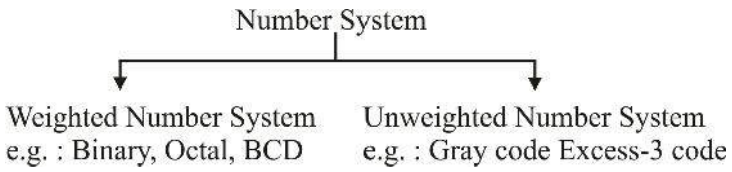
Digital Electronics (Formula Notes)

Number System and Codes:



A number system with base '*r*', contents '*r*' different digits and they are from 0 to $r-1$.

Decimal to other codes conversions: To convert decimal number into other system with base '*r*', divide integer part by r and multiply fractional part with *r*.

Other codes to Decimal Conversions: $(x_2x_1x_0 \,.\, y_1y_2)_r \rightarrow (\mathrm{A})_{10}$

$$\mathrm{A} = x_2r^2 + x_1\, r + x_0 + y_1\, r^{-1} + y_2\, r^{-2}$$

Hexadecimal to Binary: Convert each Hexadecimal digit into 4 bit binary.

$$(5AF)_{16} \rightarrow \frac{(0101}{5}\,\frac{1010}{A}\,\frac{1111)_2}{F}$$

Binary to Hexadecimal: Grouping of 4 bits into one hex digit.

$$(110101.11)_2 \rightarrow \underbrace{0011}\,\underbrace{0101}\,.\underbrace{1100} \rightarrow (35.\mathrm{C})_{16}$$

Octal to Binary and Binary to Octal: Same procedure as discussed above but here group of 3 bits is made.

Codes:

Binary coded decimal (BCD):

- In BCD code each decimal digit is represented with 4 bit binary format.

$$Eg: (943)_{10} \rightarrow \left(\underbrace{1001}_{9}\ \underbrace{0100}_{4}\ \underbrace{0011}_{9}\right)_{BCD}$$

- It is also known as 8421 code

  Invalid BCD codes

  Total Number possible $\rightarrow\ 2^4 \Rightarrow 16$

  Valid BCD codes $\rightarrow 10$

  Invalid BCD codes $16-10 \Rightarrow 6$

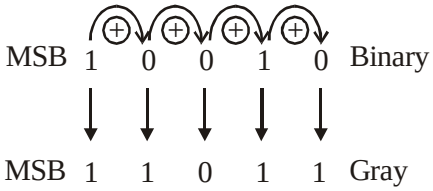
  These are 1010, 1011, 1100, 1101, 1110, and 1111

Excess-3 code: (BCD + 0011)

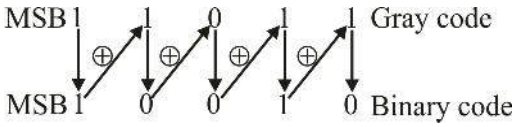
- It can be derived from BCD by adding '3' to each coded number.
- It is unweighted and self-complementing code.

Gray Code:
It is also called minimum change code or unit distance code or reflected code.

Binary code to Gray code:



Gray code to Binary code:



Alpha Numeric codes: EBCDIC (Extended BCD Interchange code)
It is 8 bit code. It can represent 128 possible characters.

- Parity Method is most widely used schemes for error detection.
- Hamming code is most useful error correcting code.
- BCD code is used in calculators, counters.

Complements: If base is **r** then we can have two complements.

(i) (r – 1)'s complement.

(ii) r's complement.

To determine (r–1)'s complement: First write maximum possible number in the given system and subtract the given number.
To determine r's complement: (r–1)'s complement + 1
First write (r–1)'s complement and then add 1 to LSB

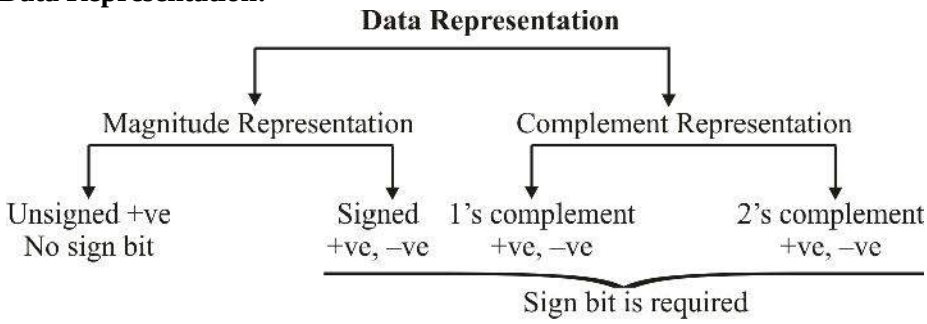
Example: Find 7's and 8's complement of 2456

$$\text{7's complement } \begin{array}{r} 7777 \\ -2456 \\ \hline 5321 \end{array} \qquad \text{8's complement } \begin{array}{r} 5321 \\ +1 \\ \hline 5322 \end{array}$$

Find 2's complement of 101.110
1's complement 010.001
For 2's complement add 1 to the LSB

$$\text{2's complement } \begin{array}{r} 010.001 \\ +1 \\ \hline 010.010 \end{array}$$

Data Representation:



Unsigned Magnitude: Range with n bit $\rightarrow 0$ to 2^{n-1} $\quad +5 \Rightarrow 101$

$-5 \Rightarrow$ Not possible

Signed Magnitude: Range with n bit $\rightarrow -(2^{n-1}-1)$ to $+(2^{n-1}-1)$

$+6 \Rightarrow 0110$ $\quad -6 \Rightarrow \underbrace{\underbrace{1}_{\text{sign bit}}\ 110}_{\text{with 4 bits}}$ $\quad \underbrace{\underbrace{1}_{\text{sign bit}}\ 0000110}_{\text{with 8 bits}}$

1's complement: Range with n bit $\rightarrow -(2^{n-1}-1)$ to $+(2^{n-1}-1)$

$+6 \Rightarrow 0110$ $\quad -6 \Rightarrow \underbrace{1}_{\text{sign bit}}\ \underbrace{001}_{\text{1's complement of 6}}$

2' complement: With n bits Range -2^{n-1} to $(2^{n-1}-1)$

$+6 \Rightarrow 0110$ $\quad -6 \Rightarrow \underbrace{1}_{\text{sign bit}}\ \underbrace{010}_{\text{2's complement of 6}}$

In any representation
+ve numbers are represented similar to +ve number in sign magnitude.

Boolean Laws:

Commutative	$x \wedge y = y \wedge x$	$x \vee y = y \vee x$
Associative	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$	$x \vee (y \vee z) = (x \vee y) \vee z$
Distributive	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
Idempotence	$x \wedge x = x$	$x \vee x = x$
Absorption	$x \wedge (x \vee y) = x$	$x \vee (x \wedge y) = x$
Combining	$(x \wedge y) \vee (x \wedge \bar{y}) = x$	$(x \vee y) \wedge (x \vee \bar{y}) = x$
DeMorgan's	$\overline{(x \wedge y)} = \bar{x} \vee \bar{y}$	$\overline{(x \vee y)} = \bar{x} \wedge \bar{y}$

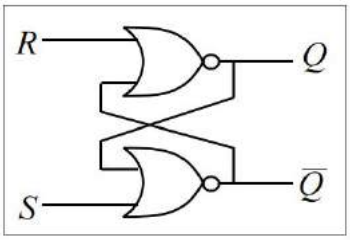
Dual of a function:

The dual of a logic function, f, is the function f^D derived from f by substituting a $\wedge$ for each $\vee$, a $\vee$ for each $\wedge$ a 1 for each 0, and a 0 for each 1.

$$f(a,b) = (a \wedge b) \vee (b \wedge c) \qquad f^D(a,b) = (a \vee b) \wedge (b \vee c)$$

Flip Flops :

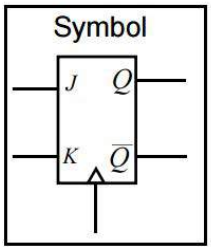
RS Latch



S R	Q' $\overline{Q}'$	comment
0 0	Q $\overline{Q}$	hold
0 1	0 1	reset
1 0	1 0	set
1 1	0 0	illegal

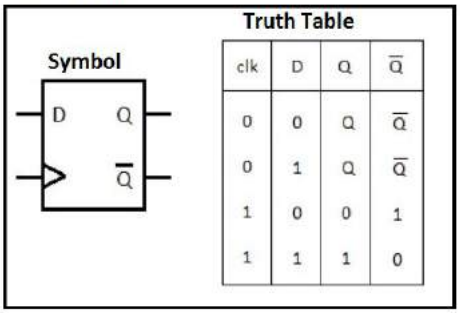
Where Q' is the next state and Q is the current state

JK Flip Flop

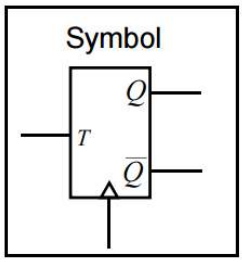


J K	Q' $\overline{Q}'$	comment
0 0	Q $\overline{Q}$	hold
0 1	0 1	reset
1 0	1 0	set
1 1	$\overline{Q}$ Q	toggle

D - Flip Flop :



T - Flip Flop :



T	Q' $\overline{Q}'$	comment
0	Q $\overline{Q}$	hold
1	$\overline{Q}$ Q	toggle

Excitation tables :

		S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

		J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

		D
0	0	0
0	1	1
1	0	0
1	1	1

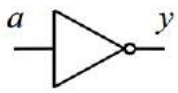
		T
0	0	0
0	1	1
1	0	1
1	1	0

- Q(n+1) = S + R′ Q
  = D
  = JQ′ + K′Q
  = TQ′ + T′ Q

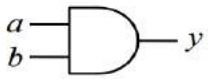
- For ring counter total no.of states = n
- For twisted Ring counter = “2n” (Johnson counter / switch tail Ring counter) .
- To eliminate race around condition $t_{pd\ clock} << t_{pd\ FF}$.
- In Master slave master is level triggered & slave is edge triggered

Combinational Circuits

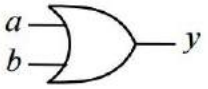
NOT Gate:

Symbol	Truth-table	Boolean
	a | y 0 | 1 1 | 0	$y = \overline{a}$

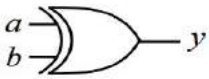
AND Gate :

Symbol	Truth-table	Boolean
	a b | y 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1	$y = a.b$

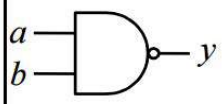
OR Gate :

Symbol	Truth-table	Boolean
	a b | y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1	$y = a + b$

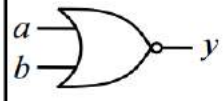
Exclusive Or Gate:

Symbol	Truth-table	Boolean
	a b | y 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0	$y = a \oplus b$

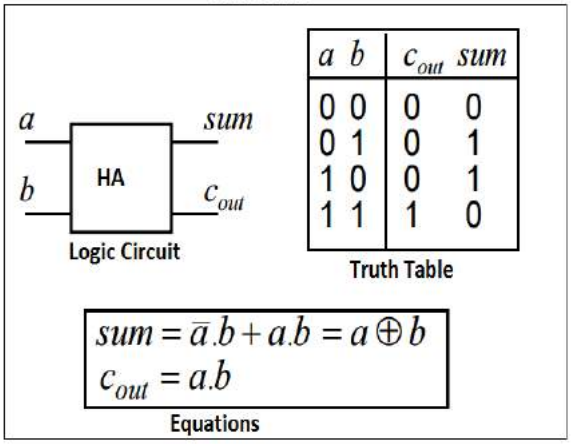
NAND Gate:

Symbol	Truth-table	Boolean
	a b | y 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0	$y = \overline{a.b}$

NOR Gate:

Symbol	Truth-table	Boolean
	a b | y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0	$y = \overline{a + b}$

Half Adder



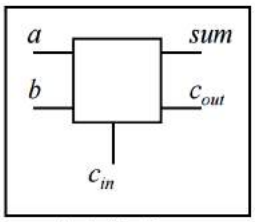
a	b	c_{out}	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Truth Table

$$sum = \bar{a}.b + a.\bar{b} = a \oplus b$$
$$c_{out} = a.b$$

Equations

Full Adder



c_{in}	a	b	c_{out}	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Truth Table

$$sum = \bar{c}_{in}.x + c_{in}.\bar{x} = c_{in} \oplus x = c_{in} \oplus a \oplus b$$
$$c_{out} = c_{in}.(a \oplus b) + a.b$$

Equations

Multiplexer :

- 2^n i/ps ; 1 o/p & 'n' select lines.
- It can be used to implement Boolean function by selecting select lines as Boolean variables
- For implementing 'n' variable Boolean function $2^n \times 1$ MUX is enough .
- For implementing "n + 1" variable Boolean $2^n \times 1$ MUX + NOT gate is required .
- For implementing "n + 2" variable Boolean function $2^n \times 1$ MUX + Combinational Ckt is required
- If you want to design $2^m \times 1$ MUX using $2^n \times 1$ MUX . You need 2 $2^n \times 1$ MUXes

Decoder :

- n i/p & 2^n o/p's
- used to implement the Boolean function . It will generate required min terms @ o/p & those terms should be "OR" ed to get the result .
- Suppose it consists of more min terms then connect the max terms to NOR gate then it will give the same o/p with less no. of gates .
- If you want to Design $m \times 2^m$ Decoder using $n \times 2^n$ Decoder . Then no. of $n \times 2^n$ Decoder required = $\frac{}{n}$.
- In Parallel ("n" bit) total time delay = $2_n\ t_{pd}$.
- For carry look ahead adder delay = $2\ t_{pd}$.

PROM , PLA & PAL :-

AND	OR	
Fixed	Programmable	PROM
Programmable	fixed	PAL
Programmable	Programmable	PLA

Asynchronous Sequential circuits: Asynchronous Sequential circuits do not use a clock and can change their output state as fast as the signal path's propagation delay from the input allows. This means they can be faster than Synchronous Sequential circuits. However, they are considerably more likely to suffer from race conditions (inputs arriving at different times causing different output states) and intermediate output states (as the outputs change from one state to the next final state) than Synchronous Sequential circuits.

Synchronous Sequential circuits: Synchronous Sequential circuits use a clock signal to alleviate the two problems mentioned above. The outputs can only change state with the clock and are designed such that all propagation delays are satisfied before the outputs are allowed to change. This however makes them potentially slower (because the whole circuit must run at the speed of the slowest path in it) and consumes significantly more power due to the extra circuitry required by distributing the clock to all flip-flops, and the continual switching.

Asynchronous Counter	Synchronous Counter
1. Clock input is applied to LSB FF. The output of first FF is connected as clock to next FF.	1. Clock input is common to all FF.
2. All Flip-Flops are toggle FF.	2. Any FF can be used.
3. Speed depends on no. of FF used for n bit . $f_{max}=1/(n*t_n)$	3. Speed is independent of no. of FF used. $f_{max}=1/t_p$
4. No extra Logic Gates are required.	4. Logic Gates are required based on design.
5. Cost is less.	5. Cost is more.

- Fan out of a logic gate = $\frac{I_{OH}}{I_{IH}}$ or $\frac{I_{OL}}{I_{IL}}$
- Noise margin : V_{OH} - V_{IH} or V_{OL} - V_{IL}
- Power Dissipation $P_D = V_{cc}\, I_{cc} = V_{cc}\left[\frac{I \quad +I}{2}\right]$ I → I_c when o/p low
  I → I_c when o/p high .
- TTL , ECL & CMOS are used for MSI or SSI
- Logic swing : V_{OH} - V_{OL}
- RTL , DTL , TTL → saturated logic ECL → Un saturated logic
- Advantages of Active pullup ; increased speed of operation , less power consumption .
- For TTL floating i/p considered as logic "1" & for ECL it is logic "0" .
- "MOS" mainly used for LSI & VLSI . fan out is too high
- ECL is fastest gate & consumes more power .
- CMOS is slowest gate & less power consumption
- NMOS is faster than CMOS .
- Gates with open collector o/p can be used for wired AND operation (TTL)
- Gates with open emitter o/p can be used for wired OR operation (ECL)
- ROM is nothing but combination of encoder & decoder . This is non volatile memory .
- SRAM : stores binary information interms of voltage uses FF.
- DRAM : infor stored in terms of charge on capacitor . Used Transistors & Capacitors .
- SRAM consumes more power & faster than DRAM .
- CCD , RAM are volatile memories .
- 1024 × 8 memory can be obtained by using 1024 × 2 memories
- No. of memory ICs of capacity 1k × 4 required to construct memory of capacity 8k × 8 are "16"

DAC	**ADC**
• FSV = $V_R\left(1-\frac{1}{2}\right)$	* LSB = Voltage range / 2^n
• Resolution = $\frac{\text{step size}}{\text{FSV}} = \frac{V_R/2^n}{V_R\left(1-\frac{1}{2^n}\right)} = \frac{1}{2^n-1}\times 100\%$	* Resolution = $\frac{\text{FSV}}{2^n-1}$
• Accuracy = $\pm\frac{1}{2}$ LSB $= \pm\frac{1}{2^{n+1}}$	* uantisation error = $\frac{V_R}{2^n}$ %
• Analog o/p = K. digital o/p	

- Flash Type ADC : 2^{n-1} → comparators
  2^n → resistors
  $2^n \times n$ → Encoder

Fastest ADC :-

- Successive approximation ADC : n clk pulses
- Counter type ADC : 2^n - 1 clk pulses
- Dual slope integrating type : 2^{n+1} clock pulses .

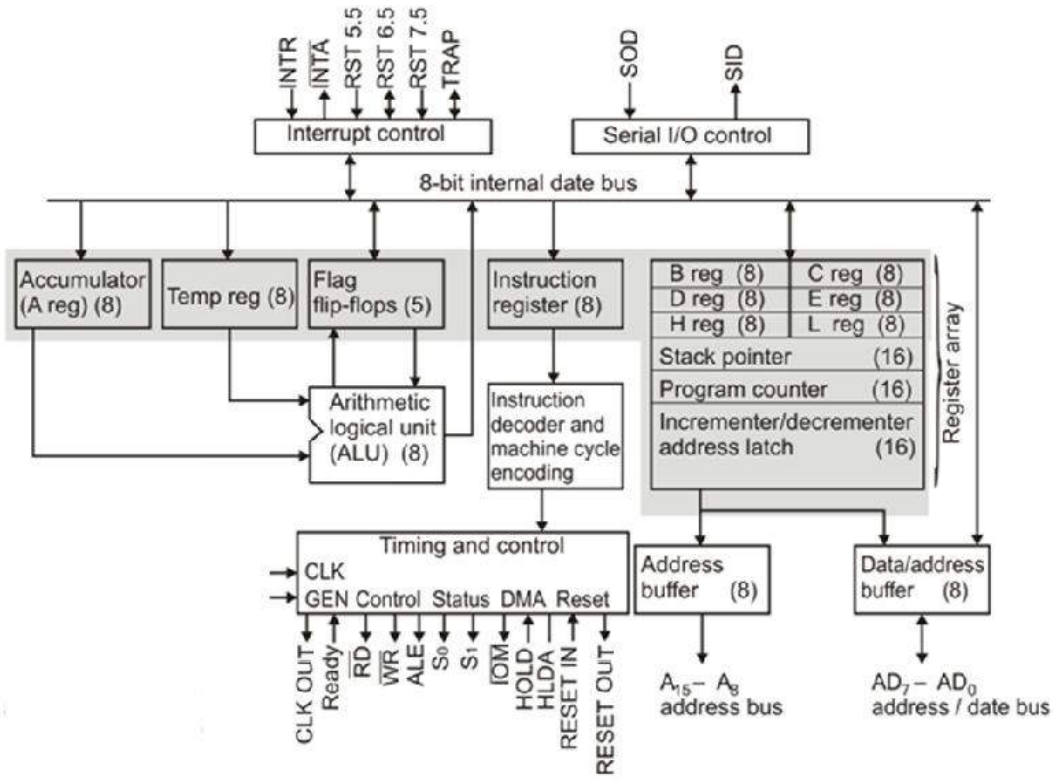
Microprocessor

A **Microprocessor** includes ALU, register arrays and control circuits on a single chip.

Microcontroller:

A device that includes microprocessor, memory and input and output signal lines on a single chip, fabricated using VLSI technology.

Architecture of 8085 Microprocessor



8085 MPU:

- 8 bit general – purpose microprocessor capable of addressing 64 K of memory.
- It has 40 pins, requires a +5V single power supply and can operate with 3 – MHz single phase clock.

Accumulator: Is an 8 bit register that is used to perform arithmetic and logic functions.

Flag Register:

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
S	Z		AC		P		CY

Carry Flag (CY): If an arithmetic operation result in a carry or borrow, the CY flag is set, otherwise it is reset.

Parity Flag (P):

If the result has au even number of 1s, the flag is set, otherwise the flag is reset.

Auxiliary Carry (AC): In an arithmetic operation

- If carry is generated by D_3 and passed to D_4 flag is set.
- Otherwise it is reset.

Zero Flag (Z): Zero Flag is set to 1, when the result is zero otherwise it is reset.

Sign Flag (S): Sign Flag is set if bit D_7 of the result is 1. Otherwise it is reset.

Program counter (PC): It is used to store the l6 bit address of the next byte to be fetched from the memory or address of the next instruction to be executed.

Stack Pointer (SP): It is 16 bit register used as a memory pointer. It points to memory location in Read/Write memory which is called as stack.

8085 Signals:
Address lines:

There are l6 address lines $AD_0 - AD_7$ and $A_8 - A_{15}$ to identify the memory locations.

- In memory mapped I/O ; I/O Devices are treated as memory locations . You can connect max of 65536 devices in this technique .
- In I/O mapped I/O , I/O devices are identified by separate 8-bit address . same address can be used to identify i/p & o/p device .
- Max of 256 i/p & 256 o/p devices can be connected .

Programmable Interfacing Devices :-

- 8155 → programmable peripheral Interface with 256 bytes RAM & 16-bit counter
- 8255 → Programmable Interface adaptor
- 8253 → Programmable Interval timer
- 8251 → programmable Communication interfacing Device (USART)
- 8257 → Programmable DMA controller (4 channel)
- 8259 → Programmable Interrupt controller
- 8272 → Programmable floppy Disk controller
- CRT controller
- Key board & Display interfacing Device

CALL & RET	PUSH & POP
• When CALL executes , µp automatically stores 16 bit address of instruction next to CALL on the Stack	* Programmer use PUSH to save the contents rp on stack
• CALL executed , SP decremented by 2	* PUSH executes "SP" decremented by "2" .
• RET transfers contents of top 2 of SP to PC	* same here but to specific "rp" .
• RET executes "SP" incremented by 2	* same here

Rotate Instructions:

RLC :- Each bit shifted to adjacent left position . D_7 becomes D_0 .

CY flag modified according to D_7

RAL :- Each bit shifted to adjacent left position . D_7 becomes CY & CY becomes D_0 .

ROC :-CY flag modified according D_0

RAR :- D_0 becomes CY & CY becomes D_7

CALL and Return Instructions

CALL → 18T states SRRWW

CC → Call on carry 9 – 18 states

CM → Call on minus 9-18

CNC → Call on no carry

CZ → Call on Zero ; CNZ call on non zero

CP → Call on +ve

CPE → Call on even parity

CPO → Call on odd parity

RET : - 10 T

RC : - 6/ 12 'T' states

Jump Instructions

JMP → 10 T

JC → Jump on Carry 7/10 T

states JNC → Jump on no carry

JZ → Jump on zero

JNZ → Jump on non zero

JP → Jump on Positive

JM → Jump on Minus

JPE → Jump on even parity

JPO → Jump on odd parity .

- PCHL : Move HL to PC 6T
- PUSH : 12 T ; POP : 10 T
- SHLD : address : store HL directly to address 16 T
- SPHL : Move HL to SP 6T
- STAX : R_p store A in memory 7T
- STC : set carry 4T
- XCHG : exchange DE with HL "4T"

XTHL :- Exchange stack with HL 16 T

- For "AND " operation "AY" flag will be set & "CY" Reset
- For "CMP" if A < Reg/mem : CY → 1 & Z → 0 (Nothing but A-B)
  A > Reg/mem : CY → 0 & Z → 0

  A = Reg/mem : Z → 1 & CY → 0 .
- "DAD" Add HL + RP (10T) → fetching , busidle , busidle
- DCX , INX won't effect any flags . (6T)
- DCR, INR effects all flags except carry flag . "Cy" wont be modified
- "LHLD" load "HL" pair directly
- " RST " → 12T states
- SPHL , RZ, RNZ …., PUSH, PCHL, INX , DCX, CALL → fetching has 6T states

- PUSH – 12 T ; POP – 10T