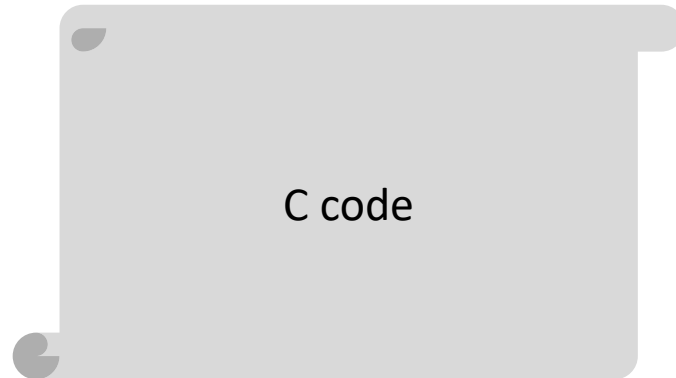


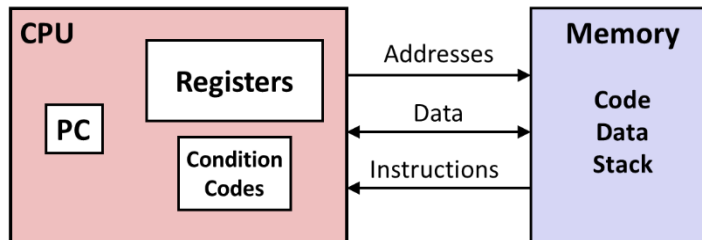
Levels of Abstraction

C programmer

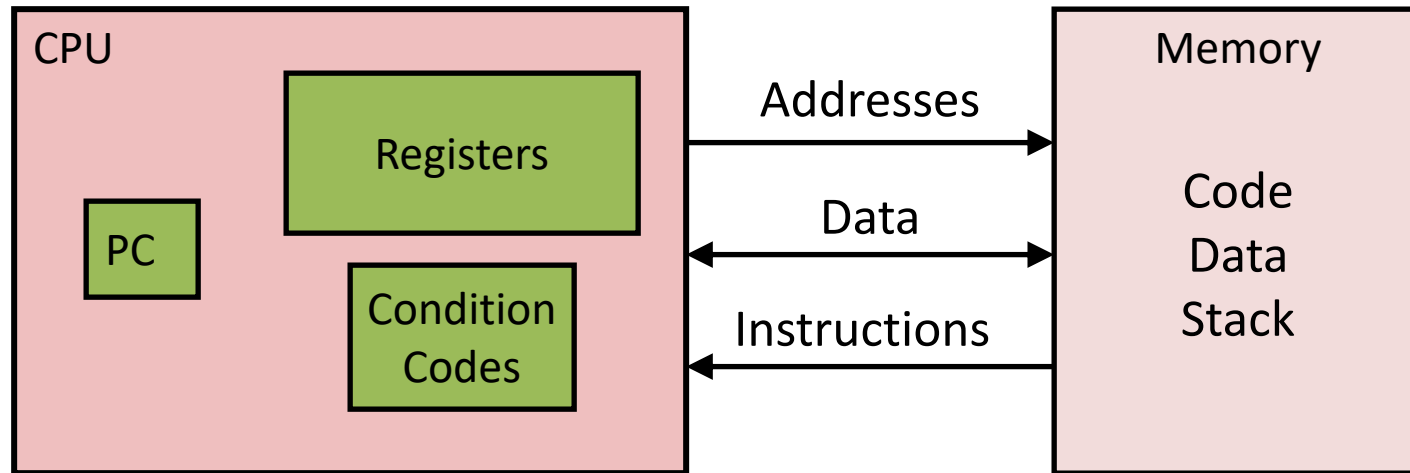


Nice clean layers,
but beware...

Assembly programmer



Assembly/Machine Code View



Programmer-Visible State

– PC: Program counter

- Address of next instruction
- Called “RIP” (x86-64)

– Register file

- Heavily used program data

– Condition codes

- Store status information about most recent arithmetic or logical operation
- Used for conditional branching

– Memory

- Byte addressable array
- Code and user data
- Stack to support procedures

8085 Architecture & Its Assembly language programming

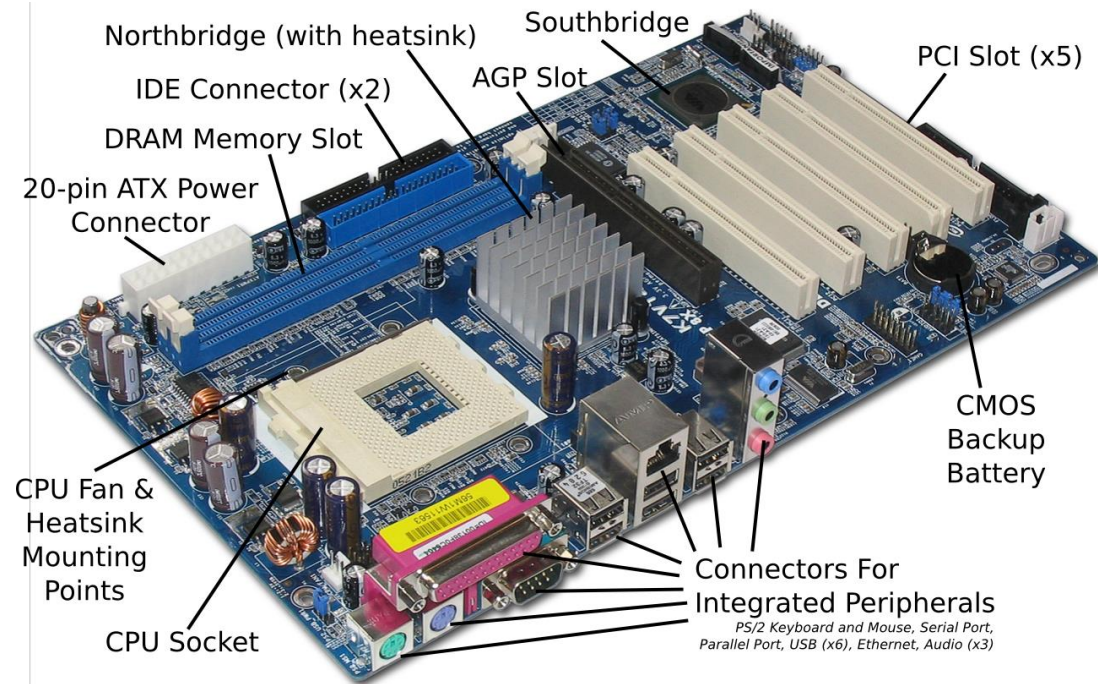
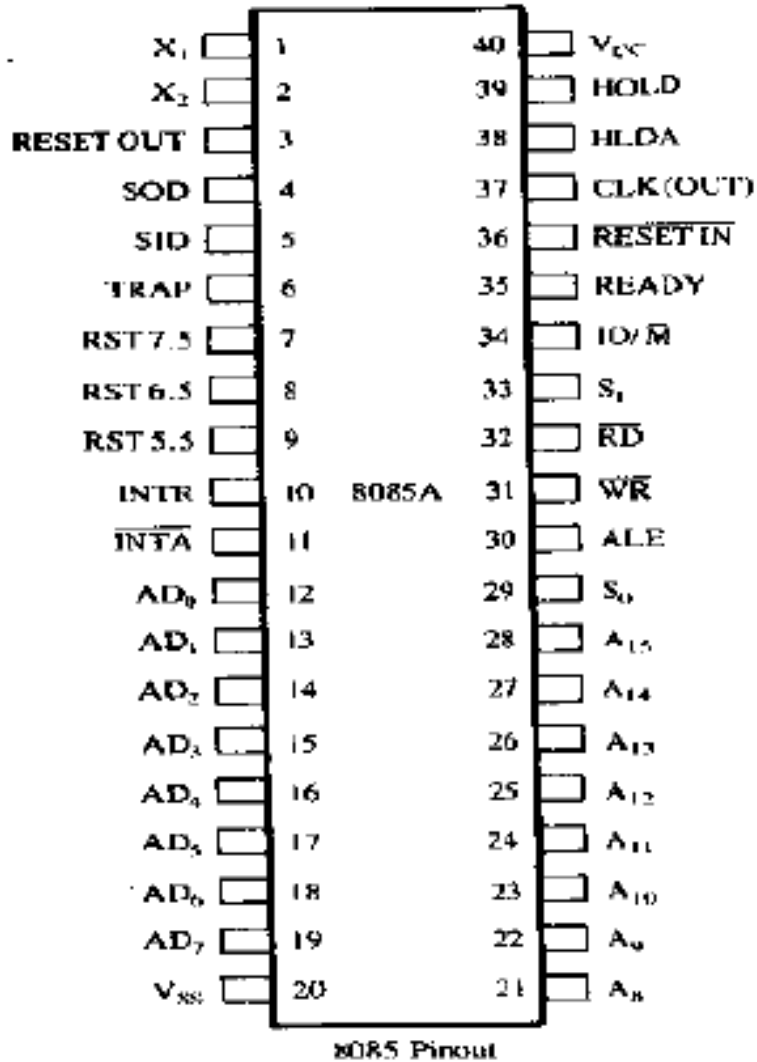
Outline

- 8085 Era and Features
- 8085
 - Block diagram (Data Path)
 - Bus Structure
 - Register Structure
- Instruction Set of 8085
- Sample program of 8085

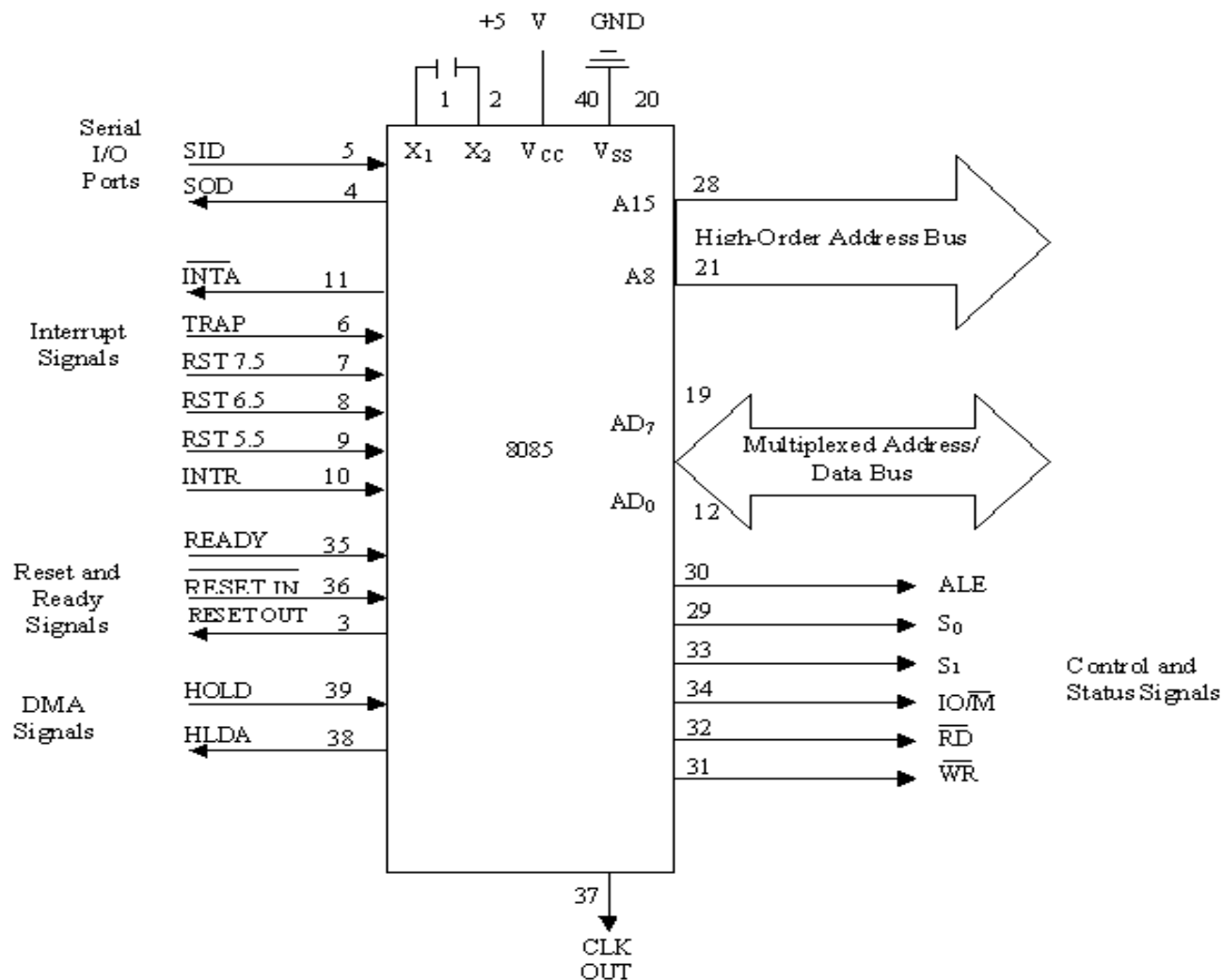
8085 Microprocessor

- 8 Bit CPU
- 3-6Mhz
- Simpler design
- ISA = Pre x86 design (Semi CISC)
- 40 Pin Dual line Package
- 16 bit address
- 6 registers: B, C, D, E, H,L
- Accumulator 8 bit

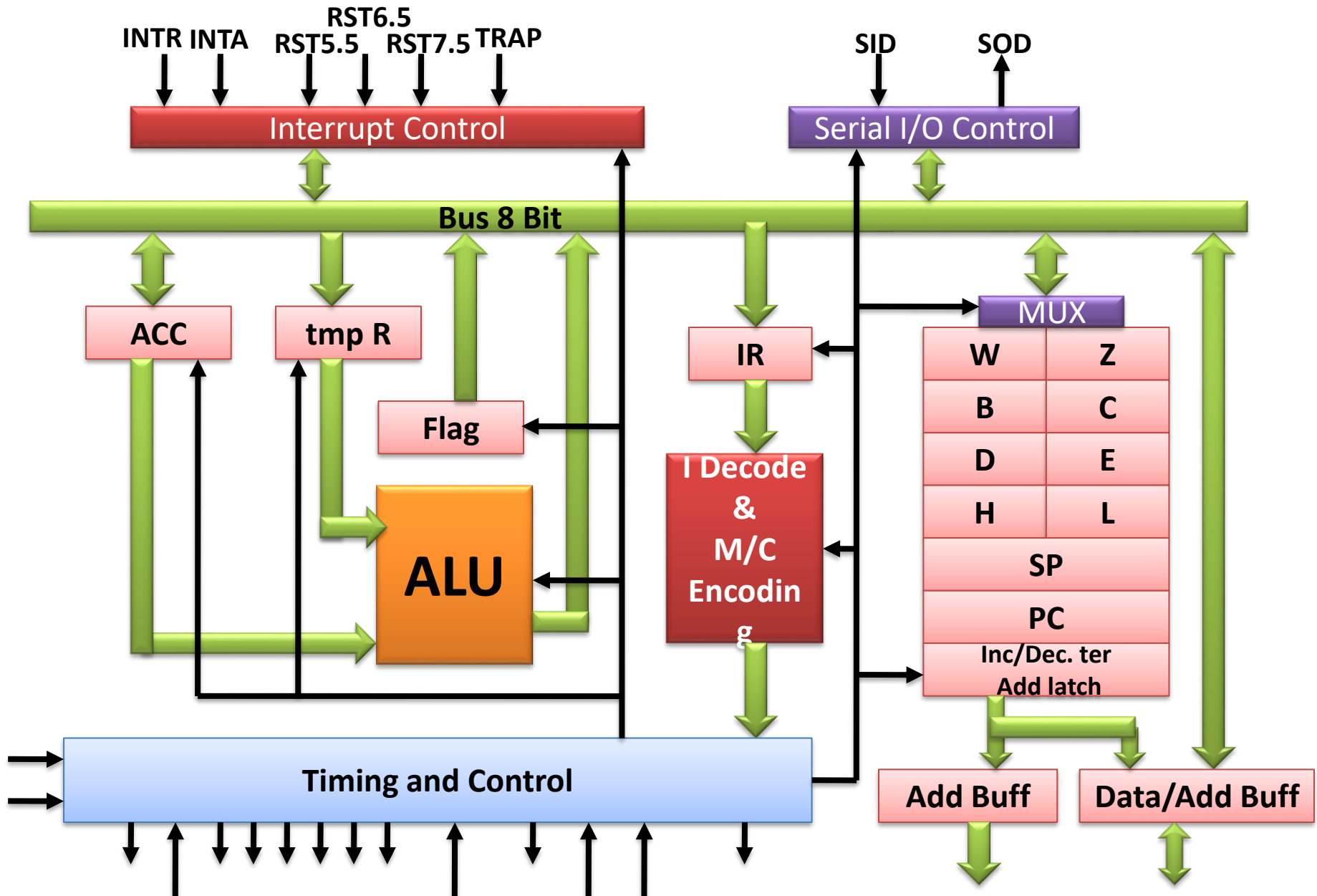
8085-Pin-diagram



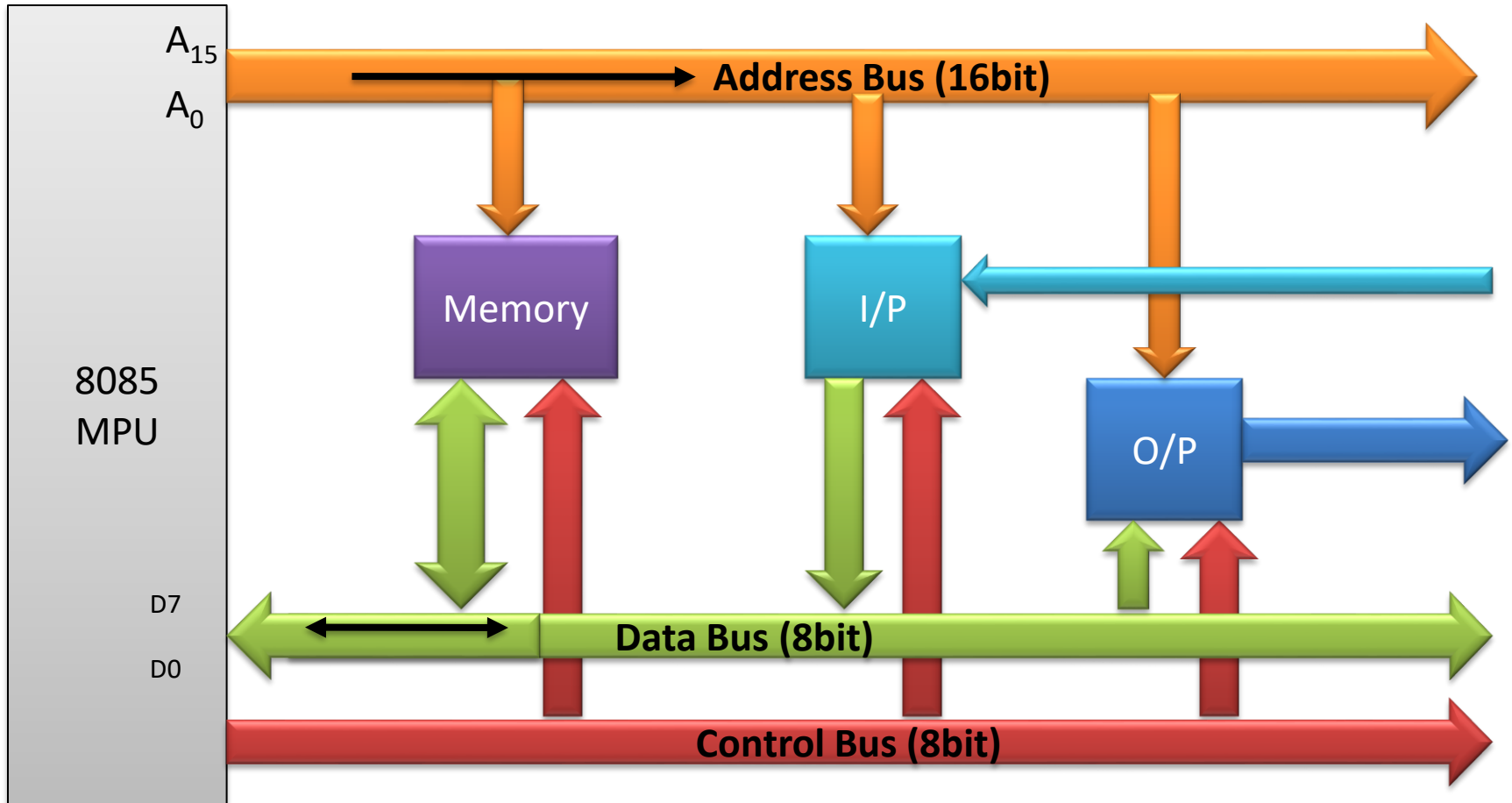
Functional block diagram



8085 Microprocessor Architecture



The 8085 Bus Structure



8085 Bus Structure

- Address Bus : Consists of 16 address lines: $A_0 - A_{15}$
 - Address locations: 0000 (hex) – FFFF (hex)
 - Can access 64K (= 2^{16}) bytes of memory, each byte has 8 bits
 - Can access $64K \times 8$ bits of memory
 - Use memory to map I/O, Same instructions to use for accessing I/O devices and memory
- Data Bus : Consists of 8 data lines: $D_0 - D_7$
 - Operates in bidirectional mode
 - The data bits are sent from the MPU to I/O & vice versa
 - Data range: 00 (hex) – FF (hex)
- Control Bus:
 - Consists of various lines carrying the control signals such as read / write enable, flag bits

8085 Registers

- Registers:
 - Six general purpose 8-bit registers: B, C, D, E, H, L
 - Combined as register pairs to perform 16-bit operations: BC, DE, HL
 - Registers are programmable (load, move, etc.)
- Stack Pointer (SP)
- Accumulator & Flag Register
 - (Zero, Sign, Carry, Parity, AuxCarry)
- Program Counter (PC)
 - Contains the memory address (16 bits) of the instruction that will be executed in the next step.

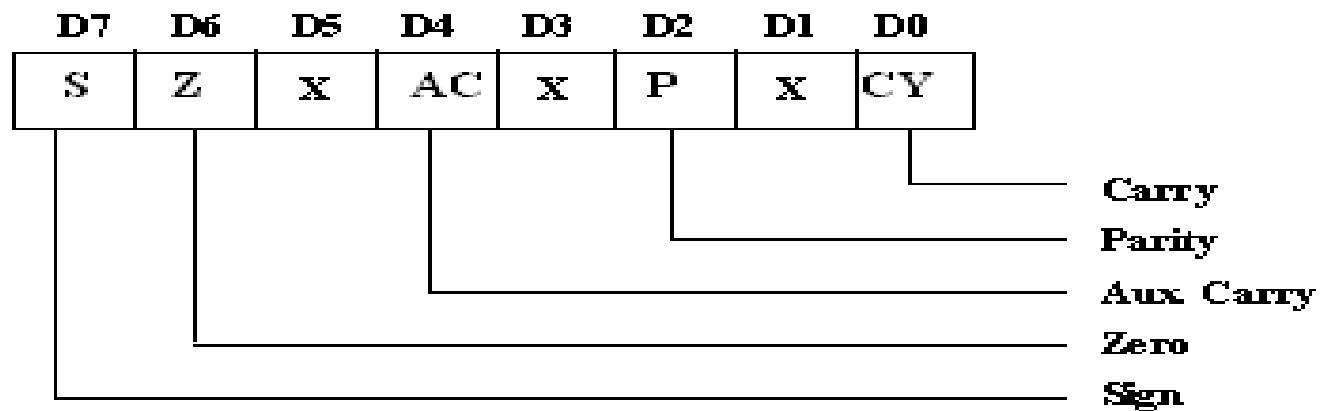
B	C
D	E
H	L
SP	
PC	

Register Addressing

- The most common form of data addressing.
 - once register names learned, easiest to apply.
- The microprocessor contains these 8-bit register names used with register addressing: AH, AL, BH, BL, CH, CL, DH, and DL.
- 16-bit register names: AX, BX, CX, DX, SP, BP, SI, and DI.

- In 80386 & above, extended 32-bit register names are: EAX, EBX, ECX, EDX, ESP, EBP, EDI, and ESI.
- 64-bit mode register names are: RAX, RBX, RCX, RDX, RSP, RBP, RDI, RSI, and R8 through R15.
- Important for instructions to use registers that are the same size.
 - *never* mix an 8-bit with a 16-bit register, an 8- or a 16-bit register with a 32-bit register
 - this is not allowed by the microprocessor and results in an error when assembled

Flag Register



Status Signals

IO/M	S1	S0	OPERATION
0	1	1	Opcode fetch
0	1	0	Memory read
0	0	1	Memory write
1	1	0	I/O read
1	0	1	I/O write
1	1	0	Interrupt acknowledge
Z	0	1	Halt
Z	x	x	Hold
Z	x	x	Reset

Reset Signal

- **RESET IN :** When this signal goes low,
 - the program counter (PC) is set to Zero,
 - μ p is reset
 - The data and address buses and the control lines are 3-stated during RESET
 - The CPU is held in the reset condition as long as RESET- IN is applied
- **RESET OUT:** This signal indicates that μ p is being reset.
 - This signal can be used to reset other devices.
 - The signal is synchronized to the processor clock and lasts an integral number of clock periods.
 - .

Serial communication

- **Serial communication Signal**
 - **SID - Serial Input Data Line:** The data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.
 - **SOD – Serial Output Data Line:** The SIM instruction loads the value of bit 7 of the accumulator into SOD latch if bit 6 (SOE) of the accumulator is 1.

DMA Signals

- **HOLD: Indicates** that another master is requesting the use of the address and data buses.
 - The CPU, upon receiving the hold request, will relinquish the use of the bus as soon as the completion of the current bus transfer.
 - Internal processing can continue. The processor can regain the bus only after the HOLD is removed.
 - When the HOLD is acknowledged, the Address, Data RD, WR and IO/M lines are 3-stated.
- **HLDA: Hold Acknowledge: Indicates that the CPU has received the HOLD request and that it will relinquish the bus in the next clock cycle.**
 - HLDA goes low after the Hold request is removed. The CPU takes the bus one half-clock cycle after HLDA goes low.

Ready Signal

- **READY:** This signal Synchronizes the fast CPU and the slow memory, peripherals.
- If READY is high during a read or write cycle, it indicates that the memory or peripheral is ready to send or receive data.
- If READY is low, the CPU will wait an integral number of clock cycle for READY to go high before completing the read or write cycle.
- READY must conform to specified setup and hold times.

Interrupts

- 8085 has 5 interrupts priority (from **lowest to highest**):
- **INTR:** (maskable and non-vectored interrupt). When the interrupt occurs, the processor fetches from the bus one instruction, usually one of these instructions:
 - One of the 8 RST instructions (**RST0 - RST7**). The processor saves current program counter into stack and branches to memory location $N * 8$ (where N is a 3-bit number from 0 to 7 supplied with the RST instruction).
 - **CALL: (3 byte instruction)**. The processor calls the subroutine, address of which is specified in the second and third bytes of the instruction.
- **RST5.5:** (maskable interrupt). When this interrupt is received the processor saves the contents of the PC register into stack and branches to 2CH address.
- **RST6.5:** (maskable interrupt) When this interrupt is received the processor saves the contents of the PC register into stack and branches to 34H address
- **RST7.5** (maskable interrupt) When this interrupt is received the processor saves the contents of the PC register into stack and branches to 3CH (hexadecimal) address
- **TRAP:** (non-maskable interrupt) When this interrupt **is** received the processor saves the contents of the PC register into stack and branches to 24H address.
- All maskable interrupts can be enabled or disabled using EI and DI instructions. RST 5.5, RST6.5 and RST7.5 interrupts can be enabled or disabled individually using SIM instruction

8085 Non-Vectored Interrupt Process

- The interrupt process should be enabled using the EI instruction.
- The 8085 checks for an interrupt during the execution of every instruction.
- If INTR is high, Processor completes current instruction, disables the interrupt and sends INTA (Interrupt acknowledge) signal to the device that interrupted
- INTA allows the I/O device to send a RST instruction through data bus.
- MP saves the memory location of the next instruction, on the stack and the program is transferred to 'call' location (ISR Call) specified by the RST instruction
- Microprocessor Performs the ISR. ISR must include the 'EI' instruction to enable the further interrupt within the program.
- RET instruction at the end of the ISR allows the MP to retrieve the return address from the stack and the program is transferred back to where the program was interrupted.

Interrupt vectors

- An interrupt vector is a pointer to where the ISR is stored in memory.
- All interrupts (vectored or otherwise) are mapped onto a memory area called the Interrupt Vector Table(IVT).
- The IVT is usually located in memory page 00 (0000H - 00FFH).
- Example:
 - Let a device interrupts the Microprocessor using the RST 7.5 interrupt line.
 - Because the RST 7.5 interrupt is vectored, Microprocessor knows in which memory location it has to go using a call instruction to get the ISR address.
 - RST7.5 is known as Call 003Ch to Microprocessor. Microprocessor goes to 003C location and will get a JMP instruction to the actual ISR address. The microprocessor will then, jump to the ISR location

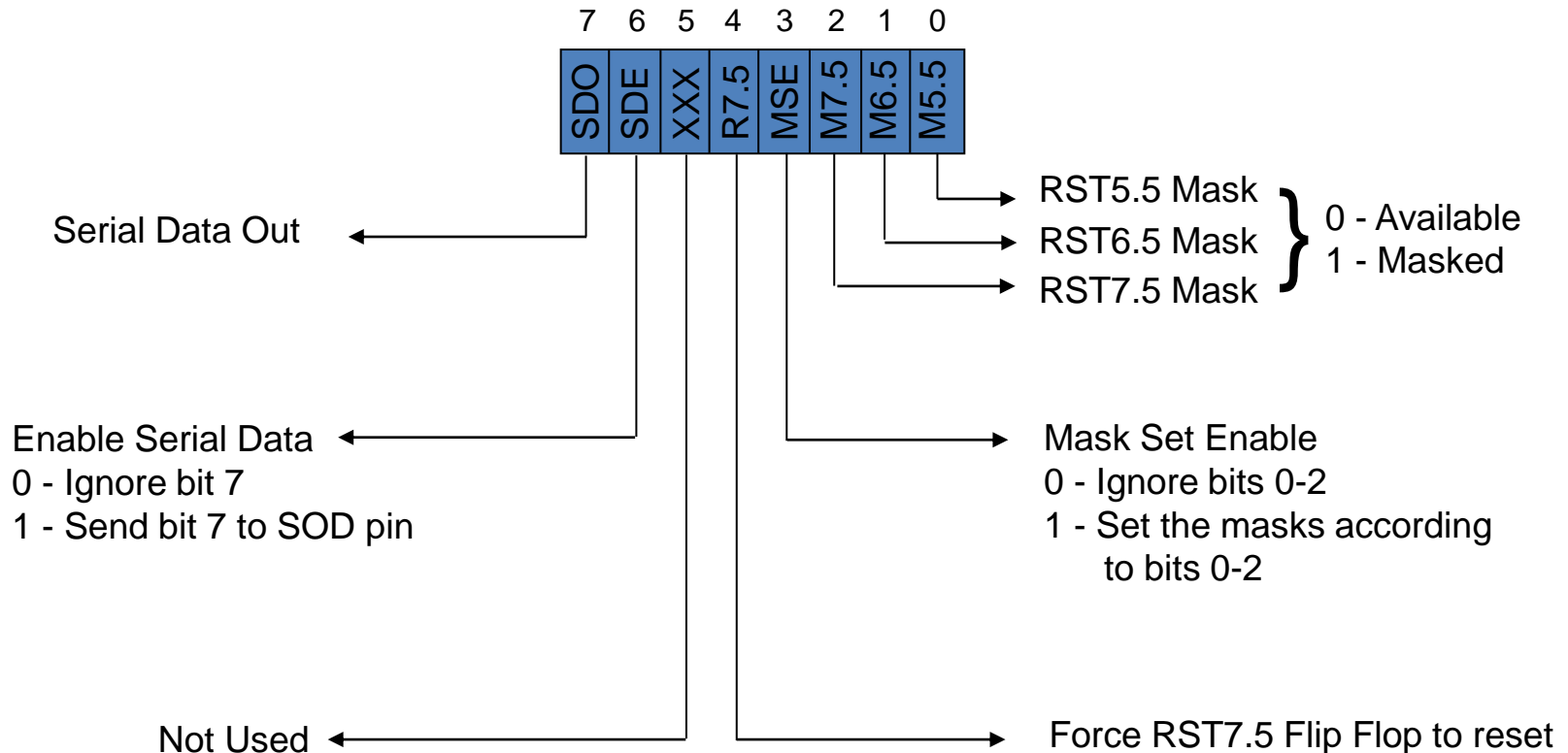
Vectored Interrupt

- The interrupt process should be enabled using the EI instruction.
- The 8085 checks for an interrupt during the execution of every instruction.
- If there is an interrupt, and if the interrupt is enabled using the interrupt mask, the microprocessor will complete the executing instruction, and reset the interrupt flip flop.
- The microprocessor then executes a call instruction that sends the execution to the appropriate location in the interrupt vector table.
- When the microprocessor executes the call instruction, it saves the address of the next instruction on the stack.
- The microprocessor jumps to the specific service routine.
- The service routine must include the instruction EI to re-enable the interrupt process.
- At the end of the service routine, the RET instruction returns the execution to where the program was interrupted.

MANIPULATING THE MASKS

- MANIPULATING THE MASKS
 - The Interrupt Enable flip flop is manipulated using the EI/DI instructions.
 - The individual masks for RST 5.5, RST 6.5 and RST 7.5 are manipulated using the SIM instruction.
 - The **SIM instruction** takes the bit pattern in the Accumulator and applies it to the interrupt mask enabling and disabling the specific interrupts.

How SIM Interprets the Accumulator

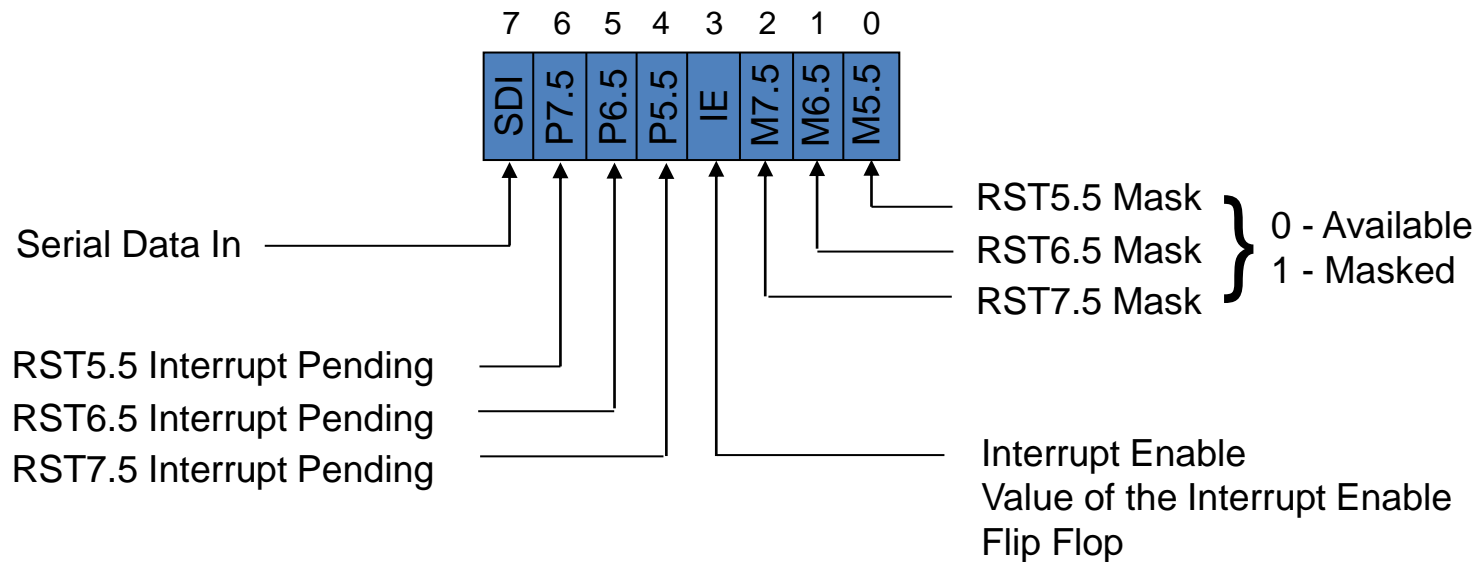


If MSE is 1

If the mask bit is 0, the interrupt is **available**.

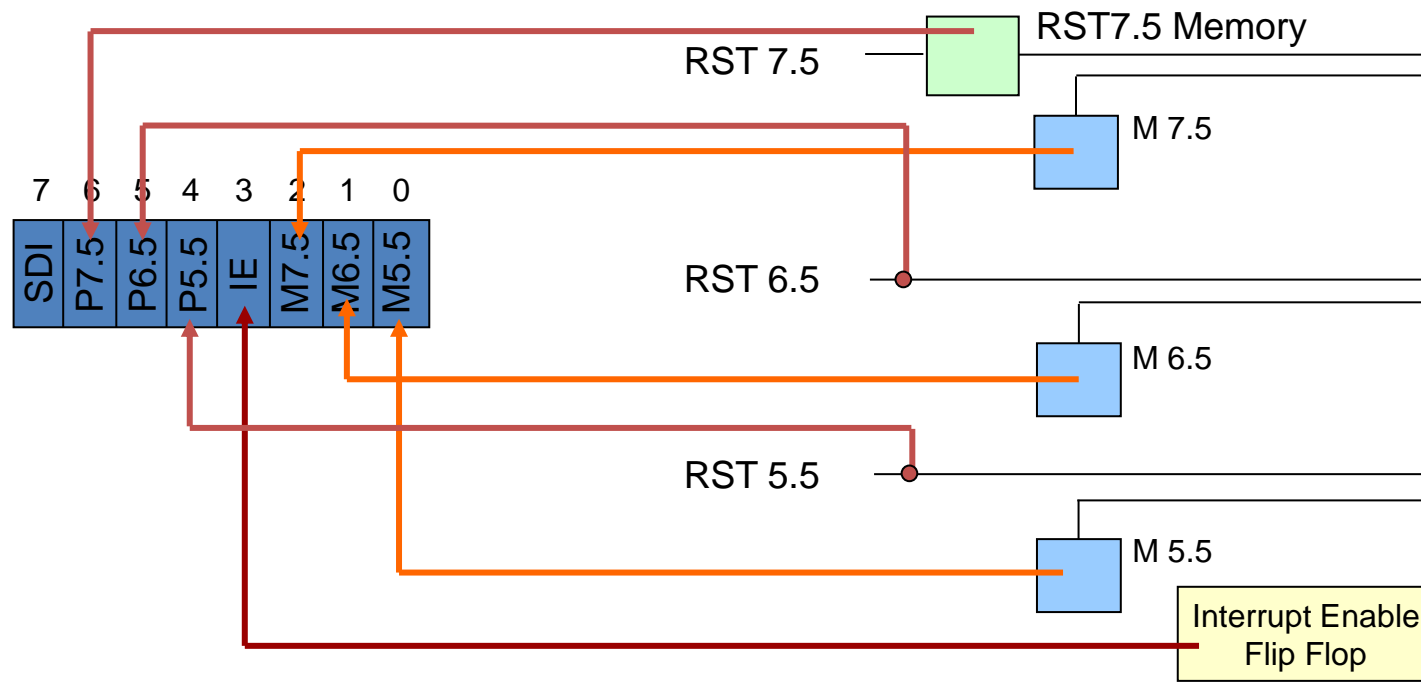
If the mask bit is 1, the interrupt is **masked**.

How RIM sets the Accumulator's different bits



Determining the Current Mask Settings

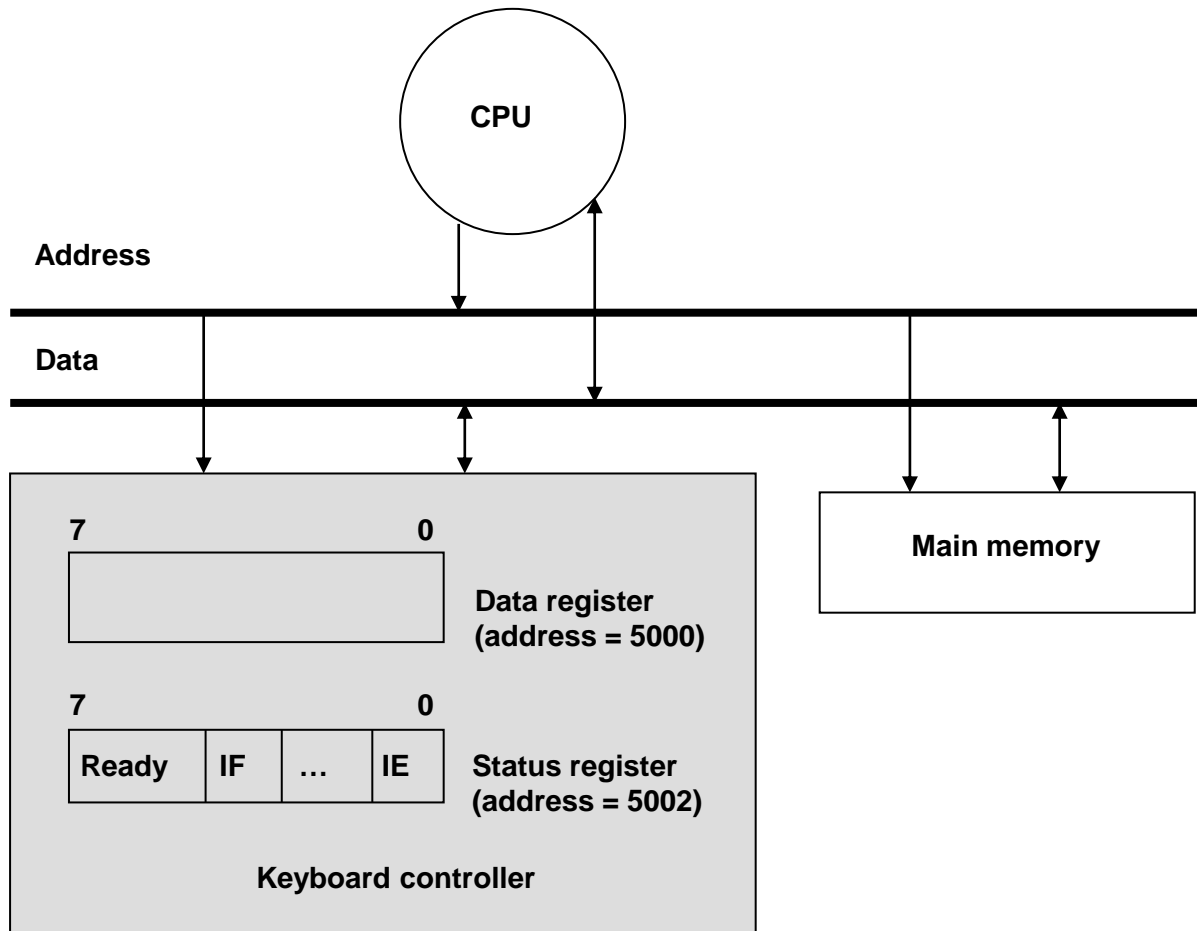
- RIM instruction: Read Interrupt Mask
 - Load the **accumulator** with an 8-bit pattern showing the status of each interrupt pin and mask.



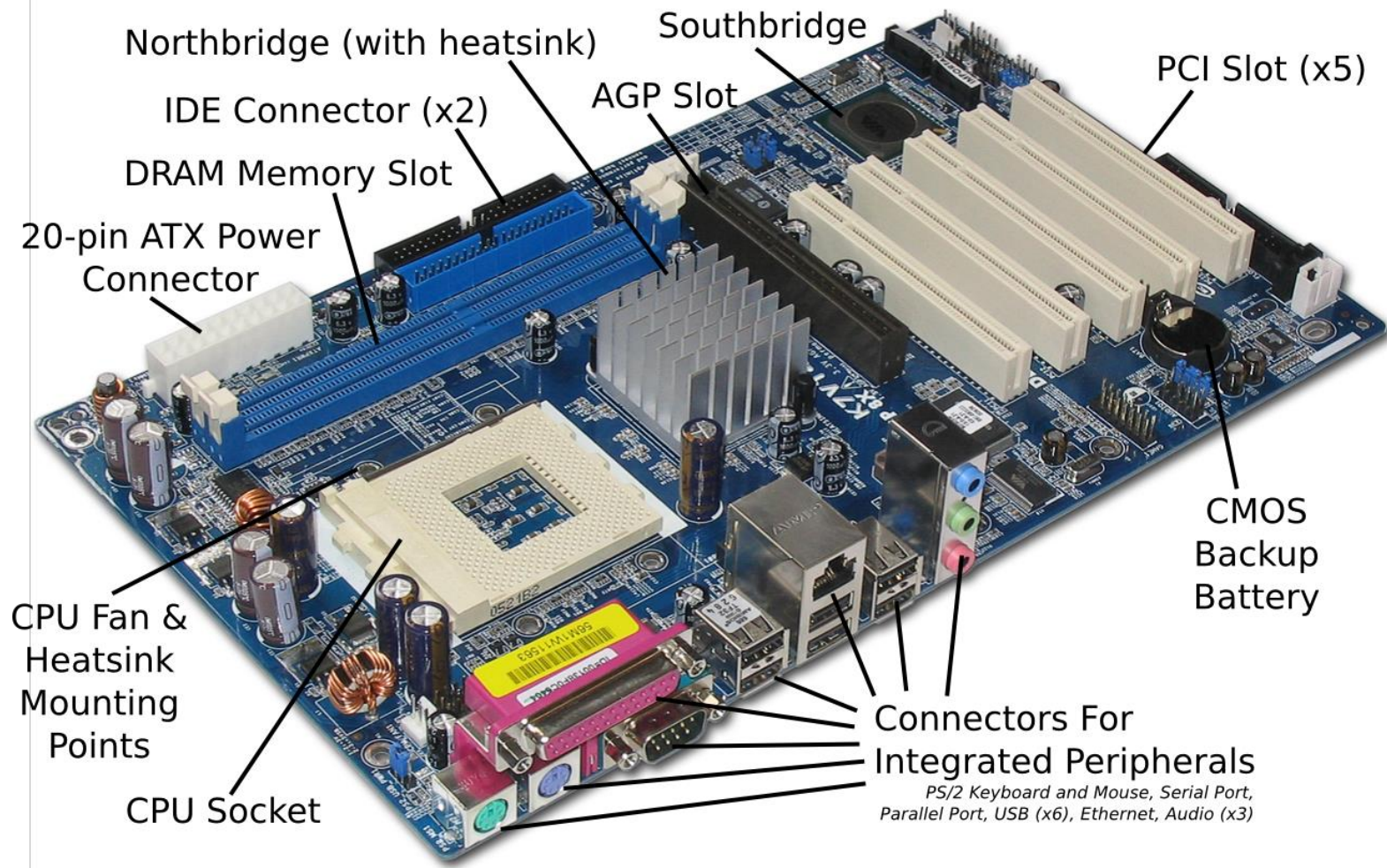
What is Memory Mapped I/O?

- Instead of having special methods for accessing the values to be read or written, just get them from memory or put them into memory.
- The device is connected directly to certain main memory locations.
- Two types of information to/from the device
 - Status
 - Value read/write

Memory Mapped I/O



Mother Board



comparison

Memory mapping	I/O mapping
16/20 bit address are provided for I/O devices	8-bit or 16-bit addresses are provided for I/O devices
The I/O ports or peripherals can be treated like memory locations and so all instructions related to memory can be used for data transmission between I/O device and processor	Only IN and OUT instructions can be used for data transfer between I/O device and processor
Data can be moved from any register to ports and vice versa	Data transfer takes place only between accumulator and ports
When memory mapping is used for I/O devices, full memory address space cannot be used for addressing memory.	Full memory space can be used for addressing memory.
⇒ Useful only for small systems where memory requirement is less	⇒ Suitable for systems which require large memory capacity
For accessing the memory mapped devices, the processor executes memory read or write cycle.	For accessing the I/O mapped devices, the processor executes I/O read or write cycle.
⇒ M / \overline{IO} is asserted high	⇒ M / \overline{IO} is asserted low

IO Mapped IO Device I/O Port Locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

Instruction format

- Based on word size (8-bit processor so byte is same as word)
 - One byte instruction
 - Ex.: Mov rd,rs
 - Binary code: 01 ddd sss
 - Mov A,B: 01 111 000 = 78h
 - Two byte instruction
 - Ex.: mvi A, data
 - Binary code: 0011 1110 data
 - Mvi A, 30: 3E 30h
 - Three byte instruction
 - Ex.: Lxi rp,data_address //rp: register pair
 - Binary code: 21 lower_byte upper byte address
 - Lxi H, 2236: 21 36 22 h

MACHINE CYCLE

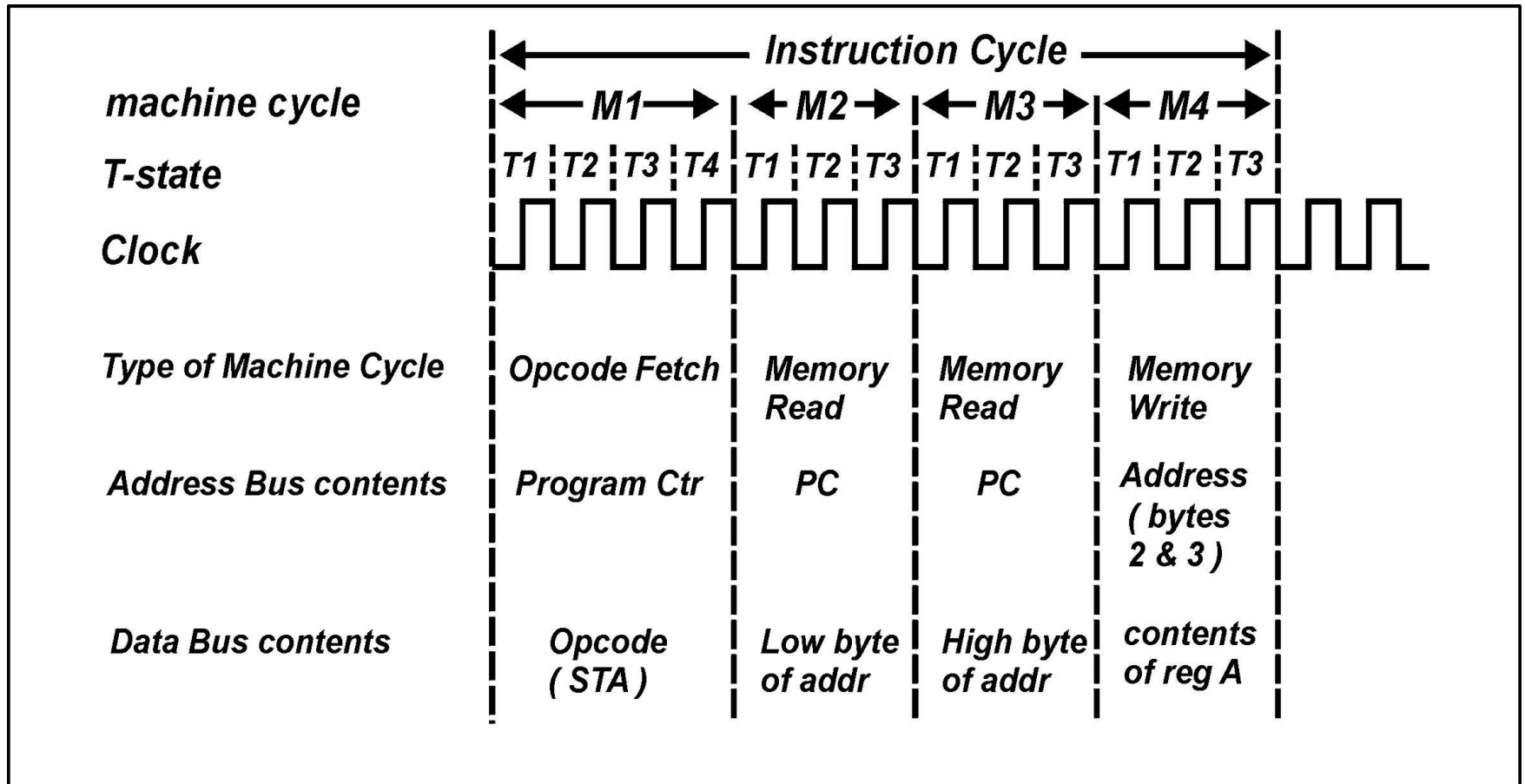
◎ The 8085 CPU can perform *seven* basic machine operations. All but the bus-idle cycle involve the transfer of data between the CPU and a peripheral device.

◎ The seven machine cycles are :

- | | |
|----------------|--|
| •Opcode Fetch | fetch the opcode of an instruction from memory |
| •Memory Read | read data stored at an addressed memory location |
| •Memory Write | write data to an addressed memory location |
| •IO Read | read data from an addressed input device |
| •IO Write | write data to an addressed output device |
| •Interrupt Ack | acknowledge an interrupt request |
| •Bus Idle | no bus operation |

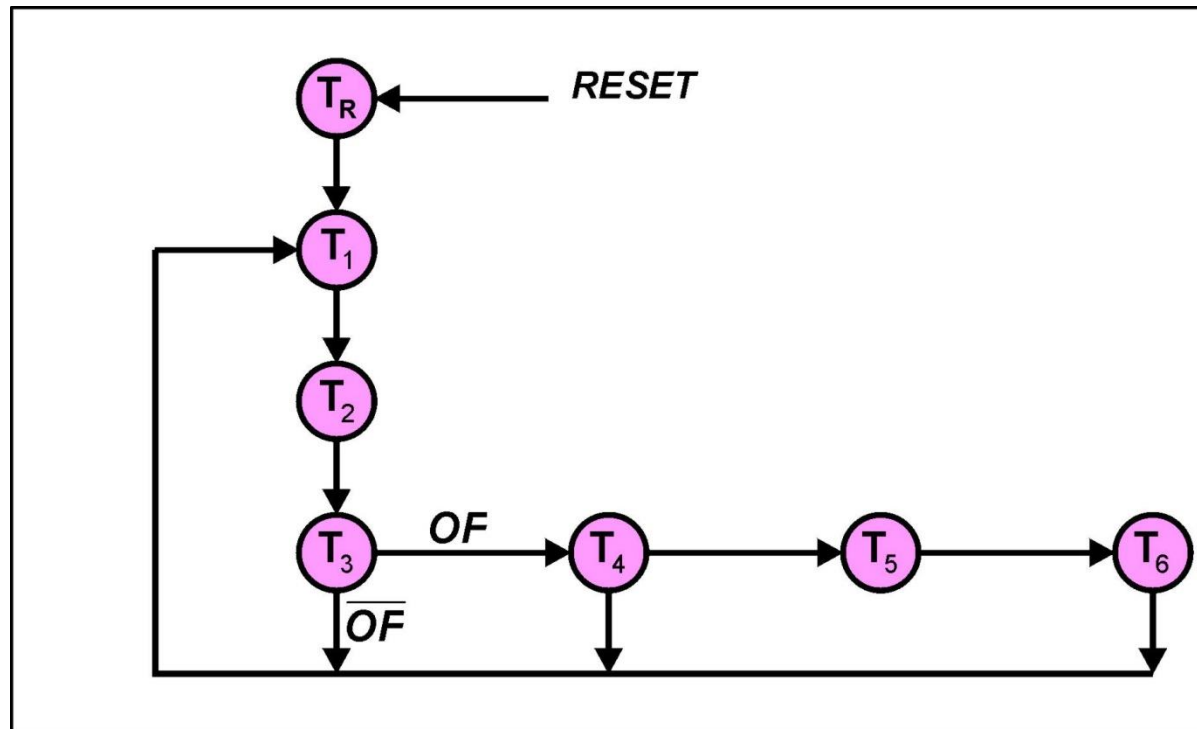
MACHINE CYCLE

- Example instruction: STA addr



T-STATE

- The operation of the 8085 can be described with respect to its **state diagram** (it is a synchronous state machine)



MACHINE CYCLE

- Opcode Fetch
 - The **first** operation in every instruction
 - Retrieves the opcode of the instruction that is being executed
 - Usually composed **4** clock cycles (T-state)
 - Some instruction required **6** T-state (CALL, INX, DCX)
 - $\overline{\text{IO/M}}$ signal is low (indicating a memory operation)

MACHINE CYCLE

● Memory Read

- Machine cycle during which memory is **read**
- Composed **3** clock cycles
- $\overline{\text{IO/M}}$ signal is low (indicating a memory operation)

● Memory Write

- This machine cycle is used when the 8085 needs to send data out from the accumulator or a specific register and then **write** it into memory
- Composed **3** clock cycles
- $\overline{\text{IO/M}}$ signal is low (indicating a memory operation)

MACHINE CYCLE

⦿ I/O Read

- Indicates that data is being **read** from an I/O device
- Occurs when IN instruction is executed
- Composed **3** clock cycles
- $\overline{\text{IO/M}}$ signal is high (indicating an I/O operation)

⦿ I/O Write







- This machine cycle is used to write data out from accumulator in the microprocessor to the I/O device specified by the port address
- Occurs when OUT instruction is executed
- Composed **3** clock cycles
- $\overline{\text{IO/M}}$ signal is high (indicating an I/O operation)

MACHINE CYCLE

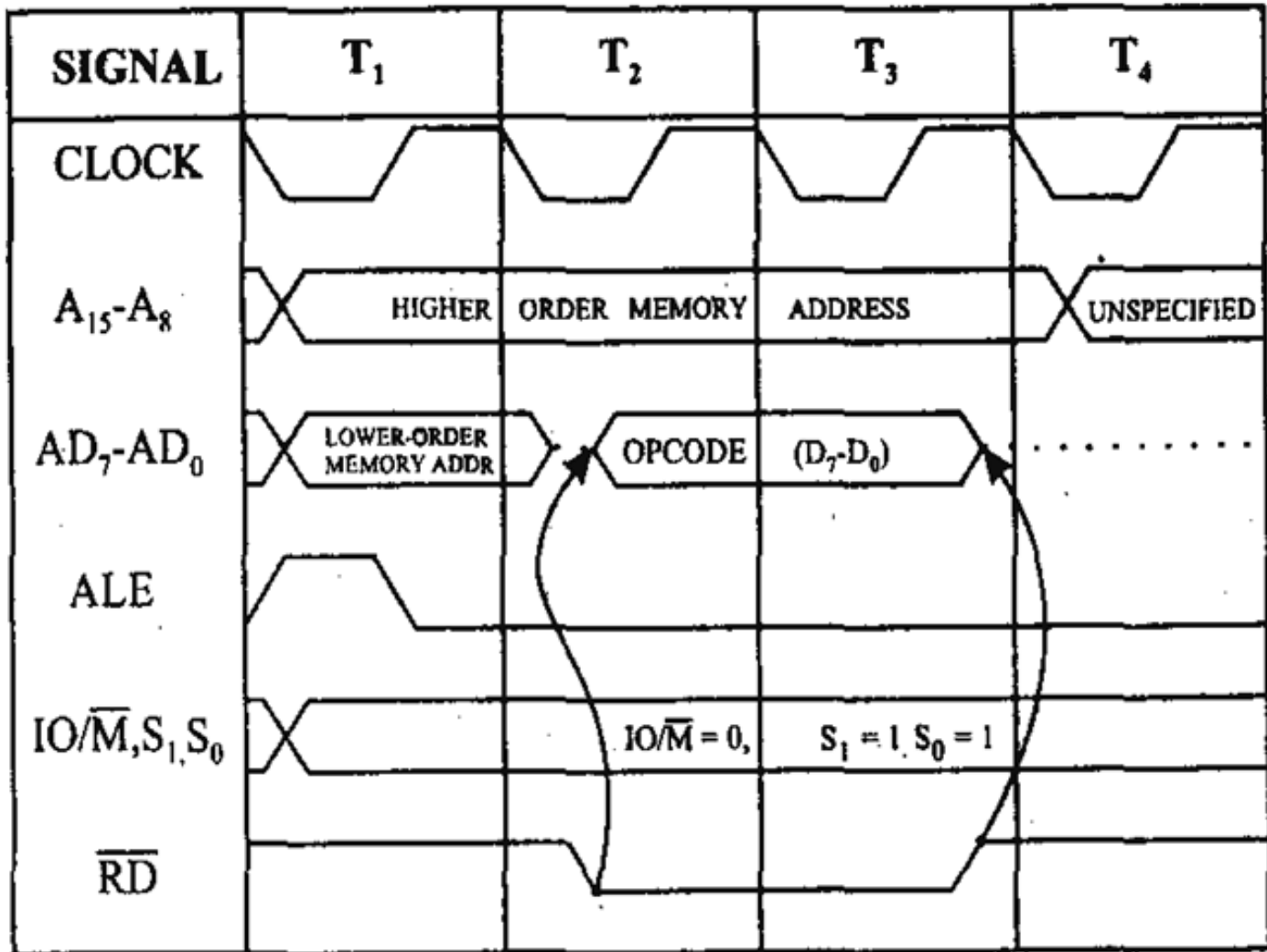
- Interrupt Acknowledge
 - Special machine cycle that is used to place of the opcode fetch cycle in the RST (restart) instruction.
 - Similar to opcode fetch instruction except that it send out $\overline{\text{INTA}}$ signal instead of $\overline{\text{RD}}$ signal and $\text{IO}/\overline{\text{M}}$ signal is High
 - Composed 6 clock cycle

T-STATE

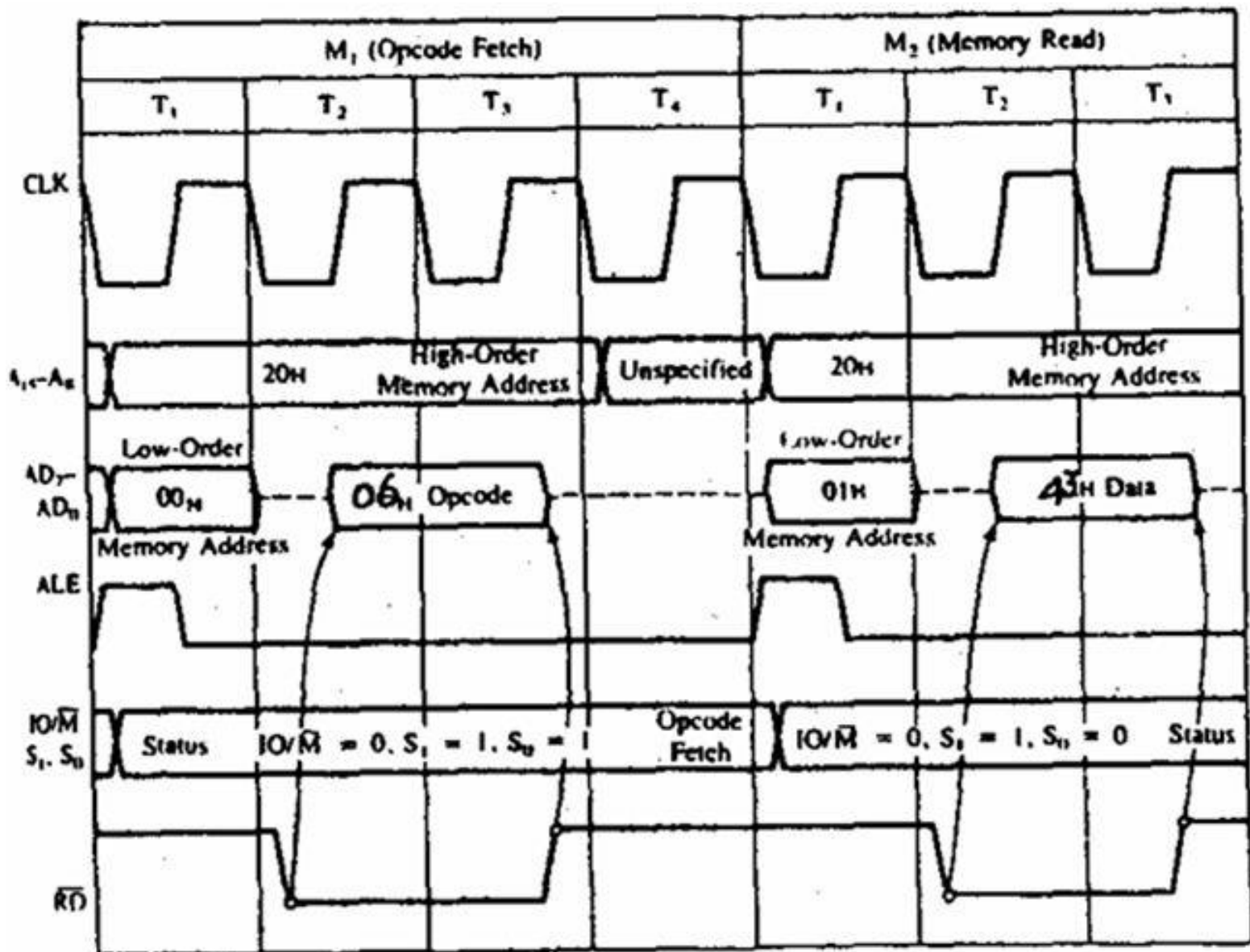
- Key to timing diagram

	Indicates a bus. Whilst the lines remain parallel the individual bits of the bus remain unchanged.
	Indicates a bus. Where the lines cross indicates a possible change in logic level of one or more bits of the bus.
	Indicates a 0 -> 1 transition of a digital signal
	Indicates a 1 -> 0 transition of a digital signal
	Indicates a bus or a bit being in the Hi-Z state (tri-state)
	The tail of the arrow indicates the cause of a signal change. The head of the arrow indicates the affected signal.

Opcode Fetch

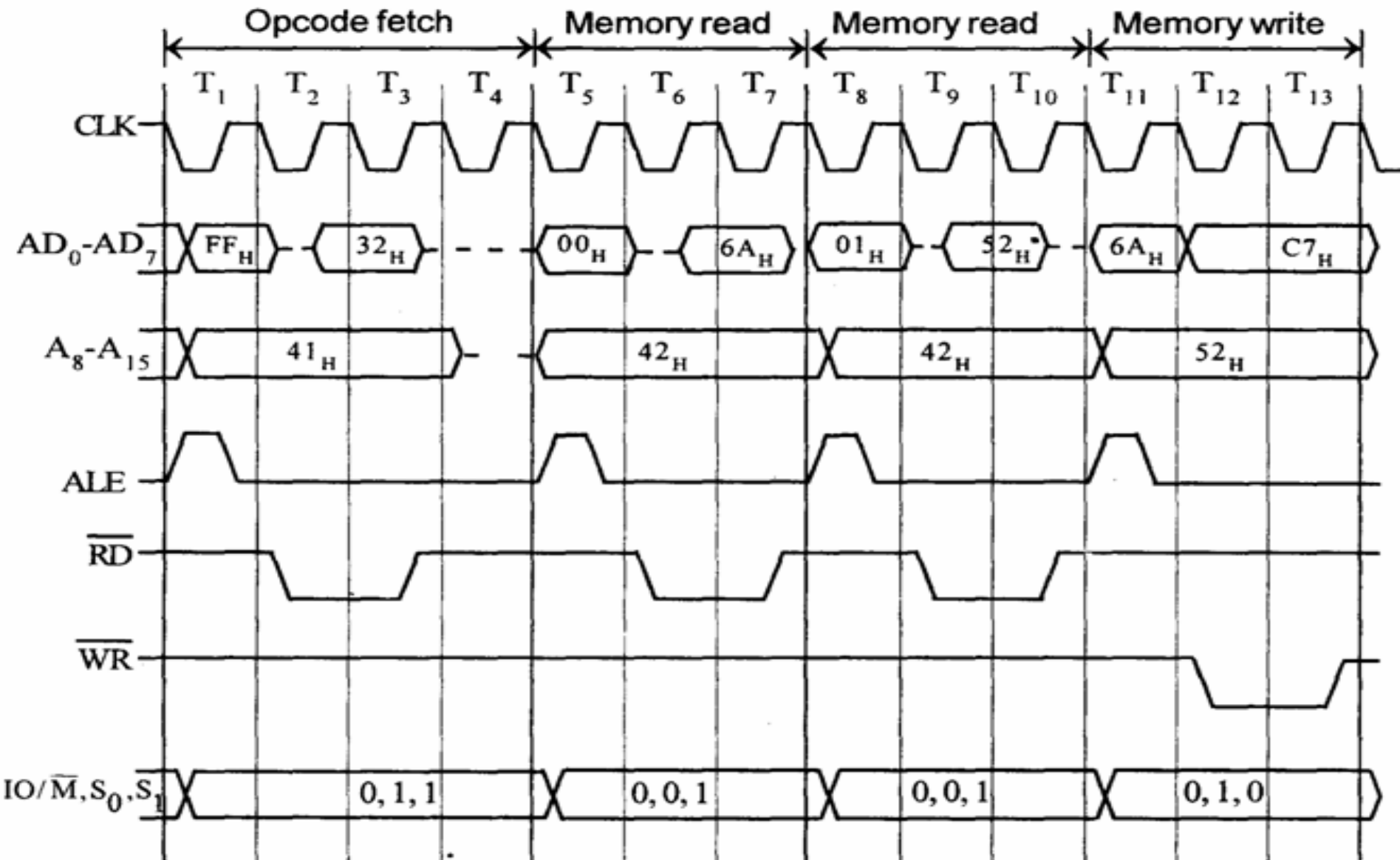


MVI

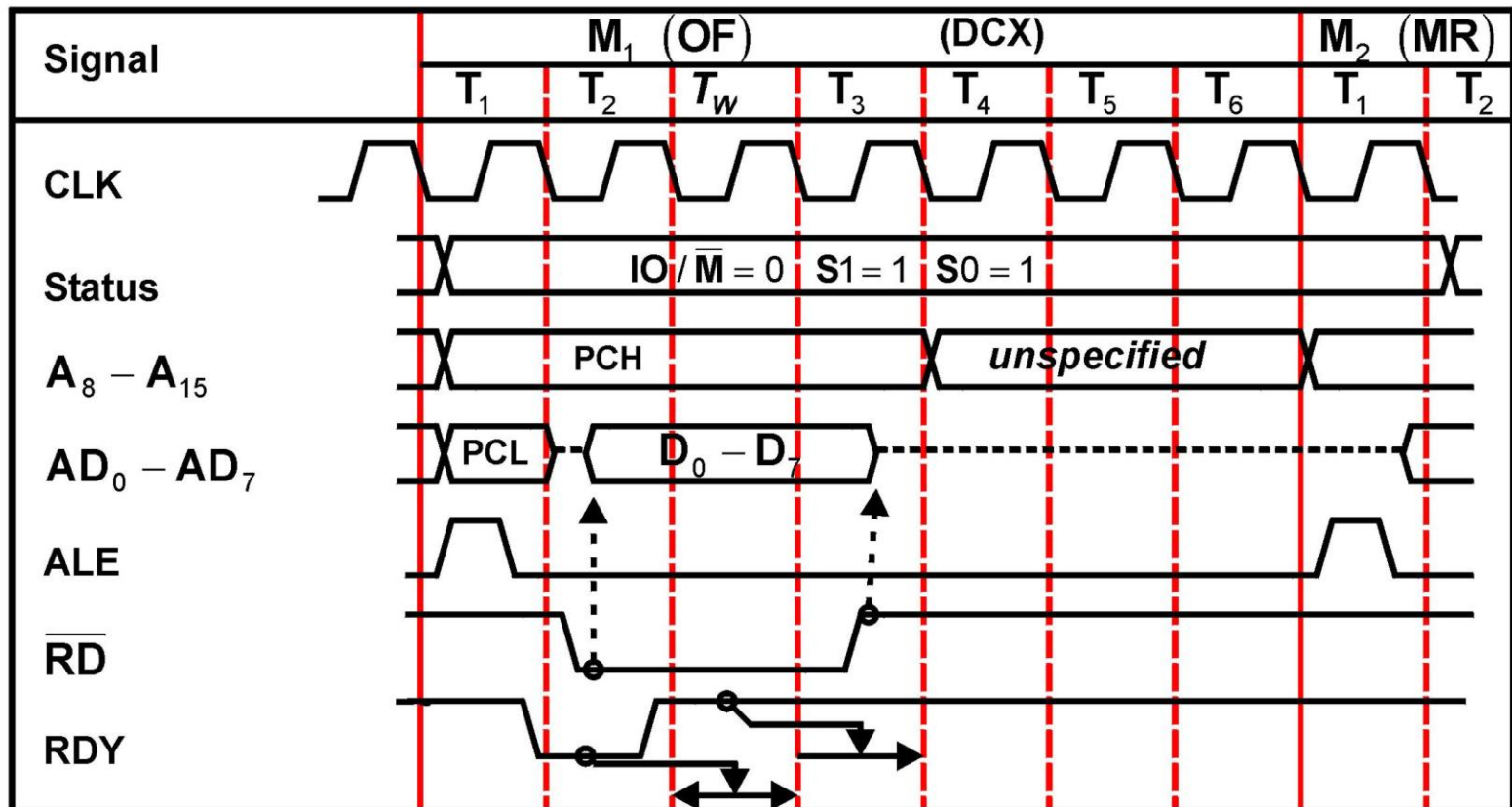


STA 526A

(32, 6A, 52 (ACC-C7))



Wait-State



Instruction Set & Addressing mode

- **Instruction Set:**
 - 8085 instruction set consists of the following instructions:
 - Data moving instructions.
 - Arithmetic - add, subtract, increment and decrement.
 - Logic - AND, OR, XOR and rotate.
 - Control transfer - conditional, unconditional, call subroutine, return from subroutine and restarts.
 - Input/Output instructions.
 - Other - setting/clearing flag bits, enabling/disabling interrupts, stack operations, etc.
- **Addressing mode**
 - **Register** - references the data in a register or in a register pair.
 - **Register indirect** - instruction specifies register pair containing address, where the data is located.
 - **Direct**,
 - **Immediate** - 8 or 16-bit data

8085 Microprocessor Memory

- **Memory:**
Program, data and stack memories occupy the same memory space. The total addressable memory size is 64KB.
- **Program memory** - program can be located anywhere in memory.
 - Jump, branch and call instructions use 16-bit addresses, i.e. they can be used to jump/branch anywhere within 64 KB.
 - All jump/ branch instructions use absolute addressing.
- **Data memory** - the processor always uses 16-bit addresses so that data can be placed anywhere.
- **Stack memory** is limited only by the size of memory. Stack grows downward.
- **First 64 bytes in a zero memory page should be reserved for vectors used by RST instructions**

8085 MP Instruction Set Architecture

- Contains several registers include B,C,D,E,H,L and an 8-bit accumulator register, A.
- The registers B,C,D,E,H,L can be accessed as pairs. Pairs are not arbitrary. B and C, D and E, H and L.
- SP is a 16 bit stack pointer register pointing to the top of the stack.
- PC is a 16-bit Program counter
- Contains five flags known as flag registers:

Instruction Set of 8085

- Arithmetic Operations
 - add, sub, inr/dcr
- Logical operation
 - and, or, xor, rotate, compare, complement
- Branch operation
 - Jump, call, return
- Data transfer/Copy/Memory operation/IO
 - MOV, MVI, LD, ST, OUT

Copy/Mem/IO operation

- MVI R, 8 bit // load immediate data
- MOV R1, R2 // Example MOV B, A
- MOV R M // Copy to R from 0(HL Reg) Mem
- MOV M R // Copy from R to 0(HL Reg) Mem

- LDA 16 bit // load A from 0(16bit)
- STA 16 bit // Store A to 0(16bit)
- LDAX Rp // load A from 0(Rp), Rp=RegPair
- STAX Rp // Store A to 0(Rp)
- LXI Rp 16bit // load immediate to Rp

- IN 8bit // Accept data to A from port 0(8bit)
- OUT 8 bit // Send data of A to port 0(8bit)

Arithmetic Operation

- ADD R *// Add $A = A + B.reg$*
- ADI 8bit *// Add $A = A + 8bit$*
- ADD M *// Add $A = A + 0(HL)$*

- SUB R *// Sub $A = A - B.reg$*
- SUI 8bit *// Sub $A = A - 8bit$*
- SUB M *// Sub $A = A - 0(HL)$*

- INR R *// $R = R + 1$*
- INR M *// $0(HL) = 0(HL) + 1$*
- DCR R *// $R = R - 1$*
- DCR M *// $0(HL) = 0(HL) - 1$*
- INX Rp *// $Rp = Rp + 1$*
- DCX Rp *// $Rp = Rp - 1$*

Other Operations

- Logic operations
 - ANA R ANI 8bit ANA M
 - ORA, ORI, XRA, XRI
 - CMP R // compare with R with ACC
 - CPI 8bit // compare 8 bit with ACC
- Branch operations
 - JMP 16bit, CALL 16 bit
 - JZ 16bit, JNZ 16bit, JC 16bit, JNC 16 bit
 - RET
- Machine Control operations
 - HLT, NOP, POP, PUSH

8085 Microprocessor Instruction Set

Contains a total of 74 different instructions.

R, R1, R2	8 bit registers representing A, B, C, D, E, H or L
M	Indicates memory location
RP	Indicates register pair such as BC, DE, HL, SP
Γ	16 bit address representing address or data value.
n	8-bit address or data value stored in memory immediately after the opcode
Cond	Condition for conditional instructions. NZ ($Z = 0$), Z ($Z = 1$), P ($S = 0$), N ($S = 1$), PO ($P = 0$), PE ($P = 1$), NC ($CY = 0$), C ($CY = 1$)

Data movement instruction for the 8085 microprocessor

Instruction	Operation
NOP	No operation
MOV r1, r2	$r1 = r2$
MOV r, M	$r1 = M[HL]$
MOV M, r	$M[HL] = r$
MVI r, n	$r = n$
MVI M, n	$M[HL] = n$
LXI rp, Γ	$rp = \Gamma$
LDA Γ	$A = M[\Gamma]$
STA Γ	$M[\Gamma] = A$
LHLD Γ	$HL = M[\Gamma], M[\Gamma + 1]$
SHLD Γ	$M[\Gamma], M[\Gamma + 1] = HL$
LDAX rp	$A = M[rp]$ ($rp = BC, DE$)
STAX rp	$M[rp] = A$ ($rp = BC, DE$)
XCHG	$DE \leftrightarrow HL$
PUSH rp	Stack = rp ($rp \neq SP$)
PUSH PSW	Stack = A, flag register
POP rp	rp = Stack ($rp \neq SP$)
POP PSW	A, flag register = Stack
XTHL	$HL \leftrightarrow \text{Stack}$
SPHL	$SP = HL$
IN n	$A = \text{input port } n$
OUT n	Output port $n = A$

Data operation instruction for the 8085 microprocessor

Instruction	Operation	Flags
ADD r	$A = A + r$	All
ADD M	$A = A + M[HL]$	All
ADI n	$A = A + n$	All
ADC r	$A = A + r + CY$	All
ADC M	$A = A + M[HL] + CY$	All
ACI n	$A = A + n + CY$	All
SUB r	$A = A - r$	All
SUB M	$A = A - M[HL]$	All
SUI n	$A = A - n$	All
SBB r	$A = A - r - CY$	All
SBB M	$A = A - M[HL] - CY$	All
SBI n	$A = A - n - CY$	All
INR r	$r = r + 1$	Not CY
INR M	$M[HL] = M[HL] + 1$	Not CY
DCR r	$r = r - 1$	Not CY
DCR M	$M[HL] = M[HL] - 1$	Not CY
INX rp	$rp = rp + 1$	None
DCX rp	$rp = rp - 1$	None
DAD rp	$HL = HL + rp$	CY
DAA	Decimal adjust	All
ANA r	$A = A \wedge r$	All
ANA M	$A = A \wedge M[HL]$	All

Data operation instruction for the 8085 microprocessor

Instruction	Operation	Flags
ANI n	$A = A \wedge n$	All
ORA r	$A = A \vee r$	All
ORA M	$A = A \vee M[HL]$	All
ORI n	$A = A \vee n$	All
XRA r	$A = A \oplus r$	All
XRA M	$A = A \oplus M[HL]$	All
XRI n	$A = A \oplus n$	All
CMP r	Compare A and r	All
CMP M	Compare A and M[HL]	All
CPI n	Compare A and n	All
RLC	$CY = A7, A = A(6-0), A7$	CY
RRC	$CY = A0, A = A0, A(7-1)$	CY
RAL	$CY, A = A, CY$	CY
RAR	$A, CY = CY, A$	CY
CMA	$A = A'$	None
CMC	$CY = CY'$	CY
STC	$CY = 1$	CY

Program control instruction

Instruction	Operation
JUMP Γ	GOTO Γ
J cond Γ	If condition is true then GOTO Γ
PCHL	GOTO address HL
CALL Γ	Call subroutine at Γ
C cond Γ	If condition is true then call subroutine at Γ
RET	Return from subroutine
R cond	If condition is true then return from subroutine
RST n	Call subroutine at $8*n$ ($n = 5.5, 6.5, 7.5$)
RIM	$A = IM$
SIM	$IM = A$
DI	Disable interrupts
EI	Enable interrupts
HLT	Halt the CPU

A Simple 8085 Program

1: $i = n$, $sum = 0$

2: $sum = sum + i$, $i = i - 1$

3: IF $i \neq 0$ then GOTO 2

4: $total = sum$

A Simple 8085 Program (contd)

LDA n	}	$i = n$
MOV B, A		
XRA A	}	$sum = A \oplus A = 0$
<i>Loop</i> :ADD B	}	$sum = sum + i$
DCR B	}	$i = i - 1$
JNZ <i>Loop</i>	}	IF $i \neq 0$ THEN GOTO <i>Loop</i>
STA <i>total</i>	}	$total = sum$

Execution trace

Instruction	1st Loop	2nd Loop	3rd Loop	4th Loop	5th Loop
LDA n MOV B, A	B = 5				
XRA A	A = 0				
ADD B	A = 5	A = 9	A = 12	A = 14	A = 15
DCR B	B = 4, Z = 0	B = 3, Z = 0	B = 2, Z = 0	B = 1, Z = 0	B = 0, Z = 1
JNZ <i>Loop</i>	JUMP	JUMP	JUMP	JUMP	NO JUMP
STA <i>total</i>					<i>total</i> = 15

Simple Assembly Program

```
MVI  A, 24H    // load Reg ACC with 24H
MVI  B , 56H    // load Reg B with 56H
ADD  B          // ACC= ACC+B
OUT  01H        // Display ACC contents on port 01H
HALT           // End the program
```

Result: 7A (All are in Hex)

Code to multiply two number

```
LDA 2000 // Load multiplicand to accumulator
MOV B,A  // Move multiplicand from A(acc) to B register
LDA 2001 // Load multiplier to accumulator
MOV C,A  // Move multiplier from A to C
MVI A,00 // Load immediate value 00 to a
L: ADD B  // Add B(multiplier) with A
DCR C    // Decrement C, it act as a counter
JNZ L    // Jump to L if C reaches 0
STA 2010 // Store result in to memory
HLT      // End
```

TABLE 1–2 Many modern Intel and Motorola microprocessors.

<i>Manufacturer</i>	<i>Part</i>	<i>Data Bus Width</i>	<i>Memory Size</i>
Intel	8048	8	2K internal
	8051	8	8K internal
	8085A	8	64K
	8086	16	1M
	8088	8	1M
	8096	16	8K internal
	80186	16	1M
	80188	8	1M
	80251	8	16K internal
	80286	16	16M
	80386EX	16	64M
	80386DX	32	4G
	80386SL	16	32M
	80386SLC	16	32M + 1K cache
	80386SX	16	16M
	80486DX/DX2	32	4G + 8K cache
	80486SX	32	4G + 8K cache
	80486DX4	32	4G + 16K cache
	Pentium	64	4G + 16K cache
	Pentium Overdrive (P24T) (replaces 80486)	32	4G + 16K cache
	Pentium Pro processor	64	64G + 16K L1 cache + 256K L2 cache
	Pentium II	64	64G + 32K L1 cache + 512K L2 cache
	Pentium II Xeon	64	64G + 32K L1 cache + 512K or 1M L2 cache
	Pentium III, Pentium 4	64	64G + 32K L1 cache + 256K L2 cache
Motorola	6800	8	64K
	6805	8	2K
	6809	8	64K
	68000	16	16M
	68008Q	8	1M
	68008D	8	4M
	68010	16	16M
	68020	32	4G
	68030	32	4G + 256 cache
	68040	32	4G + 8K cache
	68050	32	Proposed, but never released
	68060	64	4G + 16K cache
	PowerPC	64	4G + 32K cache

Table 1.1 The Evolution of Intel Microprocessors¹

Microprocessor	Year Introduced	Number of Transistors	Minimum Feature Size (microns)	External Data Bus Width	Internal Register Widths	Address Bus Width/ Memory Space	Estimated Processing Rate (MIPs) ²	Onboard Coprocessor	Internal Cache Memory	V _{CC} (volts)	P _D (watts)
4004	1971	2,250	10.0	4	4	10/1K	.06 (.108MHz)	no	no		
8080	1974	6,000	6.0	8	8	16/64K	.2 (2 MHz)	no	no	±5, 12	1.2
8086	1978	29,000	3.0	16	16	20/1 MB	.47 (4.77 MHz)	no	no	5	1.7
8088	1979	29,000	3.0	8	16	20/1 MB	.33 (4.77 MHz)	no	no	5	1.7
80286	1982	134,000	1.5	16	16	24/16 MB	2 (8 MHz)	no	no	5	3
80386DX	1985	275,000	1.5	32	32	32/4 GB	5.5 (16 MHz)	no	no	5	1.95
80386SX	1988	275,000	1.5	16	32	24/16 MB	3.9 (16 MHz)	no	no	5	1.9
80486DX	1989	1.2 million	0.8	32	32	32/4 GB	20 (25 MHz)	yes	8K	5	5
80486SX	1991	1.2 million	0.8	32	32	32/4 GB	13 (16 MHz)	no	8K	5	3.4
80486DX2	1992	1.2 million	0.6	32	32	32/4 GB	41 (50 MHz)	yes	8K	5	4.8
80486DX4	1994	1.2 million	0.6	32	32	32/4 GB	60 (75 MHz)	yes	16K	3.3	
Pentium P60	1993	3.1 million	0.8	64	32	32/4 GB	100 (60 MHz)	yes	16K	5	14.6
Pentium P100	1994	3.1 million	0.6	64	32	32/4 GB	150 (100 MHz)	yes	16K	3.3	10.1
Pentium P120	1995	3.1 million	0.35	64	32	32/4 GB	185 (120 MHz)	yes	16K	3.3	12.8
Pentium Pro 150	1995	5.5 million ³	0.6	64	32	36/64 GB	350 (150 MHz)	yes	16K/256K ⁴	3.3	29.2
Pentium Pro 200	1996	5.5 million	0.35	64	32	36/64 GB	475 (200 MHz)	yes	16k/512K	3.3	35
P7 ⁵	1997–8	12 million	0.2	—	—	—	750	yes	—	—	—

¹Specifications shown are for initial introduction of part.²Millions of instructions per second (internal clock rate shown in parentheses).³256K level two cache (separate die in same package) has 15.5 million transistors.⁴16K data/code level one cache plus 256K level two cache.⁵Best guess.