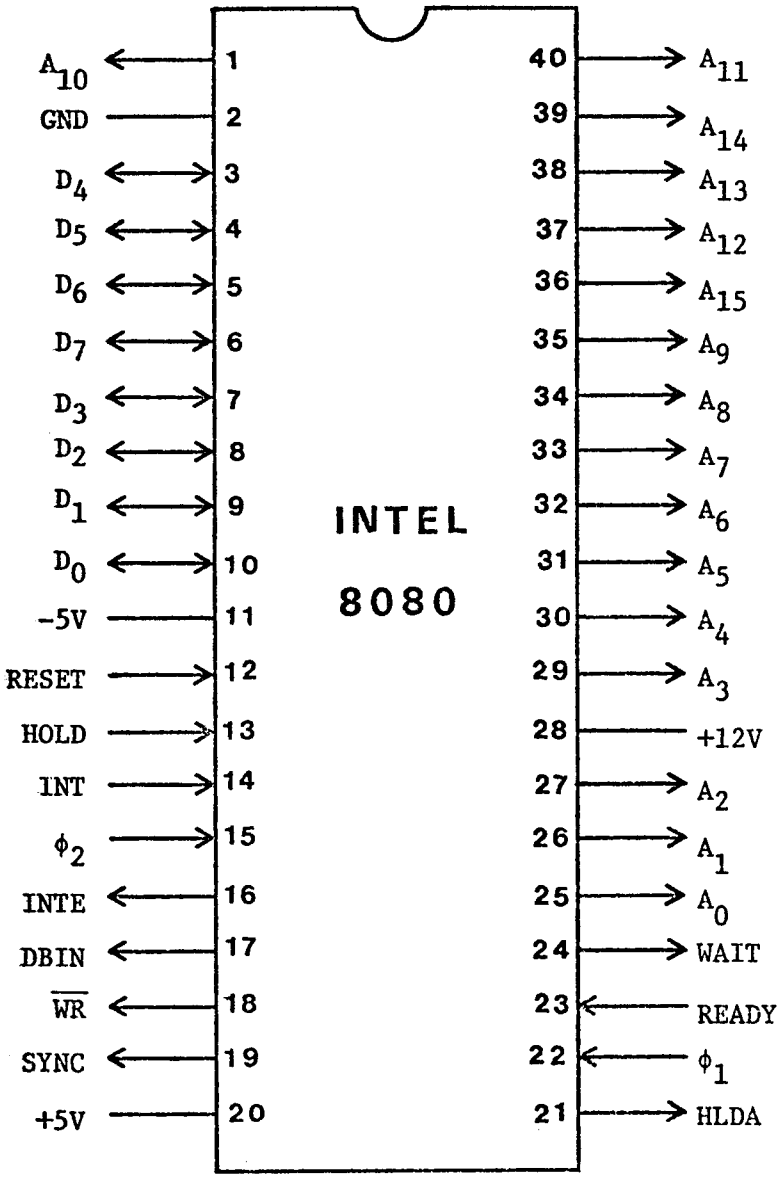


FIGURE II-1. 8080 pin configuration.

I. THE 8080 ARCHITECTURE

Figure II-2 shows the block diagram of the 8080 microprocessor. The 8080 consists of the following functional units: register array and address logic, arithmetic and logic unit, instruction register and control section, bidirectional tri-state data bus buffer.

The register section is organized into six sixteen-bit registers: a program counter, a stack pointer, six eight-bit general purpose registers arranged in pairs (referred to as B,C; D,E; H,L), and a temporary register pair called W,Z. The program counter maintains the memory address of the current program instruction and is incremented automatically during every instruction fetch. The stack pointer contains the address of the next available stack location. Any portion of the RAM can be used as a stack by properly initializing the stack pointer. The six general purpose registers are program addressable either as single registers or as register pairs, whereas, the temporary register is used only for the internal excution of an instruction.

The register-select multiplexer is used to transfer eight-bit data bytes between the internal bus and the selected register array. Sixteen-bit address proceeds between the register array and the address latch or the incrementer/decrementer circuit. The address latch drives the sixteen-bit address buffer as well as the incrementer/decrementer circuit which sends the new data back to the register array after being incremented, decremented, or left alone.

The arithmetic and logic unit (ALU) consists of an accumulator, a temporary register, a five-bit flag register (zero, carry, sign, parity and auxiliary carry), a carry link flip-flop, and a temporary

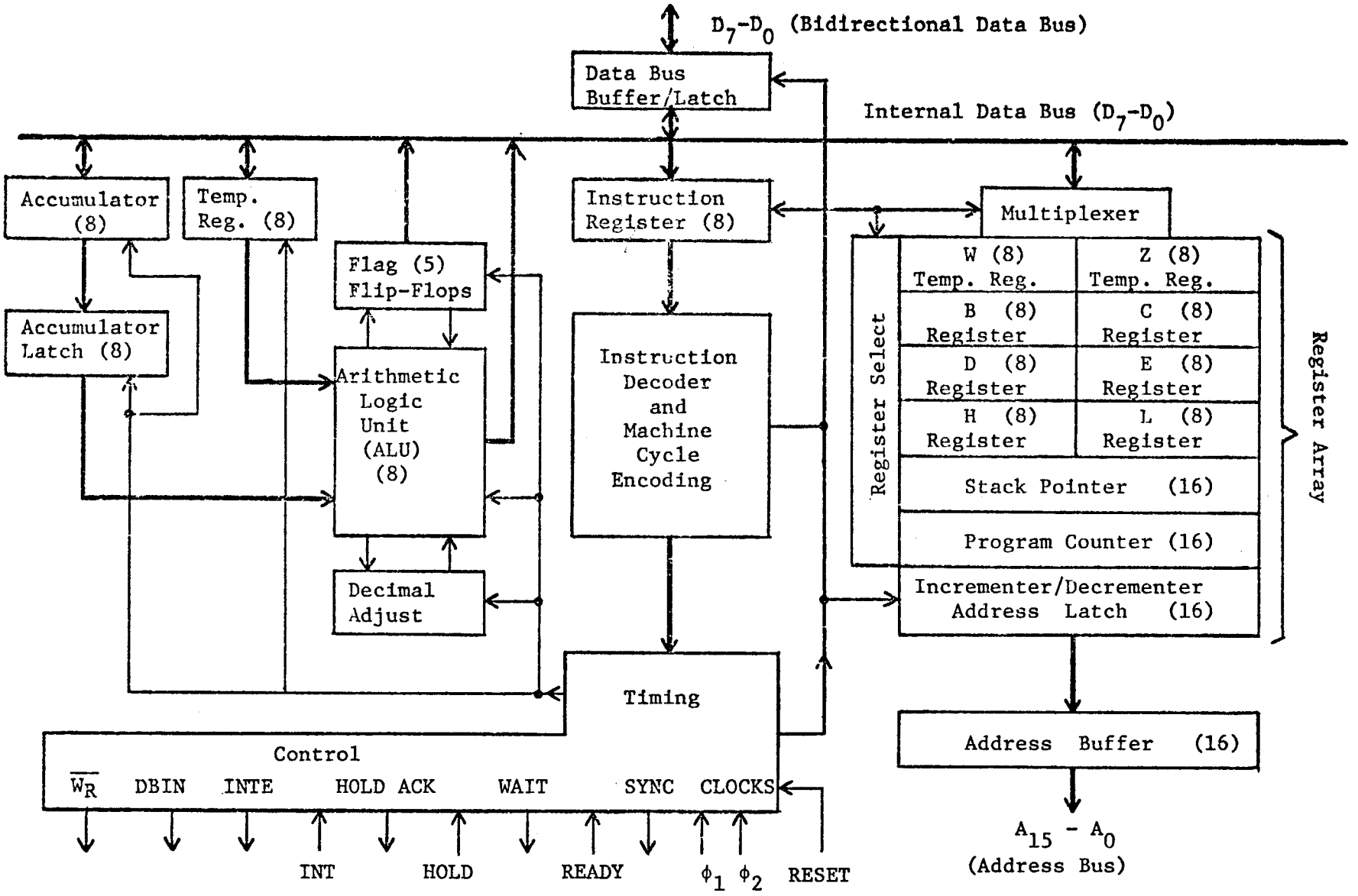


FIGURE II-2. 8080 block diagram

accumulator and a temporary carry flip-flop. Arithmetic, logical and rotate operations are performed in the ALU. The data from the internal bus to the ALU are transferred via the temporary register and the temporary accumulator and carry flip-flop. The result can be transferred to the internal bus or to the accumulator. The flag register receives the information from the temporary register and the results from the ALU. The accumulator receives data from the ALU and the internal bus and transfers data to the temporary accumulator and the internal bus.

The first byte of an instruction containing the operation code is transferred to the instruction register during an instruction fetch cycle. The contents of the instruction register are decoded in the instruction decoder to provide, in combination with various timing signals, the control signals for the register array, ALU and data buffer blocks. In addition, the outputs from the instruction decoder and external control signals feed the timing and state control section which generates the state and timing signals.

The eight-bit bidirectional three-state buffer isolates the microprocessor's internal bus from the external data bus. In the output mode the data on the internal bus are loaded into an eight-bit latch which, in turn, drives the data bus output buffers. During the input mode, the output buffers are switched off and the data from the external data bus are transferred to the internal bus.

II. THE PROCESSOR CYCLE

An instruction cycle is defined as the time required to fetch and execute an instruction. Instructions in the 8080 contain one to three bytes. Every instruction cycle requires from one to five machine cycles called M_1, M_2, M_3, M_4, M_5. The fetch portion of an instruction cycle requires one machine cycle for each byte to be fetched. The duration of the execution portion of the instruction cycle depends on the kind of instruction that has been fetched. Some instructions do not require any machine cycles other than those necessary to fetch the instruction; other instructions, however, require additional machine cycles to write or read data to or from memory or I/O devices. The DAD instruction is an exception in that it requires two additional machine cycles to complete an internal register pair add.

Figure II-3 shows the timing diagram for a basic instruction cycle. Each machine cycle requires from three to five states T_1, T_2, T_3, T_4, T_5 for its completion. Each state (except for the WAIT, HOLD, and HALT) has the duration of one clock period. Machine cycle M_1 is always a fetch cycle while machine cycles M_2 to M_5 are execution cycles.

During state T_1, the contents of the program counter are sent to the address bus, SYNC is true, and the data bus contains the status information for the current machine cycle. T_1 is followed by another state, T_2, during which the condition of the READY, HOLD, and HALT acknowledge signals are tested. If READY is true, T_3 can be entered; otherwise, the CPU will go into the wait state (TW) and stay there for as long as READY is false. During T_3, the data coming from memory is

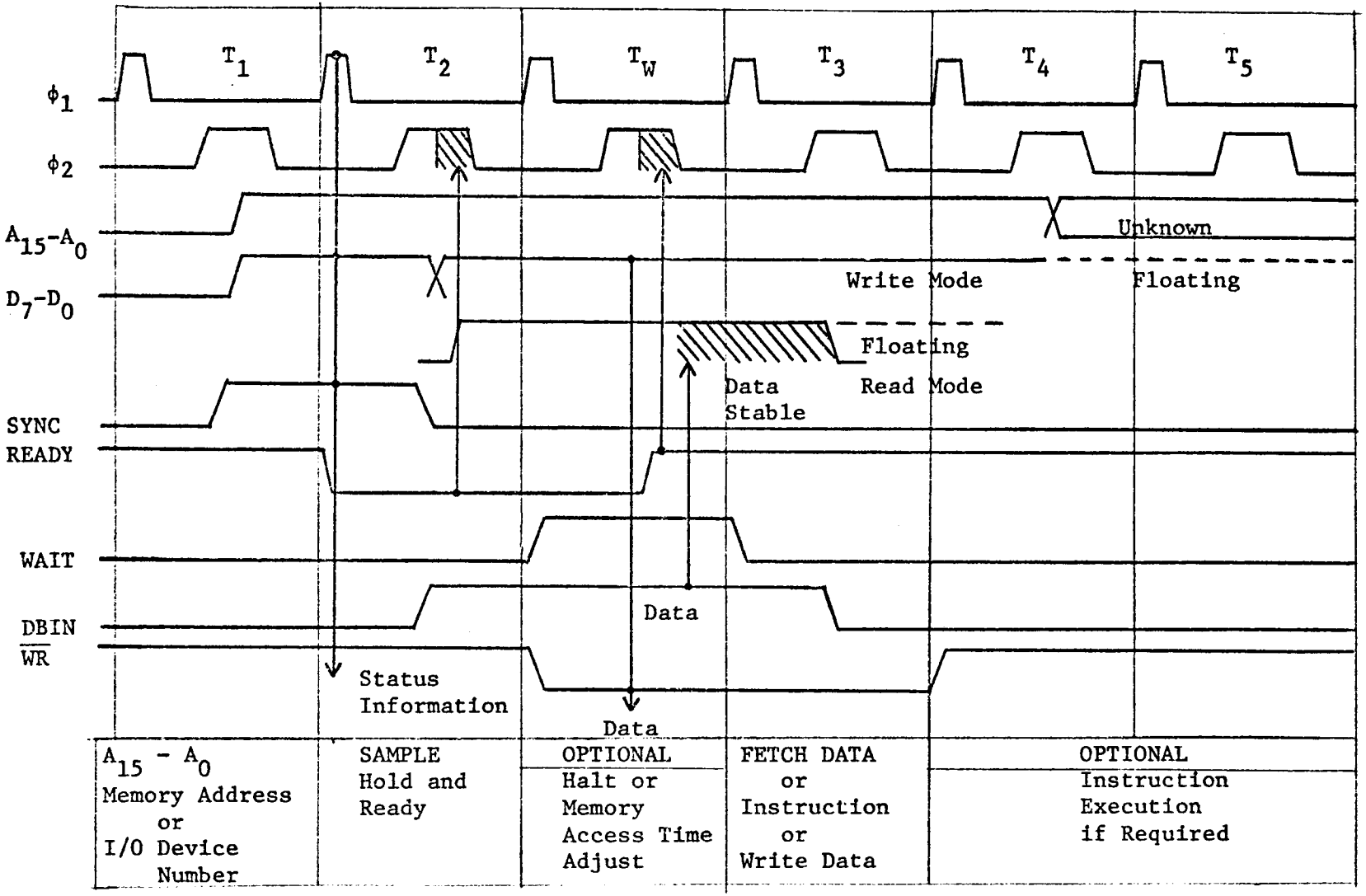


FIGURE II-3. 8080 instruction cycle.

available on the data bus and is transferred into the instruction register (during the fetch cycle only). The instruction is decoded by the instruction decoder and the basic signals to control the internal data transfers, the timing, and the machine cycle requirements of the new instructions are generated. At the end of T_4, if the instruction is complete, or else at the end of T_5, the 8080 goes back to T_1 and enters machine cycle M_2, unless the instruction requires only one machine cycle for its execution. In such cases, a new fetch cycle is entered. The loop is repeated for as many cycles and states as required by the instruction.

It is only during the last state of the last machine cycle that the interrupt request line is tested and a special M_1 cycle is entered, during which no program-counter incrementing takes place and INTERRUPT acknowledge status is sent out. During this cycle, one of eight possible restart instructions will be sent to the CPU by the interrupting device.

If a HALT instruction is executed, the processor enters a WAIT state and remains there until an interrupt is received.

III. MACHINE CYCLE IDENTIFICATION FOR VARIOUS INSTRUCTIONS

One purpose of this thesis is to study the timing waveforms on an oscilloscope for various instructions. Since the waveforms are fast, it is difficult to see them on an oscilloscope while single stepping through an instruction cycle. In order to overcome this difficulty, an instruction being studied is followed by a JMP (jump) instruction to

repeat the instruction. The oscilloscope is triggered externally by the ϕ_1 phase of the system clock. Thus the waveforms are made to appear steady on an oscilloscope. Photographs of these waveforms were taken with the help of a Polaroid oscilloscope camera using a multiple exposure technique.

As mentioned in the last section, the number of machine cycles required varies from instruction to instruction. The 8080 can transmit only one address per machine cycle. Thus the number of machine cycles required in any given instruction cycle are determined by the number of times that the processor must reference a memory address, or a peripheral device, in order to fetch and execute the instruction. The DAD instruction is the exception in the above statement since the last two machine cycles are used for an internal register-pair addition and not for memory reference. Every instruction cycle has at least one reference to memory during which the instruction is fetched. The execution of the instruction may or may not require further references to memory. The first machine cycle in every instruction cycle is therefore a FETCH. Beyond that it depends on the kind of instruction.

Figure II-4 shows the photograph of a typical fetch machine cycle. Photographs of the instruction cycles of the following instructions are taken to understand the instruction cycle.

ADD r

The ADD r instruction adds the contents of the specified register (r is the CPU's one of the six index registers; namely, B, C, D, E, H, L) to the contents of the accumulator and puts the result in the

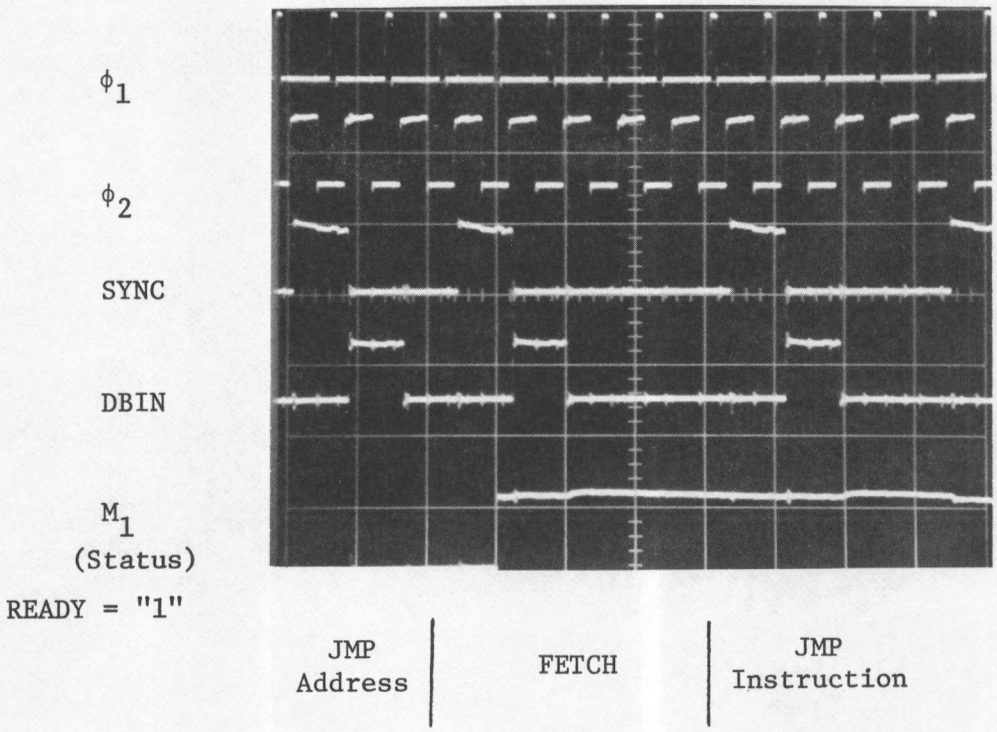


FIGURE II-4. Typical FETCH machine cycle.

accumulator. Add r instruction requires only a single machine cycle (FETCH) for its completion, since all the information necessary to execute the command is contained in the eight bits of the instruction code (Appendix B). The instruction can be loaded in the memory as follows:

	Inst.	Mnemonic	Bytes*	
	MVI	A, 00	2	Clear Register A
	MOV	B, A	1	Clear Register B
REPEAT:	ADD	B	1	Add A to B
	JMP	REPEAT	3	Repeat ADD B
		Total	7	

The above program requires seven bytes (seven memory locations) to store it. Each instruction is converted to the instruction code (instruction in the form of 0's and 1's) before it can be stored in memory. The following program repeats the above program with each instruction converted into a corresponding instruction code. The program is to be stored in the memory starting from the zero memory location.

Memory Locations	Instruction in MNEMONIC Form		Instruction in Code Form
0 0 0 0 0 0 0 0	MVI	A	0 0 1 1 1 1 1 0
0 0 0 0 0 0 0 1	00H		0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0	MOV	B,A	0 1 0 0 0 1 1 1
0 0 0 0 0 0 1 1	ADD	B	1 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0	JMP		1 1 0 0 0 0 1 1
0 0 0 0 0 1 0 1		03	0 0 0 0 0 0 1 1
0 0 0 0 0 1 1 0		03	0 0 0 0 0 0 1 1

*Refer to Appendix A for the list of the 8080 instruction set.

The photograph of the instruction cycles of the above program was taken and is shown in Figure II-5. As can be seen from Figure II-5, the instruction requires only one machine cycle which consists of four states. First three states are used to extract the instruction from memory, and the fourth state is used to accomplish the desired addition.

ADD M

ADD M instruction is used to add the contents of a specified memory location to the pre-existing contents of the accumulator. The program to observe the ADD M instruction cycle is shown below in Mnemonic as well as machine language form.

	Mnemonic	Bytes	
	MVI L, 09 H	2	Lower order address of M
	MOV H, L	1	Higher order address of M
	MVI A, 00	2	Clear A
REPEAT:	ADD M	1	
	JMP REPEAT	3	
	Total	9	

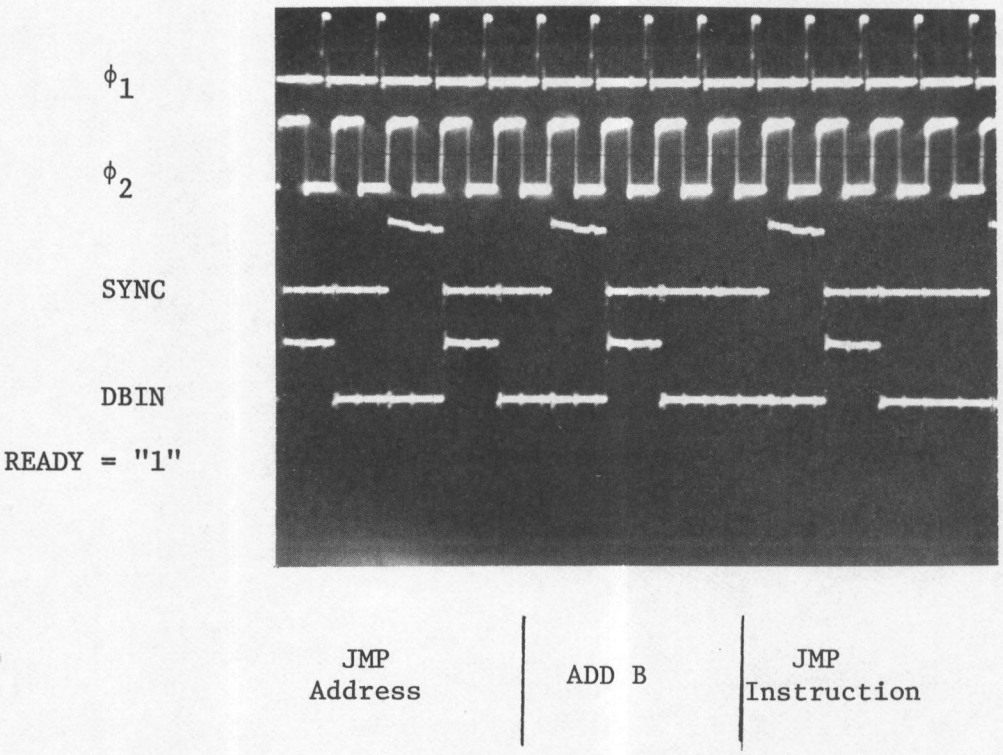


FIGURE II-5. "ADD B" instruction cycle.

Memory Location	MNEMONIC	Instruction Code
0 0 0 0 0 0 0 0	MVI L	0 0 1 0 1 1 1 0
0 0 0 0 0 0 0 1	09 H	0 0 0 0 1 0 0 1
0 0 0 0 0 0 1 0	MOV H,L	0 1 1 0 0 1 0 1
0 0 0 0 0 0 1 1	MVI A	0 0 1 1 1 1 1 0
0 0 0 0 0 1 0 0	00M	1 0 0 0 0 1 0 1
0 0 0 0 0 1 1 0	JMP	1 1 0 0 0 0 1 1
0 0 0 0 0 1 1 1	05	0 0 0 0 0 1 0 1
0 0 0 0 1 0 0 0	05	0 0 0 0 0 1 0 1

Figure II-6 shows the photograph of the instruction cycles required for ADD M instruction. Two machine cycles, identified by two SYNC pulses, are required for the completion of the ADD M instruction. During the first machine cycle consisting of three states, the processor extracts from the memory the one byte instruction word addressed by its program counter. During the next machine cycle, the processor sends out as an address the contents of its H and L registers. The eight-bit data word is placed in a temporary register inside the 8080 CPU. This takes three states. During the seventh and final state, the contents of the temporary register are added to those of the accumulator. Thus, two machine cycles, consisting of seven states in all complete "ADD M" instruction cycle.

SHLD

During a "SHLD" instruction cycle, the contents of the processor's H and L registers are deposited in two sequentially adjacent

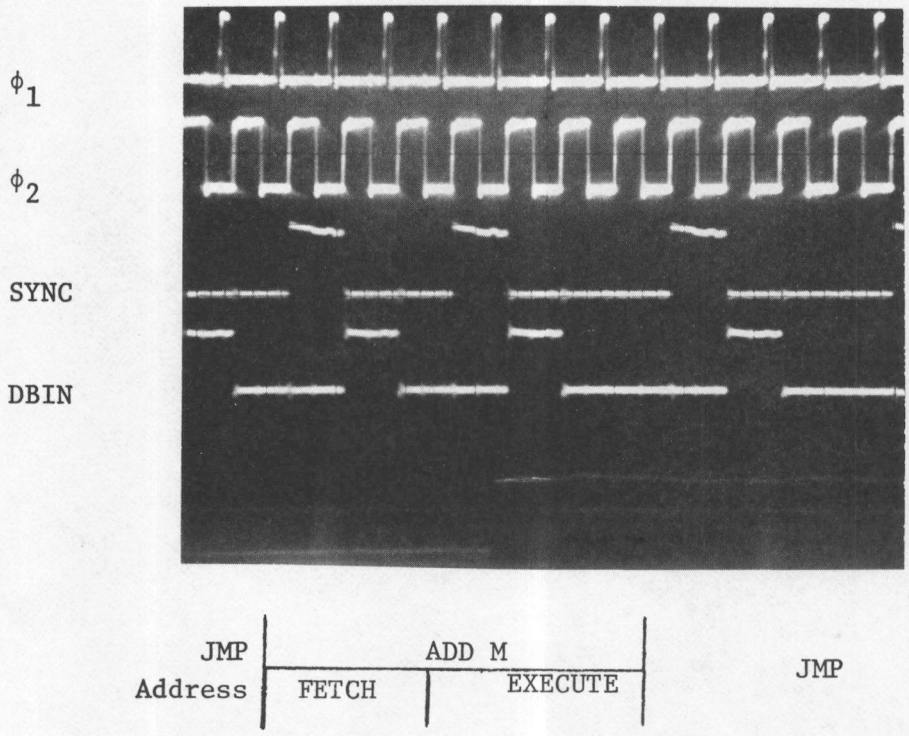


FIGURE II-6. "ADD M" instruction cycle.

memory locations immediately following the operation code byte. The program to observe the "SHLD" instruction cycles is shown below.

Memory Locations	Mnemonic Form	Bytes	Instruction Code
0 0 0 0 0 0 0 0	SHLD	3	0 0 1 0 0 0 1 0
0 0 0 0 0 0 0 1	06		0 0 0 0 0 1 1 0
0 0 0 0 0 0 1 0	06		0 0 0 0 0 1 1 0
0 0 0 0 0 0 1 1	JMP	3	1 1 0 0 0 0 1 1
0 0 0 0 0 1 0 0	00		0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1	00		0 0 0 0 0 0 0 0
	Total	6	

Figure II-7 shows the number of cycles required for the completion of the "SHLD" instruction. The following events occur: (1) a FETCH machine cycle consisting of four states. During the first three states of this machine cycle, the processor fetches the instruction. In the fourth state, the contents of the H and L registers are transferred to the processors temporary registers W and Z, respectively. (2) During the second cycle, the second byte of the instruction is extracted from the memory and placed in the processor's L register. This takes three states. (3) During the third machine cycle, consisting of four states, the third byte of the instruction is deposited in the H register. (4) During the first three states of the fourth machine cycle, the contents of the Z register (originally in the L register) are transferred to the memory location pointed to by the present contents of the H and L registers. During the fourth state, the pointer H and L

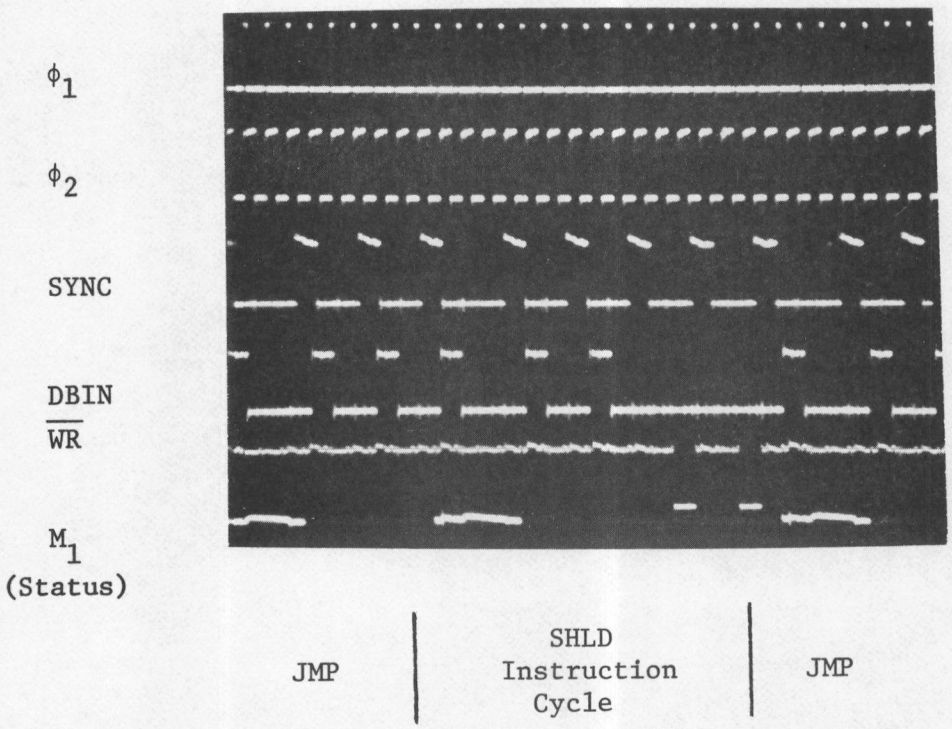


FIGURE II-7. The "SHLD" instruction cycle.

is incremented. (5) The final cycle consists of three states and is used to transfer the contents of the W register to the new memory location pointed to by the H and L registers.

Thus, the "SHLD" instruction cycle contains five machine cycles and takes seventeen states to execute.

DAD B

The "DAD" instruction adds the contents of the specified register pair to the contents of the register pair H and L. The program to observe the machine cycle requirements for the "DAD" instruction is shown below.

Memory Locations	Mnemonic Form	Bytes	Instruction Code Form
0 0 0 0 0 0 0 0	DAD B	1	0 0 0 0 1 0 0 1
0 0 0 0 0 0 0 1	JMP	3	1 1 0 0 0 0 1 1
0 0 0 0 0 0 1 0	00		0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1	00		0 0 0 0 0 0 0 0
	Total	4	

The DAD B instruction adds the contents of the register pair B and C to the register pair H and L. Figure II-8 shows the machine cycles for the "DAD B" instruction. During the first machine cycle, the instruction is fetched. This takes four states. The second machine cycle consisting of three states is used to add the lower order byte of H and L to the lower order byte of B and C. During the third and the last machine cycle, the higher order bytes of H and L are added to the higher order bytes of B and C. The result is stored into H and L

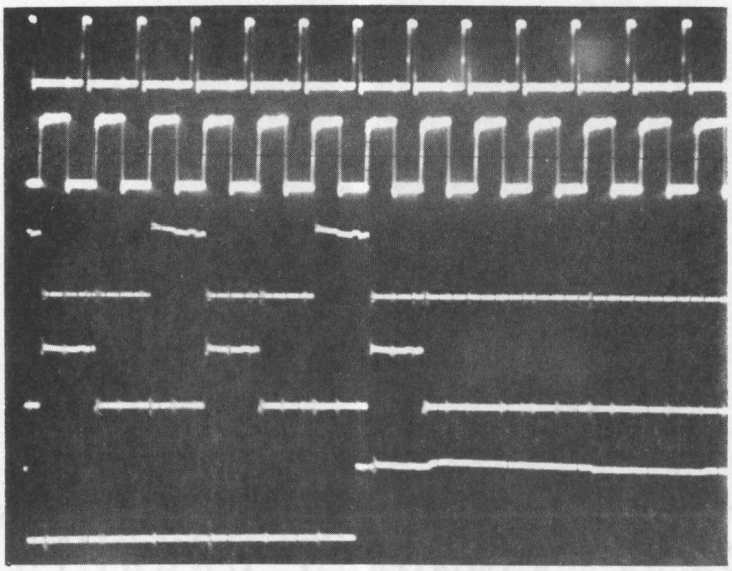


FIGURE II-8. "DAD B" instruction cycle.

registers. Note that only one SYNC signal is given out because the last two machine cycles are required for the internal register pair add only.

IN

"IN" instruction along with the address of the input device is used to obtain the data from the input device. Figure II-9 shows the machine cycles for the input instruction. The input instruction is repeated as shown in the following program.

Memory Locations	Mnemonie Forms	Bytes	Instruction Code
0 0 0 0 0 0 0 0	IN	2	1 1 0 1 1 0 1 1
0 0 0 0 0 0 0 1	00		0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0	JMP	3	1 1 0 0 0 0 1 1
0 0 0 0 0 0 1 1	00		0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0	00		0 0 0 0 0 0 0 0
	Total	5	

The first machine cycle consisting of four states fetches the instruction code from memory. The second machine cycle is also a memory read cycle during which the address of the input device (second byte of instruction) is extracted from memory. During the third machine cycle, the address of the input device is sent out on the address bus and the accumulator is loaded with the input data. Both the second and the third machine cycle takes three states. Thus, the input instruction consists of three machine cycles, having a total of ten states.

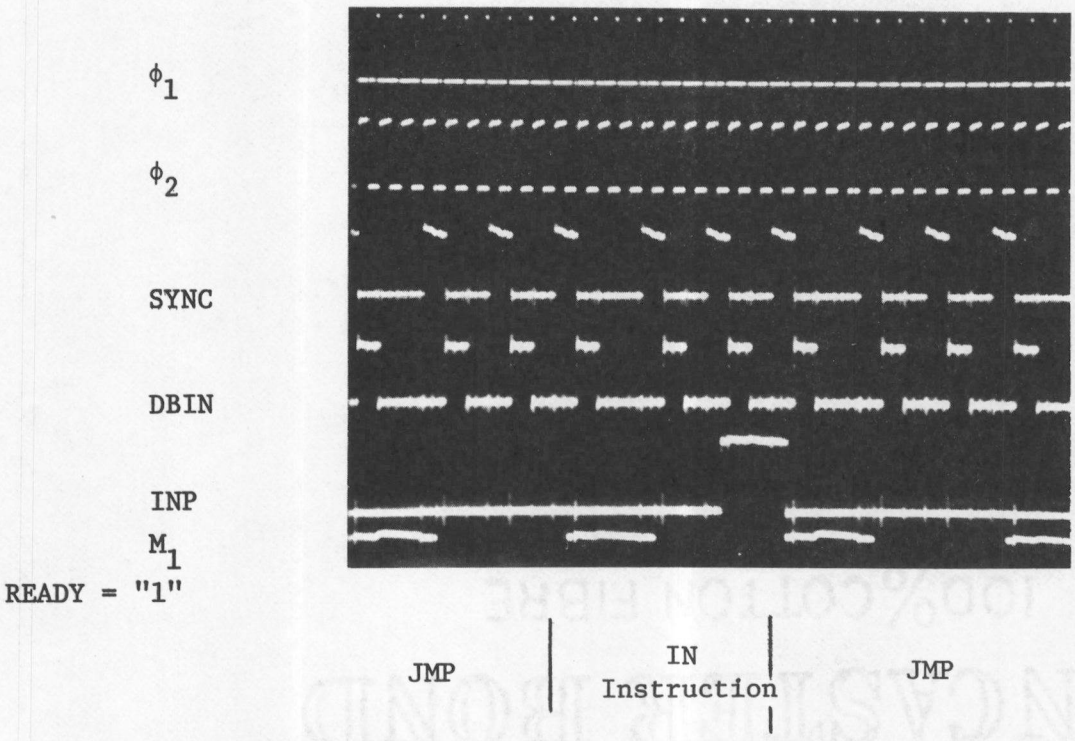


FIGURE II-9. "IN" instruction cycle.

OUT

The output instruction is similar to the input instruction except that during the third machine cycle, the data are written out from the accumulator to the output device as shown in Figure II-10. The program to monitor the waveforms could be written as follows:

Memory Locations	Mnemonic Form	Bytes	Instruction Code
0 0 0 0 0 0 0 0	OUT	2	1 1 0 1 0 0 1 1
0 0 0 0 0 0 0 1	00		0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0	JMP	3	1 1 0 0 0 0 1 1
0 0 0 0 0 0 1 1	00		0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0	00		0 0 0 0 0 0 0 0

IV. SUMMARY

There are nine types of machine cycles that may occur within an instruction cycle, though no one instruction cycle will consist of more than five machine cycles: (1) Fetch; (2) Memory Read; (3) Memory Write; (4) Stack Read; (5) Stack Write; (6) Input; (7) Output; (8) Interrupt; and (9) Halt. The machine cycles that actually do occur in a particular instruction cycle depend depend upon the kind of instruction. The first machine cycle in any instruction cycle is always a FETCH. Instruction state requirements range from a minimum of four states up to a maximum of eighteen states.

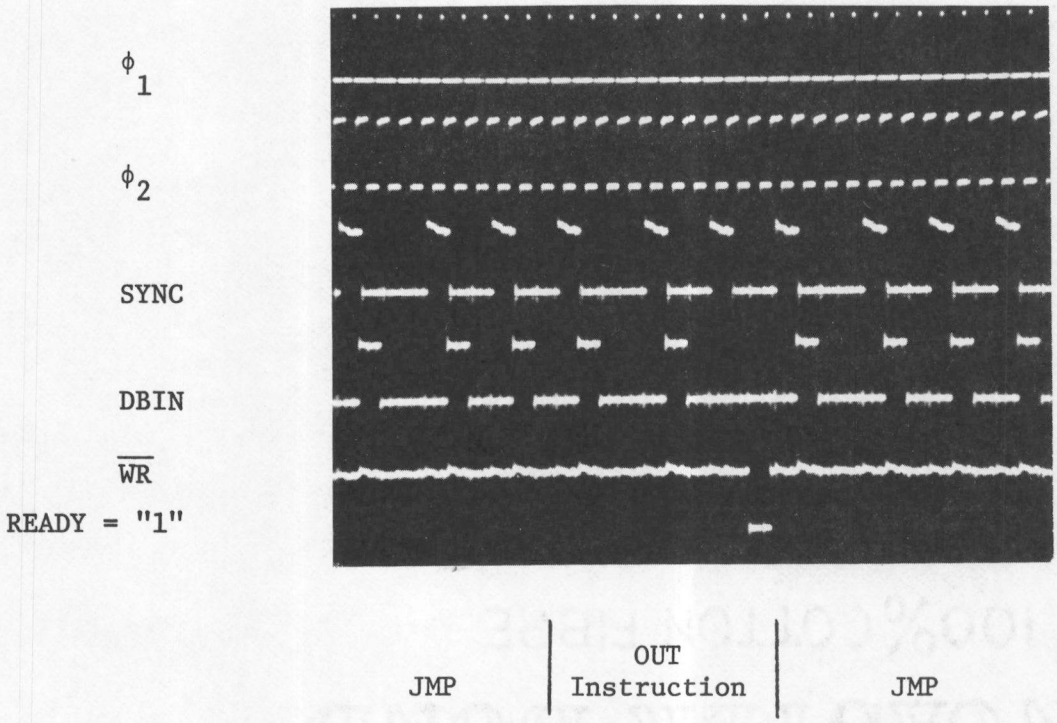


FIGURE II-10. "OUT" instruction cycle.

CHAPTER III

DEVELOPMENT OF MICROCOMPUTER SYSTEM

The 8080 is a complete parallel central processing unit for use in general purpose digital computer systems. It is fabricated on a single LSI chip using Intel's n-channel silicon gate MOS process.

It contains 16 address lines that provide capability of addressing 64 k-bytes of memory and up to 256 input and output ports. Also the processor has eight-bit bidirectional data bus to interface it to memory and the input and output ports. Figure III-1 shows how these lines are connected.

I. INTERFACING THE MEMORY AND INPUT/OUTPUT PORTS

In Figure III-1 the address and data lines are directly connected to the memories and Input/Output port logic. This cannot be done unless the microprocessor has an ability to drive a large number of gates. The 8080 can drive one TTL load; i.e., it can sink 1.6 ma when in logic "0" state and supply 40 μa when in logic "1" state. Figure III-1 does not show the complete microcomputer system. Some other circuits are needed to control the memory and to decode the address bus for memory expansion and input/output address. A larger drive current is required when the memories are connected in parallel for memory expansion. These considerations dictate the need to have bus drivers or buffers connected between the microprocessor and external circuits as shown in Figure III-2.

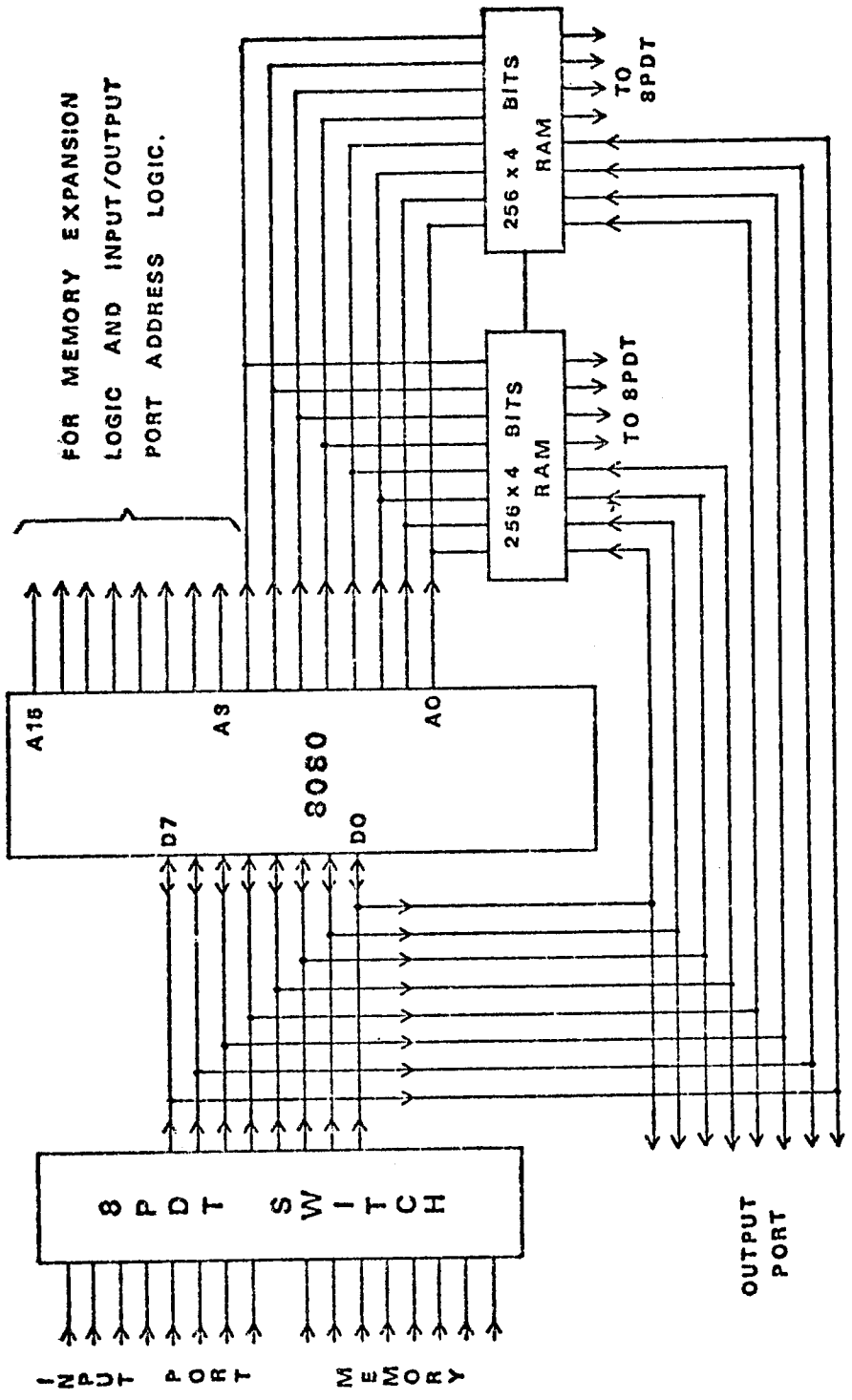


FIGURE III-1. Interfacing 8080 with memory and input/output port.

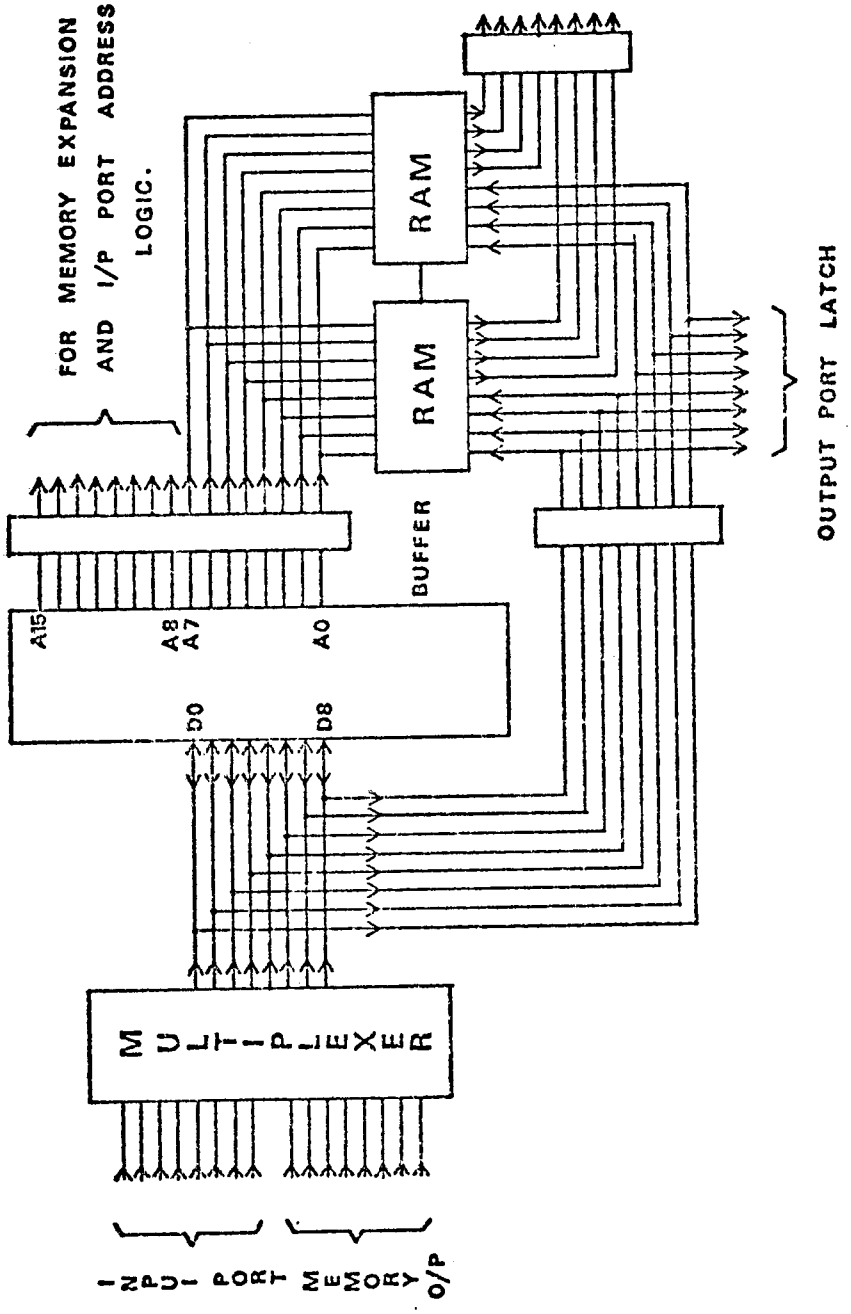


FIGURE III-2. Buffering the address and data lines.

In Figure III-2, the buffers used are 8095/8093 tri-state buffers. The control lines for the buffers for the address and data lines are connected to the ground which enables the buffers. These control lines could be controlled by HLDA output of the microprocessor for Direct Memory Access capability.

Outputs from the memory are also buffered and the control line for the buffer is controlled by MEMR (memory read) signal. Status signals are given out by the microprocessor on the data bus at the beginning of each machine cycle (during SYNE time). This information is latched at the beginning of each machine cycle using ϕ_1 and SYNC signals, so that it can be used any time during the rest of the cycle to control the flow of data. Table III-1 defines the status information.

II. STATUS INFORMATION DEFINITION

Symbols	Data Bus Bit	Definition
INTA	D_0	Acknowledge signal for interrupt request. Signal should be used to gate a restart instruction onto data bus when DBIN is active.
$\overline{WO}$	D_1	Indicates that the operation in the current machine cycle will be a WRITE memory or OUTPUT function ($\overline{WO}$ = 0). Otherwise, a READ memory or INPUT operation will be executed.

TABLE III-1

THE 8080 STATUS BIT DEFINITIONS

Data Bus Bit	Status Information	TYPE OF MACHINE CYCLE: Instruction Fetch	Memory Read	Memory Write	Stack Read	Stack Write	Input Read	Output Write	Interrupt Acknowledge	Halt Acknowledge	Interrupt Acknowledge While Halt
		STATUS WORD NUMBER: 1	2	3	4	5	6	7	8	9	10
D_0	INTA	0	0	0	0	0	0	0	1	0	1
D_1	$\overline{WO}$	1	1	0	1	0	1	0	1	1	1
D_2	STACK	0	0	0	1	1	0	0	0	0	0
D_3	HLTA	0	0	0	0	0	0	0	0	1	1
D_4	OUT	0	0	0	0	0	0	1	0	0	0
D_5	M_1	1	0	0	0	0	0	0	1	0	1
D_6	INP	0	0	0	0	0	1	0	0	0	0
D_7	MEMR	1	1	0	1	0	0	0	0	1	0

Symbols	Data Bus Bit	Definition
STACK	D_2	Indicates that the address bus holds the pushdown stack address from the stack pointer.
HLTA	D_3	Acknowledge signal for HALT instruction.
OUT	D_4	Indicates that the address bus contains the address of an output device and the data bus will contain the output data when $\overline{WR}$ is active.
M1	D_5	Provides a signal to indicate that the CPU is in the fetch cycle for the first byte of an instruction.
INP	D_6	Indicates that the address bus contains the address of an input device and the input data should be placed on the data bus when DBIN is active.
MEMR	D_7	Designates that the data bus will be used for memory read data.

There are ten types of machine cycles and each machine cycle has a distinct status word as shown in Table III-1. Each instruction in the 8080 requires from one to five machine or memory cycles for fetching and execution.

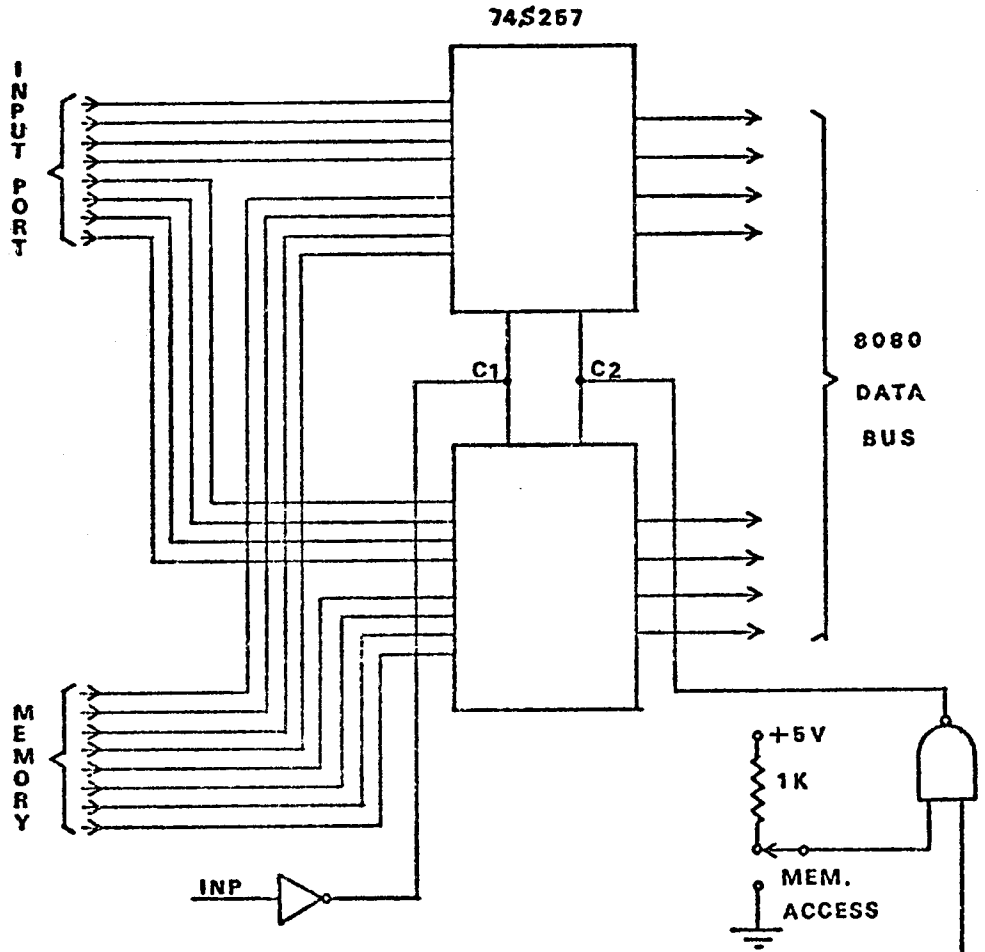
Status information and output signals should be gated in a proper way to generate control signals for the hardware external to the microprocessor. In Figure III-2, page 34, we need to develop signals to control data flow to and from the memory, signals for input multiplexer, and signals for output port latches.

III. LOGIC CIRCUIT DEVELOPMENT FOR INPUT MULTIPLEXER CONTROL LINES

The input multiplexer is used to gate the eight-bit data onto the 8080 data bus either from the input port or from the memory when the data bus is in the input mode. This means that the "INP" status signal can be used for the selection of the source of the data and "DBIN" signal should be used to gate these data onto the data bus at the proper time. Figure III-3 shows the logic circuit diagram for this purpose. In Figure III-3, RUN and single-step (S.S.) inputs to an "OR" gate are the "Q" outputs of "RUN" and "S.S." flip-flops used in front-panel control logic. The logic shows that when the "RUN" switch is in RUN mode or while single stepping, the MEM ACCESS switch should be in the down position. Otherwise, the MEM ACCESS switch should be in the up position in order to disconnect the 8080 data bus from the input port and memory output. The logical expressions for C_1 and C_2 are as follows:

$$C_1 = \overline{INP}$$

$$C_2 = \overline{[(RUN + SS)\ DBIN]\ \overline{MEM\ ACCESS}}$$



C_1: L = data from input port
H = data from memory

C_2: L = data gated onto data bus
H = output in high impedance state

FIGURE III-3. Input multiplexer control line logic.

$$= \overline{\overline{[(RUN + SS)\ DBIN]} + MEM\ ACCESS}$$

$$= [\overline{(\overline{RUN + SS}) + \overline{DBIN}}]\ \overline{MEM\ ACCESS}$$

IV. LOGIC CIRCUIT DEVELOPMENT FOR MEMORY CONTROL

Data are to be written into the memory when "$\overline{WR}$" signal is active. Also we want to write data into memory from the front panel switches when "LOAD" or "LOAD-NEXT" switch is pressed. Therefore, $\overline{WR}$ signal and a signal from "LOAD" or "LOAD-NEXT" logic are gated to generate logical "OR" of them. The "WRITE" signal is active low for 8101 type of memory chip which is accomplished by using the NOR gate instead of the OR gate.

$$R/\overline{W} = \overline{\overline{WR} + LOAD\ or\ LOAD\ NEXT}$$

when $R/\overline{W}$ signal is high, output of the memory can be read. Figure III-4 shows the logic circuit diagram for the $R/\overline{W}$ line of the memory.

V. LOGIC CIRCUIT FOR OUTPUT PORT LATCH CONTROL LINE

The output port is latched by the 2N 7495 (four-bit right-shift/Left-shift register) which can be used for parallel-in/parallel-out register operation. Two packages are required for each output port to latch eight-bit data. Data are transferred on negative transition of the clock pulse. Only two output ports are to be addresses, so that "A_8" address line is the only line required to address both of

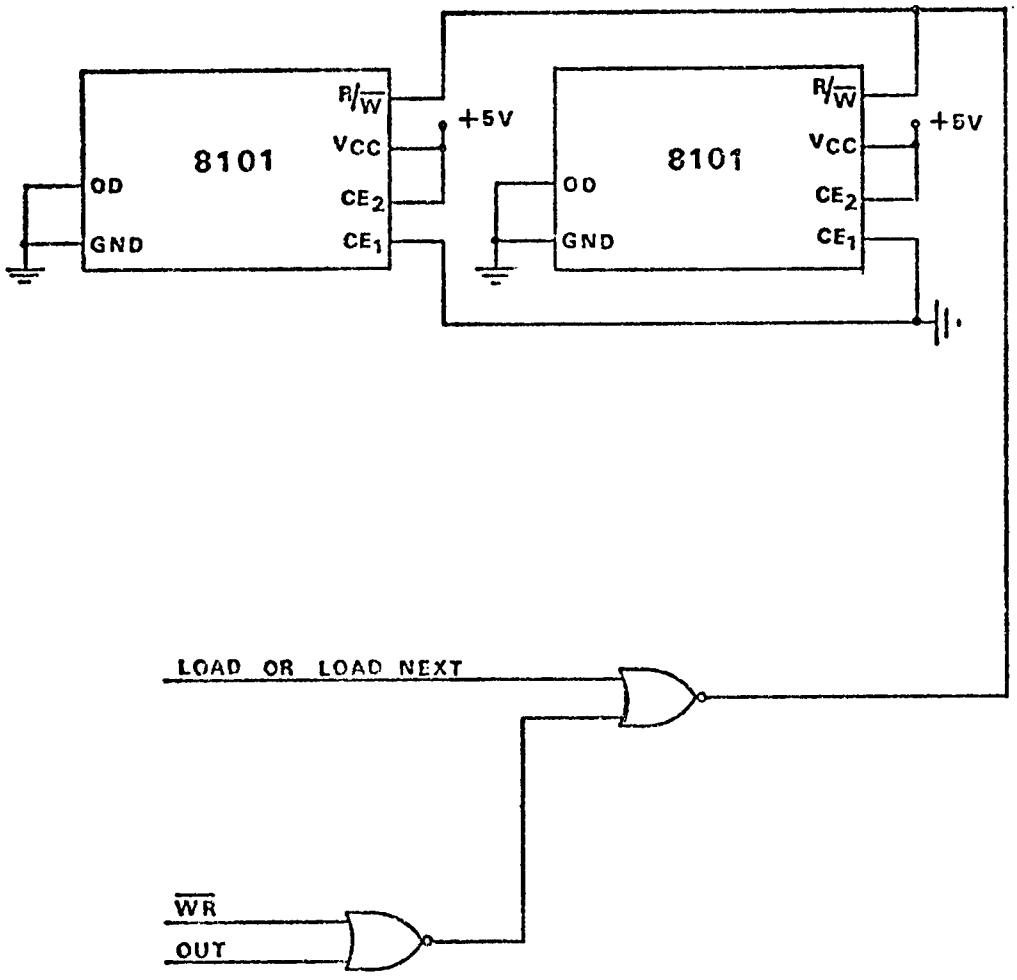


FIGURE III-4. R/$\overline{W}$ control logic.

them. When an output instruction is being executed, the microprocessor gives out $\overline{WR}$ signal, and OUT signal as status information. This information is gated to latch the data on one of the two output ports. Figure III-5 shows the gating arrangement to generate the signals for latches.

VI. FRONT PANEL CONTROL LOGIC OPERATION

The front panel consists of switches and LEDS for various functions. The main function of the front panel control logic is to control the READY line. It provides the operator with RUN/STOP and single-step control of the processor. In addition, it allows him to examine and modify the contents of any location in memory using the front panel switches. These functions can be performed by controlling the READY line. The READY line should be made high while any of the above mentioned functions are performed. So,

$$\text{READY} = \text{RUN} + \text{SS} + \text{EXAMINE} + \text{EXAMINE NEXT}$$

$$= \overline{(\overline{\text{RUN}})\ (\overline{\text{S.S.}})\ (\overline{\text{EXM}})\ (\overline{\text{EXM NEXT}})}$$

This signal is gated using ϕ_2 as shown in Figure III-6. All of the inputs to the four-input NAND gate are normally high. When any of the inputs go low, the output goes high and the ϕ_2 clock pulse transfers this information through the D-flip-flop to the READY line. So READY goes high and CPU proceeds.

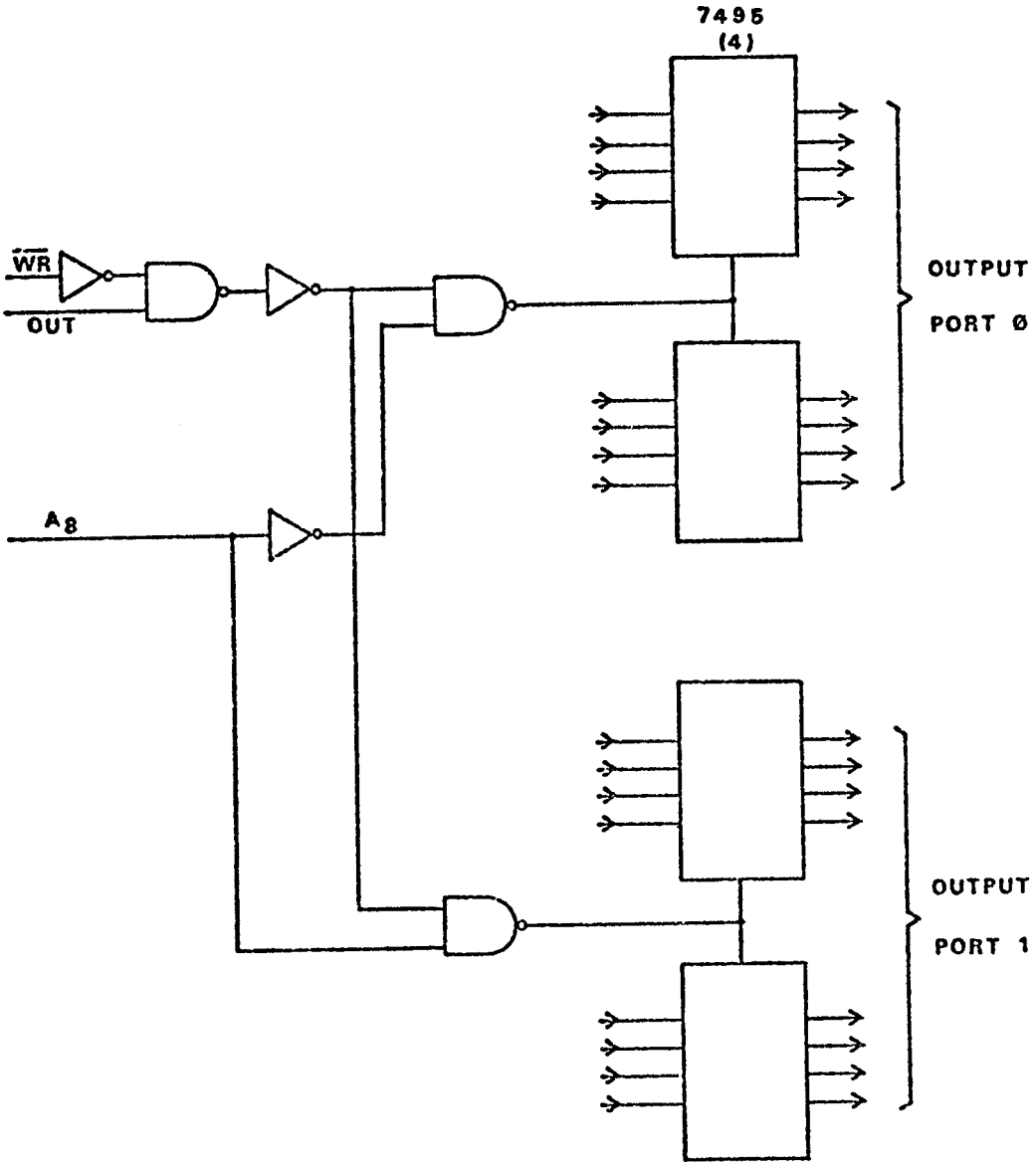
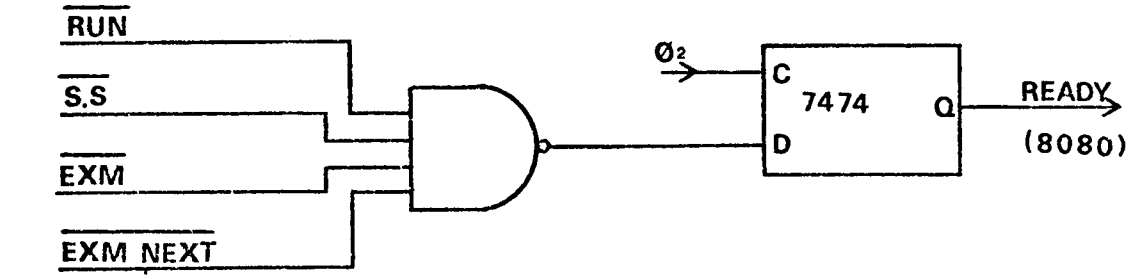


FIGURE III-5. Output port latch control logic.



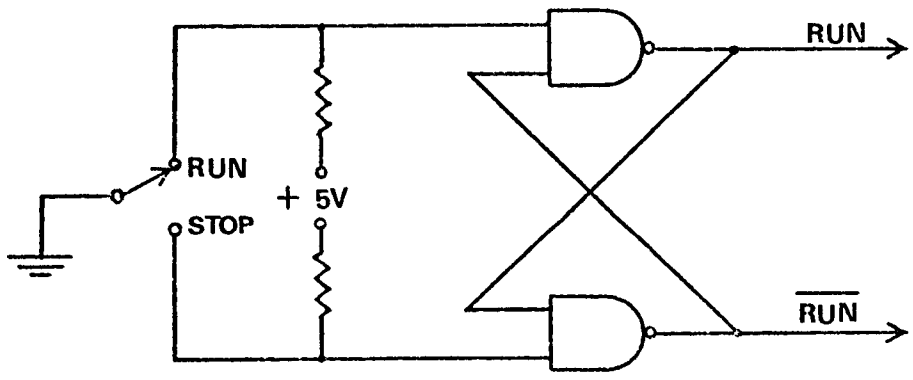


FIGURE III-6. Run/stop flip-flop and ready logic.

VII. RUN/STOP LOGIC

The logic circuit diagram for RUN/STOP switch is shown in Figure III-6. It is a simple R-S flip-flop consisting of two two-input NAND gates. The flip-flop eliminates the switching bounces. The $\overline{RUN}$ output is connected to four-input NAND gate of READY signal. When the switch is in RUN mode, this output goes low which makes READY high and the CPU proceeds. Otherwise, $\overline{RUN}$ goes high and the CPU is controlled by one of the other three inputs of the four-input NAND gate shown in Figure III-6.

VIII. SINGLE-STEP LOGIC

The machine should be in the stopped mode. The single-step logic circuit consists of a S-R flip-flop for debounce and a dual D-type flip-flop as shown in Figure III-7. Pressing the single-step switch will give a clock pulse to C_1 input. This makes $\overline{Q_1}$ low. $\overline{Q_1}$ going low will make C_1 go high during ϕ_1 phase which makes $\overline{Q_2}$ go low and hence READY goes high when ϕ_2 clock pulse occurs. The CPU proceeds and executes one machine cycle. SYNC at the beginning of the next cycle will reset the single-step flip-flop. This will pull the READY line low, and the CPU waits.

IX. EXAMINE LOGIC

The examine logic consists of a S-R flip-flop for switching debounce, a single shot, a two-bit counter, two sets of 7405's and some gating as shown in Figure III-8. When the examine switch is

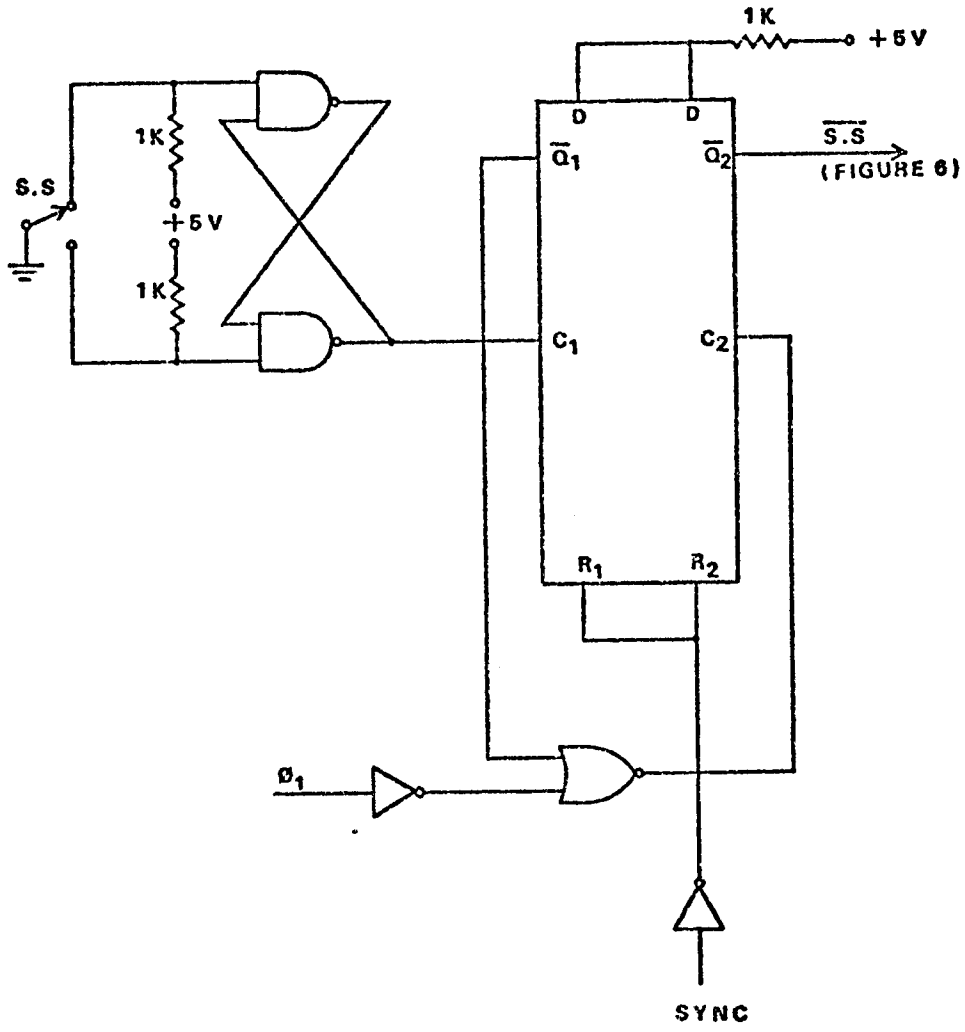


FIGURE III-7. Single-step logic.

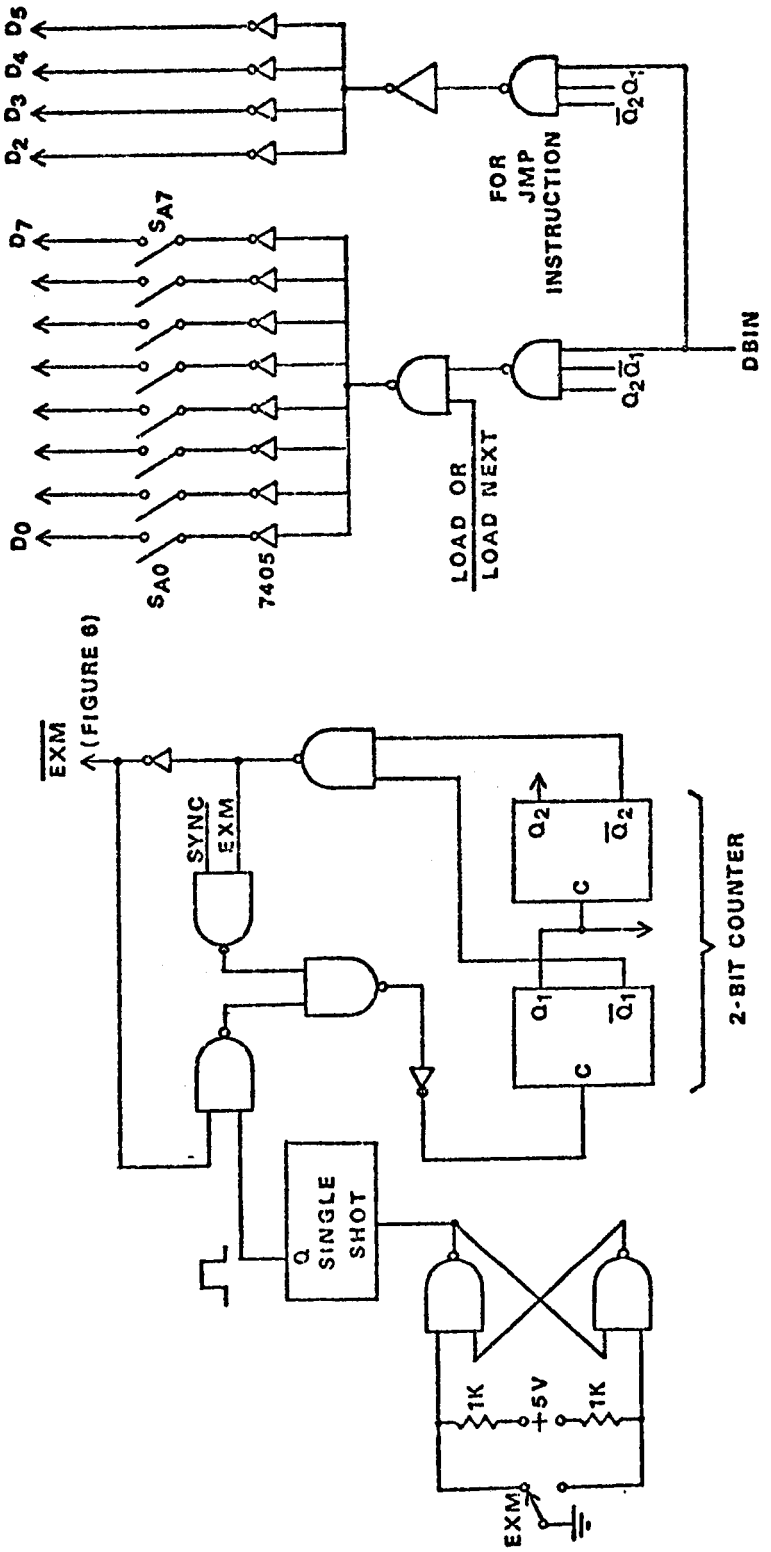


FIGURE III-8. Examine logic.

depressed the single shot will start the counter. During first, second, and third count $\overline{EXM}$ goes low which makes the READY line high and the CPU proceeds. During the first count a JUMP instruction is dumped onto the data bus when DBIN is active. This is done by pulling down data lines D_2, D_3, D_4, and D_5 which puts 1 1 0 0 0 0 1 1, the machine code for the JMP instruction, on the data bus.

The next clock pulses for the counter are derived from the SYNC pulse. During the beginning of the next machine cycle the counter will have a count of two. During this cycle the CPU is waiting for the first byte of the JMP instruction. So the count of two is used to strobe the setting on the data switches SA_0 through SA_7 onto the data bus.

The SYNC pulse at the beginning of the third machine cycle, during which cycle the CPU will be waiting for the second byte of JMP instruction, will clock the counter to count three. Nothing is done during this time as we have to address only 256 words of memory. So the CPU will see all 1's on the data bus and a JMP to the address set on the switches SA_0 through SA_7 will be executed by the CPU, allowing the examination of the contents of that particular memory location.

SYNC pulse at the beginning of the fourth cycle will reset the counter and pull the EXM line low, which in turn will pull the READY line low and the CPU will wait.

X. EXAMINE-NEXT

Depressing the EXAMINE-NEXT switch will increment the present memory address and thus allows the operator to increment and examine the memory locations. This is done by strobing a NOP instruction on

the data bus, which causes the processor to increment the program counter. Circuit for the EXAMINE-NEXT logic is shown in Figure III-9. When the switch is depressed a flip-flop is set, the READY line goes high and eight 7405's are activated, thereby dumping 0 0 0 0 0 0 0 0 on the data bus which is the mnemonic for the NOP instruction. SYNC at the beginning of the next cycle resets the flip-flop, thus pulling the READY line low and CPU waits.

XI. LOGIC CIRCUIT FOR LOAD AND LOAD-NEXT FUNCTIONS

Load switch allows the operator to load the data set on the address/data switches into the memory location addressed by the contents of the front panel LEDS. LOAD-NEXT switch allows the operator to load the data set on the data/address switches into the next memory location. This is done by first dumping the NOP instruction onto the data bus and then strobing the data set on the switches onto the data bus. The logic to strobe NOP instruction is shown in Figure III-9 and the logic to strobe the data is shown in Figure III-10. Also to load the data into the memory, a R/$\overline{W}$ signal needs to be applied to the memory. Logic circuit for this purpose is shown in Figure III-4, page 41.

XII. CLOCK CIRCUIT FOR THE MICROCOMPUTER

The 8080 is driven by a two-phase oscillator. The maximum allowable clock frequency is 2.08 MHZ and the minimum is 500 KHZ. All processing activities are referred to the period of this clock.

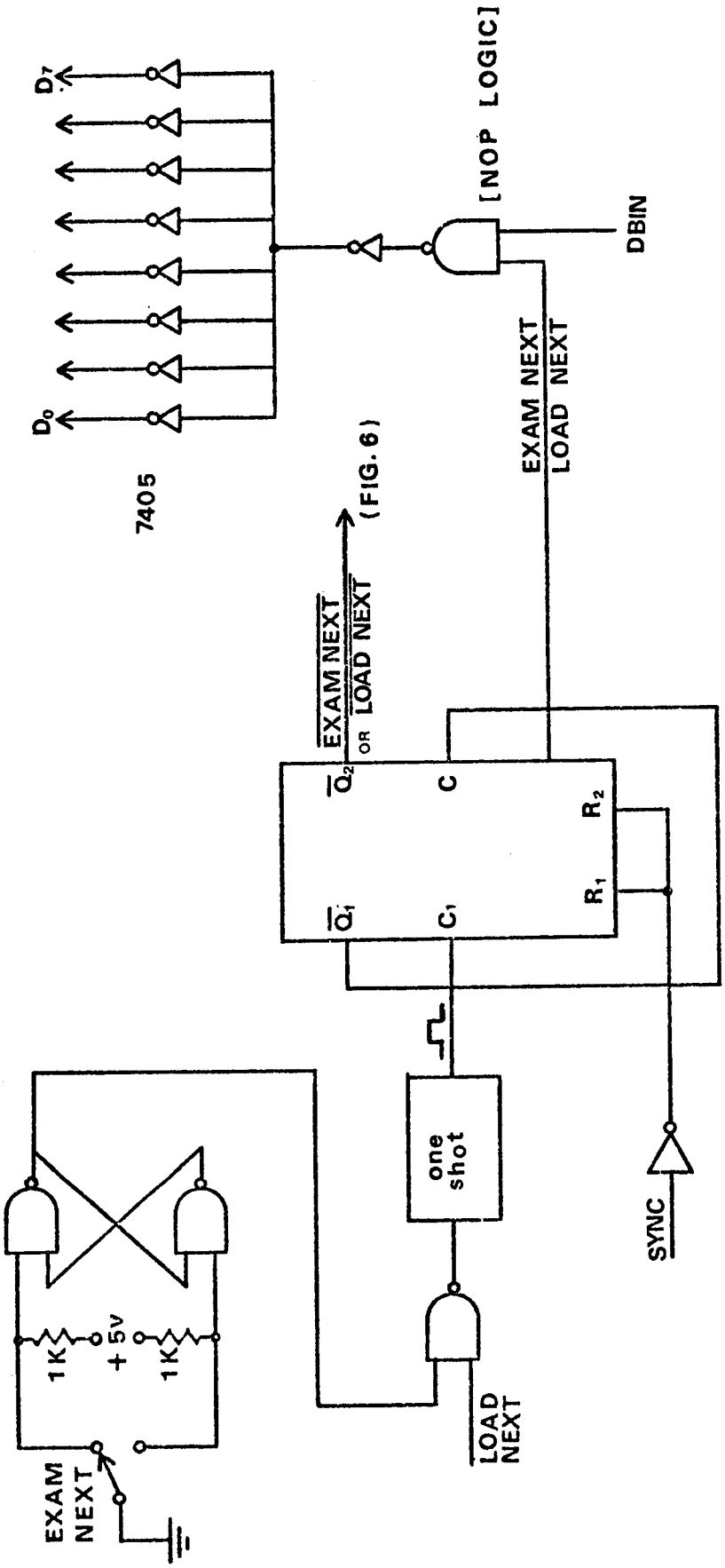


FIGURE III-9. Examine-next, load-next logic.

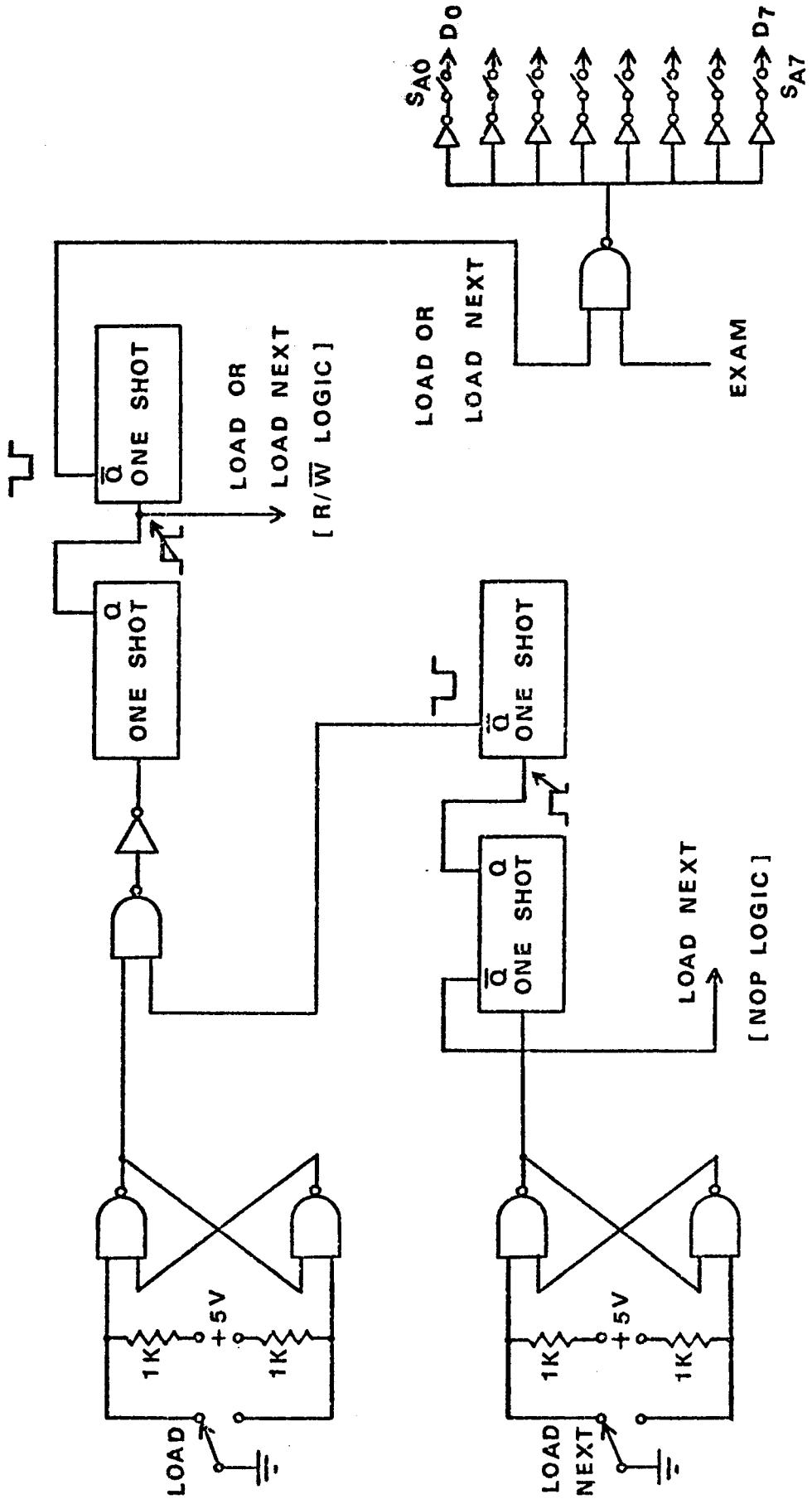


FIGURE III-10. Load, load-next logic.

The clock in our system employs a TTL oscillator with a 10.00 MHZ crystal as the feedback element. The oscillator section is followed by a "divide by sixteen" counter and the associated decode gating to give ϕ_1 and ϕ_2 phases of the clock system at a frequency of G25 KHZ. These TTL level phases are used for external timing purposes. The 8080 requires a 12-volt swing on the clock. This is accomplished by using level shifters. The circuit diagram for the clock system is shown in Figure III-11. The complete system is shown in Figures III-12, III-13, III-14, and III-15.

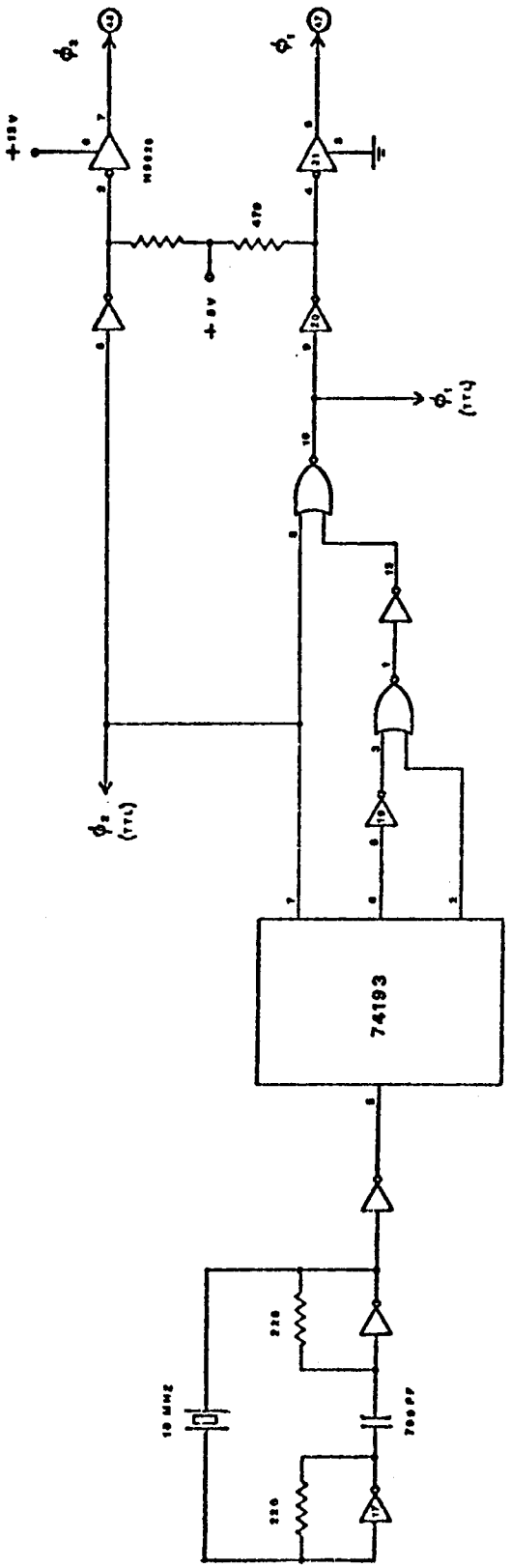


FIGURE III-11. System clock.

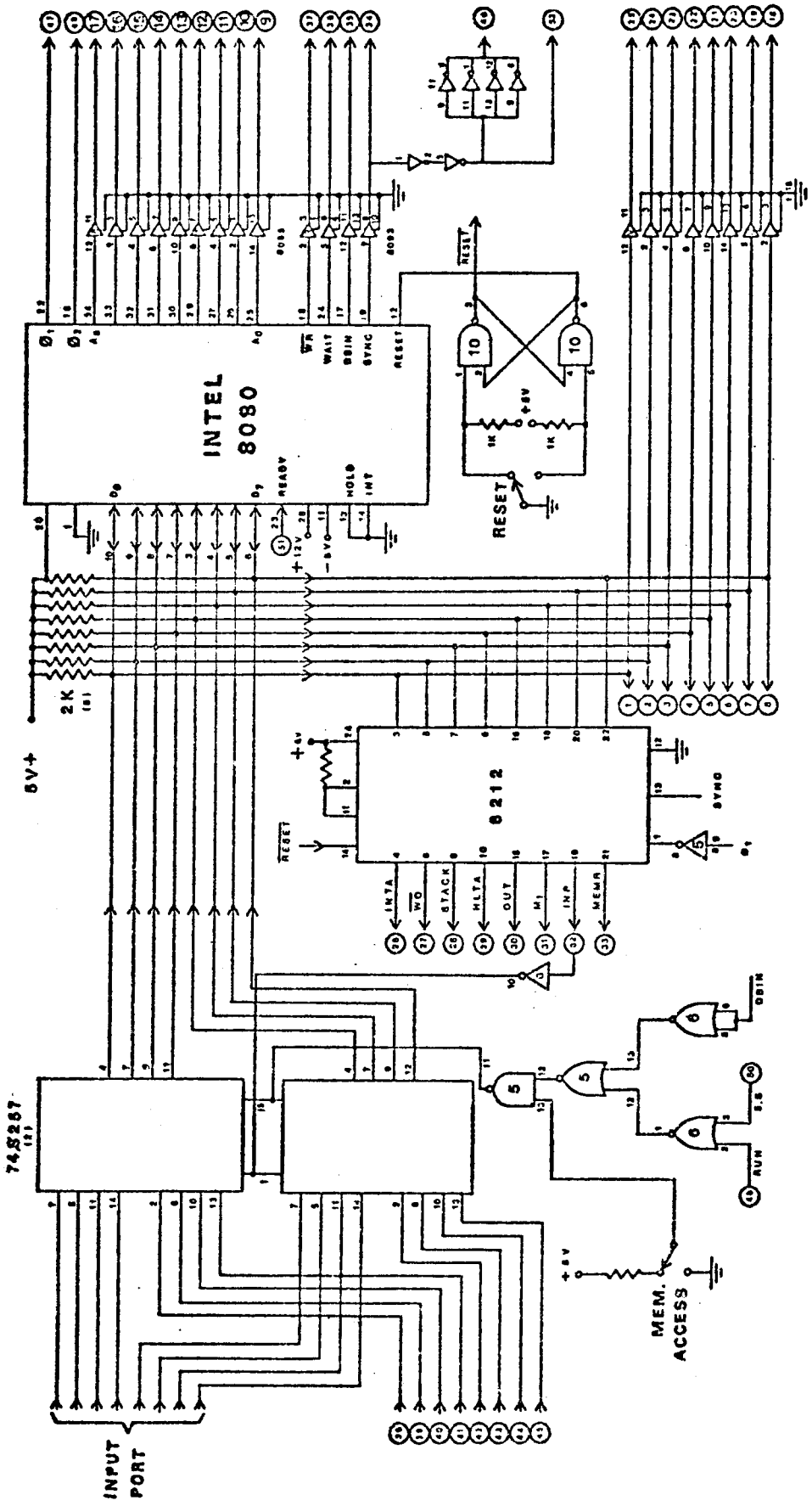


FIGURE III-12. Central processing unit (CPU).

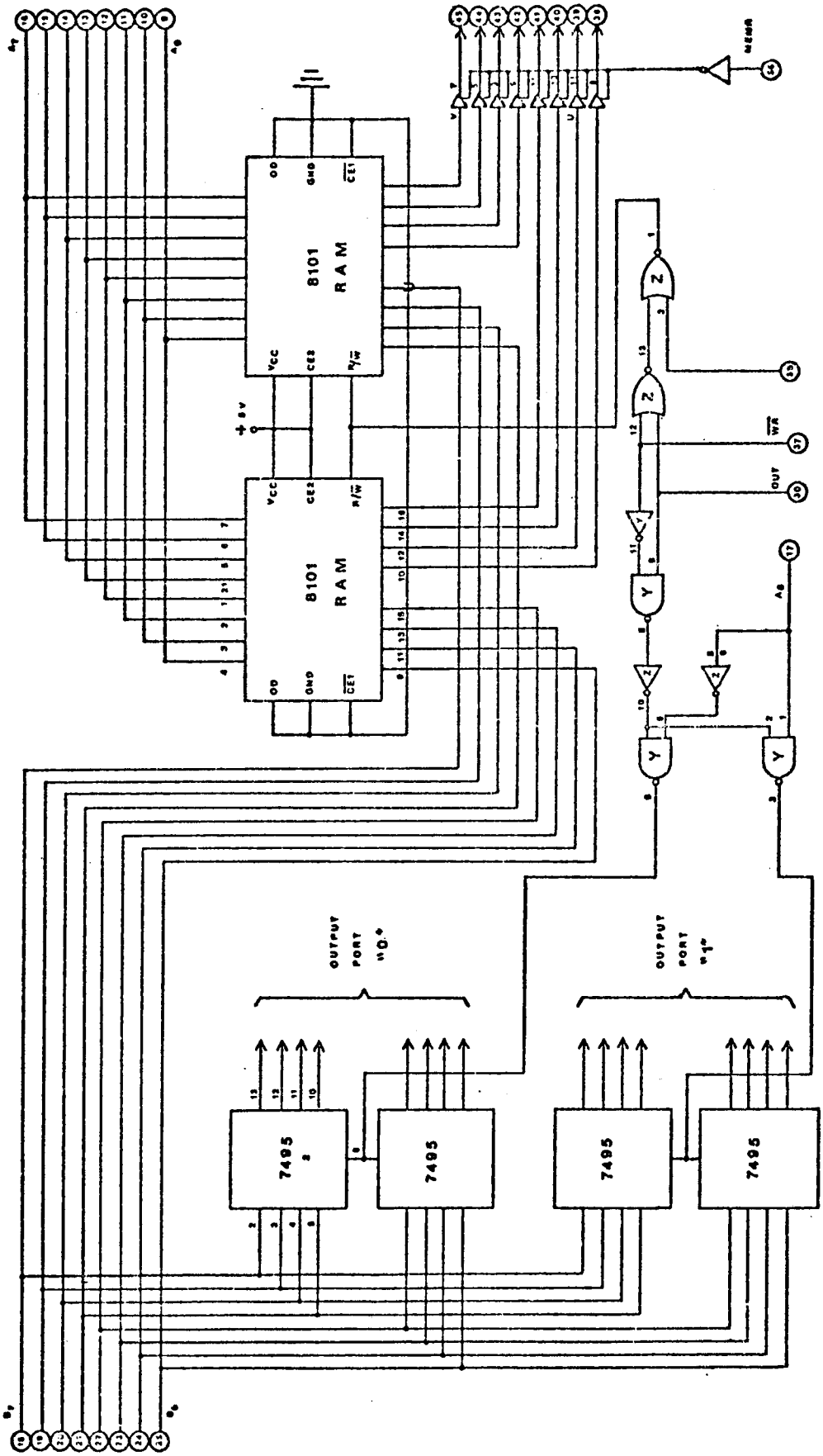


FIGURE III-13. Memory and output ports.

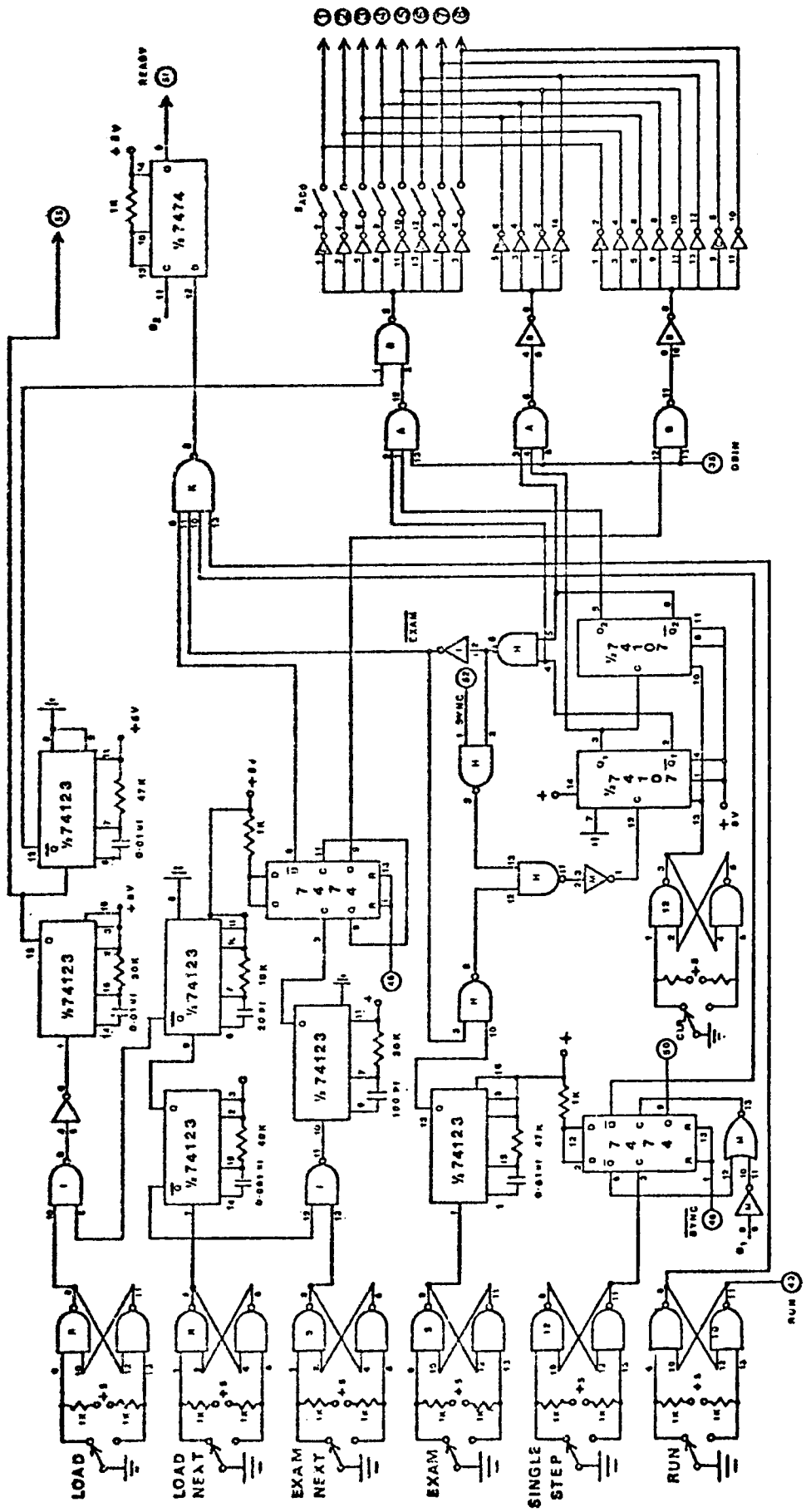


FIGURE III-14. Front panel control.

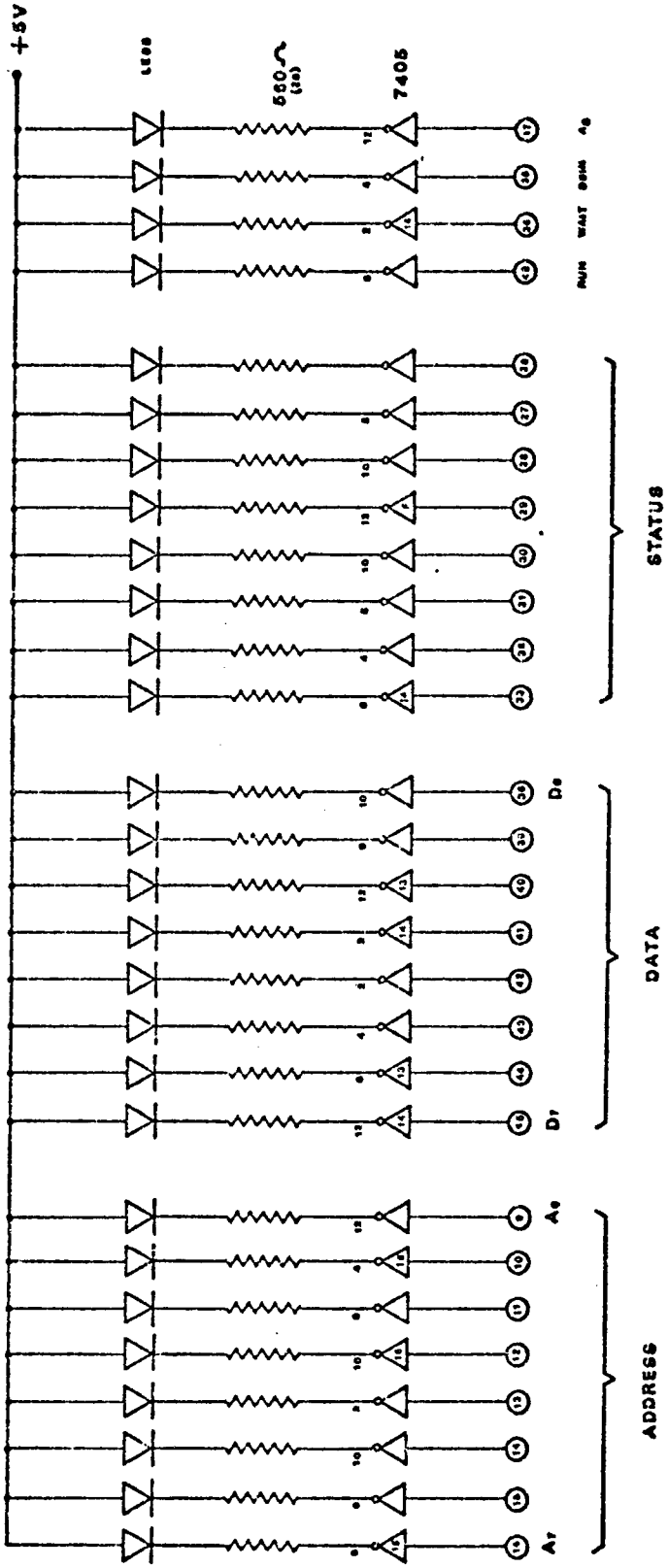


FIGURE III-15. Light-emitting diode display.

CHAPTER IV

FRONT PANEL OPERATION

Using the microcomputer requires some means of leading the program into memory. There are several ways to do this. No matter what technique is used, ultimately the program has to be converted into a machine language. Machine language is the code equivalent of the mnemonic form of an instruction. Each instruction has a specific code (in terms of 0's and 1's). The program can be loaded in memory directly in the form of machine language by using switches and by generating proper signals with the help of front panel switches. This method of loading the instructions in memory is adopted for the microcomputer that has been developed for this thesis.

Figure IV-1 shows arrangement of LEDS and switches on the front panel. Eight LEDS on the right hand top corner display the contents of the address bus and the eight LEDS below them are used to show the contents of memory location whose address is displayed by the address bus LEDS. The eight switches, designated as SAD_7-SAD_0 are used to load the instructions in the memory with the help of control switches. The status information is displayed by the status information LEDS, located at the center top part of the front panel. The states of DBIN, WAIT, READY and A_8 are also displayed by LEDS located at the right side of the front panel. The function of each control switch is explained below.

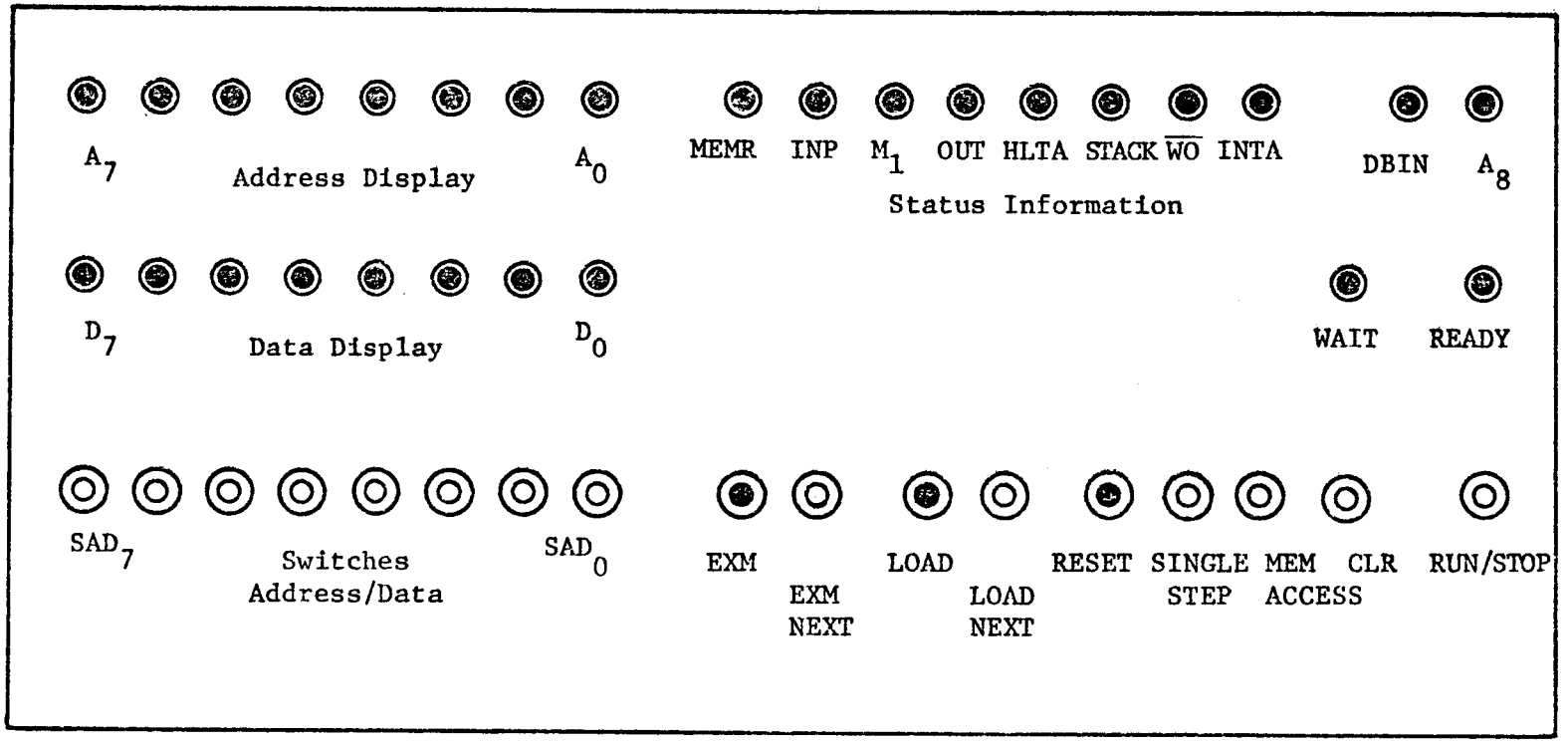


FIGURE IV-1. Front panel of the microcomputer.

I. EXAMINE (EXM)

The EXAMINE (EXM) switch is used to verify the contents of any memory location. The address of the memory location whose contents are to be examined is set on the switches SAD_7 - SAD_0 ("1" corresponds to the UP position and "0" corresponds to the DOWN position of the switch). Then the EXM switch is depressed. Both the contents and the address of memory location being examined are displayed by the corresponding LEDS.

II. EXAMINE-NEXT

Depressing the EXAMINE-NEXT (EXM NEXT) switch allows the operator to increment and examine the memory locations.

III. LOAD

Step 1: Set the address of memory location to be loaded with an instruction or data on the address/data switches SAD_7 - SAD_0 and depress the EXM switch. The address and the contents of the specified memory location will be displayed by LEDS.

Step 2: Set the data or instruction code on the switches SAD_7 - SAD_0 and depress the LOAD switch. The contents of the switches SAD_7-SAD_0 will be loaded into the location addressed by the address display LEDS; the data will be displayed by the data display LEDS.

IV. LOAD-NEXT

When this switch is depressed, the CPU increments memory address and the data set on the switches SAD7 - SAD0 are loaded into this new memory location. This switch allows the operator to load the program into successive memory locations.

V. RESET

When RESET switch is depressed, the contents of the program counter are cleared to zero and hence the CPU executes the program starting from the zero memory location.

VI. SINGLE-STEP (S.S.)

The single-step switch allows the operator to go through a single cycle of an instruction. The S.S. switch helps to debug the program.

VII. CLEAR (CLR)

Set this switch to first UP and then DOWN position after the power is applied to the computer. This clears the counter used in front panel control logic.

VIII. MEMORY ACCESS (MEM. ACCE.)

The memory access switch should be in "UP" position when any of the above mentioned functions are to be performed. This switch disables the input data bus from the input port and memory output.

IX. RUN/STOP

The RUN/STOP switch should be in STOP position (DOWN position) while using any of the other switches. This stops the CPU. To run the computer, first take the MEM. ACCE. switch to "DOWN" position and then take the RUN/STOP switch to "RUN" position.

X. POWER SUPPLY CONNECTIONS

The computer requires three power supplies: + 5 volts, + 12 volts, and - 5 volts. The terminals to connect the power supplies are taken out on the back side of the computer.

XI. CONNECTOR

The input port, output ports, status signals, and other signals are accessible through a fifty-pin connector placed on the left hand side of the computer body. Figure IV-2 shows the connector pin assignments.

XII. SUMMARY

This chapter describes the functions of various switches and LEDS on the front panel and the arrangement for the power supply connections. The connector pin assignments are shown in Figure IV-2.

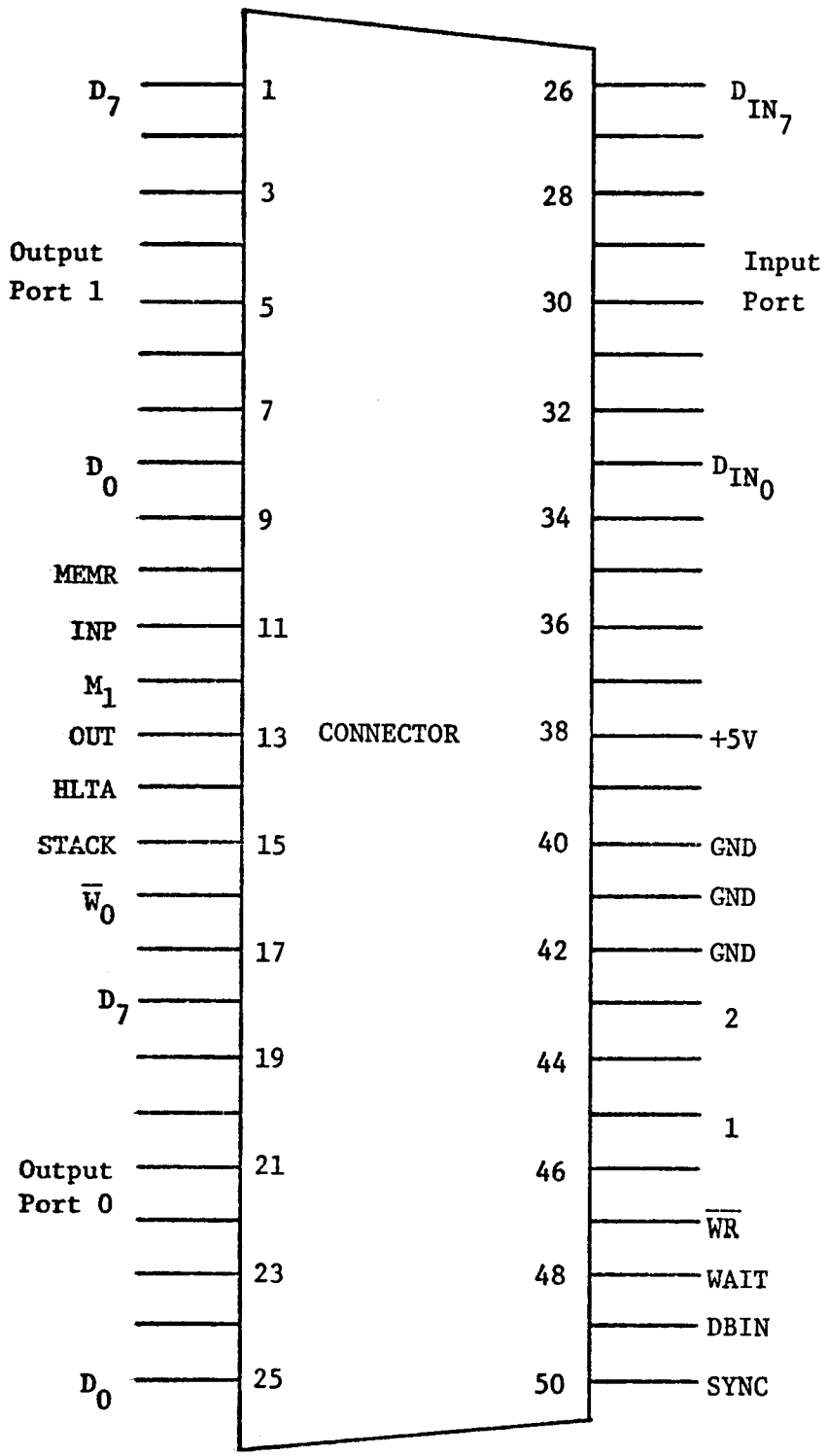


FIGURE IV-2. Connector pin assignments.

CHAPTER V

APPLICATION

Before the introduction of microprocessors, logic systems used discrete and random logic to perform the desired applications. Now, with the use of microprocessors, the applications are performed with a much lower package count and a resultant lowering of the costs.

In this chapter an application is shown of a microcomputer dedicated to perform a single task. This application allows the microcomputer to autorange a frequency counter to more nearly automate frequency measurements.

I. AUTORANGING A FREQUENCY COUNTER

A Heathkit Model IB-1100 Frequency Counter was used in this application. The counter was interfaced with the microcomputer so that the microcomputer automatically controls the range setting of the frequency counter. The block diagram of the counter is shown in Figure V-1. The counter works as follows.

The input frequency is converted to square waves using a Schmitt trigger and is applied to the clock input of the decade counting unit. The counting unit is enabled and disabled by controlling the J-K inputs of the counting unit through the range-select switch. Thus the DCU (decade counting unit) is enabled for certain periods of time depending upon the position of the range selection switch. At the end of the counting period, the contents of the counter are latched by

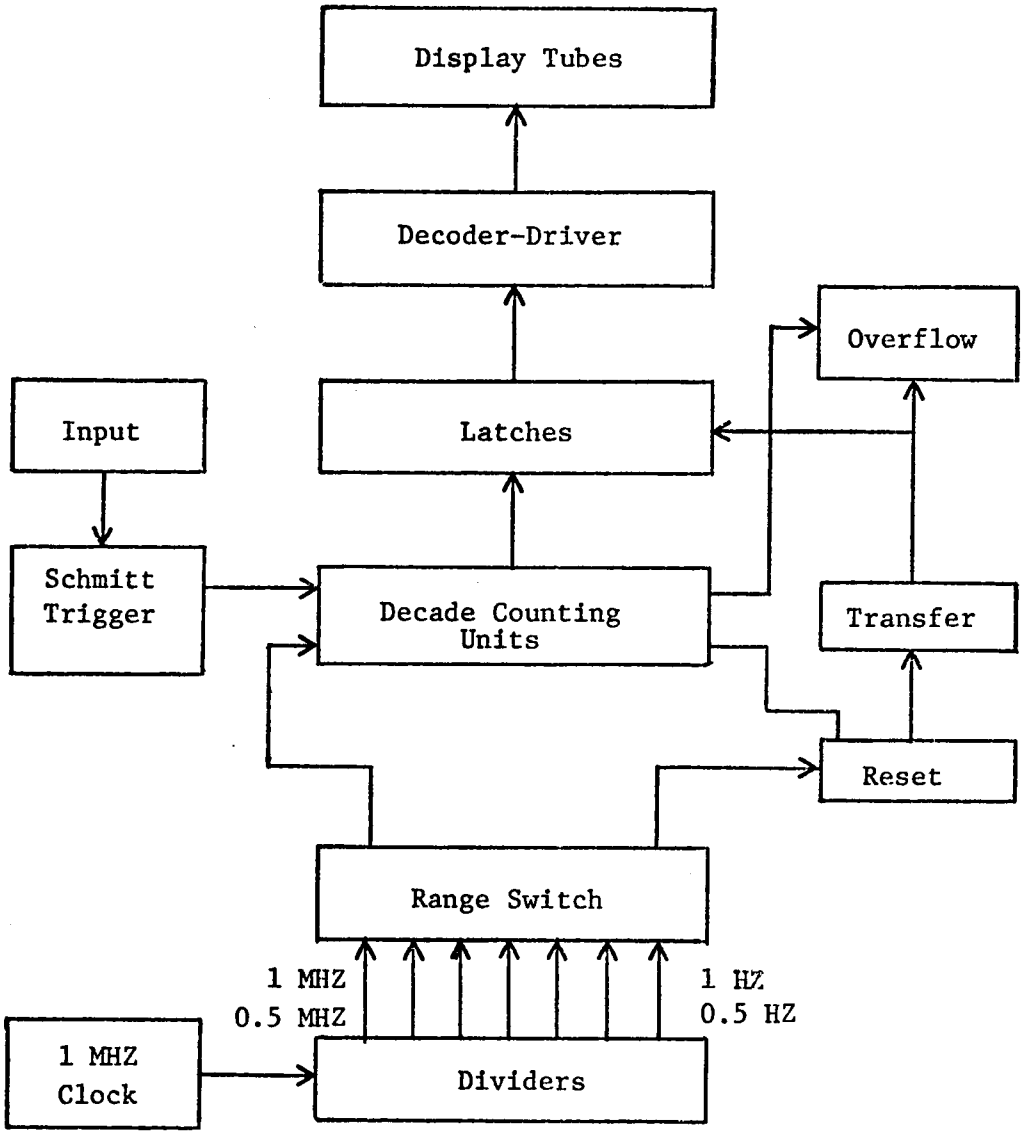


FIGURE V-1. Frequency counter block diagram.

generating a transfer signal. The contents of the latches are decoded and displayed by display tubes. If the frequency being measured exceeds the range selected by the switch, an overflow signal is generated and overflow light comes on. This tells the operator to change the position of the frequency switch to a higher range. Before the beginning of the next counting period a reset signal resets all the counters and the counters are updated during the next count period. This procedure is continuously repeated.

To convert the counter from manual ranging to autoranging, the two-range switch is replaced by the 8-to-1 multiplexers. The address lines of the multiplexers are connected to the output port 0 (D_2, D_1, D_0) of the microcomputer. The range is decided by the contents of these address lines. The multiplying factor is displayed on a seven-segment display through output port 0 (D_5, D_4, D_3). The contents of the counter display tubes are to be multiplied by ten to the power of the content of the seven-segment display. The result is in HZ. The circuit to interface the counter is shown in Figure V-2.

The computer monitors the overflow signal through an input port and gives out the proper bits on the output port 0 in order to change the range and display the multiplying factor. Other signals from the counter are properly gated in order to increase or decrease the range as the frequency being measured is increased or decreased.

Following considerations are given while developing the hardware and software for autoranging the counter.

(1) When an overflow occurs the address of the range-select multiplexures is incremented by one.

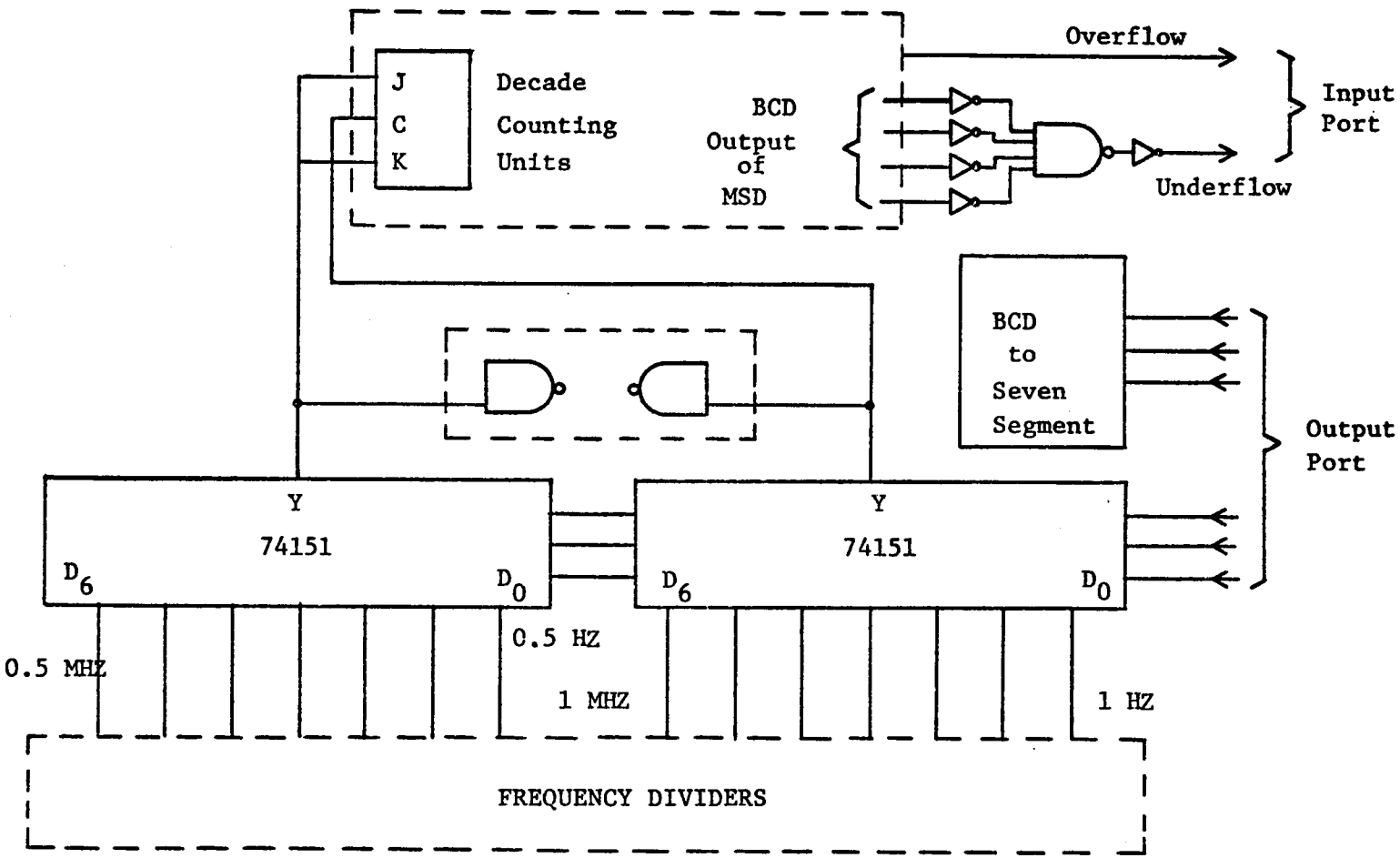


FIGURE V-2. Interface circuit for autoranging the frequency counter.

NOTE: Dotted blocks show circuit inside the counter.

(2) The contents of the seven-segment display are increased by one.

(3) When the frequency is decreased, so that it can be displayed by the lower range, the address of the multiplexer and the content of the seven-segment display is decremented by one. This can be done by monitoring the binary coded decimal (BCD) outputs of the most significant digit of the counter. When an underflow occurs the most significant digit goes from decimal 9 to decimal 0. The BCD output of this digit is decoded and used to monitor the underflow as shown in Figure V-2.

(4) The underflow is to be monitored after the first overflow.

The input and output port pin assignments are shown in Figure V-3.

The program for autoranging the counter is shown below.

```
START:      MVI     C, 00       ; Set C as a range selector.
STARC:      MVI     D, FE       ; Set D as a counter.
LOOP3:      MVI     B, 00       ; Set B as a counter.
LOOP2:      MVI     A, 00       ; Set A as a counter.
LOOP1:      INR     A           ; Jump to LOOP1
            JNZ     LOOP1       ; 256 times.
            INR     B           ; Jump to LOOP2
            JNZ     LOOP2       ; 256 times.
            INR     D           ; Repeat from LOOP3
            JNZ     LOOP3       ; once more.
```

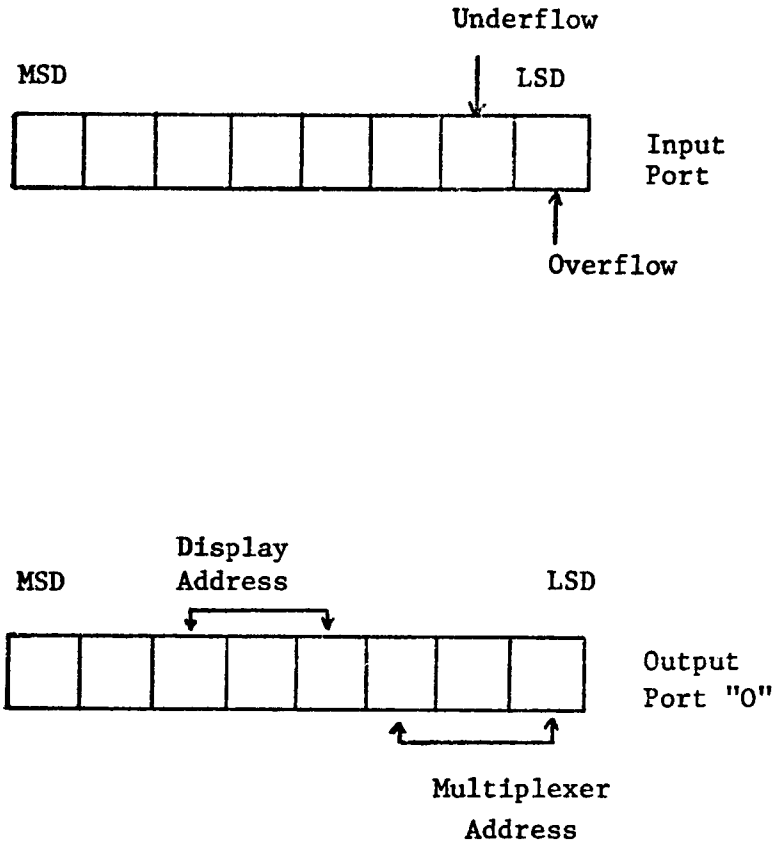


FIGURE V-3. Input/output port pin assignments.

```
        IN      0           ; Take input.
        ANI     01          ; Logical AND.
        CPI     01
        JZ      OVFLW       ; Jump if overflow.
        MOV     A, C        ; Do not check.
        CPI     00          ; For underflow unless
        JZ      STAR1       ; There is first overflow.
        IN      0
        ANI     02          ; Check for
        CPI     02          ; Underflow.
        JZ      UNFLW       ; Jump if underflow.
        JMP     STARC
STAR1:  OUT     0           ; Display zero.
        JMP     STARC
OVFLW:  INR     C           ; Increase the range.
        MOV     A, C
        CPI     07          ; To check if input
        JZ      START       ; Frequency is too high.
OVFLW1: MOV     A, C
        RLC
        RLC
        RLC
        ORA     C           ; Multiplexer address
        OUT     0           ; And display contents
        JMP     STARC         on output port 0.
```

```
UNFLW:      DCR     C               ; Decrease the range
            JMP     OVFLW1
```

The counter used has only two ranges. Although this counter has five digits, only three of them were used for the application because the instrument's owner had previously deactivated two of the digits. In this application we have implemented the counter with three digits for six ranges; by using an 8-to-1 multiplexer, as many as eight ranges could be obtained. This greatly improves the resolution of the frequency measurement when all five digits are used.

II. CONCLUSION

An application of the microcomputer to autorange a frequency counter illustrates how the addition of a dedicated computer can improve the usefulness of the instrument.

CHAPTER VI

CONCLUSION

Many manufacturers are currently designing microprocessors; the main use of these microprocessors is in the design of microcomputers. The microprocessors in the very near future will be very low cost and it will be predicted to include them to control even low-cost test instrumentation. In this thesis a microcomputer using the Intel-8080 microprocessor has been designed and interfaced to a frequency counter to show how the microcomputer can perform delicate functions.

Future developments of microcomputers are being concentrated on speed and device density. With increased speed and an increased number of instructions a microcomputer can perform many functions which previously had been accomplished with minicomputers. As the speed and device density increase and the cost falls, many new applications will emerge.

LIST OF REFERENCES

LIST OF REFERENCES

1. Shultz, Gaymond W., Holt, Raymond M., and McFarland, Harold, Jr. "A Guide to Using LSI Microprocessors," Computer, June, 1973, 13-19.

2. Gladstone, Bruce. "Designing with Microprocessors Instead of Wired Logic Asks More of Designers," Electronics, October 11, 1973, n.p.

3. Lewis, Donald R., and Siena, Ralph W. "Microprocessor or Random Logic," Electronic Design, 18 (September 1, 1973), 106-110.

4. "Intel 8080 Microcomputer Systems Users Manual," Intel Corporation, September, 1975.

APPENDICES

APPENDIX A

INSTRUCTION SET OF THE INTEL 8080 MICROPROCESSOR

Instruction Mnemonic	Description
MOV r_1, r_2	Move the contents of R_2 to R_1.
MOV M, r	Move the contents of register r to the memory location addressed by the contents of register H and L.
MOV r, M	Move the contents of the memory location addressed by the contents of registers H and L to register r.
MVI r <B_2>	Load byte two of the instruction into register r.
MVI M <B_2>	Load byte two of the instruction into the memory location addressed by the contents of registers H and L.
INR r	The content of register r is incremented by one.
DCR r	The content of register r is decremented by one.
ADD r	Add the content of register r to the content of register A and place the result into register A
ADC r	Add the content of register r and the contents of the carry flip-flop to the content of register A and place the result into register A.
SUB r	Subtract the content of register r from the content of register A and place the result into register A. Two's compliment subtraction is used.

Mnemonic	Description
SBB r	Subtract the content of register r and the content of the carry flip-flop from the content of register A and place the result into register A.
ANA r	Place the logical product of the register A and register r into register A.
XRA r	Place the "exclusive-or" of the content of register A and register r into register A.
ORA r	Place the "inclusive-or" of the content of register A and register r into register A.
CMP r	Compare the content of register A with the content of register r.
ADD M	Add the content of the memory location (M) addressed by the content of the register pair H and L and place the result in register A.
ADC M	Add memory to A with carry.
SUB M	Subtract memory from A.
SBB M	Subtract memory from A with borrow.
ANA M	Place the logical AND of memory and A into A.
XRA M	Place the "exclusive-or" of A and memory into A.
ORA M	Place the "inclusive-or" of A and memory into A.
CMP M	Compare A with memory.
ADI <B_2>	Add immediate.

Mnemonic	Description
ACI <B_2>	Add immediate with carry.
SUI <B_2>	Subtract immediate.
SBI <B_2>	Subtract immediate with borrow.
ANI <B_2>	AND (logical) immediate.
XRI <B_2>	Exclusive OR immediate.
ORI <B_2>	Inclusive OR immediate.
CPI <B_2>	Compare immediate.
RLC	Rotate the content of register A left one bit.
RRC	Rotate the content of register A right one bit.
RAL	Rotate the content of register A left one bit.
RAR	Rotate the content of register A right one bit.
JMP <B_2> <B_3>	Jump unconditionally to the instruction located in memory location addressed by byte two and byte three.
JC <B_2> <B_3>	Jump if carry = 1.
JNC <B_2> <B_3>	Jump if carry = 0.

Mnemonic	Description
JZ <B_2> <B_3>	Jump if zero = 1.
JNZ <B_2> <B_3>	Jump if zero = 0.
JP <B_2> <B_3>	Jump if sign = 0.
JM <B_2> <B_3>	Jump if sign = 1.
JPE <B_2> <B_3>	Jump if parity = 1.
JPO <B_2> <B_3>	Jump if parity = 0.
HLT	On receipt of the HALT instruction, the activity of the processor is immediately suspended in the STOPPED state.
CALL <B_2> <B_3>	Jump unconditionally to the instruction located in memory location addressed by byte two and byte three of the instruction. The content of program counter is saved in the stack and the content of stack pointer is decremented by two.
CC <B_2> <B_3>	Call if carry = 1.

Mnemonic	Description
CNC <B_2> <B_3>	Call if carry = 0.
CZ <B_2> <B_3>	Call if zero = 1.
CNZ <B_2> <B_3>	Call if zero = 0.
CP <B_2> <B_3>	Call if sign = 0.
CM <B_2> <B_3>	Call if sign = 1.
CPE <B_2> <B_3>	Call if parity = 1.
CPO <B_2> <B_3>	Call if parity = 0.
RET	Return to the instruction in the memory location addressed by the last values shifted into the pushdown stack addressed by the stack pointer. The content of stack pointer is incremented by two.
RC	Return if carry = 1.
RNC	Return if carry = 0.
RZ	Return if zero = 1.

Mnemonic	Description
RNZ	Return if zero = 0.
RP	Return if sign = 0.
RM	Return if sign = 1.
RPE	Return if parity = 1.
RPO	Return if parity = 0.
RST	Restart (Interrupt Instruction).
IN <B_2>	Byte two of the instruction, which denotes the I/O device number, is sent to the I/O device through the address lines, and the INP status information is sent out at SYNC time. New data for the accumulator is loaded from the data bus when DBIN signal is active.
OUT <B_2>	Byte two of the instruction, which denotes the I/O device number, is sent to the I/O device through the address lines and the OUT status information is sent out at SYNC time. The content of the accumulator is made available on the data bus when the $\overline{WR}$ control signal is 0.
LXI B <B_2>	Load byte two of the instruction into C.
<B_3>	Load byte three of the instruction into B.
LXI D <B_2>	Load byte two of the instruction into E.
<B_3>	Load byte three of the instruction into D.
LXI H <B_2>	Load byte two of the instruction into L. Load byte three of the instruction into D.
<B_3>	
LXI SP <B_2>	Load byte two of the instruction into the lower order eight-bit of the stack pointer and byte three into the higher order eight-bit of the stack pointer.
<B_3>	

Mnemonic	Description
PUSH PSW	Save the contents of A and F (5-flags) into the pushdown stack addressed by the SP register.
PUSH B	Save the contents of B and C registers.
PUSH D	Save the contents of D and E registers.
PUSH H	Save the contents of H and L registers.
POP PSW	Restore the last values in the pushdown stack addressed by SP into A and F. The content of SP is incremented by two.
POP B	Restore the last values in the pushdown stack addressed by SP into B and C.
POP D	Restore the last values in the pushdown stack addressed by SP into D and E.
POP H	Restore the last values in the pushdown stack addressed by SP into H and L.
STA <B_2> <B_3>	Store the accumulator content into the memory location addressed by byte two and byte three of the instruction.
LDA <B_2> <B_3>	Load the accumulator with the content of the memory location addressed by byte two and byte three of the instruction.
XCHG	Exchange the contents of registers H and L and registers D and E.
XTHL	Exchange the contents of registers H, L and the last values in the pushdown stack addressed by register SP. The SP register itself is not changed.
SPHL	Transfer the contents of registers H and L into register SP.

Mnemonic	Description
PCHL	Jump indirect.
DAD SP	Add the content of register SP to the content of registers H and L and place the result into registers H and L.
DAD B	Add the content of registers B and C to the content of registers H and L and place the result into registers H and L.
DAD H	Add the content of registers H and L to the content of registers H and L and place the result into registers H and L.
DAD D	Add the content of registers D and E to the content of registers H and L and place the result into registers H and L.
STAX B	Store the accumulator content in the memory location addressed by the content of registers B and C.
STAX D	Store the accumulator content into the memory location addressed by the content of register D and E.
LDAX B	Load the accumulator with the content of the memory location addressed by the content of registers B and C.
LDAX D	Load the accumulator with the content of memory location addressed by the content of registers D and E.
INX B	The content of register pair B and C is incremented by one.
INX H	The content of register pair H and L is incremented by one.

Mnemonic	Description
INX D	The content of register pair D and E is incremented by one.
INX SP	The content of register SP is incremented by one.
DCX B	The content of register pair B and C is decremented by one.
DCX H	The content of register pair H and L is decremented by one.
DCX D	The content of register pair D and E is decremented by one.
DCX SP	The content of stack pointer SP is decremented by one.
CMA	The content of accumulator is complimented.
STC	Set the carry flip-flop to 1.
CMC	The content of carry is complimented.
DAA	Decimal adjust accumulator.
SHLD <B_2> <B_3>	Store the contents of registers H and L into the memory location addressed by byte two and byte three of the instruction.
LHLD <B_2> <B_3>	Load the registers H and L with the contents of the memory location addressed by byte two and byte three of the instruction.
EI	Instrrupt system enable.
DI	Interrupt system disable.
INR M	The content of memory designated by registers H and L is incremented by one.
DCR M	The content of memory designated by registers H and L is decremented by one.

APPENDIX B

INSTRUCTION CODE FOR THE INSTRUCTION SET OF THE INTEL 8080 MICROPROCESSOR

Mnemonic	Bytes	Instruction Code							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
MOV r_1, r_2	1	0	1	D	D	D	S	S	S
MOV M, r	1	0	1	1	1	0	S	S	S
MOV r, M	1	0	1	D	D	D	1	1	0
HLT	1	0	1	1	1	0	1	1	0
MVI r	2	0	0	D	D	D	1	1	0
MVI M	2	0	0	1	1	0	1	1	0
INR r	1	0	0	D	D	D	1	0	0
DCR r	1	0	0	D	D	D	1	0	1
ADD r	1	1	0	0	0	0	S	S	S
ADC r	1	1	0	0	0	1	S	S	S
SUB r	1	1	0	0	1	0	S	S	S
SBB r	1	1	0	0	1	1	S	S	S
ANA r	1	1	0	1	0	0	S	S	S
XRA r	1	1	0	1	0	1	S	S	S
ORA r	1	1	0	1	1	0	S	S	S
CMP r	1	1	0	1	1	1	S	S	S
ADD M	1	1	0	0	0	0	1	1	0
ADC M	1	1	0	0	0	1	1	1	0
SUB M	1	1	0	0	1	0	1	1	0

Mnemonic	Bytes	Instruction Code							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
SBB M	1	1	0	0	1	1	1	1	0
ANA M	1	1	0	1	0	0	1	1	0
XRA M	1	1	0	1	0	1	1	1	0
ORA M	1	1	0	1	1	0	1	1	0
CMP M	1	1	0	1	1	1	1	1	0
ADI	2	1	1	0	0	0	1	1	0
ACI	2	1	1	0	0	1	1	1	0
SUI	2	1	1	0	1	0	1	1	0
SBI	2	1	1	0	1	1	1	1	0
ANI	2	1	1	1	0	0	1	1	0
XRI	2	1	1	1	0	1	1	1	0
ORI	2	1	1	1	1	0	1	1	0
CPI	2	1	1	1	1	1	1	1	0
RLC	1	0	0	0	0	0	1	1	1
RRC	1	0	0	0	0	1	1	1	1
RAL	1	0	0	0	1	0	1	1	1
RAR	1	0	0	0	1	1	1	1	1
JMP	3	1	1	0	0	0	0	1	1
JC	3	1	1	0	1	1	0	1	0
JNC	3	1	1	0	1	0	0	1	0
JZ	3	1	1	0	0	1	0	1	0
JNZ	3	1	1	0	0	0	0	1	0
JP	3	1	1	1	1	0	0	1	0
JM	3	1	1	1	1	1	0	1	0

Mnemonic	Bytes	Instruction Code							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
JPE	3	1	1	1	0	1	0	1	0
JPO	3	1	1	1	0	0	0	1	0
CALL	3	1	1	0	0	1	1	0	1
CC	3	1	1	0	1	1	1	0	0
CNC	3	1	1	0	1	0	1	0	0
CZ	3	1	1	0	0	1	1	0	0
CNZ	3	1	1	0	0	0	1	0	0
CP	3	1	1	1	1	0	1	0	0
CM	3	1	1	1	1	1	1	0	0
CPE	3	1	1	1	0	1	1	0	0
CPO	3	1	1	1	0	0	1	0	0
RET	1	1	1	0	0	1	0	0	1
RC	1	1	1	0	1	1	0	0	0
RNC	1	1	1	0	1	0	0	0	0
RZ	1	1	1	0	0	1	0	0	0
RNZ	1	1	1	0	0	0	0	0	0
RP	1	1	1	1	1	0	0	0	0
RM	1	1	1	1	1	1	0	0	0
RPE	1	1	1	1	0	1	0	0	0
RPO	1	1	1	1	0	0	0	0	0
RST	1	1	1	A	A	A	1	1	1
IN	2	1	1	0	1	1	0	1	1
OUT	2	1	1	0	1	0	0	1	1
LXI B	3	0	0	0	0	0	0	0	1

Mnemonic	Bytes	Instruction Code							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
LXI D	3	0	0	0	1	0	0	0	1
LXI H	3	0	0	1	0	0	0	0	1
LXI SP	3	0	0	1	1	0	0	0	1
PUSH B	1	1	1	0	0	0	1	0	1
PUSH D	1	1	1	0	1	0	1	0	1
PUSH H	1	1	1	1	0	0	1	0	1
PUSH PSW	1	1	1	1	1	0	1	0	1
POP B	1	1	1	0	0	0	0	0	1
POP D	1	1	1	0	1	0	0	0	1
POP H	1	1	1	1	0	0	0	0	1
POP PSW	1	1	1	1	1	0	0	0	1
STA	3	0	0	1	1	0	0	1	0
LDA	3	0	0	1	1	1	0	1	0
XCHG	1	1	1	1	0	1	0	1	1
XTHL	1	1	1	1	0	0	0	1	1
SPHL	1	1	1	1	1	1	0	0	1
PCHL	1	1	1	1	0	1	0	0	1
DAD B	1	0	0	0	0	1	0	0	1
DAD D	1	0	0	0	1	1	0	0	1
DAD H	1	0	0	1	0	1	0	0	1
DAD SP	1	0	0	1	1	1	0	0	1
STAX B	1	0	0	0	0	0	0	1	0
STAX D	1	0	0	0	1	0	0	1	0
LDAX B	1	0	0	0	0	1	0	1	0

Mnemonic	Bytes	Instruction Code							
		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
LDAX D	1	0	0	0	1	1	0	1	0
INX B	1	0	0	0	0	0	0	1	1
INX D	1	0	0	0	1	0	0	1	1
INX H	1	0	0	1	0	0	0	1	1
INX SP	1	0	0	1	1	0	0	1	1
DCX B	1	0	0	0	0	1	0	1	1
DCX D	1	0	0	0	1	1	0	1	1
DCX H	1	0	0	1	0	1	0	1	1
DCX SP	1	0	0	1	1	1	0	1	1
CMA	1	0	0	1	0	1	1	1	1
STC	1	0	0	1	1	0	1	1	1
CMC	1	0	0	1	1	1	1	1	1
DAA	1	0	0	1	0	0	1	1	1
SHLD	3	0	0	1	0	0	0	1	0
LHLD	3	0	0	1	0	1	0	1	0
EI	1	1	1	1	1	1	0	1	1
DI	1	1	1	1	1	0	0	1	1
NOP		0	0	0	0	0	0	0	0

NOTE: DDD or SSS - 000B - 001C - 010D - 011E - 100H - 101L - 110 Memory - 111A.

VITA

Jayantilal Govindbhai Patel was born on May 7, 1949, in a village of Zadeshwar, Gujarat (state), India. He graduated from the Progressive High School, Broach, India, in 1967. In 1973, he received the Bachelor of Science degree in Electrical Engineering from The Maharaja Sayajirao University of Baroda, Baroda, India. At this time he began working for the Hindustan Conductors Pvt. Ltd. in Nadiad, India. On September 9, 1974, he came to the United States of America for further studies at The University of Tennessee at Knoxville. In June, 1976 he received his Master of Science degree in Electrical Engineering from The University of Tennessee, at Knoxville.