Chapter-1 (Intel Microprocessors)

The mechanical Age:

- calculation with machine idea starts to 500 BC.
- Babylonians invented Abacus ⇒ first mechanical calculation, strings of beads perform calculation.
- 1642, Blaise Pascal invented a calculator constructed of gears and wheels, each gear contained 10 teeth.
- second gear principle use in automobile odometer. Basis of all mechanical calculator.
- 1823, Royal Astronomical society commissioned Babbage to build Programmable calculating machine to generate Royal Navy navigational tables.
- Babbage began to invent "Analytical Engine" a steam-powered mechanical computer stored a thousand 20 digit decimal numbers where variable Program could modify function to perform calculation. tasks where input through punched machine. This used in 1950's and 1960's computers.
  punched cards as input to a weaving machine started by Joseph Jacquard, 1801.
- Analytical Engine required more than 50,000 machined parts. they could not be made with enough precision to allow his engine to function reliably.

The Electrical Age:-

- 1800s saw invent of electric motor by Faraday and Boomar Brain introduced electronic calculator in 1970.
- Monroe maked desktop model four function in size of cash register
- In 1889, Herman Hollerith developed the punch card for storing data and also developed mechanical machine that counted, sorted and collaborate information stored on punched cards. He use this system to store and tabulate the 1890 census information.
- IBM (international business machines) formed by him. The punched cards used in early computer systems are often called Hollerith cards.
- 12 bit code used on a punched card is called Hollerith code.
- Zuse constructed a mechanical version of his system 1936 and constructed first electromechanical computer system Z2. which gives honours him for work in area of digital electronics.
- First electronic computer was invented by Alan Turing a special purpose computer to break secret of German military codes. the system was vacuum tube and machine called colossus. it was not programmable.

- 1800s saw invent of electric motor by Faraday and Boomar Brain introduced electronic calculator in 1970.
- Monroe maked desktop model four function in size of cash register
- In 1889, Herman Hollerith developed the punch card for storing data and also developed mechanical machine that counted, sorted and collaborate information stored on punched cards. He use this system to store and tabulate the 1890 census information.
- IBM (international Business machines) formed by him. The punched cards used in early computer systems are often called Hollerith cards.
- 12 bit code used on a punched card is called Hollerith code.
- Zuse constructed a mechanical version of his system 1936 and constructed first electromechanical computer system Z2. which gives honours him for work in area of digital electronics.
- First electronic computer was invented by Alan Turing a special purpose computer to break secret of German military codes. the system was vacuum tube and machine called Colossus. it was not programmable.

First general purpose programmable electronic computer system developed 1946.

Electronic Numerical Integrator and calculator (ENIAC)

A machine with 17000 vacuum tubes, 500 miles of wiring weighed over 30 tons and about 100000 operations per second. It was programmed by rewriting its circuits and required frequent maintenance.

- 1947 → John Bardeen, William Shockley → develop transistor
- 1958 → Jack Kilby → Integrated circuit
- 1960 → → resistor to transistor logic.
- 1971 → intel corporation → First microprocessor.

Intel engineer Federice, Ted and Stan developed 4004 microprocessor

Programming Advancement:

First machine language was constructed of ones and zeros using binary codes. Stored in the computer memory system as group of instruction called a program.

- John Von Neumann first modern person to develop a system to accept instructions and store them in memory.
- UNIVAC available in 1950, assembly language was used to simplify entering binary code.

general purpose programmable electronic computer
developed 1946.

e Numerical Integrator and calculator (ENIAC)
ine with 17000 vacuum tubes, 500 miles of wire weighed 30t
and about 100000 operations per second. It was programmed
writing its circuits and required frequent maintenance.

→ John Bardeen, William Shockley → develop transistor
→ Jack Kilby → Integrated circuit
→ → resistor to transistor logic.
→ intel corporation → First microprocessor.

engineers Federico, Ted and Stan developed 4004 microprocessor

amming Advancement:

machine language was constructed of ones and zeros using
y codes. Stored in the computer memory system as
of instruction called a program.

n Von Neumann first modern person to develop a system
accept instructions and store them in memory.

NIVAC available in 1950, assembly language was used to
plify entering binary code.

- First general purpose programmable electronic computer system developed 1946.

Electronic Numerical Integrator and calculator (ENIAC)

a machine with 17000 vacuum tubes, 500 miles of wires weighed over 30 tons and about 100000 operations per second. It was programmed by rewriting its circuits and required frequent maintenance.

- 1947 → John Bardeen, William Shockley → develop transistor
  1958 → Jack Kilby → Integrated circuit
  1960 → → resistor to transistor logic.
  1971 → intel corporation → First microprocessor.

Intel engineers Federice, Ted and Stan developed 4004 microprocessor

Programming Advancement:

First machine language was constructed of ones and zeros using binary codes. Stored in the computer memory system as group of instruction called a program.

- John Von Neumann first modern person to develop a system to accept instructions and store them in memory.
- UNIVAC available in 1950, assembly language was used to simplify entering binary code.

1957 Grace Hopper developed first high level programming language FLOWMATIC. Same year, IBM developed FORTAN (FORmula TRANslator) which allowed to use formula to solve math. similar language, ALGOL (Algorithmic language)

First successful programing language for business application COBOL (Computer Business oriented Language). Another business language RPG (Report Program Generator) allows programming by specifying form of the input, output and calculations.

BASIC is the easiest programing language to learn used in many computer systems.

BASIC used for 80% of programs written by personal computer users. C# may replace C/C++, JAVA even BASIC.

Scientific community → primarily use C/C++ → occasionally PASCAL, FORTRAN
Embedded system developers → 60% C → 30% assembly → 10% BASIC, JAVA

We use Assembly in specific video game. And some newer parallel instructions found on pentium and corez microprocessors only programmable in assembly.

ADA used by department of defense named after Augusta Ada Byron.

Micro Processor Age

- A 4 bit microprocessor programmable controller on a chip intel 4004. Addressed 4096, 4 bit wide memory locations.
- A bit is a binary digit with a value of one or zero.
- 4 bit-wide memory location often called a nibble. The 4004 instruction set contained 45 instruction.
- 4004 fabricated with then-current-state-of the art P channel MOSFET technology. Executed instruction at 50 KIPS (kilo instruction per second) slow than ENIAC. But weighed only less than an ounce.
- 4 bit microprocessor debuted in early game system and small control system.
- It use in low-end application such as microwave ovens and small control systems. Calculators still based on 4 bit BCD (binary coded decimal) codes.
- Intel released 8 bit extended version of 4004 microprocessor. Addressed expanded memory of 16K bytes. A byte is 8 bit wide binary number and K is 1024. Additional instruct contained 48 total (8008). But it has small memory size, slow speed and instruction set limitation.
- Intel introduces 8080 microprocessor in 1973 first of the modern 8 bit microprocessors.

Early 8 bit microprocessors:-

Manufacturer	Part number
Fairchild	F-8
Intel	8080
Motorola	MC6800
National semiconductor	IMP-8
Zilog	Z-8

- IBM still produce motorola style microprocessors.
- Zilog still produce microcontrollers and embedded controllers.

Specialty of 8080

1. 8080 addressed four times more memory. 64K bytes vs 16K for 8008
2. Takes less than 1ox time. Addition take 2.0 μs and in 8008 addition take 20 μs.
3. TTL (transistor-transistor logic) compatible.
4. Interfacing made easier and less expensive.

The MITS - Altair 8800, was released in 1974. BASIC language interpreter for Altair developed in 1975 by Bill gates founder of Microsoft corporation.

Assembler program for Altair 8800 was written by Digital Research corporation.

The 8085 Microprocessor:

In 1977 intel corporation introduced updated version of the 8080 - the 8085. Slighty more advanced at higher speed then 8080. 769,230 instructions per second vs 500,000 per second on (8080)

Main advantages of 8085 were internal clock generator and system controller and higher clock frequency. Higher level of component integration reduced the 8085's cost.

- Intel has sold over 100 million of 8085. Zilog sold Z-80 around 500 million.
- Over 700 million microprocessors execute 8085/Z-80 compatible code.

The Modern Microprocessor:

In 1978 intel released 8086 then 8088. Both are 16 bit microprocessors. Execute instruction in 400ns. (2.5 millions of instructions per second). They addressed 1M byte of memory. 16 times more memory than 8085. 1M byte memory contains 1024K byte sized memory locations.

Another feature was a 4 or 6 byte instruction cache or queue that prefetched instructions before they were executed. Basis for much larger instruction caches found in modern microprocessors.

instruction set included multiply and divide instructions were missing. Number of instructions increased from 45 on 4004, 246 on 8085 and over 20,000 variations on 8086 and 8088.

- 16 bit microprocessor provide more internal register storage allow software to be written more efficiently evolved to meet need for larger memory systems.
- In 1981 IBM chose 8088 in its personal computer.
- The 16 bit processor provide 1M byte of memory for spreadsheets, word processors, spelling checkers etc.

The 80286 Microprocessor

The 8088 microprocessor could use only 1MB (megabyte), it was problem for the bigger program like database. So intel introduced 16 M-Byte memory microprocessor named 80286. Instruction sets are almost identical except few additional instruction. 80286 clock speed increase and execute some instruction in less than 250 ns. 4 million instruction per second (4MPS)

80386 → 32 bit Microprocessor

It could handle 32 bits of data at a time Accept 32 bit memory address. 80386 address 4 GB (gigabytes) of memory using 32 bit address bus.

1 GB = 1024 M, 2^{30} locations that is enough to store 1 million type written, double-spaced pages of ASCII text.

1 bit can handle 2 address

Version	bus	address	memory	work.
80386SX	16 bit	24 bit	16 MB	X
80386SL	16 bit	25 bit	32 MB	X
80386LC				had internal cache for faster data handling.
80386EX	24 input/output line.	26 bit bus address, 16 bit data bus		Built in DRAM controller, chip selection logic, Printer, robotics.

Bus

A bus is like highway inside the computer that carries data, address and control signal between processor, memory and other device.

① data bus ② Address bus ③ Control bus.

Graphics Needs

Modern GUI is called (what you see is what you get). It use a lot of memory and ~~space~~ speed. A screen has 640 pixel per line containing 480 lines.

- real numbers (like 1.2, 3, 0.5) use 32 bits.
- with 8 bit data bus, it takes 4 cycles to move one 32 bit number.
- with 32 - bit bus : it takes only 1 cycle → much faster.

#

80386 has hardware to manage memory where 8086, 8088 had only software. 80386 is much more compatable, with 16 bit program so they can still run.

The 80486 Microprocessor

In 1989 Intel released a highly integrated microprocessor which combine 80386 Processor, math coprocessor 80387 and an 8 KB cache memory to speed up access.

Performance

About half the instructions ran in 1 clock cycle instead of 2. At 50 MHz that means some instruction finished in 25 ns. Performance reach 50 MIPS (50 million instructions per second). It made 50% faster than 80386 at same clock speed.

double-tripple clock version

clock	Processor	memory	work
double clock	66 MHz	33 MHz	
Triple clock	100 MHz	33 MHz	same 60 MHz Pentium

- Newer version had 16 KB cache instead of 8 KB means more speed.
- Over drive Processor replace chip (33 MHz chip with 66 MHz double clocked chip) without changing the whole system.

CS

PU has two main parts → internal clock speed (how fast CPU does calculation), bus speed (how fast CPU talks to memory).

at 100MHz, memory 33 MHz

n a CPU ~~with~~ 100 MHz & ~~but~~ memory ~~can~~ 33MHz

ding RAM and Bus faster is expensive so we make fast the PU. so it works many times faster thus CPU process data arly, so even if memory is slower the CPU works fast without aiting too much.

fference between 8008 and 8080 PP.

Inter-8008	Intel-8080
erformance is slower nd more limited	(I) Performance is much faster and more capable
Memory is MAX 16KB	(II) Memory is max 64 KB
Address bus width 14 bit	(III) Address bus width 16 bit.
Clock bit speed 0.5MHz	Clock bit speed 2MHz 4x faster than 8008.

(TNSQ)

Intel Pentium (P5 / 80586)

- introduced 1993. code name P5 or 80586. Intel chose a name instead of a number because numbers could not be copyrighted.
- initial version on 60 MHz or 66 MHz. Later two versions were introduced 120 MHz and 133 MHz (double clocked) and another fastest version 233 MHz (3.5x clocked)

Cache:

derived from 80386 and 80486 but increased from 8 KB to 16 KB. 8 KB for instruction cache and 8 KB for data cache.

Memory and data bus:

Memory addressing capacity upto 4 GB. Data bus width increase to 64 bit. Bus transfer speed 60 MHz or 66 MHz depending on version.

wider bus benefit

1. support double precision floating point numbers.
2. Enables high speed vector graphics.
3. improves video and virtual reality performance.
4. Capable a full from video at 30 Hz.

(P24T) pentium overdrive was designed as an ideal upgrade from 80486 system.

Dual integer execution ⇒

1. Ability to execute two independent instruction per clock cycle. this due to dual integer pipeline u-pipe and v-pipe.

RISC (Reduced instruction set computer (simple, faster, instructions

~~Risc of intel~~ ⇒ two integer unit, one floating point unit, Boosted performance.
RISC of MAC, IBM

CISC of intel ⇒ superclass tech, improved floating point unit, better instruction throughput.

Pentium Pro (P6) - 1995

21 million transistor; clock speed 150 MHz and 166 MHz. Two level cache. L1 has 16 KB total (8 Kb instruction + 8 Kb data). L2 has 256 KB. Execute up to 3 instruction per cycle using 3 execution engine. It can be conflict still be done in parallel. Optimize for 32 bit code and 36 bit address but; supports upto 4 GB standard, upto 64 GB.

Pentium II and Xeon version:

in pentium II CPU placed on circuit board module and L2 cache was ~~for~~ on board not fast enough to direct integration. Memory bus 100 MHz and require 8 ns SDRAM instead of 10 ns.

SDRAM (Synchronized Dynamic Random Access Memory) it operates with CPU clock.

In Xeon L2 cache size: 512 KB, 1 MB or 2 MB and L1 cache is 32 KB. Compatible with 440GX chipset.

Pentium III

Faster core the pentium II but based on P6 architecture. Clock speed upto 1 GHz. 66 MHz memory bus.

Question-Ans Info

* An abaicus is a mechanical device that can perform arithmetic calculation quickly and refer to counting frame.
* Pascal is the pioneer of mechanical calculator.
* Alan turing invented the first electronic computer, Colossus. It used vaccum tube and use to break german military codes.
* Herman hollerith began designing a machine tabulate census data more efficiently. He invented punched card device to help analyze census data. He used electricity to read, count and sort census data.
* Intel coprocessor, first microprocessor, hollerith card had 960 bits
* Ada is a statically, high level, imperative, object oriented programing language inspired by pascal and other language.
* Von Neumann is the theoritical machine that is used in Program Computer which is the base of all modern computer. It has control unit with Arithmetic logical unit, memory, Control unit, main storage and input/output
* 1977 → 8085 → 100 million caper.
* 8086 → 1M byte, 80286 → 16M byte * 80486 4G byte with 8KB cache memory.
* 1993 → pentium microprocessor, 2000 → pentium 4.

Pentium feature → superscaler architecture, separate data and instruction cache, 64 bit data bus, execution tracing, performance monitoring, internal parity checking.

* most successful programing language, COBOL, FLOWMATIC, FORTRAN ...

BASIC is a programing language

1P = 1024 T, 1G = 1024M * 4G byte memory can store 4000000 type written page

* Windows application programing area 2G byte large.
* DOS (TPA) transient program area has 640K byte memory

* Pentium 4, AthlonXP, Athlon64, Itanium, Itanium 2 have 64 bit data bus.

Core2 = 80486 microprocessor have 4 Gbyte memory.

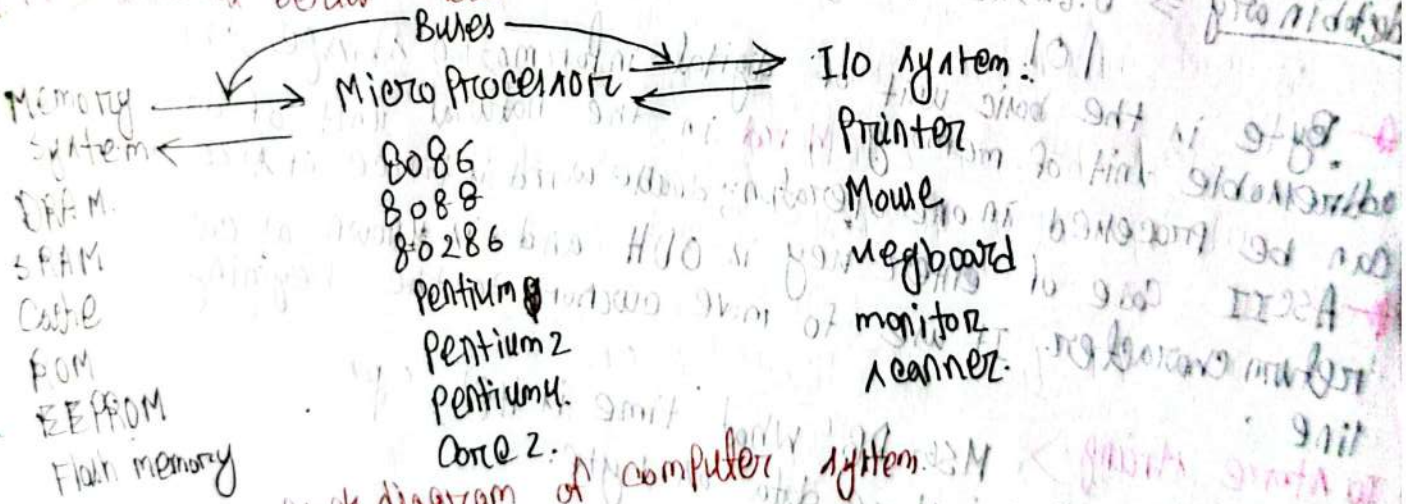
* Memory above first 1Mbyte is called protected or extended memory
* BIOS is the program use by CPU to start computer and manages data flow between computer operating system.
* DOS is disk operating system runs from disk driver.

VESA video electronic standard association works with ISA to give high speed to accelerate video operations.

ISA: instruction set architecture is a set of instruction that can perform and understand by microprocessor. It says about the program it can execute, data types it can use, registers and memory availability with architecture.

* device driver are the software components that facilates the relationship between operating system and hardware devices.
* Most popular bus for Pentium PC (PCI) Peripheral component interconnect extern.
* XMS: extended memory specification allows to access more than 1M memory. It is ability to switch between extended and convention.

Universal Serial Bus.



Block diagram of computer system.

* UNICODE is 16 bit Alphanumeric code. It allows to handle multi language text in a consistent way.

* The width of address bus can determine size of memory space a processor can access + data bus transfer data between processor, memory
* TPA (Transient Program area) refers to the memory space available in RAM to load and run program.
* Numeric Co-Processor (Math Coprocessor) special hardware component to handle complex numeric calculation. It performs complex arithmetic, increase speed, reduce CPU workload, essential for science
* Address bus transfer memory address, data bus transfer data
* 80387 numeric coprocessor compatible with 80386
* IORC signal ⇒ input output read control.
MRDC ⇒ memory read control ⇒ Processor read data from memory (0)
MWTC ⇒ memory write control ⇒ Processor write data into memory. (0)
IORC ⇒ 0. ⇒ Processor read data from an input device
IOWC ⇒ 0 ⇒ Processor sends data to an output device.
* Octal, hexa to dec ⇒ Point $0.25 = 2\times8^{-1} + 2\times8^{-2}$ // $2\times16^{-1} + 2\times16^{-2}$
decbinary ⇒ 0.625 x 2 = 1.25 ; 0.25 x 2 = 0.5 ', 0.5 x 2 = 1. Top to bottom
101
* Byte is the basic unit of digital information storage, the smallest addressable unit of memory Word is the natural unit of data that can be processed in one operation:, double word is twice of size of word
* ASCII code of enter key is 0DH and . is known as carriage return character. It use to move cursor to the begining of next line.
To store string ⇒ MSG DB 'What time is it?', '$'
DB ⇒ define byte', store data as byte.
' ' ⇒ String enclosed.
'$' ⇒ terminator for DOS interrupt.

* decimal to sign binary ⇒
① convert normal number to binary
② if it is positive it is done
③ if it is negative do the 2's complement.

-120; binary of 120 ⇒ 01111000
2's complement (10000111+1) = 10001000

* Variable declared and stored ⇒ VARI DB -34
* Variable initialize, memory allocate but not stored ⇒ Fred1 DB ?

1234H ⇒ LSB stored at lower address; 2 byte each.

difference between big endian and little endian.

Big endian	Little endian
① Most significant byte is stored at lowest memory address.	① Most significant byte is stored at highest memory address
Least is stored in highest	Least is stored in lowest
Big endian is applied in bit order	Little endian is not prefer in bit order

* decimal to packed and unpacked binary.
Break decimal digit, convert them to binary (4 bit)
① ~~For~~ Packed; each digit full byte, 4 bit convert to 8 bit by adding 4'0 in left side.

102 ⇒ 1 0 2 ; 1=0001;
unpacked ⇒ 00000001 00000000 00000010

Packed, a digit per byte. High nibble first digit, Low nibble second digit.

0=0000; 2=0010
Packed ⇒ 10 ⇒ 20
0001 0000 0010 0000
⇒ 00010000 00100000

Binary to sign decimal; MSB=0 ⇒ Pos MSB=1 ⇒ neg

00110011 ⇒ Positive
2's complement (11001101)
= 11001101
= 205

10001001 ⇒ negative
2's complement (01110110+1)
= 01110111
Dec [-119] -119

BCD to decimal ⇒ 10001001 ⇒ 1001 1001 ⇒ 99
(8) (9)

Charles Babbage ⇒ Father of computer, design difference and analytical engine, concept of Programing
Computer, memory, CPU, I/O

Konrad Zuse ⇒ German engineer and computer pioneer. Built Z series computer, first Programable digital computer. Z3. Earlier high-level programing language.

Joseph Marie Jacquard ⇒ Created Jacquard loom, use Punch cards to control weaving Patterns.

Herman Hollerith ⇒ Founded IBM Company. Electromechanical computer using Punched cards.

Language:

COBOL ⇒ english like and readable.

ALGOL ⇒ block structure, basis of Pascal, C and JAVA.

FORTAN ⇒ first high level Programing language, efficient for numeric computation

PASCAL ⇒ developed by Niklaus Wirth in 1970. design for teaching structured programing and data structuring.

Itanium ⇒ microprocessor developed by Intel corporation.

(1) 64 bit architecture
(2) high performance computing.
(3) EPIC (Explicity Parallel Instruction computing Architecture)
(4) Multiple cores.
(5) cache Architecture
(6) Floating point performance
(7) ~~higher~~ hyper threading technology
(8) Reliability, Availability,

First electronic comp Colossus
Programable digital com
Z3

45 nm Fabrication Technology refers the size of transistor on the chip: 45 nanometer.

① smaller transistor → more transistor per chip → higher performance
② lower power consumption.
③ Enables faster switching → higher clock speed
④ improve chip density and reduce cost per transistor.

* Instruction pointer keeps track of memory address of next instruction to be execute: It points the location in memory where the CPU should fetch next instruction.
* Carry flag bit not modified by logical operation.

difference between carory flag and overflow flag ⇒

① Carry Flag is set when there is a carry out or borrow into high order bit during arithmetic operation.	① Overflow flag is set when signed result of a signed arithmetic operation is too large to be represented in destination operand.
② It indicate that an arithmetic operation between two numbers resulted in a unsigned carry or borrow.	② It indicates an arithmetic operation between numbers results in an unexpected change in sign.
③ It is used for multibyte addition and subtraction.	③ It is used for indicating sign overflow.

* ~~FS → flag avail~~
FS → segment available ⇒ 80386, 80486, Pentium and Pentium Pro.
* Trap flag is set then after each instruction debugger is allowed to examine Processor.
* Stack register is use to access stack segment. It is usually use to store information about the memory segment that stores the call stack of currently executed program. It is used to store stack. SP points to current stack top.

* The descriptor contains information about the segment including it's starting address, length and access right.

** Pentium 4 protected mode allows CPU to access 4GB address space. memory address range 32 bit => 00000000 to FFFFFFFF

features in protected mode =>

① Segmentation => use segment descriptors from Global descriptor table and local descriptor table

② Paging :- Allows virtual memory mapping, divide memory into directory, Page, table

③ Protection :- Each segment / Page has access rights, previnting unauthorize access.

AV can't use By CPU only OS can use it Available bit.

* Program invisible register store the hidden part of the segment descriptor so that CPU can translate logical address to linear / physical address effeiently while enforcing protection.

*G=1 হলে Limit এর 3 কে FFF দিয়ে গুণ !

Value of limit multiplyed by 4K

* 80286 doesn't have a paging mechanism, it use only segmentation for memory management, Paging was introduced later in 80386

* OS in Protected mode =>

0103H = 0000 0001 0000 0011

0,1 Bit = Privilige level = 11 => 3 => low

2 Bit = Descriptor table => 0 => Global descriptor table.

entry => 3-15 => 0,000 0001 00000

=> 20H => 32 decimal

entry is 32th.

OS = 0105 H

= 0000 0001 0000 0101

Privilege level = high = 01 => 1

descriptor table => 1 => local

Entry => 20H => 32th entry.

ference between microprocessor and Microcomputer.

ro computer ও মধ্যে microprocessor central unit হিসাবে কাজ করে

use microcontroller for embedded work or for a specific

sh.

ine cycle or Basic operation of Microprocessor

etch

ecode

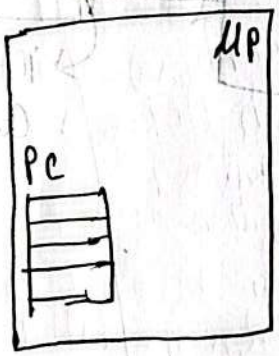
Execute

) MOV B, 04, H

opcode operand.

pc → Program counter ও

memory address রাখে।



Microprocessor category

memory extra space

RISC

Reduced instruction set computer

Instruction I → 20 min

divide করে অনেক instruction ও

সময় per instruction 1 unit

time নেয়।

CISC

complex instruction set computer

here we convert 4/5 instruction into 1 instruction.

It saves memory

special Processor

Co-processor

its computation speed is high.

special type of function perform.

I/O Processor

Transputer

Transistor Computer

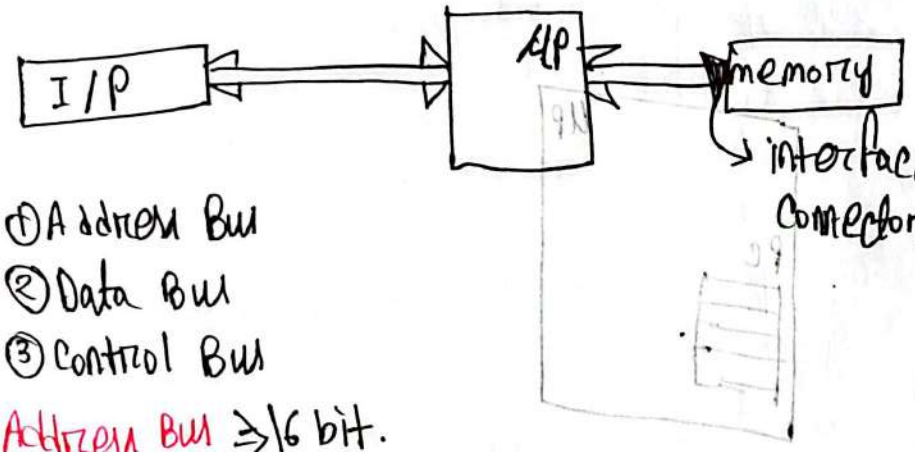
Digital signal Processor

DSP ও Component গুলো

① Program memory
② Program data
③ Computer engine
④ input / output.

System Bus in Microprocessor 8085

Bus ⇒ তার (Communication Path)



① Address Bus
② Data Bus
③ Control Bus

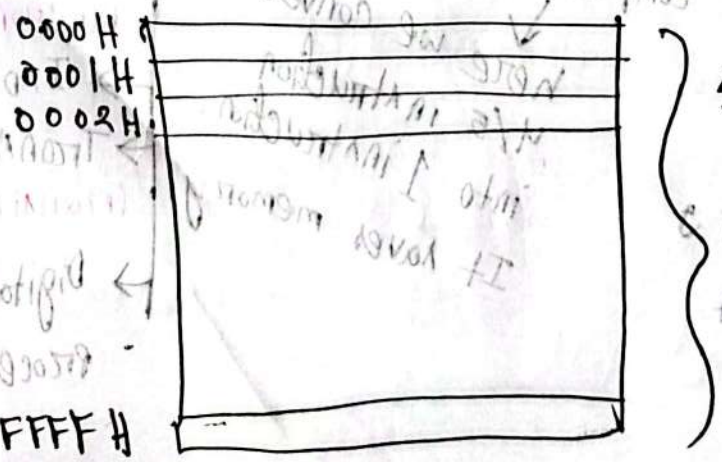
Address Bus ⇒ 16 bit.

memory ও register এর।

register is a collection of flipflop. → store 1bit (0/1)

Address bus 16 bit so we need 16 location

Here address will be in binary but it is for hexadecimal digit. It can address 2^{64} address.



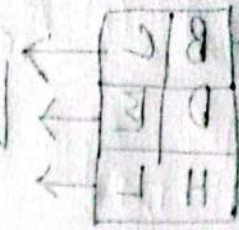
memory address করতে পারে। $2^{10} = 1KB$, $2^{20} = 1MB$, $2^{30} = 1GB$

$2^{16} = 2^6 \times 2^{10}$

$= 64 \times 1KB$

$= 64KB$

16 bit line দিয়ে memory address করতে পারে 64 KB.

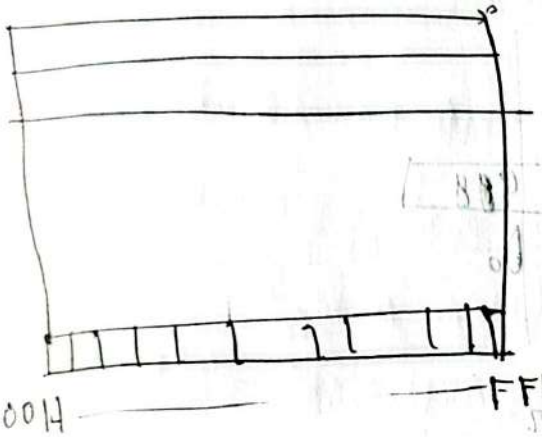


Data bus

At a time একবারে data process করতে পারে। microprocessor

8 bit for 8085

Microprocessor at a time 8 bit data process করতে পারে।



$2^8 = 256$ টি data করতে পারি।

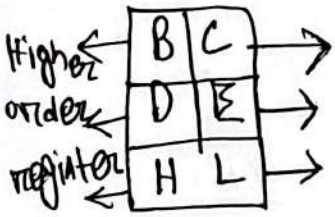
Control bus -

- memory read
- memory write
- I/O read
- I/O write.
- opcode Fetch

Registers in Microprocessor :

1. General purpose register → 6 → B, C, D, E, H and L
2. Specific purpose register
3. Memory register
4. Temporary register.

General purpose register: Every register has 8 bit.

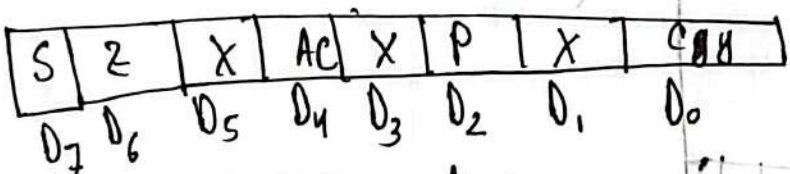


Specific purpose register:

① Accumulator → ALU এর সাথে কাজ করা হয়।

② Flag register → 8 bit → it denotes the situation or status.

Flag register.



S → sign flag → 1 (set) ⇒ Negative
0 (reset) ⇒ positive number

Z → zero flag ⇒ 1 (set) ⇒ All 8 bit in zero
0 (reset) ⇒ Not zero.

AC → Auxilary carry → 1 (set) ⇒ carry generate.
0 (reset) ⇒ no " "

P → Priority flag → 1 (set) ⇒ even number of 1's
0 (reset) ⇒ odd " of 1's.

Carry → 1 (set) → carry generate.

Auxiliary carry

1 0 0 1 1 0 0 0
1 1 0 0 1 1 0 1
1 0 1 0 0 1 0 1

Carry

most significant bit

Flag register X ⇒ don't care 0 বা 1.

0	0	0	1	0	1	0	1

Memory Register: (16 bit)

Program Counter
↓
কোন address থেকে information fetch করা হবে।

Stack Pointer
↳ stack memory address point করে।

Temporary Register:

W	Z

কুইক গণনের জন্য এটি ব্যবহৃত হয়।

Chapter-2

8085 and 8086 Microprocessor Architecture

Control signal:

ALE (Address Latch Enable) indicate when the lower order address is valid on the multiplexed lines.

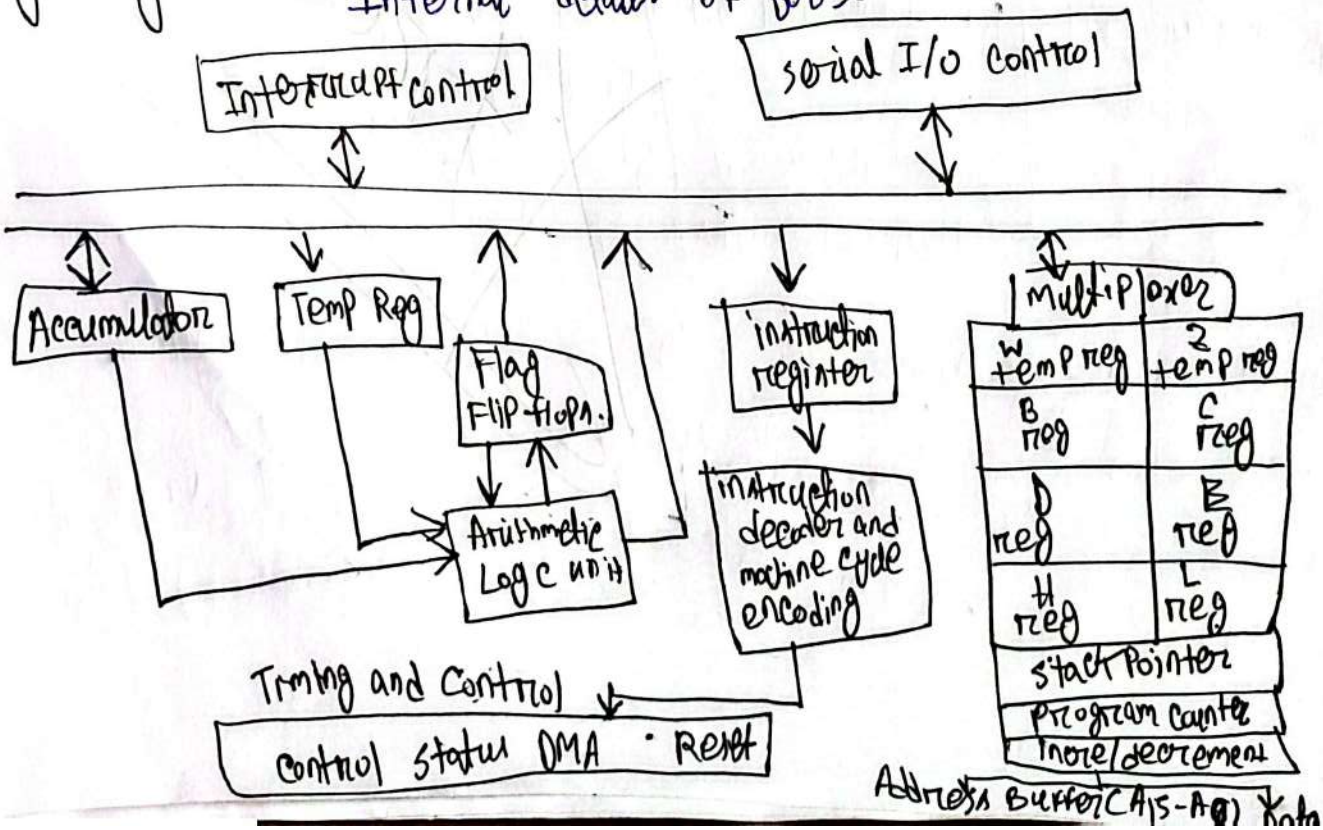
READY use to interface slower memory; CPU waits until READY is high before proceeding.

HOLD signals the CPU to release control of buses to external device like DMA (Direct memory access)

HLDA (Hold Acknowledge) : CPU response to HOLD indicating bus release

##WAIT states are inserted between clock cycles when slower memory needs more time. During WAIT states, the clock continues, but the CPU does not change its state until READY goes high.

Internal details of 8085.



The 8085 support two types of I/O access:
Memory mapped I/O :- I/O device treat like memory location
I/O mapped I/O :- I/O device have separate address space.
IO/M (Input output/memory) signals helps distinguish ⇒
IO/M = High(1) indicates I/O mode access.
IO/M = Low (0) indicates memory mode access.

- RESET IN : An external signal to reset 8085 microprocessor.
RESET OUT : Generated by 8085 in response to RESET IN, use to reset other connected support device.

Accumulator :- An 8 bit register use for arithmetic, logic, I/O and load/store operations.

Flag register :- 8 bit register containing 5 one bit flags.
sign ⇒ set if the most significant bit of result is 1 (means negative number).
zero ⇒ set if the result of an operation is zero.
Auxiliary carry ⇒ set if the carry out from bit 3 to bit 4 in the result in BCD (Binary coded decimal) arithmetic.
Parity (P) ⇒ set if the number of 1s (set bit) in the result is even
carry (CY) ⇒ set if there is carry out during addition or a borrow during subtraction.

Instruction register :- Holds the current instruction being executed

Arithmetic Logic Unit (ALU) :- perform arithmetic and logical operation

Temporary register hold intermediate data during operation.

Control unit decodes instruction and generates control signals.

Address bus (A15-A0) 16 bit unidirectional lines to address memory or I/O.

Multiplexed Address / Data bus (AD_0-AD_7) Lower 8 bits of address and data share the same lines.

General Purpose Register

BC => 8 bit B, 8 bit C when together C contain low order byte

DE => 8 " D, 8 " E " " " E " " " "

HL => 8 " H, 8 " L " " " L " " " "

Some instruction use BC / DE as data pointer.

But HL contain data pointer to refer memory address.

* Stack pointer 16 bit register increment or decrement by 2 bit use to point to the current top of stack
* PC (Program counter) holds the address of next instruction to be executed (16 bit register).

Addressing mode in 8085 =>

* Direct, * register indirect (operand is in memory location point by HL)
* register (operand in register)
* implied (operand implied by instruction)
* immediate: operand part of instruction itself.
* instruction can 1, 2, 3 byte long depend on addressing mode, the first byte is opcode which tells CPU what operations to perform.

Types of instructions :-

Data moving

Arithmetic => Add, subtract, increment, decrement

Logical => AND, OR, XOR

Control => Jumps, calls at conditional or, @ unconditional.

Input/output => read/write to I/o ports

others => set/clear flags, enable/disable interrupts.

MACHINE CYCLES =>

CLK => Processor clock signal that controls timing.

S_1, S_0 => status line, tells the type of machine cycle.

A_8, A_{15} => High order address lines (always carry address during cycle)

AD_0, AD_7 => Multiplexed line for low order address (T_1) and data (T_2-T_n)

ALE => Address Latch enable (high in T_1 to store AD_0-AD_7 into external Latch)

RD => Read control signal (active low)

WR => write control signal (active low)

READY => indicate if memory is ready for data transfer (low = insert wait state)

opcode fetch machine cycle.

T-state	CLK	S_1 S_0	A_8-A_{15}	AD_0-AD_7	ALE	RD	WR	READY
	↑↓	1 1	High Add	Low Add	1	1	1	1
T_1		1 1	High Add	Data from mem	0	0	1	1
T_2	↑↓	1 1	High Add	opcode table	0	1	1	1
T_3	↑↓	1 1	—	—	—	—	—	—
T_4	↑↓							

T_1 = Address placed on bus, ALE high to latch AD_0-AD_7

T_2 = RD low, memory puts opcode byte on AD_0-AD_7

T_3 = CPU reads opcode into instruction register.

T_4 ⇒ opcode decoding

Ready=0 in WAIT state (T_W) is inserted between T_2 and T_3 until READY = 1

* Read data from memory
* Write data from CPU.

Machine Read Machine cycle

T-state	clk	S_1 S_0	A_8-A_{15}	AD_0-AD_7	ALE	RD	WR	READY	Action
T_1	↑↓	1 1	High Add	Low Add	1	1	1	1	Address on bus, ALE high to latch
T_2	↑ ↓	1 1	"	Data from mem	0	0	1	1	RD Low, memory sends data
T_3	↑↓	1 1	"	Data stable	0	1	1	1	CPU reads data
T_4	↑↓	1 1	"	—	—	—	—	0	If ready=0, wait state inserted

Memory Write Machine cycle.

F-state	clk	S_1 S_0	A_8-A_{15}	A_0-AD_7	ALE	RD	WR	READY	Action.
T_1	↑↓	0 1	High Add	Low Add	1	1	1	1	Address Placed, ALE high to latch Low
T_2	↑↓	0 1	High Add	Data from CPU	0	1	0	1	WR low, CPU outputs data.
T_3	↑↓	0 1	High add	Data stable	0	1	1	1	memory stores data

* CPU read from memory, so memory sends data to CPU.

CPU writes to memory means CPU sends data to memory

In opcode fetch, memory read, S_1=1, S_0=1, cause in both CPU is getting data from memory

opcode fetch ⇒ the memory location contains an instruction

Memory read ⇒ The memory " " data.

Interrupts in 8085

*maskable interrupt are those which can be ignored by the processor.

TRAP

Non-maskable and cannot be disabled by instructions. Needs a high level with a leading edge.
Process: CPU finish current instruction, pushes program counter to stack. Jumps to 0024H (TRAP service Routine).
Disabled when the signal on TRAP pin falls (goes LOW). It has highest priority among all.

RST 7.5 (Edge triggered) => doorbell that rings once when button is pressed

Maskable as it can be turned ON/OFF using SIM instruction.
Trigger: Leading edge sensitive; responds when signal change from low to high.
Process: CPU finishes current instruction. Saves PC (program counter) on stack. Jumps to 003CH.
8085 remembers this interrupt using an internal D flipflop so even if signal goes low it's still remember until serviced.

RST 6.5 (Level sensitive) doorbell that keeps ringing if you keep pressing it

Maskable enable/disable with sim instruction.
High level sensitive must stay high during recognition. Finish current instruction save PC on stack Jump to 0034H.

RST 5.5 (Level sensitive) same as 6.5

Maskable also controlled via SIM. HIGH level sensitive
Finish current instruction, save PC, jump to 002CH.

INTR (lowest priority, level triggered)

Interrupt request. It is maskable interrupt can be turned ON or OFF by the CPU. It works when the INTR Pin is HIGH.

When no other higher-priority interrupts are active and the INTR signal is High, 8085 finish current instruction and send an interrupt Acknowledge (INTA) signal to device.

After INTA, external hardware must send an instruction to the CPU. This is usually,

A 1 byte call instruction → RST0 to RST7 or a 3 byte CALL with an address. This instruction tells the CPU where to jump to run the interrupt service routine.

INTR does not have a fixed address in CPU. The device or interrupt controller must provide opcode and vector address

When INTR is given the CPU temporarily disable interrupt to avoid repeated trigger from the same device

8259 programmable interrupt controller can connect 8 external interrupts (IR0-IR7) to the CPU and manage.

1. Whether the interrupt is level or edge triggered.
2. Which interrupts are masked
3. The priority order.
4. The CALL address for service routine.

8259 Programmable interrupt controller (PIC) with 8085

The 8259 manages multiple external interrupts and sends them to the CPU in priority order. One 8259 handle 8 interrupts. Multiple 8259 chips can be connected to handle upto 64 interrupts.

8259 checks if the interrupt is enable (unmasked) and has higher priority than the current task, if yes it sends INTR signal to 8085. When 8085 accepts it sends three INTA pulses (one by one)

- 8259 sends the CALL opcode
- 8259 sends low byte of ISR (interrupt service routine) address
- 8259 sends high byte of ISR.

8085 saves the current program counter (PC) on stack and jumps to the given address to execute ISR.

Working in priority mode and can mask interrupts. With cascading test it handle many more interrupt sources.

8086 microprocessor main feature: (Lec-1)

- 16 bit processor (ALU and registers are 16 bit)
- Data bus 16 bit means can read/write 16 bits or 8 bits at a time.
- Address bus: 20 bit → can access 1MB memory.
- Clock speed 6-10 MHz.
- 8086 does not get permit to do floating point operations. But 8086 work in conjunction with 8087 to do both fixed-point and floating point with complex mathematical functions.

8086 operate in two modes.
minimum mode ⇒ single processor system that generate control bus signal.
maximum mode ⇒ Multiprocessor or with co processor 8087 generates signals.

Can prefetch 6 bytes for faster execution.
It uses 40 pin dual in line package
It requires +5V.
It has two blocks. Bus interface unit ⇒ fetch, Prefetch, execution unit ⇒ execute

AD_0-AD_{15} Multiplexed address/data bus.

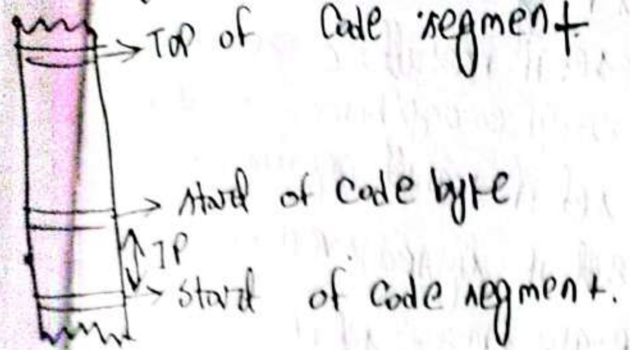
ALE ⇒ 16 bit address bus to be latched, which stores address, lower order 16 bits of the address bus can be used as data bus.

READY ⇒ memory or I/O is ready for data transfer.

INTR (Interrupt request)
* Non maskable interrupt → edge trigger
* Maskable " → level trigger

Maskable interrupt input. It is level triggered mean the signal must be held HIGH when sampled. The CPU checks this signal during the last clock cycle of each instruction to decide if it should start the interrupt process. If INTR is high and interrupts are enable CPU will give interrupt acknowledge (INTA) sequence.
It is maskable by software can enable or disable INTR the CPU internally synchronizes this signal to avoid glitches. INTR is considered active when the signal level is HIGH.

Memory.

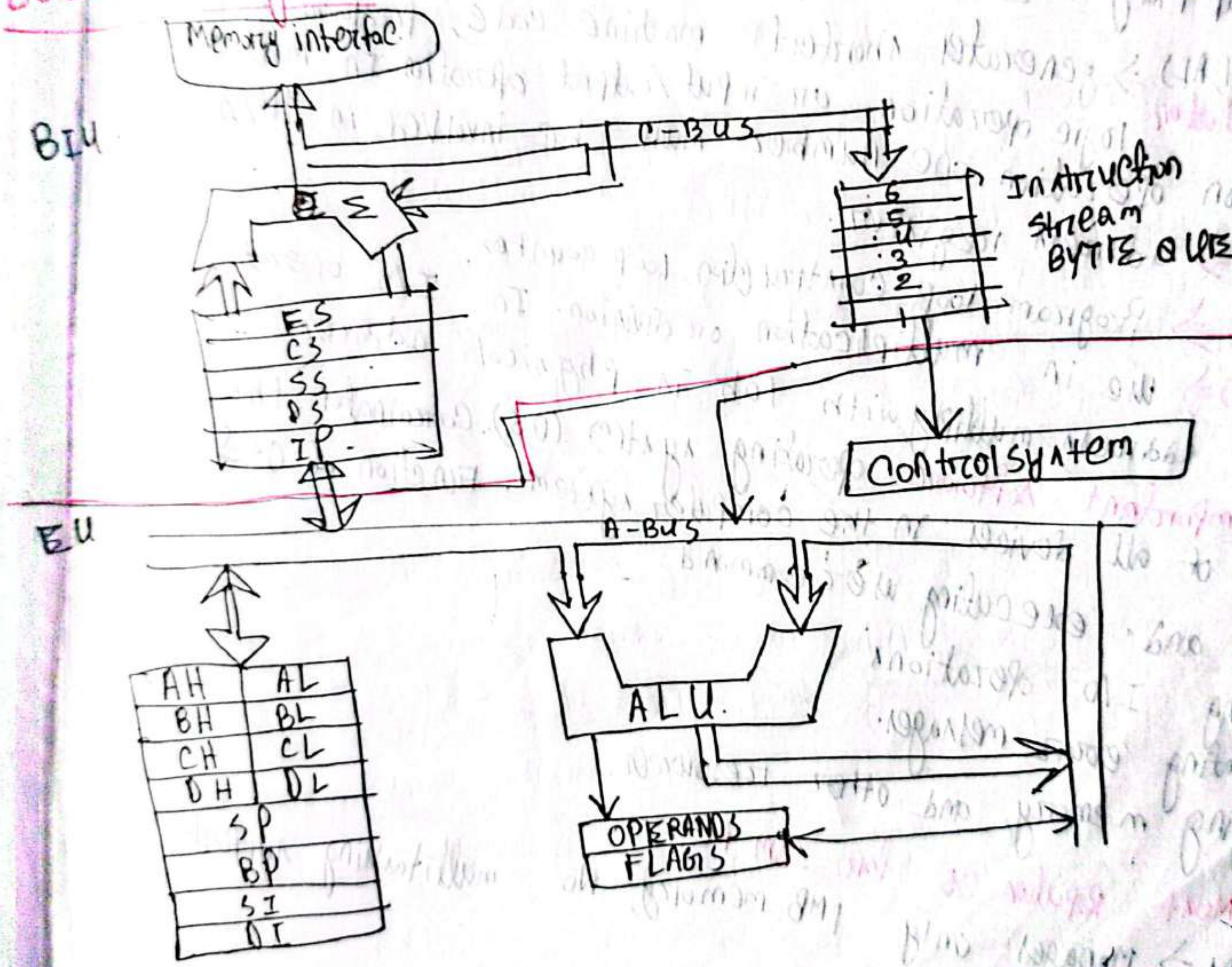


65535.

Hexa 2000H
Decimal এ দিবো 65535
সেটা থেকে Hexa তে আনলে
Top segment পাওয়া যায়।

BP → For stack data with stack segment. SI = source index for reading data with DS. DI = destination index for writing data with ES or DS. These 3 registers can be used for temporary storage of data just as general purpose register.

8086 Block diagram :



8086 has 14 registers
4 data registers (general purpose + special use)
AX, BX, CX, DX
4 segment registers (memory management)
CS, DS, SS, ES.
5 Pointer/index register (stack + address)
SP, BP, SI, DI, IP
1 Flag register (status/control)
zero, carry, sign, overflow.

Status Flag
ZF => set if result = 0 (zero flag)
CF => set if carry/borrow occurs
SF => set if result negative (sign)
OF => set if signed overflow occurs
PF => Parity flag => set if result has even number of 1

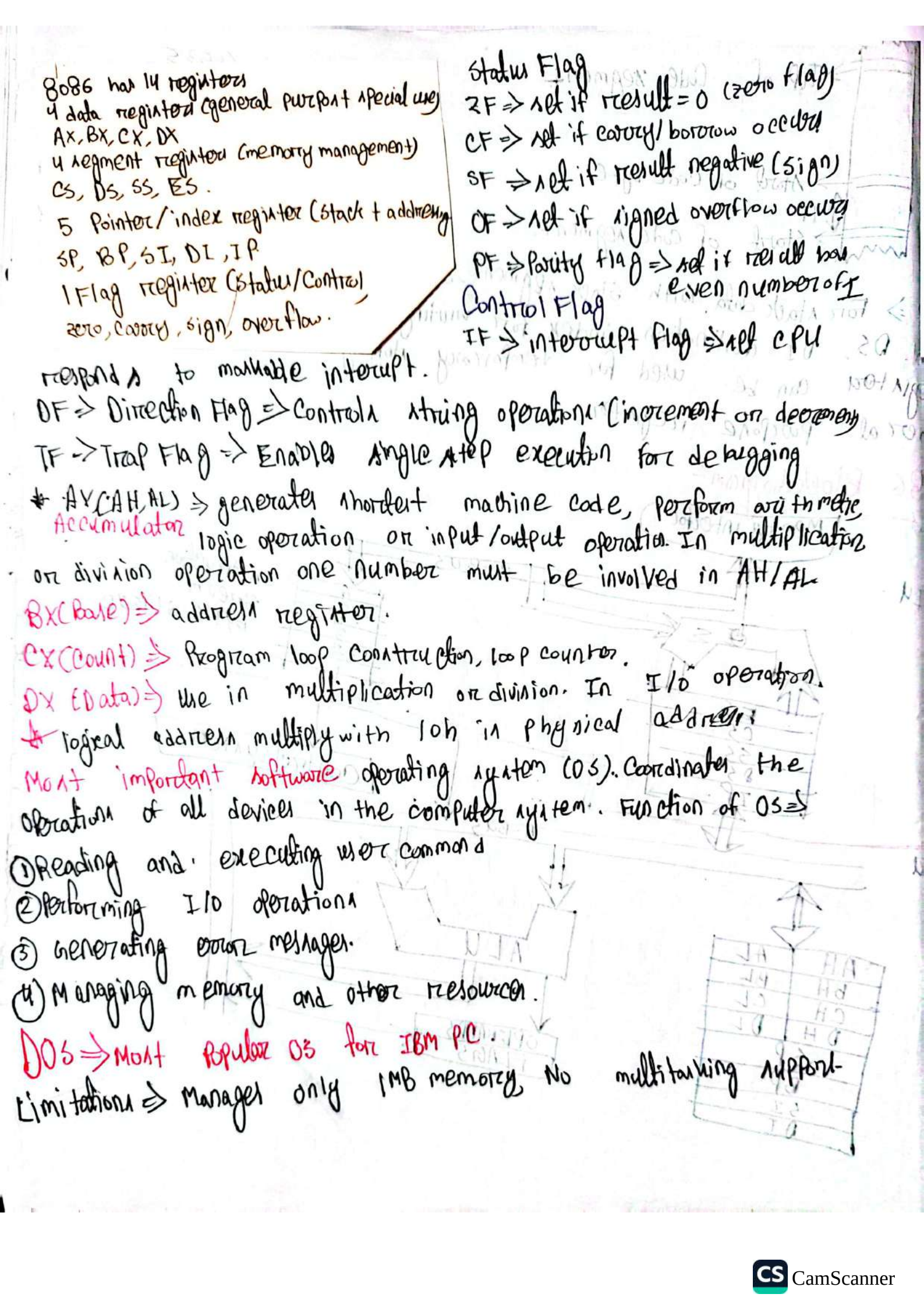
Control Flag
IF => interrupt flag => set CPU respond to maskable interrupt.
DF => Direction Flag => Controls string operations (increment or decrement)
TF => Trap Flag => Enables single step execution for debugging

* AX (AH, AL) => Accumulator => generates shortest machine code, perform arithmetic logic operation or input/output operation. In multiplication or division operation one number must be involved in AH/AL.

BX (Base) => address register.
CX (Count) => Program loop construction, loop counter.
DX (Data) => use in multiplication or division. In I/O operation.
* logical address multiply with 10h in physical address.

Most important software operating system (OS). Coordinates the operations of all devices in the computer system. Function of OS =>
1. Reading and executing user command
2. Performing I/O operations
3. Generating error messages.
4. Managing memory and other resources.

DOS => Most popular OS for IBM PC.
Limitations => Manages only 1MB memory, No multitasking support.



INTA (Interrupt Acknowledge)

It is signal sent from cpu to the interrupting device to acknowledge it has received the interrupt request and is ready to handle the interrupt. After INTA CPU expects the device to provide the interrupt vector or service routine address.

NMI (Non-Maskable Interrupt)

Non maskable, edge triggered interrupt input. Activate on a rising edge (signal goes from LOW to HIGH) causes the CPU to interrupt whatever it is doing and jump to a predefined interrupt vector in memory. It cannot be disable by software always get attention of CPU. It is used for critical event like hardware failures.

RESET

The signal causes the CPU to immediately stop its current operation and restart execution from the begining 0000. must be signal HIGH for at least 4 clock cycles to ensure proper reset. use to initialize the CPU, or recover from errors.

8086 Family

8086 has 20 bit address bus so it can access upto 2^{20} memory locations. Each memory location hold 1 byte and two consecutive bytes form a 16 bit word

If the 16 bit word starts at even address, 8086 reads both byte in one operation (one byte at a time). 8086 will it in two operations

8088 is same internal architecture as 8086 but 8 bit data bus. So it reads/write 8 bits at a time, need two operations to read/write a 16 bit word. Use in original IBM PC.

80186 and 80188 are improved version of 8086 and 8088. Include some integrated programmable peripheral devices. Their instruction set is SUPERSET of 8086/8088 (all old instructions plus new ones). Programs for 8086 and 8088 will work on 80186 and 80188 but it may not happen for other way round.

80286 a16 bit procession design for multitasking and multiuser system. It keeps the users program separate and protects the system from crashes caused by faulty programs. Use in IBM PC/AT.

80386 is 32 bit registers and data paths. Can address 4GB memory; still runs old 16 bit programs and more multitasking features.

80486 integrated floating point unit, faster, more efficient

Pentium series Added multimedia extension, Features like Power saving, hyper threading.

8086 CPU structure

Bus Interface Unit

handles communication with memory and I/O. sends address, fetch instruction, read/write data. keeps instruction queue, prefetch instruction before EU needs them.

Execution Unit (EU)

Executes instruction fetched (instruction to be processed) by the BIU. Contains ALU, Registers, Flag register, decoder and runs instruction.

Why split into BIU and EU?

While the EU is executing one instruction, the BIU can fetch the next one this is a form of pipelining, which speeds up performance.

EU Execution Unit → control instruction to ALU

tells the BIU where to fetch data/instruction from decodes the instruction Execute them using ALU.

control Circuitry → Directs operation inside the CPU.

instruction decoder → Converts fetch instruction into specific actions for the ALU and registers.

ALU → Performs Addition, subtraction, Logical AND, OR, NOT, increment, decrement, shift and rotate operation.

Flag register: A flag is a flip-flop that indicates some condition produced by the execution of an instruction or control certain operation of EU. EU contains nine active flag. Six of them are conditional set/reset automatically by ALU based on result

(Carry Flag) → 1 if there is carry out of MSB in addition or borrow in

(Parity flag) → 1 if result has even number of 1 bits.

(Auxiliary carry) → used for BCD (binary coded decimal) operations

(Zero flag) → 1 if result = 0

(Sign flag) → copy of the MSB of result (1 = negative)

(overflow flag) → 1 if signed number result is too big to fit.

The remaining three flags are control flags set/reset by programmer)

TF (Trap Flag) => Enables single-step execution for debug.

IF (interrupt) -> Enable/disable maskable interrupt

DF (Direction) => Controls direction for string operation

8086 Flag register format

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	U	U	U	U	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	1	CF

General purpose register

There are eight (8-bit) registers in the Execution unit, AH, AL, BH, BL, CH, CL, DH, DL; this can be individually store 8 bit value. AL is called the accumulator it has extra functionality like multiplication, I/O operation.

AX = AH (high byte) + AL (low byte)

BX = BH + BL, CX = CH + CL, DX = DH + DL.

These 16 bit registers can store a full words (2 byte)

Data in registers can be accessed much faster than data in external memory because it's already inside execution unit. Using register avoids delay of bus operation.

IU - Bus interface unit

ching instruction from memory. Managing the memory segments.
nterfacing with system bus.

ueue (Instruction Prefetch)

The BIU can fetch upto 6 bytes of instruction ahead of time and
tore them (First in-First out) FIFO queue.
hile execution unit (EU) is decoding or executing an instruction
hat doesnot need the bus, the BIU is busy loading next instructio
his speeds up execution.
Without queue, the cpu would have to wait for memory fetches
between every instruction
With queue, the next instruction is already in BIU memory. This
echnique is called pipelining.
he CPU is like bricklayer. The BIU is like the helper who lines
p bricks so the bricklayer can work without pausing
or Jump and CALL instruction, the queue is flushed and refilled
arting from the new target address.

egment register

The 8086 has 20 bit address but and can address 1 MB of memor
At any one time, it works with four 64KB segments.
Segment register is used to hold the upper 16 bits of the
starting address for each of the segments.

CS (Code segment) ⇒ Holds the starting address of the Program
instruction. The CPU use this to know where to
fetch and run instruction from

DS (Data segment) ⇒ stores the starting address of program's data. The CPU use this when reading or writing data.

SS (stack segment) ⇒ Holds starting address for stack area. It is used for storing temporary data, function return address, for passing data during program execution.

ES (Extra segment) ⇒ stores the starting address of an extra memory. often use in special operations like moving or comparing blocks of data.

OFFSet is the distance from the start of the segment to the actual data or instruction you want.

Each segment base address is stored in a segment register as a 16 bit value. But as 8086 can access 1MB of memory. So it needs 20 bit physical address. To get 20 bit physical address the CPU:

① Takes the segment register value
② shifts it left by 4 bits.
③ Adds the offset.

CS ⇒ holds the upper 16 bits of the starting address of code segment.

IP ⇒ holds the offset of the next instruction.

CS ⇒ 348AH
IP = 4214

CS BASE = 348A0
IP = 4214
Physical ⇒ 38AB4H.

Alternate notation can be offset from CS:IP = 348A:4214

*if there are two offsets then the addition will be only 4 bit if there are 5 bit then the MSB will be dropped. And this 4 bit will be added with 5 bit segment register.
segment address : offset address.
*code segment + instruction Pointer
stack segment + SP / BP.
Data segment + BX, DI, SI, 8 or 16 bit number
Extra segment + DI for string instruction (string destination address)
*FS, GS are extra 2 segment which is allowed by 8086 to Corez
* segment can be moved anywhere by changing segment starting address.
linear address is the address generated by the Program.
physical address is the actual address accessed by a program.
through memory paging linear address is invisibly translated to physical address.

-GDTR ⇒ global descriptor table register
IDTR ⇒ interrupt descriptor table register.
Both must be set before using protected mode.
-LDT is chosen from an entry inside the GDT.
through paging program can run even if memory is not continous.
memory can be borrowed from disk and allows creating extra memory area like upper memory blocks in DOS.
some memory areas between video BIOS and system BIOS were
mm.EXE uses paging to realign extended memory and making those area usable.
Control Register (CR0 - CR4)
R0
this a PG (Paging) bit (leftmost bit)

if PG=0, no Pagging ; linear address = physical address.
PG=1 → Pagging enable, linear address convert to physical address using Paging system.

CR3 (Page directory base register):

holds base address of Page directory (where Paging starts), must be alligned on 4KB boundary. contain PCD and PWT.
PCD → Page Cachedisable → 1 ⇒ disable caching for certain Page.
PWT → Page Write Through → 1 ⇒ Controls write through caching (how writes are handled in cache)

CR2:

Stores linear address where a Page fault happened. Helps CPU know which memory location caused error.
CPU → Controls advanced Paging feature

- 32 bit linear address divide into 3 Parts ⇒

① Page directory index, 10 bits (22-31)
selects one entry in Page directory. Each entry covers 4MB chunks of memory.
② Page Table index, 10 bits (12-21)
selects one entry in Page table chosen by the directory. Each entry covers 4KB Page.
③ Page offset (12 bits, (0-11))
Points to the exact byte inside the 4KB Page.

Translation look Aside Buffer (TLB)

Paging requires multiple memory lookups (directory + tables).
CPU use TLB cache to store the last 32 Page translations.

If same page is accessed again → CPU can get the physical address directly from TLB (super fast).
If not in TLB → CPU must read the directory and table again.
Modern CPU have separate TLB for instruction and data.
* CPU use 1 Page directory ⇒
1 Page directory contain 1024 entries (each entry points one page table)
* 1 table has 1024 entries (each entry 4 KB page of memory)
* So Page directory total (1024 x4) = 4 KB
table = (1024 x 4 KB) = 4 MB.
So to map entire 4 GB memory I need = 4 MB + 4 KB = 4.004 MB
memory for Paging structure.

Flat Mode

In flat mode, no segmentation is used, memory looks like one big continuous block (0000 0000 0000 – FFFF FFFF FFFF H) (48 bit address lines).
Code segment register is still used but in flat mode it doesnot provide packet limit. It only define (Previliege level and access right).
In 64 bit · offset = Physical address
32-bit protected mode descriptor modify the actual address.
*Address is 64 bits but only 48 bits are output to pins. Means maximum 1TB of physical memory accessible. Any higher bits are cut off

Extra Info=>2

* GDT stores segment descriptor. Each descriptor 8 byte (64 bits)
- Max GDT size => 16 bit limit => 64KB.
  Max descriptor 8192.
  descriptor size = 8 byte.

* LDT is addressed through segment selector with TI = 1
CPU use LDTR to find base and limit of LDT. Local descriptor table is a segment descriptor just like global descriptor table unlike global descriptor table performs for whole system, LDT is/are worn for each task or process.
LDT mainly used to give process specific memory segments, specially protected mode of X86 processor.
LDTR contain =>
① segment selector for LDT in GDT
② Base address of LDT (from descriptor)
③ Limit of LDT

Difference between Register and memory location =>

register is in CPU. It operation instruction. Register known by name. Register are 16, 16 bit.	Memory location in RAM. It stores data Memory location name by address. memory location is 1MB - 8 bit

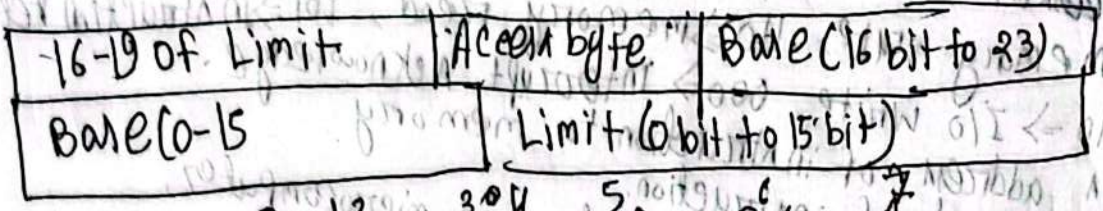
Reason for manual and Power-on reset
Manual Reset = Allows for intentional system restart or recovery
Power on reset = Ensures a known and stable state when power is applied.

ctor consideration to estimate the interrupt response time 8085.
Interrupt acknowledge time ② vector fetch time
① Context saving time ④ Jump to ISR time.
S, So helps to select different set of signals during read/write cycle.
So S1 S2 use by external hardware (memory or I/O device) to understand whether the processor wants to read, write, fetch instruction or acknowledge interrupt. 111 ⇒ memory write, 110 ⇒ memory read, 101 ⇒ instruction fetch 01 ⇒ I/O Read, 010 ⇒ I/O write, 000 ⇒ Interrupt Acknowledge.
dress bus ⇒ sends address of instruction to memory.
ntrol ll ⇒ include signal of instruction.
ata ll ⇒ Transfer the fetched instruction to microcomputer.

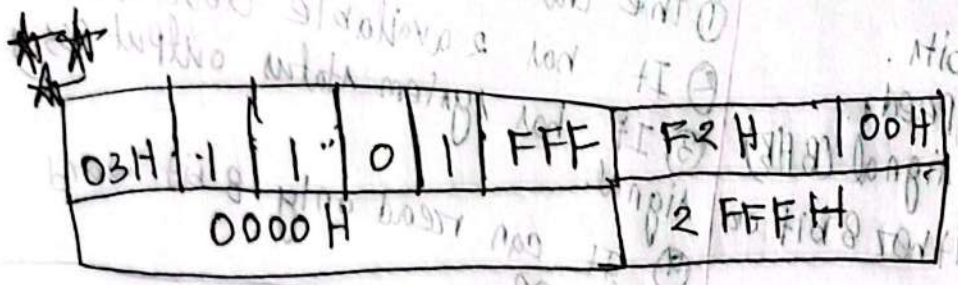
8086	8088
he data bus is of 16 bits.	① The data bus is of 8 bits.
It has 3 available clock speeds.	② It has 2 available clock speeds.
It has Bus high Enable signal (BHE)	③ It has system status output (SSO) signal.
) It can read or write either 8 bit or 16 bit at once	④ It can read only 8 bit word at once

86 queue ⇒ 6 byte instruction prefetched
antage usage of using CPU register for temporary data storage
ead of memory location.
ed: Registers are faster than memory for storing and retriving data.
ximity: Registers are located on the CPU chip, reducing physical distance.
ficiency: CPU can directly work with data in registers, avoiding the delays associated with accessing data from memory.
ecution execution. for ADD AX, BX
tch, Decode, Read, Addition, Write.
sembly language use for ⇒
mbedded systems Programming ④ Bootloader
① Device drivers. ⑤ Low-level system Programing.
② Operating system kernels

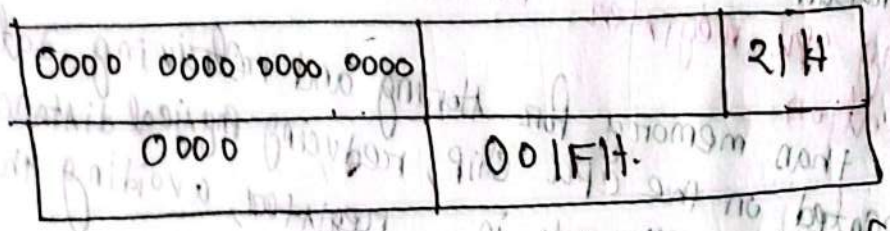
CS ⇒ stores all program instruction to be executed.
DS ⇒ stores data variable, constant, array by program.
SS ⇒ store the stack temporary data like return address, local variable.
ES ⇒ stores additional data, mainly use in string operation.



Access right ⇒ P PL DC R/W A



Access right ⇒ 1111 0010 ⇒ F2 H.



- 32 bit base address and 16 bit limit of GDT.
- description is a 8 byte entry stored in descriptor table. defna properties of memory segment of a page.

Ch 3
Intel and Microprocessors

Can run on 80286, 80386, 80486 in real address mode. Main function: Reading/writing information on disk. file naming rule → (1-8) characters for name, optional extension: (1-3) characters. COMMAND.COM

DOS routine for servicing user commands ⇒
- two types of commands:
- Internal commands: Already loaded in memory.
- External commands: Stored on disk, loaded only when needed → save memory.

BIOS (Basic Input/output system)
Stored in ROM, not erased when power is off. Performs I/O operations. DOS I/O operations are executed using BIOS routine. BIOS can also perform: Circuit checking (POST), Loading DOS' routines.

Interrupt vectors: Addresses of BIOS/DOS routines stored in memory starting at 00000h.

Memory organization of the PC.
8086/8088 can address 1MB memory, but not all for applications. Memory is divided into segments each 64 KB. segments ⇒ 0000h, 1000h, 2000h, ..., FF00h. total 16. First 10 segments (00000h-09000h) → use by DOS for programs.

640 KB memory.
- Application programs Area: 00000h-09000h (640 KB)
- Video memory: segments A0000h and B0000h
- Reserved segments: C0000h - E0000h
- BIOS: ROM basic and BIOS routine.

When power is on → 8086/8088 enters reset state. Then CPU registers has code segment = FFFFh and Instruction Pointer 0000h. First instruction executed at FFFF0h.

BIOS Execution:

Instruction at FFFF0h transfers control to the BIOS routine (stored in ROM). BIOS steps:

① Perform system and memory test (Power-on self-test, POST)
② Initialize interrupt vectors.
③ Set up BIOS data area.
④ Load the operating system (OS) from the system disk.

Boot Process: Loading OS happens in two steps:

① BIOS loads a small program name Boot Program.
② Boot program loads the main OS routine.

Boot program keeps BIOS independent of OS changes. Makes BIOS smaller and simpler. Bootstrapping is computer pulls itself up step by step.

⑤ After OS is loaded → COMMAND.COM is given control. COMMAND.COM provides the DOS prompt and manages user commands.

---

* Paragraph: 16 bytes of memory.
Paragraph boundary: Address divisible by 16.
Segment starts at paragraph boundary; 64 KB MAX.
Offset: distance from start of segment.

Local selector Table is addressed by using a segment selector to ex into the Global descriptor table.
obal description table defines segment descriptor. GDTR is a 48 bit egister that holds base address and it's limit. In protected mode, segment egister hold selector.

- Immediate addressing mode MOV CX, 437BH
  Register addressing mode MOV CL, AL
  Memory addressing mode segment: OFFSET
  Direct addressing mode MOV. BX, [437AH]
  ⇒ From data segment 437A will be copy to BX.

* segments value are set in Program initialization.

Microprocessor

Contains only CPU, RAM, ROM, I/O ; timer are separately interfaced
① Designer decide the amount of RAM, ROM, I/O Ports.
② High cost
③ General purpose
④ Higher speed
⑤ High ~~cost~~ power consumption. 8085

Microcontroller

① CPU, RAM, ROM, I/O and timers are all on a single chip
② Fix amount on chip ROM, RAM, I/O Ports.
③ Low cost
④ Single purpose
⑤ Low speed
⑥ Low power consumption. 8051

Execute → Fetch instruction
↑ Fetch operand ← decode instruction

- 80286 16 bit processor with 24 bit physical address, supporting rotected mode, multitasking, improving bus interface.

Physical memory

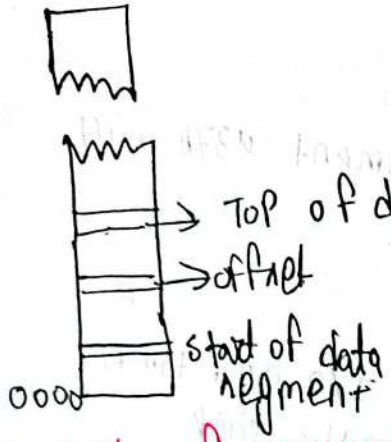
① Refers to the actual hardware components where data and instruction are stored in a comput
② physical hardware components storing data.

logical memory

① Refers to the virtual memory space that a program or process uses during execution.
② Virtual representation of available memory.

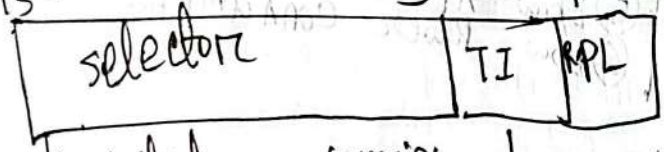
③ direct interaction by CPU.
④ user and applications don't directly interact.
⑤ managed by hardware and operating system

⑤ Accessed through logical address.
④ program and user work with logical address.
⑤ manage by operating system.



* pentium 4 and core 2 have 1TB memory

Content of segment register at protected mode :-
segment register contains a selector at protected mode memory addressing. selector that selects a descriptor from a descriptor table.



RPL ⇒ request previlige level
00 the highest
11 the lowest

↳ select one descriptor among 8192 descriptor in either the global or the local descriptor table.
↳ TI = 0, Global descriptor table
TI = 1, Local descriptor table.

segment ⇒ 2300H
starting address = 2300 × 10H = 23000
Ending address = 23000 + FFFF = 32FFFH

* Protected mode allow to access 1M byte memory area and above the memory area.

Ch-2 (Intel) Math

80286 code description

Limit (16-19 bit) 0000 bH	Access right	Base (16-23 bit
Base (0-15 bit)	Limit (0-15 bit)	

80386 to Pentium 4 code descriptor L=0; An it is not 64 bit Processor.

Base (24-31 bit)	G	D	0	L	AV	Limit (16-19 bit)	Access right	Base (16-23 bit
Base (0-15 bit							Limit (0-15 bit)	

4 bit-P4. L=1; An 64 bit Processor

0000	G	D	L	AV	0000	Access right	Base (16-23 bit
Base (0-15) bit						Limit (0-15) bit	

Starting location = 210000H Limit = (21001F − 210000) = 001F (0-15 bit)
Ending location = 21001FH

Base (0-15) = 0000
Base (16-23) = 21H.

Limit (16-19) = 00H
Access right = P DPL S E ED R/W A
1 LL 1 0 0 1 0 = F2

0000 0000 0000 0000	FFH	21H
0000	01FH	

Stard = 03000000
END = 05FFFFFF

Limit = 02FFFFFF An limit is too big
G=1, limit = 2FFFH

Access right = 1 11 0 0 10 = F2H.
G D L AV

Access right →

03H	DH	0000	0000 F2H	00H
0000 H			2FFFH	

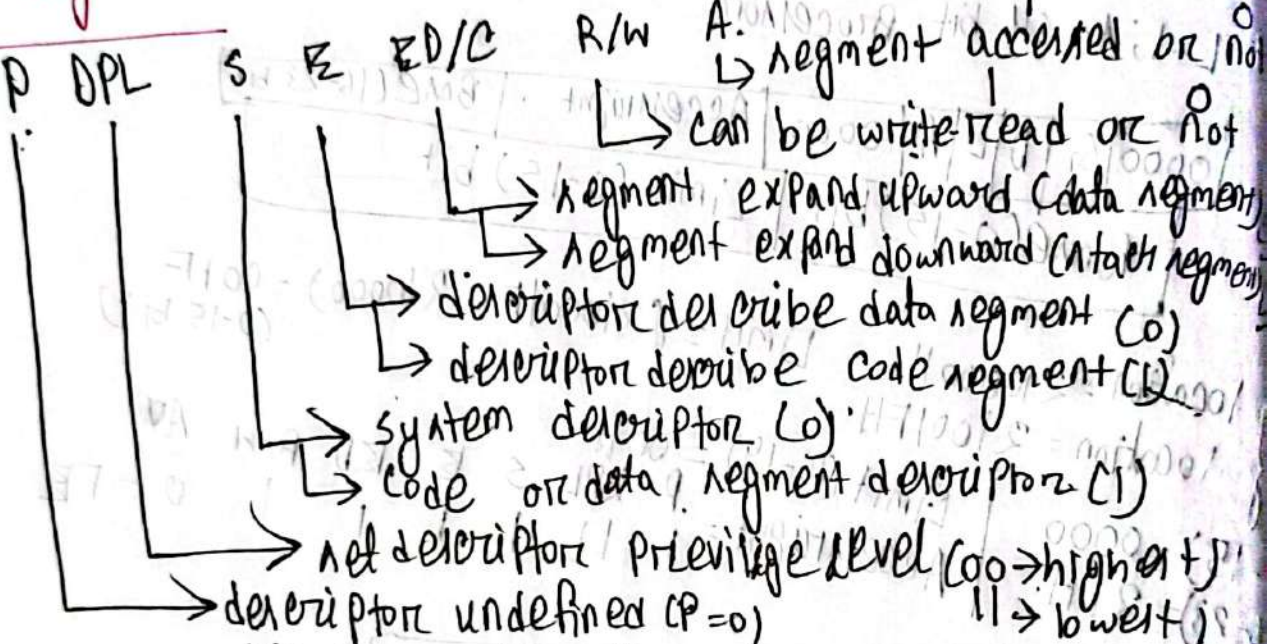
Granularity = G = 0 = limit in byte (1 MB)
G = 1 = limit in 4 KB Block (4 GB)

Default operand size, D = 0 = 16 bit mode = 8086 / 80286.
D = 1 = 32 bit mode = 80386+ instruction.

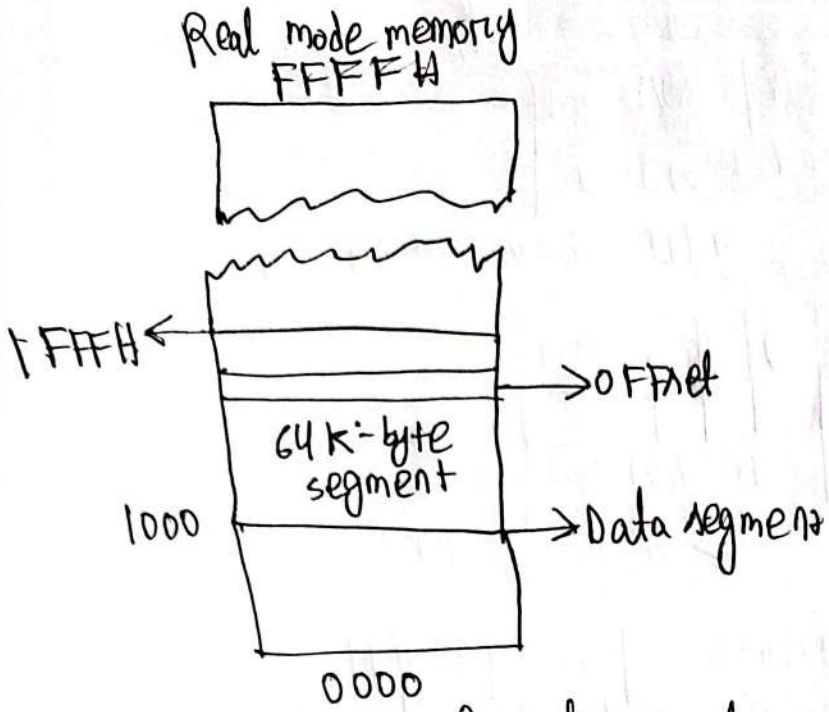
L (64 bit code segment) L = 0 = 32 bit code segment
L = 1 = 64 bit mode.

AV = 0 = segment not mark as available.
AV = 1 = segment mark as available.

- Access right



* The address of global descriptor table register contains the address of global descriptor table and it's limit. It contains 32 bit base address and 16 bit limit.
* TLB (translation look aside buffer) translate, virtual address in corresponding physical address.
* Paging gives ⇒ traditional, operating system, virtual memory system, memory protection, swaping.



descriptors contains information about segment including its starting address, length and access right.

Available bit can be use for custom purpose.

In flat mode entire memory segment appear as a block of memory. It provides linear and unified address space.

Ch-4

ypes of Push / Pop instruction ⇒

egisters → Push or Pop contents of CPU registers.
emory → Push or Pop 16/32 bit data from memory.
mediate → Push a constant (immediate data) onto the stack.
Cannot Pop into ES.
lags → PUSHF / POPF (16 bit) or PUSHFD / POPFD (32-bit)
ll registers → PUSHA / POPA (16 bit) or PUSHAD / POPAD (32 bit).
64 bit mode does not support PUSHA / POPA

ack operations :-

86-80286 → PUSH/POP always move 2 byte.
386 and above → can move 2 byte/4 byte. depending on register/memory size.
Most-significant byte (MSB) → goes to [SS:SP-1]
Least-significant byte (LSB) → stack segment = stack Pointer - 2

or 32 bit ⇒ 12345678

SP-4 → 0X78	SP-2 = 0X34
SP-3 → 0X56	SP-1 = 0X12 (MSB)

SP decrement by 4.

op

hen we do Push SP decreases when we do Pop SP increase.

tack segment (SS) → stack to segment address.
tack Pointer (SP) → stack to top of stack
Stack memory 10000H - 1FFFFH.
SS = 1000 H.
SP = 0000 H, SP = FFFF H.

Ex:4-1

DW ⇒ defined word.
100H (hexadecimal) initialized
so 0 to 99 words are 2 bytes
so total initialized.
200H.

Ex:4-2

MASM → microsoft macro assembler
automatically initialized SS and SP
and allocate 200H bytes for stack.

LEA → Load effective address

loads a register with the offset address of a memory
operand, not the data itself.

difference between LEA and MOV/OFFSET:-

LEA BX, [DI] → Loads BX with offset address in DI → Microprocessor calculates the address.

MOV BX, [DI] → Loads BX with data of address DI → Loads the contents, not address.

MOV BX, OFFSET LIST → Loads BX with offset → Faster than LEA

LEA BX, LIST → Loads BX with offset of LIST → slower than MOV OFFSET LIST because CPU computes address of LIST

LEA SI, DATA1 ; SI = offset address of DATA1

MOV DI, OFFSET DATA2 ; DI = offset address of DATA2 (faster)

LEA → load address, LDS = Data segment Load

* LDS reg, mem → reg + DS → 32 bit or 16 bit offset →
LES reg, mem → reg + ES → 11
LFS reg, mem → reg + FS → 32 bit → only 80386+ (Protected mode)
LGS reg, mem → reg + GS → 11 → 11
LSS reg, mem → SP + SS → 11 → used for stack manipulation.

2 bit memory section → 16 bit offset + 16 bit segment
8 bit memory section → 32 bit offset + 16 bit segment.

OFFSET FIRST → SEGMENT NEXT

11000H → BX (offset low) 11002H → DS (segment low)
11001H → BX (offset high) 11003H → DS (segment high)

ADDR → we will store old stack address (stack pointer and segment)
AREA → allocate new stack memory.

TI → set interrupt flag
e enabled the interrupt; cause ss/sp change time we should
f interrupts.
* we can restore old ss and sp together by LSS.

Ch-6

* A near jump changes the IP only on the other hand a far jump changes the IP and CS.
* For Jmp instruction → 1 byte opcode + 2 byte offset + 2 byte for segment
(5) In 80386-core2 Processor a near jump can go ± 2GB because of 32 bit displacement
(7) A label followed by colon is a name memory address. It marks the target location for JMP or CALL instructions
(8) A near jump only change the IP register
(10) JMP Ax means jump to the address stored in AX register. The cpu loads IP with the value of Ax. It is near or indirect jump because it jumps within the same code segment
(11) JMP DI → Jump in the address contained in DI.
JMP [DI] → Jump to the address stored in memory location pointed by DI. It is near indirect jump takes offset from memory.
(12) JMP FAR PTR [DI] → For indirect jump takes both CS+IP from memory.
(13) Five flags test in Conditional jump → P, C, Z, S, O
(14) JA ⇒ Jump if first number is greater after comparison
(16) For sign number jump instruction → JL, JLE, JG, JGE
(17) For unsign " " ⇒ JA, JAE, JB, JBE

JAE, JBE check C and C flag bits.

SETE or SETZ the destination (like AL) to 01H if ZF=1

23, 24 ⇒ 8086 loop decrement CX. registers.

Pentium 4 " ECX "

Core2 64 bit mode " RCX "

```
MOV CX, 150H
MOV DI, OFFSET DATA2
MOV AL, 00H
LOOP_Store:
    MOV ES:[DI], AL
    INC DI
    LOOP STORE-LOOP.
```

25 LOOPE will continue it's loop while CX≠0 and ZF=1, means equal and terminate CX=0 or ZF=0.

```
CMP AL, 3
JNE SKIP:
ADD AL, 2
SKIP:
```

```
; DS:SI → BlockA , ES:DI → BlockB.
.REPEAT
    LODSB ; AL = [SI] ; LODSB ⇒ load string byte.
    STOSB ; [DI] = AL
.UNTIL AL == 00h.
```

.while1 makes an infinity loop until a break point.

```
; DS:SI ⇒ BLOCKA , ES:DI ⇒ BLOCKB
WHILE 1
  LODSB ;
  STOSB
```

```
MOV AL, [SI]
MOV BL, AL
MOV BL, [DI]
```

~~BREAK IF BL == 12~~

- BREAK . IF BL == 12
  INC DI
- ENDW.

(31) Break is used to exit current loop or task.

(32) Procedure is a specific block of code to do a specific task starting with CALL and ends with RET.

(34) last executable instruction in a procedure must be RET.

(35) NEAR RET function pops IP from stack and returns to the p called in the same segment.

(36) PROC NEAR / PROC FAR declaration identified as 'near or far

(38) CUBE_CX PROC NEA
Push AX
Push DX
} → multiply result DX, AX

```
mov AX, CX
MUL CX
MUL CX
mov CX, AX
pop DX
pop AX
ret
CUBE_CX ENDP.
```

(9) RET 6 return and clean up or pop the return address then discard 6 more bytes.

(40)

```
MUL-AX PROC NEAR
    Push bx
    Push dx
    Push DI
    Push SI

mov AX, DI
MUL SI
mov BX, looh
DIV BX

POP SI
 POP DI
  POP dX
   POP Bx
ret
MUL-AX ENDP
```

(41)

```
SUMY PROC NEAR
    XOR EDI, EDI
    ADC EAX, EBX
    ADC EAX, ECX
    ADC EAX, EDX
    ADC EDI, 0
    ret
SUMY ENDP
```

(42) Interrupt stops the current program and run a small service routine (ISR) then return.

(43) The software calls interrupts are ⇒

INT n ⇒ INT 21h ⇒ call Dos services.

INT 3 ⇒ 1 byte break point interrupt use for debugging.

INTO → interrupt overflow trigger an ISR if OF = 1.

There are 256 types interrupt 00h-FFh.

(45) Real mode interrupt vector 4 bytes.
IP: 2 bytes: handlers offset
Cs: 2 bytes: handlers segment.

(46) CPU fetches Cs:IP from vector to Jump to ISR.

(47) Interrupt vector type 0 use for divide error (division by zero or quotient overflow.

(47) RET comes back to the address.
IRET comes back to the address with the flags bit.

(48) IRED (interrupt return doubleword) is used in 32-bit Protected It pops EIP, Cs and EFlags from stack to return from an interrupt. IRETQ is the 64 bit version of IRET.

(50) INTO triggers interrupt only if overflow flag = 1.

(51) Each interrupt vector 4 bytes.
Address = 4 x 400H = 1001H.
Interrupt vector 40H → stored at memory 0000:0100H - 0000:0103H

(52) INTR Pin is used for hardware interrupt.
STI (set interrupt flag) → enables hardware interrupt (I = 1)
CLI (Clear interrupt flag) → disable hardware interrupt (I = 0)
* The WAIT instruction test BUSY pin.

(54) BOUND compares a register value with lower and upper limit in memory. It causes interrupt type 5 if the register is < lower or > upper bound.

6. When ENTER instruction execute the Base Pointer register is Pushed to the stack.
7. The ESC (Escape) instruction sends opcode to numeric coprocessor

Ch-9

1. difference of 8086 and 8088

8086	8088
① Data bus 16 bit	① Data bus 8 bit
② speed faster as 16 bit transfer	② speed slower as 8 bit transfer.
③ Control pin M pin	③ control pin M/IO pin.
④ It is costly	④ It is cheaper.

2. 8086, 8088 are TTL-Compatible. They work with TTL logic families like 74LS or 74ALS because their output voltage and current match TTL requirements.
3. Fan out means how many inputs one output can drive. 8086/8088 can drive upto 10 standard logic loads without a buffer.
74 → high input current per gate → 1 Fan-out
74ALS → Very low input current → 10 Fan-out
74HC → Extremely low input → 10 or more Fan out
4. ALE=1, the address-data bus carries the address.
5. S_3 and S_4 shows which segment register is being used for current operation.

S4	S3	Segment
0	0	Extra
0	1	Stack
1	0	Code/No
1	1	data.

(6) RD=0; Processor reading data from memory/I/O
RD=1; inactive

(7) TEST Pin is checked by the WAIT instruction.
If TEST=1 → CPU continues execution.
TEST=0 → CPU waits until it becomes 1.

(8) CLK is the clock signal input. It must have 33% duty cycle: high for 1/3 of the period, low for 2/3 of the period.

(9) MN/MX=Grounded means 8086 works in maximum mode coprocessor with 8087.

(10) WR=0; the 8086/8088 writes data to memory or I/O
The strobe means "Data is valid on the bus right now". memory or I/O device should latch (save) the data at that moment.

(11) ALE goes to high-impedance when HOLD signal is active that means another device is using the bus.

(12) DT/R=1, CPU is receiving data from memory / I/O.
DT/R=0, CPU is transmitting data.

(2) HOLD=1, an external device requests the system. The CPU stops using the bus and responds with

3 minimum mode pin → S2,S0 I/O/M and DT/R needs to decoded whether CPU is halted or not for 8088 bus cycle.

S2	I/O	M	DT/R	
0	0	0	0	Interrupt Acknowledge
0	0	1	1	Memory read
0	1	0	0	memory write
0	1	1	1	HALT
1	0	0	0	Opcode Fetch
1	0	1	1	I/O read
1	1	0	0	I/O write
1	1	1	1	Passive

The LOCK pin prevents other device from using the system bus temporarily. It becomes active when the CPU executes an instruction with the LOCK prefix for atomic operation.

QS1 and QS0 queue status bit indicate:

QS1	QS0	
0	0	Queue idle
0	1	First byte of opcode fetch
1	0	Queue empty
1	1	Subsequent byte of opcode

8284A provides three main function⇒

1) generates clock signal (CLK and PCLK)
2) generates RESET signal (power-on or manual)
3) synchronized READY (for wait-state control)

(18) divide factor of 8284A clock generator
→ 8284A divider the crystal ~~generat~~ frequency by 3 to get CPU clock and then divider it again to get peripheral clock.

(19) when F/C=1 the internal crystal oscillator is disable and the external timing signal is connected to the EFI (external frequency inputs pin.

(20) crystal = 14 MHz
so CLK = $\frac{14}{3}$
so PCK = CLK ÷ 2 = $\frac{14}{6}$ = 2.33 MHz

(21) when RES=0 it reset 8284A and when RES=1 it restart the CPU.

(22,23) In 8086 AD0-AD15 and A19/S6-A16/S3 lines are multiplexed we for both address and data status, they are demultiplexed using latches and In 8088 AD0-AD7 and A19/S6-A16/S3 pin are multiplexed we we 74LS373 to demultiplex the system

(25) BHE → Bus high enable signal indicate that the high byte of data bus is active.

(26) Buffers are used to increase driving capacity of signals because each output can drive only about 10 loads. (fanout ≤ 10) - Buffer prevents signal degradation when multiple devices are connected

(27) The DT/R pin controls the direction of data in the 74LS245 bidirectional bus buffer.
DT/R = 1 → CPU reads (input to CPU)
DT/R = 0 → CPU writes (output from CPU).

(28) A bus cycle is equal 4 clock period.

(29) CLK=4 MHz, bus cycle?

1 clock period = $\frac{1}{4}$ = 0.25 μs

Bus cycle = 4 × 0.25 μs = 1.0 μs

(30) Two operation during bus cycle →

① Address transmission (Placing address on bus)

② Data transfer (read or write between CPU and memory / I/O.

(31) 8086/8088 can achieve 0.33 to 1 MIPS at 10 MHz

(32) Purpose of each T states ⇒

T_1 → Address is Placed on the bus, ALE becomes active to latch address

T_2 → Data bus is prepared to read/write, control signal become valid.

T_3 → Actual data transfer happens CPU reads or writes data.

T_4 → Bus is released and ready for next cycle.

T_w → wait state – added if READY = 0 (slow memory)

(33) At 5 MHz, clock period 1/5 = 0.2 μs

Bus cycle = 0.2 × 4 = 0.8 μs

So memory access time = 800 ns.

(34) DEN = Data enable is active 2 clock cycle during data transfer.

At 5 MHz, clock cycle = $\frac{1}{5}$ μs

So the width = $\frac{1}{5}$ × 2 = 0.4 μs

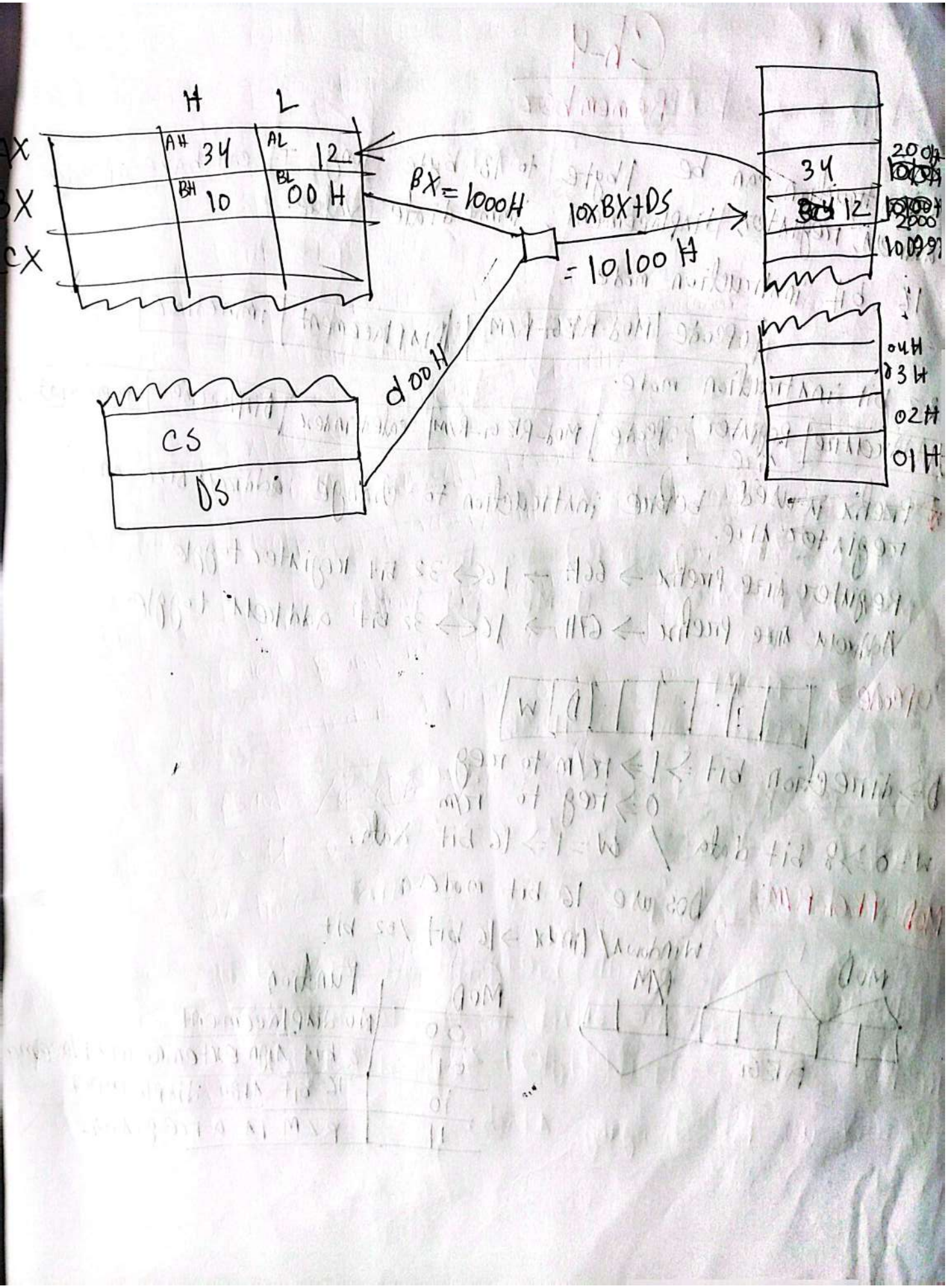
(35) If READY is grounded it introduces wait state (TW) in to the bus cycle cpu waits until it become 1.

(38) Minimum vs MAX mode.

Minimum	MAX
① select by +5V	① select by GND
② single Processor	② Multiprocessor or with coprocessor
③ Generated internally by cpu.	③ Generated externally by 8288 Bus controller.
④ RD, WR, ALE	④ S_2-S_0 → decode by 8288
⑤ use in small system	⑤ Large / multiprocessor system.

(39) 8288 Bus Controller (in MAX Mode) :-

S_2	S_1	S_0	
0	0	0	interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	HALT
1	0	0	opcode fetch
1	0	1	memory read
1	1	0	memory write
1	1	1	Passive.


H
L
AX
BX
CX
AH 34
AL 12
BH 10
BL 00 H
BX = 1000H
10xBX+DS
= 10100 H
d00H
CS
DS
34
12
2000
04H
03H
02H
01H

Ch-4

Remember

* instruction can be 1byte to 13 byte long. It can have opcode address, register, displacement, immediate value.

16 bit instruction mode

opcode	Mod-REG-R/M	Displacement	immediate

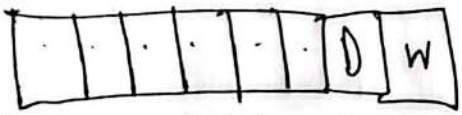
32 bit instruction mode.

Address size	Register size	opcode	Mod-REG-R/M	Scaled index	Displacement	immediate

* Prefix is used before instruction to change address size and register size.

Register size prefix → 66H → 16 ↔ 32 bit register toggle

Address size prefix → 67H → 16 ↔ 32 bit address toggle



D ⇒ direction bit ⇒ 1 ⇒ r/m to reg

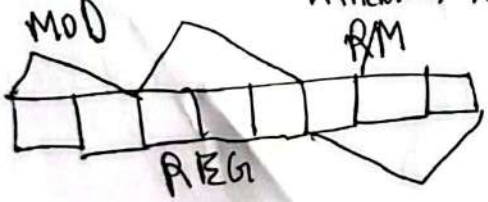
0 ⇒ reg to r/m

W = 0 ⇒ 8 bit data / W = 1 ⇒ 16 bit data.

MOD - REG - R/M :-

Dos use 16 bit mode

Windows / linux → 16 bit / 32 bit



MOD	Function
00	No displacement
01	8 bit sign extended displacement
10	16 bit sign displacement
11	R/M is a register.

displacement 8 bit then we need sign-extend to make [6 bit]

7FH ⇒ Positive → 00H will add on left

FFH ⇒ negative → FF H will add on left.

MOD = 11 → R/M select, register means Register to Register

MOD ≠ 11 ⇒ Memory addressing mode select.

	W=0	W=1
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

R/M Combination

Bits	Effective Address	Description
000	[BX + SI]	Base + Index
001	[BX + DI]	Base + Index
010	[BP + SI]	Base + Index
011	[BP + DI]	Base + Index
100	[SI]	Index only
101	[DI]	Index only
110	Direct address.	[BP + Displacement]
111	[BX]	Base only

R/M Code with addressing mode

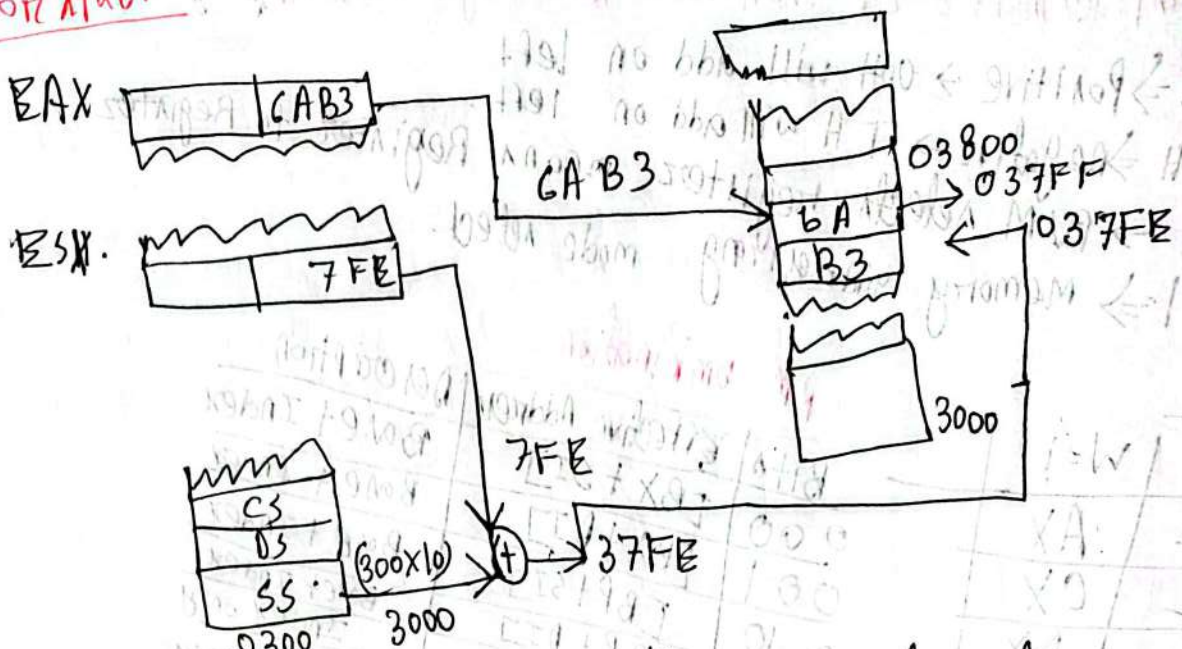
00	DS: [BX + SI]
01	DS: [BX + DI]
10	SS: [BP + SI]
11	SS: [BP + DI]
00	DS: [SI]
01	DS: [DI]
10	SS: [BP]
11	DS: [BX]

MOV DL, [DI + 1000H]

8A 95 00 10 ⇒ little endian

MOD	REG	R/M	Displacement
10	DL	[DI]	1000 H

For stack :



*we can't POP code segment. stack is in cyclic nature.

*LEA (load effective address) load the address of data in register.

MOV AX, NUMB. NUMB DW 1234H

⇒ AX = 1234H

LEA AX, NUMB

⇒ AX = offset address of NUMB

MOV BX, OFFSET LIST ⇒ Assembler → 1 cycle → faster.

LEA BX, LIST → Processor → 2 cycle.

* LDS, LES, LFS, LGS, LSS brings the offset address and segment address from far address.

LDS reg, mem ⇒ it will bring offset and segment from memory and give offset to register and segment to DS.

in far address ⇒ offset (4 byte) + segment (2 byte)

LSS is use for changing stack.

lag ⇒ direction flag. D=0 ⇒ Auto increment D=1 ⇒ Auto decrement

S ⇒ memory to memory (Move string)

⇒ Port to memory (Input string)

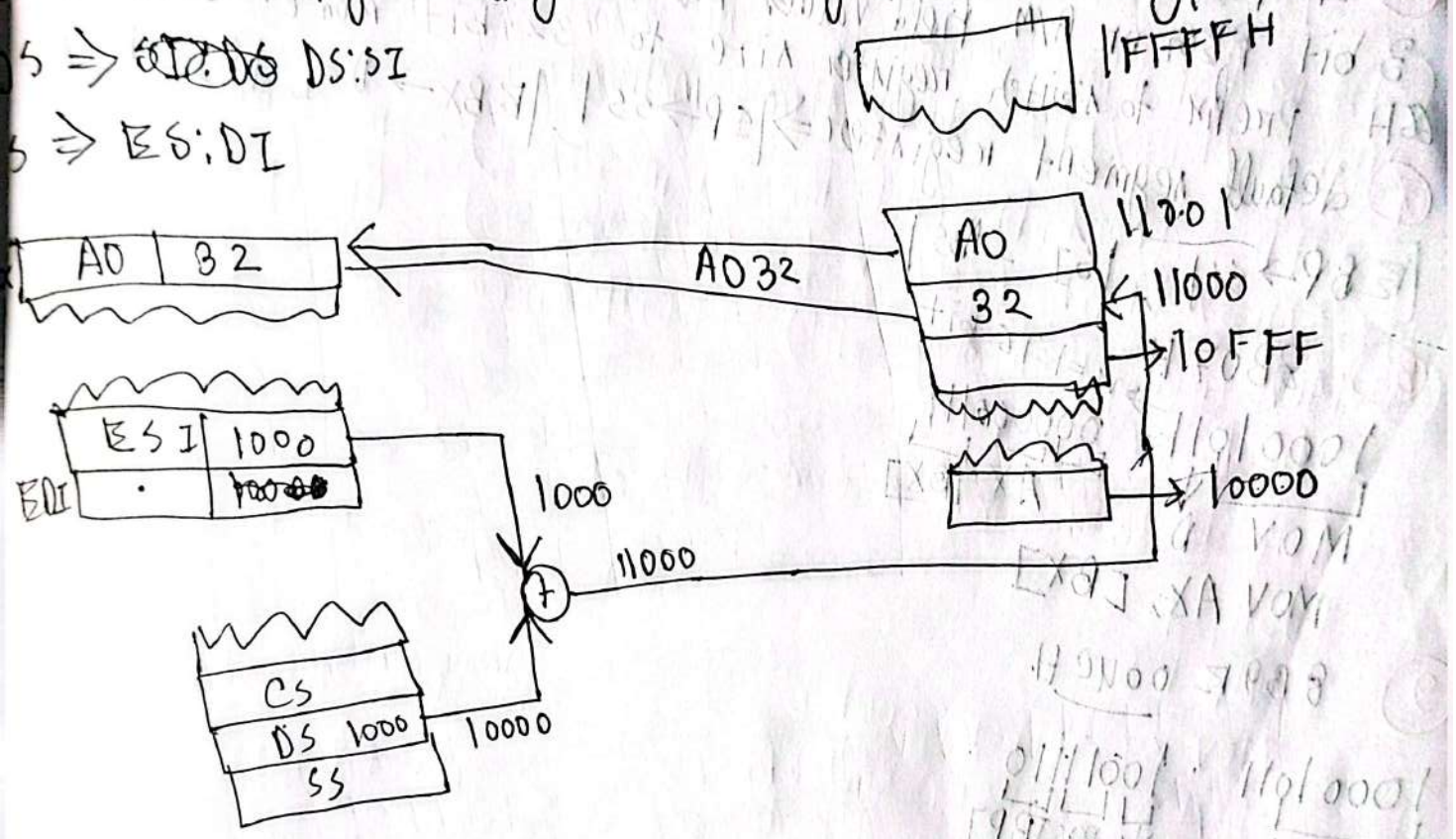
⇒ memory to Port (Output string)

S ⇒ load string → brings data from memory to register.

⇒ store string → bring data from register to memory.

S ⇒ ~~DS:DI~~ DS:SI

⇒ ES:DI



Q5 - Solve

(3) MOD mode specifies addressing mode whether the data is in register or memory and if memory use a displacement.

(5) In pentium 4, 32 bit registers are selected by using same 3 bit REG or R/M field values but the instruction includes a 66H prefix to switch register size to 32 bit mode.

(7) default segment register ⇒ |SP → SS|, |EBX → DS|, |DI → ES|, |EBP → SS|, |SI → DS|

(8) 8B07H → w=1 ⇒ 16 bit

1000|011 · 00000111

MOV D AX [BX]

MOV AX, [BX]

(9) 8B9E 00 4C H

1000|011 · 10011110 ·

MOV BX [BP]

16 bit displacement

MOV BX, [BP + 004CH]

(10) MOV SI, [BX+2]

1000|011 00|110 111

Displacement 02H

8B 77 02

(11) MOV ESI, [EAX]

= 8B . 00.110 000

⇒ 8B 30 For 32 bit register 66 8B 30

prefix.

REX Prefix is use in 64 bit mode to exten register size and access new 64 bit registers.

MOV AX, 1000H | MOV DS, 1000H ⇒ not correct as segment register
MOV DS, AX | can't load immediate value.

PUSHA pushes all the 16 bit general register in this order ⇒
AX, CX, DX, BX, SP, BP, SI, DI

PUSHAD Pushes 32 bit registers.

Instruction → operation.

PUSH AX → Decrement SP by 2 and Places AX on the stack
POP ESI → Takes 4 bytes from stack into ESI
PUSH [BX] → Pushes 16 bit data from memory at [BX] onto stack
PUSHFD → Pushes 32 bit EFlags register onto the stack.
POP DS → Loads DS register with 16 bit value from stack
PUSH 04 → Pushes immediate value 00000004H onto stack.

PUSH EAX where SS = 0100H SP = 0100H
Push EAX Stack grows downward 4 value.
SS:SP = 0100:(0100-4) = 0100:00FCH.

[SS:SP] = AL
[SS:SP+1] = AH
[SS:SP+2] = middle byte
[SS:SP+3] = high byte.
Data stored at [0200:00FCH], to [0200:00FFH]

(28) LDS BX, NUMB
here we load 4 byte, 2 bytes added to BX as offset and last two bytes added to DS as segment.

(29) LDS → loads data segment → for normal data addressing
LSS → loads stack segment → use for stack setup or switching

(31) Direction flag controls the auto update direction of SI/ESI/RSI and DI/EDI/RDI during string instructions.

(32) CLD → clears DF (sets forward/increment) ⇒ make DF = 0
STD → sets DF (sets backward/decrement) ⇒ DF = 1

(33) MOVS (memory → memory copy) and CMPS (memory ↔ memory compare) use both DS:SI and ES:DI

(36) Copy 12 bytes from source to DEST

```
PUSH DS
PUSH ES
CLD
MOV SI, OFFSET SOURCE
MOV DI, OFFSET DEST
MOV CX, 12
REP MOVSB.
```

(37) REP use prefix for decrementing counter CX. to zero
REPZ ⇒ CX ≠ 0 and ZF = 1 repeat
REPNZ ⇒ CX ≠ 0 and ZF ≠ 1 repeat.

(41) INSB instruction we have the port number to DX.

(42) LAHF and SAHF means load AH from flags and store AH to flag

XLAT translate AL => XLAT treats AL as an index into a lookup table whose base is in BX replacing AL with table entry

AL := [DS:BX+AL]

মেমরি থেকে ডেটা পড়তে MOV => MOV AL, [address]

I/O device থেকে ডেটা পড়তে IN => IN

IN AL, 12H. => Reads one byte from I/O port 12h into AL

Immediate Port form.

OUT DX, AX => Takes 16 bit data from AX to I/O Port address by DX

Low byte goes first on the bus.

segment override prefix forces a memory operand to use non-default segment MOV AL, ES:[BX]. we can ES override instead of default DS/ES rules.

MOV AX, ES:[BX]

language directive tell the assembler how to arrange, define or reserve memory/segment which is not actual CPU instruction.

DB 25 → 25 is stored in 1 byte

DW 1234H → 1234 is store in 2 byte.

CMOVNE CX, DX

Conditional move if not equal

ZF=0 means copy DX to CX. else CX is unchanged

DW => define word

30 byte for array LIST1

LIST1 DB 30 DUP(?)

We use EQU to fix a value for name for assembler time not for runtime as CPU doesn't take it.

(56) .Model selects memory model.

(59) INT 21h with AH=4Ch do terminate the program and return to DOS, AL holds the return code.

Ch-5

Solve

1. add the data address by SI to AL → ADD AL, [SI]

Add CX to memory variable FROG → ADD WORD PTR FROG, CX

Add 234H to RCX → ADD RCX, 234H.

*** segment registers can't be use in Arithmetic or logic (Add, AND, OR...) instruction as destination or source

1001 + 20FF
= 3100H here we had half carry so A=1. CAF=1

INC [BX] in meaning we donot know the size of memory. Size is ambigious, specify the size ⇒ INC BYTE PTR [BX]

Subtract 10 words after SI from CH ⇒ SUB CH, WORD PTR [SI+10]
↳ 10 words = 20 bytes

Subtract AL from memory FROG ⇒ SUB FROG, AL

SBB [DI+4], DX ⇒ subtract DX with borrow (carry flag) from the word in memory at address [DI+4]. Result store back in memory.

SUB → destination = destination - source ⇒ store result → arithmatic subtraction.

CMP → only set flags ⇒ No store result → Bit comparison.

Two 8 bit number multiple result in AX (16 bit)

Two 16 bit " " result in DX : AX.

LSB MSB

If upper half ≠ 0 C=1, O=1.

If Upperhalf = 0 C=0, O=0

IMUL BX, DX, 100H

⇒ signed multiply BX = DX * 100H

ivide 8 bit number:

AX / 8 bit ⇒ AL = quotient

AH = remainder.

6 bit number divided:

DX.AX / 16 bit ⇒ quotient AX, Remainder DX

Two main division error:

1) Divide by zero. Divisor = 0
2) Divide overflow. result too large to fit in quotient register.

→ CPU triggers interrupt.

3) divide CL by CL, multiply result by 2 to force final 16 bit in DX

```
MOV AL, BL          XOR AH, AH          MUL BL
DIV CL              MOV AL, BL          MOV dx, ax.
MUL 2H              div CL
MOV DX, AL.         MOV BL, 02H
```

3) DAA (decimal adjust after addition) and DAS (decimal adjust after subtraction) in use with BCD arithmetic.

4) AAM (ASCII Adjust after Multiplication) : it splits a Binary product in AL into two decimal digits:

High digit → AH

Low digit → AL

So binary → unpacked BCD form.
AL = 25 → AAM ⇒ AH = 2, AL = 5

35) Instruction use in ASCII Arithmatic operation?

AAA ⇒ Adjust after addition
AAS ⇒ Adjust after subtraction
AAM ⇒ Adjust after multiplication
AAD ⇒ Adjust after division

36) Convert unsigned number in AX to 5 digit BCD number.

AX contains a 16 bit binary number we must convert it into decimal digit and store each digit separately.

① Load AX with number and BX with starting memory address where digits will be stored.

② Divide AX by 10000 (decimal)
- Quotient in the most significant digit (D_4)
- Remainder = rest of the number
- Store Q_0 at [BX]

③ Then divide the remainder (R_0) by 1000, get the next digit (D_3) and store at [BX+1]

④ Repeat the process with 100, 10 and finally 1.

Each time store the quotient in [BX+2], [BX+3], [BX+4]

⑤ The last remainder after dividing by 10 is the least significant digit D_0.

8 digits packed by 4 bytes using AX:BX and CX:DX

1) starting from the least significant byte (lower bytes in DX and BX)
   - Add BL+DL
   - Apply DAA (decimal adjust after addition) so the result remain valid in BCD.
   - Save result in DL and keep any carry in CF
2) Move to next byte (BH+DL) add them with ADC.
   - again we DAA to adjust decimal digit.
3) Repeat for the next two bytes (AL+CL, AH+CH) adjusting them with DAA each time.
4) After 4 such additions all decimal digits are correctly added
5) The final 8 digits is saved in CX:DX

Clear 3 leftmost bit of DH store in BH

```
MOV BH,DH
AND BH, 1Fh
```

14) ⇒ AND X with Y
AND Y, X

14) to invert we need XOR with that bit and 1.

15) AND Computes logical AND stores the result in destination & updates flags.
TEST computes logical AND to set flag does not modify test.

13) e) XOR WORD PTR [EBX], DX ⇒ here memory is destination.