

Microprocessor & Assembly language:-

peripheral & Interfacing (Microprocessor related)

8086/8088, we discuss this too Microprocessor.

High level language

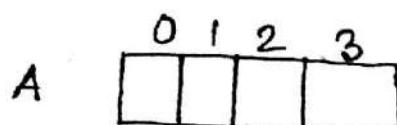
下

-Assembly language (ફોર્મ ફોર્મ વૉર્ડ ફિક્સેડ કોડ અને ક્રોન્ડ કોડ એન્ટ્રીસ)

Machine language

→ यहाँ **polocatable** था कि, जैसे language ने address
जैसे address fixed ना।

Machine ~~के~~ address fixed घरेका ।



Poloatable করা হয় Assembly Jangar

A एको machine लाई start भय्ये A आराकोटो machines

1000

Start करना चाहे

A123

A123

A123

A123

A123

A123

A123

A123

A123

पहले पुस्तक आदर्श offset लिखा कराया रखा एवं New
address देखी रखा अर्थात् relocatable रखा।

03-05-17

3st (d) day

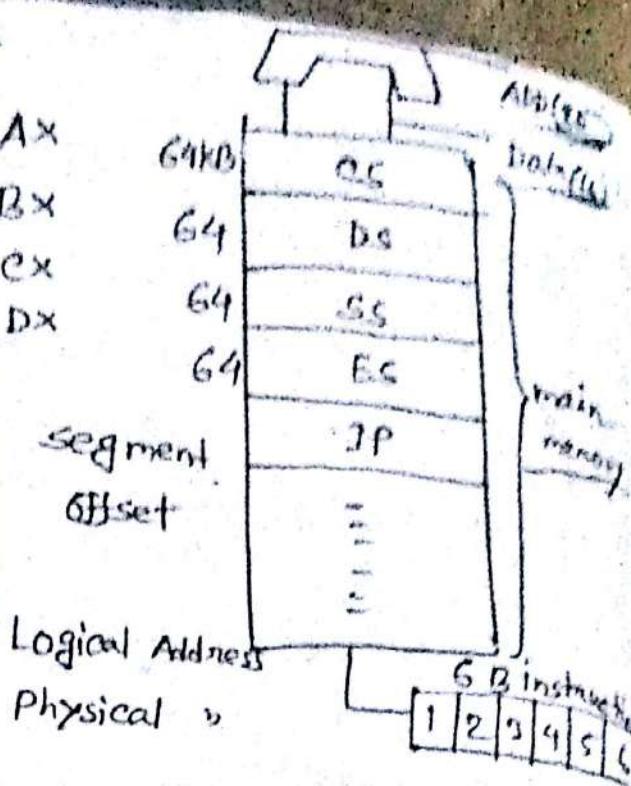
8086 ✓ 8 + 8 = 16 bit

एलगोरिदम

16 bit

8086 → At a time
16 bit data process
एलगोरिदम

AH	AL
BH	BL
CH	CL
DH	DL
SP	
BP	
SI	
DI	



8086: 16 bit Microprocessor. At a time 16 bit data process करता पात्र।

ALU → processing इवे, धूर fast व्ही

BUS → 3 बियनें (i) Data bus/
(ii) Address,
(iii) control.

Register → 8 bit / 16 bit Now 32 bit

ALU fast, Memory slow आई एलगोरिदम cash
memory बायर। memory. current data Registers
यात्रे।

220
270

CS → code segment
DS → Data ↴
SS → Stack ↴
* ES → Extra ↴
IP → Instruction pointer

Data bus → 16 bit
Address, → 20 bit
एलगोरिदम 20 अंदर्यार्दम memory
Address लेनी करता पात्र,
RAM लेनी करता पात्र।

IP → Next instruction का point में।
 6B instruction queue का next instruction होगा।
 Current Address → 32 bit।

Data bus → 16 bit
 Reg: → 16 bit
 ADD bus → 20 bit

32 bit
 16 bit
 16 bit
 16 bit

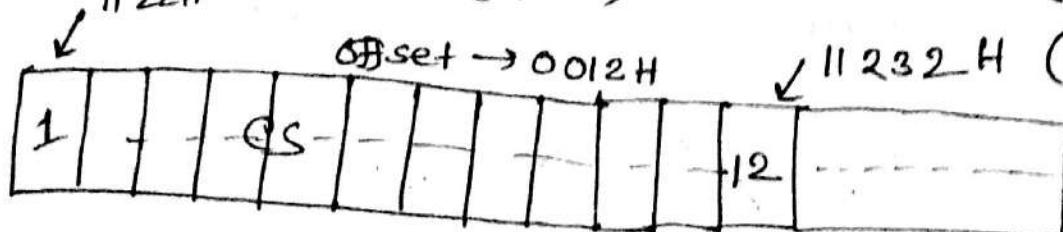
20 bit का 16 bit का convert करा जाए तो 4 bit का offset होता है।
 logical Address का 1 segment का offset होता है।
 इस logical Address का logical Address + segment (16 bit) होता है।

segment व अफ्सेट के बीच 20 bit का होता है।
 physical Address. (20 bit)

left shift
 segment → 1122 (Hex) → 11220 H

offset → 0012 (Hex) + 00012 H

1122H or 11220H (segment) + 00012H (offset) = 11232 H (Physical Address)



MASM.exe
 LINK.exe

→ download करेंगे।

→ better

RISC → Reduced Instruction Set Computer

CISC → Complex

H.W.	RISC / CISC
प्रारंभिक सुला पक्ष	

~~of
2nd (A) day~~

Name

Operation
[Opcode]

Operands

; comments

B

Name → किसी code का group करने level से
कहते हैं जिनमें, then पुढ़ाने पर आवश्यक
(GO TO इसे कहते हैं)

; comments → Comment कहा जाता

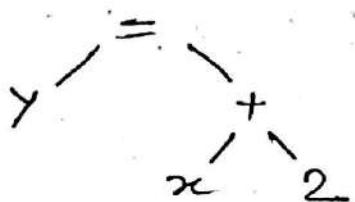
Operation → Mathematical operation

Operation को command या key word कहा
जाता है और opcode बहुत।

Operands → Operation पर पड़े operands
लिए जाते,

$$Y = x + 2 \rightarrow C$$

ADD x 2 → Assembly



Parsing tree के अधिकार्य इन्हें ADD को शामिल
करते हैं।

Operands → 0, 1, 2

रुपांतर Operand संस्करण में (0)
वार्ता , , , \Rightarrow नियम (increment
decrement)
जूड़ी , , , नियम ADD X 2

pseudo-op → opcode

ADD AX, 2
↓ ↓ ↓
Fixed Hexa binary
binary numbers binary

आवश्यक: एडटोए गलावन

— कोड 110101

main()

}

function वा अन्यथा

MAIN PROC → function शब्द
pseudo op ← {
MAIN ENDP → " " शब्द
अन्य function वा आवाहन निम्नलिखित
Name होते।

keyword शब्द — कोड राहत देते (Good practice)

Numbers

Binary \rightarrow 11010...B (Byte 8 bits)
Decimal \rightarrow 10, 10D (Byte 4 bits, On by default)
Hexadecimal \rightarrow 1001H, FFFFH (Byte 4 bits)
↓
Hexadecimal \rightarrow 1001H
0 bits 255

Char type

'A', "A", "Hello", 'H', 'e', 'l', 'l', 'o'

Type

DB \rightarrow BYTE (1 byte)
DW \rightarrow Double BYTE
DQ \rightarrow Four word
DT \rightarrow Ten Bytes

Initialization :-

variable name = var1, Byte type ~~to~~ 8 bits
value ~~is~~ 102

var1 DB 102 \rightarrow Initialize

var1 DB ? \rightarrow Un

char var1 \rightarrow variable

ARRAY1 DB 10H, 20H, 30H, - - -

constant defn:-

define PROMPT Enter choice
↓
EQU PROMPT Enter choice

= 0 =

MOV destn, source

Gen Reg → AX, BX, CX, DX

Seg Reg →

Mem location →

Constant →

x = y এর ক্ষেত্রে

MOV x, y ← কোথায় না

কবল হবে,

MOV Ax, y } Register
MOV x, Ax } Memory to memory transfer

10 = 20 এর ক্ষেত্রে,

MOV Ax, # 20

MOV 10, Ax

Exchange reg:-

XCHG destⁿ, source
swap dest

Reg to Reg or and memo loc to mem loc

NEG

AX

→ 2's complement $\overline{X} \overline{Y} \overline{Z} \overline{C}$

= 0 =

- MODEL SMALL
- STACK 100H
- DATA, (variable declaration) ^{size} → Data definition
- CODE

MAIN PROC

→ Instruction.

MAIN ENDP

END MAIN

$$R = P + 3Y$$

~~ADD~~ MOV AX, P ¹⁶ ✓ ↓
ADD AX, Y
ADD AX, Y
ADD AX, Y
MOV R, AX

`MOV AX, P`

जूदे same size वा बहुत बड़ी
AX बाटे 16 बिट पर 16
बिट बहुत ।

~~08-06-17
2nd (B) day~~

Microprocessor योग्यता का तथा अवलम्बन interrupt
3 types:

- i) software interrupt
- ii) Hardware " "
- iii) Non maskable (NMI)

interrupt → रोग कारण ग्राहक

i) software द्वारा किया code द्वारा कारण I/O करने
आयाम सफ्टवेर द्वारा ग्राहक घटे, एवं इस
problem इसे लोड सफ्टवेर interrupt कहा

एल्फाकोडे interrupt द्वारा against किया function
या use करने द्वारा एक चला द्वारा routine
interrupt vector routine.

input, output device - & interrupt routine को
call करने के नहीं output वा प्रोग्राम को लिखना
corresponding interrupt routine को call करना input
द्वारा पढ़, तथा एवं output ---

10

Now value display:-

INT interrupt Number

Function No.

AH $\left\{ \begin{array}{l} 1 \rightarrow \text{single char. Input} \\ 2 \rightarrow \text{single char. Output} \\ 9 \rightarrow \text{Character string output} \end{array} \right.$

1

AL \leftarrow ASCII

2

DL \leftarrow output char

interrupt Number use 21h (21h)

21h द्वारा DOS द्वारा विभिन्न function को call करने पाये।

AX द्वारा हटाए गए part \rightarrow

AH	AL
----	----

1, 2, 9 द्वारा ही कैसे करने वा अपन्ना यथा प्रयोग करते AH द्वारा।

Value लिये व शोव करना, (single char input, single char output)

- MODEL SMALL
- STACK 100h
- DATA
- CODE

MAIN PROC

```
MOV AH, 1  
INT 21h  
MOV DL, AL  
MOV AH, 2  
INT 21h  
MAIN ENDP
```

END MAIN

→ code filename • ASM file → save ~~mpa~~

MASM.exe
LINK.exe

disktto command को प्रोग्राम लिख आज्ञा

SUB AL, 30h

~~18-05-2017
2nd (D) day~~

9 ← AH (string output)

String ये address से ग्राहक द्वारा DX परिणाम
, " के end करा इस \$ परिणाम।

प्रश्न: 'Enter your choice: \$'

LEA → Load Effective

• DATA

MSG DB 'Enter your choice: \$'
↓
MSG ये Address

• CODE

LEA DX, MSG

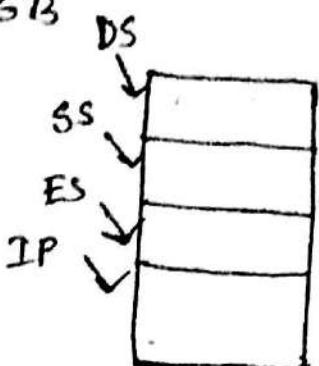
प्राप्त ये मूल

MOV AH, 9

INT 21h

= O =

PSP → 256B



DS → Data segment
G3 Register

DS Point दूरी

Programme load होता है PSP → 256B के बाहर यहके
data के point होते हैं। विलु Point कहाँ पर
विलुके

* CODE

MOV AX, @ Data

Data Segment G3
register add. के
Point दूरी

MOV DS, AX

AX → General
Registers

LEA DX, MSG

DS → segment
Registers.

MOV AH, 9

INT 21h

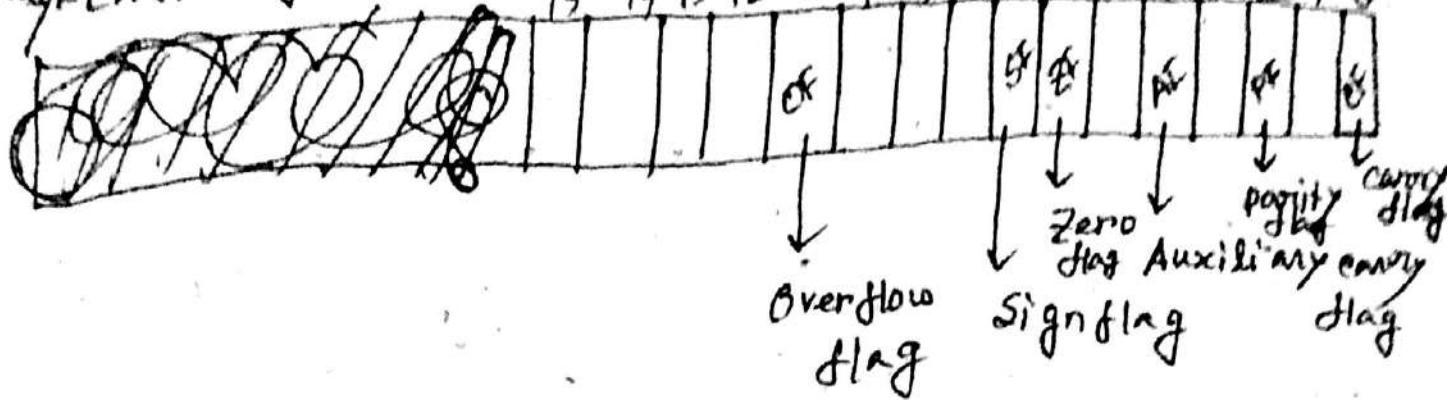
= O =

MSG DB 'You have entered'

MSG DB ?, '\$'

Scan करना करना

Status/FLAG Registers



QB Register GOT,

C

while ($c > 0$) {

}

तार्किकी करने के लिए idea से general
प्रोग्राम

carry flag जा कर either 0 or 1

ADD / SUB

$$\begin{array}{r}
 \text{15th bit } G \\
 \text{15th} \\
 0 \ 1 \ 0 \ \dots \\
 + \ 1 \ 1 \ 0 \\
 \hline
 0 \ 0
 \end{array}$$

मध्यन विना 1 के लिए तथा carry flag
के value 1

$$\begin{array}{r}
 \text{15th} \quad \text{14th} \quad \text{16th bit } G \\
 0 \quad 0 \ 0 \ \dots : \quad \text{लिए } 0 \\
 1 \quad 1 \ 0 \ \dots : \quad \text{लिए } 1 \\
 \hline
 \end{array}$$

मध्यन carry flag के value = 0 कार्य 1 करना

14

Parity flag: Low byte $AX \rightarrow \boxed{AH} \boxed{AL}$

Result

$FFF_h \rightarrow$ आउटपुट

Low Byte एवं even अधिक 1 प्राप्त करने पर parity flag 1 otherwise 0

Auxiliary carry flag:-

Low Byte एवं MSB check करते

8 bit एवं आउटपुट करते हैं

Zero flag: Result Non zero इसे zero

Zero \rightarrow One

Sign flag:

Result

$FFFF_h$

Result यदि 1 आउटपुट एवं sign
0 , , , unsign

Last bit का आउटपुट परामर्श यहाँ same
इस उत्तर overflow घटेगी।

Now example:

$$\begin{array}{r}
 0121 \text{ h} \\
 + 1010 \text{ h} \\
 \hline
 1131 \text{ h}
 \end{array}$$

binary $\rightarrow 0011\ 0001$ carry flag $\rightarrow 0$

Auxiliary carry flag $\rightarrow 0$

Zero flag $\rightarrow 0$

Sign flag $\rightarrow 0$

Overflow flag $\rightarrow 0$

$= 0 =$

JNZ \rightarrow Non zero value check करें। (नियमित वल
check करावा रखें)

15-05-17
3rd (A) day

MOV AL, 80h

MOV BL, 80h

ADD AL, BL

Flag \rightarrow SF=0
ZF=1
DF=1
CF=1
PF=1

AL - 6 गले 8 हो 0

जो ZF = 1

MSB और sign flag = 1 जो SF = 1 Result 100h

$$\begin{array}{r}
 80h \\
 + 80h \\
 \hline
 160h
 \end{array}$$

$\rightarrow 100h$

$80 \rightarrow 1000$

1000

1000

10

अटे Flag = 1

even परिष्वक One \Rightarrow PF = 1

वैनन OF = 1

Signed overflow \rightarrow घरेग ~~bit 6~~ आण, तो

आआहारद \rightarrow यारेह

Unsigned overflow \rightarrow यधन carry flag 1 इव

Result 6 या आआव कर्या, ता ना आआले उष अस

Signed overflow.

Jump (conditional)

0. J*** destination - label

1 MOV —————

2 ADD —————

3 * CHG —————

4 MOV —————

5 ADD —————

6 J*** label

7 MOV —————

8 MOV —————

label :

10 MOV —————
 11 INT —————
12 ADD —————

Jump flag अनाव चेप्त dependent
 ADD द्वा चेप्त लोस करते status flag change
 अलग

JNZ → G depend करते ZF अपेक्षित

0 और यदि JNZ तो अलग ZF यदि 0 तो ZF
 अव अधन 6 ~ point करते then 6 ~
 अलग IP डे 9 ~ point करते 9 then
 label G याव।

JNZ depend करते
 ZF flag द्वा अलग

Now Jump Instruction:-

Signed Jump :-

i) JG / JNLE $\rightarrow ZF = 0, SF = OF$

JGE / JNL

JL / JNGE

JLE / JNG

Unsigned Jump :-

JA / JNB $\rightarrow CF = 0, SF = 0$

JAE / JNB

JB / JNAE

JBE/JNA

Single Flag Jump:-

JE \rightarrow ZF = 1.

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

;

Now the real code:-

AL ←

MOV BL, AL

AL ←

CMP AL, BL

JE print - same

print - same ?

MOV AH, 9

LEA DX, MSG1

INT 21H

*DATA

MSG1 DB 'same \$'

MSG2 DB 'NOT same \$'

JE print - same

MOV AH, 9

LEA DX, MSG2

INT 21H

exit ?

16
05-2017
3rd (B) day

Signed
conditional Jump:- JG/JNLE $ZF = 0 \ \& SF = OF$
Conditional
Unconditioned signed Jump:- JA/JNBE $CF = 0 \ \& ZF = 0$
 Jump above.

Example:-

$$AX = 0FFFh$$

$$BX = 08000h$$

as usual BX यहाँ, sign version of AX का

CMP AX, BX

[यहाँ, AX यहाँ, sign version
को ना करें]

JA print - AX

MOV AX, 0 → अगे कागड़ काल्पनिक है।

Print - AX :

→ → , , , true होता

$$0FFFh \rightarrow 0111\ 1111$$

$$08000h \rightarrow 1000000$$

$$\hline (-) \hline 1111\ 1111$$

JA/JNBE एवं जैसा

अगे नहीं

$ZF = 0$
$SF = 1$
$OF = 1$
$CF = 1$

The above portion is diff. between signed & Unsigned Jump.

Unconditional Jump:-

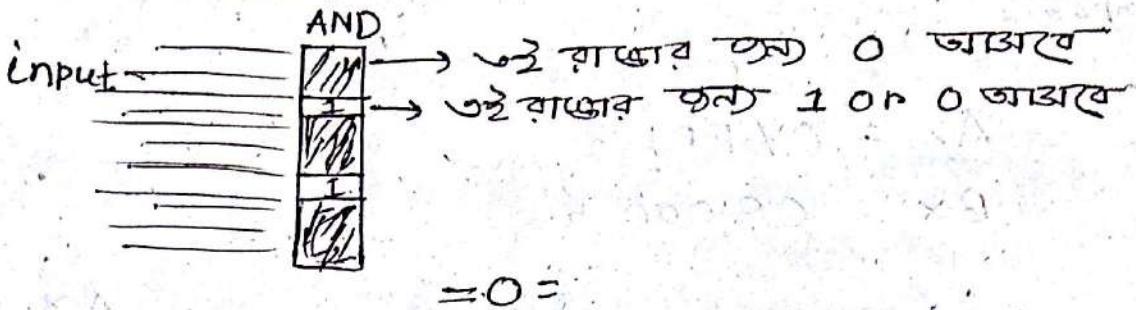
JMP destination-label } → break वा छुट्टा
 } → कागड़ करें
 } → loop दृष्टिकोण कागड़ करें

AND → destn, source

OR → " , "

XOR → " , "

AND use कठाव क्षण इस masking कठाव क्षण ।



String Input character Input तिल्य या lower case

इस अद्यते Upper case:-

MOV AH, 1

ASCII अक्षर signed

INT 21h

Unsigned अक्षर problem
नाही

CMP AL, 'a'

JLE JL exit

input < 'a' → exit

CMP AL, 'Z'

input > 'Z' → ,

JG exit

JL → Lower than

JG → greater ,

AND AL, 0DFh

'a' → 61h

'A' → 41h

MOV AH, 2

MOV DL, AL

INT 21h

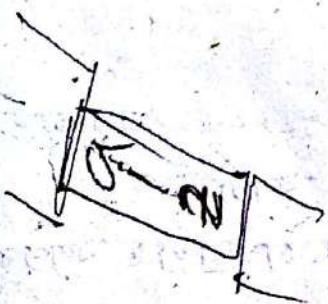
'a' → 0110 0001

'A' → 0100001000000000

exit :

'a' कठव Mask

$\frac{11011111}{D \quad F}$



02-07-17
4th (A) day

23

JMP Label

पदान्त, JMP के द्वारा
if, else पर कैसे प्रभाव
कार्य करते हैं।
किसी code skip करते हैं।

Label :

JMP Label

आरोक्षकारी प्रिक्लॉपिंग
पर कार्य करते हैं

= O =

MOV CX, 5

Label :

LOOP Label

→ loop में 5 तक सुनिश्चित
CX loop counting किया जाएगा।

CX का loop पर कार्य करते हैं, set करते हैं, लिखते हैं
प्रिक्लॉपिंग infinity अंधकारी कार्य करते हैं।

= O =

'a' के 10 तार प्रिंट किया?

MOV DL, 'a'

MOV AH, 2

MOV CX, 10

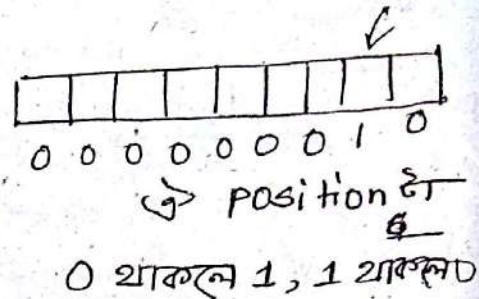
PRINT:

INT 21h

LOOP PRINT

=0=

AND → AND AL, 2



OR → OR AL, 2

→ position ० ते १
चाकरवे ।

XOR →

Assembly language कने use करा?

- Ans:- i) Hacking व कार्ड नाई
- ii) Compiler design ते त्या
- iii) Assembly ते code कर्ते feature क्षेत्रे-
add करा यास ते किंतिझोर आणार
करा नाई त्यातील cost तेही आणार्ग-
त्या use कर्ते code मुश्यान, तर अन्य
cost करा होय।

TEST destⁿ, source } AND operation वा गुणन
करते destⁿ change होते

एकटे number odd वा even check करें,

AND Ax, 1.

JZ

PRINT - Even.

00 → 0

01 → 1

10 → 2

11 → 3

JZ = 0 तब जूँह रखें.

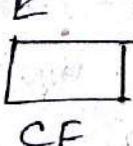
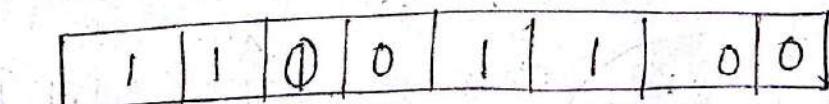
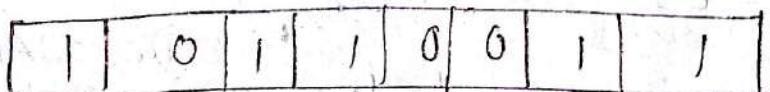
PRINT - Odd.

→ एकटे AND वा गुणन TEST use करवें,

= 0 =

Left shift

SHL AL, 2 → 2 घट याकरिता shift होता है।



एकटे एक रखें एकटे CF वा बॉल बानावें।

MOV CL, 2

SHL AL, CL

SHL AL, 2

MOV CL, 2

MOV AL, CL

RIGHT SHIFT,

SHR AL, CL

MOV

Check करवे CF व कठज़रूरी

JC → Jump if carry

CF = 1

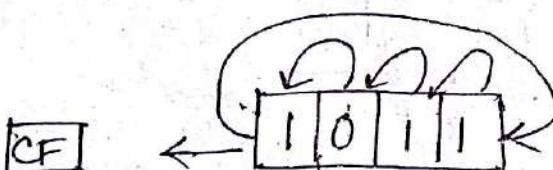
CF = 1 एवं JC का करवे

एकाई bit stream ना रखेंगे

1 आए तो CF का use
करवे, 1 पाईजे Counter
वापस दिये।

= 0 =

RCL → Rotate carry left



AL

RCL AL, 1

then इसे

AL

0	1	1	1
---	---	---	---

-15

AL

1	1	1	1	0	0	0
---	---	---	---	---	---	---

15 → 00001111

11110000

1

11110001

एयर 2 टिक्के जग
करवे इसे Rightshift

AL

0	1	1	0	1	0	0	0
---	---	---	---	---	---	---	---

CF

वालादेके positive अस्त्र

जब इसे positive

अपेक्षित SHR use

करवे ना SAR use

करवे।

यद्यपि SAR use करवे

then

AL

1	1	1	1	1	0	0	0
---	---	---	---	---	---	---	---

NO CO, अमास 2's complement

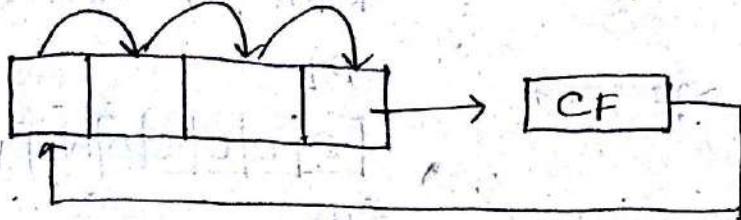
000000111

03-07-17
4th (B) day

27

RCL / RCR

Rotate carry left / Rotate carry right

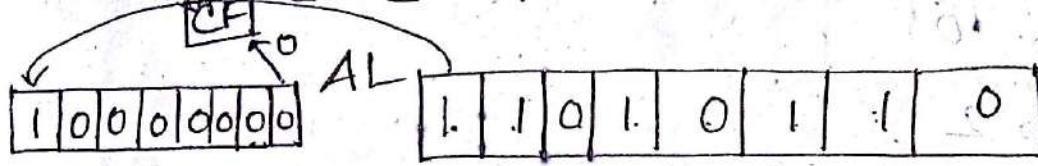


RCR destination, CL

Or

RCR destination, CL

left side এর টাই সেম, তাই আলাদা



BL

কাব্য চাই

01101011

SHL AL, 1

RCR BL, 1

BL এর কাব্য অপর কিছু 0 রয়ে থাএ, then,

XOR BL, BL

Binary Input:- MOV CX, 8

XOR BL, BL

MOV AH, 4CH

TAKE

INT 21h

AND AL, 0FH

1 টা Hexadecimal

code এর 31H

00001111 → 0Fh

31h

100---

TAKE:

OR BL, AL

SHL BL, 1

Loop TAKE

၁၃၅

Store এবং BL বা স্টোর

BL

1	0	1	1	0	1	0	1
---	---	---	---	---	---	---	---

Now display. काढ़ते काढ़ि

PZ 0

MOV DL, '0'

MOV AH, 2

INT 21h

Po

MOV DL, '1'

MOV AH, 2

INT 21h

SHL BL, 1

J.C. P.O.

AL 24

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

BL

00000000

then on ~~কাদুলি~~,

BL

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

SHL

00000010

→ वृत्त एवं लूप की विवरण

Hexadecimal Input:-

Input नंबर 24

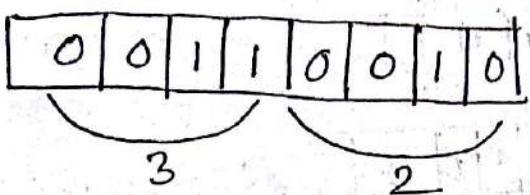
29

30h } 0
|
39h } 9

प्रथम 2 एवं उनका आउटपुट 32h

INT 21h

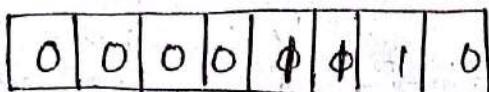
AL



उत्तरार्थाव के प्रकार

Now,

BL



प्रथम चारों 6+ one का

Now, code,
MOV CL, 4

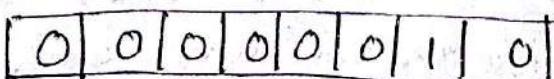
INT 21h

MOV BL, OFH

AND BL, AL

Now and करें,

BL

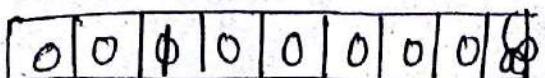


→ जूही 2 आउट

एवं BL के चारवां left shift करें,

SHL BL, CL

BL



MOV BH, BL → एजे ना करने masking
दो नहीं हो

Now,

AL

0	0	1	1	0	1	0
---	---	---	---	---	---	---

BL

0	0	0	0	1	1	1
---	---	---	---	---	---	---

BL

0	0	0	0	0	1	0
---	---	---	---	---	---	---

एवं BL & BH merge करने 24 परियां

याहै।

digit यादि ना हो तब A --- F तक पर्याप्त

अध्यन INT 21h के पर एकले CMP operation

आवर masking शुल्क change होते।

Lab:-

write a programme that

- take character from user until CR is pressed
- output the 1st & last lower case letter entered by the user entered by in alphabetical Order.
- If all entered characters are not lowercase letter, it will display one message
- "NO lower case letters"

10-07-17
6th (B) day

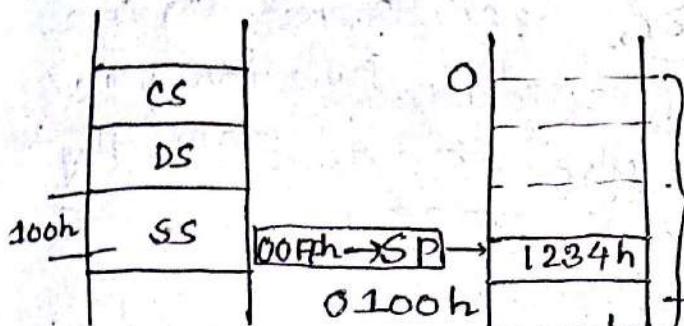
31

STACK

• STACK 100H

→ Stack वर size (optional, size ना

पड़ना चाहे, size ना दिया
by default 1 KB होता)



→ प्रत्येक बट्टे diff. एवं 2¹⁶ करते।

→ प्रथम जाते।

→ TOP (TOP element को point करा
(प्रथम add से बढ़े जाए)

Stack pointer → SS:SP

↓
Segment add.

offset add.

NOW SP 100h

4¹⁶
Stack bit वर जाते push & pop करते पात्र,

PUSH AX

AX 1234h

Stack pointer decrementation

→ Dec. SP

→ push the values to location SS:SP

PUSHF → Flag Register को push करते।

→ Push the contents of the flag register

Flag Register → 16 bit

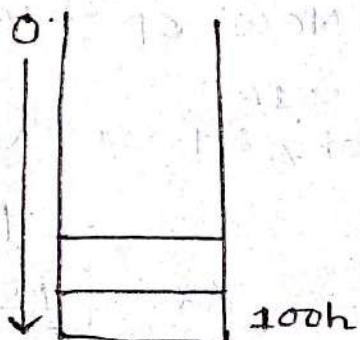
Usefulness:

Computer use करते हों, वहन पर्याप्त जाटे Statement, Run करते, एकदो फंक्शन करते और आखें फंक्शन use करते, उनका flag register मूलाना store करते।

POP

POP destination BX

जब उनको घटेना होय, आगे value होवे करते then increment होवे।



String Reverse

push and pop करते,

MOV AH, 1

MOV CX, 0

TAKE:

INT 21h

CMP AL, ODH

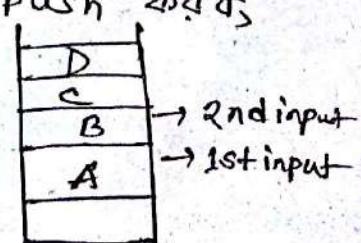
JMP NEXT - PROCESSING

यदि ना हम उन एक stack v push करते,

PUSH AX

JMP TAKE

INC CX



Now need to pop :-

MOV AH, 2
PRINT -

POP DX

INT 21h

JMP PRINT

LOOP PRINT -

push कराब अब य कम्यने value आजडेते ता cx ~ गणा
परें ex ~ value अनुआदे उसे POP रहे।

procedure or function

MAIN PROC

name PROC

→ function → name

RET

END Name

= O =

my-proc PROC

Ret

End my-proc

ज्यूटे IP रो नाकरवे

0123h

then

0125h

then

0000h

MAIN

0121h ←

0123h ←

0125h ←

5557h ←

CALL my-proc

— — —

— — —

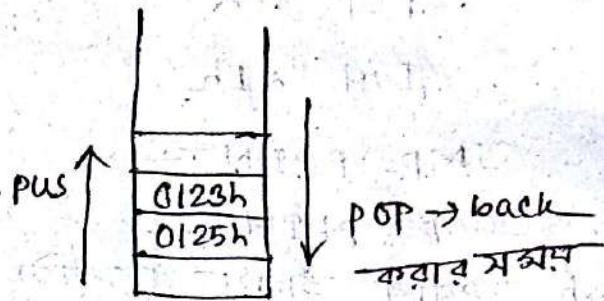
— — —

my-proc

କିନ୍ତୁ ପରି ଯଥିର ରି back କରିବ, ତଥାର ଲୋ ଆବଶ୍ୟକ
ନାହାଏ 0125h. ଫଳେ ଓଧନ stack ରୁ ମାତ୍ର ଯା
necessity ନାହାଏ।

Now,

Adding two value:-



MAIN PROC

MOV AX, 2h

MOV BX, 3h

CALL SUM

Main Endp

SUM PROC

ADD AX, BX

RET

SUM ENDP

END Main

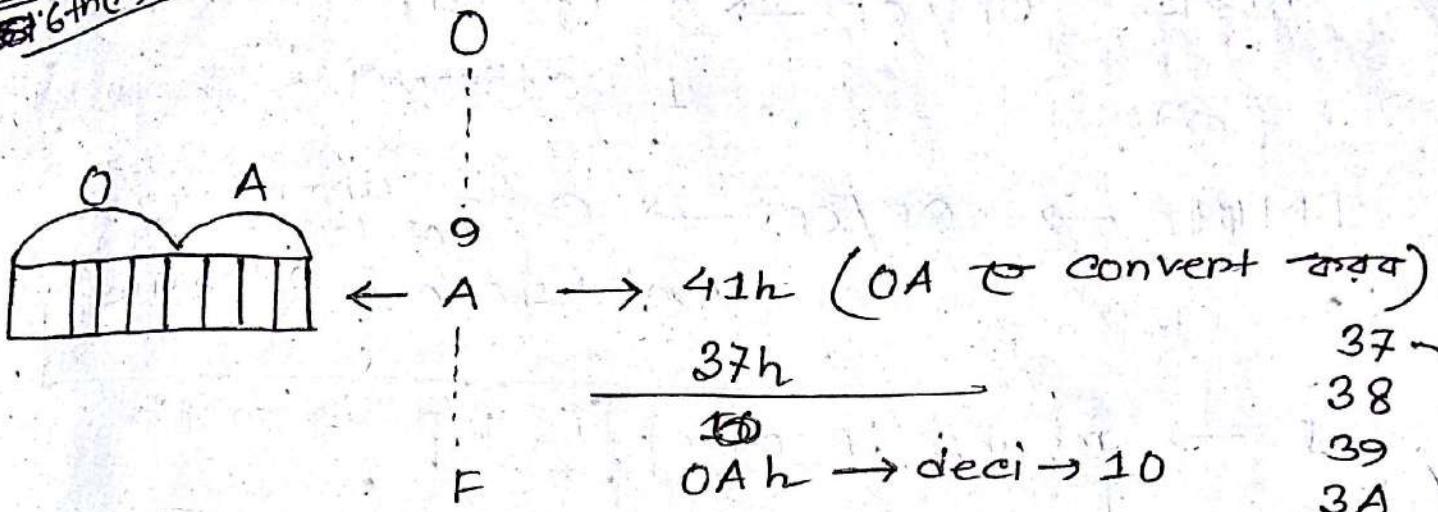
[PCB]

ଏକି ଆଜ୍ୟ ଅନୁକୂଳ୍ୟ ପ୍ରୋଗ୍ରାମ୍ ରୁ କାହାରେ
କୋଟି ଆଜ୍ୟ Run କରିବାକୁ କାହାରେ କେବଳକୁ
stack ସମ୍ପଦିତ ।

Lab:4

1. Take two 8 bit binary numbers from user
2. Add the numbers & put the result into a 16 bit space
3. Show the numbers & their sum

16-07-17
6th(A)day



এখন এই 0Ah টি 8 bit রে save
করলে কোথা যাবে।

37
38
39
3A
B
C
D
E
F
40
41

Multiplication

Signed:- ~~MUL AL / MUL AX~~ source

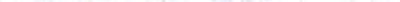
unsigned:- ~~IMUL AL / IMUL AX~~ source

→ Integer multiplication

জ্যাল একটি Operand আকর্ত্তব্য AL/AX ন

জারুণ্ডো → source MUL

8 bit রে result আকর্ত্তব্য 16 bit রে AX রে Result

Signed, Unsigned 

36

Source 8 bit \Rightarrow Result 8 bit Ax

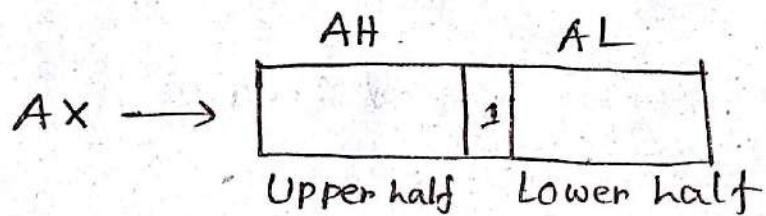
3 16 , , , , , DX : AX =

b
FLAG:-

MUL:- OF/CF \rightarrow 0 if upper half 0
 1 otherwise

IMUL → OF/CF → 0 if upper half sign
of lower
extension 1 otherwise

1 → means product is too big to fit



Lower half ଏବଂ କେବଳ bit ୨ ୦ ଥାକଣ୍ଡ ଅତିରିକ୍ଷଣୀୟ ।

ଯା ୧ ୨ ୩ ୪ ୫ ୬ ୭ ୮ ୯ ୧୦ ୧୧ ୧୨ ୧୩ ୧୪ ୧୫ ୧୬ ୧୭ ୧୮ ୧୯ ୧୩

তথ্য অন্তর্ভুক্ত sign extension.

यद्यपि AX Register ने बाधा अपने दूसरे Result के

କଥାଳ AL ଏବେ ଉପରେ ଯାଏନ୍—

$$DX, Ax = 0 \quad \text{and} \quad \boxed{Ax = 0}$$

\boxed{H} FFFF_h} ~ ইউটেলক প্রোসেসর
0010_h

$$\begin{aligned} AX &\leftarrow \text{FFFFL} \\ BX &\leftarrow 0010h \end{aligned}$$

MUL BX

'Now Integer Multiplication'

IMUL BX

$$\begin{array}{r} \text{FFFFh} \rightarrow 65537 \\ \text{0010h} \rightarrow 16 \\ \hline \text{প্রুণ করে} \rightarrow 1 \text{ lakh} \\ \text{hexa} - \underbrace{\text{00100000B}}_{\text{something}} \\ \text{DX} \cdot \text{AX} \end{array}$$

-16 ~ দ্বাদশ

$$\begin{array}{r} 0000 \quad 0000 \quad 0001 \quad 0000 \\ 1111 \quad 1111 \quad 1110 \quad 1111 \\ \hline \end{array}$$

$$\begin{array}{r} 1111 \quad 1111 \quad 1111 \quad 00000 \\ +1 \\ \hline \end{array}$$

$$\begin{array}{r} 1111 \quad 1111 \quad 1111 \quad 00000 \\ \hline \end{array}$$

$$\begin{array}{r} \text{hexa} \rightarrow \frac{\text{FFF}0}{\text{DX} \quad \text{AX}} \\ \text{Sign extension} \rightarrow \underline{\underline{23CE}} \end{array}$$

OF/CF

অন্তর্ভুক্ত OF/CF = 0

বাস্তুত মান 2 -16 represent

অন্তর্ভুক্ত

OF/CF = 0 হলে upper half এর
বাস্তুত মান

lower half ~ fit

otherwise opposite

OF/CF = 1, Lower half ने 2128 extra bit परिणाम
अथवा upper half परिणाम

= 0 =

Now

$$N \times (N-1) \times (N-2) \times \dots \times 1$$

factorial calculate - वर्षा, 51 एवं वर्षा

MOV AX, 1

AX = 2000h

MOV CX, 5

BX = 500h

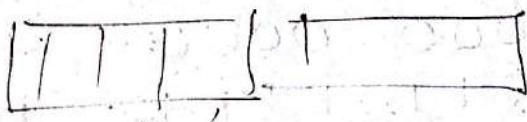
FACT :

MUL CX

MUL BX

LOOP FACT

$\boxed{BX := AX}$ CF = 1
je



17-07-17
6th (B) day

39

SAMPLE INPUT : 3592

OUTPUT : 2953

MOV CX, 4
Loop :-
PUSH

MOV AH, 1

MOV CX, 0

TAKE :

INT 21h

CMP AL, 0DH

JMP NEXT

PUSH AX

INC CX

Decimal Input

=0=

Q 235 Number কে বাইনারি এবং স্টোর করুন,

প্রথম আছে 0 then

0x10

$$\begin{array}{r} + 2 \\ \hline 2 \times 10 \end{array}$$

$$\begin{array}{r} + 34 \\ \hline \end{array}$$

$$\begin{array}{r} 23 \times 10 \\ + 54 \\ \hline \end{array}$$

$$\begin{array}{r} 235 \\ \hline \end{array}$$

একই পদ্ধতি আবশ্যিক approach

प्रारंभ 0 के store करने रखें

MOV BX, 0 ; SUM

Now Input किया,

Looping
Keep

MOV AH, 1

INT 21h

AND AL, OFH

एवं AL \leftarrow आदे 02h

PUSH AX \leftarrow 2 store करें

MOV AX, 10

MUL BX, 10

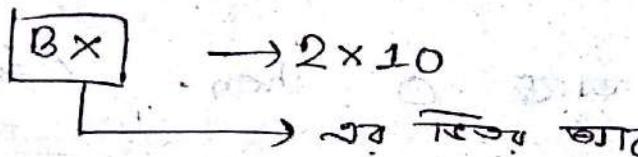
Result store 200 TAX एवं निकलें

POP BX

एवं BX एवं निकलें एवं 2

AX \rightarrow 0

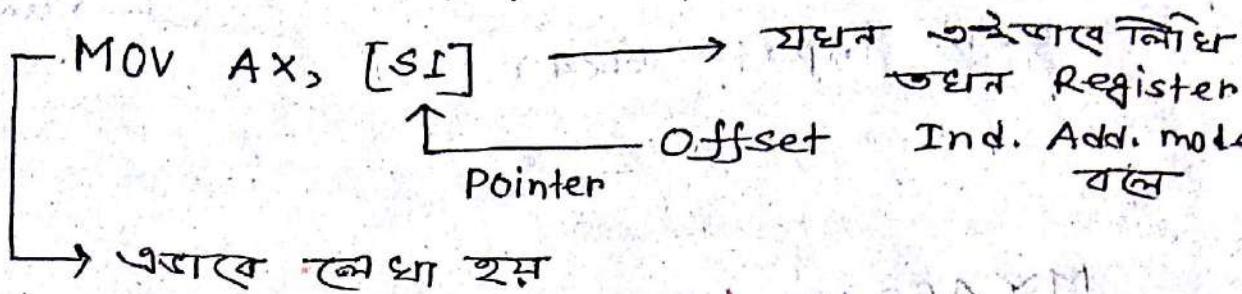
ADD BX, AX



32h	From
OF	
0,2 h	

Addressing MODE

Register Indirect Addressing MODE:-



SI এ একটি Add. আছে, DS এর Add নয় মাঝে যোগ
যোগ করে Physical Add. পাওয়া যাবে। DS: SI

$$= O =$$

$$*P = 1331h$$

0300---

যথান *P কে print দিবার অধ্যন
0300 --- কে পাওতাহ, high

Level language v 1

DS : BX / SI / DI

SS : BP

MOV AX, [SI] \rightarrow এই কাষণগাম্য যদি BP ব্যবহার
অসম্ভব DS এর সার্ফ পুরু হবে
না ফলে Phy. Add পাওয়া যাবে

ARRAY DW 10, 10, 10

এখন ARRAY কে অব্যুক্ত কিছু রাখতে চাইছ

MYARRAY DW 100 DUP (10)

Unit 1 in size. value 10, 100 কে 10

यदि NULL use करते हों तो then,

MYARRAY DW 100 DUP (?)

User input होता है।

=0=

MYARRAY DW 10, 20, 30, ——

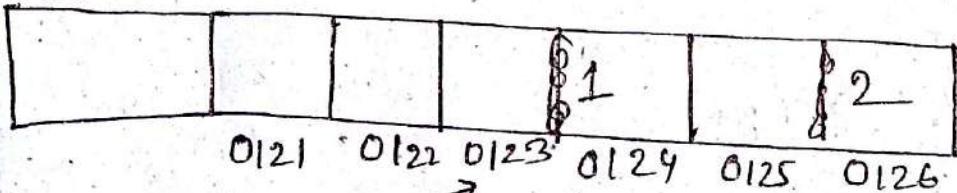
जब जो ARRAY डेटा, print किया, Register
indirect mode का होता है,
वहाँ से element (15 के element) का

MOV AX, [SI] → Load effective add.

LEA SI, MY ARRAY

→ My ARRAY का add होता है
SI → डायारी,

0000



My ARRAY 0123 को allocate करते हैं

परन्तु SI ~ जावते 0123

Add. का

~~logical bytes → physical~~

~~logical bytes → 20-bit → add.~~

off : 0000h

DS : 0000h

CO-processor:

यद्यपि processor microprocessor को help करने वाले co-processor वही है।

Register व cash एवं आर्थिक प्रणाली connect वाले co-processor ही है।

प्रोग्राम Register एवं via ट्रैम्पलेट करता, Not address तक।

ADD AX, [SI]

MOV [DI], AX

Memory to memory योग करने वाले कार्ड Register processor धूम्रधूम्र fast, Memory धूम्रधूम्र slow. Processor परिवर्तन करने का Register वही है, परिवर्तनी instruction एवं यादी। (लेट) memory to memory यह वाले adding वाले operations हैं।

8086 → 8087

processor Math coprocessor

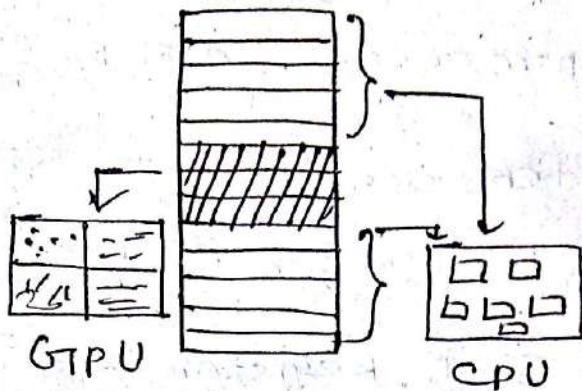
80186 → 80187

processor CO processor

अनेक CO processor आदि, Graphics, Math etc. CO-processor, Main processor या Precessor जैसे Graphics CO-processor द्वारा। ऐसे वज्ञा हैं GPU जैसा Graphics processing Unit.

एवं parallel computing द्वे वला इस GPU accelerated computing

CPU → RAM, processor,
Cache memory, Register
आरूप ।



GPU नव आरूप memory आरूप, fast रजिस्टर वब;
विशेष memory विशेष fast रहे।

Core 7 और core आरूप जहे Generation change
इस, shared memory, shared cache, large
7 shared memory आरूप।

तो कैसे यह प्राप्त होता है?

जिस CPU द्वे 6 से core/आरूप/प्रिमिय
core आरूप एकत्रित करता है

Shared cache → L₁, L₂, L₃

L₁, L₂, L₃ search किसे पड़ता है

" , " , " के लिए एवं प्रस्तुतीनीमां → Assignment

GPU नव लॉजिकों द्वारा core, GPU

6 और 7 आरूप 8 नव core आरूप आरूप CPU

→ 7 के core आरूप, फले GPU एवं
fast

State of the art GPU
Units & their total no. of core

→ Assignment

31-07-2017
7th (B) day

CT-2 ques: Analysis

□ Conditional Jump:-

Example:-

JC Point

----- }

Point

126B cover करे जाने Point
→ यह पार्ट ना, उड़े यह

c. Jump की limitation. यह
unconditional jump ~ होते,

□ even - odd check:-

Bx.

0	0	0	0	1	0	1
---	---	---	---	---	---	---

2 2 2 0

0 0 0 1 0 1

And, Test करने के लिए use करे यह पार्ट

□ AX का मूल 1 आए :-

~~Next~~: MOV CX, 16

SHR AX, 1

JC Count

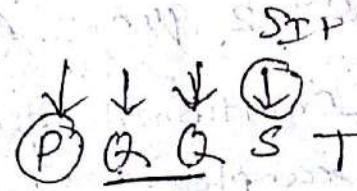
loop ~~jmp~~ next

Count :-

inc DX

Start print 40*40

PRINT2:
~~MOV *BX, 40~~
 MOV BX, 0
 MOV AH, 2
 MOV DL, '*'
 PRINT:
 INT 21h



INC BX

~~JMP PRINT~~

CMP BX, 39

JNE Exit

JMP PRINT

PRINT New line

loop PRINT2

02-08-17
7th (d) day

Registers Indirect Addressing Mode:-



एधान या गाधवात

Point करते।

DS: SI → DS एवं आर्थically SI use 25

R	U	E	T	B	D
---	---	---	---	---	---

→ Array एवं वार्ड
W (DB type)

एधन द्य एवं point किए।

LEA SI, W ; SI एवं इसका W एवं offset
add. हो।

$[SI] \leftarrow$ w जू अधिक घटक value point
करते

47

$DS \rightarrow$ Segment }
 $SI \rightarrow$ Offset } जो हुए सिल्स add. होता है
 $MOV AH, 2$
 $MOV DL, [SI]$
 $INT 21h$
 $INC SI$

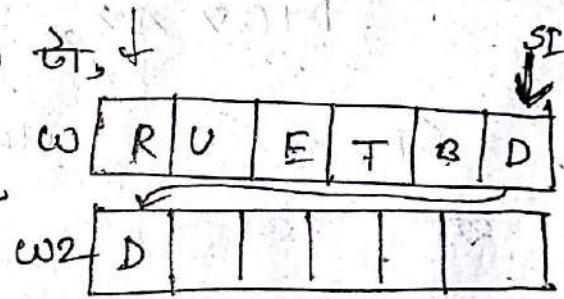
} loop वाला

User input देखाने के लिए:-

$LEA SI, W$
 $INT 21h$
 $MOV [SI], AL$

एधन Reverse print कर दो तो,

Reverse करने w2 के माध्यम से,



$LEA SI, W$

$LEA DI, W2$

$ADD SI, 5$

$MOV CX, 6$

Next:

$XCHG [DI], [SI]$

$INC DI$

$DEC SI$

Loop Next

$MOV AX, [SI]$

$MOV [DI], AX$

49

Based and Indexed addressing mode:-

Offset address obtained by adding displacement

दिनांक
प्रतिक्रिया
नियम

दिनांक
प्रतिक्रिया
नियम

दिनांक
प्रतिक्रिया
नियम

Registers वाले limitation वाले

$\underbrace{BX, SI, DI}_{DS}$ SS: BP

यह करि, SI पर स्थित हो जाए

MOV AX, $[W + SI]_{BX}$ CO [a | b | c | d | e]

MOV AX, [W, 2]

SI, DI के displacement परिवर्त्य use करा
एवं एकत्र।

lower case के upper case बदल आज़ाद

अनुवास अट्री रखें।

$$SI = 000$$

$$a = 61h$$

MOV AX, W[SI]

$$A = 41h$$

SUB AX, 20h

MOV W[SI], AX

INC SI

49

0	\rightarrow	00
1	\rightarrow	01
2	\rightarrow	10
3	\rightarrow	11
4	\rightarrow	100
5	\rightarrow	101
6	\rightarrow	110

Another way:-

$$SI = 0$$

MOV AX, 0 [SI]

$$61h \rightarrow 01100001$$

$$\begin{array}{r} MUSK \rightarrow 01001111 \\ \hline AND \rightarrow 0100\ 0001 \end{array}$$

SUB- अनुकूल gate का तो नाहीं AND gate येतेहे 0# gate use येतेहे अहो। फले AND use करा better. faster यस्ता। SUB नव एकान्वन ट्रिप्लि आन्सा काढ करो। $2^6 \rightarrow 6^2$

$$\begin{array}{r} 01100011 \\ \hline 01001111 \\ \hline 01000010 \end{array}$$

06-08-2017
8th (A) day

MOV [BX], AX

येथाले type ठें confusion घालत type ठें declare करते दिले पावत। अर्थात PTR PTR declare करत, नेहार form ठें इस,

Type PTR DATA / Memory

प्रश्न:-

MOV BYTE PTR [BX], 1

; नेहार BX ठें byte
किंवा आज करते।

MOV WORD PTR [BX], 1

= 0 =

LABEL :-

HOUR DB 10	MOV AX, HOUR
MIN DB 5	

Confusion ~ झगड़े

AL ना AH न हो याहे HOUR

Confusion ~ एवाचार छ असे,

MOV WORD PTR AX, HOUR

LABEL क्षेत्र वाले structure:-

TIME LABEL WORD ; Time की word type

HOUR DB 10

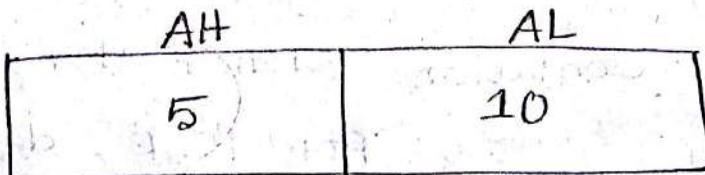
MIN DB 5

यहे प्रत्येक byte किंवा

word को होते हैं

MOV AX, HOUR

MOV AX, TIME



अे operation

को प्रत्येक word

करते

पहले AL fill up करते हैं तो तब then AH

यद्यपि प्रत्येक byte को प्रकारे प्रत्यागत, प्रथम % time को define करते हैं बिना हो सकते प्रत्यागत।

Based Indexed Addressing :-

BX, BP

यथा: MOV AX, W [BX]

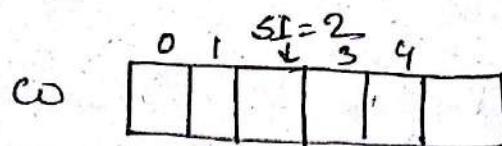
→ Based Addressing mode

SI, DI

यथा: MOV AX, W [SI]

→ Indexed Addressing mode

अपेक्षित दृष्टिकोण mix करते पाएँ, based Indexed Addressing



→ यहाँ रैम का उपकरण दृष्टिकोण
में अपेक्षित हो जाएगा।
use करना है

यदि two dimensional Array है तो, तो,

W

AL
R

w+2	w+4	w+6	w+8	w+10
w+12	w+14	w+16	w+18	w+20
R	X	X	X	X

W D W - - -

→ यहाँ position से fill
up करते वाला है?

Based Indexed Addressing की धार्या निम्नलिखित :-

(i) variable [base register] [Index register]

(ii) [Base register + Index register + variable constant]

variable [base + Index + constant]

(i) यह आवश्यक use करें।

Now अपेक्षित POS XOR SP, SL Answer निम्नलिखित :-
Next:- INT 21h MOV BX, 20 ↑
MOV W [BX] [SI], AX ↑ column

ADD SP, 2

3 नंबर पॉवर्ट जाएगा तो ?

92

$$(POCO-1) \times \text{column} \times \text{size} = (3-1) \times 5 \times 2$$

= 0 =

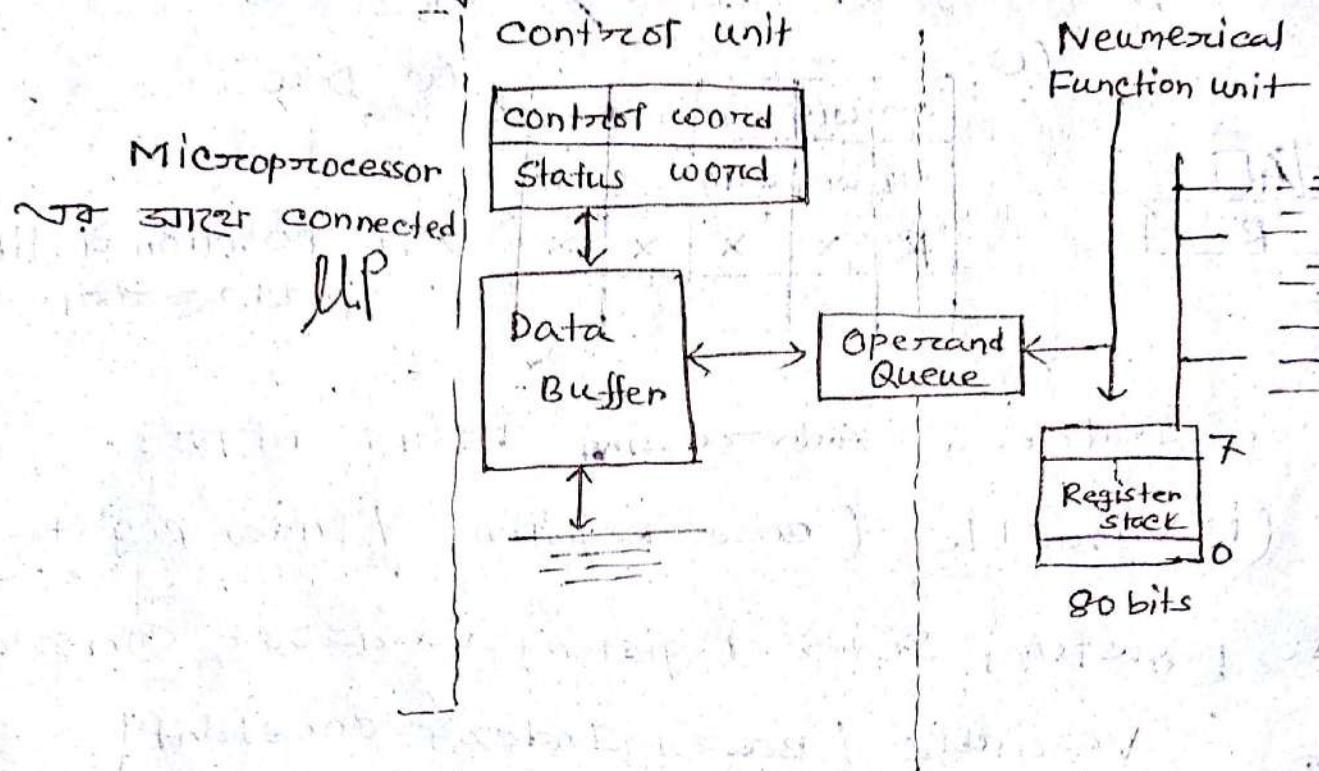
MOV AX, W[BX]; वर्ड प्रिंट कराएँगे

फुटेले matrix यह वा योग्य रूप नहीं आएगा।

07-08-2017
8th (B) Day

CO-processor वर इस तरह विषय प्रश्न निम्न है:-

Block diagram of 8087



Complete figure क्या होता ?

BOOK: Intel Microprocessor

by BEE Brey

Status register:-

0	B	C3	(ST)	C2	C1	C0	E	P	F	U	O	Z	E	DE	FE
1															

Busy: Busy = 0 स्टैक खाली/यादि $\neq 0$ - प्रोसेसर
0 खाली

Busy: co-processor free स्टैक खाली

$B = 0$, busy नाकाली $B = 1$

ST: Indicates the current register addressed as the top of the stack.

	C3	C2	C1	C0	Conditions J.I.
	0	0	0	0	ST > Operand JA
X	0	0	1	0	ST < Operand JB
	1	0	0	0	ST = Operand JE

ES: PE, DE, OE, ZE, DE, IE यादि
एक रोमांच आको यादि 1 यादि then ES=1
Otherwise ES=0

PE: when the selected operand exceeds the precision.

VE: Indicates a ~~result~~ non zero result which is too small to represent.

OE: Indicates a result that is too big to represent.

IE: Indicates a stack overflow or underflow, the use of a NaN as an operand.

ZE: when the divisor was 0 and the dividend is non zero or infinity.

D Denormalized error: Indicates that one operand is denormalized.

control unit द्वारा Microprocessor ने आत्म interfacing करता है। Micro. एक data control unit व संबंधित करता है। Micro. जो इन आर्थ द्वारा BUS आए जाते, data flow करता है।

Data buffer, status word द्वारा data operand Queue करते हैं।

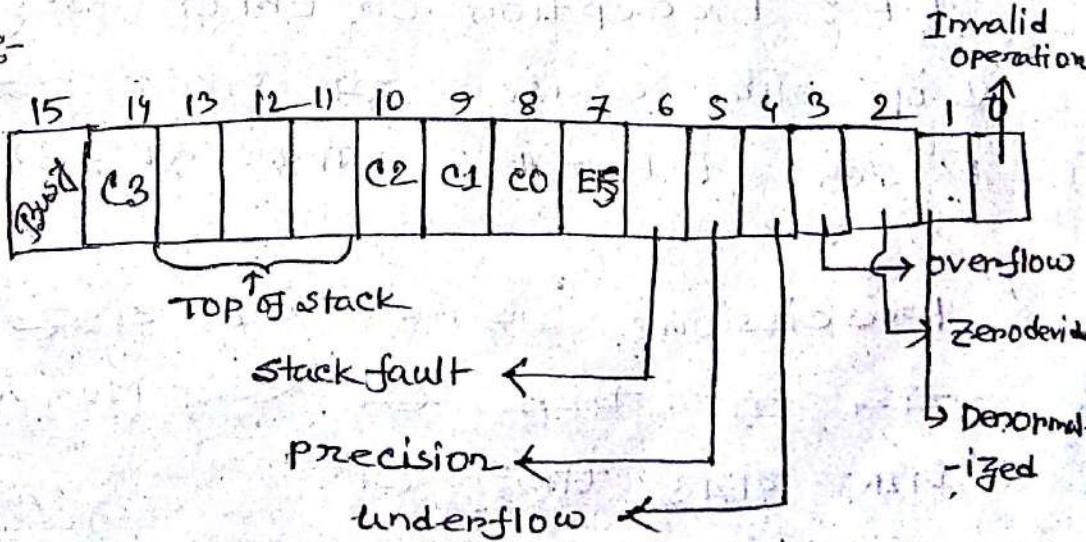
data व Main execution operation तथा Num. function unit व Register stack वाले 80 bits. 8 टो register वे Operand size वाले, एवं Operand Queue में सहायता help करते हैं।

Operand Queue द्वारा data buffer द्वारा Register Stack or Numerical function unit व Queue में कार्रा करते हैं।

=0=

Status Register:-

8087



FSTSW

Condition code

(C0-C3) : NAN, Denormal

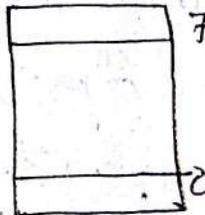
Busy% का co-processor एवं कार्ड मुला लेबल देख
किना, free कार्ड = 1, आवे busy कार्ड 0

BUSY = 0 ~~means~~ Active, BUSY = 1 ~~means~~ Not Active

Co - C3 : Operation ଯେଣା ପ୍ରକାର ଦ୍ୟାନ NAN ଆହୁ
Output, (not a number), Co - C3 କେ କୋନ
bit storing କାହାକା ଭବନ, 16 bit ଏବଂ 32-

Denormal: यदि होन exponent part ने -235
 पर कल्प रूपने रहे then denormal. 5^{-235} , negative
 number एवं असाधु होन Number एवं यादे exponential आणहे
 आकर्षे एवं डिटोर यादे डावाहूच Number एवं आकर्षे ऑटी com" रो deformat
TOP of the stack: वाचन रुपे Number दे आणु

जे denote करा. ये एक त्रिं bit होता,
 Stack एवं top ~ आठवें Number टो



EF: Exception or Error flag, ପରେବ ଝଲାସ

ବୁଝେ ଯାଏ ଅର୍ଥ ହସ୍ତ ପରିବାର ଦ୍ୱାରା କୌଣସି
 1 ଟଙ୍କା $E F = 1$, ଆବଶ୍ୟକ କାମ ଟଙ୍କା $E F = 0$

precision: अनुमान दर्शाते होते से यहि
 एक अंक तभी तभी अंक एवं अंक
 अंक तभी तभी तभी अंक एवं अंक

Stack faults: Stack operation overflow or underflow.

Underflow: Memory એ નાટે કિન્તુ Access કરાણ જાઓ

Overflow: " এ কান্দা নাই ভাই Access করলু চাই"

Invalid O/P: 8 bit यादि 16 bit \rightarrow गँधारु जै आउल \rightarrow विषय तो
arise करवते।

55

FSTSW:-

CO-processor एवं instruction F प्रिम्य शुक या। यादि
ADD करें, then ~~इसे~~ FADD (F means 'float')
zero dividable; check करते zero devide रखें किना
8 bit \sim हेक्साडेसिमल आए

एधाल, 2 bit से ऊरु check कराव नहीं

00000000 00.00 0~~4~~00
00 04 h

00 04 h छाग mask करवते

COS Processor व check कराव नहीं,

FSTSW AX

TEST AX, 0004h

Then Jmp condition / JZ print-zero

then इसे decision किये divide by zero रखें किना

ज रहें।

■ CPU एवं कानून कानूनाना अर्थात् यात्रा faster कराव नहीं

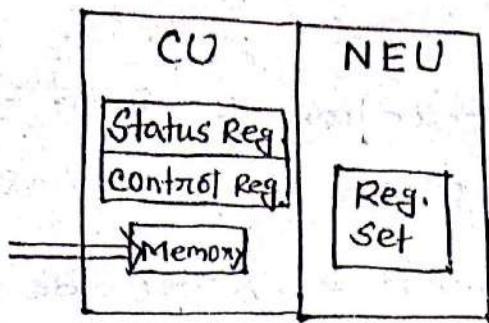
CO-processor युवत्ता करा यस। प्रयोग: GPU एवं

विषयन का CO-P

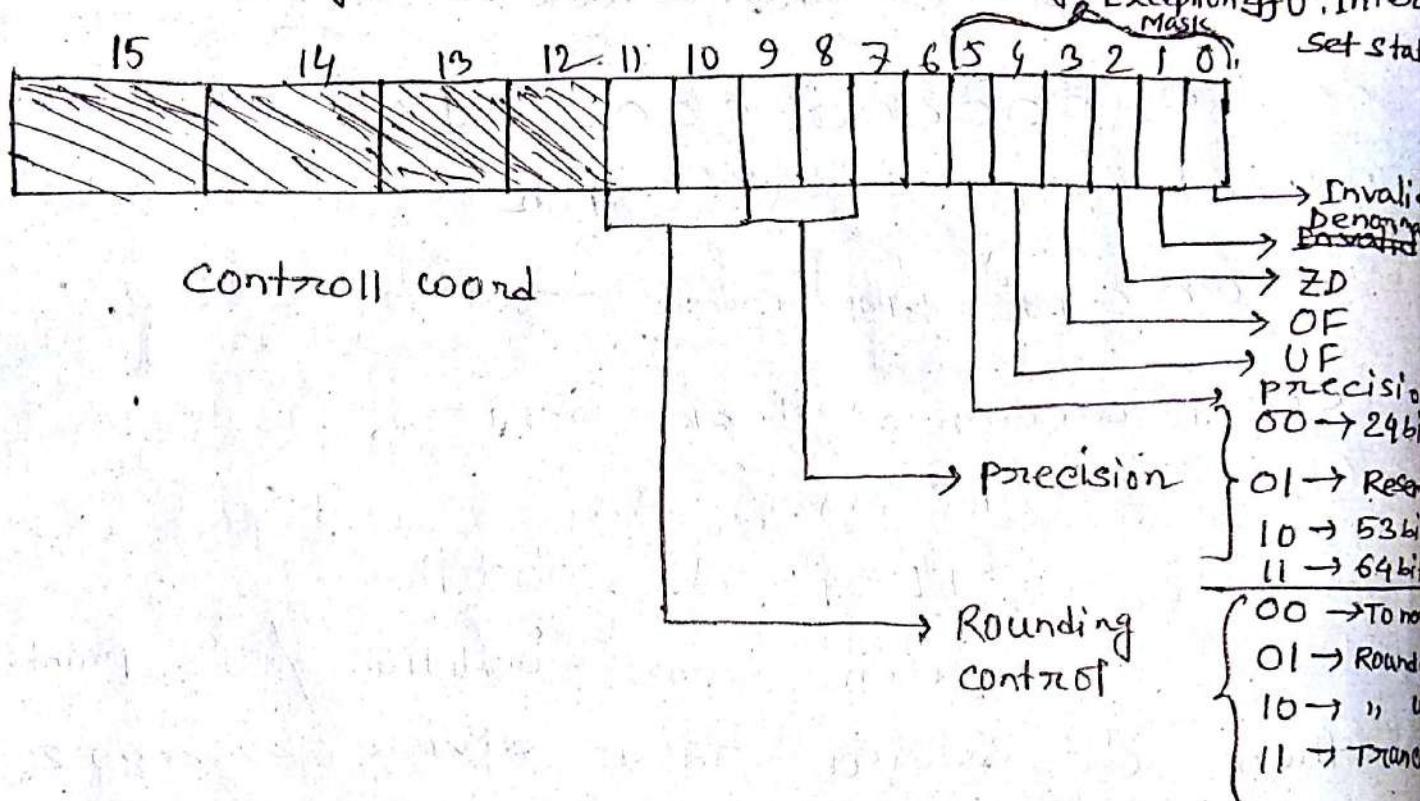
Operand queue: ये instruction लागते जाते हैं और उनके बारे में
एक operand उनमें store करते रहते हैं ऐसी operation
जो continue करते रहते।

09-08-17
8th (D) day

52



Control Reg:- 16 bit ഏ ഫോർമേറ്റ് memory, Exception 4f0: Inter



12-15 वरुणा unused

Rounding control :-

2.0346

Round up \rightarrow 2.04

Round down $\rightarrow 2.02$

Titanate \rightarrow 2.034

precision:

IEEE standard for representing floating point No.

Single precision / Double precision तरीका,

24 bit \rightarrow Single precision

~~64 bit~~ \rightarrow Double \rightarrow (10) line

6, 7 \rightarrow unused

Status word \sim 0/1 रखते हैं वह नियन्त्रण नहीं।
Control word नहीं रखते हैं।

Control word में ~~if~~ Overflow = 1

then ignore it

यदि 0, तो, Overflow \rightarrow 1000

then Status - में Overflow \rightarrow 1000

हमें 1, error \rightarrow 1 और interrupt रखते हैं।

प्राप्त करते हैं। then stop it.

Lab work

19-08-2017
9th (c) day

- Take input character from user
- sort the input and display at the next line
- Ignore duplicates

P
PQT
PQTS
PQSAT

\rightarrow PQSAT

22-08-17
10th (A) day

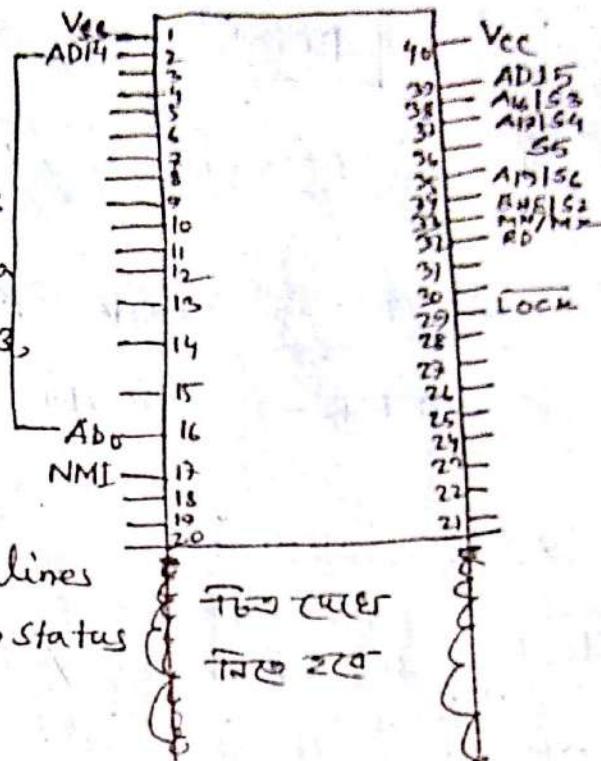
53

Signal/PIN description of 8086 :-

PIN शैलास कारणः-

$AD_0 - AD_{15}$: Address remains on the lines during T1. Data is on the lines during T2, T3, T4 & T5.

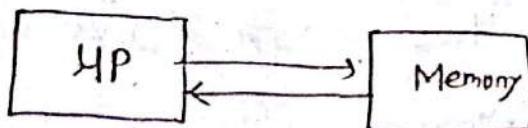
$A_{19}/S_6 - A_{16}/S_3$: The most significant address lines during T1. During I/O \rightarrow Status information.



$S_5 \rightarrow$ Status of Interrupt enable flag bit.

$S_6 \rightarrow$ When low indicates 8086 is in control of bus.

8086 कम 16 bit नहीं आवश्यक $AD_0 - AD_{15}$ पर्याप्त 16 bit line -



Microprocessor द्वारा data memory के नियंत्रण के address नियंत्रण, memory द्वारा MP के आवाहन के address द्वारा नियंत्रण।

* Tri-state

T₁, T₂, T₃ | T₄, T₅ \rightarrow एक 5 दो लिंग cycle हैं।

T₃ पर्याप्त Address read हैं। memory प्रॉटोकॉल, then memory द्वारा data MP द्वारा (16 bit) . एक नियंत्रण

data read कराने अथवा read cycle

data write कराने अथवा Write cycle

$A16/S6 \rightarrow A19/S5$ वाले जनवरी आआदेश एवं यहां पर्याप्त। S_3-S_6

सिस्टम status denote करते।

S_4	S_3	Indication
0	0	Extra segment
0	1	Stack segment
1	0	Code segment
1	1	Data segment

BHE/S7: When enabled it denotes the data transfer over higher order ($D_8 - D_{15}$) data bus.

RD - Read: Indicates that the MP is performing Read operation.

READY: Acknowledgement from the slow device or memory.

$D_8 - D_{15}$ अवलम्बन

NMI \rightarrow Non Maskable interrupt

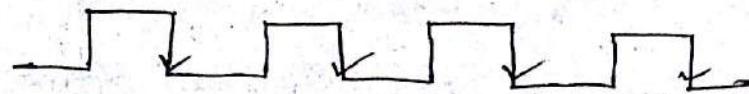
Hard interrupt किसकारी: Hardware interrupt
Software interrupt NMI

NMI pin द्वारा interrupt process करने वाला -

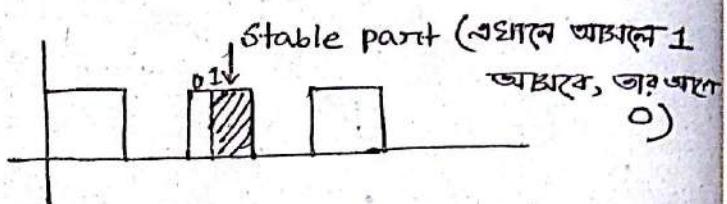
अब काउंटर द्वारा अथवा process को पूरा करने वाला -
प्रोग्राम shut down operation.

INTR → Interrupt request

CLK → clock



Read →



ALE → Address Latch Enable (अर्कने लाईलै परेंट
address^{जागा} address से जागे किना तो confirm करते)

DT/R → data + transmit हुए तो, receive करते
जो denote करते।

DEN → Data bus Enable करते, data line enable

M/IO → Memory या IO device हुए तो confirm करते

WR → Write या काढ़े किए गए हुए किना प
confirm करते।

TEST

$\overline{RQ} / \overline{GTO}$ → New bus Master, request-प्राप्ति

$\overleftarrow{RQ} / \overleftarrow{GTL}$ → Old bus Master → Access
करते तो Grant

Bus Master: ये Bus जोड़ने control करते।

BUS cycle indication

٦١

S_0	S_1	S_2	
0	0	0	Interrupt Acknowledge
0	0	1	Read I/O
0	1	0	Write I/O
0	1	1	Halt Halt
1	0	0	Halt Read
1	0	1	Read
1	1	0	Write
1	1	1	Passive state

- (i) ~~WPS~~ Lecture.
 - (ii) I/O device ~ read
 - (iii) " " " " write

- (iv) Memory Error read
Write

Microprocessor का धरते होते थारे ना, काले कृष्ण तीया यहां
यद्यपि वे काले करते ना उधर NOP करते, न किसी भैंस के
Operation (No operation), Activated थारे वर
थारम् ।

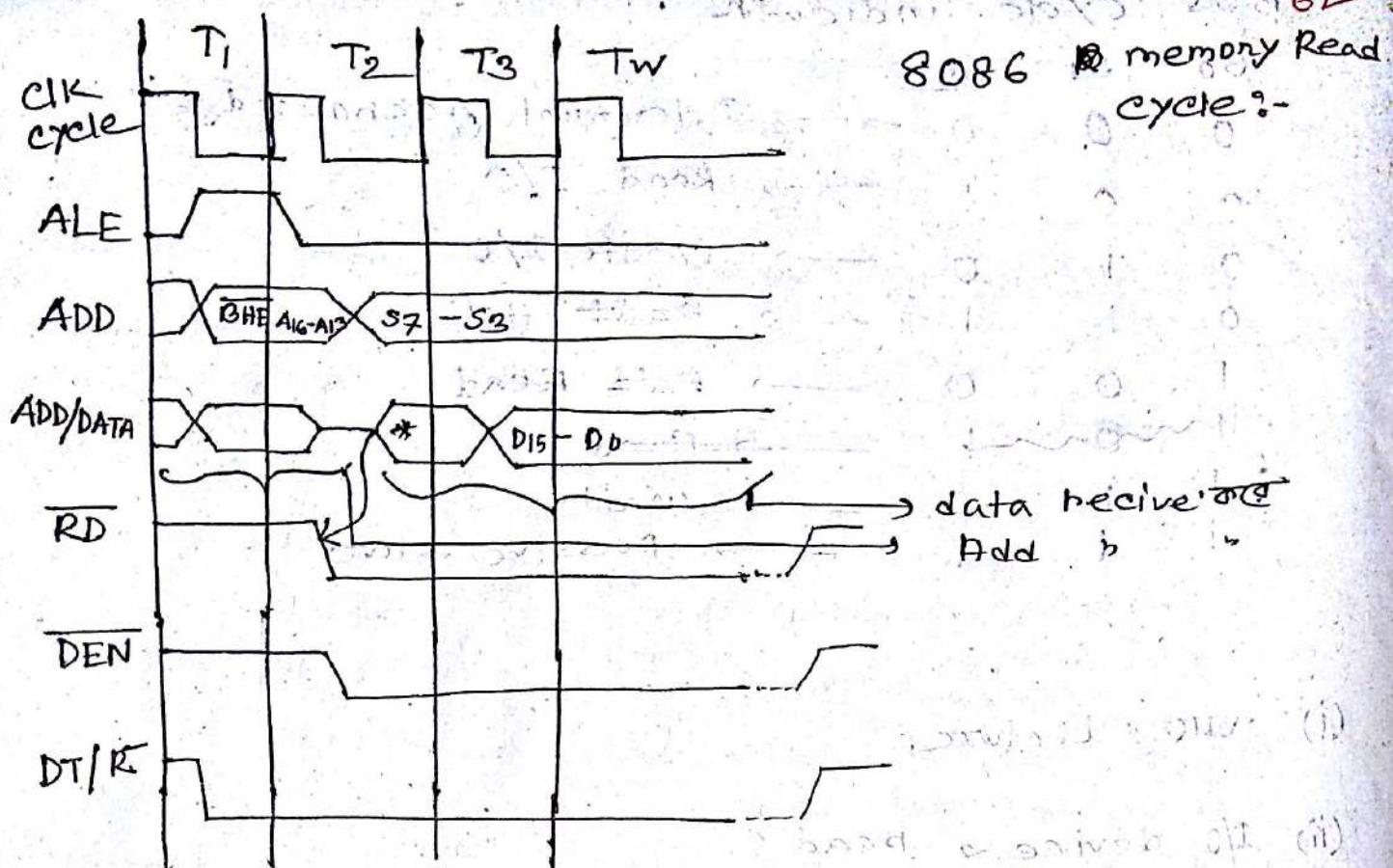
RD → Read operation

DEN → Availability of data (data enabled)

DTR → Transmit or receive

62

8086 memory Read cycle :-



* Bus Reserved for DT / R- data

अधिक Add. इलान ADD के थाकरे एवं ADD/DATA
के बाध्यते, ADD एवं line इला 0/1 की combination
करके तारें। अपने Address latches enable
करते। अपने memory के Read रद्द, RD
के enable करते, अधिकenable करते, यहाँ
bus reserved for data. DT/R- को Read

करते। ADD / DATA के D₁₅ - D₀ के data
place करते पर, Read करते हैं, the
Data read हो। Read करने के लिए \overline{DEN}
DT/R- के signal के लिए change करते।

⇒ Memory write cycle (निर्माण घटक प्रक्रिया) 63

26-08-17
10th (C) day

DF → Dual Direction flag, direction determining
करते। Status register में 21वें बिट।

CLD → clear direction flag, direction flag
 $DF = 0$ करते हैं।

STD → Set DF = 1 करते हैं ताकि दूसरी ओर
MOVSB ES: DI DS: SI

Memory to memory

MOVSB → एकले बाटे कोप करते।

ES: DI DS: SI

DS: SI से ES: DI कोप करते।

SI →

R	U	E	T
---	---	---	---

MOVSB एकलाइनर copy

एक SI से DI को

DI →

एवं SI व DI को update

R	U	E	T
---	---	---	---

करते।

Left to right SI व DI यारे नाही, right to left यारे।

DF = 1 ताकि Right to left हो

DF = 0 → left → right →

CLD एवं सार्वत्रिक STD फिल्स

MSG1 द्वारा MSG2 को किसे

SI

MSG1 DB 'RUET'

MSG1

R	U	E	T
---	---	---	---

MSG2 DB 4 DUP(?)

DI

--	--	--	--

LEA SI, MSG1

LEA DI, MSG2

CLD

Direction flag 0 जैसे बदलने का तरीका

MOVSB

MOVSB

MOVSB

MOVSB

→ Another way

REP MOVSB

then MOV CX, 4 करने वाले बदलने का तरीका

इसे छोड़ते हैं।

MOVSW → word type जैसे बदलने का उपयोग।
इसे बदले करने के लिए move इसे।

जटिल Reverse करने की ट्रिक

MSG1 DB 'RUET'

1	2	3	4
R	U	E	T

MSG2 DB 4 DUP(?)

1	2	3	4
T			

LEA SI, MSG1

0	1	2	3	4
T				

LEA DL, MSG2

1	2	3	4
T			

ADD SI, 4

DI + 2	1	2	3	4

STD → Direction Flag 1 जैसे बदलने का तरीका

बदलने का तरीका

MOVSB

ADD DI, 2

(2 instruction जैसे बदलने का तरीका DI 0 से शुरू होने का तरीका)

MOVSB

ADD DI, 2

MOVSB

ADD DI, 2

MOVSB

ADD DI, 2

एक बार word type वाला (H.W)

= 0 =

STOSB → AL को फिर स्टोर करें ताकि ES:DI
store करते।

input अवश्यक AL थाएं ताकि AL use करें

Reverse order → input फिर स्टोर RUET प्रिमियम

store करते,

~~STD~~

~~STOSB~~

LEA DI, MSG

ADD DI, 9

STD

MOV AH, 1

INT 2Ah

~~STD~~

STOSB

R U E T

DI

4 बार loop रहे

लॉप AL तक input ES:DI से आयते।

66

LODSB DS:SI
 AL ← }
 STOSB अे अन्त,
 DS:SI द्येते AL ~ अपना
 COPY आहो।

= 0 =

MOVSB
 LODSB
 STOSB

→ एकांक word version →

MOVSW
 LODSW
 STOSW

Marathi वा वर्णमाला

chapter - 11 अवृत्ति

DIVISION (H.W.)

29-08-17
11th(A)day

Assignment:

MOVSB

STOSB

LODSB

w



वा chapter वा programming exercise

वा एकांक वा सोडा string. (11th cycle वा lab no.)

09-09-17
11th (B) day

67

या या किनिंग पक्ष्युत्तम इत्यः-

- (i) XLAT instruction
- (ii) SCASB / w (Chapter 11)

Chapter 4 → 11 (अपार्क्ट)

Chapter 1 - 3 के पक्ष्युत्तम इत्यः।

Addressing mode :-

कठोर विवरणीय addressing mode वा असं सम्पन्न

History of Microprocessor:-

Technical फैलिशगुलाम एवं तात्त्व

8086

8088

80186

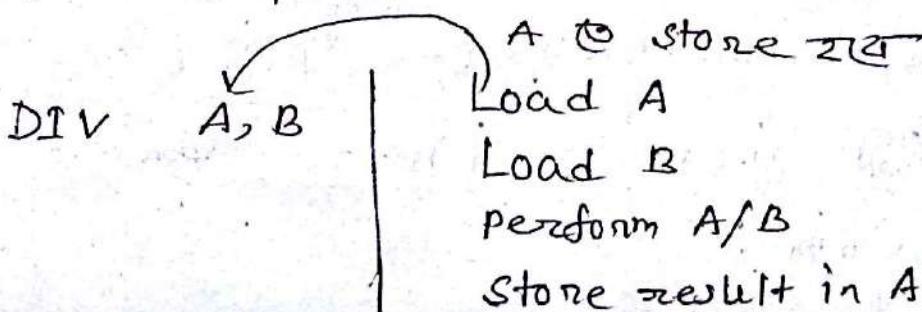
General trend युला पक्ष्युत्तम इत्यः

Generic change युलान।

= O =

i) RISC :- Reduced Instruction Set Computer

ii) CISC : Complex



एकाक्रम चारों case एको instruction लाई बाब्दामा
हुँदै, तरीको रूप CIS C

एवन्स RISC

MOV AX, A

MOV BX, B

DIV AX, BX

MOV A, AX

आठोबाटोको भूमि

(i) एकाक्रम coding काहाटो easy, Code size
योजित, memory योजित नागर्को हाल
ware design ले complex, transistor
मात्राया योजित नागर्को, एकोले flexibility जा
एकाक्रम Reapt Reapet किछु process हस्ता
न्याय राखे Explicit पाइया पाउँ।

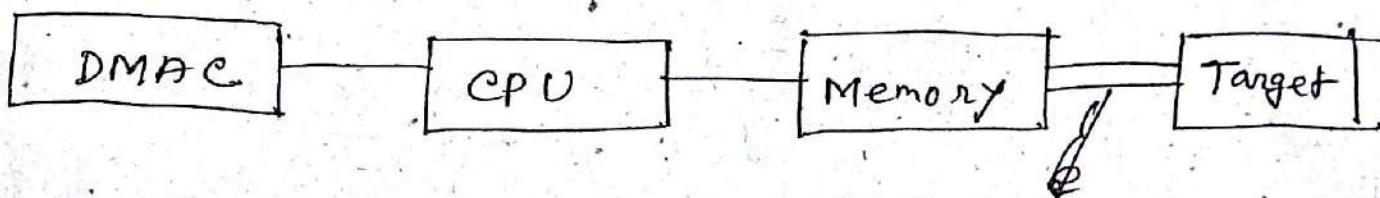
(ii) ~~ज्ञान~~ (i) लाई विपरीत,

DMA :- { Direct memory Access } (DMA)
When → Microprocessor को bypass को
data transfer लाई आयुक्त अस्त्रकोणी

- i) transparent mode :- CPU is not using system bus
- ii) Burst mode : Entire block of data transfer
at a time.

- iii) cycle stealing mode; 1 byte of data transferred at a time

DMA controller (8257) → Block diagram, pin diagram, Read, write cycle → त्रिलोक



B.

- यद्यपि CPU नाम थाकरे अपन काठ करते
- CPU नव कधा चेष्टा करते ता, data transfer आप ना उड़ा पर्याप्त काठ करते!
- 1 byte करे data transfer करे, control हाले दिये।

Microprocessor नव PIN:-

HOLD :-

HLDA :-

Interrupt vector table :- (અન્ટરપ્રોક્રેસ ટેબલ)

આવાજ મેમોરી કે ગોત્ર ફંક્શન કાર્ડ,

$f_1() \{$

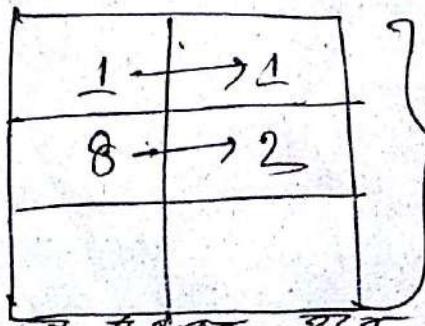
2
3
4
5
6
7
}

$f_2() \{$

9
10
11
12
13
}

એ નાં ફંક્શન કે execute કરતું રહ્યું, તેણું interrupt request આવતું, ત્થાં ~~f_2()~~ ફંક્શન, કે execute રહ્યું, તેણું જો $f_10()$ કે call કરે, તેણું $f_9()$ એવું રહ્યું રહ્યું, ત્થાં એવું એવું એનું સ્લોવન. એનું

Hashing કરતું રહ્યું।



હેચટેબલ ઉદ્દેશ્ય
કરતું faster કરાયા

71

