

Microprocessor : 8086, 8088

What is a core?

chip, port no.

interrupt: give address that interrupt request.

priority interrupt

→ segment  
→ offset

flag: priority flag

Q2: 1. Assembly language programming & organization of IBM PC.

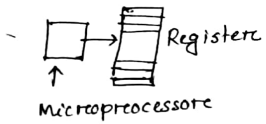
→ Ytha Yu

→ Charles Marcot

2. The Intel microprocessors —  
Barry B Brey.

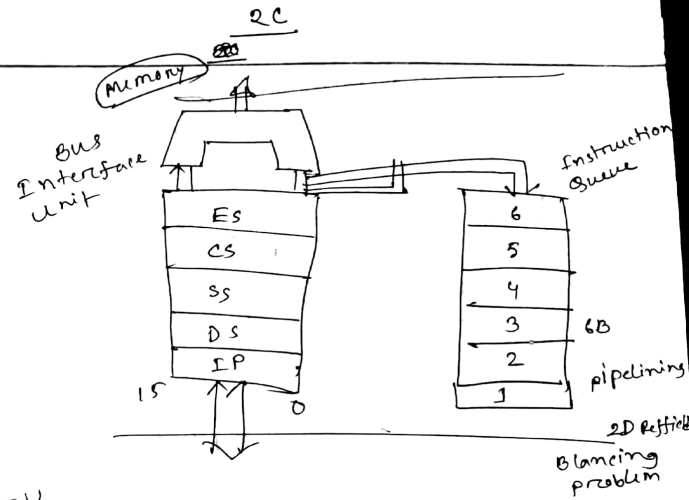
2A

Assembly language? → completely high level or 1 বাক্যে বোঝায়।  
 High level language.  
 \* Assembly to design করতে cost কম লাগবে।



Microprocessor is a chip which is embedded with register & Kash. (Cache memory)

- \* register is a fast device.
- \* Assembler convert assembly language convert করে।
- \* computer register to help ছাড়া কিছু করতে পারে না।
- \* Register এর নাম AX, BX, CX, DX এর ক্ষেত্রে 16 bit
- \* এর মধ্যে 8 bit high byte, 8 bit low byte
- এ ভাবে পারবে।
- যেমন: AX এর ভেতরে AH (8 bit high byte)
- AL (8 bit low byte)



EV

\* Microprocessor এর 2nd Generation করে আরও কিছু architecture follow করে।

\* 8086 microprocessor এর নাম রাখা হয়েছে।

8088.  
 microprocessor 20 difference এর মধ্যে রাখা হয়েছে।  
 numbers first express এর tough, এর ফিল্ড এর  
 ফিল্ড to এর নাম। carry etc.

## Bus Interface Unit

- \* Bus is the data transmit and receive.
- \* address bus (20 bit)
- \* data bus (16 bit)
- \* Control bus (16 bit)
- e.g. address line & address information data line
- \* microprocessor is the size of bus
- \* bus size is the complexity of microprocessor
- \* 20 address memory microprocessor
- \* address interface and RAM

## Execution unit

- ES = Extra segment → extra segment
- CS = Code segment → code segment
- DS = Data segment → data segment
- SS = Stack segment → stack segment

Suppose

12340H  
code segment → 20 bit number

\* as 20 bit number base 16

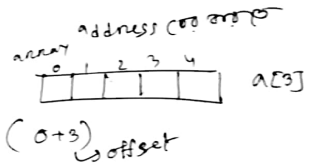
1234H

segment: offset → address

1234H: 000H

4 bit shift and 16

12340H



segment \* 10 + offset

IP: Instruction pointer (16 bit)  
instruction address

Instruction Queue:

Instruction Queue is the instruction to be executed and the 6B reserve and

Instruction Queue for program?

### pipelining:

#### 2B refilled:

2B for execute instruction (2B)

Load (2B)

\* 2B for DS, CS, SS, 16 bit and 16 bit

2 Load and 2B instruction queue (2B)

if ( ) {

}

else {

}

\* Branching problem:

- MODE SMALL
- STACK 100H → stack segment
- CODE → code segment

MAIN PROC



MAIN END  
END MAIN

main { }



return 0;  
}

\* characters, binary

1101 (Decimal)

1101 B (Binary)

1A80H (Hexadecimal)

MOV Dest<sup>n</sup>, Source

ADD Dest<sup>n</sup>, Source

Dest<sup>n</sup> register  
Source register (2B)

A=2 where A variable

MOV AX, 2 → register AX A 2 value

MOV A, AX → register AX for A 2 value

MOV AX, 4

ADD A, AX

result A 2 + 4 = 6

DB  $\rightarrow$  B  
 DW  $\rightarrow$  2B  
 DD  $\rightarrow$  2W  
 DQ  $\rightarrow$  4W  
 DT  $\rightarrow$  10B

variable declaration is same as data type.  
 এখানে হল Data type.

variable declaration:

Var1 DB 20H [20H রাখা]  
 Var1 DB ? [Blank variable]  
 Var1 DB 10H, 20H, 30H [Var1 একটি array হল  
 তখন।]

SUB DS  
 ADD DS  
 MOV DS [সবচেয়ে বামের দিকের স্মারক]  
 XCHG DS E দুইটির value exchange করে  
 INC D [একটি বড় value বৃদ্ধি করে দিবে]  
 DEC D [একটি দিবে]

### Interrupt

Somehow microprocessor to interrupt দিবে  
 আরও কোন কোন interrupt. নতুন function  
 match করে। তবেই (এ অংশটি কেবল করে।)

1	$f_1()$
2	$f_2()$
3	$f_3()$

\* Error msg দেওয়ার interrupt.

INT 21h [21h নম্বরের function এর নিচে  
 আসবে সবকিছু]

INT 21h যখন নিখর তখন microprocessor  
 input/output সুযোগ দেবে।

AH  $\leftarrow$  1 যখন  
 single character input নিবে এবং AL এ

AH  $\leftarrow$  2  
 single character output from DL and  
 copy to AL.

AH  $\leftarrow$  9  
 string output from DX  
 DX should contain the offset of string  
 to be printed.

\* program segment  
offset segment after data  
segment has program segment?

MODEL SMALL / LARGE / MEDIUM  
STACK 100H → stack size (100H)  
CODE

MAIN PROC

MOV AH, 1  
INT 21h

MOV DL, AL  
MOV AH, 2  
INT 21h

MAIN ENDP // main func. as per  
END MAIN ; code total func. as per

\* variable memory or memory area data  
segment. data segment design memory  
area

MODEL SMALL

STACK 100H

DATA

MSG DB 'Please enter a character \$'

CODE

MAIN PROC  
MOV AX, @DATA ; mov ds, @DATA  
LEA DX, MSG ; msg go offset address for code func  
MOV AH, 9  
INT 21h

MOV AH, 1  
INT 21h

MOV DL, AL

MOV AH, 2

INT 21h

MOV AH, 4CH  
INT 21h

MAIN ENDP  
END MAIN

google

Program segment prefix 256 B

\* program run go to segment load area 64K  
program segment go prefix DS, ES a 64K  
area Data segment later read area 256K  
DS, ES a data segment a restore area 256K

3A

4 bit  $\rightarrow$  4004 (1971 intel)  $\rightarrow$  6800 (motorola)  $\rightarrow$  6808 (motorola)

8 bit  $\rightarrow$  8008, 8080, 8085

16 bit  $\rightarrow$  8086, 8088, 80186-286

32 bit  $\rightarrow$  80386, 486

64 bit  $\rightarrow$  Pentium, PI  $\rightarrow$  IV, Core 2 Duo, Dual core;

power PC  
G3, G4, G5  
motorola  
Xeon, Core i3 - i7

Microprocessors are ~~are~~ register of 8 size (micro)  
and 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152, 4194304, 8388608, 16777216, 33554432, 67108864, 134217728, 268435456, 536870912, 1073741824, 2147483648, 4294967296, 8589934592, 17179869184, 34359738368, 68719476736, 137438953472, 274877906944, 549755813888, 1099511627776, 2199023255552, 4398046511104, 8796093022208, 17592186044416, 35184372088832, 70368744177664, 140737488355328, 281474976710656, 562949953421312, 1125899906842624, 2251799813685248, 4503599627370496, 9007199254740992, 18014398509481984, 36028797018963968, 72057594037927936, 144115188075855872, 288230376151711744, 576460752303423488, 1152921504606846976, 2305843009213693952, 4611686018427387904, 9223372036854775808, 18446744073709551616, 36893488147419103232, 73786976294838206464, 147573952589676412928, 295147905179352825856, 590295810358705651712, 1180591620717411303424, 2361183241434822606848, 4722366482869645213696, 9444732965739290427392, 18889465931478580854784, 37778931862957161709568, 75557863725914323419136, 151115727451828646838272, 302231454903657293676544, 604462909807314587353088, 1208925819614629174706176, 2417851639229258349412352, 4835703278458516698824704, 9671406556917033397649408, 19342813113834066795298816, 38685626227668133590597632, 77371252455336267181195264, 154742504910672534362390528, 309485009821345068724781056, 618970019642690137449562112, 1237940039285380274899124224, 2475880078570760549798248448, 4951760157141521099596496896, 9903520314283042199192993792, 19807040628566084398385987584, 39614081257132168796771975168, 79228162514264337593543950336, 158456325028528675187087900672, 316912650057057350374175801344, 633825300114114700748351602688, 1267650600228229401496703205376, 2535301200456458802993406410752, 5070602400912917605986812821504, 10141204801825835211973625643008, 20282409603651670423947251286016, 40564819207303340847894502572032, 81129638414606681695789005144064, 162259276829213363391578010288128, 324518553658426726783156020576256, 649037107316853453566312041152512, 1298074214633706907132624082305024, 2596148429267413814265248164610048, 5192296858534827628530496329220096, 10384593717069655257060992658440192, 20769187434139310514121985316880384, 41538374868278621028243970633760768, 83076749736557242056487941267521536, 166153499473114484112975882535043072, 332306998946228968225951765070086144, 664613997892457936451903530140172288, 1329227995784915872903807060280344576, 2658455991569831745807614120560689152, 5316911983139663491615228241121378304, 10633823966279326983230456482242756608, 21267647932558653966460912964485513216, 42535295865117307932921825928971026432, 85070591730234615865843651857942052864, 170141183460469231731687303715884105728, 340282366920938463463374607431768211456, 680564733841876926926749214863536422912, 1361129467683753853853498429727072845824, 2722258935367507707706996859454145691648, 5444517870735015415413993718908291383296, 10889035741470030830827987437816582766592, 21778071482940061661655974875633165533184, 43556142965880123323311949751266331066368, 87112285931760246646623899502532662132736, 174224571863520493293247799005065324265472, 348449143727040986586495598010130648530944, 696898287454081973172991196020261297061888, 1393796574908163946345982392040522594123776, 2787593149816327892691964784081045188247552, 5575186299632655785383929568162090376495104, 11150372599265311570767859136324180752990208, 22300745198530623141535718272648361505980416, 44601490397061246283071436545296723011960832, 89202980794122492566142873090593446023921664, 178405961588244985132285746181186892047843328, 356811923176489970264571492362373784095686656, 713623846352979940529142984724747568191373312, 1427247692705959881058285969449495136382746624, 2854495385411919762116571938898990272765493248, 5708990770823839524233143877797980545530986496, 11417981541647679048466287755595961091061972992, 22835963083295358096932575511191922182123945984, 45671926166590716193865151022383844364247891968, 91343852333181432387730302044767688728495783936, 182687704666362864775460604089535377456991567872, 365375409332725729550921208179070754913983135744, 730750818665451459101842416358141509827966271488, 1461501637330902918203684832716283019655932542976, 2923003274661805836407369665432566039311865085952, 5846006549323611672814739330865132078623730171904, 11692013098647223345629478661730264157247460343808, 23384026197294446691258957323460528314494920687616, 46768052394588893382517914646921056628989841375232, 93536104789177786765035829293842113257979682750464, 187072209578355573530071658587684226515959365500928, 374144419156711147060143317175368453031918731001856, 748288838313422294120286634350736906063837462003712, 1496577676626844588240573268701473812127674924007424, 2993155353253689176481146537402947624255349848014848, 5986310706507378352962293074805895248510699696029696, 11972621413014756705924586149611790497021399392059392, 23945242826029513411849172299223580994042798784118784, 47890485652059026823698344598447161988085597568237568, 95780971304118053647396689196894323976171195136475136, 191561942608236107294793378393788647952342390272950272, 383123885216472214589586756787577295904684780545900544, 766247770432944429179173513575154591809369561091801088, 1532495540865888858358347027150309183618739122183602176, 3064991081731777716716694054300618367237478244367204352, 6129982163463555433433388108601236734474956488734408704, 12259964326927110866866776217202473468949912977468817408, 24519928653854221733733552434404946937899825954937634816, 49039857307708443467467104868809893875799651909875269632, 98079714615416886934934209737619787751599303819750539264, 196159429230833773869868419475239575503198607639501078528, 392318858461667547739736838950479151006397215279002157056, 784637716923335095479473677900958302012794430558004314112, 1569275433846670190958947355801916604025588861116008628224, 3138550867693340381917894711603833208051177722232017256448, 6277101735386680763835789423207666416102355444464034512896, 12554203470773361527671578846415332832204710888928069025792, 25108406941546723055343157692830665664409421777856138051584, 50216813883093446110686315385661331328818843555712276103168, 100433627766186892221372630771322662657637687111424552206336, 200867255532373784442745261542645325315275374222849104412672, 401734511064747568885490523085290650630550748445698208825344, 803469022129495137770981046170581301261101496891396417650688, 1606938044258990275541962092341162602522202993782792835301376, 3213876088517980551083924184682325205044405987565585670602752, 6427752177035961102167848369364650410088811975131171341205504, 12855504354071922204335696738729300820177623950262342682411008, 25711008708143844408671393477458601640355247900524685364822016, 51422017416287688817342786954917203280710495801049370729644032, 102844034832575377634685573909834406561420991602098741459288064, 205688069665150755269371147819668813122841983204197482918576128, 411376139330301510538742295639337626245683966408394965837152256, 822752278660603021077484591278675252491367932816789931674304512, 1645504557321206042154969182557350504982735865633579863348609024, 3291009114642412084309938365114701009965471731267159726697218048, 6582018229284824168619876730229402019930943462534319453394436096, 13164036458569648337239753460458804039861886925068638906788872192, 26328072917139296674479506920917608079723773850137277813577744384, 52656145834278593348959013841835216159447547700274555627155488768, 105312291668557186697918027683670432318895095400549111254310977536, 210624583337114373395836055367340864637790190801098222508621955072, 421249166674228746791672110734681729275580381602196445017243910144, 842498333348457493583344221469363458551160763204392890034487820288, 1684996666696914987166688442938726917102321526408785780068975640576, 3369993333393829974333376885877453834204643052817571560137951281152, 6739986666787659948666753771754907668409286105635143120275902562304, 13479973333575319897333507543509815336818572211270286240551805124608, 26959946667150639794667015087019630673637144422540572481103610249216, 53919893334301279589334030174039261347274288845081144962207220498432, 107839786668602559178668060348078522694548577690162289924414440996864, 215679573337205118357336120696157045389097155380324579848828881993728, 431359146674410236714672241392314090778194310760649159697657763987456, 862718293348820473429344482784628181556388621521298319395315527974912, 1725436586697640946858688965569256363112777243042596638790631055949824, 3450873173395281893717377931138512726225554486085193277581262111899648, 6901746346790563787434755862277025452451108972170386555162524223799296, 13803492693581127574869511724554050904902217944340773110325048447598592, 27606985387162255149739023449108101809804435888681546220650096895197184, 55213970774324510299478046898216203619608871777363092441300193790394368, 110427941548649020598956093796432407239217743554726184882600387580788736, 220855883097298041197912187592864814478435487109452369765200775161577472, 441711766194596082395824375185729628956870974218904739530401550323154944, 883423532389192164791648750371459257913741948437809479060803100646309888, 1766847064778384329583297500742918515827483896875618958121606201292619776, 3533694129556768659166595001485837031654967793751237916243212402585239552, 7067388259113537318333190002971674063309935587502475832486424805170479104, 14134776518227074636666380005943348126619871175004951664972849610340958208, 28269553036454149273332760011886696253239742350009903329945699220681916416, 56539106072908298546665520023773392506479484700019806659891398441363832832, 113078212145816597093331040047546785012958969400039613319782796882727665664, 226156424291633194186662080095093570025917938800079226639565593765455331328, 452312848583266388373324160190187140051835877600158453279131187530910662656, 904625697166532776746648320380374280103671755200316906558262375061821325312, 1809251394333065553493296640760748560207343510400633813116524750123642650624, 3618502788666131106986593281521497120414687020801267626233049500247285301248, 7237005577332262213973186563042994240829374041602535252466099000494570602496, 14474011154664524427946373126085988481658748083205070504932198000989141204992, 28948022309329048855892746252171976963317496166410141009864396001978282409984, 57896044618658097711785492504343953926634992332820282019728792003956564819968, 115792089237316195423570985008687907853269984665640564039457584007913129639936, 231584178474632390847141970017375815706539969331281128078915168015826259279872, 463168356949264781694283940034751631413079938662562256157830336031652518559744, 926336713898529563388567880069503262826159877325124512315660672063305037119488, 1852673427797059126777135760139006525652319754650249024631321344126610074238976, 3705346855594118253554271520278013051304639509300498049262642688253220148477952, 7410693711188236507108543040556026102609279018600996098525285376506440296955904, 14821387422376473014217086081112052205218558037201992197050570753012880593911808, 29642774844752946028434172162224104410437116074403984394101141506025761187823616, 59285549689505892056868344324448208820874232148807968788202283012051522375647232, 118571099379011784113736688648896417641748464297615937576404566024103044751294464, 237142198758023568227473377297792835283496928595231875152809132048206089502588928, 474284397516047136454946754595585670566993857190463750305618264096412179005177856, 948568795032094272909893509191171341133987714380927500611236528192824358010355712, 1897137590064188545819787018382342682267975428761855001222473056385648716020711424, 3794275180128377091639574036764685364535950857523710002444946112771297432041422848, 7588550360256754183279148073529370729071901715047420004889892225542594864082845696, 1517710072051350836655829614705874145814380343009484000977978445108

Stack segment: stack pointers / base pointers

\* offset निर्धारण करके डेटा डालें।

ES: DI } 20 bit physical address निर्धारित करें।  
DS: SI }

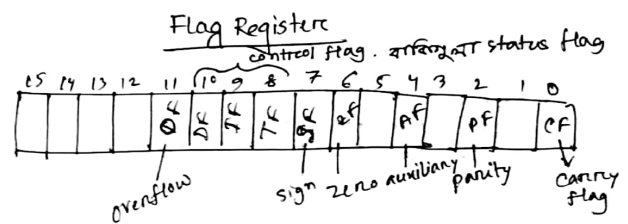
\* Segmentation से benefit कि?

+ 20 bit physical address से 16 bit register  
का उपयोग करें।

### Assignment

assembly respect } Relocatable Code  
Absolute machine Code  
difference, Advantages, Disadvantages

3C



CF ← If carry out from msb or borrow into msb

PF ← Even no. of 1 in low byte

AF ← If carry out from 3rd bit or borrow into 3rd bit

ZF ← If zero result

SF → If sign bit 1.

अगर case 2 में 1 है तो flag 1 होगा 0 नहीं।

Control flag:

जिस operation control से control flag.

Direction flag:

Left to right or right to left.



AX = 1111 1111 1111 1111

0X = 1111

AX = 1111 1111 1111 1110  
BX = 1111 1110 1111 1111

AX = 1111 1110 1111 1111  
BX = 1111 1110 1111 1111

AX = 1111 1110 1111 1101  
AH = 1111 1110 1111 1101  
AL = 1111 1110 1111 1101

MOV AX, FFFE  
MOV BX, FFFF

CF = 1  
PF = 0  
AX = 1  
ZF = 0  
SF = 1

3D

conditional jump instruction  
misaligned representation.

Jump Instructions

cmp D.S

> jump above → JA → CF=0, ZF=0  
< jump below → JB → CF=0  
≥ jump above equals → JAE → CF=2  
≤ jump below equals → JBE → CF=1, ZF=2

JMP Destination Level

as 10 ?

JA true then skip the current operation if CF=0, ZF=0 then

Sub AX, 10

JA ~~~~~

destination change then error. after skip the sub use  
if not cmp use error 2.

cmp AX, 10

JA ~~~~~

if error then. first register AX is store  
then AX  
→ e is got error.

JA label

Ex: JA myPos

myPos:  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

JMP Destination Level:

condition at present jump error and error  
then jmp Destination level.

### benefit of JMP:

conditional jump is not same segment as  
arrange 220 1 flag JMP to same segment 220 220.

a 220 start 220 10 67 characters print 8

CODE

MOV DL, 65 → PRINT:

MOV AH, 2

INT 21h → CMP DL, 75

ADD DL, 1 JA EXIT

JMP PRINT

EXIT:

MOV AH, 4CH

INT 21h

MAIN ENDP

END MAIN

### Assembling

MOV AH, 1

INT 21h

AL ← C1 AL 20 value

MOV BL, AL BL 20 value

INT 21h

AL ← C2

MOV AH, 2

CMP AL, BL

JA Print (AL > BL)

MOV DL, BL

Jump Printit

Print:

MOV DL, AL

INT 21h

ATMEGA 32 — 200/300K

4C

### Overflow(flag)

FFFF	1111	1111	1111	1111	→ -1
+0001	0000	0000	0000	0001	→ +1
	1	0000	0000	0000	

Carry flag

→ unsigned 2's overflow 220 220

2's - 10  
01  
10  
10  
10

11  
00 (2's)  
1  
01

00 → 0  
01 → 1  
10 → -2  
11 → -1

- ① signed overflow
- ② Unsigned overflow
- ③ Both
- ④ None

unsigned 11 → 3  
01 → 1 overflow  
100 → 4

signed 11 → -1  
01 → 1  
00 → 0 not overflow

- carry flag 1 means unsigned overflow
- signed integer 2 limit for limit cross means signed overflow.

overflow = 1 means signed.

msb is not same as, and that means it's different same

signed overflow | msb-in  $\neq$  msb-out

Signed overflow:

7FFF  $\rightarrow$  0111 1111 1111 1111

7FFF  $\rightarrow$  0111 1111 1111 1111

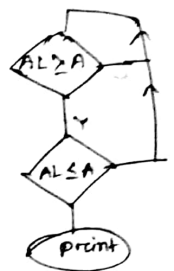
JG/JNLE | signed jump Jc  
JGE/JNL  
JL/JNGE  
JLE/JNG

CF = 1 or 0 depend.

4D

CODE  
MOV AX, @DATA  
MOV DS, AX  
try-Again:  
MOV AH, 1  
INT 21h ; same program

COMP AL, 'A'  
JGE CHK-2 ; True then execute it  
JMP try-Again



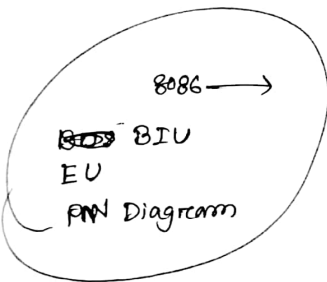
CHK-2:  
COMP AL, 'F'  
JLE PRINT  
JMP try-Again

PRINT:  $\rightarrow$  MOV BL, AL  
MOV AH, 2  
MOV DL, AH ; MOV DL, '1' ; 01010001 A-F is 01010001 something 0-4  
INT 21h  
SUB BL, 'A' ; A is 01000001 B is 00000001  
MOV AL, BL  
INT 21h.

# LOOP

Label  
=  
=  
loop label

mov cx, 10 / cx counter register. count register loop karta hai.



# SA

AND destination, source } mask design ka  
OR destination, source }  
XOR destination, source.  
SHL destination, CL  
SHR destination, CL  
NOT destination, source

mask kaha hai? filtering karne ke liye.  
mask kaha hai? filtering karne ke liye.  
mask kaha hai? filtering karne ke liye.

↓  
1 0 1  
?  
00 → 0  
01 → 1  
10 → 2  
11 → 3  
100 → 4  
101 → 5  
position 0 aur 1  
check.

set karne ke liye OR. (set karne ke liye)

0000 1101  
0000 1011  
-----  
0 2  
OR AL, 02h

XOR →

MOV AL, 02h

AL register value clear karke dena hai

AND, XOR karke karke karke hai

XOR AL, AL (karke AL, AL karke XOR karke hai)

AND

'a' → 41H → 01000001  
 01H → 00000001

'A' → 61 → 01100001

(OR)

OR, 02h  
 OR AL, 20h

a → A = 41 → 61

b → B = 42 → 62

100 = 4

110 = 6

SHL:

<<

1 bit shift karke

SHL destination, 1

\* CL 9 karke 10 bit karke shift

AL 10111110

1 bit shift karke

AL 01111110

Left most bit carry flag 9 karke karke hai  
 JE karke carry flag check karke karke hai

SHR:

ODD even check karke hai

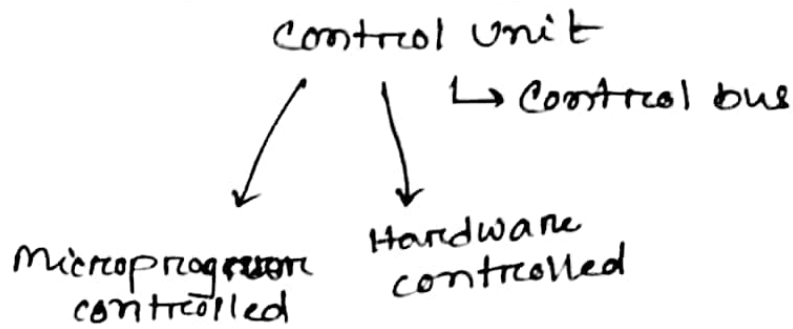
karke right shift karke JE karke of check karke hai

AL 10111110  
 4 3 2 1

NOT:

10111110  
 01000001

MOV AL, 0Fh  
 01h  
 or 10111110  
 karke most left  
 bit karke  
 karke karke karke



## RISC & CISC architecture

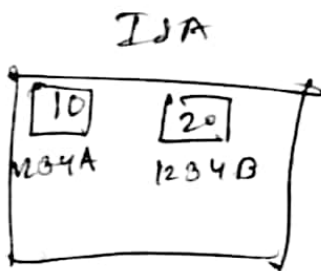
RISC: Reduced Instruction Set Computer  
 CISC: Complex Instruction Set Computer

basic difference

ISA → Instruction Set Architecture.

~~MOV AX, 1234~~ `MOV AX, 1234`

memory address 1234



2 No way

RISC  
 LOAD AX, [1234A]  
 LOAD DX, [1234B]  
 ADD AX, BX  
 STORE [1234A], AX

\* single cycle

H.W 20 easy.

1 No. way

CISC ADDITION 1234A, 1234B

H.W 20 tough

\* multiple cycle

\* Basic for instruction set common.

IF: Instruction Fetch

DE: Decode

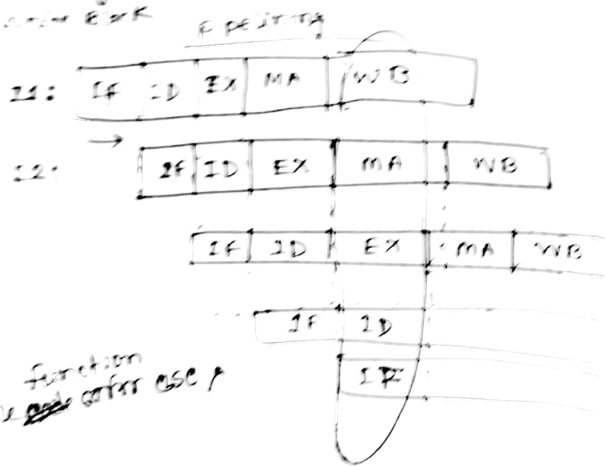
EX: Execute

MA: Memory Access

WB: Write Back

pipelining

→ RAM operation



function available after cycle 1

Power consumption:

not available after cycle 1

RAM to array connection (after 2nd cycle)

RISC to code memory (after 2nd cycle), connection (after 2nd cycle)

2nd RISC to array

RISC to register (after 2nd cycle)

RISC vs RISC

Time =  $10 \text{ PT}$  → Clock period (s)

No. of instructions/program

cycle/inst

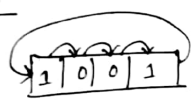
clock period/cycle

CT Flag register এর  
সমস্ত বার্তা।

SD

ROR → Rotate Right  
ROL → Rotate Left  
RCR →  
RCL →

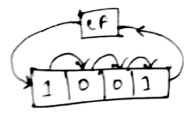
ROR:



ROR AX, CL

একটি ডাটা shift হবে। ডাটার ধ্রুপদী ডাটা হবে। আরও একটা  
সহিত CF এ ফলাফল  
SHR দিলে চারটি ডাটা ডাটা ডাটা ডাটা ডাটা  
replace হবে ডাটা।  
যদি ROR ডাটা ডাটা ডাটা।

RCR:



CF এ ডাটা ডাটা ডাটা ডাটা ডাটা। ডাটা ডাটা CF এ  
ফলাফল।

AX → 16 bit ডাটা ডাটা। 16 bit input ডাটা ডাটা।

AL = 01 ডাটা ডাটা 1 ডাটা ascii ডাটা।  
AL = 31h or 00110001

DX = 10110... ডাটা Input

AL = 00110001  
AND 00000001  
00000001

DL = 00000000  
AL = 00000001  
(00) 00000001

XOR DL, DL ; DL 0 ডাটা ডাটা  
MOV AH, 1  
INT 21H → MOV CX, 8  
AND AL, 01h  
SHL DL, 1  
OR DL, AL

8 times  
for each  
bit in  
2000



Hexadecimal:

0-9 → AL = 00110001  
A-F → AL = 00001111  
AL = 00000001

(OR) DL = 00000000  
AL = 00001001  
DL = 00001001

(4 bit shift) DL = 00010000  
AL = 00000001  
DL = 10010001

3192

FOR 0-9

XOR DL, DL  
MOV AH, 1  
INT 21h

if (0-9)

AND AL, 0Fh  
SHL DL, 4  
OR DL, AL

FOR A-F

0ABACH

4 1 h  
AL = 0100 0001  
SUB 0011 0111 h  
3 7 h  
AL = 0000 1010

else AND AL, 3  
SUB AL, 37h

65-41h  
55 → 37h  
10

7C

Stack Operation



sp → stack pointer

16 bit operation

PUSH source  
PUSHF → Flag registers  
POP destination  
POPF → Flag registers  
reload 16 bit

Function:

NAME PROC

MAIN PROC

MAIN ENDP

RET  
NAME ENDP

\* myfunc after func.

myfunc PROC

parameter passing

RET  
myfunc PROC

\*\* stack 2000 concept  
for no allocate 2000 deallocate 2000

CALL myfunc.

sub

\* Calling 2000 (2000 address) 2000

multiplication

$$A \times = \text{Source} \times A_L$$

mul source → reg

### IMUL Source

upper half

↳ lower half-

signed operation 20 to Integer multiplication

$$Ax = 1111 \dots 1 \quad B$$
$$BX \rightarrow \cdot$$

- \* Pipeline
- \* Multiplication, AND, OR, XOR, SHL, SHR, ROR, ROL, NOT
- \* Stack
- \* Overflow
- \* RISC, SISC
- \* Masking
- \* Hex

\* pipeline

\* multiplication %/.

\* Stack

\* Overflow  
on SIS

\* RISC / 1010-  
- king

\* mask

\* HCLX

8(c)

$$100000 \times 5\% \times C_s \times t_c + 100000 \times (100 - 5)\% \times C_e \times t_e$$

Decimal input:

0-9

134 একটি number দিতে চাচ্ছি।

BY AX

$$0 \times 10 + 1 = 1$$
$$1 \times 10 + 3 = 13$$
$$13 \times 10 + 4 = 134$$

Input:  $AH \leftarrow 1$ ,  
 $INT \leftarrow 0$

AND A X,000 Fh

```
mov DX, AX; 1 (Input)
```

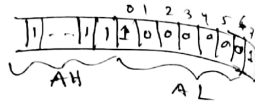
mov ~~AX~~, 10

MUL BX; Result in AX

ADD AX, DX

MOV BX, AX

DIV divisore



~~CEW~~ AL 20 0 CO sign bit  
ONE 1 SO AH 2 WORKING  
1 20 20 10000000

### ADDRESSING MODE

## Based/Indexed Addressing mode

[Offset + displacement] or

offset [displacement]

অন্যভাবে স্ট্রিং, array mode for pointer address  
যদিও এটি 1 word array এ হতে পারে।

memory [AX]

MOV AX, 0

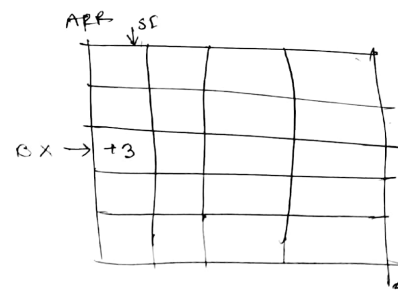
MOV DI, myarray [AX]

ADD AX, 2

ARR [R] [C]

\* Base no. register এর value বদলাবে  
নয়!

↓ word type  
ARR [BX] [SI]



MOV CX, 3 → CX 3 বারের 3 টি word বদলাবে।  
XOR SI, SI → SI 0 হলে loop চলেবে না। তাহলে CX  
MOV BX, 16 → change SI এর value। তাহলে jump দিয়ে  
loop control করে।

ADD ARR [BX] [SI], CX

ADD SI, 2

```
MOV AX, STUDENT
MOV BH, ROLL
```

LABEL

```
STUDENT LABEL WORD
ROLL DB 153001
MARKS DB 30
```

PTR

```
MOV BYTE PTR [BX], 10
WORD
```

BYTE pointer with byte type as value for

```
MSG DB 'LET $'
LEA MSG
```

mov AX, BL ? problem  
AX = 16  
BL = 8 bit

XLAT

Translate or convert using XLAT instruction. It takes 2 arguments. One is instruction and other is table address.

DX ← table index

AL ← Byte value to be converted

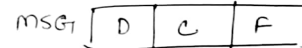
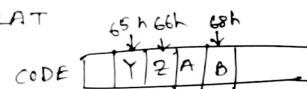
XLAT (DX, AL) → value of DX depend on register and



LEA DX, CODE

MOV AL, 01h

XLAT



CODE array array.  
BX of array & index value  
AL 20 value for

01h CODE is 20h  
20h 20h 1 20h 20h  
value 20h 20h 20h  
replace 20h 20h

2 table for CODE table  
20 value for replace  
20h

LEA MSG,

LEA SI, MSG

MOV AL, [SI] (D for AL value 1 MSG)

XLAT

MOV DL, AL

MOV AH, 2

INT 21h

INC SI

BAD

DUP → Duplicate data ko

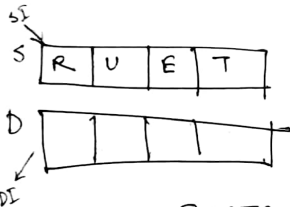
CODE DB 65 DUP(' '), 'YZAB...

String

Byte

• movsb → mov byte from source to mov byte  
 movsw → mov word from source to mov word

ES:DI → Destination array  
 DS:SI → Source Array



SI value D to mov array

MOV AX, @DATA  
 MOV ES, AX [DATA segment as Address  
 CLD  
 LEA SI, S  
 LEA DI, D  
 DF  
 0 → SI/DI (array name)  
 1 ← SI/DI (array index)  
 CLD → STD

LODSB/W AL/AX DS:SI  
 STOSB/W ES:DI AL/AX

\* LODSB takes AL register LODSW takes AX register

ES:DI [A B C D] DS:SI [C D]  
 \* ES:DI points to one byte and DS:SI points to one byte  
 REPE REPEZ SCANB/W  
 REPEZ CMPSB/W

ES:DI [A B C D] AL/AX  
 target string

AL to compare 1st byte  
 match then ZF=1  
 for next A-A=0  
 ZF=1

CMPSB/W:  
 SCAN array and compare with array  
 array and array are same then

REPE (repeat while equal)  
 REPNE (while not zero)

Assembly is  
 hardware related

Interrupts:

Hardware Int →  
 Software Int

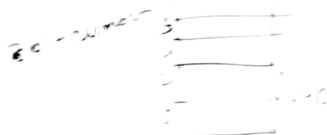
Exception or an error in the program  
 For eg code is exception to the  
 CPU hardware.

Register

2<sup>nd</sup> reg → 2<sup>nd</sup>

2<sup>nd</sup> reg → 2<sup>nd</sup>

2<sup>nd</sup> reg



2<sup>nd</sup> reg

f 21()

}

f 22()

...

UN

→ ...  
 → ...



### Timing of cost

For read/write cycle,  
Co-processor

max → Multiprocessor  
Min

8086 is not co-processor like other  
micro.

8086 max mode is 20 pins

mode check pin 1 pin 16  
data bus mode 2 pin 16

8087 → math co-processor  
mathematical unit 8086 for micro

8086 is block diagram

Latch →

Transceiver →

memory →

I/O →

Assignment on —

Latch, Transceiver, Clock generator

for microprocessor as unit  
and also as unit for

Latch → D FF logic design

ALE

ADD/status

ADD/Data

$\overline{WR}$

$\overline{DEN}$

$PT/\overline{R}$

Minimum  
mode write  
operation

→ separates  
data from  
address/  
data line  
(transceiver  
vers)

Maximum mode 20 operation,  
read 40 ns write cycle 20 ns  
Self-study

max mode

→ Co proc

→ Bus controller



### DMA operation:

Direct memory Access.

For device access data information without microprocessor get disturbed data is.

1. Microprocessor is not in control of DMA operation.
  2. Micro. p. acknowledgement given.
  3. Control given DMA controller for given data.
- अधिकतम DMA controller है।

5. Form to performance का अधिक  
D form to का? D form to का का?  
का का? quality के का का?

Interrupt:  
I/O, INTR

Co-processor → block diagram

RISC

CISC

Direct video ram X का का

Micro controller to use for video data is.