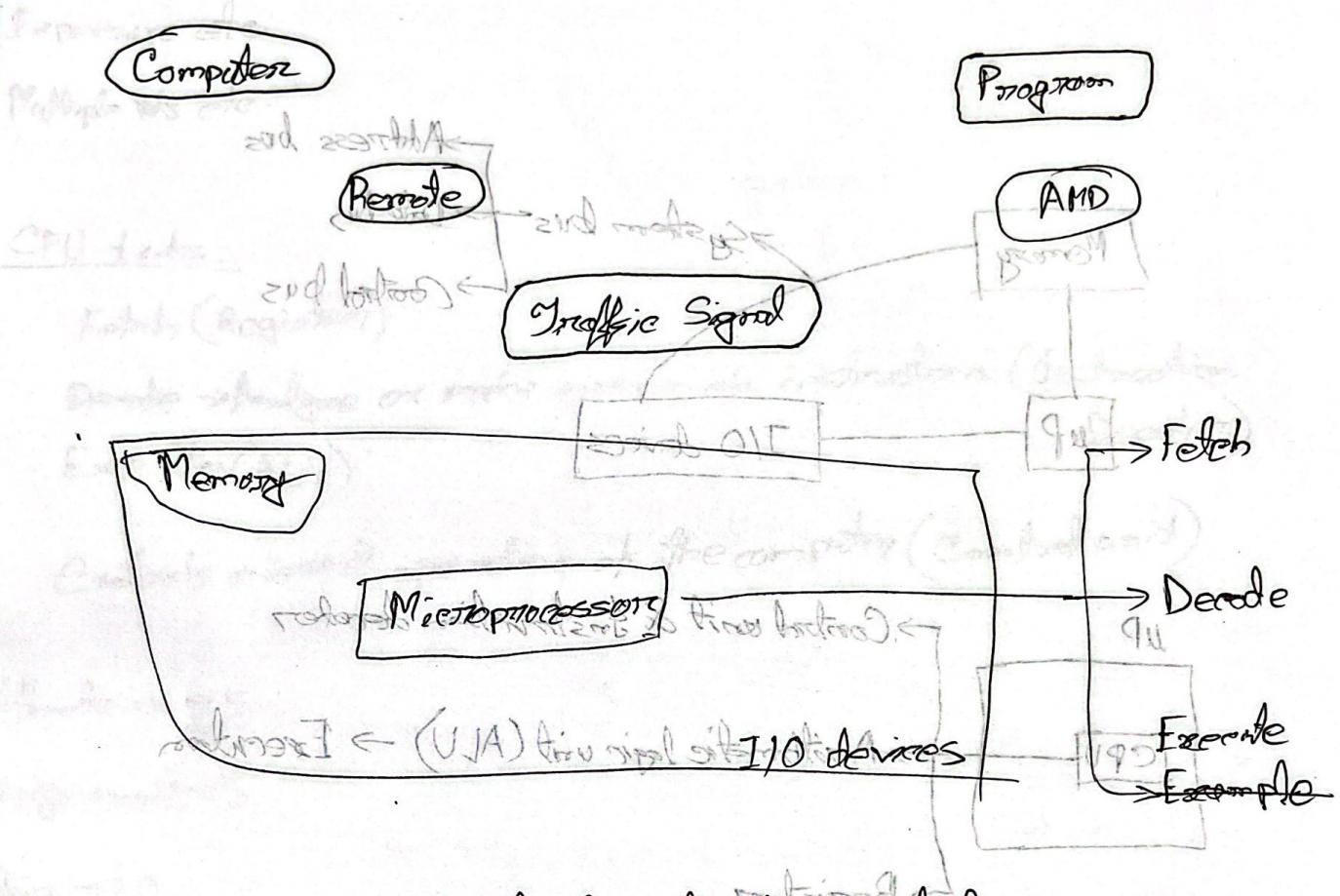


## Microprocessor



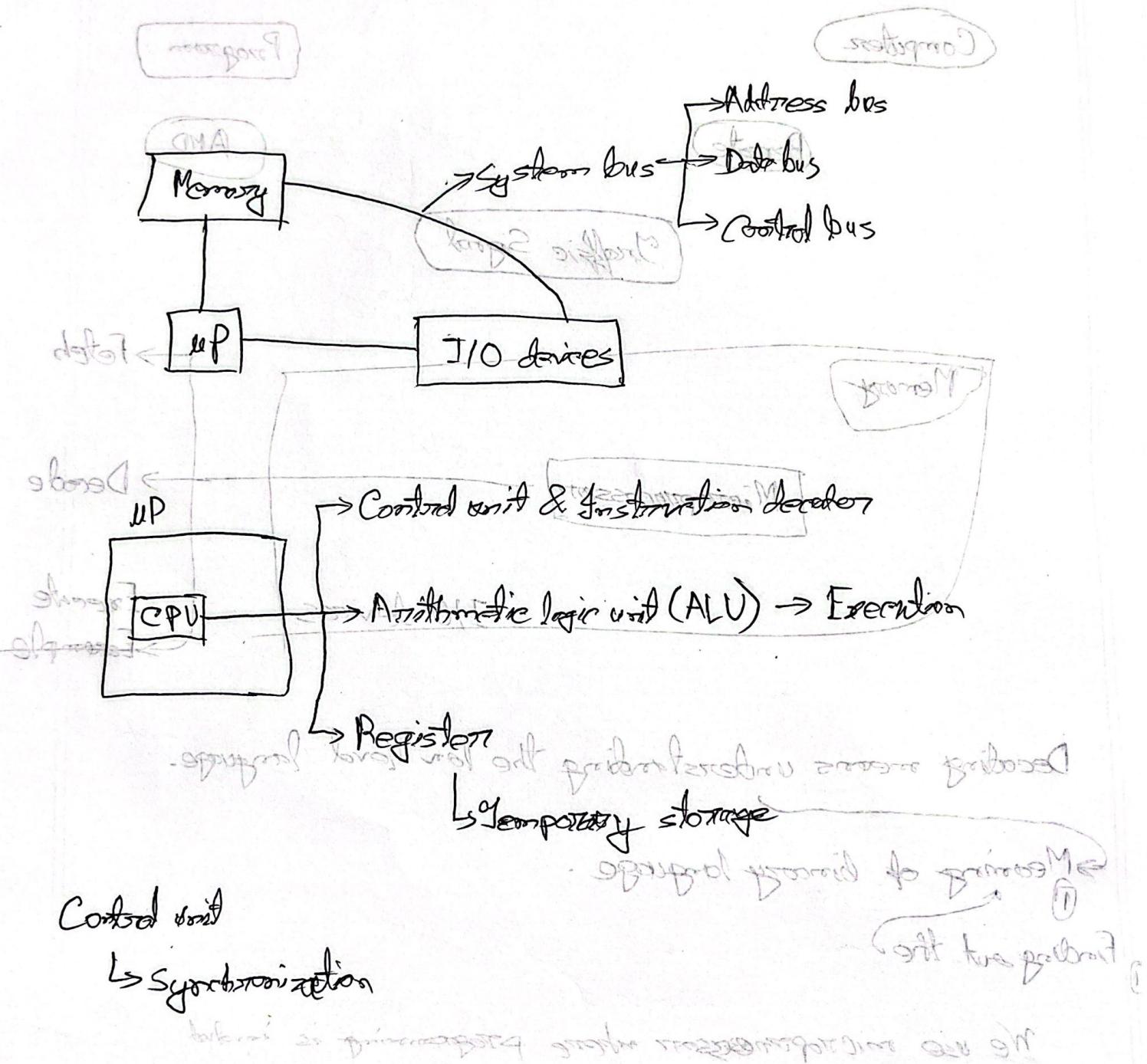
Decoding means understanding the ~~written~~ language.

## Meaning of binary language

1

## Finding out the

We use microprocessors where programming is involved.



Important

Block diagram of a Microcontroller.

Write & Read operation (Def)

CE - A30

MP vs. MC

SDC

Expositive etc.

Multiple etc.

instruction → address register



program



Instruction

CPU tasks:

Fetch (Register)

Decode → Analyze or make sense of instructions. (Instruction  
Decoder)

Execute (ALU)

Controls overall operation of the computer (Control unit)

clock

clock

latency 2  
bottom

latency 1  
middle

Attendance → 5  
How many upsets?

Assignment → 5

examples of good flow

Quiz → 10

LAB → 25

exercises were difficult

Mid → 25

primitives

Final → 30

organized

instructions

4 parts of CPU register:

tid / Program counter:  
Instruction pointer  
size system M ← size and assembly  
segment address = 4 bytes  
segment address = 8 bytes  
segment address = 16 bytes  
segment address = 32 bytes  
segment address = 64 bytes  
segment address = 128 bytes  
segment address = 256 bytes  
segment address = 512 bytes  
segment address = 1024 bytes  
segment address = 2048 bytes  
segment address = 4096 bytes  
segment address = 8192 bytes  
segment address = 16384 bytes  
segment address = 32768 bytes  
segment address = 65536 bytes  
segment address = 131072 bytes  
segment address = 262144 bytes  
segment address = 524288 bytes  
segment address = 1048576 bytes  
segment address = 2097152 bytes  
segment address = 4194304 bytes  
segment address = 8388608 bytes  
segment address = 16777216 bytes  
segment address = 33554432 bytes  
segment address = 67108864 bytes  
segment address = 134217728 bytes  
segment address = 268435456 bytes  
segment address = 536870912 bytes  
segment address = 1073741824 bytes  
segment address = 2147483648 bytes  
segment address = 4294967296 bytes  
segment address = 8589934592 bytes  
segment address = 17179869184 bytes  
segment address = 34359738368 bytes  
segment address = 68719476736 bytes  
segment address = 137438953472 bytes  
segment address = 274877906944 bytes  
segment address = 549755813888 bytes  
segment address = 1099511627776 bytes  
segment address = 2199023255552 bytes  
segment address = 4398046511104 bytes  
segment address = 8796093022208 bytes  
segment address = 17592186044416 bytes  
segment address = 35184372088832 bytes  
segment address = 70368744177664 bytes  
segment address = 140737488355328 bytes  
segment address = 281474976710656 bytes  
segment address = 562949953421312 bytes  
segment address = 1125899906842624 bytes  
segment address = 2251799813685248 bytes  
segment address = 4503599627370496 bytes  
segment address = 9007199254740992 bytes  
segment address = 18014398509481984 bytes  
segment address = 36028797018963968 bytes  
segment address = 72057594037927936 bytes  
segment address = 144115188075855872 bytes  
segment address = 288230376151711744 bytes  
segment address = 576460752303423488 bytes  
segment address = 1152921504606846976 bytes  
segment address = 2305843009213693952 bytes  
segment address = 4611686018427387904 bytes  
segment address = 9223372036854775808 bytes  
segment address = 18446744073709551616 bytes  
segment address = 36893488147419103232 bytes  
segment address = 73786976294838206464 bytes  
segment address = 147573952589676412928 bytes  
segment address = 295147905179352825856 bytes  
segment address = 590295810358705651712 bytes  
segment address = 1180591620717411303424 bytes  
segment address = 2361183241434822606848 bytes  
segment address = 4722366482869645213696 bytes  
segment address = 9444732965739290427392 bytes  
segment address = 18889465931478580854784 bytes  
segment address = 37778931862957161649568 bytes  
segment address = 75557863725914323299136 bytes  
segment address = 151115727458228646598272 bytes  
segment address = 302231454916457293196544 bytes  
segment address = 604462909832914586393088 bytes  
segment address = 1208925819665829172786176 bytes  
segment address = 2417851639331658345572352 bytes  
segment address = 4835703278663316691144704 bytes  
segment address = 9671406557326633382289408 bytes  
segment address = 19342813114653266764578816 bytes  
segment address = 38685626229306533529157632 bytes  
segment address = 77371252458613067058315264 bytes  
segment address = 154742504917226134116630528 bytes  
segment address = 309485009834452268233261056 bytes  
segment address = 618970019668904536466522112 bytes  
segment address = 1237940039337809072933044224 bytes  
segment address = 2475880078675618145866088448 bytes  
segment address = 4951760157351236291732176896 bytes  
segment address = 9903520314702472583464353792 bytes  
segment address = 19807040629404945166928715920 bytes  
segment address = 39614081258809890333857431840 bytes  
segment address = 79228162517619780667714863680 bytes  
segment address = 158456325035239561335429727360 bytes  
segment address = 316912650070479122670859454720 bytes  
segment address = 633825300140958245341718909440 bytes  
segment address = 1267650600281916490683437818880 bytes  
segment address = 2535301200563832981366875637760 bytes  
segment address = 5070602401127665962733751275520 bytes  
segment address = 1014120480245533192546750255040 bytes  
segment address = 2028240960491066385093500510080 bytes  
segment address = 4056481920982132770187001020160 bytes  
segment address = 8112963841964265540374002040320 bytes  
segment address = 1622592768392853108074004080640 bytes  
segment address = 3245185536785706216148008161280 bytes  
segment address = 6490371073571412432296016322560 bytes  
segment address = 1298074214714282486459232665120 bytes  
segment address = 2596148429428564972918465330240 bytes  
segment address = 5192296858857129945836930660480 bytes  
segment address = 10384593717714259891673861320960 bytes  
segment address = 20769187435428519783347722641920 bytes  
segment address = 41538374870857039566695445283840 bytes  
segment address = 83076749741714079133390890567680 bytes  
segment address = 166153499483428158266781781155360 bytes  
segment address = 332306998966856316533563562310720 bytes  
segment address = 664613997933712633067127124621440 bytes  
segment address = 1329227995867425266134254249242880 bytes  
segment address = 2658455991734850532268508498485760 bytes  
segment address = 5316911983469701064537016996971520 bytes  
segment address = 10633823966939402129074033993943040 bytes  
segment address = 21267647933878804258148067987886080 bytes  
segment address = 42535295867757608516296135975772160 bytes  
segment address = 85070591735515217032592271951544320 bytes  
segment address = 170141183471030434065184543903088640 bytes  
segment address = 340282366942060868130369087806177280 bytes  
segment address = 680564733884121736260738175612354560 bytes  
segment address = 1361129467768243472521476351224709120 bytes  
segment address = 2722258935536486945042952702449418240 bytes  
segment address = 5444517871072973890085905404898836480 bytes  
segment address = 1088903574214594778017180580977767360 bytes  
segment address = 2177807148429189556034361161955534720 bytes  
segment address = 4355614296858379112068722323811069440 bytes  
segment address = 8711228593716758224137444647622138880 bytes  
segment address = 1742245718743351644827488929524427760 bytes  
segment address = 3484491437486703289654977859048855520 bytes  
segment address = 6968982874973406579309955718097711040 bytes  
segment address = 13937965749946813158619911436195422080 bytes  
segment address = 27875931499893626317239822872388844160 bytes  
segment address = 55751862999787252634479645744777688320 bytes  
segment address = 111503725999574505268959291489555376640 bytes  
segment address = 223007451999149010537918582979110753280 bytes  
segment address = 446014903998298021075837165958221506560 bytes  
segment address = 892029807996596042151674331916443013120 bytes  
segment address = 1784059615993192084303348663832886026240 bytes  
segment address = 3568119231986384168606697327665772052480 bytes  
segment address = 7136238463972768337213394655331544104960 bytes  
segment address = 14272476927945336674426789306663088209920 bytes  
segment address = 28544953855890673348853578613326176419840 bytes  
segment address = 57089907711781346697707157226652352839680 bytes  
segment address = 11417981542356273339541431445330470567360 bytes  
segment address = 2283596308471254667908286289066094113520 bytes  
segment address = 4567192616942509335816572578132188227040 bytes  
segment address = 9134385233885018671633145156264376454080 bytes  
segment address = 1826877046777003734326580231252755288160 bytes  
segment address = 3653754093554007468653160462505510576320 bytes  
segment address = 7307508187108014937306320925011021152640 bytes  
segment address = 14615016374216029874612641850220422305280 bytes  
segment address = 29230032748432059749225283700440844610560 bytes  
segment address = 58460065496864119498450567400881689221120 bytes  
segment address = 11692013099372823899690134801763217844240 bytes  
segment address = 23384026198745647799380269603526435688480 bytes  
segment address = 46768052397491295598760539207052871376960 bytes  
segment address = 93536104794982591197521078414105742753920 bytes  
segment address = 187072209589965182395042156828211485507840 bytes  
segment address = 374144419179930364790084313656422917015680 bytes  
segment address = 748288838359860729580168627312845834031360 bytes  
segment address = 1496577676719721459160337254625691668062720 bytes  
segment address = 2993155353439442918320674509251383336135440 bytes  
segment address = 5986310706878885836641349018502766672270880 bytes  
segment address = 11972621413757771673282688037005533344541760 bytes  
segment address = 23945242827515543346565376074011066688583520 bytes  
segment address = 47890485655031086693130752148022133377167040 bytes  
segment address = 95780971310062173386261504296044266754334080 bytes  
segment address = 191561942620124346772523008592088533508668160 bytes  
segment address = 383123885240248693545046017184177067017336320 bytes  
segment address = 766247770480497387090092034368354134034672640 bytes  
segment address = 1532495540960994774180184068736708268069345280 bytes  
segment address = 3064991081921989548360368137473416536138690560 bytes  
segment address = 6129982163843979096720736274946833112277381120 bytes  
segment address = 1225996432768795819344147254989366622455476240 bytes  
segment address = 2451992865537591638688294509978733244910952480 bytes  
segment address = 4903985731075183277376589019957466489821904960 bytes  
segment address = 9807971462150366554753178039914932979643809920 bytes  
segment address = 19615942924300733109506356079829865959287619840 bytes  
segment address = 39231885848601466219012712159659731918575239680 bytes  
segment address = 78463771697202932438025424319319463837150479360 bytes  
segment address = 156927543394405864876058848638638927674300958720 bytes  
segment address = 313855086788811729752117697277277855348601917440 bytes  
segment address = 627710173577623459504235394554555710697203834880 bytes  
segment address = 1255420347155246919008470789109114211394407669760 bytes  
segment address = 2510840694310493838016941578218228422788815339520 bytes  
segment address = 5021681388620987676033883156436456845577630678040 bytes  
segment address = 10043362777241975352067766312872913691553261356080 bytes  
segment address = 20086725554483950704135532625745827383106526712160 bytes  
segment address = 40173451108967901408271065251491654766213053424320 bytes  
segment address = 80346902217935802816542130502983295332426106848640 bytes  
segment address = 16069380443871605633264426100596659066485221369280 bytes  
segment address = 32138760887743211266528852200193318128970442738560 bytes  
segment address = 64277521775486422533057704400386636257440885477120 bytes  
segment address = 12855504355097284506611540880077327254881776954240 bytes  
segment address = 25711008710194569013223081760154654509763553908480 bytes  
segment address = 51422017420389138026446163520309309019527107816960 bytes  
segment address = 102844034840778276052892327040618618038554215633920 bytes  
segment address = 20568806968155655210578465408123723607707843127840 bytes  
segment address = 41137613936311310421156930816247447215415686255680 bytes  
segment address = 82275227872622620842313861632494894430831372511360 bytes  
segment address = 164550455745245241684627323264989788616626750022720 bytes  
segment address = 329100911490490483369254646529975577233253500055440 bytes  
segment address = 658201822980980966738509293059951154666507000110880 bytes  
segment address = 1316403645961961933477018586119852309333014000221760 bytes  
segment address = 2632807291923923866954037172239704618666028000443520 bytes  
segment address = 5265614583847847733908074344479409237332056000887040 bytes  
segment address = 10531229167695695467816148688958818474664112001774080 bytes  
segment address = 2106245833539139093563229737791763694932824003548160 bytes  
segment address = 4212491667078278187126459475583527389865648007096320 bytes  
segment address = 8424983334156556374252918951167054779731296014192640 bytes  
segment address = 16849966668313112788505837902334109598662592028385280 bytes  
segment address = 33699933336626225577011675804668219193325184056770560 bytes  
segment address = 67399866673252451154023351609336438386550368113511120 bytes  
segment address = 134799733346504902308046703218672676773100736227022240 bytes  
segment address = 269599466693009804616093406437345353546201472454044480 bytes  
segment address = 53899893338601960923218681287468567073140294490808960 bytes  
segment address = 107799786673203921846437362574937134146805889917717920 bytes  
segment address = 215599573346407843692874725149874268293411779835435840 bytes  
segment address = 431199146692815687385749450299748533646823559670871680 bytes  
segment address = 862398293385631374771498900599497067293647119341743360 bytes  
segment address = 1724796586771262749542978001198994134587294386883486720 bytes  
segment address = 3449593173542525499085956002397982269174588773767373440 bytes  
segment address = 6899186347085050998171912004795964538349177547534746880 bytes  
segment address = 13798372694170101996343824009591929076698355095064933760 bytes  
segment address = 27596745388340203992687648019183858153396710190129867520 bytes  
segment address = 55193490776680407985375296038367716306793420380257735040 bytes  
segment address = 110386981553360815970750592766754326135586847604555470080 bytes  
segment address = 22077396310672163194150118553350865227117369520911094160 bytes  
segment address = 44154792621344326388300237106701730454234739041822188320 bytes  
segment address = 88309585242688652776600474213403460908469478083644376640 bytes  
segment address = 176619170485377305553200948426806818168938956167288753280 bytes  
segment address = 353238340970754611106401896853613636337877912334577506560 bytes  
segment address = 706476681941509222212803793707227326675755824669155013120 bytes  
segment address = 1412953363883018444425607587414454653351511649338305026240 bytes  
segment address = 2825906727766036888851215174828909306703023298676610052480 bytes  
segment address = 5651813455532073777702430349657818613406046597333220104960 bytes  
segment address = 1130362691066414755540486069931563722681209194466644021920 bytes  
segment address = 2260725382132829511080972139863127445242418388933288043840 bytes  
segment address = 4521450764265659022161944279726254884844837777866576087680 bytes  
segment address = 9042901528531318044323888559452509769689675555733152175360 bytes  
segment address = 18085803057062636088647777118905019539379351114466304350720 bytes  
segment address = 36171606114125272177295554237810039078758702229132608701440 bytes  
segment address = 72343212228250544354591108475620078157117404458265217402880 bytes  
segment address = 14468642445450108870918221691240015634234880891653043580560 bytes  
segment address = 28937284890900217741836443382480311266867761783266087161120 bytes  
segment address = 57874569781800435483672886764960622533735523566532174322240 bytes  
segment address = 115749139563600870967345773529921244667471067132664348644480 bytes  
segment address = 231498279127201741934691547059842489334822134265328697288960 bytes  
segment address = 462996558254403483869383094119684978669644268530657394577920 bytes  
segment address = 925993116508806967738766188239369957339288537061314789155840 bytes  
segment address = 1851986233017613935475332376478799114678577154122629578311680 bytes  
segment address = 3703972466035227870950664752957598229357154308245259156623360 bytes  
segment address = 7407944932070455741901329505915196458714308616490518313246720 bytes  
segment address = 14815889864140911483802659011830392917428617232981036626493440 bytes  
segment address = 29631779728281822967605318023660785834857234465962073252986880 bytes  
segment address = 59263559456563645935210636047321571669114468931924146505973760 bytes  
segment address = 1185271189131272918704

25-10-2023

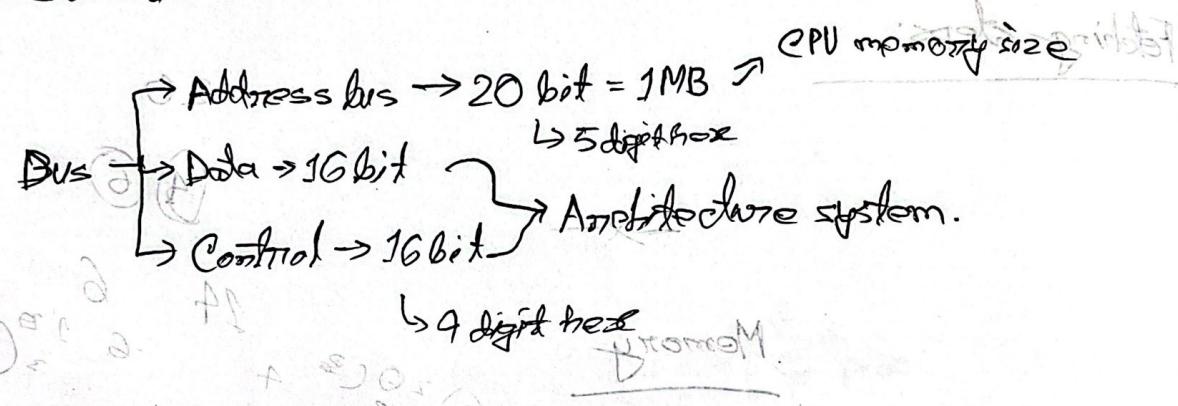
1 MB = 64 KB  
1 MB = 1024 KB

register 199 to string A

Address bus size → Memory size  
bit → byte

Byte

8086 → 16 bit CPU



Address  
(4 hex digits)

1 MB ←  
8 × 10<sup>6</sup> ←  
8 × 10<sup>6</sup> ←

1 MB ←

20 bit

1 MB

→ 5 hex digit

1 segment

SEG = 16 bits  
1 MB = 1024 × 1024 bytes

Segmentation:

$$\frac{1 \text{ MB}}{64 \text{ KB}} = \frac{1 \times 1024 \times 1024}{364 \times 1024}$$

= 16 segments

PSOR X1 XPSOR

$$2 \text{ MB} = 2^{20} \times 2^1$$

Date: 05-10-25

Formal Date: 18/3/23

16 bit

2 MB

32 bit

2 MB

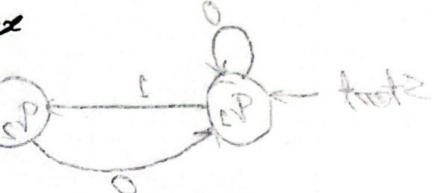
per segment  $\rightarrow$  64 kB

$\hookrightarrow$  16 bit

$\hookrightarrow$  4 hex

Memory segment Address = SP

int out address goes



per location  $\rightarrow$  8 bit -

(alternate shift determines) A7D

CSE-341(LAB)

Date: 01-10-22

A27

A7H

A7L

Registers:

GPR

16 bit

AX

AH BL

BX

BH BL

CX

CH CL

DX ~~DX~~ DL change to the shift is 3



Segment registers -  $3 \times 8 = 24$  without multiplier shift is 3

CS 64kb

D3 op state flags (bitwise shift is 0)

DS 64kb

SS 64kb

D37 8 state brief to the op is 7

ES 64kb

20 bit  $\rightarrow$  Address Bus  $\rightarrow 2^{20} \rightarrow 1\text{ MB}$  DD

Data Bus / Control / Registers  $\rightarrow 2^{16}$

Logical Address = A4FB:4872

$\Rightarrow$  Segment  $\downarrow$  Offset

Physical Address = Logical Segment  $\times 10h + \text{Offset}$

Types of segmentation.

Overlapping segmentation

Non-overlapping segmentation

October - 15<sup>th</sup>  $\rightarrow$  Quiz - 1

Loc 1, 2, 3 [including 8086 architecture]

IT 24 bit  $\rightarrow$  Address bits

Q. Calculate the total memory size that it can address

Q. Determine the first and last address of that memory

$$\text{Ans: } 2^4 = 2^{20} \times 2^4$$

$$= 1 \text{ MB} \times 16$$

$$= 16 \text{ MB}$$

$$\text{Ans: } 24 \text{ bit, hex} = \frac{24}{4} = 6 \text{ hex}$$

$$\text{Start: } 000000h \leftarrow \text{EIP: } 0000$$

$$\text{End: } FFFFFFFFh \leftarrow \text{EIP: } 1000$$

$$\text{Address} = \text{EIP} + \text{Offset} \leftarrow \text{EIP: } 1000$$

Logical Address:

$$1234:5678h, 05:23A4$$

Address

Calculate the physical address -

$$PA = 1234 \times 10h + 5678$$

$$= 379B8h$$

$$\begin{aligned} &\text{MAES: } 2000 \\ &PA = 0005 \times 10h + 23A4 \\ &\text{MAES: } 2000 \\ &= 023A4h \end{aligned}$$

Assuming Non-overlapping Segmentation:

and segment size is 16

Physical Address: 12345h

What is the segment no./which segment does the address belong to? (Segment -1)

What is the first and last address? ( $10000h \rightarrow F$ )  
 $(FFFFh \rightarrow L)$

Calculate 3 logical addresses given 12345h.

$$1. 1000 : 2345 \rightarrow 1000 \times 10 + 2345 = 12345h$$

$$2. 1001 : 2335 \rightarrow 1001 \times 10 + 2335 = 12345h$$

$$3. 1002 : 2325 \rightarrow 1002 \times 10 + 2325 = 12345h$$

0237A

$$1. 0005 : 232Ah$$

$$PAES + 10 \times 2000 = A9$$

$$2. 0003 : 234Ah$$

$$PAES =$$

$$8522 + 10 \times 2000 = A9$$

$$PAES =$$

PA = 23427h

IMM5

What is the highest segment address and lowest segment address for the given physical address?

Segment

\*\* Segment address should be divisible by 10:

MM5



JMM5

Logical A:

2342 = 00007h ← Highest 4 bits LA

Segment shift

1342

1343 : 9997h ← Lowest LA

Logical Address &lt; shift

will have 4X &lt; 9X ← bit manipulation required

23427 - FFFF

→ 13430h → Lowest Segment 70  
 → 13428h ← offset ← Logical Address  
 → 13420h → Physical Address

bit manipulation required

Segment 70

Required

Physical Address

Temporary logical offset ← KIT

Temporary logical offset ←

Physical Address

Some modified bracket

This modified bracket

3897AH

Logical Address:

Seg no : Offset

↳ 4 hex  
digit

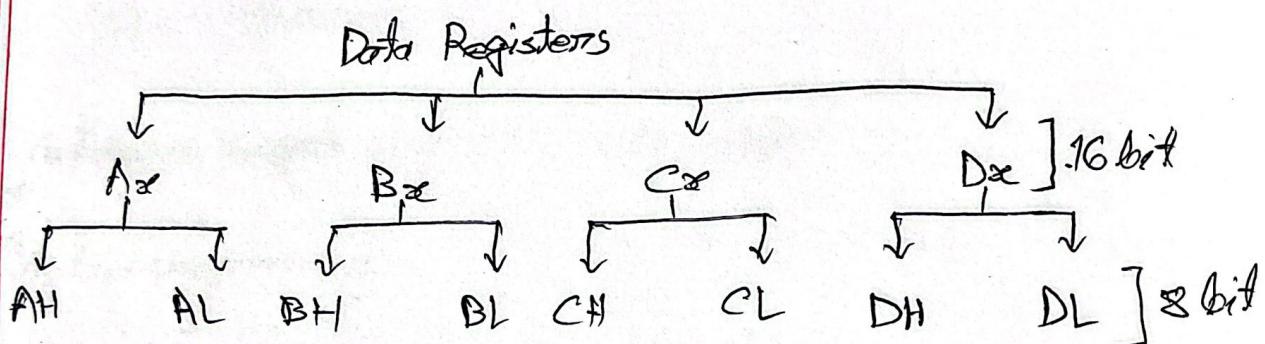
↳ 4 hex  
digit

Seg no. → starting address of that seg

concept of 32 bit

All of seg, faster  
less work

### Registers



### Segment Registers } 16 bit

#### Offset registers

CS → Code Segment → IP - Instruction pointer

Destination index

DS → Data Segment → Stores address → DI, SI, BX → Base

Source index

ES → Extra segment → DI, SI, BX

SS → Stack segment → SP, BP

Base pointer

### Flag registers (Later)

## Offset Registers

Eg:  $\text{DS} = 1234\text{h}$

$\hookrightarrow$  Offset

Physical Address =  $\text{DS} \times 10 + \text{SI}$

Address Register

Q) A microprocessor which contains 28 bit address bus.

a. What is the total memory size that it can access?

b. ... .. first address and last address?

$2^{\text{20}}$

$$\therefore \text{Memory size} = 2^{20} \times 2^8 \quad \times \text{last digit}$$

$$= (1 \times 256) \text{ MB}$$

$\hookrightarrow$  start address: 0000000

end address: FFFFFFFF

Q) Assuming non-overlapping:

$34 \times 28 \text{ h} \rightarrow \text{Physical Address}$

What is the segment address / base address?

$\hookrightarrow 30000\text{h}$

$\hookrightarrow$  Segment Address

(Q) Given,

2743AH

Determine 3 possible segment addresses and their corresponding logical address.

Logical Address

Base address : 27430h

2743:000Ah

" " : 27420h

" " : 27400h

Segment register has to be programmed with one value.

Segments have to be loaded with .. .. .

Logical address =  $S \times 16^3 + \text{segment offset}$

00000000 ← segment base address

$8M(64 \times 1) =$

77777777 ← segment limit

Programmed segments A(0)

Segment base ← 123456

Segments and segment limits will be same

10000000 ←

Segment limit ←

## Internal Architecture of 8056



Bus interface unit

CSE-331

Date: 11-10-22

## Regular Language

## Three operations

OR Concatenation Kleene closure

$$\Sigma = \{0, 1\}$$

$$\Sigma = \{a, b, c\}$$

$$0^* = \{0, 0^{\circ}, 0^{\circ}, 0, 00, \dots\}$$

01.1

alb | e

$$01^* = \{0, 01, 011, \dots, \infty\}$$

〇一三九

\*3 (\*(( )) | \*((( ))))

$$(11)^* = \{11, 1111, \dots, \infty\}$$

$$(01)^* = \{q, p, 1, 00, 11, \dots\} \quad (001)^* = \{e, 001, 001001, \dots, e\}$$

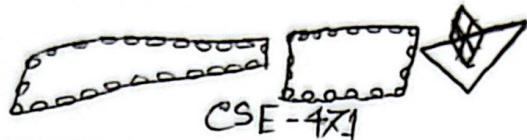
$$= \{E, O\}^1,$$

$$\frac{1}{\sqrt{3}} \left( (\psi_0) \otimes \mathbb{I} + \mathbb{I} \otimes (\psi_0)^\dagger \right) \frac{1}{\sqrt{3}}$$

$$(0|1)^* = \{ \infty, 0, 1, \text{double } 0, \text{double } 1, \dots, \text{double } \infty \}$$

$\Sigma^*$  → adopted cleaner discrete

$$(11\bar{3})^*(0110) \quad \cancel{+(01)}$$



Date: 13-10-22

Scrum → Most widely used methodology

Scrum master → Manages the project

Project master/owner

Product backlog

↓  
Sprint backlog

↳ Smaller part of product

Burndown chart is used to keep track of daily

Backlog.

progress.

1956h

00010001 0111 < Pending

16-bit 8086 Architecture

[Did it 2 more times] 0 = 70

Bus Interface Unit (BIU) → Fetch

[Did it 2 more times] 0 = 71

FU → Execution Unit

0 = 70

(Did it 2 more times) 0 = 72

Decode & Execute

Instruction Queue  
↳ Pending

0 = 73 < Pending

0 = 74 < Pending

0 = 75 < Pending

0 = 76 < Pending

0 = 77 < Pending

0 = 78 < Pending

0 = 79 < Pending

0 = 80 < Pending

0 = 81 < Pending

0 = 82 < Pending

0 = 83 < Pending

0 = 84 < Pending

0 = 85 < Pending

0 = 86 < Pending

0 = 87 < Pending

0 = 88 < Pending

0 = 89 < Pending

0 = 90 < Pending

0 = 91 < Pending

0 = 92 < Pending

0 = 93 < Pending

0 = 94 < Pending

0 = 95 < Pending

0 = 96 < Pending

0 = 97 < Pending

0 = 98 < Pending

0 = 99 < Pending

0 = 100 < Pending

0 = 101 < Pending

0 = 102 < Pending

0 = 103 < Pending

0 = 104 < Pending

0 = 105 < Pending

0 = 106 < Pending

0 = 107 < Pending

0 = 108 < Pending

0 = 109 < Pending

0 = 110 < Pending

0 = 111 < Pending

0 = 112 < Pending

0 = 113 < Pending

0 = 114 < Pending

0 = 115 < Pending

0 = 116 < Pending

0 = 117 < Pending

0 = 118 < Pending

0 = 119 < Pending

0 = 120 < Pending

0 = 121 < Pending

0 = 122 < Pending

0 = 123 < Pending

0 = 124 < Pending

0 = 125 < Pending

0 = 126 < Pending

0 = 127 < Pending

0 = 128 < Pending

0 = 129 < Pending

0 = 130 < Pending

0 = 131 < Pending

0 = 132 < Pending

0 = 133 < Pending

0 = 134 < Pending

0 = 135 < Pending

0 = 136 < Pending

0 = 137 < Pending

0 = 138 < Pending

0 = 139 < Pending

0 = 140 < Pending

0 = 141 < Pending

0 = 142 < Pending

0 = 143 < Pending

0 = 144 < Pending

0 = 145 < Pending

0 = 146 < Pending

0 = 147 < Pending

0 = 148 < Pending

0 = 149 < Pending

0 = 150 < Pending

0 = 151 < Pending

0 = 152 < Pending

0 = 153 < Pending

0 = 154 < Pending

0 = 155 < Pending

0 = 156 < Pending

0 = 157 < Pending

0 = 158 < Pending

0 = 159 < Pending

0 = 160 < Pending

0 = 161 < Pending

0 = 162 < Pending

0 = 163 < Pending

0 = 164 < Pending

0 = 165 < Pending

0 = 166 < Pending

0 = 167 < Pending

0 = 168 < Pending

0 = 169 < Pending

0 = 170 < Pending

0 = 171 < Pending

0 = 172 < Pending

0 = 173 < Pending

0 = 174 < Pending

0 = 175 < Pending

0 = 176 < Pending

0 = 177 < Pending

0 = 178 < Pending

0 = 179 < Pending

0 = 180 < Pending

0 = 181 < Pending

0 = 182 < Pending

0 = 183 < Pending

0 = 184 < Pending

0 = 185 < Pending

0 = 186 < Pending

0 = 187 < Pending

0 = 188 < Pending

0 = 189 < Pending

0 = 190 < Pending

0 = 191 < Pending

0 = 192 < Pending

0 = 193 < Pending

0 = 194 < Pending

0 = 195 < Pending

0 = 196 < Pending

0 = 197 < Pending

0 = 198 < Pending

0 = 199 < Pending

0 = 200 < Pending

0 = 201 < Pending

0 = 202 < Pending

0 = 203 < Pending

0 = 204 < Pending

0 = 205 < Pending

0 = 206 < Pending

0 = 207 < Pending

0 = 208 < Pending

0 = 209 < Pending

0 = 210 < Pending

0 = 211 < Pending

0 = 212 < Pending

0 = 213 < Pending

0 = 214 < Pending

0 = 215 < Pending

0 = 216 < Pending

0 = 217 < Pending

0 = 218 < Pending

0 = 219 < Pending

0 = 220 < Pending

0 = 221 < Pending

0 = 222 < Pending

0 = 223 < Pending

0 = 224 < Pending

0 = 225 < Pending

0 = 226 < Pending

0 = 227 < Pending

0 = 228 < Pending

0 = 229 < Pending

0 = 230 < Pending

0 = 231 < Pending

0 = 232 < Pending

0 = 233 < Pending

0 = 234 < Pending

0 = 235 < Pending

0 = 236 < Pending

0 = 237 < Pending

0 = 238 < Pending

0 = 239 < Pending

0 = 240 < Pending

0 = 241 < Pending

0 = 242 < Pending

0 = 243 < Pending

0 = 244 < Pending

0 = 245 < Pending

0 = 246 < Pending

0 = 247 < Pending

0 = 248 < Pending

0 = 249 < Pending

0 = 250 < Pending

0 = 251 < Pending

0 = 252 < Pending

0 = 253 < Pending

0 = 254 < Pending

0 = 255 < Pending

0 = 256 < Pending

0 = 257 < Pending

0 = 258 < Pending

0 = 259 < Pending

0 = 260 < Pending

0 = 261 < Pending

0 = 262 < Pending

0 = 263 < Pending

0 = 264 < Pending

0 = 265 < Pending

0 = 266 < Pending

0 = 267 < Pending

0 = 268 < Pending

0 = 269 < Pending

0 = 270 < Pending

0 = 271 < Pending

0 = 272 < Pending

0 = 273 < Pending

0 = 274 < Pending

0 = 275 < Pending

0 = 276 < Pending

0 = 277 < Pending

0 = 278 < Pending

0 = 279 < Pending

0 = 280 < Pending

0 = 281 < Pending

0 = 282 < Pending

0 = 283 < Pending

0 = 284 < Pending

0 = 285 < Pending

0 = 286 < Pending

0 = 287 < Pending

0 = 288 < Pending

0 = 289 < Pending

0 = 290 < Pending

0 = 291 < Pending

0 = 292 < Pending

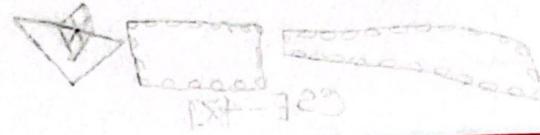
0 = 293 < Pending

0 = 294 < Pending

0 = 295 < Pending

0 = 296 < Pending

0 = 297 < Pending</p



## 8086 Flag Registers

Control Flag

Status Flag  $\rightarrow$  Overflow Flag  $\rightarrow$  OF

Sign Flag  $\rightarrow$  SF

Zero Flag  $\rightarrow$  ZF

Auxiliary Flag  $\rightarrow$  AF

Parity Flag  $\rightarrow$  PF

Carry Flag  $\rightarrow$  CF

All for output

MOV AL, 50h       $\begin{array}{r} 0010 \\ \overline{0101} \\ \hline 01010000 \end{array}$       → 1 nibble

MOV BL, 32h       $\begin{array}{r} 00110010 \\ \overline{10000010} \\ \hline 10000010 \end{array}$

ZF = 0 [Output is not 0]

CF = 0 [No carry out from 8th bit]

SF = 1 [MSB is 1]

AF = 0 [No carry out after 1 nibble]

ADD AL, BL

1 word = 16 bit

1 byte = 8 bit

1 nibble = 4 bit

OF = 1

PF = 1 (Even number of 1's in output)

stored & stored

If there is a carry from 7th bit

to 8th bit = OF = 1

If there is a carry out from 8th bit

to 9th bit = OF = 1

OR operation

2 1  $\rightarrow$  0

SC-0525.361

SC-0525.720

Mov AL, FFh

$\begin{array}{ccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}$

ZF = 1

Mov AL, 09h

$\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$

CF = 1

ADD AL, BL

SF = 0

DS = 2246h

BP = 32h

AF = 1

SS = 1256h

SP = 20h

OF = 0

CS = 2243h

IP = 1426h

PF = 01

SI = 2200h

2346h 00100011 0000 0110

1256h 00010010 0101 0110

MOV AX, [BX]

ZF = 0

CF = 0

SF = 0

AF = 0

OF = 1

PF = 001 : [Low Byte - 8]



Address

[ ]  
off

Suppose,

$$DS = 2246h$$

$$IP = 32h$$

$$SS = 12FFh$$

$$SP = 2200h$$

$$CS = 2243h$$

$$BP = 14264h$$

$$SI = 2200h$$

$$DI = 1246h$$

$$BX = 20h$$

$MOV AX, [BX + SI]$

for 16 bit word = 16 bits

Q. Calculate the physical address from which data will be accessed.

$$P.A = (DS \times 10) + (BX + SI)$$

$$= 2246 \times 10 + 2220$$

$$= 247629680h$$

Q. Calculate the 2<sup>nd</sup> highest and 3<sup>rd</sup> lowest segment addresses from the physical address calculated. Determine the respective logical address.

$$\begin{aligned}
 & 24680 \xrightarrow{\text{2nd highest}} 24670 \rightarrow 2467:0010 \\
 & \quad \text{d00E5 = 92} \quad \text{d7751 = 22} \\
 & \text{2nd highest} = 24680 - \text{FFFF} \\
 & \quad \text{dF08P0 = 9A} \quad \text{d146X0 = 2D} \\
 & = 14681 \quad \begin{array}{l} \rightarrow 19690 \\ \rightarrow 14680 \end{array} \quad \begin{array}{l} \cancel{14680} \\ \cancel{14670} \end{array} \\
 & = \cancel{14680} \quad \begin{array}{l} \cancel{14670} \\ \cancel{14680} \end{array} \\
 & \quad \text{1st} \quad \text{2nd} \quad \text{dAF51 = 10} \\
 & \quad \text{d05 = XE0}
 \end{aligned}$$

[EXER] XA VOM

Logical = Segment : Offset

Answer of this question is 20700 because offset is 10. D  
Note from pic

$$(12+X8) + (0(X2)) = A.9$$

$$0555 + 01X2A55 =$$

$$1023P5 25FA =$$

Ans

MOV CX, [SI+22H]

ADRES, XA VOM

Q - Calculate the physical address.

X A. JESUS VOM

$\rightarrow [SeD_2] \cdot [SeD_2]$

$$P-A = 55 \times 10 + (5P+22h)$$

$$= 12FFh \times 10 + (2300h + 22h)$$

1EE-32

$$= 15312h$$

Mov [BX + DJ + 200], AX  $\rightarrow$  4000h bytes are copied to it.

Q - Calculate the physical address whose data will be stored.

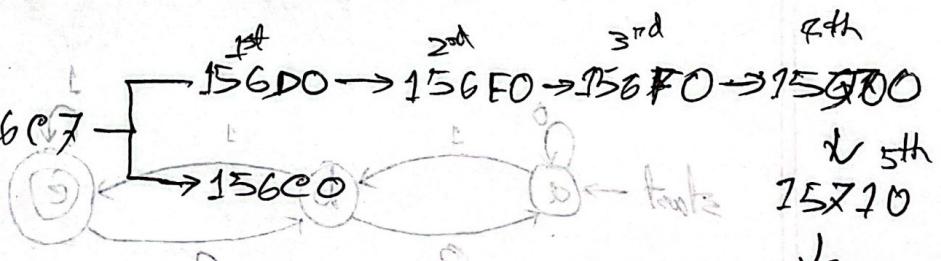
$$P \cdot A = DSX90 + (BX + DJ + 2000)$$

$$= (2296 \times 10) + (20 + 1246 + 2000)$$

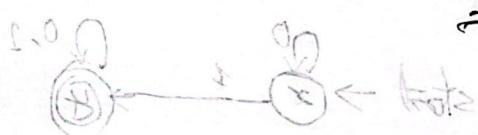
$$= 256 \text{ c6}$$

$$2^{\text{nd}} \text{ highest} = 256\text{CO} \rightarrow 256\text{BO} \rightarrow 256\text{AO} \rightarrow 256\text{A}:0026$$

$$256CG - FFFF = 156$$



1571.FFB6h



Mov Ax, 2356h

Mov 2356, AX

ADD [4356], [222] &

[435+92], 22 22

$$(435+92) + 01x22 = A9$$

Date: 16-10-22

(435+10000) + 01x22 =

(P.B., P. 3.0)

$\begin{matrix} CS \\ \uparrow \end{matrix}$        $\begin{matrix} IP \\ \uparrow \end{matrix}$   
~~JMP 2000, 2346~~ → direct addressing  $\leftarrow 3 \times 0 - 2$

$P.A = CS \times 10 + IP$

Addressing Program codes from ~~instruction!~~!

$$CS = 2000H$$

JMP AX

$$DS = 4000H$$

$$P.A = CS \times 10 + AX$$

$$BX, AX = 2236H = IP$$

JMP [BX]

$$P.A = (DS \times 10) + BX$$

$$= 4000 \times 10 + 2236$$

$$= 42236H$$

$$\begin{aligned} 42237 &\rightarrow 42H \\ 42236 &\rightarrow 30H \end{aligned} \quad \{ \quad \}$$

$IP = 4230H \leftarrow$

$$P.A = CS \times 10 + IP$$

Coll [BX]

$$CS = 2322h$$

DS-12464h

$$Bx = 2200h$$

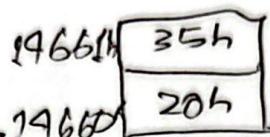
no fixed  $\mathbb{R} = \emptyset$

$$P \cdot A = DS \times 10 + BX$$

$$\text{GDP per capita} = 1246 \times 10 + 2200$$

JA 9660

QOM no abrigos = 148



*Brachypterus* sp. = 352nh

std R = diag)

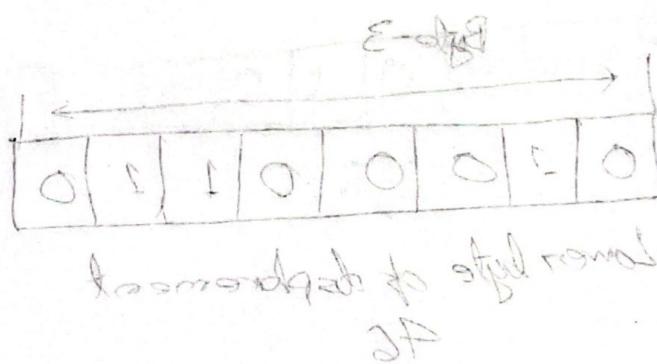
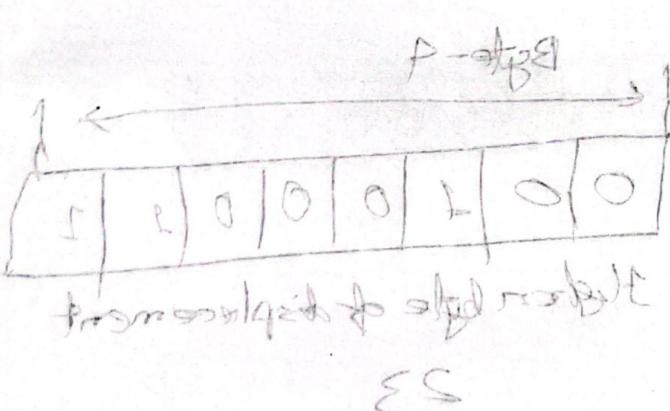
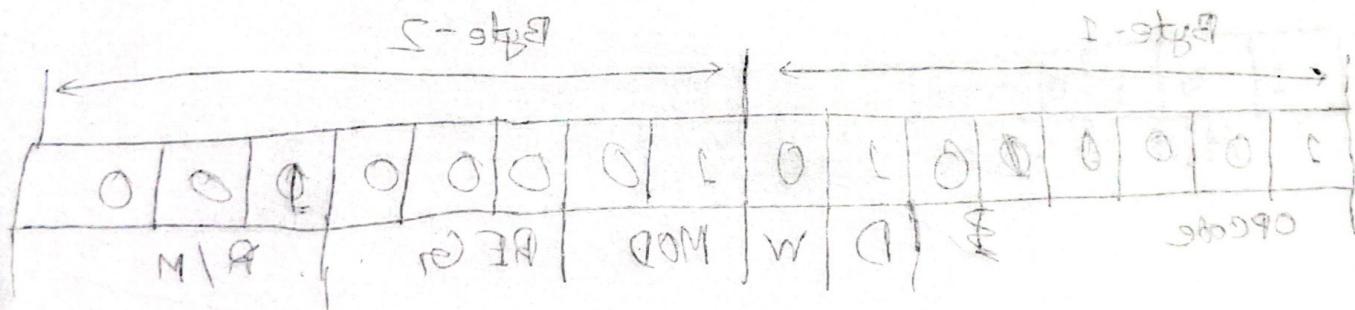
P.A = CS X 10-1 JP

[A]AES + [2], JA - VOM

read right

eff to transmigrate as strength DOM

[F PESCI + 52], 1A



Call [BX]

$$CS = 2322h$$

$$DS = 12464h$$

$$BX = 2200h$$

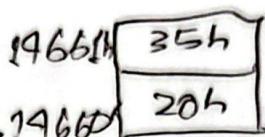
$$\text{mid word} \ll 1 = 0$$

$$P.A = DS \times 10 + BX$$

$$\text{addr calc} = 1246 \times 10 + 2200$$

$$= 14660$$

$$10M \text{ no change} = MA$$



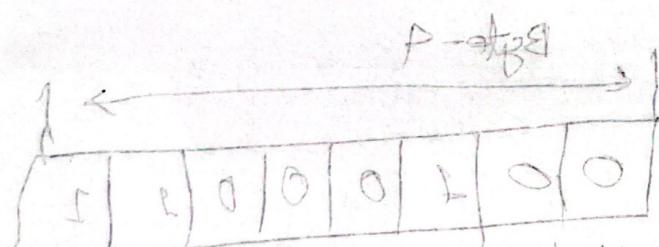
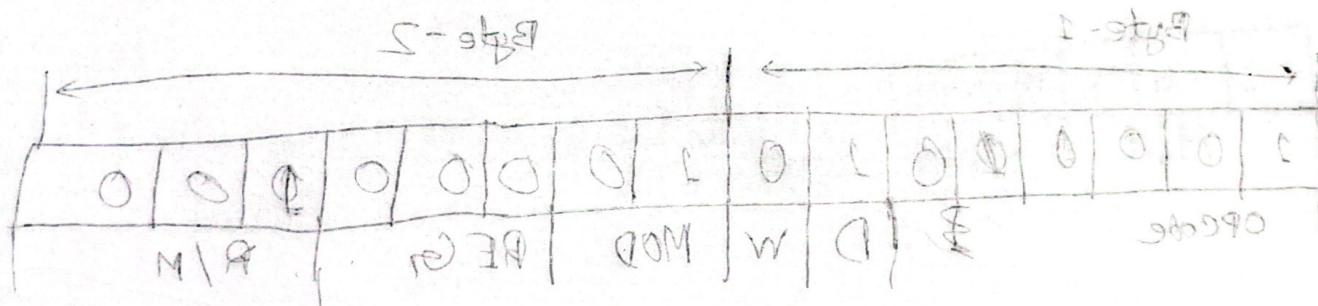
$$P.A = CS \times 10 + JP$$

$$= \underline{\underline{[1246]}} \cdot 10 + VOM$$

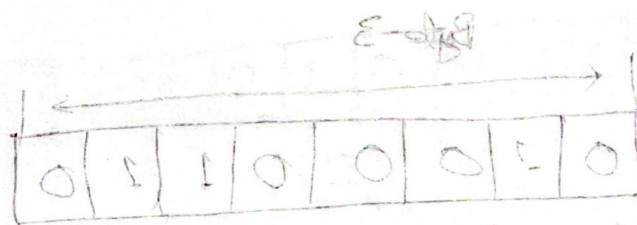
word select

map to memory as address MDR

[1246 + 32], 1A

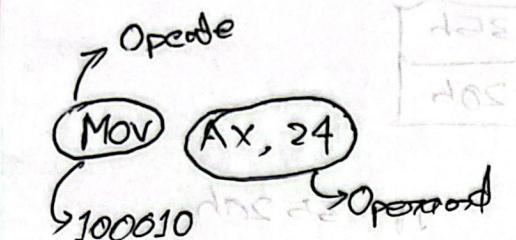


ES



from memory to shift register  
EF

Instruction queue minimum 2 bytes, maximum 6 bytes



Opcode = 6 bits

MOV AL, [SI + 2346h]  
dollar sign  
dollar sign

~~descrip - 20~~  
~~descrip - 20~~

$W=1$ : for 16 bit

$W=0$ : for 8 bit

$D=1$ : Destination

$D=0$ : Source  $\rightarrow$  A.T

Data register

~~descrip - 20~~  
~~descrip - 20~~

$R_Eq = 3$  bit data register value

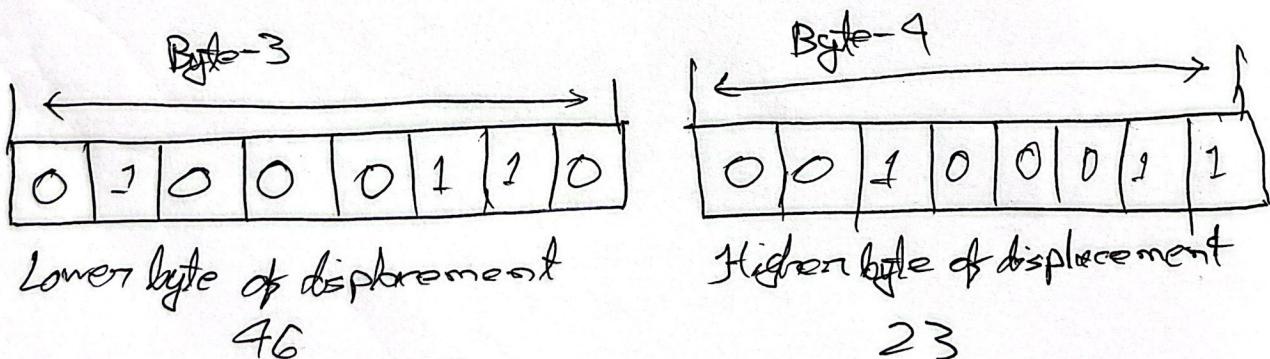
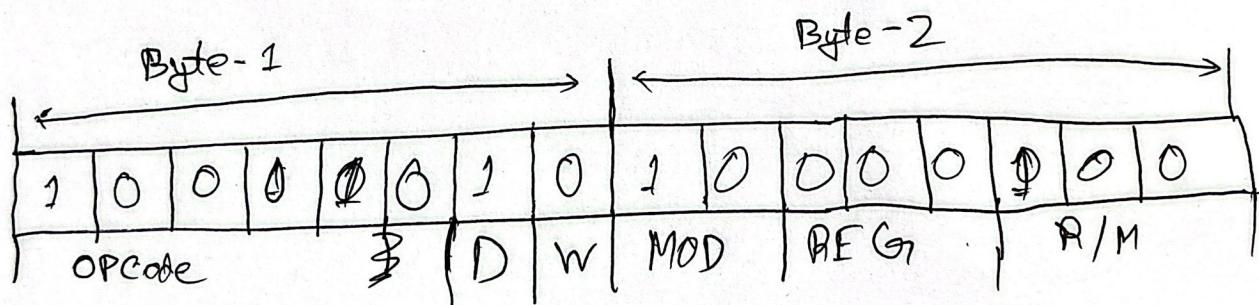
~~descrip - 20~~  
~~descrip - 20~~

000  $\rightarrow$  AL

$R_M = \text{depends on MOD}$

MOD depends on displacement of offset

AL, [SI + 1234h]



46

23

MOV DH, [BX+SI]+22h

Byte-1								Byte-2																																							
<table border="1"><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>OPCode</td><td>(D)</td><td>w</td><td>MOD</td><td>REG</td><td>R/M</td><td>PE(G)</td><td>R/M</td></tr></table>								1	0	0	0	1	0	1	0	OPCode	(D)	w	MOD	REG	R/M	PE(G)	R/M	<table border="1"><tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>REG</td><td>R/M</td><td>PE(G)</td><td>R/M</td><td>REG</td><td>R/M</td><td>REG</td><td>R/M</td></tr></table>								0	1	1	1	0	0	0	0	REG	R/M	PE(G)	R/M	REG	R/M	REG	R/M
1	0	0	0	1	0	1	0																																								
OPCode	(D)	w	MOD	REG	R/M	PE(G)	R/M																																								
0	1	1	1	0	0	0	0																																								
REG	R/M	PE(G)	R/M	REG	R/M	REG	R/M																																								
M14   M15   M16   M17   M18   M19   M20   M21								M22   M23   M24   M25   M26   M27   M28   M29																																							

Byte-3

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

Lower byte of the displacement

22h

8A7022h

→ Hexadecimal representation  
of the selected machine language

MOV [DI + 2240], AX

→ Source

1	0	0	1	0	0	1	2	0	0	0	1	0	1					
OPCode	(D)	w	MOD	REG	R/M	M14   M15   M16   M17   M18   M19   M20   M21   M22   M23   M24   M25   M26												

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

40

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

22

89854022

→ Hex representation.

Mov [2240h], AX

HW DH [64+128]

Byte

1	0	0	0	1	0	0	0	0	1	1	0
OC	≈	D	w	MOD	REG	R/M					

ASCS

Byte

0	0	0	0	0	0	0
---	---	---	---	---	---	---

0100 0000 0010 0010

SS

Mov AX, BX

XA [00SS + D] ROM

source

100010 111100011011

1	0	1	1	0	1	0	0	1	0	0	0
R/M	w	MOD	REG	D	R/M	w	D	R/M	w	D	R/M

source

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

SS

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

OP

SSOP2808

~~MOV CX,[BX+SI+2246h]~~

A10

a. State the addressing mode of the function.

b. Find the physical address of memory location if DS=2240h,

$$BX = 12\text{FFFH}, SI = 2346h \rightarrow (DS \times 10) + (BX + SI + 2246h)$$

c. Find the 3rd highest and 3rd lowest.

d. Find the machine code of the instruction.

(88 B4 43 8B)

100010 0010 1011010100  
opcode D W MOD REG R/M

01000011

10001011

MOV [ ] , DH  
 $\Rightarrow SI + 8B93$

Date: 27-10-22

CSF-341

\* \* \* MOV AX, [BP + 2740]  
SP  
DSX10 + BP + 2290 } Exceptional  
→ -

DTP → Parallel and both can take input

min mode - solo

CLK → Synchronization (duty cycle indicates complete high period)

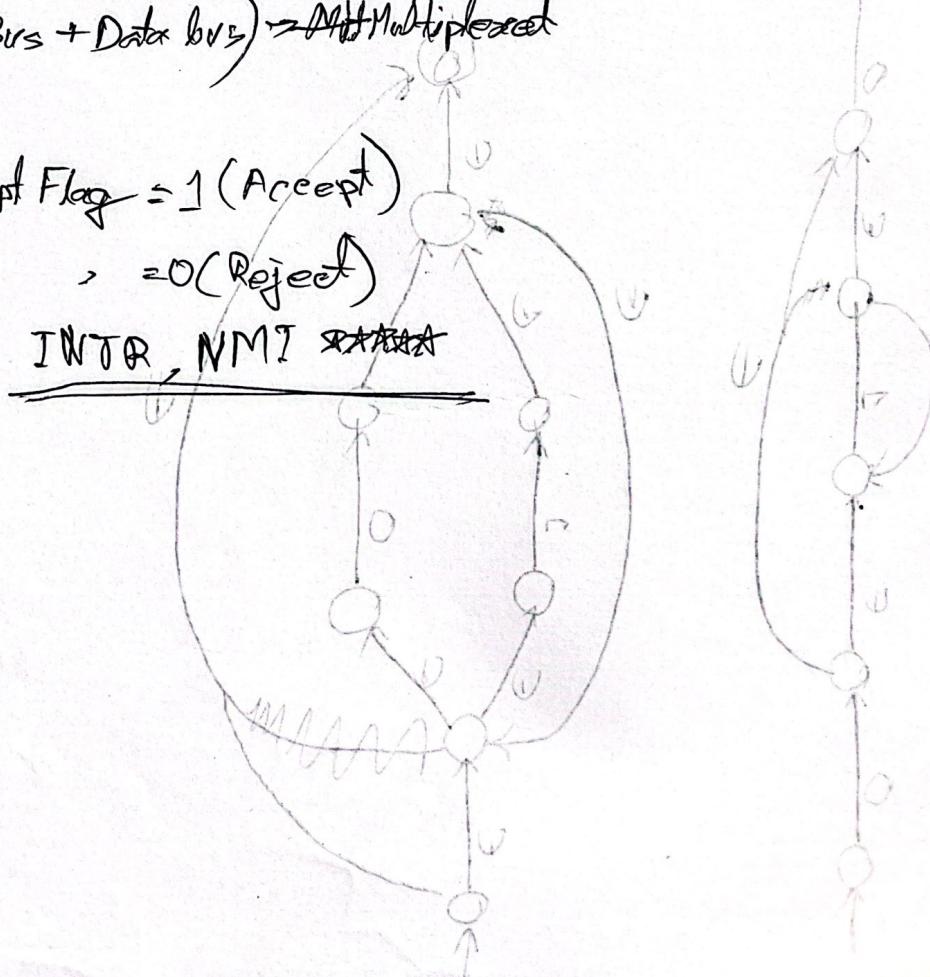
" active "

(Address Bus + Data bus) → Mux Multiplexed

S<sub>5</sub> → Interrupt Flag = 1 (Accept)

→ = 0 (Reject)

INTR, NMI



$\frac{123456}{\text{OpCode}}$   $\frac{7}{D}$   $\frac{8}{n}$   $\frac{9,10}{\text{MOD}}$   $\frac{11,12,13}{\text{REG}}$   $\frac{14,15,16}{R/M} =$   
 Details:  
 1. D  
 2. n  
 3. MOD  
 4. REG  
 5. R/M

MOV AX, [BX + SI + 2246h] SVM

100010101010000000  
Transliteration test

01000110 00100010 CSEB

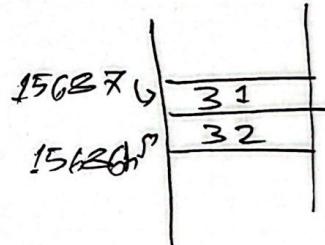
VA & Address

JMP [BX]

DS = 1234h

BX = 3346h

CS = 2200h



$$PA = DS \times 10 + BX$$

$$IP = 3132h$$

$$= 15686h$$

$$PA \text{ of that instruction} = CS \times 10 + IP$$

$$= \underline{\hspace{2cm}}$$