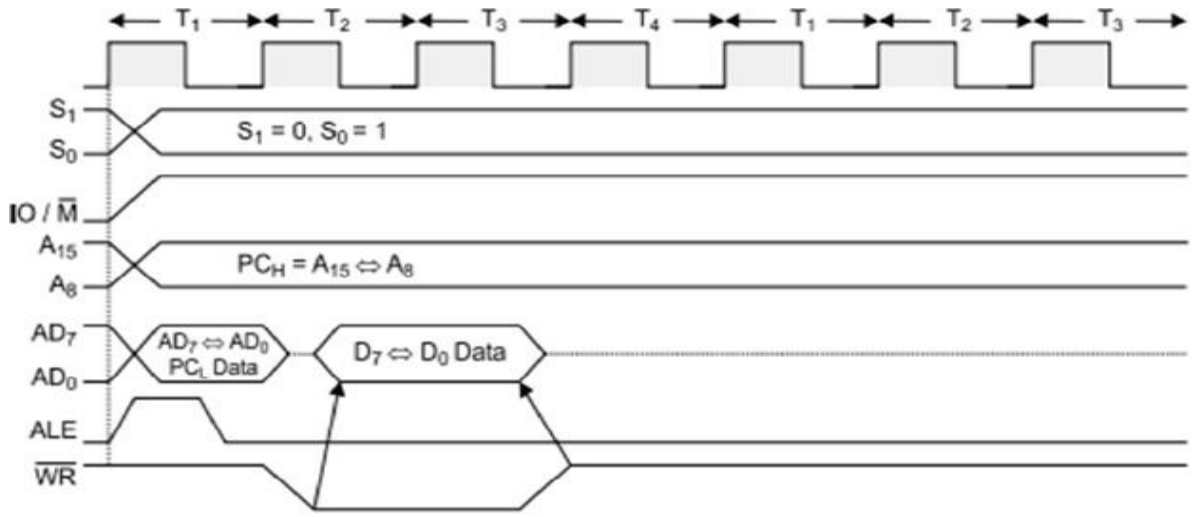

Fig 14. IO Write timing diagram

INTERRUPT:

An interrupt is a signal initiated by an external device to the microprocessor. Once this signal is received, the microprocessor completes the execution of the current instruction and responds to the interrupt

SOFTWARE INTERRUPTS OF 8085

The software interrupts are program instructions. When the instruction is executed, the processor executes an interrupt service routine stored in the vector address of the software interrupt instruction. The software interrupts of 8085 are RST 0, RST 1, RST 2, RST 3, RST 4, RST 5, RST 6 and RST 7.

The vector addresses of software interrupts are given in table below

Interrupt	Vector address
RST 0 RST 1	0000_H 0008_H
RST 2 RST 3	0010_H 0018_H
RST 4 RST 5	0020_H 0028_H
RST 6 RST 7	0030_H 0038_H

Interrupt	Vector address
RST 7.5	$003C_H$
RST 6.5	0034_H
RST 5.5	$002C_H$
TRAP	0024_H

The software interrupt instructions are included at the appropriate (or required) place in the main program. When the processor encounters the software instruction, it pushes the content of PC (Program Counter) to stack. Then loads the Vector address in PC and starts executing the Interrupt Service Routine (ISR) stored in this vector address. At the end of ISR, a return instruction - RET will be placed. When the RET instruction is executed, the processor POP the content of stack to PC. Hence the processor control returns to the main program after servicing the interrupt. Execution of ISR is referred to as servicing of interrupt. All software interrupts of 8085 are vectored interrupts. The software interrupts cannot be masked and they cannot be disabled. The software interrupts are RST0, RST1, … RST7 (8 Nos).

HARDWARE INTERRUPTS OF 8085

These are the interrupts provided as signals to the microprocessor. There are five interrupt signals in 8085. They are Trap, RST 7.5, RST 6.5, RST 5.5 and INTR. The priority of the interrupts is from TRAP to INTR. The program executed for the service of the interrupting device is called the service routine.

TRAP

1. This interrupt is a Non-Maskable interrupt (NMI). It is unaffected by any mask or interrupt enable.
2. TRAP is the highest priority and vectored interrupt(as vector address is fixed i.e. memory location where to transfer control).
3. TRAP interrupt is edge and level triggered. This means hat the TRAP must go high and remain high until it is acknowledged.
4. In sudden power failure, it executes a ISR and send the data from main memory to backup memory.
5. The signal, which overrides the TRAP, is HOLD signal. (i.e., If the processor receives HOLD and TRAP at the same time then HOLD is recognized first and then TRAP is recognized).
6. There are two ways to clear TRAP interrupt.

    - By resetting microprocessor (External signal)
    - By giving a high TRAP ACKNOWLEDGE (Internal signal)

RST 7.5

- The RST 7.5 interrupt is a Maskable interrupt.

- It has the second highest priority.
- It is edge sensitive. i.e. Input goes to high and no need to maintain high state until it recognized.
- Maskable interrupt

It is disabled by,

- DI, SIM instruction
- System or processor reset.
- After reorganization of interrupt.

RST 6.5 and 5.5

- The RST 6.5 and RST 5.5 both are level triggered (i.e.) Input goes to high and stay high until it recognized.
- Maskable interrupt

It is disabled by,

- DI, SIM instruction
- System or processor reset.
- After reorganization of interrupt.
- Enabled by EI instruction.
- The RST 6.5 has the third priority whereas RST 5.5 has the fourth priority.

These interrupts are classified further into two classes based on the destination address and response. Based on the service routine address, interrupts are classified in to vectored and non-vectored interrupt.

VECTORED INTERRUPT:

If the address of the service routine is known to the microprocessor, i.e. if the service routine begins at a predefined address, then the interrupts are called vectored interrupts. The vectored address is calculated as $(nx8)_{16}$ where n is the number of RST.

For example:

The vectored address of RST 7.5 is 7.5 x 8=60.0

60 in hexadecimal number system is 003C. Therefore the branching address of RST 7.5 is 003C.

Interrupt	**Address**

RST 7.5	003C
RST 6.5	0034
RST 5.5	002C
TRAP (RST 4.5)	0024

NON VECTORED INTERRUPT:

The address of the service routine is not known in prior to the microprocessor. It is sent by the interrupting device.When the interrupt flipflop is enabled and INTR is high, microprocessor executes the current instruction and makes INTA low. Based on the flexibility to enable or disable interrupt, the interrupts are classified as maskable interrupt and non maskable interrupt.

Maskable Interrupt: Even if the interrupt signals are high, microprocessor will respond to these signals only when interrupt flip flop is enabled. Example RST 7.5, RST 6.5, RST 5.5, INTR

Non-Maskable Interrupt: Once the signal is enabled, the microprocessor immediately responds to this interrupt. Example: TRAP

STACK

Stack is the upper part of the memory used for storing temporary information. It is a Last In First Out Memory (LIFO). In 8085, it is accessed using PUSH and POP instructions. During pushing, the stack operates in a decrement then store‖ style. The stack pointer is decremented first, then the information is placed on the stack. During poping, the stack operates in a use then increment style. The information is retrieved from the top of the the stack and then the pointer is incremented. The SP pointer always points to the top of the stack‖.

PROGRAM STATUS WORD (PSW)

The 8085 recognizes one additional register pair called the PSW (Program Status Word). This register pair is made up of the Accumulator and the Flags registers. It is possible to push the PSW onto the stack, do whatever operations are needed, then POP it off of the stack. The result is that the contents of the Accumulator and the status of the Flags are returned to what they were before the operations were executed.

SUBROUTINES

A subroutine is a group of instructions that will be used repeatedly in different locations of the program. Rather than repeat the same instructions several times, they can be grouped into a subroutine that is called from the different locations. In Assembly language, a subroutine can exist anywhere in the code. However, it is customary to place subroutines separately from the

main program. The 8085 has two instructions for dealing with subroutines. The CALL instruction is used to redirect program execution to the subroutine. The RET instruction is used to return the execution to the calling routine.

CALL

CALL 4000H (3 byte instruction)

When CALL instruction is fetched, the MP knows that the next two Memory location contains 16bit subroutine address in the memory.

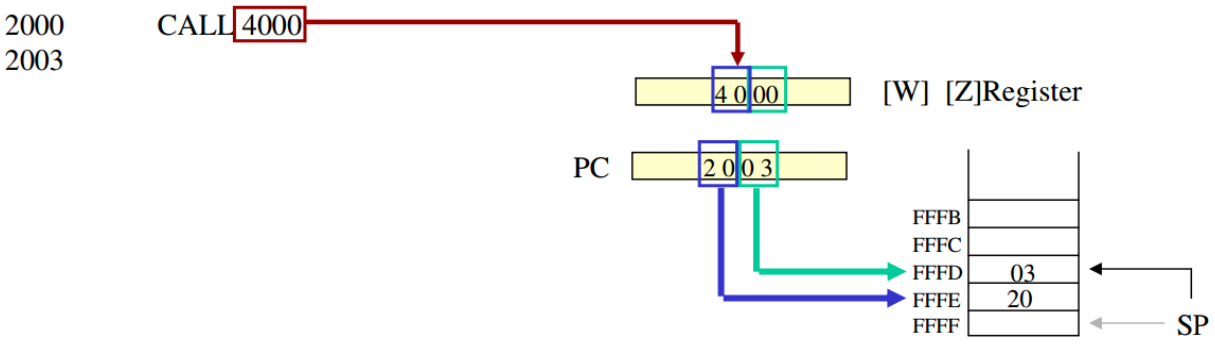

Fig 17.Work flow of CALL instruction

MP Reads the subroutine address from the next two memory location and stores the higher order 8bit of the address in the W register and stores the lower order 8bit of the address in the Z register. Push the address of the instruction immediately following the CALL onto the stack [Return address]. Loads the program counter with the 16-bit address supplied with the CALL instruction from WZ register as shown in fig 17.

RET (1 byte instruction)

Retrieve the return address from the top of the stack. Load the program counter with the return address as seen in fig 18.

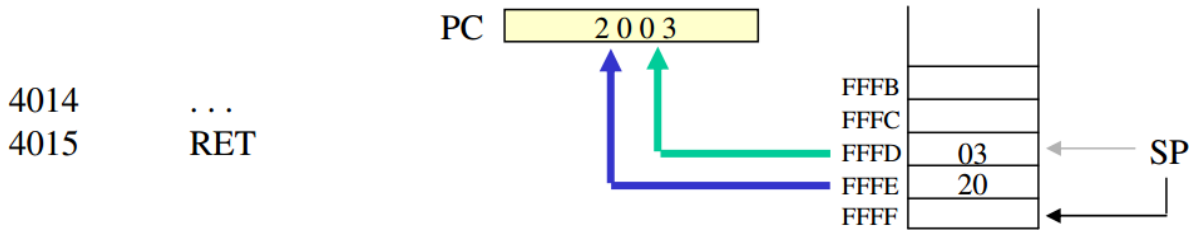

Fig 18.Work flow of RET instruction

The processor can regain the buses only after the Hold is removed. When the Hold is

acknowledged, the Address, Data, RD, WR, and IO/M lines are stated.

HLDA (Output)

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

INTR (Input)

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

INTA (Output)

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

RESTART INTERRUPTS: These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted. RST 7.5 ~~ Highest Priority RST 6.5 RST 5.5 Lowest Priority

TRAP (Input)

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

RESET IN (Input)

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

RESET OUT (Output)

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

X1, X2 (Input)

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

CLK (Output)

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

IO/M (Output)

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

SID (Input)

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

SOD (output)

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

Vcc +5 volt supply.

Vss Ground Reference.

8085 PROGRAMS

I. 8-BIT ADDITION.

Program:

MVI C, 00 - Clear one register for carry (Reg C)
LDA 9100 -Load the accumulator with the first data
MOV B, A - Move the accumulator content to one register (Reg B)
LDA 9101 -Load the accumulator with the second data
ADD B - Add Reg B content to accumulator
JNC L1: -Check for carry, if there is no carry, go to step 8
INR C-Increment reg C to indicate carry
L1 : STA 9200 - Store the results
MOV A, C- carry in Reg C
STA 9201- sum in accumulator to memory locations
RST 1 -stop

Sample Data:

Input	Output
9100 – 04	9200 – 0C
9101 – 08	9201 – 00
9100 – FF	9200 – FE
9101 – FF	9201 – 01

8-BIT SUBTRACTION

Program:

MVIC, 00 - Clear one register for borrow (Reg C)
LDA 9200 -Load the accumulator with the first data
MOV B, A - Move the accumulator content to one register (Reg B)
LDA 9201-Load the accumulator with the second data
SUB B- Subtract Reg B content from accumulator content
JNC L1-Check for carry, if there is no carry, go to step 10
CMA- Complement accumulator content
INR A-increment accumulator content
INR C- Increment reg C to indicate borrow
L1 :STA 9200 - Store the results
MOV A, C - borrow in Reg C
STA 9201- difference in accumulator to memory locations
RST 1-stop

Sample Data:

Input	Output
9200 – FF	9200 – 55
9201 – AA	9201 – 01
9200 – BB	9200 – 44
9201 – FF	9201 – 00

16-BIT ADDITION

Program:

MVI C, 00 -Clear one register for carry (Reg C)
LHLD 9100 -Load the first data in HL register pair
XCHG -Swap the contents of HL and DE pairs
LHLD -Load the second data in HL register pair
DAD D -Double add the contents of HL and DE pairs
JNC L1- Check for carry, if there is no carry go to step 8
INR C- Increment Reg C
L1: SHLD 9104- Store the result which is in HL pair in a memory location
MOV A, C - Move the carry in Reg C to accumulator
STA 9106 - Store the accumulator content in memory
RST 1 - Stop

Sample Data:

Input	**Output**
9100 – 06 9101 – 05	9104 – 09 9105 – 0B
9102 – 03 9103 – 06	9106 – 00
9100 – 06 9101 – F0 9102 – 03 9103 – F0	9104 – 09 9105 – E0 9106 – 01

REVERSE THE STRING

Program:

MVIB, 06- Initialize one register (Reg B) with the length of the string
LXI H, 9100- Initialize one register pair (HL) with the starting address of the source array
LXI D, 9205 - Initialize one register pair (DE) with the ending address of the destination array
L1: MOV A, M - Move the memory content to accumulator
STAX D - Store the accumulator content in DE pair
INX H - Increment HL pair
DCX D - Decrement DE pair
DCR B - Decrement the counter register – Reg B
JNZ L1 - Check for zero, if not zero, goto step 4
RST1 - Stop

Sample Data:

Input	Output
9100 – 0E	9200 – 0C
9101 – 0E	9201 – 00
9102 – 0F	9202 – 0F
9103 – 0F	9203 – 0F
9104 – 00	9204 – 0E
9105 – 0C	9205 – 0E

FACTORIAL OF A NUMBER

Program:

LDA 9100-Load the accumulator with the given data
MOV B, A-Move the accumulator content to a register (Reg B)
MOV C, A-Move the accumulator content to another register (Reg C)
DCR C-Decrement Reg C
L2: MOV D, C -Move the content of Reg C toReg D
MVI A, 00-Clear the accumulator content
L1 : ADD B-Add Reg B content to accumulator
DCR D-Decrement Reg D
JNZ L1-Check for zero, if not zero, goto step 7
MOV B, A-Move accumulator content to Reg B
DCR C-Decrement Reg C
JNZ L2-Check for zero, if not zero, goto step 5
STA 9101-Store the accumulator content in a memory address
RST 1-Stop

Sample Data:

Input	Output
9100 – 04	9101 – 18