5 January

Tuesday

5-360 / Week 2

January 2021

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

8085 MICRO PROCESSOR

What is Architecture?

its a design that defines how the process inside a µP will be done is internally

which tells how

*) data flows between cpu, memory, I/o devices

*) Instruction are fetch, decode and executed
*) Memory is addressed and accessed
*) what instruction sets (ISAS) the processor use

Two Architecture generally used

1. Von-Nuemann → Same memory and buses for Instruction and data
2. Hardvard → Von-Nuemann Vice-Versa

1. Von-Nuemann Architecture

Core components :

1- Memory (RAM/ROM/flash)

*) A linear address space

2. CPU:

i) Registers:

*) PC (program Counter): address of next instruction
*) IR (Instruction Register): holds current ~~regis~~ Instruction
*) General purpose - Registers: holds operands, results.
*) MAR/MDR: Memory Addres/Data registers used in micro-operations
*) status / flags: Z, N, C, etc.
*) ALU
*) Control unit: orchestrates fetch / decode / execute; can be hardwired (or) microcoded

↓ which means

Arrange the elements (or) Instruction to give the desired output.

3. Buses:

*) Address Bus (CPU → Memory): which location to access
*) Data Bus: (CPU ↔ Memory): Values being transferred
*) Control Signals (RD/WR, clock, Interupts, etc)

4. I/O:

*) Interupts and DMA (Direct memory Access) to move data efficiently.
*) Either Memory Mapped (or) Port Mapped

Q1) Von-Neumann Architecture is a revolutionary ?

Because,

Stored-program Computers can,

*) load different programs without rewiring.

*) Treat program as data

Limitation - Von-Neumann Bottleneck

*) only one can be done at a time (Instruction fetch / data access)

Harvard Architecture:

uses two physically separate memories

*) Instruction memory → Typically ROM, flash (or) EPROM → non-volatile

*) Data memory → Typically SRAM → volatile.

Each memory has its own buses separately

CPU can fetch instruction and data simultaneously.

Types of Memory:

1. Primary - Memory:

1. RAM - Volatile

- DRAM (Dynamic) - use capacitor to store bits - slower, cheap → used for main system memory (laptop, Mobile).
- SRAM (Static) - uses flip-flop - faster, expensive - cache memory

2. ROM - Non-Volatile

- Masked ROM - programmed once at chip manufacturing - High-Volume Embedded devices
- PROM - Programmable once by user - Rare now.
- EPROM - Erasable with UV light reprogrammable - Developement boards.
- EEPROM - Electically Erasable, byte wise write - BIOS chip, Small configuration storage.
- Flash - fast, block wise erase/write - Firmware update, SSDs, Mobile storage.

3. Cache Memory: Very high speed SRAM — near the CPU
   *) store frequently accessed instructions / data to reduce DRAM latency

Secondary Memory:

Hard disk, optical discs, Magnetic tape
SSDs, External drives / USB sticks

Instruction set: The set of instruction that the microprocessor can execute.

Bandwidth: The number of bits processed in a single instruction.

10 Sunday

clock speed: Given in megahertz (MHz), the clock speed determines how many instructions per second the processor can execute.

8085

*) It is an 8 bit processor.

*) It is a single chip N-MOS device with 40 pins.

*) multiplexed Address and data bus. (AD_0 - AD_7)

*) 5V supply.

*) clock frequency 500kHz to 3MHz

*) 74 instructions with 5 different addressing modes.

*) 16 address line (A_0-A_{15}), it can access 64K bytes of memory.

*) 8 bit I/O address, it can access 256 input Ports.

*) 5 hardware interrupts : TRAP, RST 7.5, RST 6.5, RST 5.5, INTR.

*) It provides Accumulator, 6 general purpose register, Two special purpose register, one flag register

*) it provides serial lines SID, SOD .. its peripherals can be interfaced with 8085 directly.

12 January

Tuesday

New Moon

AD_0-AD_7 → multiplexed Add and data

A_8-A_{15} → Tri-Stated higher order address line

ALE → output signal, goes high when operation is started by processor.

S_0, S_1 → status signal to indicate type of operation

$\overline{RD}$ → active low input signal used to read data from I/O device/Memory

$\overline{WR}$ → active low output signal used to write data on memory or I/O device

Ready → This is an output signal used to check the status of output device. if it is low, µP will wait untill it is high.

TRAP → it is an Edge triggered highest priority, non-maskable interrupt. After TRAP restart occurs and execution starts from address 0024H.

RST 5.5, 6.5, 7.5 → Maskable and low priority than TRAP

$\overline{INTR}$ / INTA → request signal after which µP generates INTA

IO/$\overline{M}$ → Signal used to indicate whether 8085 is working in I/O mode(1) or Memory mode (0)

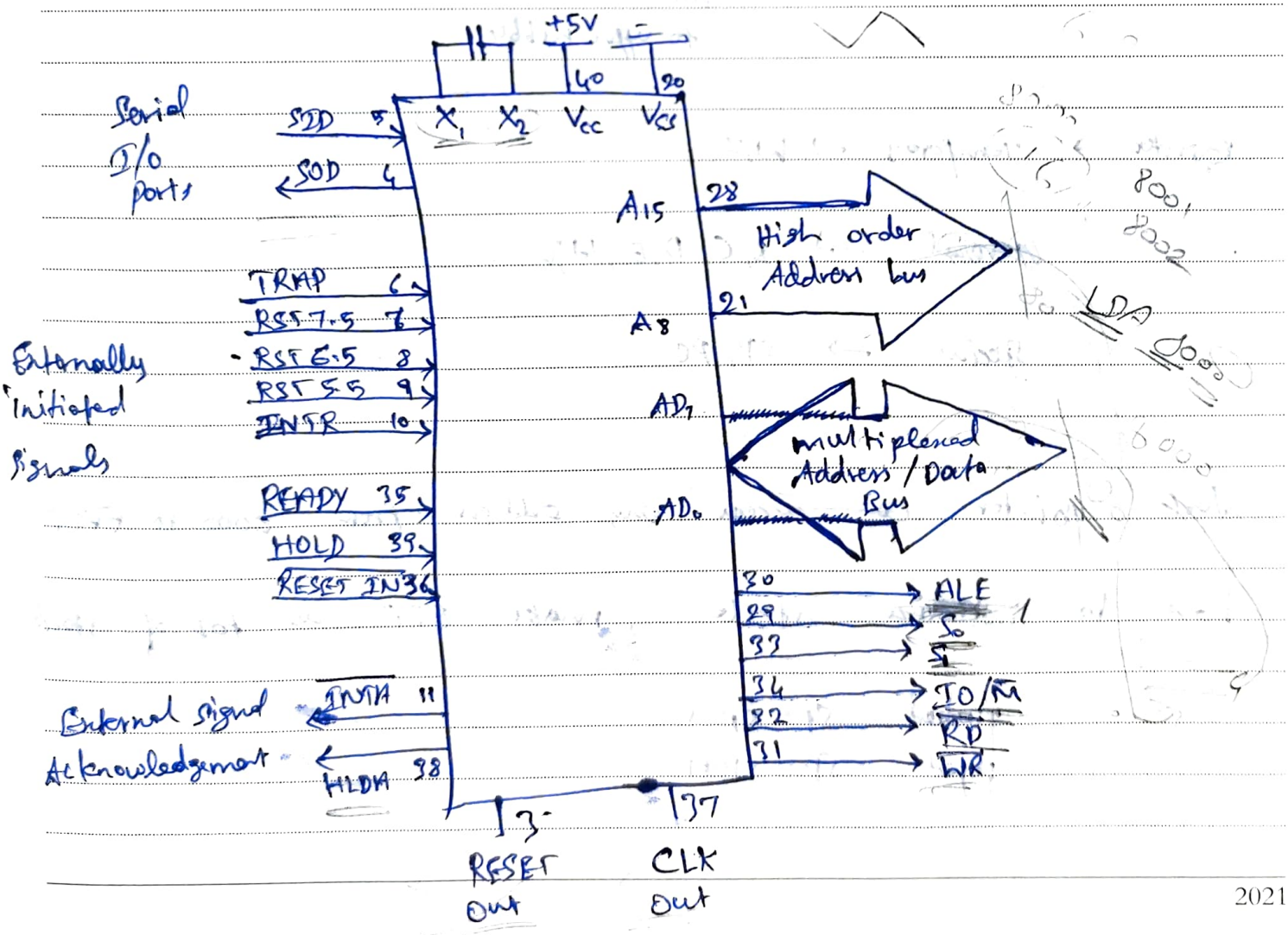
HOLD/HLDA → Input signal. When µP receives HOLD signal it completes the current machine cycle and stops executing next instruction. If it responds to HOLD µP generates HLDA that is HOLD Acknowledge Signal.

RESET IN → when reset in is low, µP starts executing from 0000H location.

SID → Serial input data is input pin used to accept serial 1 bit data.

X_1 X_2 → These are clock input signal and are connected to external LC or RC circuit. These are divide by two, so if 6MHz is connected to X_1 X_2, the operating frequency becomes 3 MHz.

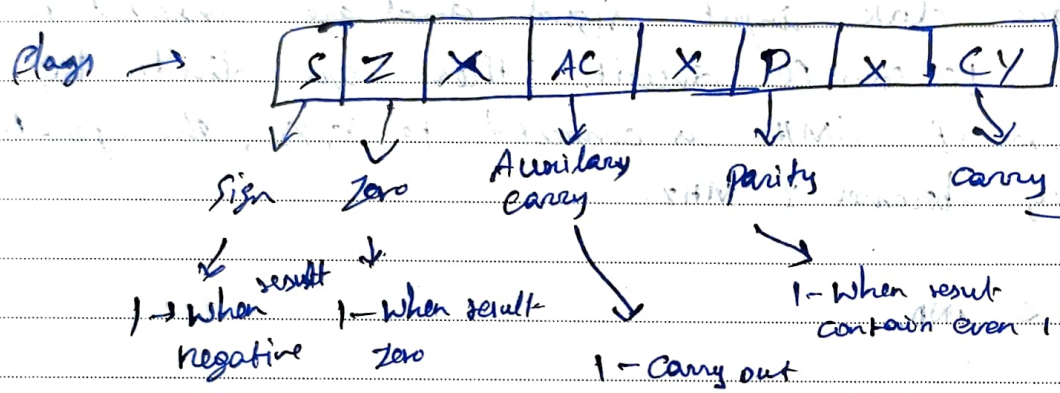
Vcc & Vss : +5V & GND.



The flag register along with accumulator is called PSW

↓

Program Status Word.



Register → Temporary → W, Z

General → B, C, D, E, H, L

Special → SP, PC

Stack pointer can access any address from 0000 to FFFF

how to initialize stack, make FFFF as top of stack

pushed SP = SP-1

poped SP = SP+1

Each vector address is separated by 8 bytes

January 15 Friday

15-350 / Week 3

5.5 × 8 = 44 in Hexadecimal

Interrupt: (Rst 4.5)

TRAP	– Non-Maskable, Vectored	–	0024H
RST 7.5	– Maskable, "	–	003CH
RST 6.5	– " "	–	0034H
RST 5.5	– " "	–	002CH
INTR	– " Non-Vectored	–	location given by externally

Instructions:

EI → Enable all maskable intrs
DI → Disable all " "
RIM → Read interrupt mask status and pending interrupts
SIM → SET interrupt Mask (mask or unmask specified)
RETI → Return from ISR (special RET to signal end of interrupt)

T-cycle – 1 clock period of the system clock.

Machine cycle – (3–6) T cycle – which is

January 2021

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Lower Order Address/Data bus

During first clock it acts as Address line

For second and third clock cycles it acts as data bus

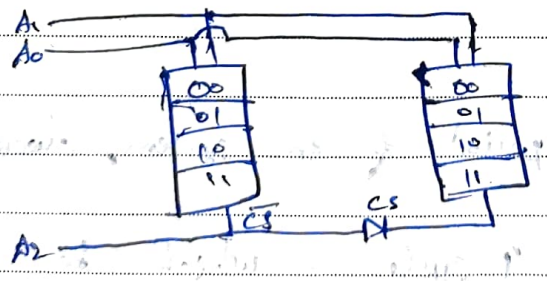
Higher order Address bus

It remains Address as lines as long operation is not completed

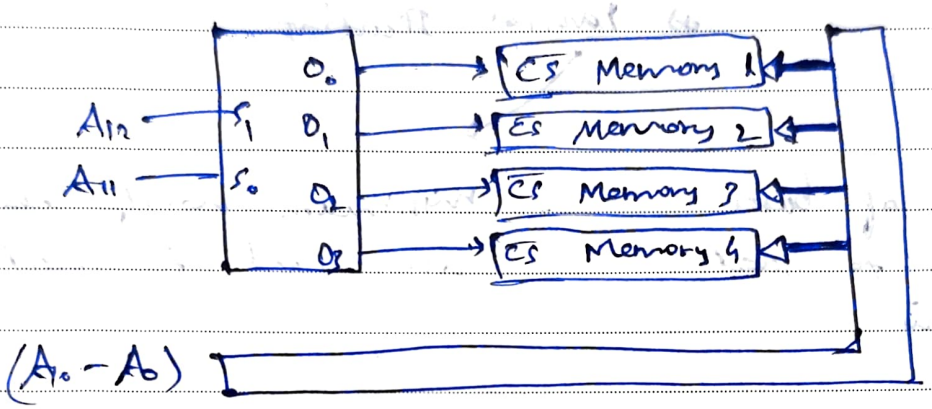
Interfacing

two memory are used, then 1 Address line was used to act as chip select.

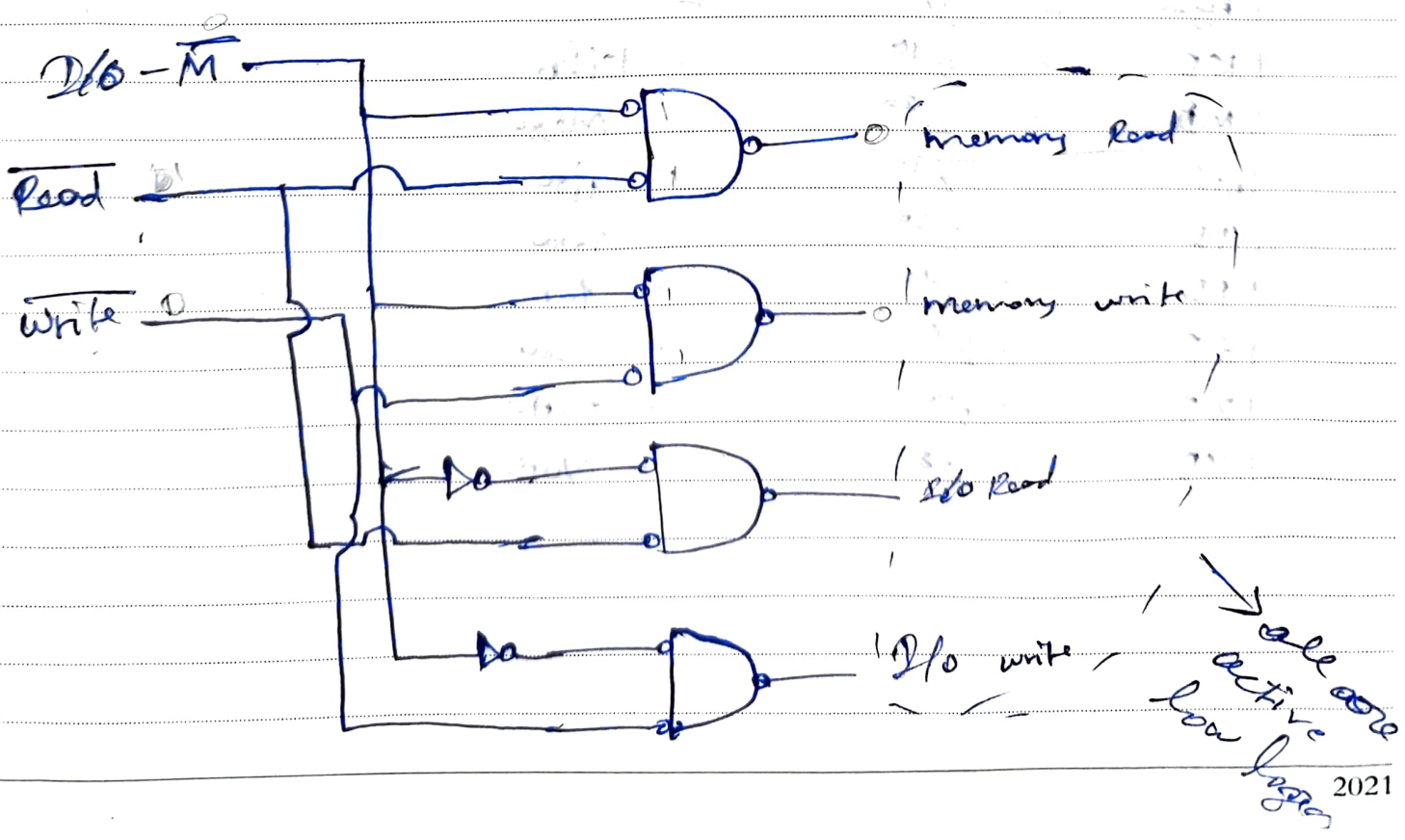
17 Sunday



multiple chips are used, then we go with Address decoders,



Memory Read - Write



There are two types : *) Exhaustive Decoding

*) Partial Decoding

many number of bases are used for to select particular chip.

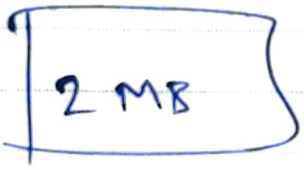
minimum no. of Address lines are used to select chip

Ex: previous page

Ex: previous, previous page

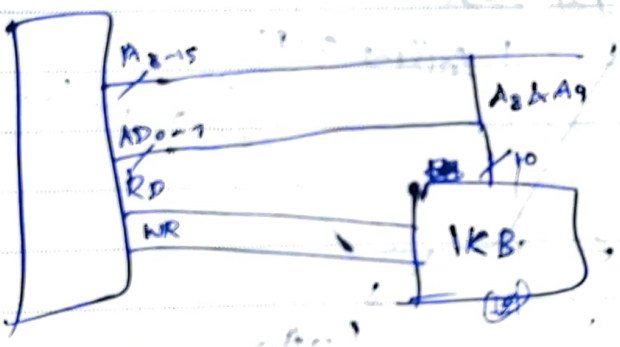
1 byte	$2^{(0)}$	
1 KB	2^{10}	- kilo
1 MB	2^{20}	- Mega
1 GB	2^{30}	- Giga
1 TB	2^{40}	- Tera
1 PB	2^{50}	- Peta
1 EB	2^{60}	- Exa
1 ZB	2^{70}	- Zetta
1 YB	2^{80}	- Yotta

January 20
Wednesday



$2 \times 2^{20} =$

20,97,152

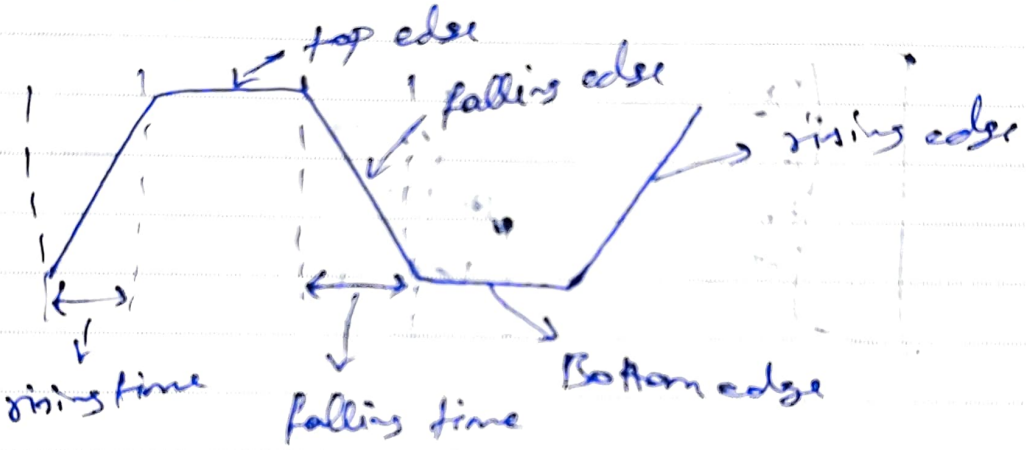


T State - a portion of an operation carried out in single clock pulse is T state.

Machine cycle - set of T states to finish one complete action.
eg: opcode fetch
(or) Memory read
(or) Memory write

Instruction cycle - set of machine cycle is called Instruction cycle.

Opcode fetch - 4T - 6T
Memory Read - 3T
Memory Write - 3T
I/o Read - 3T
I/o write - 3T
Interrupt Acknowledge cycle - 6T - 12T
Bus Idle cycle



Opcode fetch

the purpose is to put the opcode value in IR register by the PC pointing address memory address.

January 22
Friday

Instruction that requires 6T states in opcode fetch Machine cycle.

CALL - SP is decremented by 1

Register pair

CALL conditional - SP is dec by 1

PUSH (Rp)

DCX Rp - Register pair is dec by 1

INX Rp - Register pair is Inc by 1

PCHL - HL pair transfered to PC

SPHL - HL pair transfered to SP

RET conditional - the condition to return checked.

S_{01}	S_0	
0	0	→
0	1	→ Memory write - IO/M = 0
1	0	→ Memory read - IO/M = 0
1	1	→ Opcode fetch - IO/M = 0

23 January

Saturday

January 2021

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

23-342 / Week 4

XRA A has 1 Machine cycle

Data Transfer Instruction

LDA 16 bit address

STA "

MOV Rd, Rs

MOV M, Rs

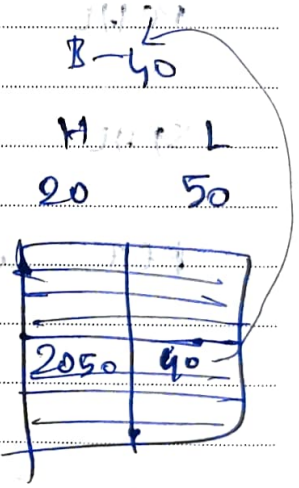
MOV Rd, M

MVI M, #value

MVI R, #value

M → Memory location which is in HL pair (value)

MOV B, M



LDAX Register pair

STAX "

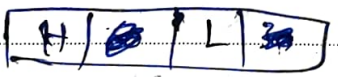
24 Sunday

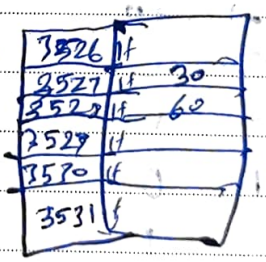
LHLD 16 bit address

SHLD "

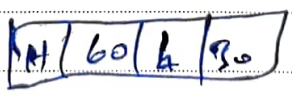
Before

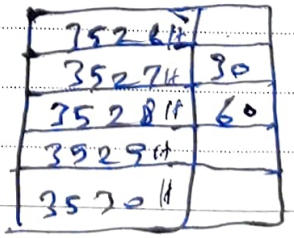
LHLD 3527 H





After





January 25
Monday

before

SHLD 2500 H

H	60	L	30

204F H	
2500 H	30
2501 H	60
2502 H	

After

H	60	L	30

204F H	
2500 H	30
2501 H	60
2502 H	

XCHG - The contents of HL are exchanged with DE Register pair.

SPHL - Move data from HL to SP

XTHL - ~~Transfer data of top of stack~~ L was exchanged with the data of SP and H was exchanged with the data of SP+1

PCHL - The contents of registers HL are copied into the PC

PUSH R_p - Push register pair onto stack

POP R_p - Pop stack to register pair

26 January
Tuesday

January 2021

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

26-339 / Week 5

I/O Instructions

- IN 8bit port address → Copy data to accumulator from a port with 8 bit address.

IN 80H

before: Port 80H

	10

A

After: Port 80H

	10

A	10

- OUT 8bit port address → Copy data from accumulator to 8 bit port address.

ARITHMETIC INSTRUCTION

Addition ~~ADD R / ADD M~~ Addition

Subtraction

Increament

Decrement

ADD R
ADD M
ADC R → R+C+A (C → carry)
ADC M → R+C → Address of HL
ADI 8bit data
ACI 8bit data
DAD ~~16 bit~~ Register pair → Contents of ~~16~~ Rp bit is added to HL Pair. result stored in HL

Subtraction

SUB R =A-R
SUB M =A-M
SBI 8 bit data - A-Data
SBB R - A-R-CY
SBB M - A-M-CY
SBI 8 bit data - A-data-CY

Increment

INR R
INR M
INX Rp

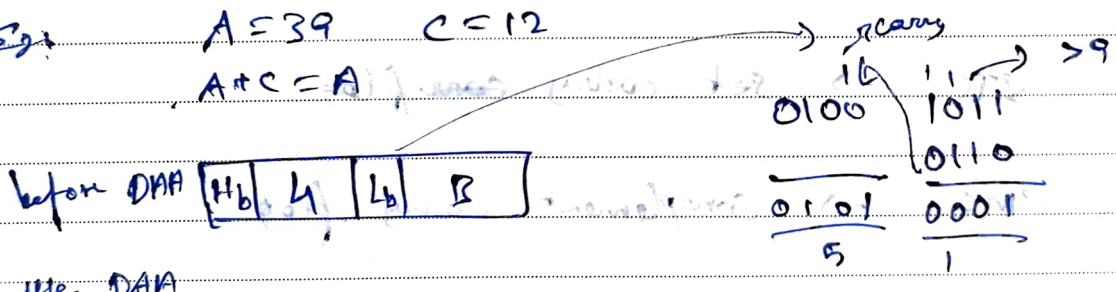
Decrement

DCR R
DCR M
DCX Rp

DAA — Decimal Adjust Accumulator

if $L_b > 9$ add 6 with if
if $H_b > 9$ add 6 with if

Eg: A=39 C=12
A+C=A



Eg: A=96 C=07
A+C=A

28 January

Thursday

January 2021

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

28-337 / Week 5

Full Moon ☺

Logical Instruction

	AND	XOR
AND	ANA R	XRA R
OR	ANA M	XRA M
XOR	ANI data	XRI data
Compare		

OR Compare

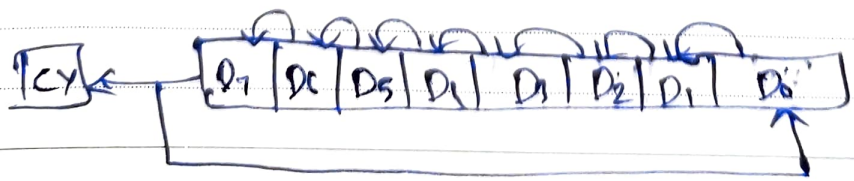
OR	Compare
ORA R	CMP R
ORA M	CMP M
ORI data	CPI data

STC → Set carry flag

CMC → Complement carry flag

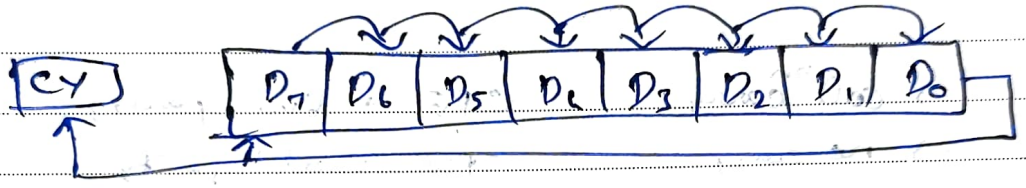
CMA → Complement each bit of Accumulator

RLC → Rotate accumulator Left

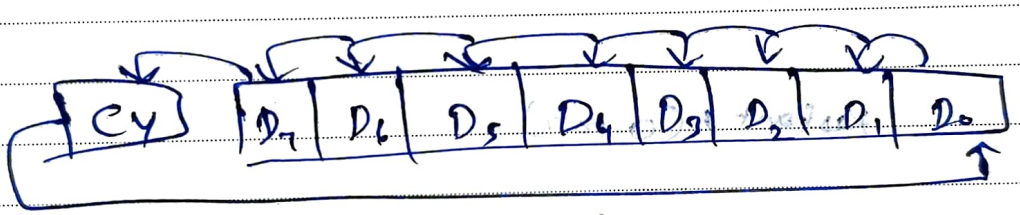


January 29
Friday

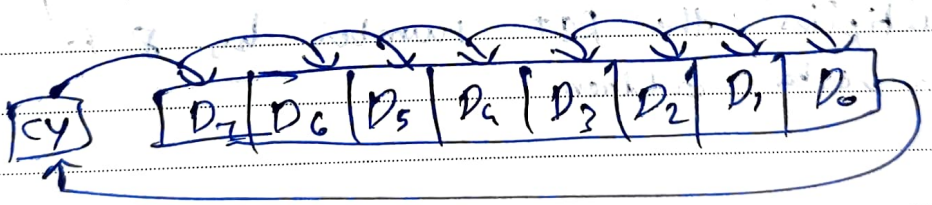
RRC → Rotate Right accumulator



RAL → Rotate accumulator through carry.



RAR → Rotate acc Right through Carry.



Branching Instruction

CALL address
RET

Jumping Instruction

JUMP Address
JC → carry
JNC → No carry
JP → positive

JM → Minus
JPE → Parity Even
JPO → Parity odd
JZ → Zero
JNZ → No zero

30 January

Saturday

January 2021

S	M	T	W	T	F	S
31					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

30-335 / Week 5

~~JNZ → Jump if zero~~

Conditional calls

CC → call on carry
CNC → no carry
CP → positive
CM → minus
CPE → Even parity
CPO → odd parity
CZ → zero
CNZ → no zero

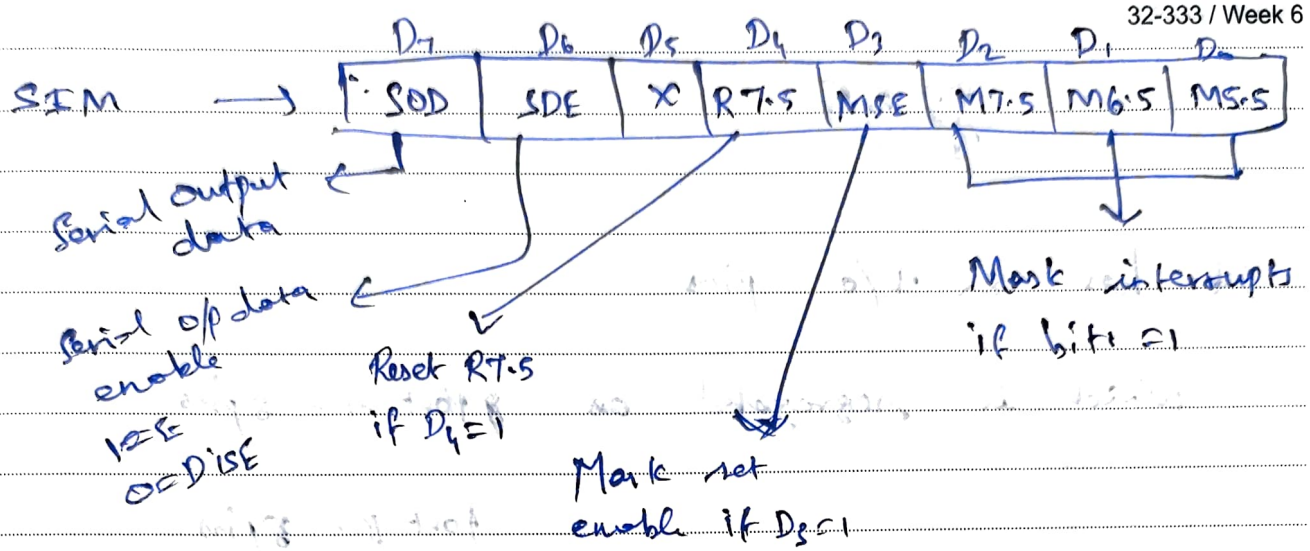
RST n → Restart n (0 to 7)

↳ this instruction transfers the program control to a specific memory address. The processor multiplies the RST number by 8 to calculate the vector address.

31 Sunday

Control Instructions

NOP → No operation → Fetch, decode but no execution
HLT → Halts the program
SIM
RIM → Set Interupt Mask → multipurpose instruction used to implement RST 7.5, 6.5, 5.5 and Serial data ~~input~~ output.



RIM → ~~SID~~ Read Interrupt Mask

Multipurpose instruction used to read the status of interrupt 7.5, 6.5, 5.5 and serial data input.