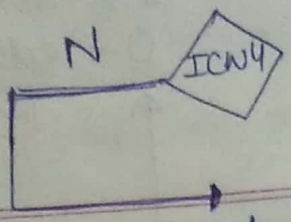


ASHOKA
Date:
Page No.

Ready to accept Interrupt

Call 2030 :-

			MW	MW	
$T_1 - T_6$ OF	MR (30)	MR (20)	PCL SPH SPL PCL	(PCH)	SP_H SP_L

RET :- uncoditional

			RET conditional
$T_0 - T_4$ OF	MR	MR (SP)	$T_0 - T_6$ OF

29/10/18 call means, program counter keeps the program in stack. Put the content of PC in stack · SPH and SPL.

What if we want to do POP?

Pop rp, (HL)

stack content is getting out

T_6	OF (POP)	MR SPH SPL (L) 1	MR SPH SPL (H) 2

CNC 8020

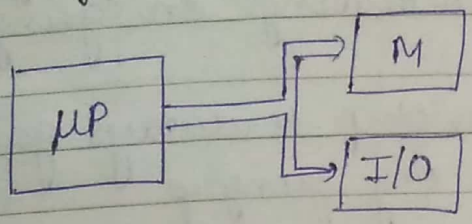
OF	MR	MR	MW	MW
$T_1 - T_6$	HOB LOB (20)	HOB LOB (80)	SPH SPL PCL	SPH SPL PCH

SPHL, PCHL, XTHL, conditional call, INX, DCX, conditional jump

$T_1 - T_6$ → OF

ASHOKA
Date:
Page No.

Memory :-



How CPU and I/O devices linked to Memory

→ Type of Memories and Interfacing of Memory
How μP and Memory work together?

classification of Memory :-

- Semiconductor
- Magnetic Disk
- Magnetic Tape

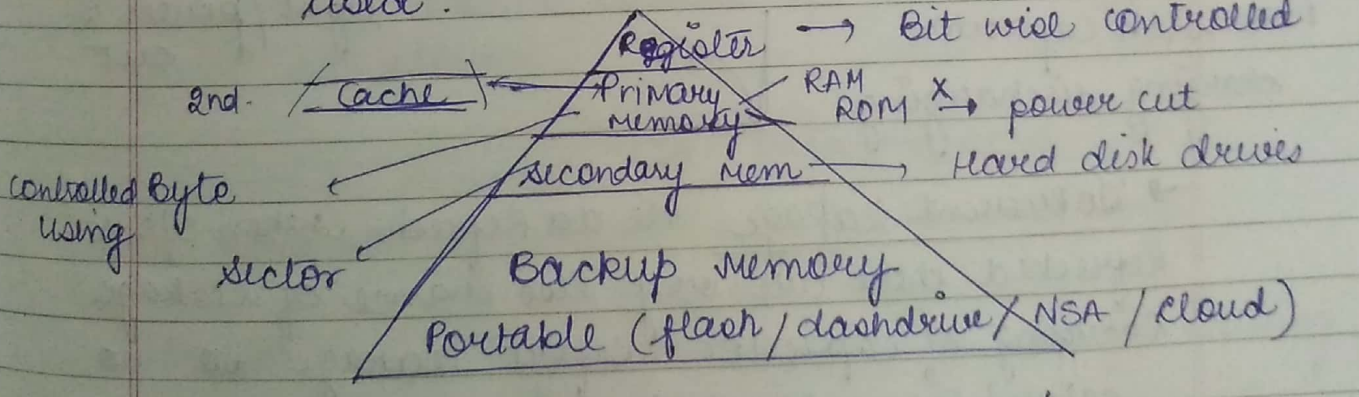
Access time (Access to Memory location)

Storage capacity

Access time → Time which we spend just to locate Memory address

Memory Pyramid :-

Label wise how much Memory is going to be used.



Cache

label 1 → inside CPU (on chip)
label 2 → on Board
label 3 → out of Board

Main Memory = semiconductor

Sec : HDD

ASHOKA
Date:
Page No.

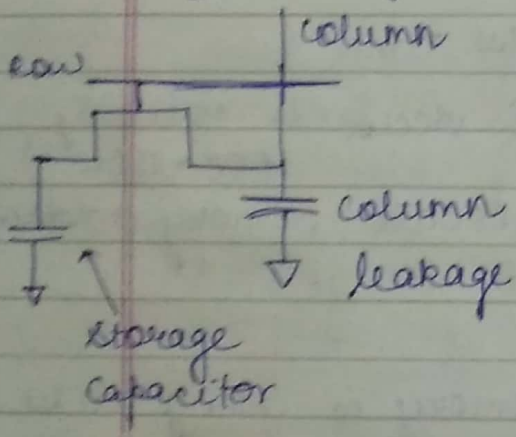
On the Pyramid

1) Memory size → Increasing on going down
2) Access time → Increasing on going upwards

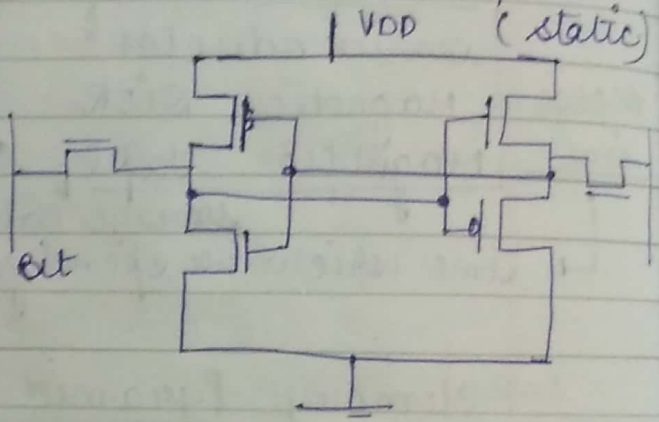
3) cost per byte → on chip is always going to be expensive. ↑sing on going up
4) Unit of Transfer →

Semiconductor RAM

DRAM (A) (dynamic)

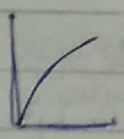


SRAM (B) (static)



Inf. stored even when light/power is cut

Prob with capacitor charging discharging



→ To prevent leakage we do Refresh, when S/S is Refreshed there are very less chance of leakage. Charging of capacitor Before leakage so no corruption of data.

→ A have more memory in less Area. Most Memory is always DRAM. Density of DRAM is high

1)
2)
electrica
3)
progra
4)
electron
erasa
5)

31/10/18

#

	Memory Type	Category (R/W)	Erase	write	Volatility
1)	RAM	R/W	electrically byte level	Electric -ally	Volatile
2)	ROM	R	— x —	Mask	Non Volatile
3)	electrically programmable E PROM	Read Mostly	full chip level	electri cally	Non volatile
4)	EEPROM electronically erassable Prog	—"—	—"—	—"—	—"—
5)	Flash	R/W	Byte level —"—	—"—	—"—

31/10/18

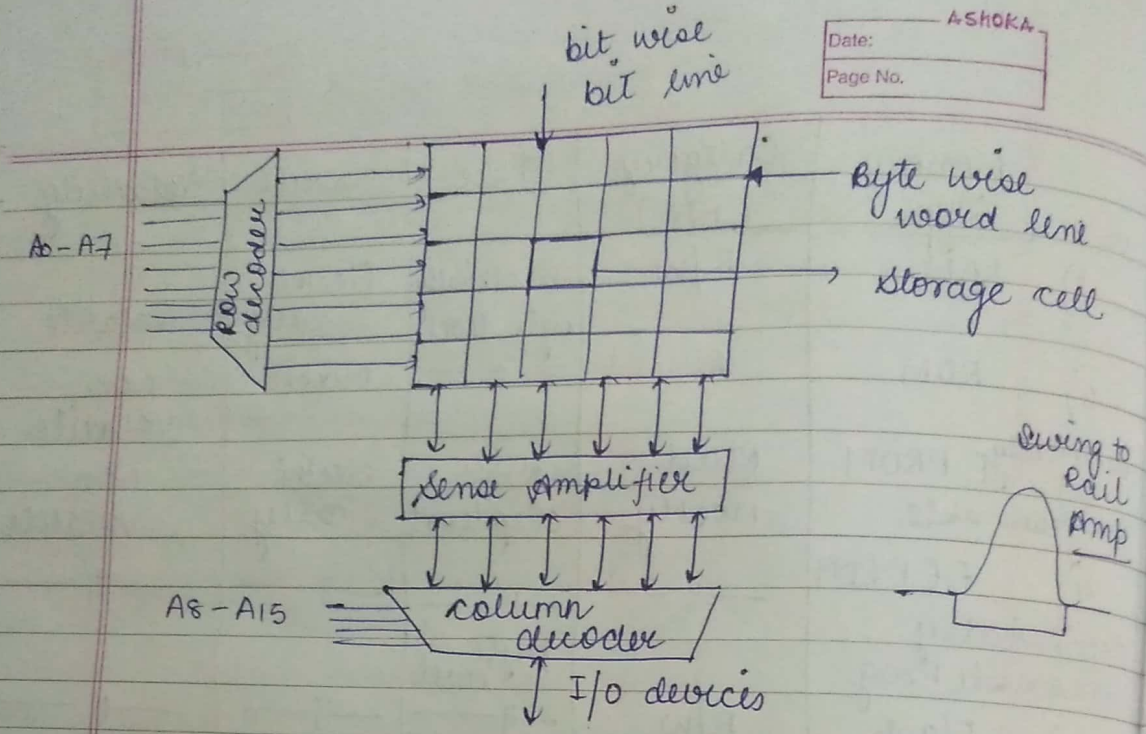
Density = No. of components in chip.

SRAM is faster because it is purely semiconductor devices. SRAM is very expensive compared to SRAM. DRAM consits of electrical components also so slower.

Primary memory consists of → Data memory
→ Program memory

Primary memory is for usability Not for saving.

In flash shadow memory get generated that's why no of cycles gets decreased in this. so we are not able to load data freshly every time That's why we use DRAM inplace of a flash.

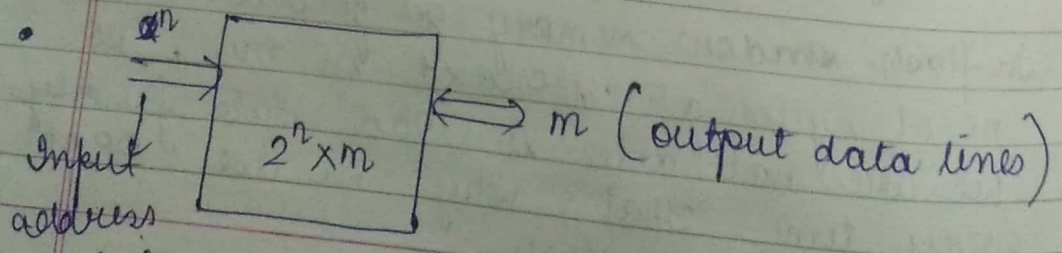


→ Purpose of sense amplify is to swing to rail (it generates a pulse → byte line amplitude or bit line)

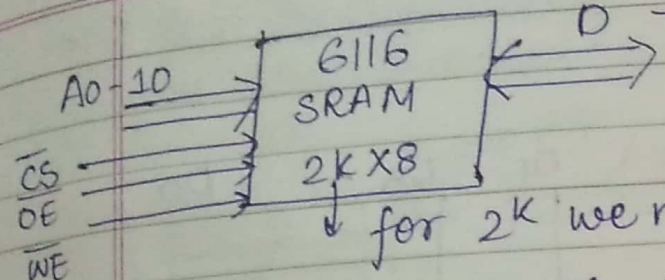
To Read & write we need some voltage levels for that we need amplifiers.

Chip Packaging →

- Adder lines
- Data lines
- Power lines
- GND lines
- $\overline{CE}$ → when it is active low to select the status of chip
- OE (output enable)
- R/W control



Chip structure :-



SRAM.

↓ for 2^K we need 11 address lines

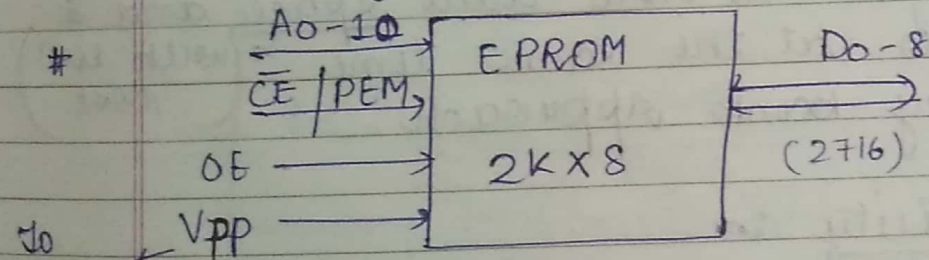
How many n we want · decided by K.

No. of K directly associated with address lines.

→ for 6K we need 16 address lines

128K memory ⇒ 17 address lines, But in 8085 we have only 16 Address lines Then how we will attach this memory ?

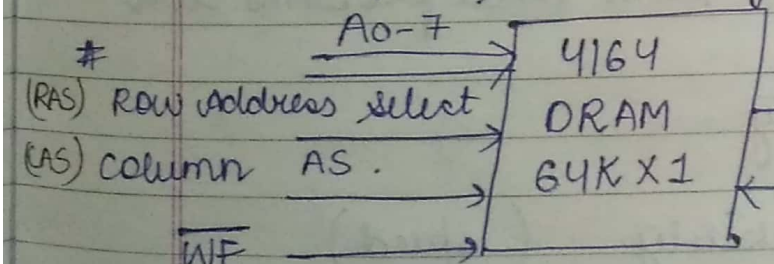
- CS → chip select once it is on only then we can access our full memory.
- WE → for Read / write.



EPROM

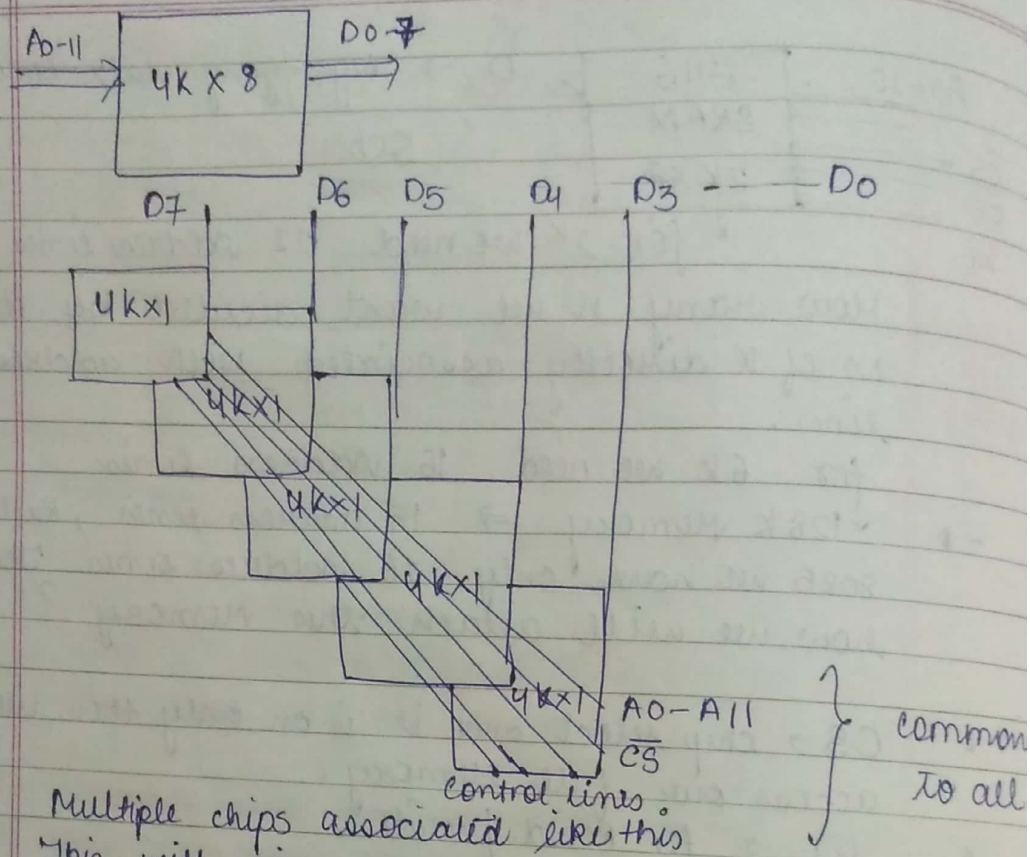
To get extra voltage levels

→ only 8 Bit data Bus for Address



DRAM

(Here we use Rows and columns to point to a particular Memory location).



Multiple chips associated like this
This will give us more data space and 8 data lines at the same time (with low cost)
Memory Bank Approach.

→ Incompatibility :-

while working as a team what problems can occur →

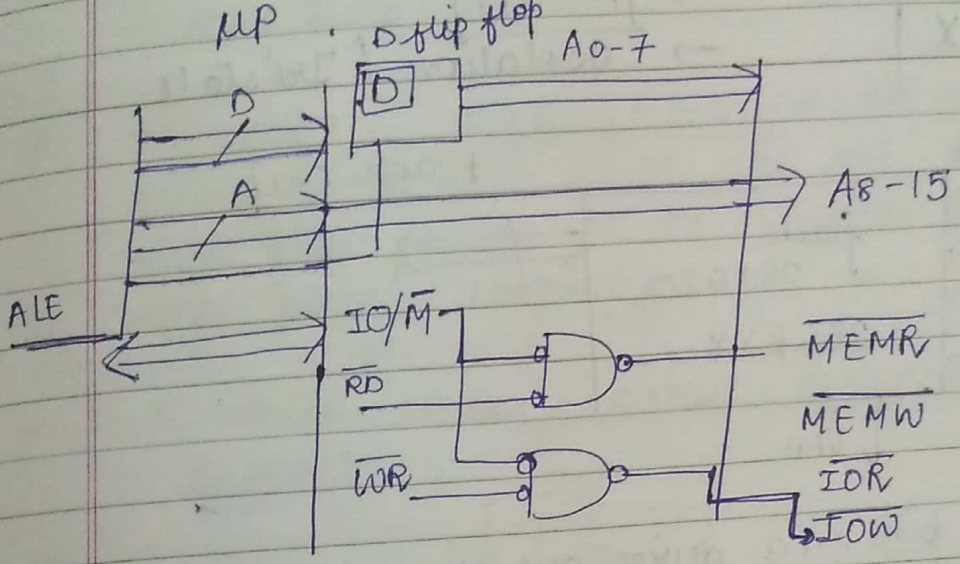
1) Bus Incompatibility
2) Electrical "
3) Timing incomptability (speed)

→ we need to on some devices and off some device to get some more power.

chip select is the way how we will organize the whole circuitory.

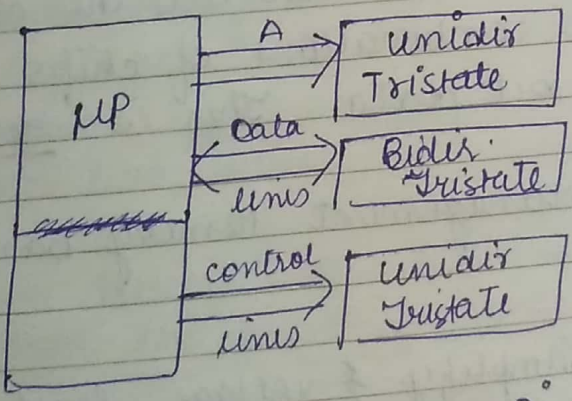
→ (1) Bus Incomptability → chip select and other signals are also involved in device. Data Bus has to be address bus for some time we use D latch here.

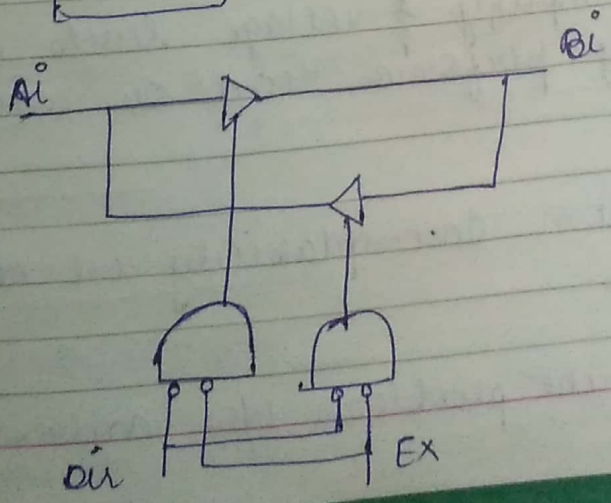
µP . D flip flop

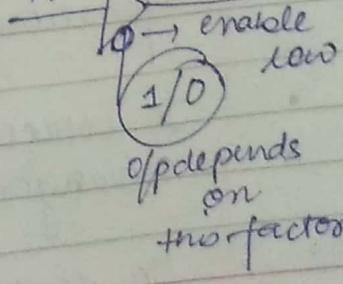


(we need to create logic on our own.

2) Electrical: Tristate Buffer →







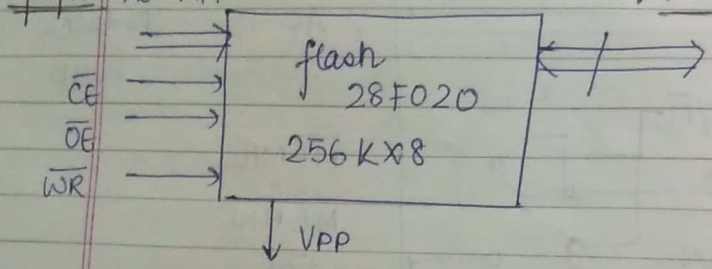
D | E

ASHOKA
Date:
Page No.

direction of data flow →

D	E	direction
0	0	→ B → A
1	0	→ A → B
X		→ Isolation / Tri'state

2/11/18 A0 - A17



No. of pins in a device are very important. To know the complexity of circuit and other things we need to know about no. of pins.

- On a single wire we can't have all the data That's why we make combinations of chips to get 8 Bit or 16 bit data. This is Memory Banking.
For memory management Memory Bank work fine.

- swing pulse Amplify & voltage levels so that we can easily perform Read or write operation.

- To overcome from Incomptability we need some arrangements.

- To sort out the problems b/w address and

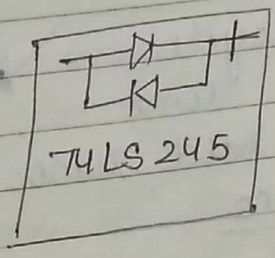
Address lin
CS
Not
B/W µP

ASHOKA
Date:
Page No.

Data bus we use D latch so that Data Bus can be made Address bus for some time.

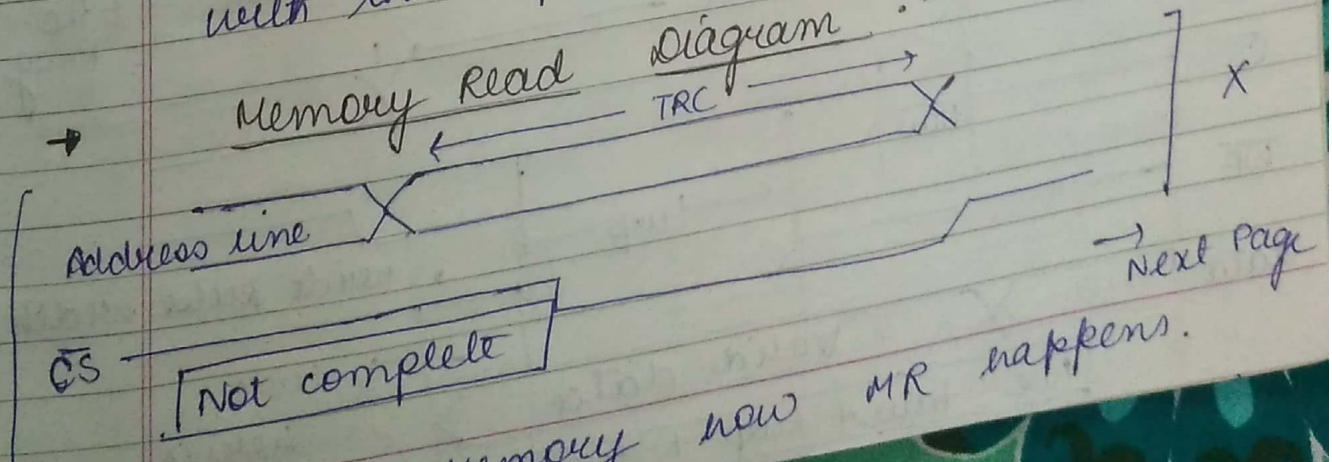
→ In electrical Incomptability → we simply shut down the devices which are not in use and give power to the devices which need power urgently.

→ IC which contains circuit of Bidirectional Buffer :-



- With many devices connected it is not possible to track all the softwares related with devices.
- Even two same device, connected Because of Incomptability Two device can't communicate properly
- with the help of Ready signal. one slower communicate

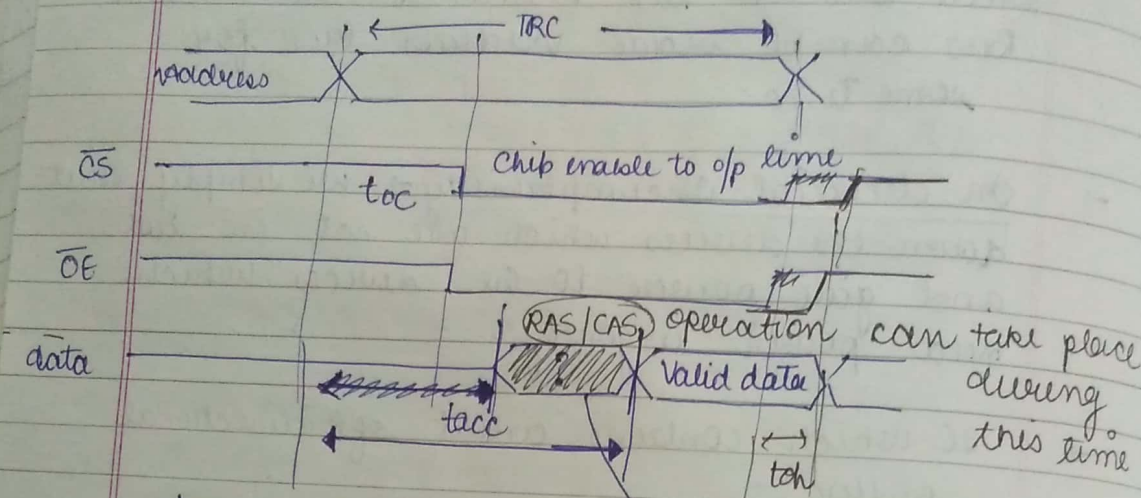
→ Memory Read Diagram :-



B/w μP and memory how MR happens.

ASHOKA
Date:
Page No.

Memory Read cycle :-



tRC → Read cycle
tacc → Access time
toH → O/p hold time
tRC = tacc

This box is for the accessibility and to perform some calculations

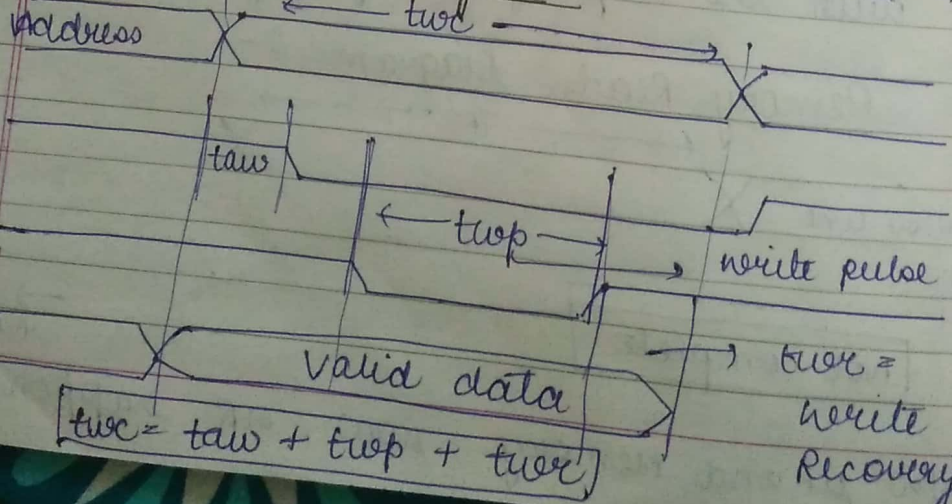
We assuming that CPU have already enabled the Read pin.

→ After tacc and after one block, Valid data starts get loading on the CPU.

→ toh → to close the operation

Write cycle diagram :-

write → Elec. Prog



twr = write Recovery time

In this cycle one operation gets over

Scanned by CamScanner

taw = Address setup time

ASHOKA
Date:
Page No.

If we know tacc time then we get to know about the full operation.

→ After access Read operation starts taking place

In case of Memory Read cycle → when we know about tacc → we get to know about the time of initiation and then Memory Read cycle continues and operation takes place.

→ In case of write cycle → only when we have trw → only then write operation takes place. In this we need to add all the time.

→ why extra Box not present in Write cycle? RAS and CAS need not require extra time or setting time for write cycle.

• Whenever we design a Memory two things are important →

→ Address setup time & data setup time

tad + tds = 225 ns (for MW / MR)

In this cycle one operation gets over → $2\frac{1}{2} \times 320$ = one state time = 800 nsec

tad + tds + tACC = 800 nsec (both)

tACC = 800 − 225 = 575 nsec (for MR / MW)

tacc ≤ 575 nsec

Address setup, data setup and time to access the Memory → Imp

MIPS → Million Ins. per second.

ASHOKA
Date:
Page No.

- In 800 nsec both Read and write cycle can take place.
- tacc should not be more than 575 nsec
tacc is neither Read time nor write time
It is the time Required to go to a particular Memory location.

12/11/18

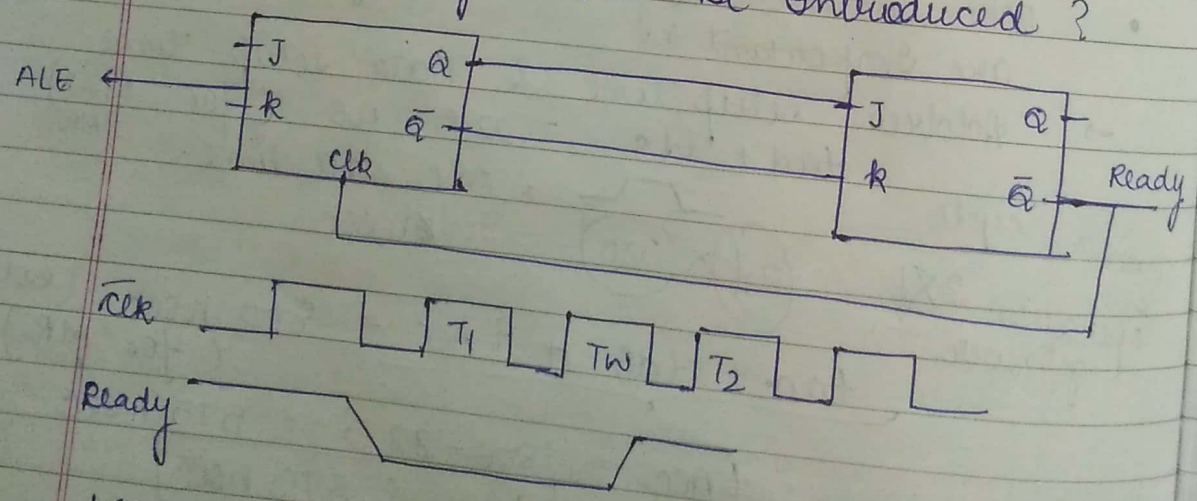
Interfacing of slow Devices :-

- How Many MIPS you can Made Makes you Powerful. If we itself gets slow for a slow device that is not a feasible option.

1) Slow down → Reduce frequency of your own device.
the frequency

2) Introduce wait cycle → Master is connected with a no of different devices and any change in Master affects all.

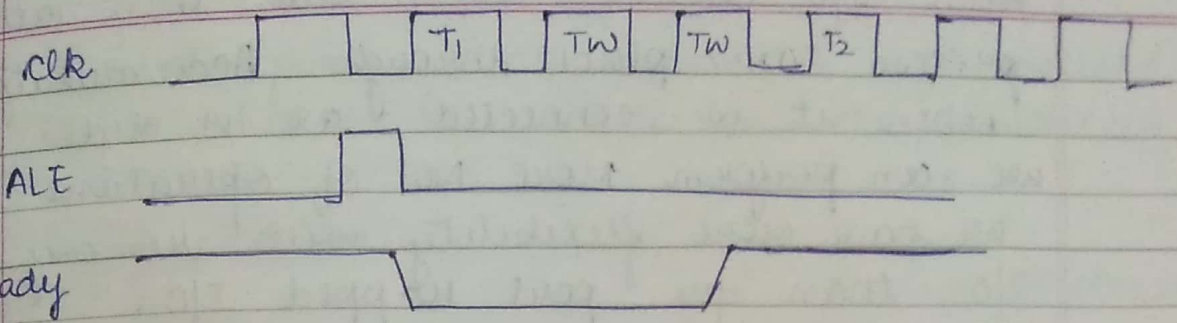
How wait cycle can be Introduced ?



We use Ready, to know about the state of occupancy or not.

Scanned by CamScanner

Ready active low → device is Ready to communicate.



Ready is '0' then device go to wait state.

$\overline{Q}$ is low → we set the circuit of JK flip flop

Address space :-

Address means how address lines we have and space means how far we can go.

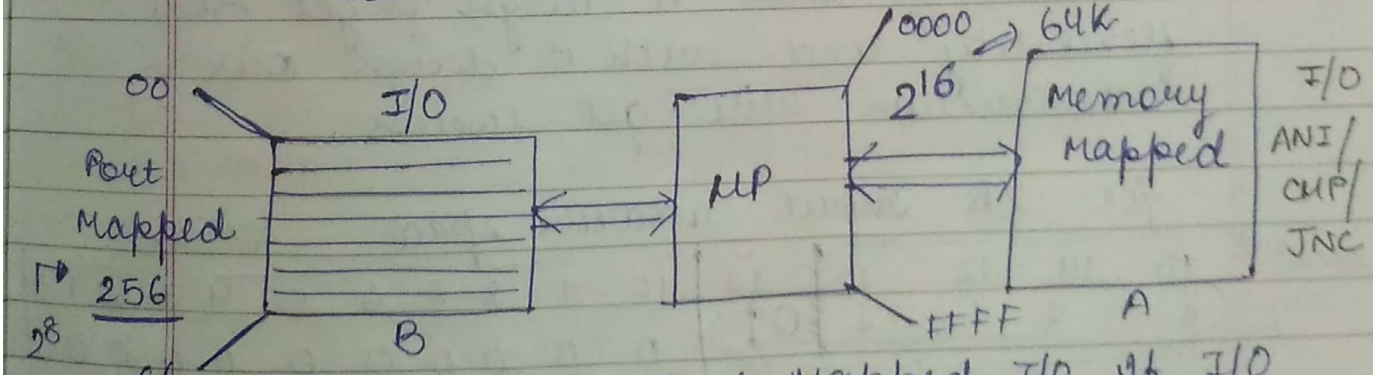
eg $\frac{2K \times 8}{\Downarrow}$

we need 11 address lines so by 11 add lines we can create 2K address space.

Memory Mapped and Port Mapped

↳ we use full address Bus

↳ we use less size of address Bus.



This design is Port Mapped I/O if I/O attached to A side then Memory Mapped I/O

At B side IN/OUT operation will take place

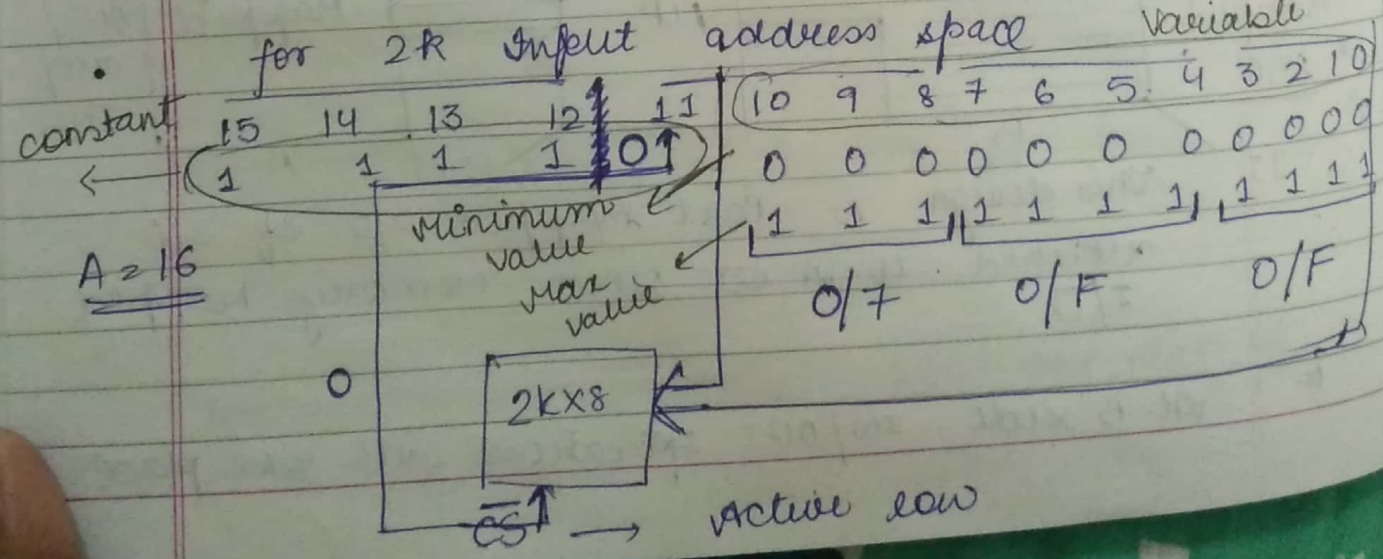
ASHOKA
Date:
Page No.

when I/O connected at B side then all the process takes place through Accumulator But when it is connected at A side Then we can perform more No. of operations. We have more flexibility with memory mapped I/O than for port mapped I/O.

MM I/O	PM I/O
→ Varieties of addressing mode.	→ don't have such kind of facility
→ No need of any separate Instruction	→ IN/OUT PS (Instruction)
→ address space is occupied	→ Here we have diff address space But we need one extra pin
→ communication happens through memory address	No data Manipulation/ corruption (Port address)

- 2k X 8 Memory chip
we will go with A design to get more flexibility and with B design our Instruction will get useless.

- for 2k Input address space



and the pins 15-12 can't be left unused so we have to utilise all the resources.

Total address = F000 (Initial Address)

↓

Packed BCD / Hexadecimal format (Byte wise)

Interfacing is making device communicate to the fullest.

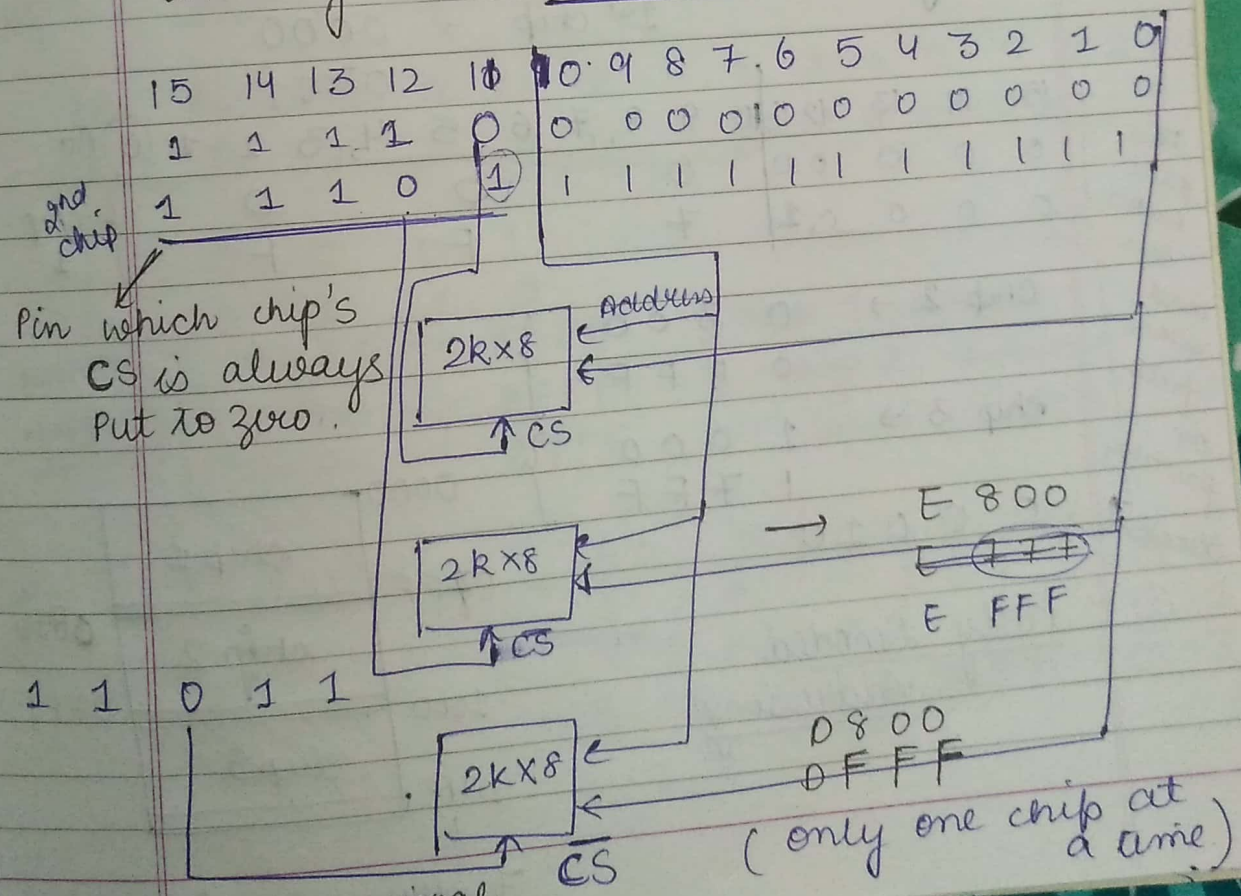
F000
F7FF
FFFF
→ Full Address

when given 2 × 2k × 8

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1

2nd chip

Pin which chip's CS is always Put to zero.



Partially decoded Addressing

ASHOKA
Date: 14 | 11 | 18
Page No.

chip1

IC1 [F000
F700]

→ shadow memory

IC2 [E800
EFFF]

→ shadow memory

IC3 [0800
0FFF]

→ Not strategic just utilising Resources But not in managed way.

→ We need to avoid shadow memory. How can I avoid shadow memory.
Memory should be like one after one But it is not coming this way
For this IC's connected should be in a planned way.

1st chip → 0000
07ff

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	A0
0	0	0	0	0	0			0				0				chip 1
0	0	0	0	1	7			F				F				

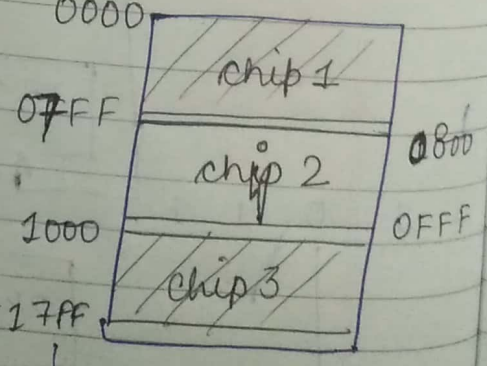
this part makes our addi either full or partially decoded

Chip 2 → 0800
0FFF

chip 3 → 1000
17FF

00010

Fully Decoded Addressing



Further expansion

step 1
step 2
step 3
step 4

A15
1
1
1
memory mapping

Bit

ASHOKA
Date:
Page No.

In Partially decoded Addressing →

1) wastage of Memory
2) Possibility of bus contention

Interfacing of 6K x 8 Memory :-
Design and Interface 6K x 8 Memory using 8085 with 2K x 8. Starting Address is 8000H.

Step 1 for 6K = 2K + 2K + 2K
Step 2 Address lines = A0 - A11 (for each 2K)

Step 3 Remaining Address lines → Use them for chip select $\overline{CS}$

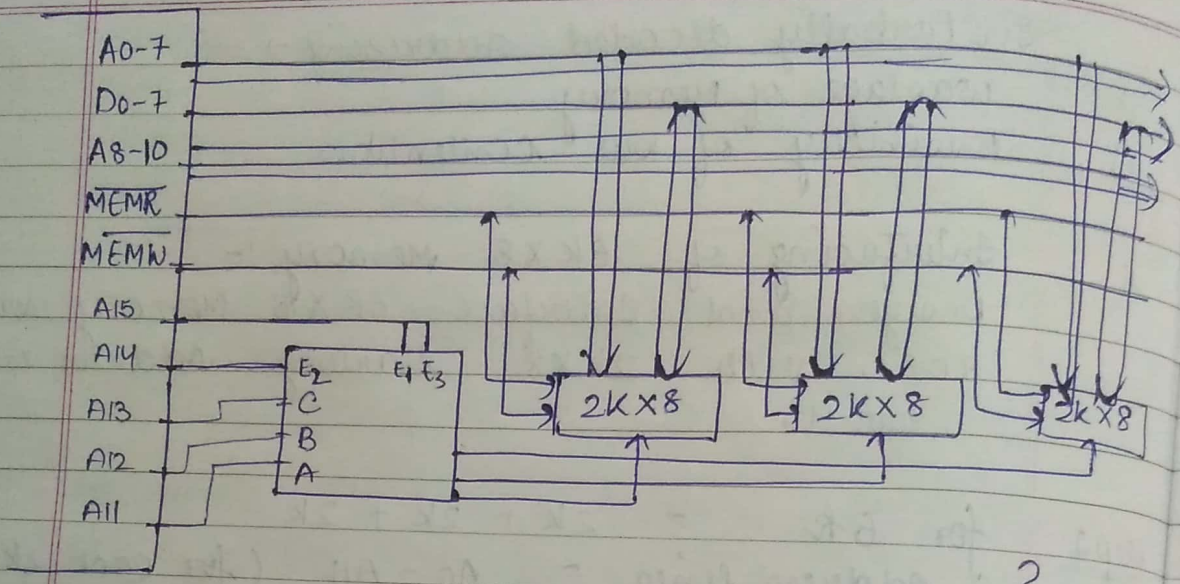
Step 4 Memory Mapping →

A15	A14	A13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1											
1	0	0	1	0											

Memory mapping {
8000 — 87FF] chip 1 Address
8800 — 8FFF] chip 2 Address
9000 — 97FF → chip 3 Address.

Bit Pattern →

A15	14	13	12	11
1	0	0	0	0 → $\overline{CS}$ - 1
1	0	0	0	1 → $\overline{CS}$ - 2
1	0	0	1	0 → $\overline{CS}$ - 3



?