Microprocessor & Interfacing

15-2-24

Lec-1

→ Hall , DOGLUS V Hall

→ Marut

* fundamental processor : 8086

Processor → First → 4 bit

↳ Present → 64 bit

↳ একই সময়ে 64 bit ডাটা process করতে পারে।

Ch-0

25-2-24

Lec-2

Computer Number System & digital device

⇒ Hexa

⇒ BCD

⇒ table [ASCII code]

Capital [Hexa, decimal] [P5]

world

small

* Binary যোগ, বিয়োগ, multiplication [Lab]

* Microprocessor → brain/heart → Programming device

Micro + Processor

↳ Process করার ক্ষমতা [input নেওয়া ও feedback এর মাধ্যমে কার্যপ্রণালী]

8086 → Address bus এত তত memory কে access করা যায়।

bit সংখ্যা 5 হলে তা হবে 32 bit এর memory

64 bit - Operating System 64 bit এর ফিচার হবে।

8086 → 40 টা
→ ছোট ছোট IC থাকে → তাদের core বলে।
→ যত বেশি core, তত বেশি fast হবে।
→ core certain Level পর্যন্ত ফিচার হয়।

[core - i5, i7, i9] → online থেকে কিনতে

ch - 1

1.6

Vonn Neumann → Architecture] self-study
Harvard → "]

1.2

RISC and CISC [diff]

1.10

Defination - theory [year-wise]

Microprocessor: → Micro and Processor

, Micro computer & Microcontroller diff.

↳ I/O, memory

↳ Processor for specific কাজে ব্যবহৃত

↓ general কাজে

Processor → I/O, storage, Logical operation

↳ Register থাকে memory

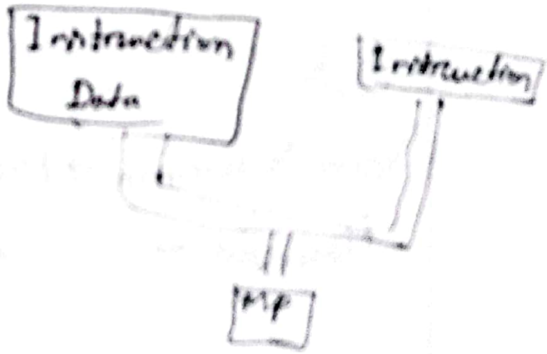
RAM → Pramary, ↳ R-W ↳ bi-directional

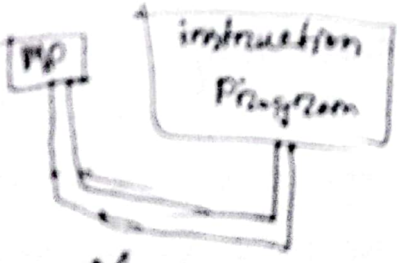
ROM → Secondary ↳ Read only ↳ unidirectional

Direction safety for

Architecture of Computers:

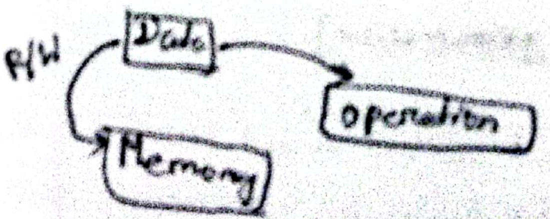
1. Von Neumann → single specific task
2. Harvard → 2 টা Memory





Von Neumann

Data Bus:



8086
↳ 16 bit

Bus: Data travel করে

1. Data Bus → Bidirectional, memory to processor or processor to memory

2. Address Bus → Unidirectional, processor to memory → $2^{20} = 1024 = 1MB$

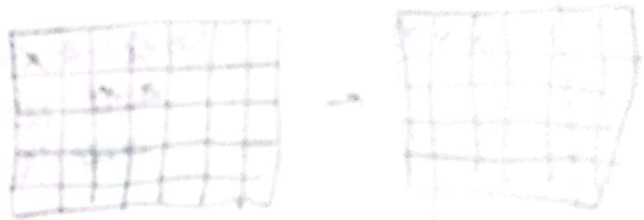
3. Control → → " " , R এবং W এর operation, করে bit
R হলে এবং W হলে না

↳ একেবারে দরকার,
bidirectional

16 bit / 20 bit ← 8086 MicroProcessor

↳ Data bus ↓ Address bus

Disk defragment : এলোমেলো বা unordered অবস্থা কে order করে memory তে।



* Memory size কতটুকু তা Address Bus এর উপর depend করে

Bus system : Figure আঁকার পরে দেখবো

* 8086 Features:

1. 16 bit MP → Data bus 16 bit
2. 16 bit word size
3. 20 bit address bus

* Hexa ও last এ H থাকতে হবে

5 bit এর Address
↳ 00000H থেকে FFFFFH

4. Memory byte addressable

↳ 8bit R or W

5. 64k I/O ports ← support

6. 40 pin dual line package এর Leg

7. AD0 - AD15

A16 - A19

A0 - A19 = 20

D0 - D15 = 16

→ A = Address
D = Data

16 + 4 = 20

8. 2 mode → Minimum, Maximum

↳ একটা Processor এর কাজ করে

↳ co-processor: একাধিক processor এর কাজ করে,

* Distance বেশি হলে Current বেশি লাগে

9. 6 byte instruction একসাথে কাজ করে.

Intel 8086 Internal Architecture:

1. Bus Interface Unit → Data R.W. → communicating করার { asynchronous, Pipelining

2. Execution Unit → Execute করে AL করা হয়
(Arithmetic Logic)

Σ → Adder

ES, CS, SS, DS, IP → Register
↓ Data Segment ↳ Instruction pointer

6 byte Queue

AX = AH + AL → Register System
16 8 8

H MSB 0000 0000 L LSB 1101 1111

General Purpose Register

SI → Read
DI → Write
and SP, IP → Pointer and Index Register

OPERAND, FLAG → Error control
(এরর Register এ স্টোর operation হয়)

* Instruction Queue → Control system → ALU → Register.
↓ fetch queue এ decode → EU

EU →

1. Instruction decoder: Machine language ↔ Convert করে.
2. Control Unit

SI → কোথায় Data আছে
DI → Memory তে Data Write

SP → Immediate কোনটা run/execute হবে তা নির্ধারণ করে
DP → " " কোথায় Data নিবে

* Accumulator Register: Basic I/O operation, Arithmatic Operation
+, -, *, /

* Machine Language → Assembly → Lab ১ → AX, BX, CX, DX

BX → Addressing purpose ১ → Base Register

CX → Count Register → loop, shift, rotate

DX → Data " → Data নিয়ন্ত্রণ করে,

↳ Multiplication কিছু bit রেখে হবে, Division ১ remainder
আলাদা Addition এ carry ↘ DH ১ Addition হলে আলাদা bit

* EMU → 8086 → online কোনো
↳ Lab

* Address bus bit size 16/32 bit হলে memory কত?

* Flag → operation এর status justify করে
→ controlling signal
→ Flip-flop ⇒ bit store করে

1 → Set
0 → Reset

2 types:

1. Conditional → বিভিন্ন register এর status দেখবে
2. Control → " operation কে Control করে

16 bit
↳ 9 active → fixed → 6 টা contional, 3 টা control
↳ 7 → undefined

CF → Extra bit → 1101 + 1101 = 11010 → Carry bit → CF = 1

P → " " → Error detection check করার জন্য
↳ 1 or multiple bit হতে পারে
↳ 2 ধরনের → 1. even, 2. Odd → এটা use করব, [কিছু না বললে odd নিয়ে কাজ করব] → xm
1 কে odd এ নিতে হবে
↳ 1 কে even এ নিতে হবে

1101 → EP = 1
1111 → EP = 0

1111 :
OP → 0
OP → 1

1010 → PF = 1
1110 → PF = 0

AF=0

1111 0000
0000 1111

Auxiliary Flag:

↳ Nible থেকে carry আসছে কি না

↳ 1st 4 bit → LSB

↳ (0-3) position

↳ position 3 থেকে 4 এ কোনো carry আসে ↳ AF = 1

1111 1000
0000 1111

ZF:

↳ Result যদি 0 হয় তাহলে ZF=1 ; result 0 না হলে ZF=0

→ যত bit এর data তত bit এর calculation করবে, Carry বাদ

Sign:

↳ +ve নাকি -ve

Control Flags:

Trap:

↳ Logical Error এর জন্য debugging করা হয়

↳ line by line execute

Interrupt:

↳ প্রত্যেকটা device interrupt করে

↳ Processor এর যাবে Interruption

Direction:

↳ Left to right

নাকি Right to Left

Over flow:

↳ extra bit generate হলে সেটা register এর size এর থেকে বেশি bit হলে capacity

ZF:

* 1 1 1 1 1 1 1 1 → 8 bit Data

+ 1

---

1 0 0 0 0 0 0 0 0 → 8 bit result

↑ Carry বিট হয়

↳ result = 0

ZF = 1

Sign Flag:

MSB LSB

1 1 1 1 1 1 1 1

↑ MSB ↑ LSB

↳ SF = 1

→ SF = 0

1 → 1 → -ve value

0 → +ve "

MSB হয় Sign Flag তে

Interrupt Flag (IF):

INTR → Interrupt Request

INTA →

Overflow:

signed → - to +

Unsigned →

28 → -127 to 128

+ - হলে + বা - ও হতে পারে

Valid operation এ OF = 0

Invalid → OF = 1

↳ + + = -ve → OF = 1

Carefully → 0455 x m 3

ADD AL.

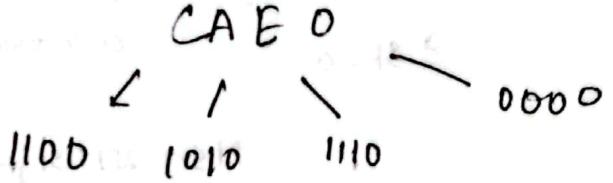
↳ Accumulator Lower

10 = A
11 → B
12 = C
13 = D
14 = E
15 = F

```
  34F5
+ 95EB
------
  CAE0
```

```
  15
  14
----
  29
  15 → F
----
  14 → E
```

8421
3 = 0011 → +ve
9 = 1001 → -ve
C = 1100 → -ve



0011 0100 1111 0101

1100 1010 1110 0000

AL = CAE0h

CF = 0, → No carry

PF = 0 → 7 & 1 → CAE0 → odd number of ones

AF = 01 → Nibble → No carry

ZF = 0 → result not zero

SF = 1 → MSB = 1 or 15 bit is 1

OF = 0 → valid

↳ 3 = 0011 → +ve
9 = 1001 → -ve
C = 1100 → -ve → valid

3bit → 4bit convert
6bit → 8bit convert

*** Binary Subtraction

```
  1
 101
 110
----
 101
```

8421
1
10
100

BIU:

- ↳ 6 byte Queue → একটার পর একটা থাকলে memory থেকে নতুন Ins. নেয় না
  - কমপক্ষে 2 " "
- ↳ Segment register → ES, CS, SS, DS
  - ↳ Starting address স্টোর থাকে
- ↳ IP
- ↳ Summing block
  - ↳ 5 bit but 4 bit Add হয়
  - LSB বাদ যাবে

Maximum 6 byte → Queue ← 8086 এর জন্য

* Pipeling এর Advantages

* 6 byte কেন Long হয়?

* Memory Segmentation:
  - ↳ 1MB Memory কে 4টা segment এ ভাগ করা হয়
  - ↳ 512 use, 512 রাখে
  - ↳ Not Fix নাই
  - Harvard এ সুবিধা
  - ↳ Instruction Data আলাদা
  - Fetch করা হয়

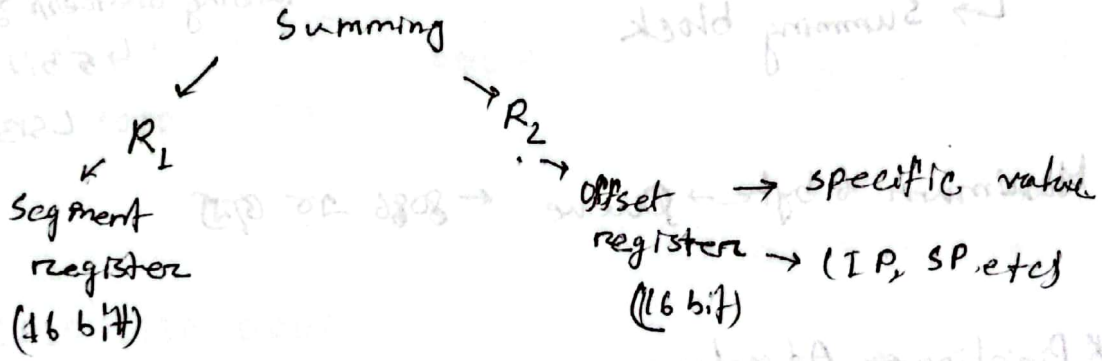
সর্বোচ্চ 64KB নিবে:
- ES → Extra segment
- CS → Code "
- SS → Stack segment
  - ↳ temporary value এর জন্য
- DS → Data থাকবে

*Adv. Segment Memory

↳ 1. security

4. Summing block:

Data = 16 bit
Address = 20 bit
⤵ conversion
২টি Summing block ↴



*left shift → right এ বামে 0 যোগ হবে

↳ ~~Offset~~ Segment রে · -> তারপর Segment → 20 bit

Segment 20 bit + offset 16 bit = Address 20 bit

↳ Effective Add.

CS + IP
DS + BX/DI/SI
SS + SP/BP
ES + BX/DI/SI
] Fixed

5-3-24
Lab-1

Lab

4 টা segment

* 1. model → Directives → small, Large, extralarge
2. stack → কিছু temp. value → 100h
3. data → variable declare
4. code → means body → instruction থাকবে

↳ end main → কোড শেষ করবে
↳ return 0 এর মতো

.model small
.stack 100h
] fixed

main procedure → main proc ← must থাকবে

এর .proc
এর endp
] New procedure →

int → interrupt → call করবে

* int 21h means?

DOS OS → Machine → ...

Disk Operating System

mov ah, 4ch
int 21h
] → DOS কোড Processor এ চালায়
Processor → DOS ...

↳ code terminate করার

* case sensetive হয়
↳ যেমন small এর বদলে capital দিলে

Statement; syntax:

Level : operation operand; comment
(optional)

Level : → function/এক বা একাধিক লাইন → Block of code এক করার block

L1 : MOV Ah, 2
.....
L2 : ADD AL, BL

ALU
↳ +, -, or, not, and, increment, decrement operation

↳ MOV → copy করে দিয়ে যায়
ADD → যোগ
SUB
MUL
DIV
INC
DEC

→ Processor এ Memory থেকে কম → register

Ax → Primary Arithematic Operation

Bx → Base → Address নিয়ে

Cx →

Dx → Normally output এর জন্য

* যেকোনো register নিয়ে যেকোনো কাজে করানো যাবে,

Ax → 16 bit

8 bit → AL or AH

AH কিছু Fix value নিয়ে কাজ করে
তাই AL use করার সাজেশন,
AH - যে predefined value মুখস্থ এর জন্য ব্যবহৃত

1, 9, 2 → predefined করার AH এ

Ah, 2 → output বা print করাতে → single character print করাতে
Ah, input

Ah, 9 → string এর জন্য
1 → single character user input

mov Ah, 2 → Ah এর মধ্যে 2 assign হচ্ছে

Add AL, BL → AL + BL = result → result টা AL এ store হচ্ছে

```
MOV AX, @DATA
MOV DS, AX
```

@DATA
↳ .DATA এর Address কে Ref. দেয়

↖ initialize করার জন্য DATA segment এ জন্য

Ch-1, 2 ↳ table Hexadecimal
↳ sample example

Ch-1, 2, 3 → Hall + Marut → Theory
↳ DOS → 3.3.1

* Power switch on করলে কী হয়? or
* PC on

** Ch-4 → Lab code

Variable নিয়ে কাজ → same as C, C++
↳ size declare করব, type নিয়ে মাথা ঘামাব না

# a db
DB = 8 bit → byte
DW = 16 bit → word

Declare & initialize 2টাই করা যায়

Add ~~ax, bx~~
Add .

ABCH → invalid
0ABCH → valid
↳ 1st এ এ দিয়ে Number শুরু Must

Add AL, BL → AL, BL register + ~~এর~~ AL ~~এ~~ result store ~~হবে~~

Add VAR1, VAR2 → VAR1, VAR2 variable

Memory . Memory operation ~~valid~~ invalid

→ invalid, ~~2টা~~ variable operation ~~করা~~ যায়

Add AL, BL

MOV BL, VAR1

~~Add MOV BL, VAR2~~

Add BL, VAR2

MOV VAR1, BL

← Add VAR1, VAR2

solve

AC

DL

4.6

MOV B, A → invalid

MOV Ax, A

MOV B, Ax

C = A + B

↳ MOV AX, A

~~Add B, AX~~

Add AX, B

MOV C, AX

MOV Ax, 1

MUL 1, 2

* ~~এই~~ Exercise practise

MOV DL, 0DH → carriage return → margin ছাড়ে

0AH → New line

যখন 0AH → new line হবে তখন First এ না তো আসতে পারে

" 0DH → এ Line এর First এ চলে আসবে

Appropiate next line → 0DH, 0AH 2টিই দিতে

user থেকে input নিচ্ছে তাহলে AL এ input store হবে

* যাদের sum বা result 10 এর নিচে তাদের sum বের করবে।

5+6 ≠ , 3+4 ✓

• Data → চারভাবে নেই variable না দিলে

* Case conversion

* Ch-2 → ASCII value

TSM
MSM

10-3-24

Lec-5

Pin Diagram

8086 → 40 pin → dual package

↳ 20pin 20 pin

signal আসা-যাওয়ার জন্য pin প্রয়োজন

communication

40 pin

↳ ১টা processor

↳ বা একাধিক ও

2 modes—

Minimum, Maximum Mode → pin দিয়ে Mode select করা

↳ single processor ↳ Multiple processor

→ 5V এ কাজ করে → Active হয় processor

Input দেয়ার জন্য আলাদা pin

2 টা pin → excessive voltage gain out out করে

S6 S4/A19 - A19/S3

S6/A16 - A19/S0 S0 S6 S5

AD0 - AD15

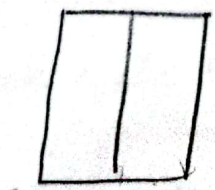
A0-A15 D0-D10

A16-19 S0-4

* pin এর নাম ও কাজ হচ্ছে, Direction

1st → General Pin diagram

2nd → Directional



→ Dual package

$V_{cc}(5V)$ → 5 volt input Power on করার পিন

V_{DD} → input voltage নেয়ার পিন

$V_{ss}(gnd)$ → 0V →

Clock signal

↳ synchronize → Data আদান প্রদান-

↳ execution purpose. এর পিন

Add. bus → 20 bit

Data " → 16 bit

1st clock cycle / odd cycle → Add. যাবে

2nd " " → Data আসবে-যাবে

ALE → Address Latch

↳ Active হলেই → Address যাবে

→ off " → Data "

ADO → Address এর পিন

S6 → status এর পিন

S_5 = IF S3 =

S6 = 0 S4 =

AB/S6 - A16/S3

↳ S4 S3

0 0 → Extra

0 1 → Stack

1 0

( )

RD :-

*Complement আকারে থাকে এবং পাঠাতে হয় Active করার জন্য

Ready →

↳ Slower device → secondary memory/disk

↳ Synchronization purpose এ use হয়

↳ Ready চেক করার জন্য

↳ Ready না হলে Wait state এ যাবে

↳ Memory Ready আছে কিনা চেক করার জন্য

Maskable

INTR → Interrupt Request → Request আসছে → device থেকে P

INTA → Interrupt Acknowledgement

↑ Non-Maskable

↳ Processor থেকে device এ যায়

S3,S4 → segment এর জন্য

TEST:

↳ Co-processor কে চেক করবে

↳ এর মাধ্যমে communication করার জন্য

RESET:

↳ Initial stage এ আবার ফিরে যাবে

↳ pin কে Active করার জন্য

↳ 4 clock cycle পর্যন্ত Active রাখতে হবে

NMI
↳ Non-Maskable Interrupt
↳ Decision দিতে পারে না Processor

Maskable → Decision দিতে পারে Interrupt

MN/MX
↳ Minimum, Maximum Mode select করে.

M/IO
↳ Memory / Input Output
Processor থেকে signal পাবে কাজ করে
↳ I/O device
↳ Memory/Register

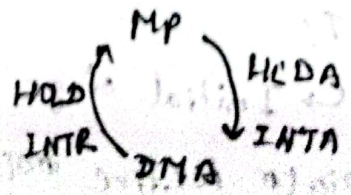
Test → Co-processor
Ready → Memory
HOLD → DMA

ALE
↳ Address Latch Enable
↳ Data or Address নেয়

WR
↳ Memory/IO → Data Write করে.

HLDA

DMA → HLDA, HOLD
↳ Direct Memory Access →
↳ Permission দিয়ে দিবে
↳ Memory থেকে সরাসরি Data নিতে পারে



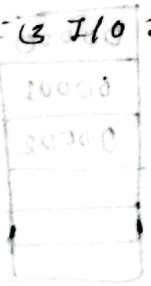
- → 0 মিনিমাম Active
1 " Inactive

DEN → Higher portion এ কাজ করার জন্য

DT/R → Data Transfer or Receive

↳ Active হলে → transfer

S0, S1, S2 → Read-Write operation হলে Memory ও I/O device এর সাথে communication করে → Bus Stage



Lock
↳ কাজে কাউকে Access করে না
↳ শুধু নিজের কাজ করে

RQ/GT1, RQ/GT0 → Request Grant

↳ Acknowledgement ↳ Grant দেয়া হয়

QS1, QS0,
↳ Queue এর status বের করে

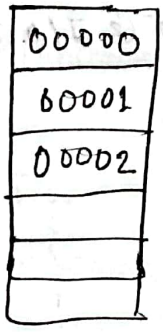
Queue
↳ কিছু instruction fetch করে store করার জন্য প্রয়োজন Execution এ পাঠায়।

Memory Banking

$2^{20} \rightarrow$ 1MB

512KB (Odd / Higher) 512KB (Even / Lower) 512KB

byte Addressable → 8 bit / 1 byte



Memory

3 category

↳ Min and Max Mode → (24 - 31) pin → Max, Min common

↳ Min
↳ Max → (1-22), (33-40) pin

* Maximum mode এর জন্য → (24 – 31) pin

* Minimum " → (24 - 31) pin

* Common → (1-22, 33-40)

BHE

↳ Bus High Enable

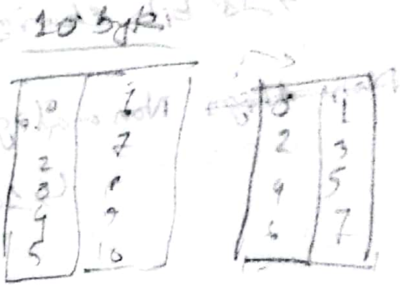
↝ Higher portion

16 → H / L

D = 16
A = 20

* Memory byte/8 bit Addressable — আর Data 16 bit এর
↳ 2 ভাগে ভাগ করে
512KB . 512KB

* Mis align Data



20 bit

- 00000
- 00001
- 00002

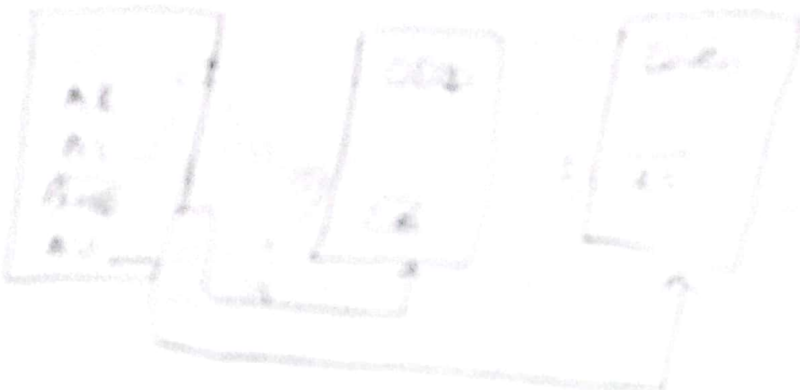
A19 ... A2	A1	A0
0 . . 0	0	0
0 . . 00	0	1
	1	0
	1	1
	6	6

A_2	A_1	A_0
0	0	0
0	1	0
1	0	0
⋮		

A_2	A_1	A_0
0	0	1
0	1	1
1	0	1
⋮		

1 → odd
0 → even

A_0 → selector
↳ chip selector



Align Memory → Memory ba

↳ 32 bit change offset

Then Align Non-align

↳ 8086 bit change 8085

RAM, ROM এর cost → material, size এগুলোর উপর নির্ভর করে। তাই Normally কোনটার price বেশি তা বলা যায় না

Lec-6

Ch-8

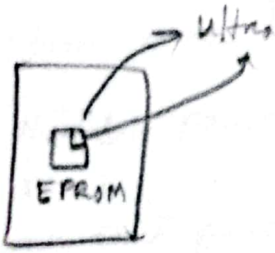
Semiconductor memory সম্পর্কে

Memory
↳ Primary
 Secondary

↳ RAM → volatile → power off → Data নাই
 ROM → Non-" → " " হলেও Data store থাকে
 → Instant কাজ করার জন্য RAM or Primary Memory ব্যবহার করা হয়

ROM
↳ PROM → Programmable ROM → User.
↳ EPROM → Electrically " " OTP → যেকোন জায়গায় হয় / ... হয়
↳ EEROM ↳ One Type Programmable
Manufacturer ↳ Shadow ~~ROM~~ RAM/NVRAM
↳ Flash ROM

User ↓ বার; OTP ↓ Manufacturer



→ Ultra voilet ray দিয়ে EPROM এ নিয়ে যাবে
↳ এর মাধ্যমে erase করা যায়

ROM ↳ ... ↳ Second

RAM ↳ speed ...

Hard disk

* size, ওজন, ... etc. → Book → Hall

* Flash ROM → Block by block Data Erase/Write করতে পারি

* Shadow RAM → অনেকটা RAM এর মতো
↳ Power off করলেও Data চলে যায়, on করলে আবার চলে আসে
↳ EPROM এ রাখা থাকে
↳ RAM, ROM 2টার characteristic ই আছে, cost বেশি কারণ EPROM extra use করতে হয়

RAM

↳ SRAM → Refresh করা লাগে না → Flip-flop দিয়ে তৈরি → faster, costly

↳ DRAM (Dynamic RAM) → Refresh করা লাগে → capacitor..

↳ Leakage current থাকে

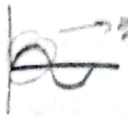
↳ cost কম, slower

↳ বেশি ব্যবহৃত

↳ Enhanced DRAM → (SRAM + DRAM)

↳ Dual data rate DRAMS (DDR)

Clock cycle →



→ শুধু +ve side এ কাজ করে E DRAM

+ve, −ve 2টাই কাজ করে DDR DRAMS

↳ faster → যেকোন দুটি দিয়েও করে

[Data দিয়েও করা যায় Error ধরা যায়]

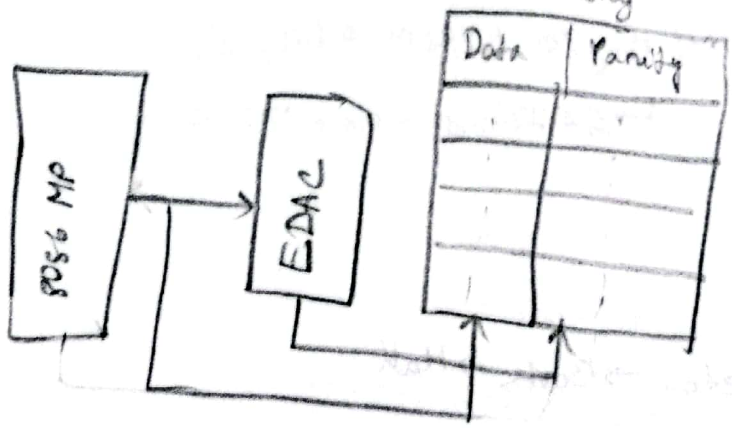
Hamming Code:

→ Parity position $= 2^n$; $n = 0 \ldots \infty$

→ Data bit " = other than 2^n

→ Ex: 10110

– – 1 – 011 – 0



EDAC = Error Detect AND Correction

7GAS632 → Error

Error

↳ Single bit Error → CRC, LRC

↳ Multiple " "

Hamming code → detection & Correction

↳ " " → Only detection

*Hamming code → position টা বের করে, error কে flip করে দেয়

2^n = 0, 1, 2, 4, 8, 16, 32, 64 ... → এই position এ parity bit থাকবে

Data parity + Codeword

P P D P D
↑ ↑ ↑ ↑ ↑

*Position wise Data Consider & skip

*যত Number position তত Number consider and skip ..... করতে

৩য় " " (যেটা check হচ্ছে [illegible], 2nd position হলে 2, 3 position consider, 6, 7 consider 4, 5 skip

* Even, ODD parity

1 0 1 1 0

S-1: 0 _ 1 _ 0 1 1 _ 0 → Step-1: 1st position [illegible]

St-2: 2nd "

St-3: 4th "

S-4: 8th "

P1 = 0

Step-2: 0 1 1 _ 0 1 1 _ 0

P2 = 1

S-3: 0 1 1 0 0 1 1 — 0

P3 = 0

0 → Even

P = 0

S-4: 0 1 1 0 0 1 1 0 0

P4 = 0

CW = 011001100

DW = 10110

* একটি Data দেয়া থাকলে, Data টি EDAC এ কীভাবে store করবে?

↳ উপরের মত করে codeword বের করব; CW এর value টাই store করবে

1 ↓

S-1: 0 1 1 0 0 0 1 0 0

P1 = 0

S-2: 0 1 1 0 0 0 1 0 0

P2 = 1

S-3: 0 1 1 0 0 0 1 0 0

P3 = 1

S-4: 0 1 1 0 0 0 1 0 0

P4 = 0

$P_4 P_3 P_2 P_1$

Parity → $(0110)_2 = 6$

↳ Error bit position

→ position এর bit কে 0 থাকলে 1 করব, 1 থাকলে 0 করব

Parity = 0 → No error

* Correction Code Word · থেকে Data Word বের করা

Code: 0 1 1 0 0 1 1 0 0
Data: 1 0 1 1 0

*RAM, ROM, Error Deetect & correction, Hamming Code

*Figure – 8.10

* এই উদাহরণে ~~Even parity~~ Odd Parity নিয়ে করবে

Lab
12-3-24
Lec-3

·Data
MSG DB 'Hello World $' → এভাবেই লিখতে

·CODE
MOV AX, @Data
MOV DS, AX

MOV AH, 9
LEA DX, MSG
INT 21H

Microprocessor এর ২০ নাম্বার .exe ২০ page এ

• Data

Var1 A DB 2; Var_Name Var_size value
Var2 B DB ?

• Code

```
MOV AH,1
INT 21H
MOV BL,AL ; MOV Var2,AL
MOV BL,B  ; MOV BL,Var2

ADD BL,A . ADD BL,VAR1

MOV AH,2
MOV DL,BL
INT 21H
```

* যেকোনো value user থেকে নিলে by default AL এ সেটা আসে। Then AL থেকে অন্য Register/variable এ AL এর value shift করতে

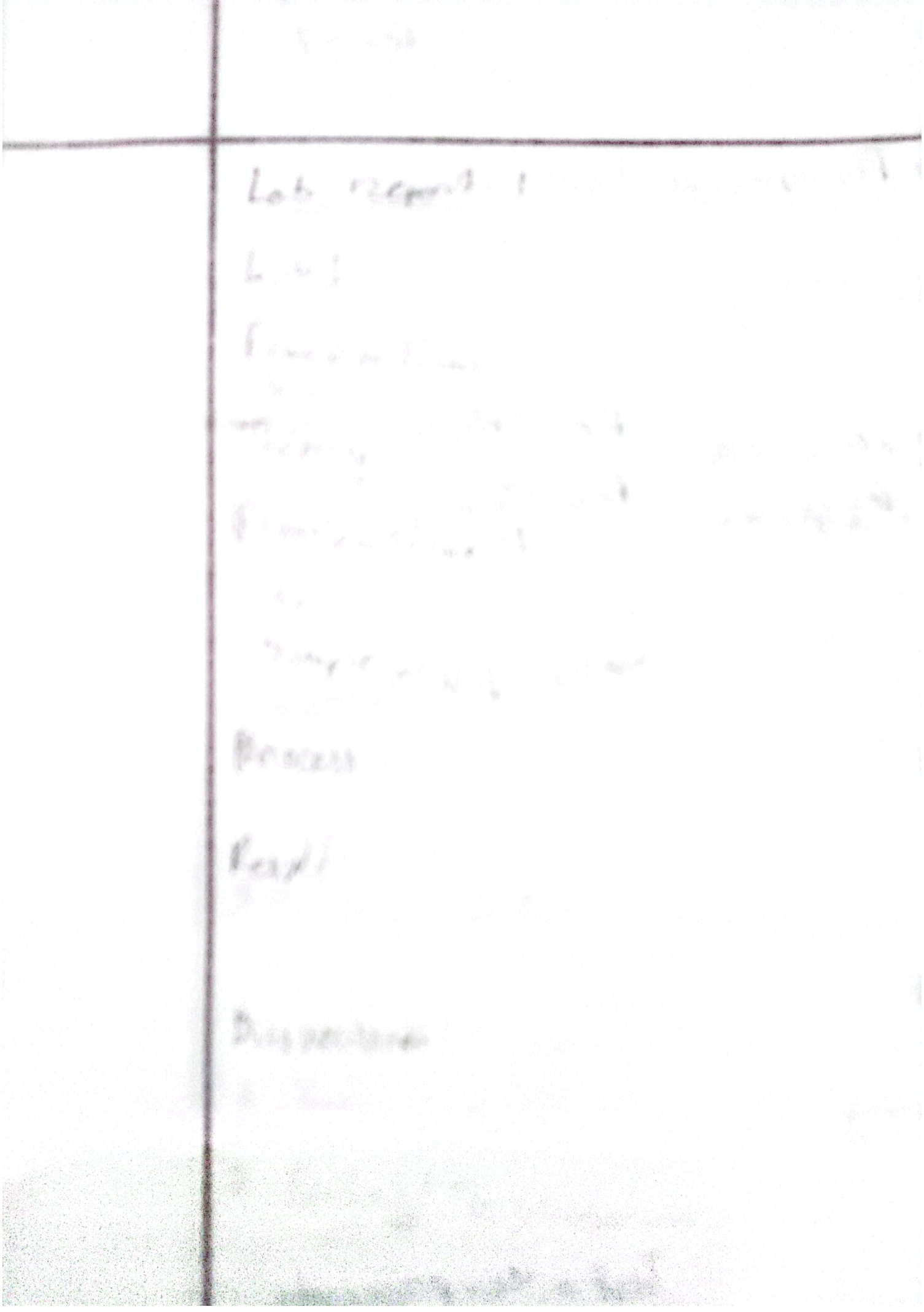
1 → 34 → 0
2 → 35 → char
3 → 69

LEA → Load Effective Address

↳ variable এর Address বের করে

LEA 12

8,7,(0),11,12
8,9,11,12 → EA



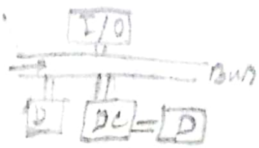
Ch-9 → Pheripheral Device নিয়ে

Maintain, control

Peripheral Device
- I/O device
- Controller
  - General DC → Basic কাজ
  - Special DC → কম speed এ

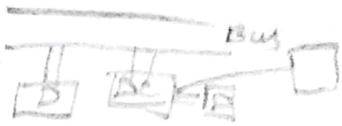
Read/Write → Data command

Read/Test
- Device টা Ready আছে কি check করে
- Non-Data command



→ I/O Modes
- Programmed/Pooled → DMA
- *** Interrupt Method
- DMA (Direct Memory Access) → Halt
- I/O channels
- I/O processing → Data chaining

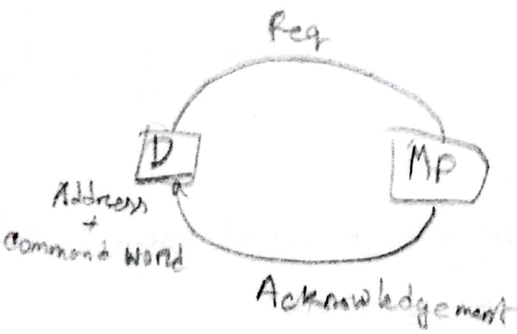
Memory এর কাজ →



Addressing:
- I/O Mapped
- Memory "

Test = co-processor

Ready = Memory



← Programmed/Pooled

→ বারবার knock করে দেখে — Disadv.

* Processor Device কে Address করে

I/O Mapped Addressing → প্রত্যেকটা I/O device এর জন্য আলাদা আলাদা Address ও Command word থাকবে।

Adv → secured, easily

Disadv. → ready না হওয়া পর্যন্ত Processor idle থাকে
Slower device হলে ...

I/O:

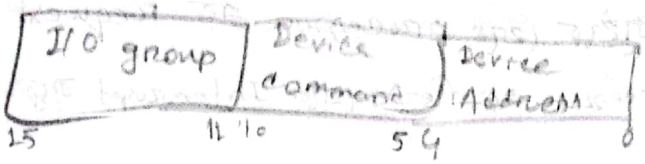
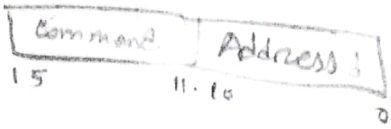


Fig → 9.2, 9.3, 9.4

Memory:



↳ group wise same command & Address

↳ একাধিক Device এর same Address হতে পারে

3 Br Stage ⇒ Data Communication হয়

1. Initialize
2. Transfer
3. Ending/ Termination

2. Interrupt Method: → Processor এর কাছে Request

↳ One type of signal

~~↳ কোনো কাজের জন্য সংকেত প্রদান করে~~

↳ কোনো কাজের জন্য সংকেত প্রদান করে

* Memory Map
* I/O Map

Muskable → Processor কোনো সংকেত উপেক্ষা করতে পারে

Non- " →

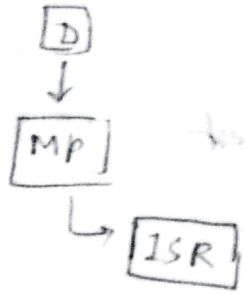
18-3-24
Lec-8

1. External Interrupt: বাইরে থেকে processor এর request আসলে
2. Internal : Processor এর ভিতর থেকে Interrupt আসে

External 2 types:
1. Muskable
2. Non-Muskable
→ Hardware

D → Device
DC → " Controller

Internal Int. 2 types: → software
1. Software Int. → Coding/software দিয়ে Interrupt করলে
2. Specialized Int → Invalid operation এর জন্য, power off হওয়ার আগে এই ধরনের Interrupt ব্যবহার করা হয়,



ISR → Interrupt service Routine
↳ এমন একটা প্রোগ্রাম কী ধরনের Interrupt হলে তার একটা routine আছে,

Interrupt response time: Request পাঠানোর পর Interrupt reponse করতে যে time লাগে

Device:

~~ISR (Interrupt Service Row~~

ISP (Interrupt Service Procedure)

ISR: Interrupt Service provide করে

* CS →

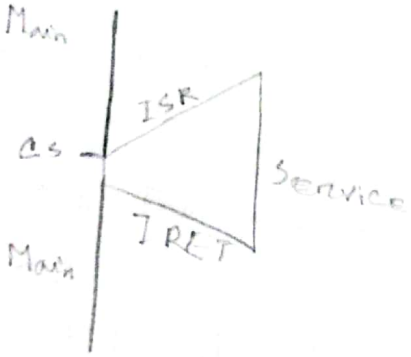


Fig-2.9

IRET → Interrupt Return

Segue status: Flag, pointer এর value store করে

Main program থেকে Jump করে ISR এ যাই,

6 steps:

1. Stack decrement → 2 bytes
2. Flag Register এর value store
3. Interrupt Flag disable করে (IF=1)
4. Trap Flag " " ; Trap Flag → Reset করে
5. CS → segment + Offset ; CS, IP store
6. ~~Stack segment decrement করে, Fla~~

↳ Instruction pointer, Flag store করে

6. ISR এ Jump করে

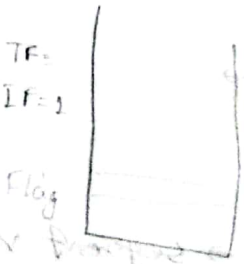


Fig: 2.13

~~ISR (Interrupt Service Rou~~

ISP (Interrupt Service Procedure)

ISR: Interrupt Service provide করে

* CS →

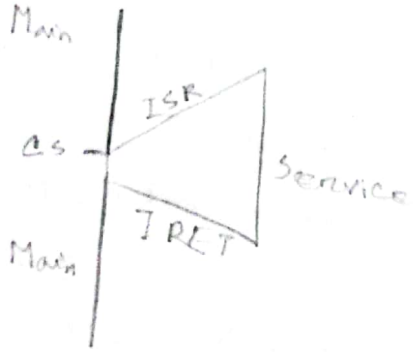


Fig-9.9

IRET → Interrupt Return

Seque status: Flag, pointer এর value store রাখতে

Main program থেকে Jump করে ISR এ যাবে।

6 steps:

1. Stack decreament → 2 bytes
2. Flag Register এর value store
3. Interrupt Flag disable করে (IF=1)
4. Trap Flag " " ; Trap Flag → Reset করে
5. CS → segment + Offset ; CS, IP store
6. ~~Stack segment decrement করে~~, Fla

↳ Instruction pointer, Flag store করবে

6. ISR এ Jump করে

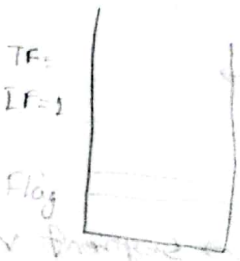


Fig: 9.13

Opposite order এ Load করতে:

1.

2.

3.

4.

5.

6.

ISR → Address নির্ণয় করতে

↓

Service Procedure Address → IS

↳ 256 ধরনের Interrupt ব্যবহার করতে পারি

CS → 2 byte .] → 4 byte → ISR এর কাজ

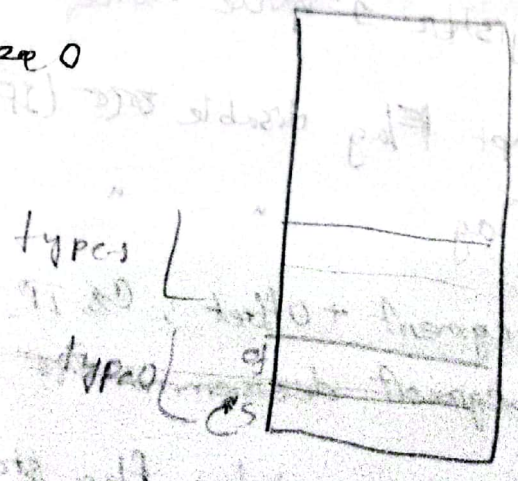
IP → 2 "

Fig - 9.14

1KB →

Lower → segment value → type 0

Upper → Offset →



8086 থেকে Developed processor এ লক্ষ করবো:

5 - 31 → reserve INT [Advance processor এই আসলে INT use করতে পারে]

32 - 255 → user এ নিজের ইচ্ছামতো কাজে লাগে

1st 5 টা → predefined

Type
↓
type No×4 = CS
CS+2 = IP

Fig 9.16

29-255

type-0: divide 0

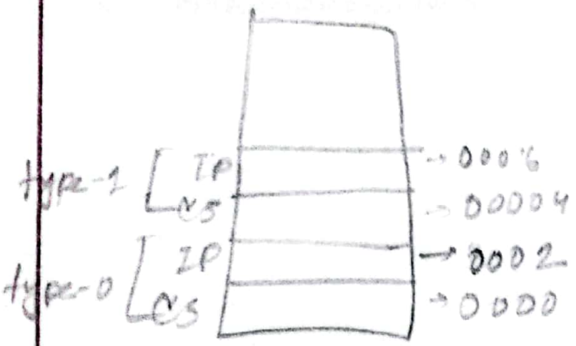
" → 1: Debugging

" → 2: Non-maskable

" -3:

" -4: Overflow interrupt

IP এর জন্য 4 byte প্রয়োজন হয়



[Device MP কে প্রথমে INT পাঠায়
তারপর type No ও পাঠায়]

5 types → Self study

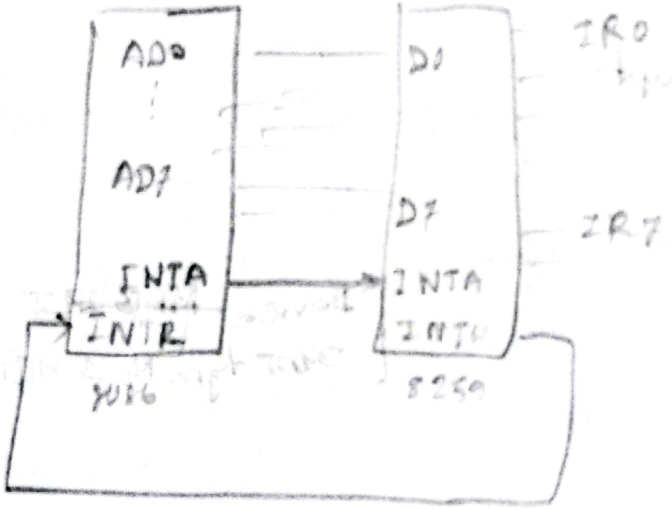
↳ Address
↳ Service Provide
↳ কোথায় ব্যবহার করতে পারি /
Application
↳ Missing মানুষকে বের করতে online থেকে

Interrupt Handle:

* Multiple Interrupt কীভাবে Handle করে?
  - একাধিক একটি Device নিয়ন্ত্রণ
    - 8259 processors

* Nomally single interrupt handle করতে পারে।

* Interrupt Controller কীভাবে কাজ করে?



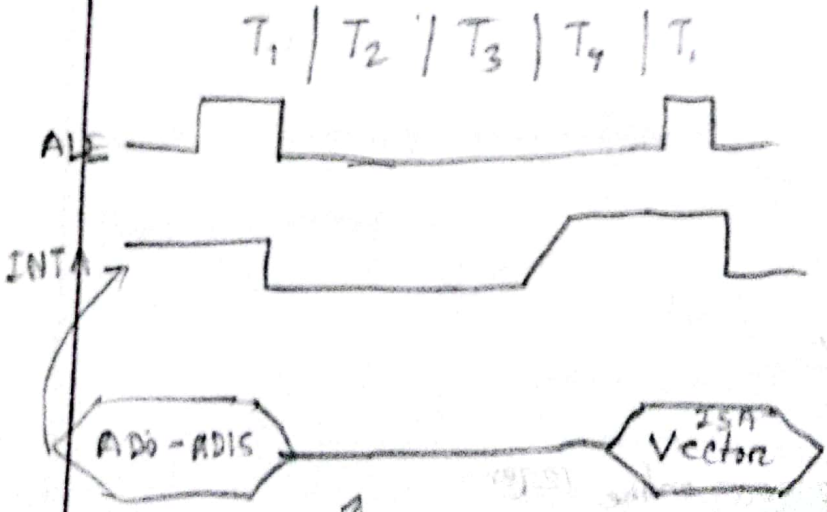


Fig - 9.17, 9.18

1st INTA Active

2nd Acknowledgement Active

Data Bus এর type No

Floating Point, Active হয়ে যায়

Priority: Multiple INT আসলে কোনটার আগে সার্ভ

IR0 - IR7 → Interrupt আসবে 8259 এর কাছে

INT একটা Interrupt কে select করে 8086 এর INTR এ পাঠাবে.

2 cycles:

1. Ready করে

2. Type Number টা পাঠাবে

ALE → Address Latch Enable

↳ 1st এ Address পাঠায়

2nd " Data "

1টা cycle 4টা Part

× প্রত্যেকটা timing diagram 4টা time এ

2টা Acknowledgement → Ready, type No.

↳ Address পাঠাবে

↳ 2nd

1st

* 8259 Internal Component:

↳ Fixed Priority: ঠিক করে Priority নিয়ে কাজ করে

IR0 → Priority বেশি

IR7 → " কম

IRR (Interrupt

Interrupt Inservice Register (ISR)

↳ কোন Interrupt service এ আছে

trace করতে হবে কার কে service এ আছে, pause করতে পারে

Mask Register : IR0 - IR7 : যেগুলোকে চাইলে on/off করতে পারি
ইচ্ছামত পারি

A_0 = কোন Address টা Define হবে → 2টা Add. কে define করতে পারি

CS → Chip Select → কোন Device টা Select করবে

8259 → 1টা বা 2টা use করতে পারি তবে INT একটাই দিবে

* 1টা 8259 এর 8 টার সাথে আর 8259 এর 8 টার সাথে connect করলে

তা case coding হবে।

* SP/EN → single নাকি Multiple use করছি
  ↳ Processor এর সাথে connected → master
  অতিরিক্তগুলো → slave

* Control Logic → Acknowledgement পাঠাবে

* Cascade Buffer.

* Read-Write