L-1, W-1 24/06/25

gxo5j3d2 MnM Lab

(3118)

μP : 8086

μC : 8051, Arduino

7 classes : Course Content

2 × Presentation

Project

* Proteus : Circuit

Keil :

Class Performance : 20

Continuous Assesment : 10

Project : 70 → 20 : 50

initial submission

VI turns on at 14.74 V

15 MHz

KIC → Keyboard IC

AD = Address

DA = Data

+, − = manipulates address (inc, dec)

RESET = self explanatory → processing once will not delete values

→ holding will del all values

STP = Step

GO = Execute at once

MON = Non Maskable interrupt → highest priority

can be halted

cannot be halted

: = Separating offset and address

2 RAM: U9, U10 → 32 kb × 2 = 64

EEPRO: U7, U8 → 32 kb × 2 = 64

Address

RAM ~~starts~~ : 00000 → 0FFFF

ROM " : 10000 → FFFF

D/A = Digital to Analog

A/D = Analog to Digital

INT 3: Used for breakpoints

H.W: All 10 codes in pdf

C-2, W-1 MNM Theory 24/06/25

*Historical Background:

→ Vacuum Tube

↓

Transistor (1947) Solid State: No movement externally
Electron flows internal

↓

IC (1959)

↓

μP → 4004 - 4 bits
4040
8008 - 8 bits
8080
8085
8086 - 16 bits, 1 mb
8088
80286 - 16 bits, 16 MB
80386 - 32 bits, 4 GM
80486
pentium (P5)
RISC
Itanium (P7)

16/8

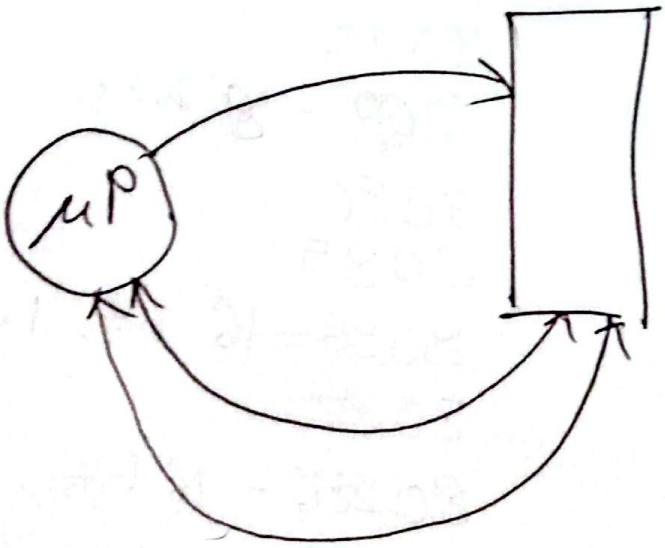
*Basics of Computer Architecture:

8 bits system contain 1 MBs $(0 \rightarrow 2^{20})$

∴ Address bus will be 20 bits (unidir)

∴ Data " " " 8 " (bidir)

∴ Control " " " 1 (bidir)

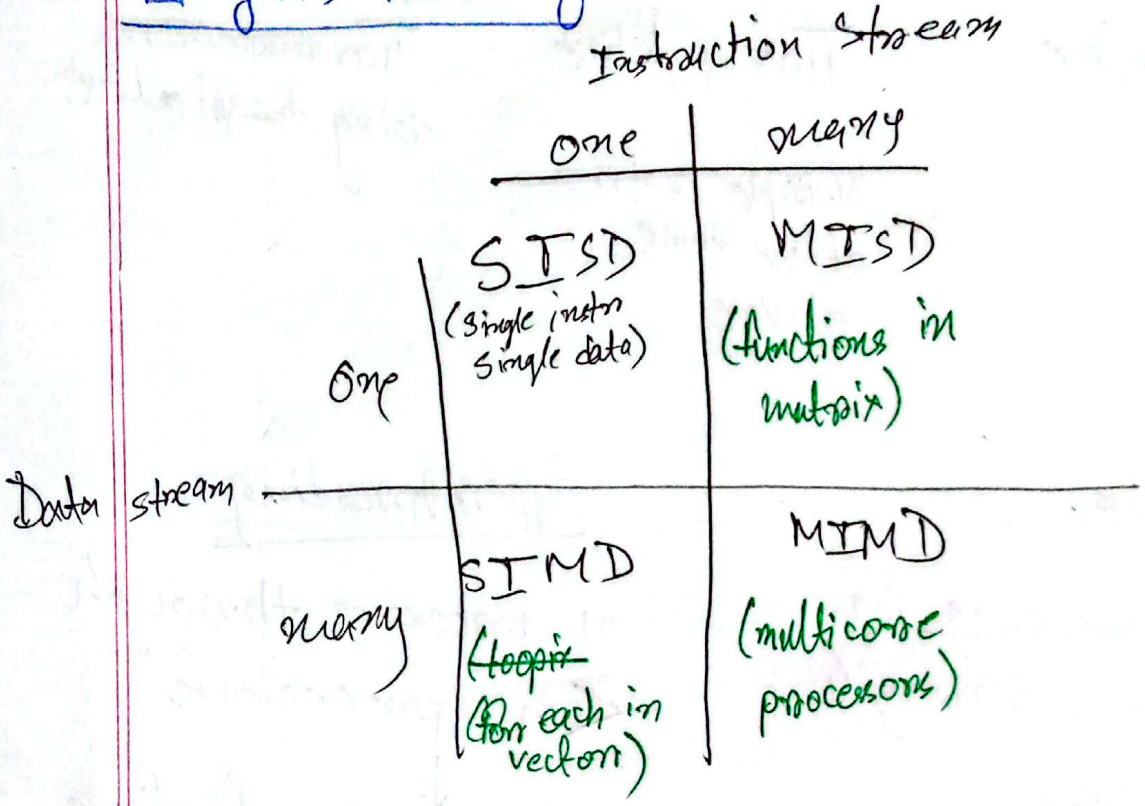


For n bits data bus ⟶ $\frac{2^n}{8}$ slots

$2^n - 1$ addresses

$2^n - 1 \div \frac{2^n}{8}$ (?) size

Flynn's Taxonomy:



Scalar Processors → Single data → loops

Vector " → All Single instr → does all at once

Pipeline

One pipeline

Super Pipeline

Two pipeline

↓

Multiple instr at the same stage

Super Scaler

Two processors doing the pipeline

Multicore *

~~Multiple processes in~~

n processes through

n processors

Hyperthreading

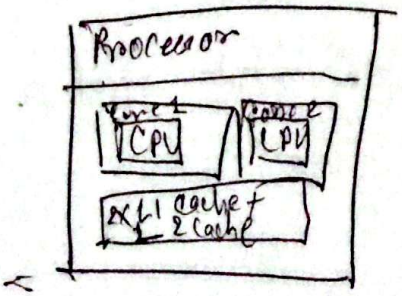
n processes through

< n processors

↓

enables other functions if any function is not fully working

(virtual processing)



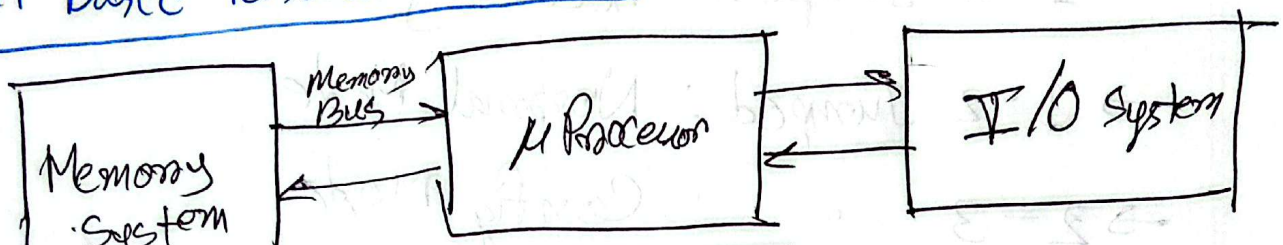
*Linker merges everything to exe → happens in hard disk

↳ runs in RAM

C-4, W-2 MnM Theory 01/07/25

x86 Based PC

µP Based Personal Computer:



DRAM
Cache

8086
8088
XT Machine

80286
80386
80486
} AT Machine

Pentium } ATX Machine

* MotherBoard: Contains: i) CPU ii) DIMM (Memory) Socket iii) AGP slot (GPU) iv) ATA Cable (with hard disk v) South Bridge vi) North Bridge vii) PCI slot (for I/O)

→ Some I/O devices are connected using cards

* Some jumpers are also in Motherboard which configure are for connections

→ 1-2 Open : Recovery Mode

→ 1-2 Jumped : Normal Mode

→ 2-3 " : Config Mode

* Jumpers were more imp then when Plug and Play wasn't around

*Chipset: Group of ICs working together to control flow of data from/on to and within the motherboard

→ They include a north bridge chip and south bridge

→ A chipset contains two microchips:

North Bridge ——— South Bridge (I/O controllers)

North Bridge
- → CPU
- → RAM
- → Video Card

(memory communication)

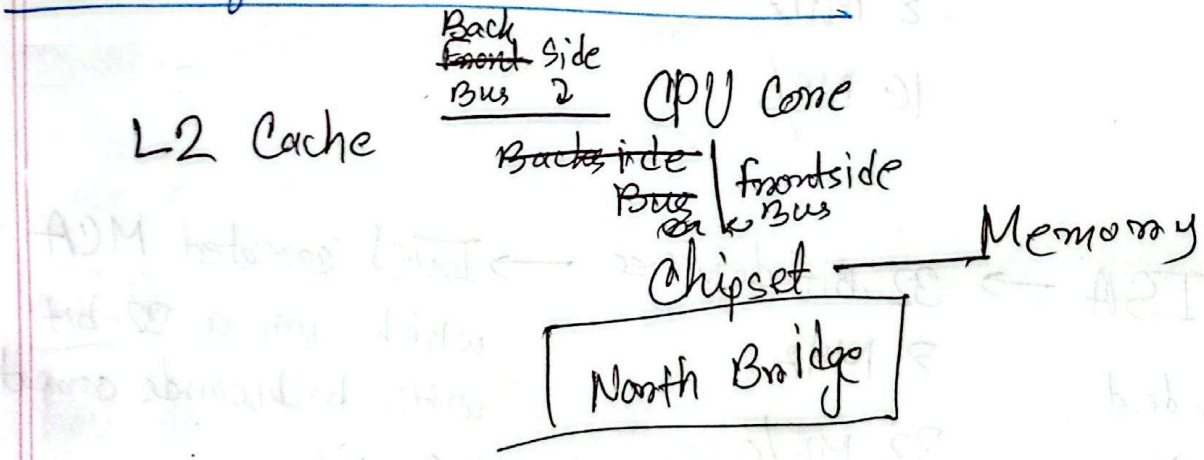
South Bridge
- → PCI
- → USB
- → ATA
- Low Pin Count Bus

(I/O device communication)

* Intel then made its own chipsets:

→ Graphics and Memory Controller Hub (GMCH)
↓
North Bridge
South Bridge
↑
→ I/O Controller Hub (ICH)

* Bus signal Transfer Carriers:



* AGP (Accelerated (Advanced) Graphics Port):

~~* Contains extra on~~

* Video cards have extra onboard RAM, a specialized graphics processor
↳ Useless unless high-speed CPU

Something Something Check slide

* Expansion Bus:

I/O buses through which I/O devices are connected to the South Bridge or I@H

* ISA (Industry Standards Architecture)

ISA → 16-bit devices
8 MHz
16 MB/s

EISA → 32 bit devices → Intel created MCA which was a 32-bit with backwards compatibility
8 MHz
33 MB/s
16-bit

(Extended ISA)

VESA → 64 bit μP

*PCI: 133 MBps

i) Supports 3.3 V and 5 V

ii) Processor Independent

iii) Plug and Play

iv) Allows 'Bus Mastering' (Bus controller is in North Bridge)

→ But wastes space since ~~one~~ same lane size

∴ we use ——

*PCI Express:

has different sizes.

→ uses serial communication which sends data faster

C-5, W-3 MnM Theory 08/07/25

* Whats not in slides:

*Multiple bus interference (?) is called Contention
↳ solved using daisy chain, prioritized - -

*Skew (problem): in parallel, ~~multiple~~ there could be an issue with time. ~~ext~~

*Serial Communication loses time but fixes skewness and pin issues

*PCIe: (third gen) - 250 MBps
- Duplex wire connection is called lane
- PCIe fixes the time issue by multiple lanes

*USB: (third gen)
- Contains twisted pair cable to prevent interference
- Power is obtained ~~one of~~ from one of the cables
  - ~~Dev~~ Cannot provide more than 5V and thus those devices require external power supply

~~N~~

- NRZI → changes at 1

*ATA/SATA:

- Used for hard disk
- IDE: Integrated Drive Electronics
  - Devices that contain its own controllers and RAM
- Controllers are usually in bridges

*SIMM and DIMM:

*System BIOS:

- Basically ROM (?)
- Softwares used by OS to communicate with I/O embedded in the ROMs
- Make it editable and customizable using EEPROM

C-6, W-3 09/07/25

Whats not in slide:

Memory Map of PC: (RAM

- What is kept, why is kept and how is kept in RAM
- First 1 MB is called real (conventional) memory
  - ↳ Usually before Windows 98
  - ↳ 20^2 2^{20} bytes = 20 pin address
    - ↳ Addressing these is called real mode addressing
- Two parts: i) Transient Programmable Area (TPA)
  - ↳ 640 KB
  - ↳ Temporary
  - ii) System Area
    - ↳ 384 KB
- 80286 contains 15 MBs so Extented Memory is 14 MB
- Any addressing mode in extended memory is protected memory addressing

TPA: 0x00000 to 0x9FFFF

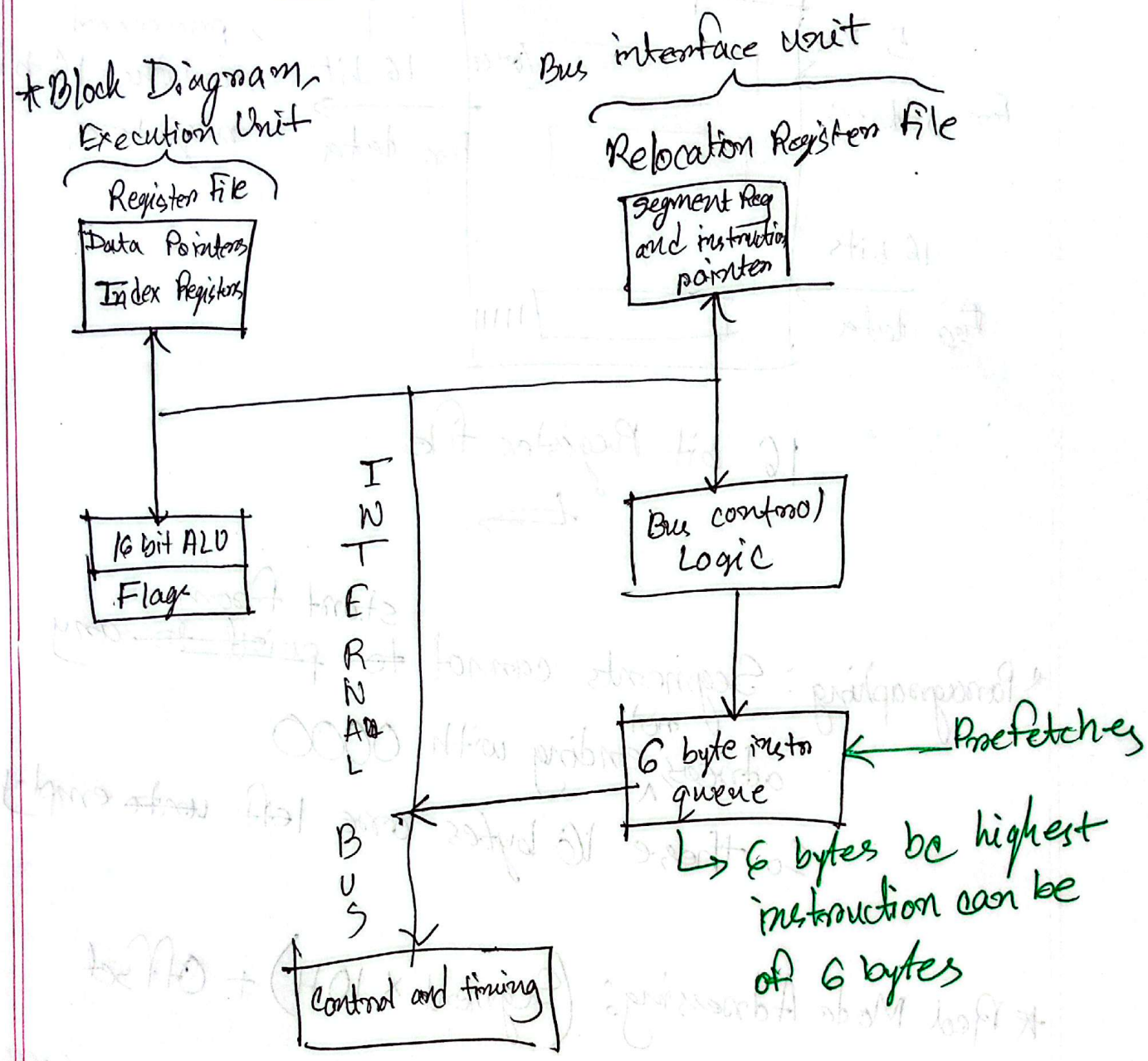
- DOS parts, TSR (terminate and stay resident) etc remain here
- Interrupt Vector Table stays in first 1 kb
  - ↳ Num to pick what kind of Interrupt called 'Interrupt Number'
  - ↳ 256 interrupts (0 – 3FFH)
  - ↳ IVT contains address in ISR
- Next 1kb contains BIOS communication
- " " " DOS "

System Area: 0xA0000 to 0xFFFFF

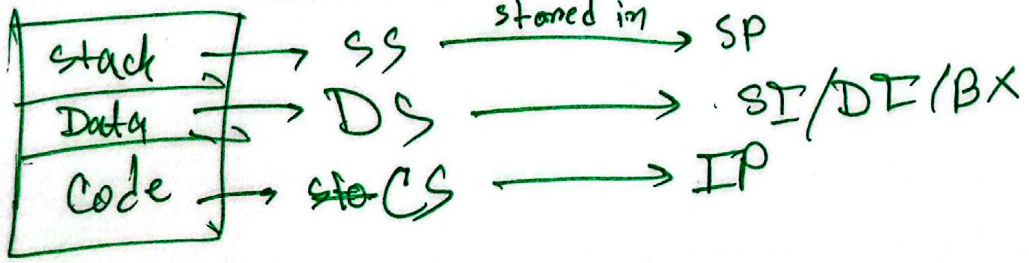
- Video RAM (graphics) is in A0000 to 80000
- " (text) " upto C0000
- If system Boot used to fail, a BASIC language visued would be used using cassettes which contained BASIC compiler (only on early computers)
- BIOS system ROM contained Booting Program and I/O control Program.

C-6, W-3 09/07/25

Architecture of x86



* ~~Linkers~~ Any program is broken into
Addresses are stored in the processor's registers



C-7, W-4 15/07/25

* Flag Register:

* Trap flag (8): 1 = Debugging mode
* Interrupt Enable (9): 1 = Send interrupt signal
  - → Maskable: Processor can stop
  - → Non-Maskable: " cannot stop and executes immediately (NMIP)
* Direction Flag (10): 1 = Decrement (of index); [R to L]
  - ↓
  - SI, BI --
* Overflow: If the result is correct (+ve and -ve errors)
  - ⓞ O = C_m to MSB ⊕ Carr from MSB
* IOP: IO Privilege: Priority of Accessibility
  - → 2 bits
* NT: Nested Task: Child task starts ~~before~~
* RF: Resume Flag: Opposite to Trap Flag

* VM: Virtual ~~Flag~~ Mode: Multiple BIOS

* AC: Alignment Check: 1 = Shifts based on alignment

* VIF: Also Virtual Interrupt(?) → Interrupt Flag (I)

* VIP: Interrupt during virtual mode

* ID: CPU Identification: 1 = Executable

Segment Register

* CS = Code Segment
* DS = Data Segment
* ES = Extra Segment
* SS = Stack Segment
* FS & GS = Supplementary segment

Real Mode Adressing:

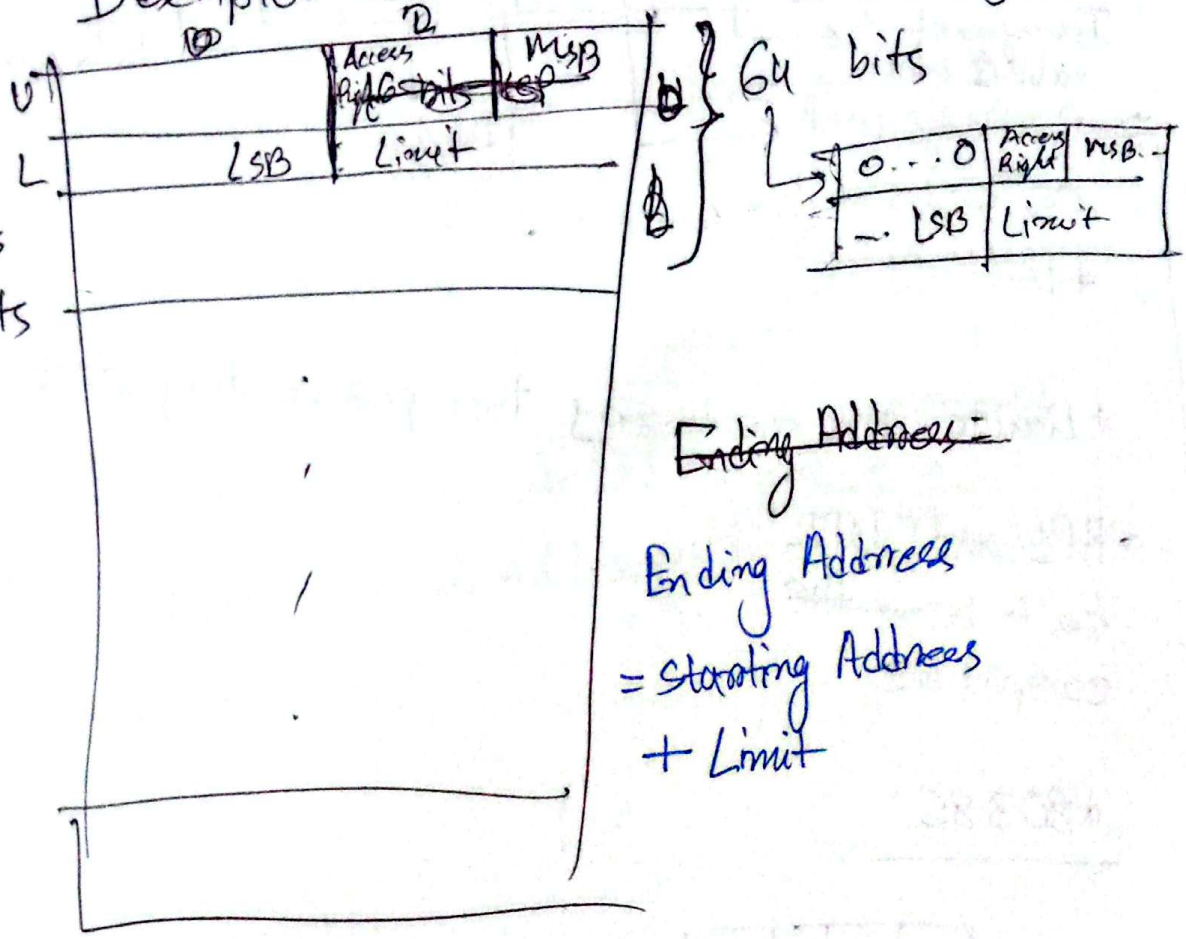
* Real mode cannot access more than 1 MB
* Paragraphing memory space
* No Protection System

Protected Mode Adressing

* Accesing after 1 MB
* No ~~more~~ paragraphing
* Starting Addresses are kept in Descriptor Table

80286 (24 bit memory bit)

Descriptor Table

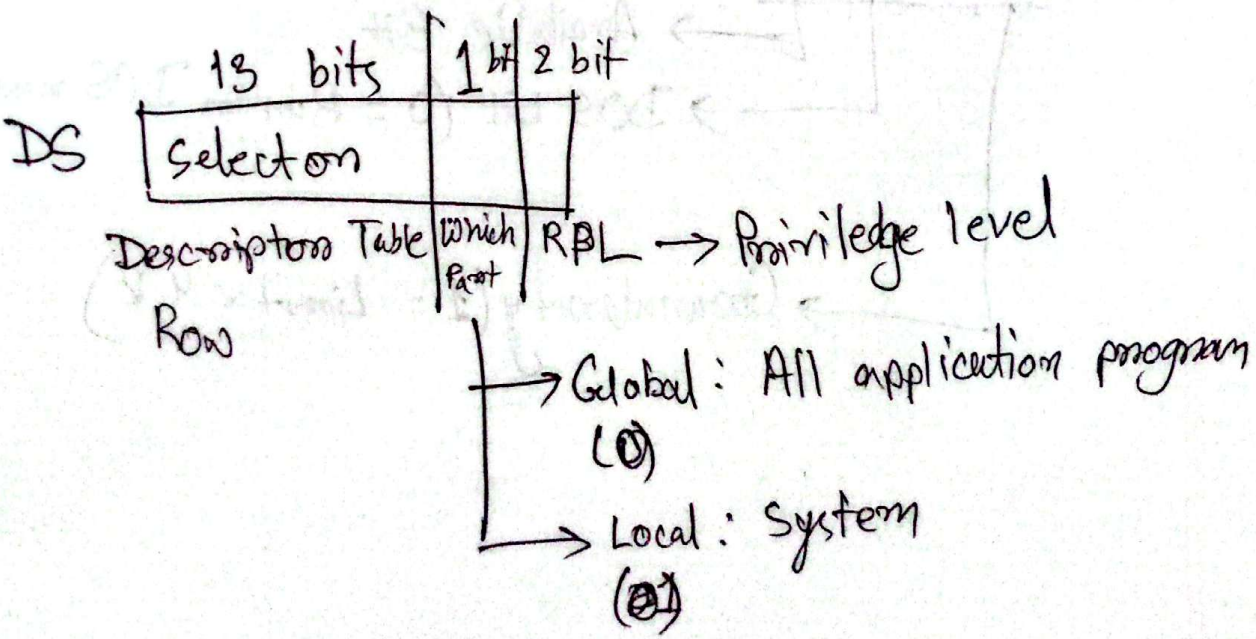


Limit = 16 bits

Starting Address = 24 bits

~~Ending Address =~~

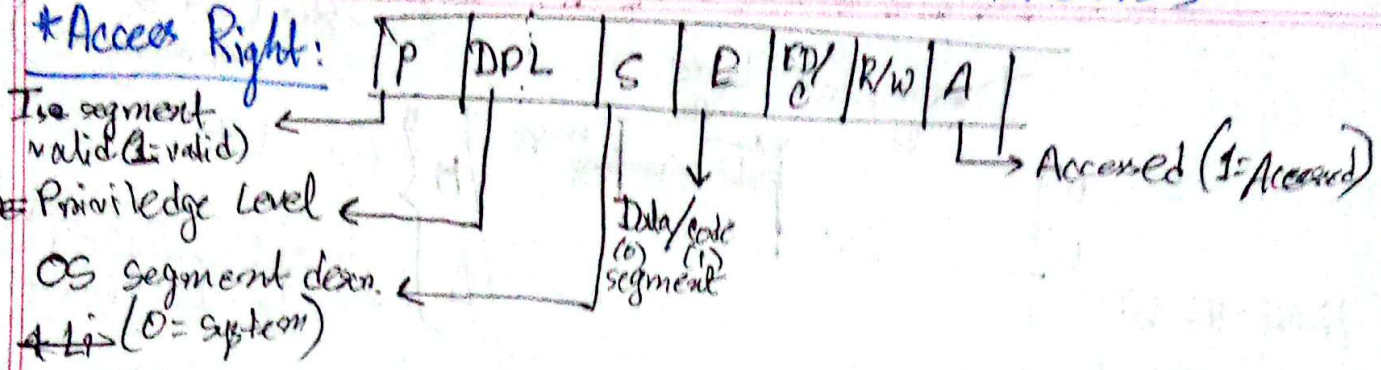
Ending Address
= Starting Address
+ Limit



C-8, W-4

16/07/25

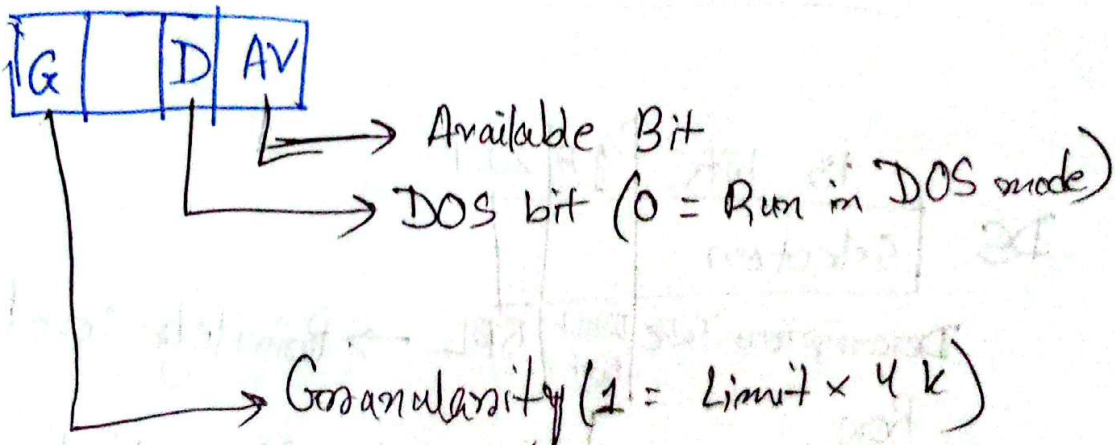
*Access Right:



* Limits are necessary for preventing stack overflow

RPL and DPL has to be ~~on the~~ compatible

*80386:



* GDTR = Global Descriptor Table Register
  ↳ stores address Base Address of DT
  ↳ while accessing, always 8 * Index
    ↳ 64 bit GDT is the size of every DT part
~~DS~~ addresses and limit
* Local DT stays inside GDT
. LDTR stores the pt address in GDT where LDT is stored

C-9, w-4 17/07/25

Addressing Mode of ~~8086~~ 8086

* Instructions converted to Binary is ISA

MOV AX, 23H

Destination

Always an operand will be register

* Addressing mode is to find the value and location of any variable

* 8086 is CISC → Instructions are different sized
  → highest is 6 byte

Addressing

Data Addressing (where the address is)

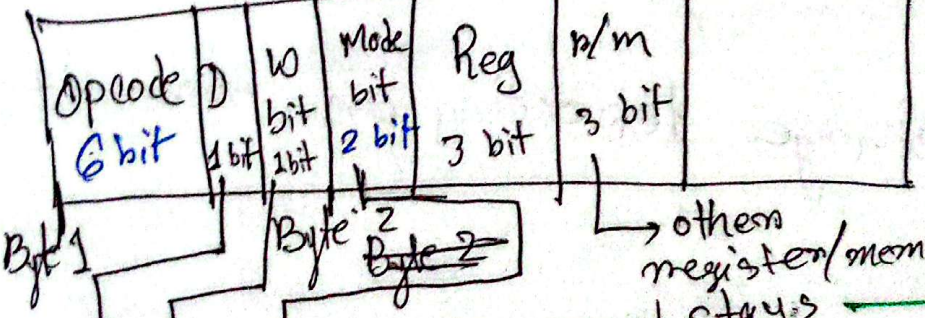
Memory Addressing

everything else

Register Addressing

Immediate Addressing

6 byte



MOV AX, BX

D = 1

w = 1

Reg = 000

mod = 11

r/m = 011

→ others register/memory location

→ When operand stays

00 } No displ

01 } Memory location

10 }

11 } Register

→ Decides if 8 or 16 bit
(0) (1)

→ Direction bit
↳ decides if source or dest of Reg
(0) (1)

To find memory location,

[. .]

Mode Bit,

00 ⟶ No displacement [BX]

01 ⟶ 8 Bit [BP + 10H]

10 ⟶ 16 bit [BP SI + 1000H]

* R/M changes depending on mode bit

Data Addressing Mode

* For Reg MOV Reg, Reg

100 010 ⟶ op code (first 6 bits)

~~MOV~~ # MOV BX, CX

	Op code	D	W	mod	Reg	R/M
	100010	1 (BX is dest)	1 (if bit ~~BX is dest~~)	11 (Register)	011 BX	001 CX
&	100010	0 (CX is src)	1	11	001 CX	011 BX

Immediate Addressing : Data is in instr.

* Moves in Little Endian

* For Move Reg, Imm

1011 wrrrr ← opcode

Mov AX, 5001 H

~~0010111 0000 0001~~

Reg 000

001011 1 0000 0001 0101 0000

opcode w 01 50

C-10, W-15 20/07/25

[1] Memory Addressing:

* Direct and Displacement:
  - Operand is the address of (from DS) internal data memory
  - Direct Mode ^ segment but for any segment

→ Move Mem, Reg / Move Reg, Mem

100010 dW mm reg mm disp

#MOV BL, [1234H]

100010 1 0 00 011 110 0011 0100 0001 0010

opcode | reg is dest | 8 bit | Mode | BL | R/M | 34 | 12

#MOV Ax, [1234H] → for Accumulator its diff

1010000 1 1 0011 0100 0001 0010

opcode | reg is dest | 16 bit | 34 | 12

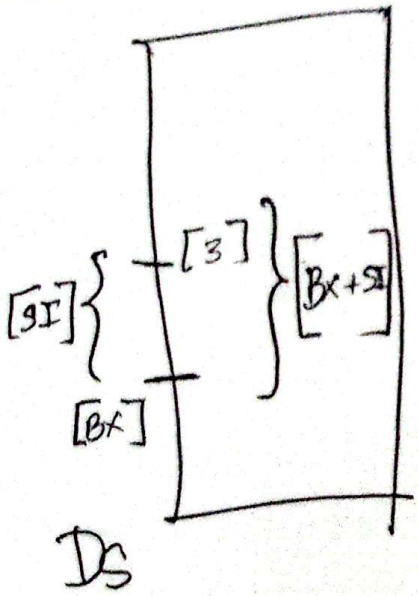
*Register Indirect : Value inside of a register is the register

MOV AX, [BX]

1000 10 1 1 000 111

opcode

*Base Plus Index Addressing:



*Register Relative Addressing: [check slide]

MOV DX, [BX + 08H]

100010 11 01 010 111 ~~010~~ 0000 1000

C-011, 10-5 MnM Theory 22/07/25

[4] Register Relative Addressing:

MOV AX, [Bx + 1024 H]
↳ D8/D16

100010 d w 00 rrr mmm disp

#MOV DX, [BX+08H]

100010 1 1 010 111 0000 1000
op code | word DX BX 08H

DX
is dest

[5] Base Relative Plus Index:

#MOV DX, [BX + DI + 3H]

100010 1 1 01 010 001 0000 0011

* In FLAT mode, ~~reg~~ value in register is the address
* In 64 bit, the processor needs addressing pins which cannot exceed 2^{40}

Hardware Structure of x86

↓

chip based

[Pin from slide]

* If pin on both sides → we say DIP
* ~~16 bit~~ follows 16 bit processing

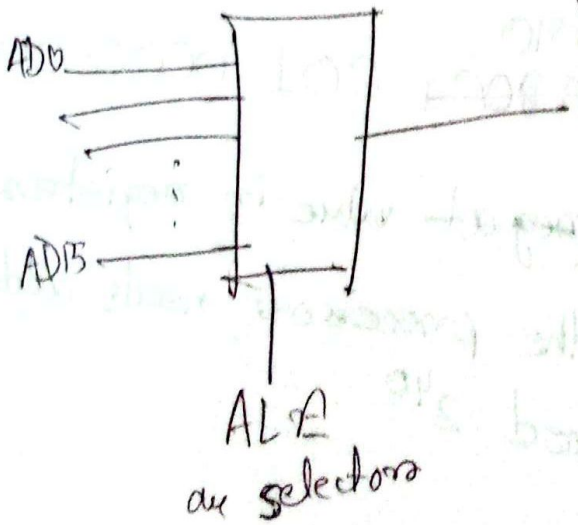
* AD n = Address Data → 0 to 15

We do multiplexing with ALE

↳ Address Latch Enable

↳ 1 = Address

0 = Data



[1] MIN and MAX mode:

* MIN: - for single processor (unique processor)

* MAX: - Used multiple processor
  - Increases more functions
  - less pins required
  - four bus controllers
  - 3 bits generate signed bit generates functions

→ diff between the two: 24 → 31 pins

[2] Power Supply Requirements:

8086 / 8088 → +5 V → 360 mA / 340 mA → 32°F - 18°F

80C86 / 80C88 → +5 V → 13 mA → 400°F - 25°F
- less current requirement
- more survivability

CMOS

Logic level	Inp	Out
0	+0.8 V max	+0.45 V max
1	+2.0 V min	+2.45 V min

} I'll tell you comes later

C-12, W-5 24/07/25

Data Access from Memory:

4 diff ways:

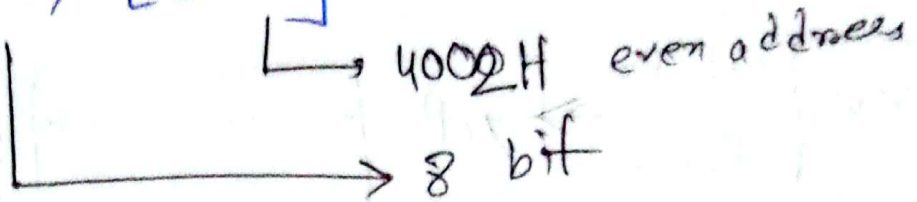
8 bits from Lower (Even) address
8 " " Upper (Odd) "
16 " " Even address
16 " " Odd "

*when taking from even bank, we take from next odd
when taking from odd bank, we take from next even

* A1 – A19 are addresses
A0 ~~are~~ is used to pick which ~~dat~~ bank
* A0 is used ~~for~~ demux for taking value from
even bank | A0 = 0 → take from even (lower)
A0 = 1 → dont " " " "

~~BHE~~ is used as demux for taking from odd bank

# MOV SI, 4000H
MOV AL, [SI+2]



∴ A0 = 0, BHE = 1

# MOV SI, 4000H
MOV AL, [SI+3]

A0 = 1, BHE = 0

# MOV SI, 4000H
MOV AX, [SI+2]

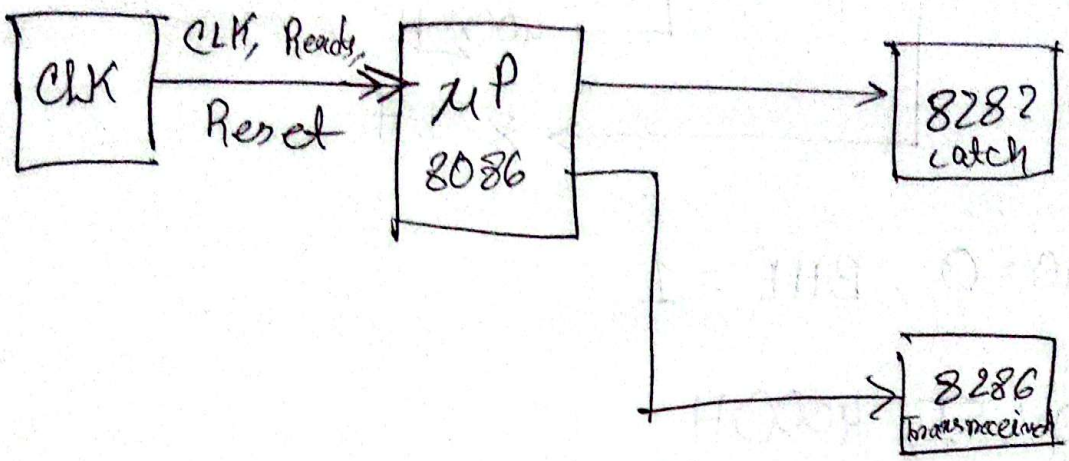
A0 = 0, BHE = 0

MOV AX, [SI+5] → we cannot do it at once

1: A0 = 1, BHE = 0 → put at AL → increment

2. A0 = 0, BHE = 1 → put at AH

ALE:



ALE → selects Address / Data

- → 1 → $\overline{DEN}$ = 1 → Disables 8286
- → 0 → $\overline{DEN}$ = 0 → Enables 8286
  - → DT/$\overline{R}$ = 1 → ~~Rec~~ Send
  - → DT/$\overline{R}$ = 0 → Receive

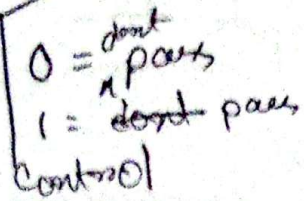
* ~~Buffer~~ Latch → Seq^{n} circuit

Buffer, Transreceiver → Combinational

* Transreceiver is not used in address bus as it is unidirectional

* Diff between bus, latch and transreceiver

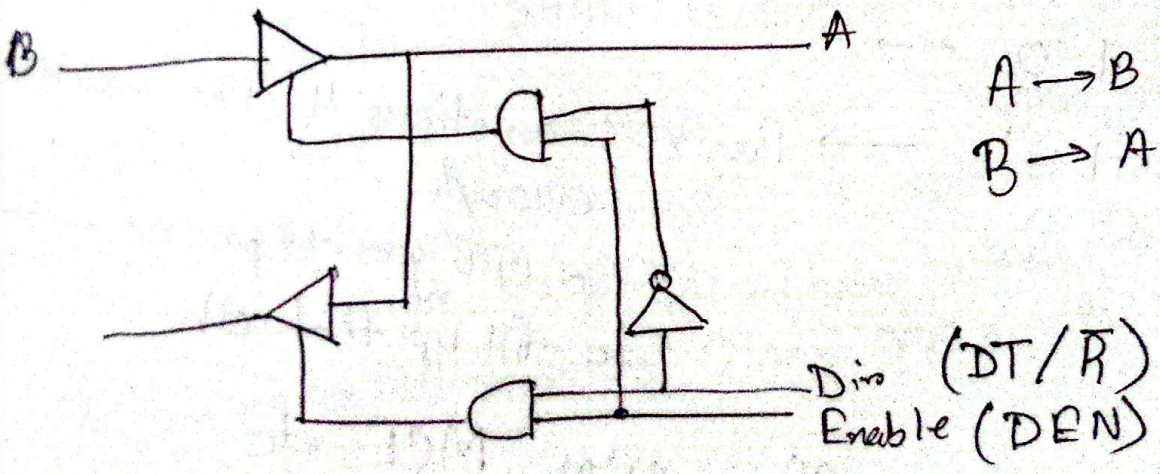
* Input Output



(Tristate) Buffer

* Latches can propagate and store
  - if ALE = 0 ~~Q = D~~ D → Q
  - ALE = 1 ~~Q = D = Q~~ D → D

* Bidirectional Buffers (transreceivers)



MAX Mode:

8288 → Bus controller

takes three inputs to

S_0 S_1 S_2 → status of processor

0 0 0 → Interrupt ack
0 0 1 → I/O Read
0 1 0 → I/O write
0 1 1 → Halt: stop ~~Bus~~ ALU
1 0 0 → opcode fetch
1 0 1 → Memory read
write
1 1 0 → ..
1 1 1 → Passive: Sometimes 4 clks are not enough
So, CPU goes to passive mode
to fill up that clk

AIOWC, AMWC } extra clk is necessary so we start early

→ Also sends AEN, CEN, MCE etc

* MCE: Master cascade / peripheral data (I don't get it?)
for interrupt controllers (sends interrupt pin numbers through data bus)

8284: Clock → Continuous signal

* Activities: i) Generates clock (clk), peripheral clock (pclk)
↓
for µp

ii) RESET

iii) Wait State / Ready signal

* Clock freq can be generated by itself (internal) or from another device (external)
↳ has to be synchronized

* Clk is made from crystal oscillator
* CSYNC is used to synchronize → 1 = Reset counter
* RESET: If 1 for 4 Clk cycles, it becomes reset and runs FFFF0H address which then makes all registers 0 and IF as 0

C-14, W-6 29/07/25

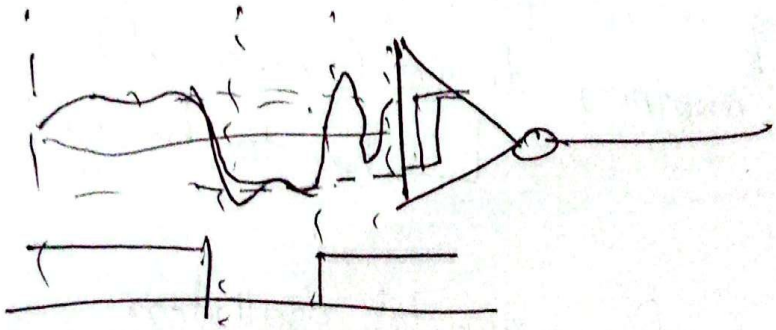
RES : If 0 ⟶ resets to 1

↳ System reset to 0

If 1 ⟶ resets to 0

↳ system resets to 1

* Schmitt triggers converts analog signal to digital



RDY : Decides wait state (?)

* Whenever extra clk is needed for slower devices
* Clk starts from negative edge
  But in μP it starts from positive edge

→ so it has to be synchronized

- if ASYNC = 0, 2 state synch
- " " 1, 1 " "

BUS Cycle:

* Read Cycle:
* Write Cycle:

} check slide

C-15, us-6 30/07/25

Memory Interfacing

* How memory connects with μP
* How addresses are distinguished is called Interfacing
* Memory contains $2^N \times M$ bits
  - N → input lines,
  - M → output lines,
  - for 8086 → $2^{20} \times M$ bits
* RAM has 2 control pins: $\overline{OE}$ = Output Enable
  $\overline{WE}$ = ~~Input~~ Write Enable
  ROM " 1 control pin: R/$\overline{W}$
* $\overline{CS}$ → Chip selector (if address is here)

2716 2K × 8

$= 2^1 \cdot 2^{10} \times 8$

$= 2^{11} \times 8$ bits

↳ output ~~bits~~ pins

↳ input pins

* V_{PP} : Used to program (Requires 25 V to be active)

* PD/PGM: When PD (Power down) (0) → Read

When PGM (1) → Write (Requires V_{PP} as active)

TMS4016 2K × 8

* S → CS

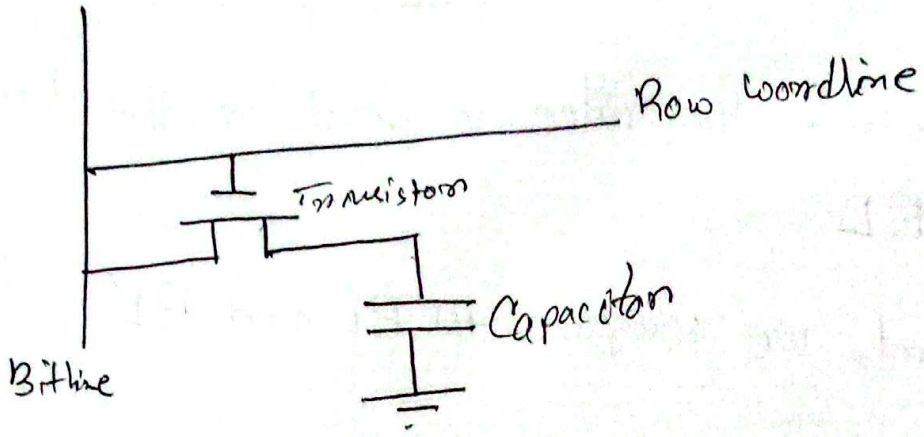
G → OE

W → WE

SRAM and DRAM:

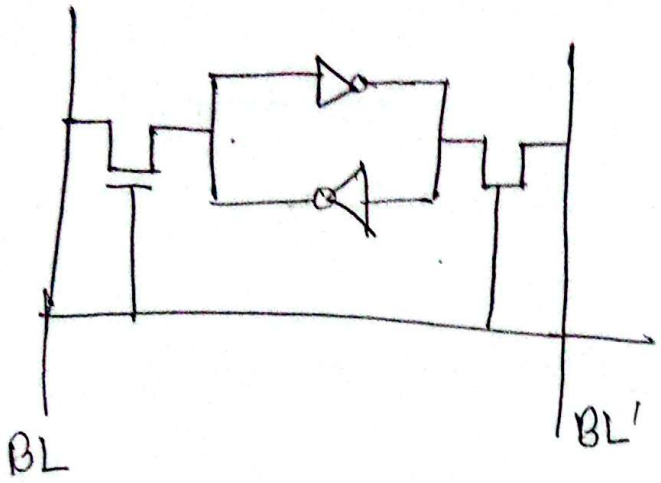
- SRAM: Does not require refresh
- DRAM: Requires Refresh

R/W with DRAM:



* Sends data using Bit line to store value in capacitor
* Retrieves data from capacitor using Bitline for reading
  ↳ But this is destructive as any value read empties this cell
* Capacitors leaks charges
  ∴ after a while 1 may become 0
  which is why we need to refresh.

SRAM:



* To store, the value is sent in the transistors using BL
* To read, we compare ~~to~~ BL and BL'

4) Address Decoder:

* 20 bit addressing
  so, we take ~~11 bit~~ 2716 EPROM
  with 11 bits
  The ~~first~~ 9 bits are ~~taken from~~ managed using
  a NAND decoder
  ↳ to change this use NOT

C-16, W-7 03/07/25

3 to 8 Line Decoder (74138)

↳ Inputting the line will send 0 on that line instead of 1

design with from slide

Dual 2 to 4 Line Decoder

Simple memory System with 74139

PLD: ~~Since~~

* PLA: Since every circuit can be expressed as SOP, we tak k AND and m OR