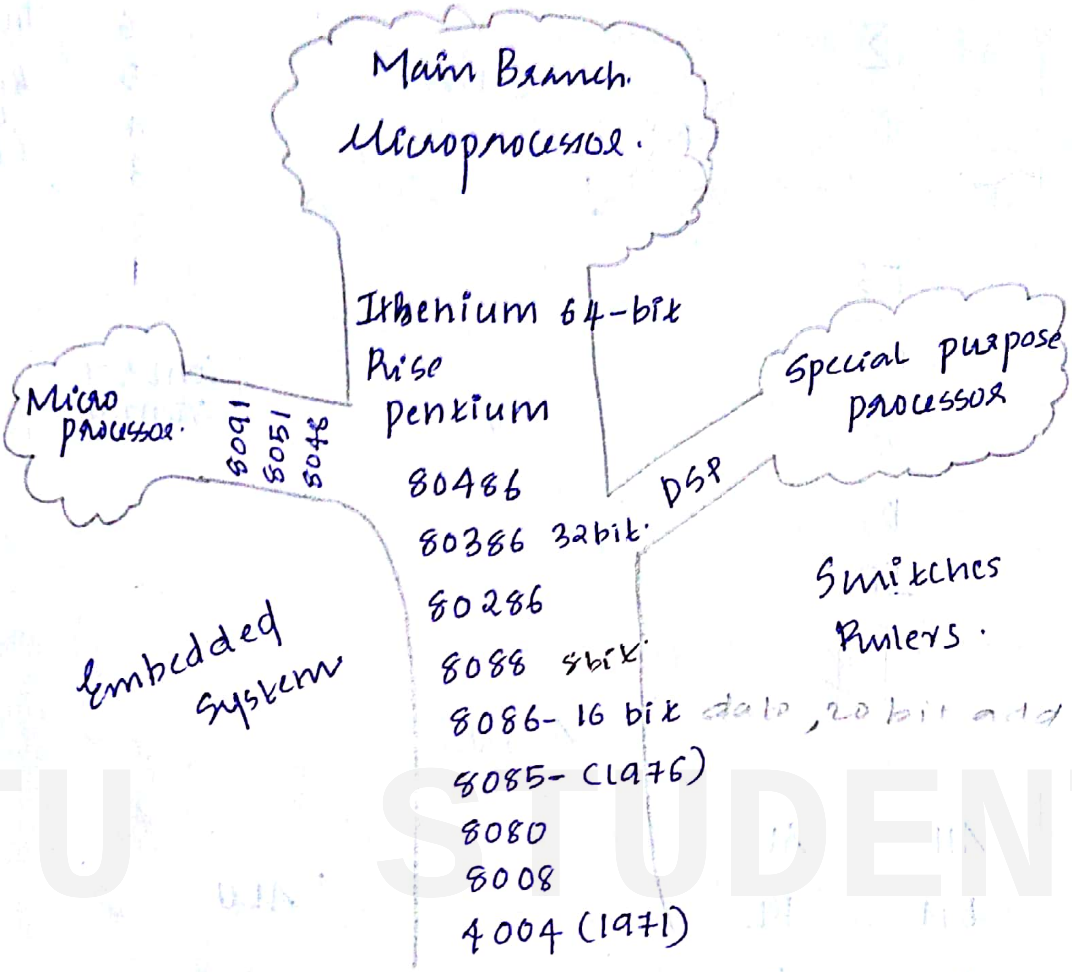
1/8/17

Module 1

Evolution of Microprocessor



2/8/17

X.

8086 microprocessor Architecture & Signals.

16 bit

CPU on a chip is MP

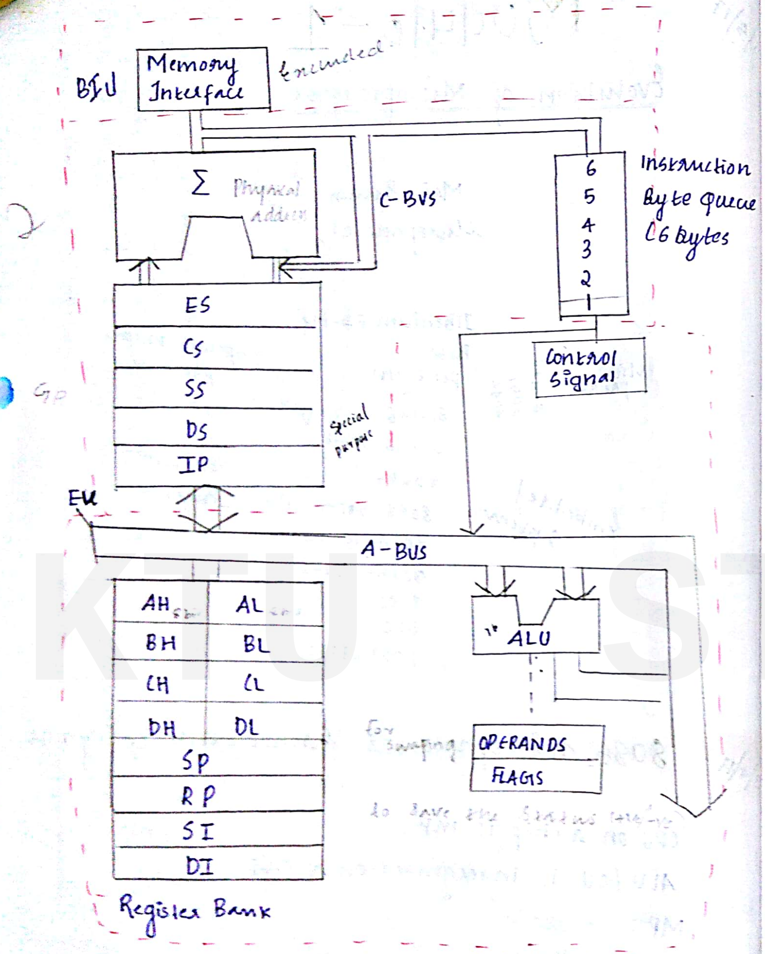
ALU / CU is intergraded on a chip.

MP is a brain

MC is a body

INTEL Developed 4004 in 1971 for calulation.

Data, speed (no:of bits at one second), memory are Parameter of MP.

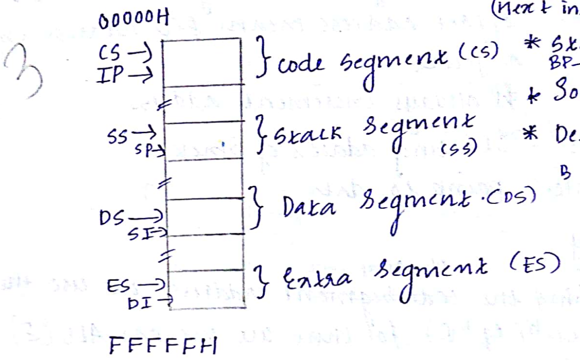


8086 – 16 bit data bus & 20 bit dataline

2^{20} data locations to store data. (1MB)

Flag ← Control flag
  ↳ Status flag – to store the status of ALU.

Memory Segmentation.



* Instruction pointer – offset address (next instruction adress)
* Stack pointer (SP)
  BP – random location
* Source Index (SI)
* Destination Index (DI)

physical address = Segment x 10h + offset. —①

MP can understand only assembly language

programmer writes instructions on mly. MP fetch this instruction from mly to MP. Then decode the instruction ie ~~in~~ 16 bit binary. Then executes the decoded instructions.

Code ~~Signal~~ Segment – Includes programmer code.

Stack segment – ordering the instruction (Push, PoP)

Data segment – Data

Extra segment – additional operations

Pipeline technology is used to speed up the operation

~~Bit~~ Bus Interface unit and Execution unit are two parts.

BIU – fetches the intructions always

EU – Executes.

For more study materials>www.ktustudents.in

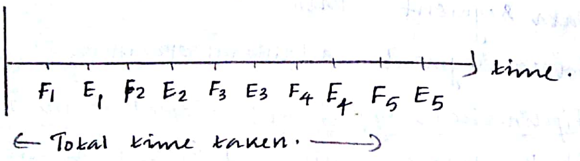
CS address - Starting address.
IP address - Address of executing instruction.
- offset address means file address in a folder.
- IP always increment address
SS address - Starting address of Stack.
Stack pointer - point to data.

Working.

To ~~find~~ the Physical ~~code segment~~ address we use the above equatⁿ(eqtⁿ①) for that we use an ALU(Σ) O/p of Σ goes to Memory Interface then fetches the instruction from ~~queue (FIFO)~~ M/y then insert into queue (FIFO). Then the first instruction moves to control S/m it decodes f assembly language to m/c language. Register Bank is handled by programmer. Data move to ALU from Register bank, then performs the operations. At last the final result is moved to Register Bank to store through A-BUS.

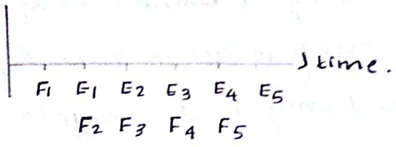
pipelining

8085: Non pipelined processor.



8086: pipelined processor

5



overlapping ~~developing~~ fetching and execution.

← total time →

Disadvantes

Branching

1 → F_1
2 → F_2
3
4
5
6

5/8/17

FLAG REGISTERS

control Status

X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	~~PF~~ X	PF	X	CF

Interrupt flag
1 = Enable Interr
0 = Disable "

Auxillary Flag

zero flag, 1 = result = 0.
0 = result ≠ 0.

overflow flag
1 = overflow
0 = no "

Directⁿ flag
1 = Auto decrement
0 = " Increment

Sign flag
1 = MSB of result is 1 (-ve)
0 = MSB of result is 0 (+ve)

Trap Flag
1 = Single stepping = carry from lower ~~flag~~ nibble to higher nibble.
0 = Do not perform.

Parity flag
1 = even
0 = odd

carry flag
1 = carry out of MSB
0 = No such carry

Parity: Depending on the no: of ones.

Auxillary:

H	L
00001	1011
0001	1011

+ if carry coming to HL from LL then this carry is Stored in Auxillary flag

Zero 5-2 = 3, 0 will be Passed.
6-6 = 0 1 will be Passed.

* Sign figag – Signed numbedrs +ve or –ve
eg: $\underline{1}$001, assume –ve number.
unsigned number only +ve.

- In the case of overflow we can't use Sign flag. It can be applied only to a range (0 – 7F)

* Trap flag: whenever an interrupt occurs during executing then we use debugging for that purpose we use this flag.

* Interrupt flag: Incoming call during a game

* Direction flag: To increment / decrement the execution of a set of instructions.

- Signed numbers.

−128 .... −10 .... +127

−80 ...... 100 ..... 7FH

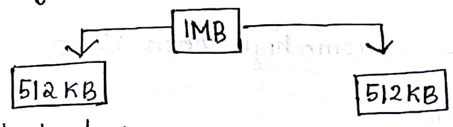
eg: 37H + 29H = ... 42H + 43H

```
eg:  37H      42H      0011   0111  37
     29H      43H      0010   1001  29
     ---      ---      -----------
                       0110   0000
```

CF = 0   SF = 0
PF = 1   OF = 0
ZF = 0

Memory Organization.

Physical Memory Organisation:

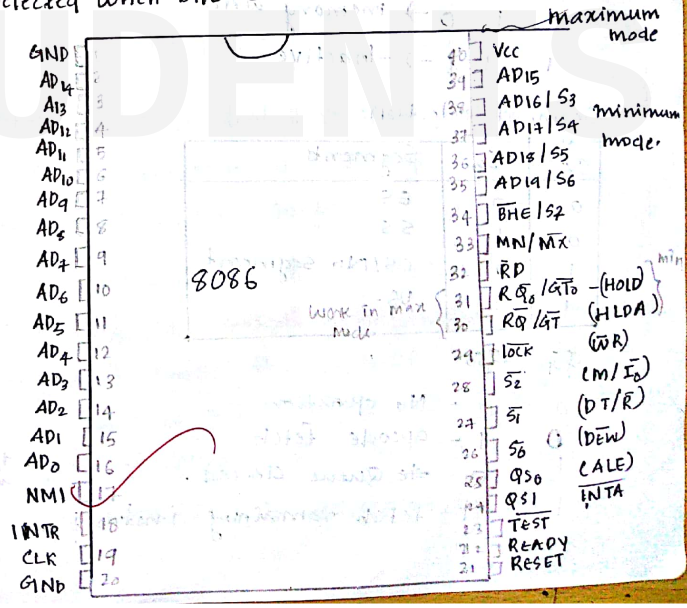


higher bank

lower bank

address range:

higher bank	lower bank
000 01H	0000H
000 03H	0002H
00 005H	0004H
⋮	⋮
FFFFFH	FFFFE
Selected when $\overline{BHE}$ = 0	Selected when A_0 = 0.



* minimum mode → Single processor } Two mode of
maximum mode → Multi processor. } operations.

* $\overline{S_1}$, $\overline{S_2}$, $\overline{Test}$, ... becomes high when i/o = 0.

7/6/16 machine cycle

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	operations
0	0	0	→ INTA cycle (Acknowldgement)
0	0	1	→ I/o Read
0	1	0	→ I/o write
0	1	1	→ half
1	0	0	→ opcode fetch
1	0	1	→ memory read
1	1	0	→ memory write
1	1	1	→ Inactive

Segment Selection.

S4	S3	Segments
0	0	ES
0	1	SS
1	0	CS/No segment
1	1	DS.

QS1	QS0	
0	0	- No operation.
0	1	- opcode fetch
1	0	- ~~de~~ Queue cleared.
1	1	- Fetch remaining instruction.

Reset - Clear every thing
Ready - The bus will ready to receive/send signals
Test - (min mode - only one MP
max mode - multiple MP) To test whether the MP is free or not.

QS0, QS1: at high voltage Instructn will fetch. (11)
to check the status of queue.

S0, S1, S2 : Memory, i/o device, MP - Status can be checked using the values of S0, S1, S2.

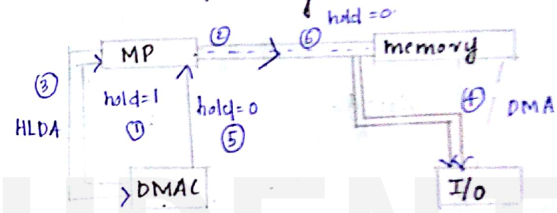


fig: HLD - HLDA

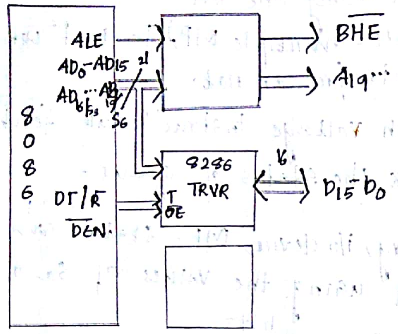
- memory and i/o don't have direct connection.
- To solve direct memory problems.

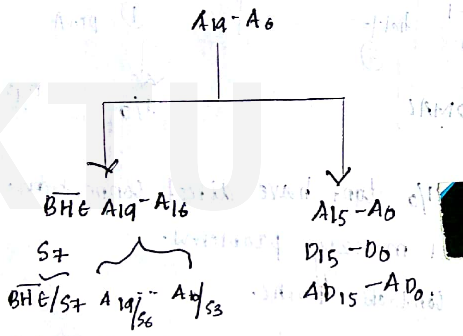
HLD & HLDA controls DMAC.

Here MP 8086 is the BusMaster. At Hold = 1 MP releases the control. This information is acknowledgement to DMAC through HLDA. Then signals pass to I/o ~~hoo~~ now I/o is the bus master. Then hold = 0, again MP becomes the bus master

S5 - INTR flag is accepted by this pin.

Minimum Mode Configuration.





Lock : locks the previous machine instruction till next instructh comes.

In the case of ~~lock~~ DMA lock doesn't ~~wait~~ wait.

- R̄ get activates when 0 is input.

B̄H̄Ē - Bank High Enable. Values are inserted based on this value B̄H̄Ē (odd/even)

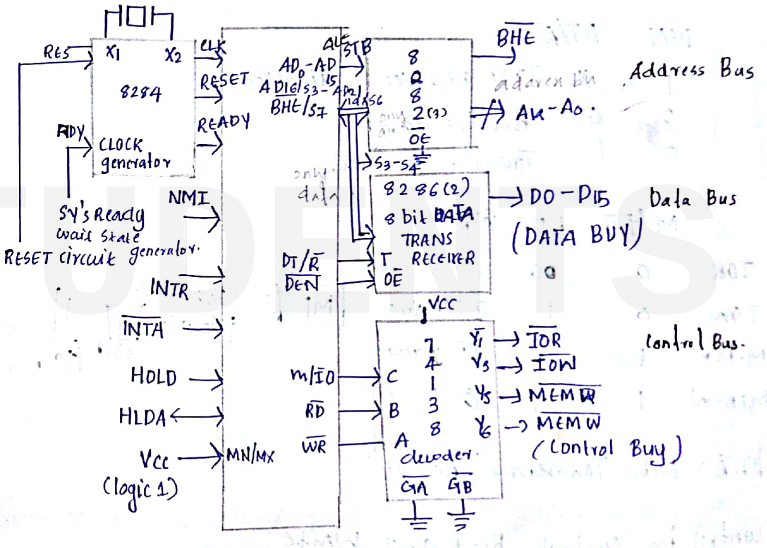
$AD_0 - AD_{15}$: Address and data bus is combined into single bus. Thereby no:of lines is reduced.

NMI : (Non-Maskable Interrupt) - High priority Int

INTR : (Intrrupt Reques) - low priority.

CLK → It is an 8284 ic. & it is a signal

Minimum Mode Configuration: only one MP



- In bus only transmission but in latch storing also included

- ALE is used to identify address ~~ore~~ and data (Address Latch Enable)

ALE=1 → (address pass), ALE=0 (data pass)

- $\overline{OE}$ (output Enable)
- If STB = 1 → address pass → latch enables (unidirection
- ALE = 0 → data pass (bidirection : TRANS-RECEIVER)

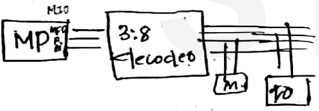
ALE = 1 then $\overline{DEN}$ = ~~0~~1 (no data pass)

ALE = 0 then $\overline{DEN}$ = 0 (active)

- In address chip we need 3 latch bcz 20 bit addra
- In ~~addr~~ data " " " 2 latch bcz 16 data.

$\overline{DEN}$	DT/$\overline{R}$	
1	X	Transreceiver is disabled.
0	0	Receive data
0	1	Transmit data

	M/$\overline{IO}$	$\overline{RD}$	$\overline{WR}$
IOR	0	~~0~~0	1 (I/o read)
IOW	0	1	0 (I/o Write)
MEMR	1	0	1 (M/y read)
MEMW	1	1	0 (M/y write)



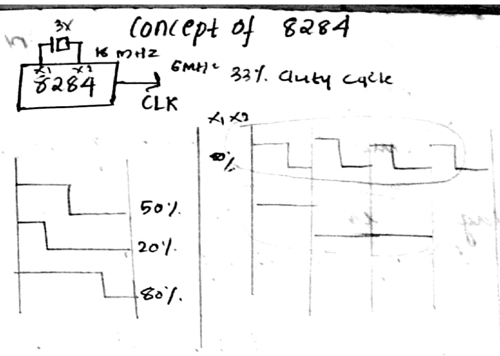
DT/$\overline{R}$: (Data Transmit / Receiver)

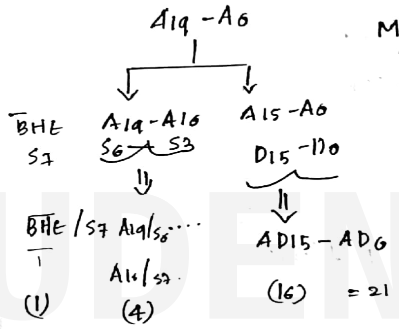
Control bus controls Read and Write.

- $\underline{Vcc}$: min - Vcc = 1

  $\overline{max}$ - Vcc = 0.

Concept of 8284





= 21 after multiplexing.

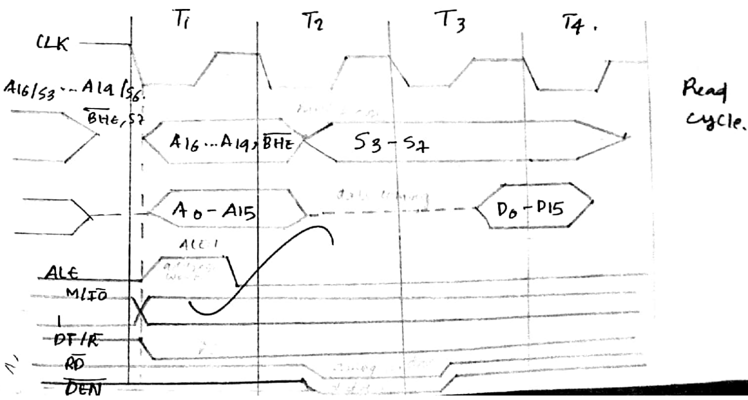
Multiplexing

21 + 21 = 42

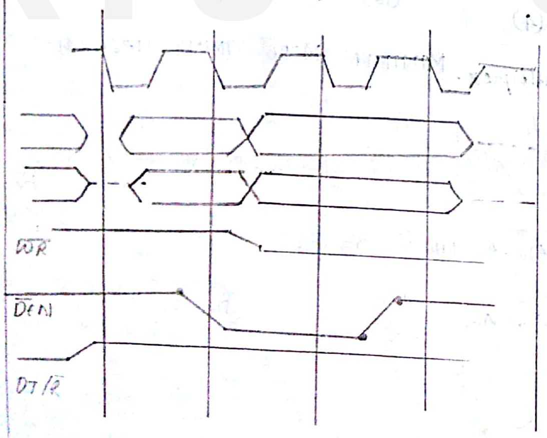
by Multiplexin

21 lines

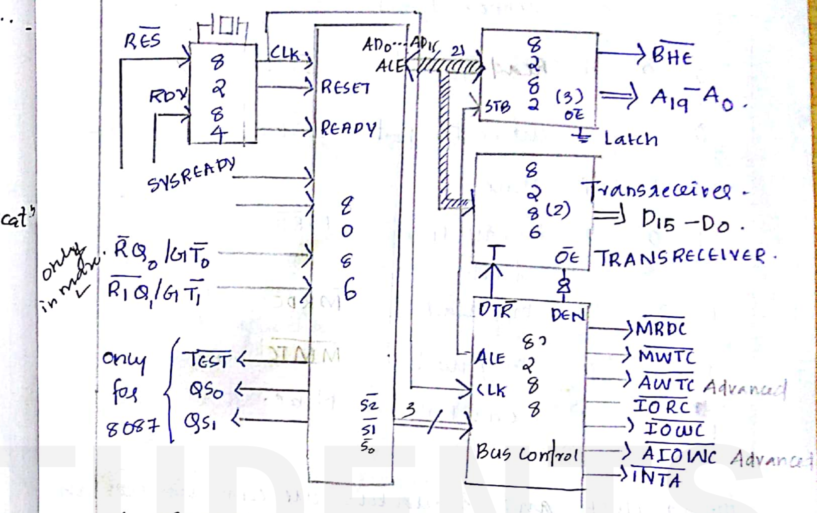
9/8/17 ~~Then demultiplex~~. MINIMUM MODE TIMING DIAGRAM.



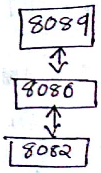
- MP knows when the operations is completed only using trigger.
- m/c cycle : instruct^n reading, fetching, executing...
- Time diagram includes the time taken for reading, fetching, executing... It takes only nano seconds.
- T_1 (trigger) - instruct^n fectchig, identifying locat^n address
- T_2 - Need data → $\overline{R}=0$
- T_3 - Data Transmission
- T_4: Data storing.
- $DT/\overline{R}$: $\overline{R}=0$ → Read
- ALE should be lowered first.



8086 - maximum mode configuration.



- Multiprocessor -
- $\overline{RQ_0}$ & $\overline{RQ_1}$ is used to identify which processor should be the busmaster.
- defaultly 8086 will be the busmaster. If 8089 requestes for the busmastership then 8086 grant the request.



- 8288 bus controller controls all the address & data bus.
- Information is given to Bus controller . through ~~for~~ 3 Status signal $\overline{S_0}$, $\overline{S_1}$ $\overline{S_2}$

$\overline{S_2}$	$\overline{S_1}$	$\overline{S_0}$	PROCESSOR	8288 OPERATIONS.
0	0	0	interrupt acknowledgment	$\overline{INTA}$
0	0	1	Read I/o Port	$\overline{IORC}$
0	1	0	write I/o Port	$\overline{IOWC}$
0	1	1	Halt	
1	0	0	Instruction fetch	$\overline{MRDC}$ m/y
1	0	1	m/y read	$\overline{MRDC}$
1	1	0	m/y write	$\overline{MWTC}$
1	1	1	inactive	None.

- Using clock and 3 status we can understand the value of ALE

- If CLK - 1st state - address. — ALE = 1
If CLK - 2nd state - Data. - ALE = 0 . . . .

- DT = 1 → write   DT = 0 → Read

- AMWTC - Advanced memory write controller.
MRDC - Memory Read controller.
MWTC - Memory write controller.

Comparision b/w 8086 and 8088.

8088	8086
• 16 bit processor with 16 bit ALU. and 8 bit data bus.	• 16 bit processor with 16 bit ALU and 16 bit data bus.
• Memory banking not needed. hence ckt is simpler.	• Memory is divided into two banks and ckt is more complex.
• Since data bus is 8 bit, it can transfer one byte in one cycle hence it is slower.	Since data bus is 16 bit, it can transfer two bytes in one cycle hence it is faster
• BHE is not needed instead as a signal called SSO used for single stepping	$\overline{BHE}$ is needed to enable the higher bank
• Pre fetch queue is 4 bytes	• pre fetch queue is 6 byte.
• It uses I/o or $\overline{M}$. compatible with 8086	It uses M or $\overline{I/o}$ to differentiate between m/y & I/o operatn