

Chapter 2
8085A
Functional
Description

MOS

8085A

MOS

8085A



CHAPTER 2

8085A FUNCTIONAL DESCRIPTION

2.1 WHAT THE 8085A IS

The 8085A is an 8-bit general-purpose microprocessor that is very cost-effective in small systems because of its extraordinarily low hardware overhead requirements. At the same time it is capable of accessing up to 64K bytes of memory and has status lines for controlling large systems.

2.2 WHAT'S IN THE 8085A

In the 8085A microprocessor are contained the functions of clock generation, system bus control, and interrupt priority selection, in addition to execution of the instruction set. (See Figure 2-1.) The 8085A transfers data on an 8-bit, bidirectional 3-state bus (AD_{0-7}) which is time-multiplexed so as to also transmit the eight lower-order address bits. An additional eight lines (A_{8-15}) expand the MCS-85 system memory addressing capability to 16 bits, thereby allowing 64K bytes of memory to be accessed directly by the CPU. The 8085A CPU (central processing unit) generates control signals that can be used to select appropriate external devices and

functions to perform READ and WRITE operations and also to select memory or I/O ports. The 8085A can address up to 256 different I/O locations. These addresses have the same numerical values (00 through FFH) as the first 256 memory addresses; they are distinguished by means of the IO/\overline{M} output from the CPU. You may also choose to address I/O ports as memory locations (i.e., memory-map the I/O, Section 3.2).

2.2.1 Registers

The 8085A, like the 8080, is provided with internal 8-bit registers and 16-bit registers. The 8085A has eight addressable 8-bit registers. Six of them can be used either as 8-bit registers or as 16-bit register pairs. Register pairs are treated as though they were single, 16-bit registers; the high-order byte of a pair is located in the first register and the low-order byte is located in the second. In addition to the register pairs, the 8085A contains two more 16-bit registers.

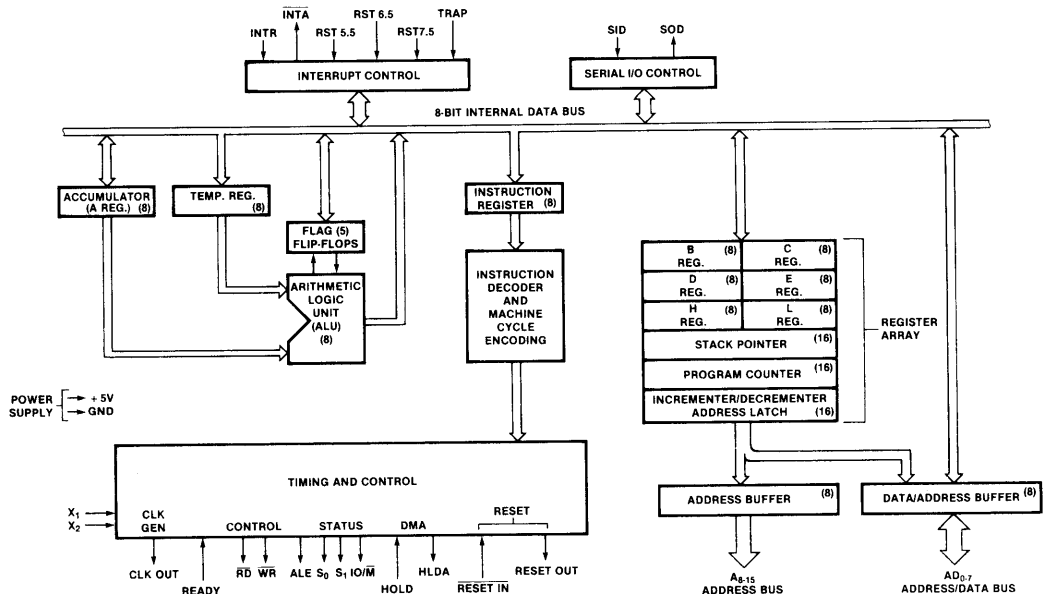


FIGURE 2-1 8085A CPU FUNCTIONAL BLOCK DIAGRAM

2.2.4 Arithmetic-Logic Unit (ALU)

The ALU contains the accumulator and the flag register (described in Sections 2.2.1 and 2.2.2) and some temporary registers that are inaccessible to the programmer.

Arithmetic, logic, and rotate operations are performed by the ALU. The results of these operations can be deposited in the accumulator, or they can be transferred to the internal data bus for use elsewhere.

2.2.5 Instruction Register and Decoder

During an instruction fetch, the first byte of an instruction (containing the opcode) is transferred from the internal bus to the 8-bit instruction register. (See Figure 2-1.) The contents of the instruction register are, in turn, available to the instruction decoder. The output of the decoder, gated by timing signals, controls the registers, ALU, and data and address buffers. The outputs of the instruction decoder and internal clock generator generate the state and machine cycle timing signals.

2.2.6 Internal Clock Generator

The 8085A CPU incorporates a complete clock generator on its chip, so it requires only the addition of a quartz crystal to establish timing for its operation. (It will accept an external clock input at its X_1 input instead, however.) A suitable crystal for the standard 8085A must be parallel-resonant at a fundamental of 6.25 MHz or less, twice the desired internal clock frequency. The 8085A-2 will operate with crystal of up to 10 MHz. The functions of the 8085A internal clock generator are shown in Figure 2-2. A Schmitt trigger is used interchangeably as oscillator or

as input conditioner, depending upon whether a crystal or an external source is used. The clock circuitry generates two nonoverlapping internal clock signals, ϕ_1 and ϕ_2 (see Figure 2-2). ϕ_1 and ϕ_2 control the internal timing of the 8085A and are not directly available on the outside of the chip. The external pin CLK is a buffered, inverted version of ϕ_1 . CLK is half the frequency of the crystal input signal and may be used for clocking other devices in the system.

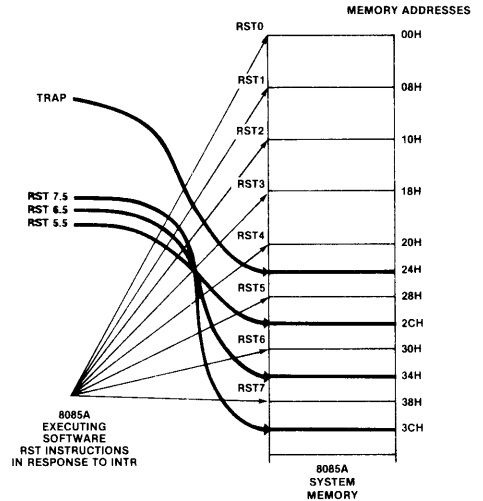
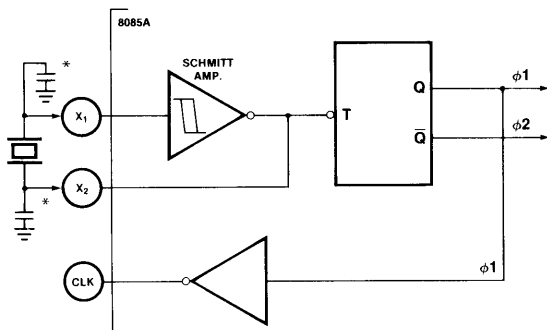


FIGURE 2-3 8085A HARDWARE AND SOFTWARE RST BRANCH LOCATIONS

2.2.7 Interrupts

The five hardware interrupt inputs provided in the 8085A are of three types. INTR is identical with the 8080A INT line in function; i.e., it is maskable (can be enabled or disabled by EI or DI software instructions), and causes the CPU to fetch in an RST instruction, externally placed on the data bus, which vectors a branch to any one of eight fixed memory locations (Restart addresses). (See Figure 2-3.) INTR can also be controlled by the 8259 programmable interrupt controller, which generates CALL instructions instead of RSTs, and can thus vector operation of the CPU to a preprogrammed subroutine located anywhere in your system's memory map. The RST 5.5, RST 6.5, and RST 7.5 hardware interrupts are different in function in that they are maskable through the use of the SIM



*EXTERNAL CAPACITORS REQUIRED ONLY FOR CRYSTAL FREQUENCIES ≤ 4 MHz.

FIGURE 2-2 8085A CLOCK LOGIC

FUNCTIONAL DESCRIPTION

instruction, which enables or disables these interrupts by clearing or setting corresponding mask flags based on data in the accumulator. (See Figure 2-4.) You may read the status of the interrupt mask previously set by performing a RIM instruction. Its execution loads into the accumulator the following information. (See Figure 2-5.)

- Current interrupt mask status for the RST 5.5, 6.5, and 7.5 hardware status.
- Current interrupt enable flag status (except that immediately following TRAP, the IE flag status preceding that interrupt is loaded).
- RST 5.5, 6.5, and 7.5 interrupts pending.

RST 5.5, 6.5, and 7.5 are also subject to being enabled or disabled by the EI and DI instructions, respectively. INTR, RST 5.5, and RST 6.5 are level-sensitive, meaning that these inputs may be acknowledged by the processor when they are held at a high level. RST 7.5 is edge-sensitive, meaning that an internal flip-flop in the 8085A registers the occurrence of an interrupt the instant a rising edge appears on the RST 7.5 input line. This input need not be held high; the flip-flop will remain set until it is cleared by one of three possible actions:

- The 8085A responds to the interrupt, and sends an internal reset signal to the RST 7.5 flip-flop. (See Figure 2-6A.)

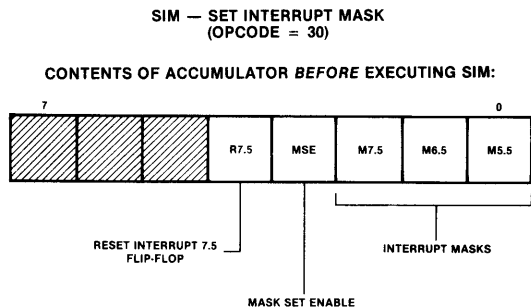


FIGURE 2-4 INTERRUPT MASKS SET USING SIM INSTRUCTION

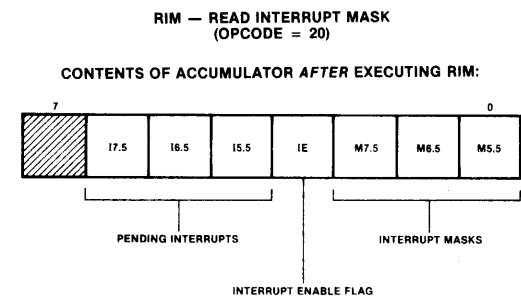


FIGURE 2-5 RIM — READ INTERRUPT MASK

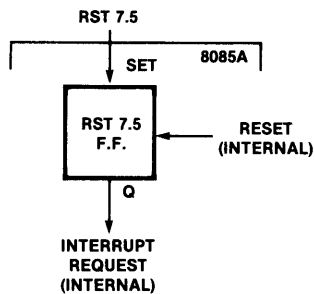


FIGURE 2-6A RST 7.5 FLIP FLOP

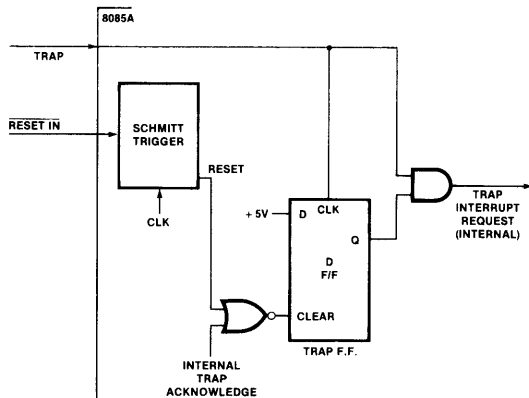


FIGURE 2-6B TRAP INTERRUPT INPUTS

FIGURE 2-6 RST 7.5 AND TRAP INTERRUPT INPUTS

FUNCTIONAL DESCRIPTION

- The 8085A, before responding to the RST 7.5 interrupt, receives a RESET IN signal from an external source; this also activates the internal reset.
- The 8085A executes a SIM instruction, with accumulator bit 4 previously set to 1. (See Figure 2-4.)

The third type of hardware interrupt is TRAP. This input is not subject to any mask or interrupt enable/disable instruction. The receipt of a positive-going edge on the TRAP input triggers the processor's hardware interrupt sequence, but the pulse must be held high until acknowledged internally (see Figure 2-6B).

The sampling of all interrupts occurs on the descending edge of CLK, one cycle before the end of the instruction in which the interrupt input is activated. To be recognized, a valid interrupt must occur at least 160 ns before sampling time in the 8085A, or 150 ns in the 8085A-2. This means that to guarantee being recognized, RST 5.5 and 6.5 and TRAP need to be held on for at least 17 clock states plus 160 ns (150 for 8085A-2), assuming that the interrupt might arrive just barely too late to be acknowledged during a particular instruction, and that the following instruction might be an 18-state CALL. This timing assumes no WAIT or HOLD cycles are used.

The way interrupt masks are set and read is described in Chapter 4 under the RIM (read in-

terrupt mask) and SIM (set interrupt mask) instruction listings. Interrupt functions and their priorities are shown in the table that follows.

Name	Priority	Address (1) Branched to when inter- rupt occurs	Type Trigger
TRAP	1	24H	Rising edge AND high level until sampled
RST 7.5	2	3CH	Rising edge (latched)
RST 6.5	3	34H	High level until sam- pled
RST 5.5	4	2CH	High level until sam- pled
INTR	5	(2)	High level until sam- pled

NOTES:

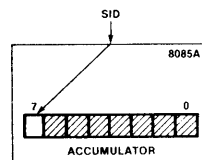
- (1) In the case of TRAP and RST 5.5-7.5, the contents of the Program Counter are pushed onto the stack before the branch occurs.
- (2) Depends on the instruction that is provided to the 8085A by the 8259 or other circuitry when the interrupt is acknowledged.

2.2.8 Serial Input and Output

The SID and SOD pins help to minimize chip count in small systems by providing for easy interface to a serial port using software for timing and for coding and decoding of the data. Each time a RIM instruction is executed, the status of the SID pin is read into bit 7 of the accumulator. RIM is thus a dual-purpose instruction. (See Chapter 4.) In similar fashion, SIM is used to latch bit 7 of the accumulator out to the SOD output via an internal flip-flop, providing that bit 6 of the accumulator is set to 1. (See Figure 2-7.) Section 2.3.8 describes SID and SOD timing.

SID can also be used as a general purpose TEST input and SOD can serve as a one-bit control output.

EFFECT OF RIM INSTRUCTION



EFFECT OF SIM INSTRUCTION

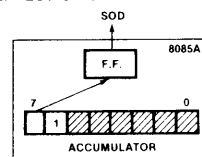


FIGURE 2-7 EFFECT OF RIM AND SIM INSTRUCTIONS ON SERIAL DATA LINES

FUNCTIONAL DESCRIPTION

2.3 HOW THE MCS-85 SYSTEM WORKS

The 8085A CPU generates signals that tell peripheral devices what type of information is on the multiplexed Address/Data bus and from that point on the operation is almost identical to the MCS-80™ CPU Group. A multiplexed bus structure was chosen because it freed device pins so that more functions could be integrated on the 8085A and other components of the family. The multiplexed bus is designed to allow complete compatibility to existing peripheral

components with improved timing margins and access requirements. (See Figure 2-8.)

To enhance the system integration of MCS-85, several special components with combined memory and I/O were designed. These new devices directly interface to the multiplexed bus of the 8085A. The pin locations of the 8085A and the special peripheral components are assigned to minimize PC board area and to allow for efficient layout. The details on peripheral components are contained in subsequent paragraphs of this chapter and in Chapters 5 and 6.

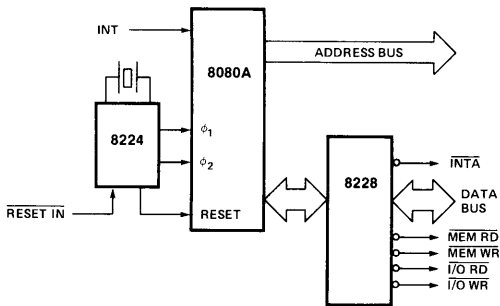


FIGURE 2-8A MCS-80™ CPU GROUP

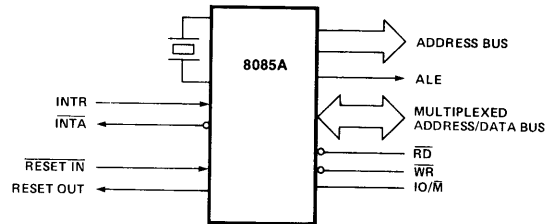


FIGURE 2-8B MCS-85™ CPU/8085A (MCS-80 COMPATIBLE FUNCTIONS)

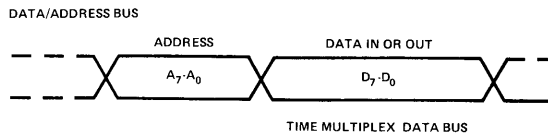


FIGURE 2-8C MULTIPLEXED BUS TIMING

FIGURE 2-8 BASIC CPU FUNCTIONS

2.3.1 Multiplexed Bus Cycle Timing

The execution of any 8085A program consists of a sequence of READ and WRITE operations, of which each transfers a byte of data between the 8085A and a particular memory or I/O address. These READ and WRITE operations are the only communication between the processor and the other components, and are all that is necessary to execute any instruction or program.

Each READ or WRITE operation of the 8085A is referred to as a machine cycle. The execution of each instruction by the 8085A consists of a sequence of from one to five machine cycles, and each machine cycle consists of a minimum of from three to six clock cycles (also referred to as T states). Consider the case of the Store Accumulator Direct (STA) instruction, shown in Figure 2-9. The STA instruction causes the contents of the accumulator to be stored at the direct address specified in the second and third bytes of the instruction. During the first machine cycle (M_1), the CPU puts the contents of the program counter (PC) on the address bus and performs a MEMORY READ cycle to read from memory the opcode of the next instruction (STA). The M_1 machine cycle is also referred to as the OPCODE FETCH cycle, since it fetches the operation code of the next instruction. In the fourth clock cycle (T_4) of M_1 , the CPU interprets the data read in and recognizes it as the opcode of the STA instruction. At this point the

CPU knows that it must do three more machine cycles (two MEMORY READs and one MEMORY WRITE) to complete the instruction.

The 8085A then increments the program counter so that it points to the next byte of the instruction and performs a MEMORY READ machine cycle (M_2) at address (PC + 1). The accessed memory places the addressed data on the data bus for the CPU. The 8085A temporarily stores this data (which is the low-order byte of the direct address) internally in the CPU. The 8085A again increments the program counter to location (PC + 2) and reads from memory (M_3) the next byte of data, which is the high-order byte of the direct address.

At this point, the 8085A has accessed all three bytes of the STA instruction, which it must now execute. The execution consists of placing the data accessed in M_2 and M_3 on the address bus, then placing the contents of the accumulator on the data bus, and then performing a MEMORY WRITE machine cycle (M_4). When M_4 is finished, the CPU will fetch (M_1) the first byte of the next instruction and continue from there.

State Transition Sequence

As the preceding example shows, the execution of an instruction consists of a series of machine cycles whose nature and sequence is determined by the opcode accessed in the M_1

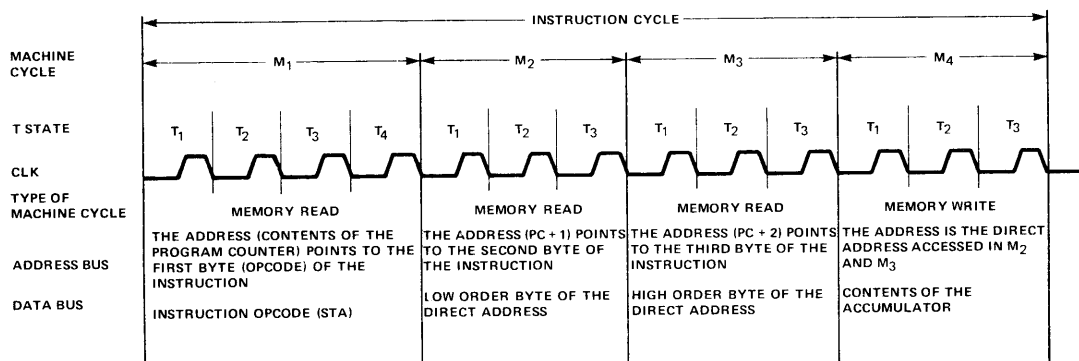


FIGURE 2-9 CPU TIMING FOR STORE ACCUMULATOR DIRECT (STA) INSTRUCTION

FUNCTIONAL DESCRIPTION

MACHINE CYCLE		STATUS			CONTROL		
		IO/M	S1	S0	RD	WR	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
INTR ACKNOWLEDGE (INA)		1	1	1	1	1	0
BUS IDLE (BI): DAD		0	1	0	1	1	1
	INA(RST/TRAP)	1	1	1	1	1	1
	HALT	TS	0	0	TS	TS	1

0 = Logic "0" 1 = Logic "1" TS = High Impedance X = Unspecified

FIGURE 2-10 8085A MACHINE CYCLE CHART

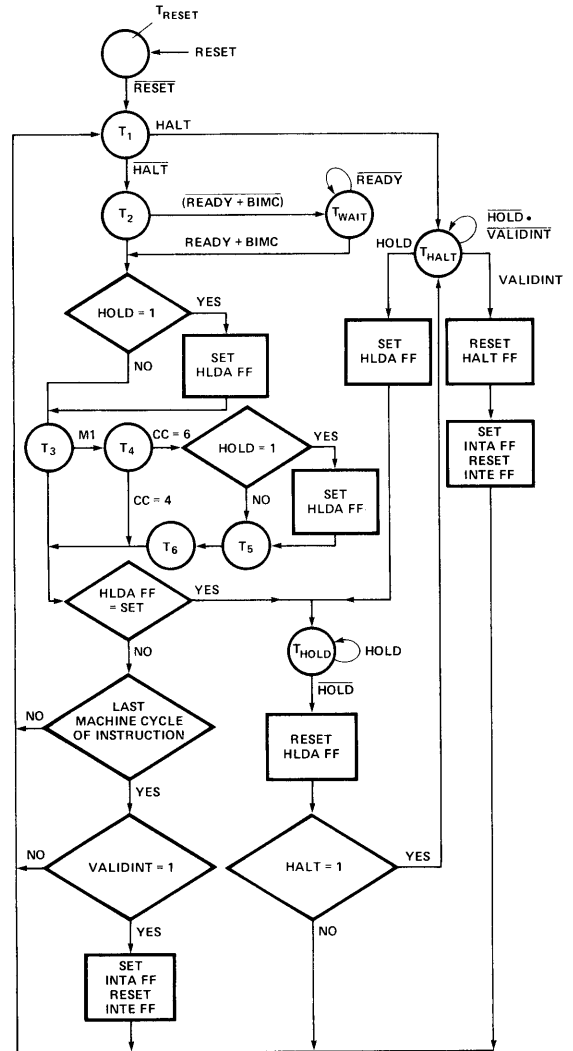
machine cycle. While no one instruction cycle will consist of more than five machine cycles, every machine cycle will be one of the seven types listed in Figure 2-10. These seven types of machine cycles can be differentiated by the state of the three status lines (IO/M, S₀, and S₁) and the three control signals (RD, WR, and INTA).

Most machine cycles consist of three T states, (cycles of the CLK output) with the exception of OPCODE FETCH, which normally has either four or six T states. The actual number of states required to perform any instruction depends on the instruction being executed, the particular machine cycle within the instruction cycle, and the number of WAIT and HOLD states inserted into each machine cycle through the use of the READY and HOLD inputs of the 8085A. The state transition diagram in Figure 2-11 illustrates how the 8085A proceeds in the course of a machine cycle. The state of various status and control signals, as well as the system buses, is shown in Figure 2-12 for each of the ten possible T states that the processor can be in.

Figure 2-11 also shows when the READY, HOLD, and interrupt signals are sampled, and how they modify the basic instruction sequence (T₁-T₆ and T_{WAIT}). As we shall see, the timings for each of the seven types of machine cycles are almost identical.

OPCODE FETCH (OF):

The OPCODE FETCH (OF) machine cycle is unique in that it has more than three clock cycles. This is because the CPU must interpret the opcode accessed in T₁, T₂, and T₃ before it can decide what to do next.



NOTE: SYMBOL DEFINITION

- T_x = CPU STATE T_x . ALL CPU STATE TRANSITIONS OCCUR ON THE FALLING EDGE OF CLK.
- X = A DECISION (X) THAT DETERMINES WHICH OF SEVERAL ALTERNATIVE PATHS TO FOLLOW.
- X = PERFORM THE ACTION X.
- = FLOWLINE THAT INDICATES THE SEQUENCE OF EVENTS.
- X — = FLOWLINE THAT INDICATES THE SEQUENCE OF EVENTS IF CONDITION X IS TRUE.
- CC = NUMBER OF CLOCK CYCLES IN THE CURRENT MACHINE CYCLE.
- BIMC = "BUS IDLE MACHINE CYCLE" = MACHINE CYCLE WHICH DOESN'T USE THE SYSTEM BUS.
- VALIDINT = "VALID INTERRUPT" - AN INTERRUPT IS PENDING THAT IS BOTH ENABLED AND UNMASKED (MASKING ONLY APPLIES FOR RST 5.5, 6.5, AND 7.5 INPUTS).
- HLDA FF = INTERNAL HOLD ACKNOWLEDGE FLIP FLOP. NOTE THAT THE 8085A SYSTEM BUSES ARE 3-STATED ONE CLOCK CYCLE AFTER THE HLDA FLIP FLOP IS SET.

FIGURE 2-11 8085A CPU STATE TRANSITION

FUNCTIONAL DESCRIPTION

Machine State	Status & Buses				Control		
	S1,S0	IO/ \overline{M}	A ₈ -A ₁₅	AD ₀ -AD ₇	RD,WR	INTA	ALE
T ₁	X	X	X	X	1	1	1 [†]
T ₂	X	X	X	X	X	X	0
T _{WAIT}	X	X	X	X	X	X	0
T ₃	X	X	X	X	X	X	0
T ₄	1	0*	X	TS	1	1	0
T ₅	1	0*	X	TS	1	1	0
T ₆	1	0*	X	TS	1	1	0
T _{RESET}	X	TS	TS	TS	TS	1	0
T _{HALT}	0	TS	TS	TS	TS	1	0
T _{HOLD}	X	TS	TS	TS	TS	1	0

0 = Logic "0" 1 = Logic "1" TS = High Impedance X = Unspecified

[†]ALE not generated during 2nd and 3rd machine cycles of DAD instruction.

*IO/ \overline{M} = 1 during T₄-T₆ states of RST and INA cycles.

FIGURE 2-12 8085A MACHINE STATE CHART

Figure 2-13 shows the timing relationships for an OF machine cycle. The particular instruction illustrated is DCX, whose timing for OF differs from other instructions in that it has six T states, while some instructions require only four T states for OF. In this discussion, as well as the following discussions, only the relative timing of the signals will be discussed; for the actual timings, refer to the data sheets of the individual parts in Chapters 5 and 6.

The first thing that the 8085A does at the beginning of every machine cycle is to send out three status signals (IO/ \overline{M} , S1, S0) that define what type of machine cycle is about to take place. The IO/ \overline{M} signal identifies the machine cycle as being either a memory reference or input/output operation. The S1 status signal identifies whether the cycle is a READ or WRITE operation. The S0 and S1 status signals can be used together (see Figure 2-10) to identify READ, WRITE, or OP CODE FETCH machine cycles as well as the HALT state. Referring to Figure 2-13, the 8085A will send out IO/ \overline{M} = 0, S1 = 1, S0 = 1 at the beginning of the machine cycle to identify it as a READ from a memory location to obtain an opcode; in other words, it identifies the machine cycle as an OP CODE FETCH cycle.

The 8085A also sends out a 16-bit address at the beginning of every machine cycle to identify the particular memory location or I/O port that the machine cycle applies to. In the case of an OF cycle, the contents of the program counter is placed on the address bus. The high order byte (PCH) is placed on the A₈-A₁₅ lines, where it will stay until at least T₄. The low order byte (PCL) is placed on the AD₀-AD₇ lines, whose three-state drivers are enabled if not found already on. Unlike the upper address lines, however, the information on the lower address lines will remain there for only one clock cycle, after which the drivers will go to their high impedance state, indicated by a dashed line in Figure 2-13. This is necessary because the AD₀-AD₇ lines are time multiplexed between the address and data buses. During T₁ of every machine cycle, AD₀-AD₇ output the lower 8-bits of address after which AD₀-AD₇ will either output the desired data for a WRITE operation or the drivers will float (as is the case for the OF cycle), allowing the external device to drive the lines for a READ operation.

Since the address information on AD₀-AD₇ is of a transitory nature, it must be latched either internally in special multiplexed-bus components like the 8155 or externally in parts like the 8212 8-bit latch. (See Chapter 3.) The 8085A provides a special timing signal, ADDRESS LATCH ENABLE (ALE), to facilitate the latching of A₀-A₇; ALE is present during T₁ of every machine cycle.

After the status signals and address have been sent out and the AD₀-AD₇ drivers have been disabled, the 8085A provides a low level on RD to enable the addressed memory device. The device will then start driving the AD₀-AD₇ lines; this is indicated by the dashed line turning into a solid line in Figure 2-13. After a period of time (which is the access time of the memory) valid data will be present on AD₀-AD₇. The 8085A during T₃ will load the memory data on AD₀-AD₇ into its instruction register and then raise RD to the high level, disabling the addressed memory device. At this point, the 8085A will have finished accessing the opcode of the instruction. Since this is the first machine cycle (M₁) of the instruction, the CPU will automatically step to T₄, as shown in Figure 2-11.

During T₄, the CPU will decode the opcode in the instruction register and decide whether to enter T₅ on the next clock or to start a new machine cycle and enter T₁. In the case of the DCX instruction shown in Figure 2-13, it will enter T₅ and then T₆ before going to T₁.

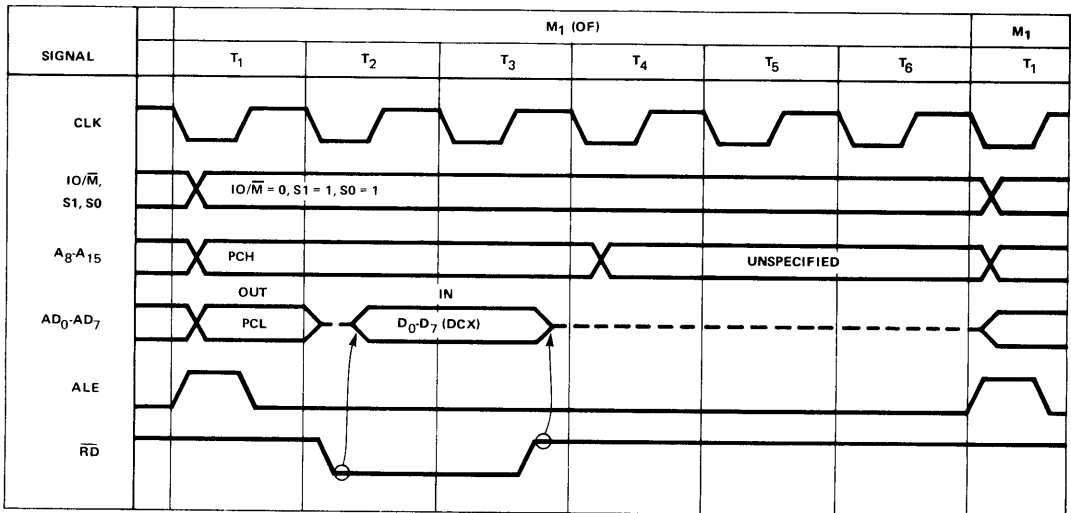


FIGURE 2-13 OPCODE FETCH MACHINE CYCLE (OF DCX INSTRUCTION)

During T₅ and T₆, of DCX, the CPU will decrement the designated register. Since the A₈-A₁₅ lines are driven by the address latch circuits, which are part of the incrementer/decrementer logic, the A₈-A₁₅ lines may change during T₅ and T₆. Because the value of A₈-A₁₅ can vary during T₄-T₆, it is most important that all memory and I/O devices on the system bus qualify their selection with \overline{RD} . If they don't use \overline{RD} , they may be spuriously selected. Moreover, with a linear selection technique (Chapter 3), two or more devices could be simultaneously enabled, which could be potentially damaging. The generation of spurious addresses can also occur momentarily at address bus transitional periods in T₁. Therefore, the selection of all memory and I/O devices must be qualified with \overline{RD} or \overline{WR} . Many new memory devices like the 8155 and 8355 have the \overline{RD} input that internally is used to enable the data bus outputs, removing the need for externally qualifying the chip enable input with \overline{RD} .

Figure 2-14 is identical to Figure 2-13 with one exception, which is the use of the READY line. As we can see in Figure 2-11, when the CPU is in T₂, it examines the state of the READY line. If the READY line is high, the CPU will proceed to T₃ and finish executing the instruction. If the READY line is low, however, the CPU will enter T_{WAIT} and stay there indefinitely until READY goes high. When the READY line does go high, the CPU will exit T_{WAIT} and enter T₃, in order to complete the machine cycle. As shown in

Figure 2-14, the external effect of using the READY line is to preserve the exact state of the processor signals at the end of T₂ for an integral number of clock periods, before finishing the machine cycle. This "stretching" of the system timing has the further effect of increasing the allowable access time for memory or I/O devices. By inserting T_{WAIT} states, the 8085A can accommodate even the slowest of memories. Another common use of the READY line is to single-step the processor with a manual switch.

2.3.2 Read Cycle Timing

MEMORY READ (MR):

Figure 2-15 shows the timing of two successive MEMORY READ (MR) machine cycles, the first without a T_{WAIT} state and the second with one T_{WAIT} state. The timing during T₁-T₃ is absolutely identical to the OPCODE FETCH machine cycle, with the exception that the status sent out during T₁ is IO/M = 0, S1 = 1, S0 = 0, identifying the cycles as a READ from a memory location. This differs from Figure 2-13 only in that S0 = 1 for an OF cycle, identifying that cycle as an OPCODE FETCH operation. Otherwise, the two cycles are identical during T₁-T₃.

A second difference occurs at the end of T₃. As shown in Figure 2-11, the CPU always goes to T₄ from T₃ during M₁, which is always an OF cycle. During all other machine cycles, the CPU will always go from T₃ to T₁ of the next machine cycle.

FUNCTIONAL DESCRIPTION

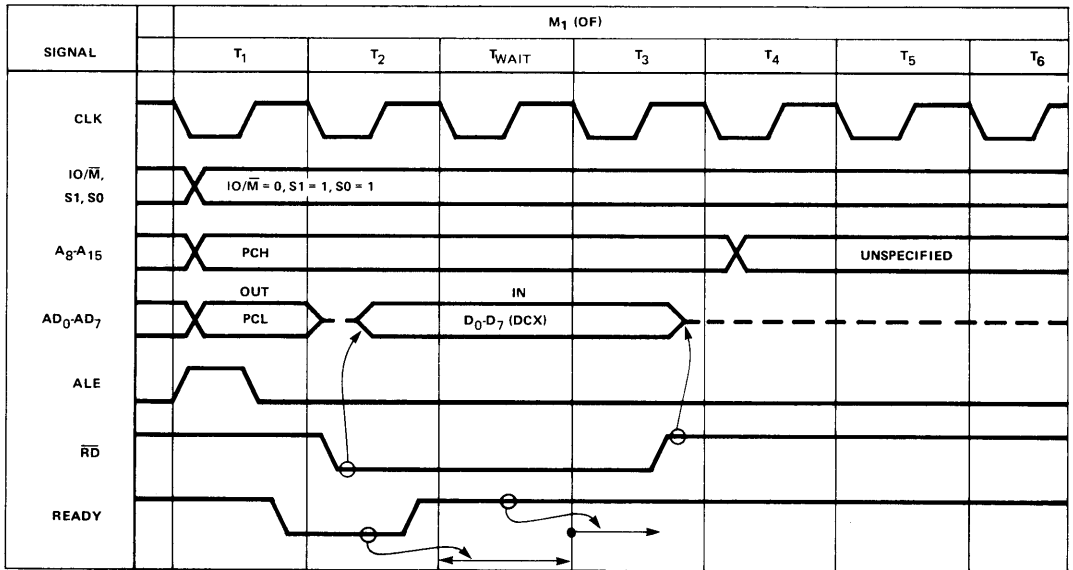


FIGURE 2-14 OPCODE FETCH MACHINE CYCLE WITH ONE WAIT STATE

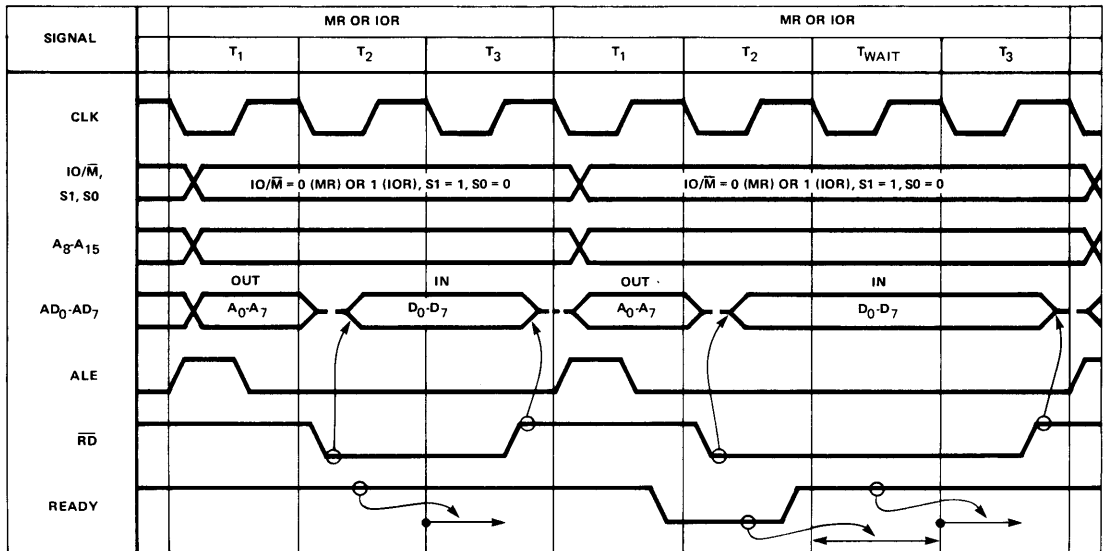


FIGURE 2-15 MEMORY READ (OR I/O READ) MACHINE CYCLES (WITH AND WITHOUT WAIT STATES)

The memory address used in the OF cycle is always the contents of the program counter, which points to the current instruction, while the address used in the MR cycle can have several possible origins. Also, the data read in during an MR cycle is placed in the appropriate register, not the instruction register.

I/O READ (IOR):

Figure 2-15 also shows the timing of two successive I/O READ (IOR) machine cycles, the first without a T_{WAIT} state. As is readily apparent, the timing of an IOR cycle is identical to the timing of an MR cycle, with the exception of $IO/\bar{M} = 0$ for MR and $IO/\bar{M} = 1$ for IOR; recall that IO/\bar{M} status signal identifies the address of the current machine cycle as selecting either a memory location or an I/O port. The address used in the IOR cycle comes from the second byte (Port No.) of an INPUT instruction. Note that the I/O port address is duplicated onto both AD_0-AD_7 and A_8-A_{15} . The IOR cycle can occur only as the third machine cycle of an INPUT instruction.

Note that the READY signal can be used to generate T_{WAIT} states for I/O devices as well as memory devices. By gating the READY signal with the proper status lines, one could generate T_{WAIT} states for memory devices only or for I/O devices only. By gating in the address lines, one can further qualify T_{WAIT} state generation by the particular devices being accessed.

2.3.3 WRITE Cycle Timing

MEMORY WRITE (MW):

Figure 2-16 shows the timing for two successive MEMORY WRITE (MW) machine cycles, the first without a T_{WAIT} state, and the second with one T_{WAIT} state. The 8085A sends out the status during T_1 in a similar fashion to the OF, MR and IOR cycles, except that $IO/\bar{M} = 0$, $S_1 = 0$, and $S_0 = 1$, identifying the current machine cycle as being a WRITE operation to a memory location.

The address is sent out during T_1 in an identical manner to MR. However, at the end of T_1 , there is a difference. While the AD_0-AD_7 drivers were disabled during T_2-T_3 of MR in expectation of the addressed memory device driving the AD_0-AD_7 lines, the drivers are not disabled for MW. This is because the CPU must provide the data to be written into the addressed memory location. The data is placed on AD_0-AD_7 at the start of T_2 . The \bar{WR} signal is also lowered at this time to enable the writing of the addressed memory device. During T_2 , the READY line is checked to see if a T_{WAIT} state is required. If READY is low, T_{WAIT} states are inserted until READY goes high. During T_3 , the \bar{WR} line is raised, disabling the addressed memory device and thereby terminating the WRITE operation. The contents of the address and data lines are not changed until the next T_1 , which directly follows.

Note that the data on AD_0-AD_7 is not guaranteed to be stable before the falling edge

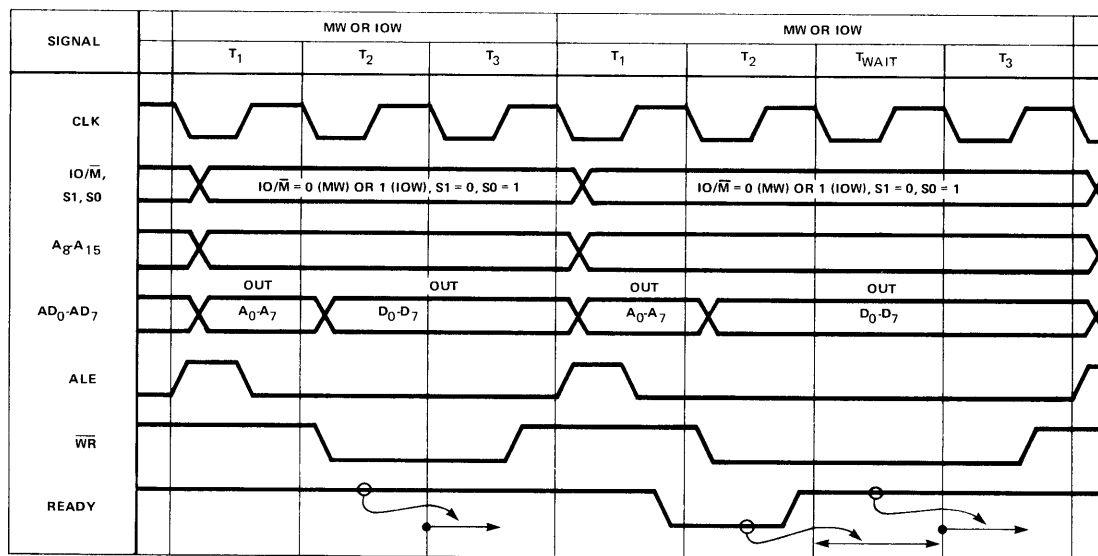


FIGURE 2-16 MEMORY WRITE (OR I/O WRITE) MACHINE CYCLES (WITH AND WITHOUT WAIT STATES)

of \overline{WR} . The AD_0 - AD_7 lines are guaranteed to be stable both before and after the rising edge of \overline{WR} .

I/O WRITE (IOW):

As Figure 2-16 shows, the timing for an I/O WRITE (IOW) machine cycle is the same as an MW machine cycle except that $IO/\overline{M} = 0$ during the MW cycle and $IO/\overline{M} = 1$ during the IOW cycle.

As with the IOR cycle discussed previously, the address used in an IOW cycle is the I/O port number which is duplicated on both the high and low bytes of the address bus. In the case of IOW, the port number comes from the second byte of an OUTPUT instruction as the instruction is executed.

2.3.4 Interrupt Acknowledge (INA) Timing

Figures 2-17 and 2-18 (a continuation of 2-17) depict the course of action the CPU takes in response to a high level on the INTR line if the INTE FF (interrupt enable flip-flop) has been set

by the EI instruction. The status of the TRAP and RST pins as well as INTR is sampled during the second clock cycle before $M_1 \cdot T_1$. If INTR was the only valid interrupt and if INTE FF is set, then the CPU will reset INTE FF and then enter an INTERRUPT ACKNOWLEDGE (INA) machine cycle. The INA cycle is identical to an OF cycle with two exceptions. \overline{INTA} is sent out instead of \overline{RD} . Also, $IO/\overline{M} = 1$ during INA, whereas $IO/\overline{M} = 0$ for OF. Although the contents of the program counter are sent out on the address lines, the address lines can be ignored.

When \overline{INTA} is sent out, the external interrupt logic must provide the opcode of an instruction to execute. The opcode is placed on the data bus and read in by the processor. If the opcode is the first byte of a multiple-byte instruction, additional \overline{INTA} pulses will be provided by the 8085A to clock in the remaining bytes. RESTART and CALL instructions are the most

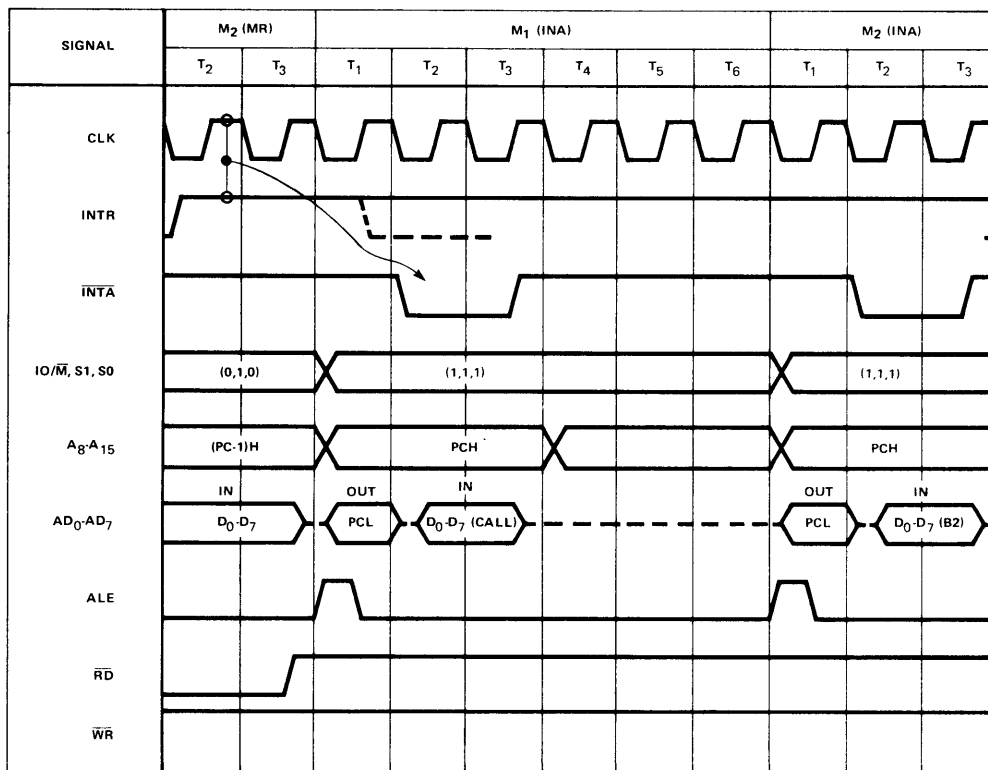


FIGURE 2-17 INTERRUPT ACKNOWLEDGE MACHINE CYCLES (WITH CALL INSTRUCTION IN RESPONSE TO INTR)

FUNCTIONAL DESCRIPTION

logical choices, since they both force the processor to push the contents of the program counter onto the stack before jumping to a new location. In Figure 2-17 it is assumed that a CALL opcode is sent to the CPU during M₁. The CALL opcode could have been placed there by a device like the 8259 programmable interrupt controller.

After receiving the opcode, the processor then decodes it and determines, in this case, that the CALL instruction requires two more bytes. The CPU therefore performs a second INA cycle (M₂) to access the second byte of the instruction from the 8259. The timing of this cycle is identical to M₁, except that it has only three T states. M₂ is followed by another INA cycle (M₃) to access the third byte of the CALL instruction from the 8259.

Now that the CPU has accessed the entire instruction used to acknowledge the interrupt, it will execute that instruction. Note that any instruction could be used (except EI or DI, the instructions which enable or disable interrupts), but the RESTART and CALL instructions are the most logical choices. Also notice that the CPU inhibited the incrementing of the program counter (PC) during the three INA cycles, so that the correct PC value can be pushed onto the stack during M₄ and M₅.

During M₄ and M₅, the CPU performs MEMORY WRITE machine cycles to write the upper and then lower bytes of the PC onto the top of the stack. The CPU then places the two bytes accessed in M₂ and M₃ into the lower and upper bytes of the PC. This has the effect of jumping the execution of the program to the location specified by the CALL instruction.

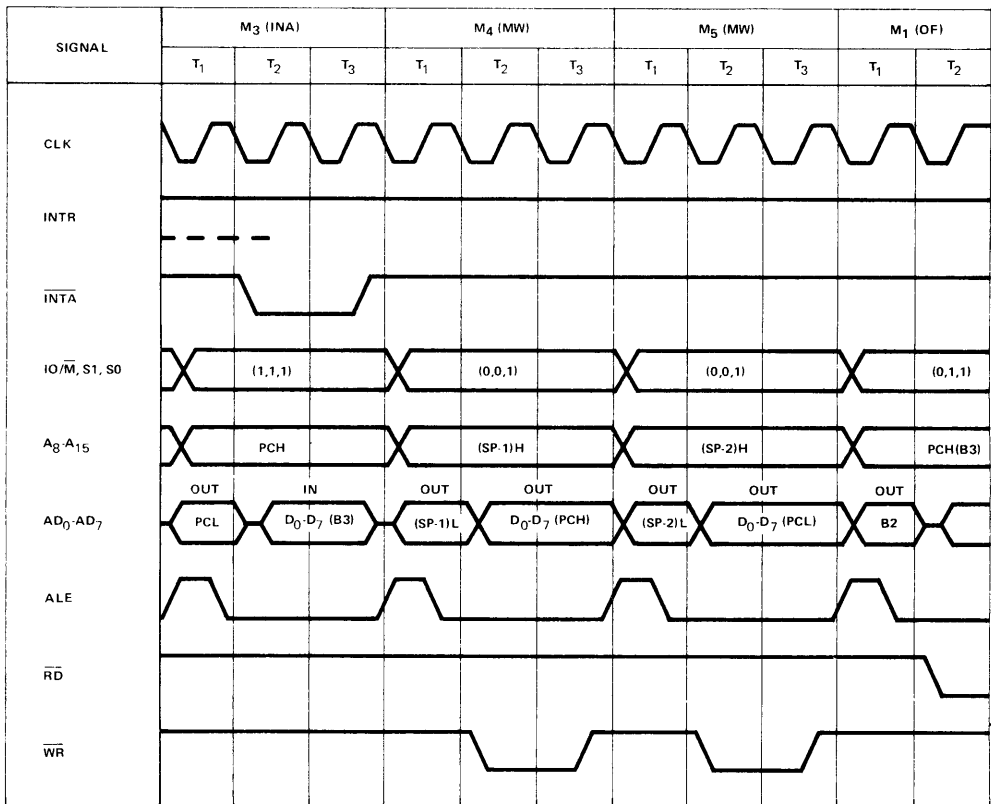


FIGURE 2-18 INTERRUPT ACKNOWLEDGE MACHINE CYCLES (WITH CALL INSTRUCTION IN RESPONSE TO INTR)

2.3.5 Bus Idle (BI) and HALT State

Most machine cycles of the 8085A are associated with either a READ or WRITE operation. There are two exceptions to this rule. The first exception takes place during M_2 and M_3 of the DAD instruction. The 8085A requires six internal T states to execute a DAD instruction, but it is not desirable to have M_1 be ten (four normal plus six extra) states long. Therefore, the CPU generates two extra machine cycles that do not access either the memory or the I/O. These cycles are referred to as BUS IDLE (BI) machine cycles. In the case of DAD, they are identical to MR cycles except that RD remains high and ALE is not generated. Note that READY is ignored during M_2 and M_3 of DAD.

The other time when the BUS IDLE machine cycle occurs is during the internal opcode generation for the RST or TRAP interrupts. Figure 2-19 illustrates the BI cycle generated in response to RST 7.5. Since this interrupt is rising-edge-triggered, it sets an internal latch; that latch is sampled at the falling edge of the next to the last T-state of the previous instruction. At this point the CPU must generate its own internal RESTART instruction which will (in subsequent machine cycles) cause the processor to push the program counter on the stack and to vector to location 3CH. To do this, it executes an OF machine cycle without issuing RD, generating the RESTART opcode instead. After M_1 , the CPU continues execution normally in all respects except that the state of the READY line is ignored during the BI cycle.

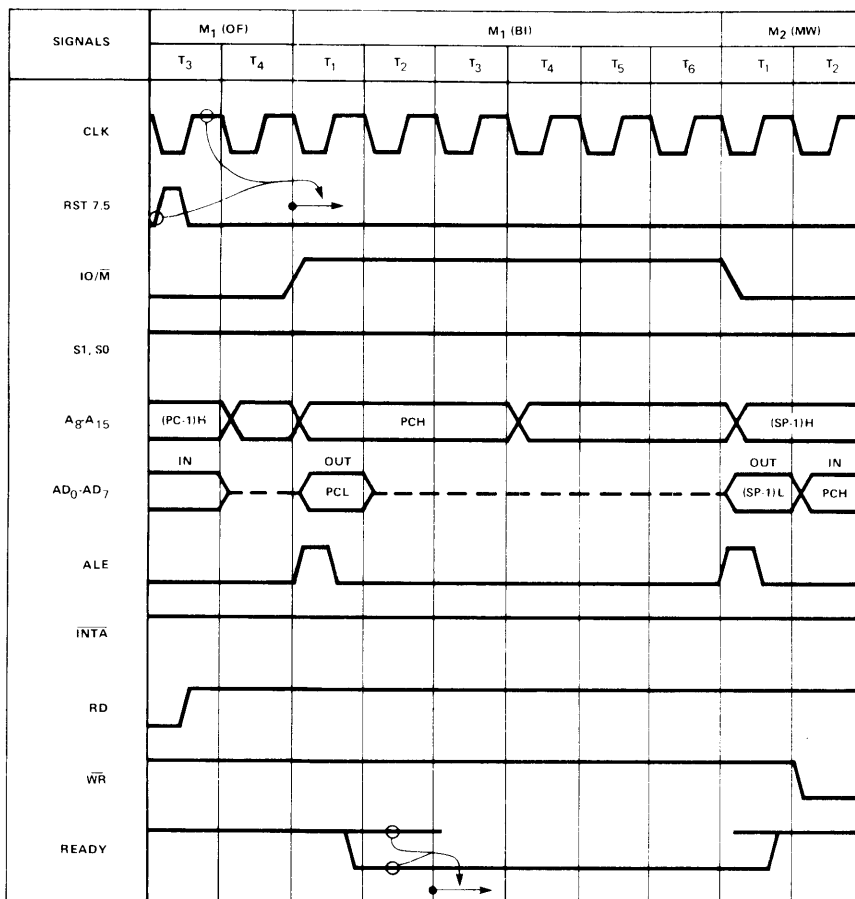


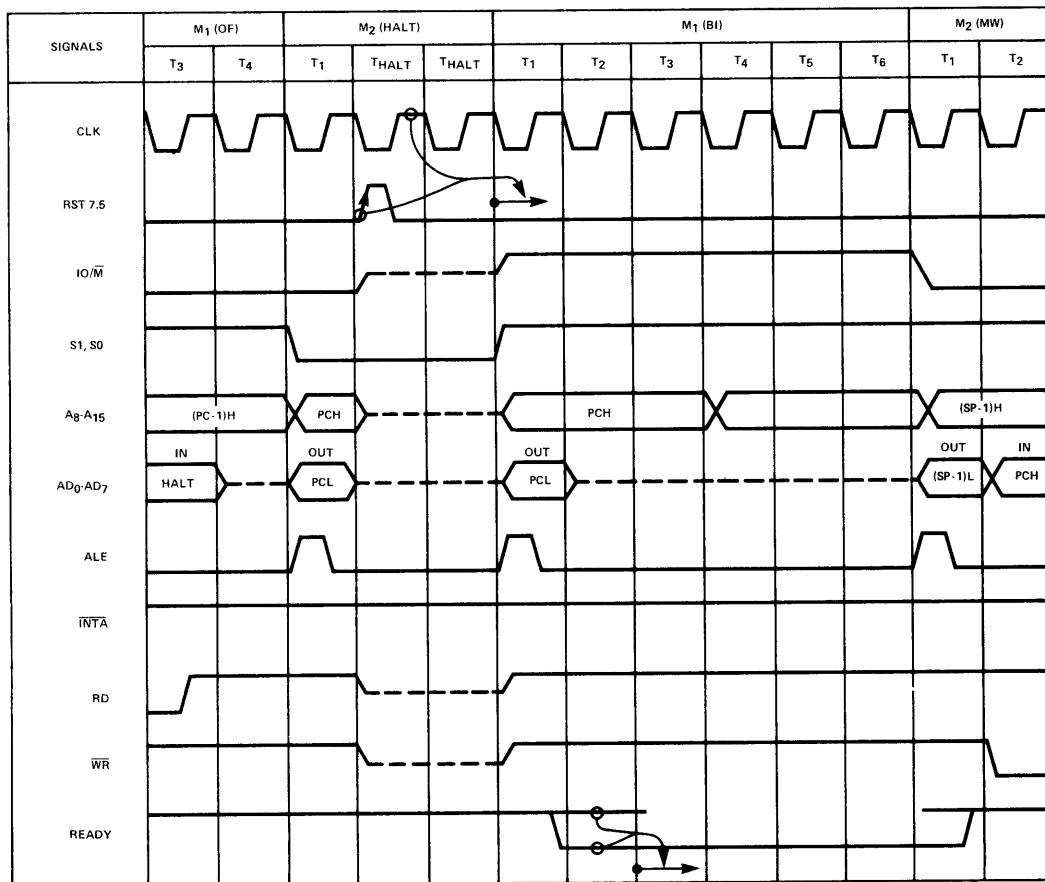
FIGURE 2-19 RST 7.5 BUS IDLE MACHINE CYCLE

FUNCTIONAL DESCRIPTION

Figure 2-20 illustrates the BI cycle generated in response to RST 7.5 when a HALT instruction has just been executed and the CPU is in the T_{HALT} state, with its various signals floating. There are only two ways the processor can completely exit the T_{HALT} state, as shown in Figure 2-11. The first way is for RESET to occur, which always forces the 8085A to T_{RESET} . The second way to exit T_{HALT} permanently is for a valid interrupt to occur, which will cause the CPU to disable further interrupts by resetting INTE FF , and to then proceed to $M_1 \cdot T_1$ of the next instruction. When the HOLD input is activated, the CPU will exit T_{HALT} for the duration of T_{HOLD} and then return to T_{HALT} .

In Figure 2-20 the RST 7.5 line is pulsed during T_{HALT} . Since RST 7.5 is a rising-edge-triggered interrupt, it will set an internal latch which is sampled during $\text{CLK} = "1"$ of every T_{HALT} state (as well as during $\text{CLK} = "1"$ two T states before any $M_1 \cdot T_1$.) The fact that the latched interrupt was high (assuming that $\text{INTE FF} = 1$ and the RST 7.5 mask = 0) will force the CPU to exit the T_{HALT} state at the end of the next CLK period, and to enter $M_1 \cdot T_1$.

This completes our analysis of the timing of each of the seven types of machine cycles.



**FIGURE 2-20 HALT STATE AND BUS IDLE MACHINE CYCLE
RST 7.5 TERMINATES T_{HALT} STATE**

2.3.6 HOLD and HALT States

The 8085A uses the T_{HOLD} state to momentarily cease executing machine cycles, allowing external devices to gain control of the bus and perform DMA cycles. The processor internally latches the state of the HOLD line and the unmasked interrupts during $CLK = "1"$ of every T_{HALT} state. If the internal latched HOLD signal is high during $CLK = "1"$ of any T_{HALT} state, the CPU will exit T_{HALT} and enter T_{HOLD} on the following $CLK = "1"$. As shown in Figure 2-21 this will occur even if a valid interrupt occurs simultaneously with the HOLD signal.

The state of the HOLD and the unmasked interrupt lines is latched internally during $CLK = 1$ of each T_{HOLD} state as well as during T_{HALT} states. If the internal latched HOLD signal is low during $CLK = 1$, the CPU will exit T_{HOLD} and enter T_{HALT} on the following $CLK = 1$.

The 8085A accepts the first unmasked, enabled interrupt sampled; thereafter, all interrupt sampling is inhibited. The interrupt thus accepted will inevitably be executed when the CPU exits the HOLD state, even at the expense of holding off higher-priority interrupts (including TRAP). (See Figure 2-22.)

When the CPU is not in T_{HALT} or T_{HOLD} , it internally latches the HOLD line only during $CLK = 1$ of the last state before T_3 (T_2 or T_{WAIT}) and during $CLK = 1$ of the last state before T_5 (T_4 of a six T-state M_1). If the internal latched HOLD signal is high during the next $CLK = 1$, the CPU will enter T_{HOLD} after the following clock. When the CPU is not in T_{HALT} or T_{HOLD} , it will internally latch the state of the unmasked interrupts only during CLK of the next to the last state before each $M_1 \cdot T_1$.

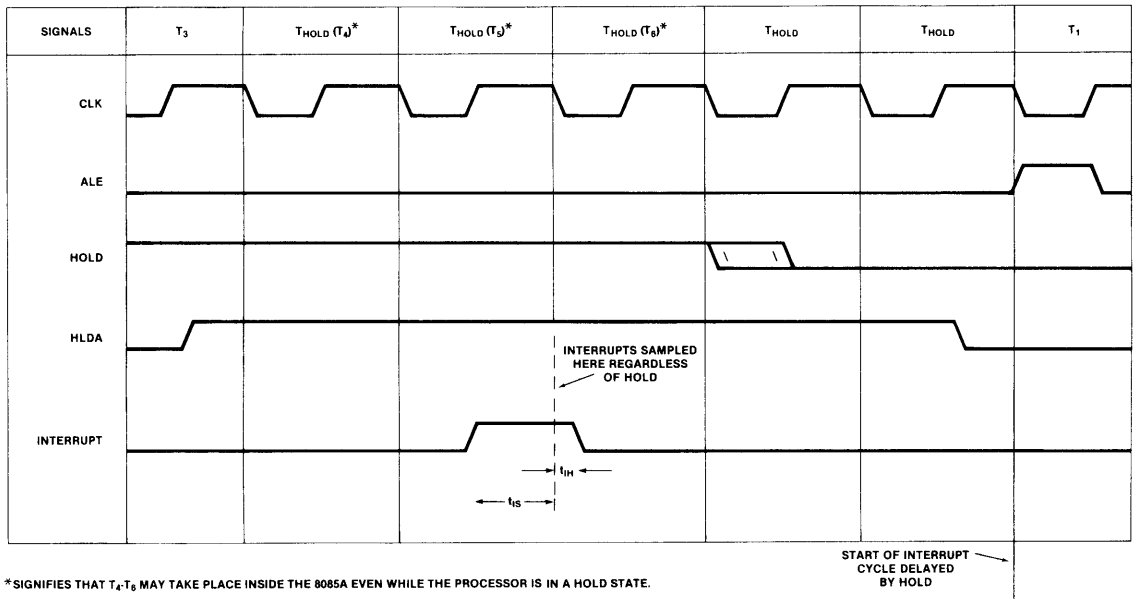


FIGURE 2-21 HOLD VS INTERRUPT — NON HALT

FUNCTIONAL DESCRIPTION

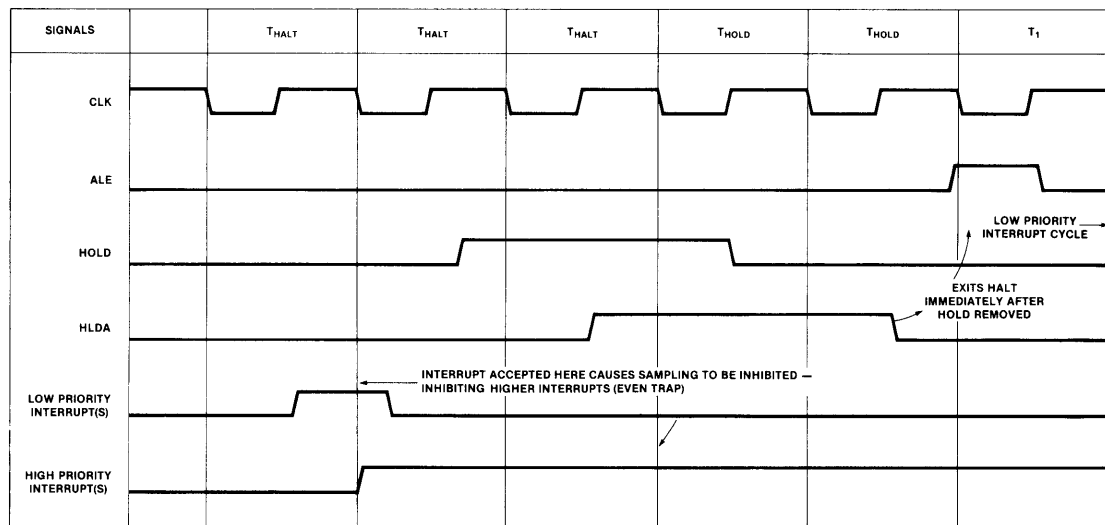


FIGURE 2-22 8085A HOLD VS INTERRUPTS — HALT MODE

2.3.7 Power On and RESET IN

The 8085A employs a special internal circuit to increase its speed. This circuit, which is called a substrate bias generator, creates a negative voltage which is used to negatively bias the substrate. The circuit employs an oscillator and a charge pump which require a certain amount of time after POWER ON to stabilize. (See Figure 2-23.)

Taking this circuit into account, the 8085A is not guaranteed to work until 10 ms after V_{CC} reaches 4.75V. For this reason, it is suggested that RESET IN be kept low during this period. Note that the 10 ms period does not include the time it takes for the power supply to reach its 4.75V level — which may be milliseconds in some systems. A simple RC network (Figure 3-6) can satisfy this requirement.

The RESET IN line is latched every CLK = 1. This latched signal is recognized by the CPU during CLK = 1 of the next T state. (See Figure 2-24.) If it is low, the CPU will issue RESET OUT and enter T_{HALT} for the next T state. RESET IN should be kept low for a minimum of three clock periods to ensure proper synchronization of the CPU. When the RESET IN signal goes high, the

CPU will enter M₁ • T₁ for the next T state. Note that the various signals and buses are floated in T_{RESET} as well as T_{HALT} and T_{HOLD}. For this reason, it is desirable to provide pull-up resistors for the main control signals (particularly WR).

Specifically, the RESET IN signal causes the following actions:

RESETS

- PROGRAM COUNTER
- INSTRUCTION REGISTER
- INTE FF
- RST 7.5 FF
- TRAP FF
- SOD FF
- MACHINE STATE FF's
- MACHINE CYCLE FF's
- INTERNALLY LATCHED FF's for HOLD, INTR, and READY

SETS

- RST 5.5 MASK
- RST 6.5 MASK
- RST 7.5 MASK

RESET IN does not explicitly change the contents of the 8085A registers (A, B, C, D, E, H, L) and the condition flags, but due to RESET IN occurring at a random time during instruction execution, the results are indeterminate.

FUNCTIONAL DESCRIPTION

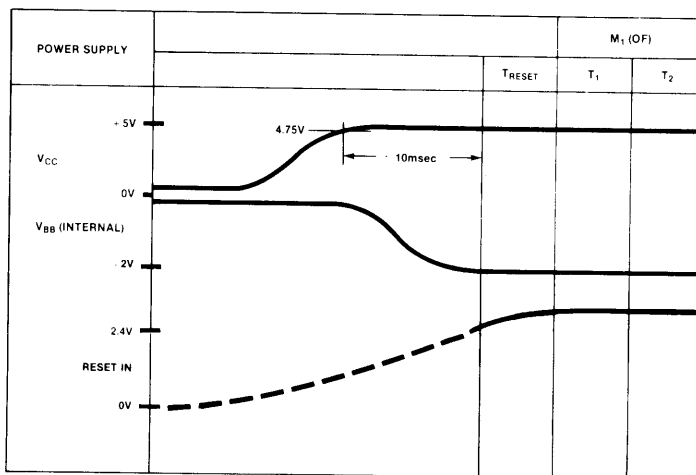


FIGURE 2-23 POWER-ON TIMING

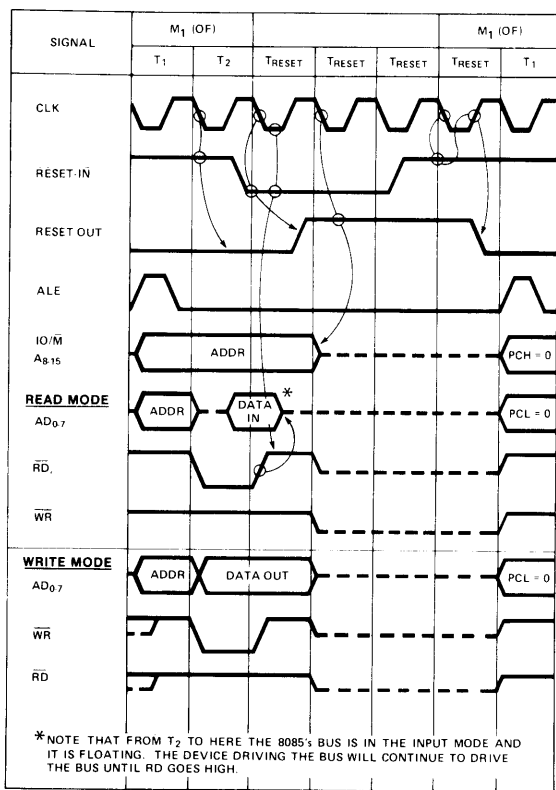


FIGURE 2-24 RESET IN TIMING

Following RESET, the 8085A will start executing instructions at location 0 with the interrupt system disabled, as shown in Figure 2-24.

Figure 2-24 also shows READ and WRITE operations being terminated by a RESET signal. Note that a RESET may prematurely terminate any READ or WRITE operation in process when the RESET occurs.

2.3.8 SID and SOD Signals:

Figure 2-25 shows the timing relationship of the SID and SOD signals to the RIM and SIM instructions. The 8085A has the ability to read the SID line into the accumulator bit 7 using RIM instructions. The state of the SID line is latched internally during T₃ · CLK = 0 of the RIM instruction. Following this, the state of the interrupt pins and masks are also transferred directly to the accumulator.

The 8085A can set the SOD flip-flop from bit 7 of the accumulator using the SIM instruction. (See Figure 2-26.) The data is transferred from the accumulator bit 7 to SOD during M₁ · T₂ · CLK = 0 of the instruction following SIM, assuming that accumulator bit 6 is a 1. Accumulator bit 6 is a "serial output enable" bit.

FUNCTIONAL DESCRIPTION

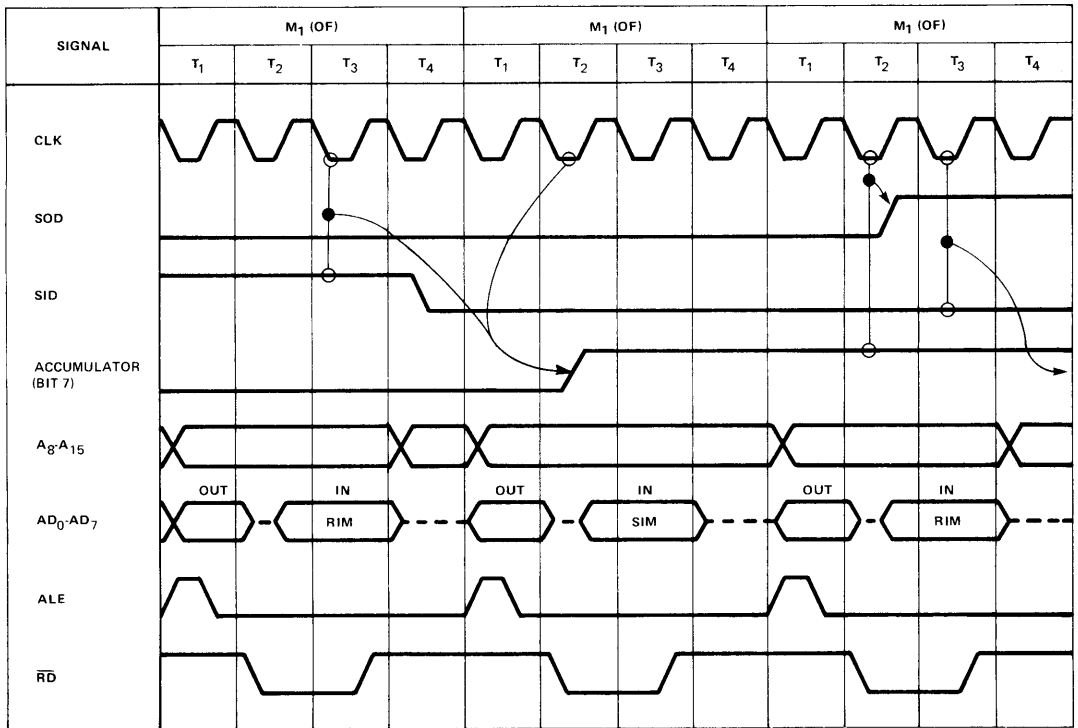
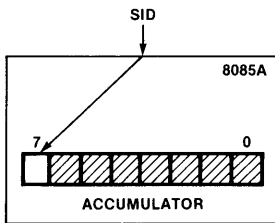


FIGURE 2-25 RELATIONSHIP OF SID AND SOD SIGNALS TO RIM AND SIM INSTRUCTIONS

EFFECT OF RIM INSTRUCTION



EFFECT OF SIM INSTRUCTION

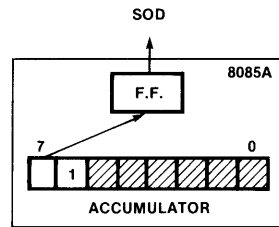


FIGURE 2-26 EFFECT OF RIM AND SIM INSTRUCTIONS

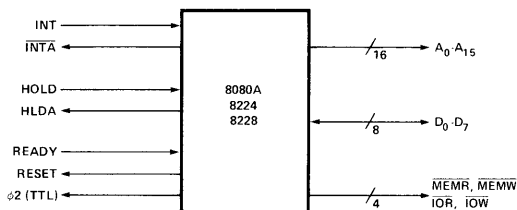
2.4 COMPARISON OF MCS-80 AND MCS-85 SYSTEM BUSES

This section compares the MCS-80 bus with the MCS-85 bus. Figure 2-28 details the signals and general timing of the two buses; the timing diagrams are drawn to the same scale (8080A clock cycle = 480 ns and 8085A clock cycle = 320 ns) to facilitate comparison.

FUNCTIONAL DESCRIPTION

MCS-80™ System Bus

The MCS-80 bus is terminated on one end by the CPU-GROUP (consisting of the 8080A, 8224, 8228) and on the other end by the various memory and I/O circuits. The following figure shows the major signals of the MCS-80 bus.



MCS-85™ System Bus

The MCS-85 bus is terminated on one end by the 8085A and the other end by various memory and I/O devices. The MCS-85 bus may be optionally de-multiplexed with an 8212 eight bit latch to provide an MCS-80 type bus. The following figure shows the major signals of the MCS-85 bus.

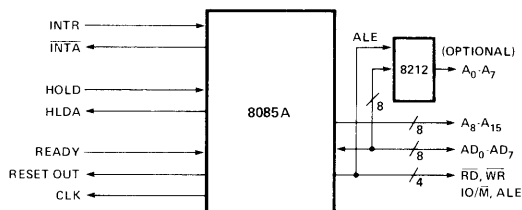


FIGURE 2-27 COMPARISON OF SYSTEM BUSES

MCS-80™ System Bus

SIGNAL(S)	FUNCTION
A_0-A_{15}	The 16 lines of the address bus identify a memory or I/O location for a data transfer operation.
D_0-D_7	The 8 lines of the data bus are used for the parallel transfer of data between two devices.
\overline{MEMR} , \overline{MEMW} , \overline{IOR} , \overline{IOW} , \overline{INTA}	These five control lines (MEMORY READ, MEMORY WRITE, I/O READ, I/O WRITE, and INTERRUPT ACKNOWLEDGE) identify the type and timing of a data transfer operation.
READY, RESET, HOLD, HLDA, ϕ_2 (TTL), INT	These signals are used for the synchronization of slow speed memories, system reset, DMA, system timing, and CPU interrupt.

MCS-85™ System Bus

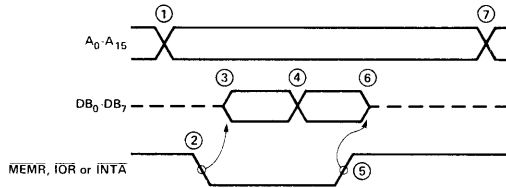
SIGNAL(S)	FUNCTION
A_8-A_{15}	These are the high order eight bits of the address, and are used to identify a memory or I/O location for a data transfer cycle.
AD_0-AD_7	These eight lines serve a dual function. During the beginning of a data transfer operation, these lines carry the low order eight bits of the address bus. During the remainder of the cycle, these lines are used for the parallel transfer of data between two devices.
\overline{RD} , \overline{WR} , \overline{INTA}	These signals identify the type and timing of a data transfer cycle.
$\overline{IO/\overline{M}}$	The I/O/MEMORY line identifies a data transfer as being in the I/O address space or the memory address space.
ALE	ADDRESS LATCH ENABLE enables the latching of the A_0-A_7 signals.
READY, RESET OUT, HOLD, HLDA, CLK, INTR	These signals are used for the synchronization of slow speed memories, system reset, DMA, system timing and CPU interrupt.

FIGURE 2-28 COMPARISON OF SYSTEM BUSES

FUNCTIONAL DESCRIPTION

MCS-80™ System Bus for READ CYCLE

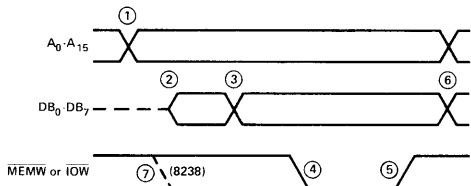
The basic timing of the MCS-80 BUS for a READ CYCLE is as follows:



The MCS-80 first presents the address (1) and shortly thereafter the control signal (2). The data bus, which was in the high impedance state, is driven by the selected device (3). The selected device eventually presents the valid data to the processor (4). The processor raises the control signal (5), which causes the selected device to put the data bus in the high impedance state (6). The processor then changes the address (7) for the start of the next data transfer.

MCS-80™ System Bus for WRITE CYCLE

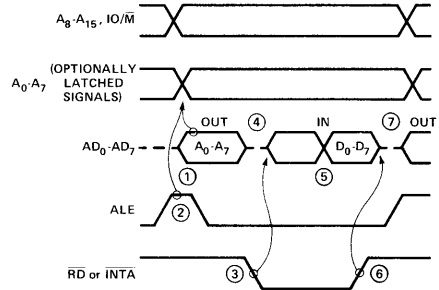
The basic timing of the MCS-80 BUS for a WRITE CYCLE is as follows:



The MCS-80 first presents the address (1), then enables the data bus driver (2), and later presents the data (3). Shortly thereafter, the MCS-80 drops the control signal (4) for an interval of time and then raises the signal (5). The MCS-80 then changes the address (6) in preparation for the next data transfer. The advance write signal of the 8238 is also shown (7).

MCS-85™ System Bus for READ CYCLE

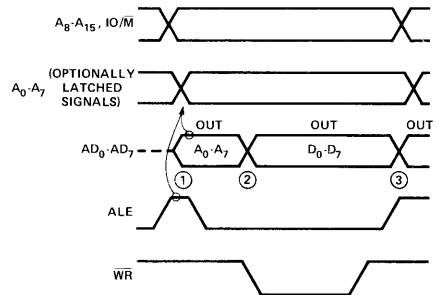
The basic timing of the MCS-85 BUS for a READ CYCLE is as follows:



At the beginning of the READ cycle, the 8085A sends out all 16 bits of address (1). This is followed by ALE (2) which causes the lower eight bits of address to be latched in either the 8155/56, 8355, 8755A, or in an external 8212. RD is then dropped (3) by the 8085A. The data bus is then tri-stated by the 8085A in preparation for the selected device driving the bus (4); the selected device will continue to drive the bus with valid data (5), until RD is raised (6) by the 8085A. At the end of the READ CYCLE (7), the address and data lines are changed in preparation for the next cycle.

MCS-85™ System Bus for WRITE CYCLE

The basic timing of the MCS-85 BUS for a WRITE CYCLE is as follows:



The timing of the WRITE CYCLE is identical to the MCS-85 READ CYCLE with the exception of the AD₀-AD₇ lines. At the beginning of the cycle (1), the low order eight bits of address are on AD₀-AD₇. After ALE drops, the eight bits of data (2) are put on AD₀-AD₇. They are removed (3) at the end of the WRITE CYCLE, in anticipation of the next data transfer.

FIGURE 2-28 (Continued) COMPARISON OF SYSTEM BUSES

The following observations of the two buses can be made:

1. The access times from address leaving the processor to returning data are almost identical, even though the 8085A is operating 50% faster than the 8080.
2. With the addition of an 8212 latch to the 8085A, the basic timings of the two systems are very similar.
3. The 8085A has more time for address setup to \overline{RD} than the 8080.
4. The MCS-80 has a wider \overline{RD} signal, but a narrower \overline{WR} signal than the 8085A.
5. The MCS-80 provides stable data setup to the leading and trailing edges of \overline{WR} , while the 8085 provides stable data setup to only the trailing edge of \overline{WR} .
6. The MCS-80 control signals have different widths and occur at different points in the machine cycle, while the 8085A control signals have identical timing.
7. While not shown on the chart, the MCS-80 data and address hold times are adversely affected by the processor preparing to enter the HOLD state. The 8085A has identical timing regardless of entering HOLD.
8. Also not shown on the chart is the fact that all output signals of the 8085A have $-400\mu\text{a}$ of source current and 2.0 ma of sink current. The 8085A also has input voltage levels of $V_{IL} = 0.8\text{V}$ and $V_{IH} = 2.0\text{V}$.

CONCLUSION:

The preceding discussion has clearly shown that the MCS-85 bus satisfies the two restrictions of COMPATIBILITY and SPEED. It is compatible because it requires only an 8212 latch to generate an MCS-80 type bus. If the four control signals $\overline{\text{MEMR}}$, $\overline{\text{MEMW}}$, $\overline{\text{IOR}}$ and $\overline{\text{IOW}}$ are desired, they can be generated from \overline{RD} , \overline{WR} ,

and $\overline{\text{IO}/\overline{\text{M}}}$ with a decoder or a few gates. The MCS-85 bus is also fast. While running at 3MHz, the 8085A generates better timing signals than the MCS-80 does at 2MHz. Furthermore, the multiplexed bus structure doesn't slow the 8085A down, because it is using the internal states to overlap the fetch and execution portions of different machine cycles. Finally, the MCS-85 can be slowed down or sped up considerably, while still providing reasonable timing.

TO USE. The \overline{RD} , \overline{WR} , and $\overline{\text{INTA}}$ control signals all have identical timing, which isn't affected by the CPU preparing to enter the HOLD state. Furthermore, the address and data bus have good setup and hold times relative to the control signals. The voltage and current levels for the interface signals will all drive buses of up to 40 MOS devices, or 1 schottky TTL device.

The MCS-85 system bus is also EFFICIENT. Efficiency is the reason that the lower eight address lines are multiplexed with the data bus. Every chip that needs to use both A_0-A_7 and D_0-D_7 saves 7 pins (the eighth pin is used for ALE) on the interface to the processor. That means that 7 more pins per part are available to either add features to the part or to use a smaller package in some cases. In the three chip system shown in Figure 3-6, the use of the MCS-85 bus saves $3 \times 7 = 21$ pins, which are used for extra I/O and interrupt lines. A further advantage of the MCS-85 bus is apparent in Figure 3-7, which shows a printed circuit layout of the circuit in Figure 3-6. The reduced number of pins and the fact that compatible pinouts were used, provides for an extremely compact, simple, and efficient printed circuit. Notice that great care was taken when the pinouts were assigned to ensure that the signals would flow easily from chip to chip to chip.