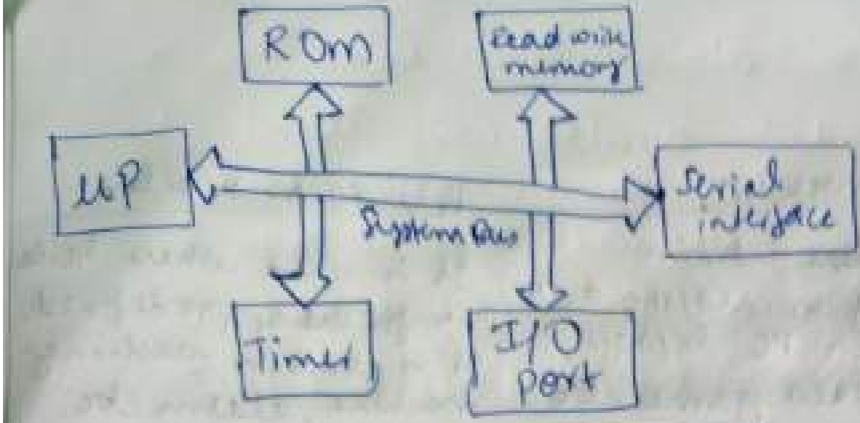
1.

Parameter	Microprocessor	Microcontroller
1. Application	Used where intensive processing is required It is used in PC, laptops, mobiles, video games etc.	It is used where task is fixed & predefined. It is used in washing machine, alarm etc.
2. Structure	It has only the CPU in chip other devices like I/O port, memory, timer is connected externally Structure of µP is flexible, Users can decide the amt of memory, the no of I/O port & other peripheral devices	CPU, memory, I/O port & all other devices are connected on single chip Structure is fixed. Once is is defined the user can't change the peripheral devices
3. Clock speed	Clock speed is high. It is terms of GHz. Ranges from 1GHz to 4GHz	Clock speed of µC is less. It is terms of MHz. Ranges b/w 1MHz to 300MHz
4. RAM	RAM is in range of 512MB to 32GB	volatile mem (RAM) is in range of 2KB to 256KB
5. ROM	Hard disk (ROM) for µP is range of 128GB to 2TB	Hard disk or flash mem. (ROM) is in range of 32KB to 2MB
6. Peripheral interface	USB, UART, high speed ethernet	I2C, SPI, UART
7. Programming	The program of µP can be changed for different applications. Programming is difficult compared to µC.	Program can is fixed once its designed
8. Bit size	available in 32bit & 64bit	available in 8-bit, 16 bit and 32 bit
9. Cost	Higher compared to µC	Cheaper.
10. Power consumption	High	Low
11. Size	overall size is large	Overall size is small
	µP is heart of computer system	µC is heart of embedded system

Scanned with CamScanner



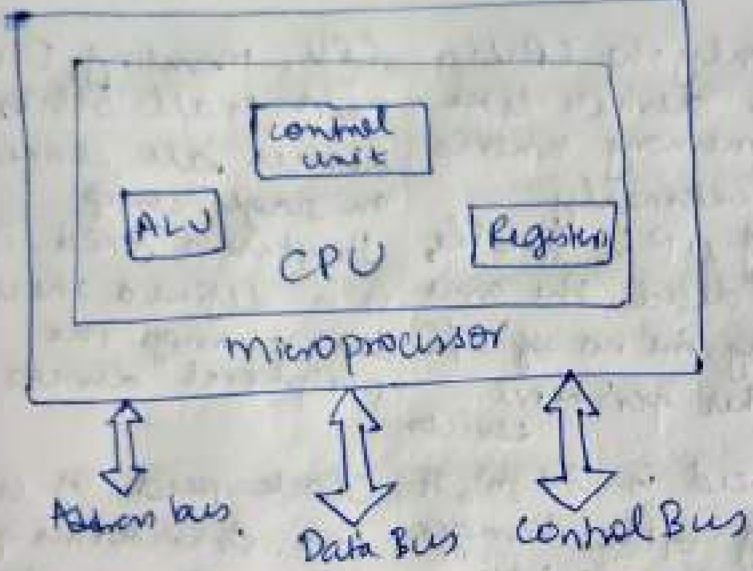


Fig:- Micro processor

Microcontroller	Read only memory	Read write memory
Timer	I/O port	Serial Interface

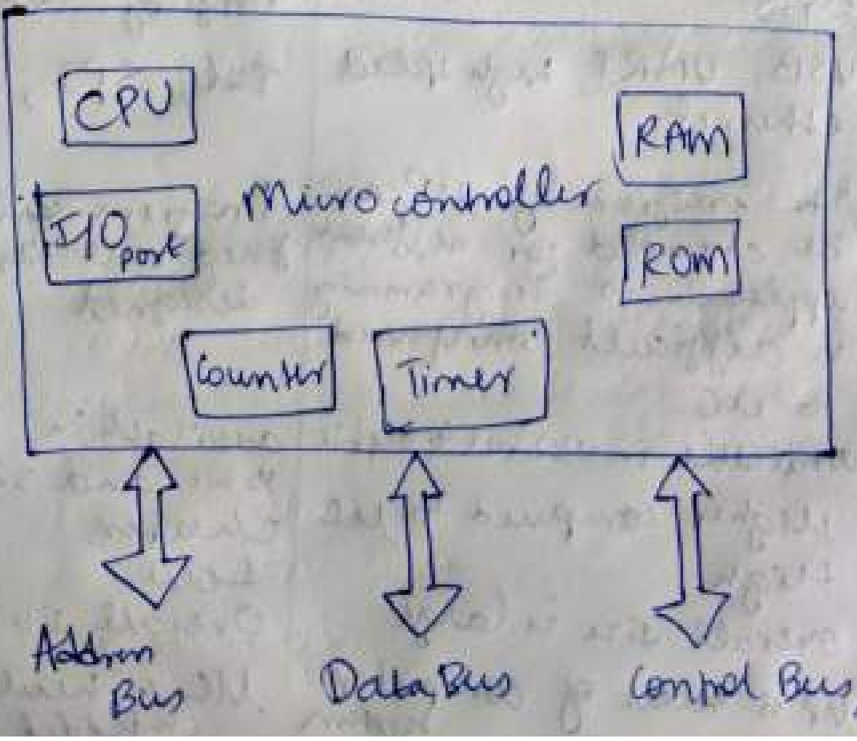


Fig: microcontroller

2.

Microprocessor	Word length	Memory addr capacity	No of Pins	Clock	Remark
8085	8 bit	64kB	40	3-6MHz	Popular 8 bit uP
8086	16 bit	1MB	40	5-8MHz	Widely used in PC
Pentinum	32 bit	4GB real 32 bit addr, 64 bit data bus	237 Pin Grid Array (PGA)	60 - 200 MHz	Contains 2 ALUs, 2 caches, FPU (Floating Pt Unit), 3.3 million transistors, 3.3V, 7.5 million transistors
Atom	64 bit	4GB	423 PGA	800 MHz -1.6GHz	
Core 2Duo	64 bit	4GB	423 PGA	1.9GHz	

3. (a) STM32F103C8T6

- Is a popular member of STM32F103xx medium-density performance line family of uC that feature a high performance ARM Cortex-M3 32 bit RISC core operating at 72 MHz frequency and process an extensive range of enhanced I/Os & peripherals connected to 2 APB buses.
- All members of STM32F103x family, including the C8T6, offer two 12 bit ADCs, three general-purpose 16-bit timers plus one PWM timer as well as std & advanced comm. interfaces upto 2 I2Cs & SPIs, 3 UARTs, a USB & a CAN.

(b) ATmega 328

- 1 kB EEROM, 2kB SRAM
- 23 general purpose I/O lines, 32 general purpose working registers

- 3 flexible timer/counters with compare modes,
- internal and external "interrupt"
- Serial programmable USART, a byte oriented 2-wire serial interface, SPI serial port, 6 channel 10 bit A/D converter (8 channels in TQFP & QFN/MLF packages)

(C) PIC 16F877A

Total no of pins - 40

Total no of ports - 5 (portA, portB, portC, portD, portE)

Operating voltage - 2 to 5.5V

No of I/O pins → 33

No of ADC pins - 14

ADC resolution - 10 bit

no of comparators - 2

no of timer - 3

Comm. protocols - UART, SPI, I2C

external oscillator - upto 20MHz

Program memory - 14KB

RAM - 368 bytes

EEPROM - 256 bytes max

PWM resolution - 10

Support both hardware pin & timer interrupt

4. 1st microprocessor was Intel 4004

Features

- max clock speed of 740kHz
- upto 92600 instructions per second
- separate program & data storage
- 12 bit addr
- 8 bit instructions
- 4 bit words

5. First microcontroller was TMS 1802

Features:

- had 5000 transistors
- 3000 bits of program memory
- 128 bits of access memory
- possible to "program" to perform a range of functions

6.

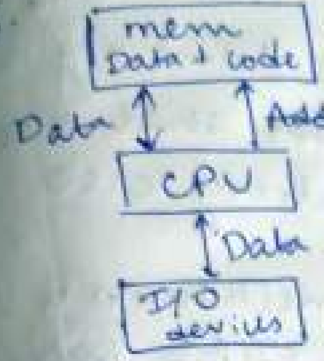


Fig: Von Neumann

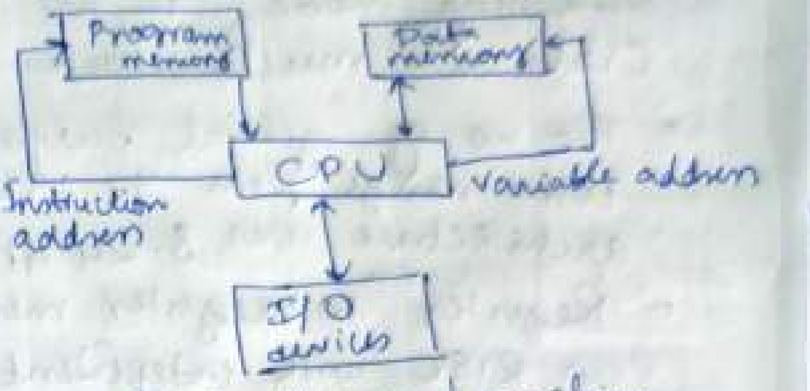
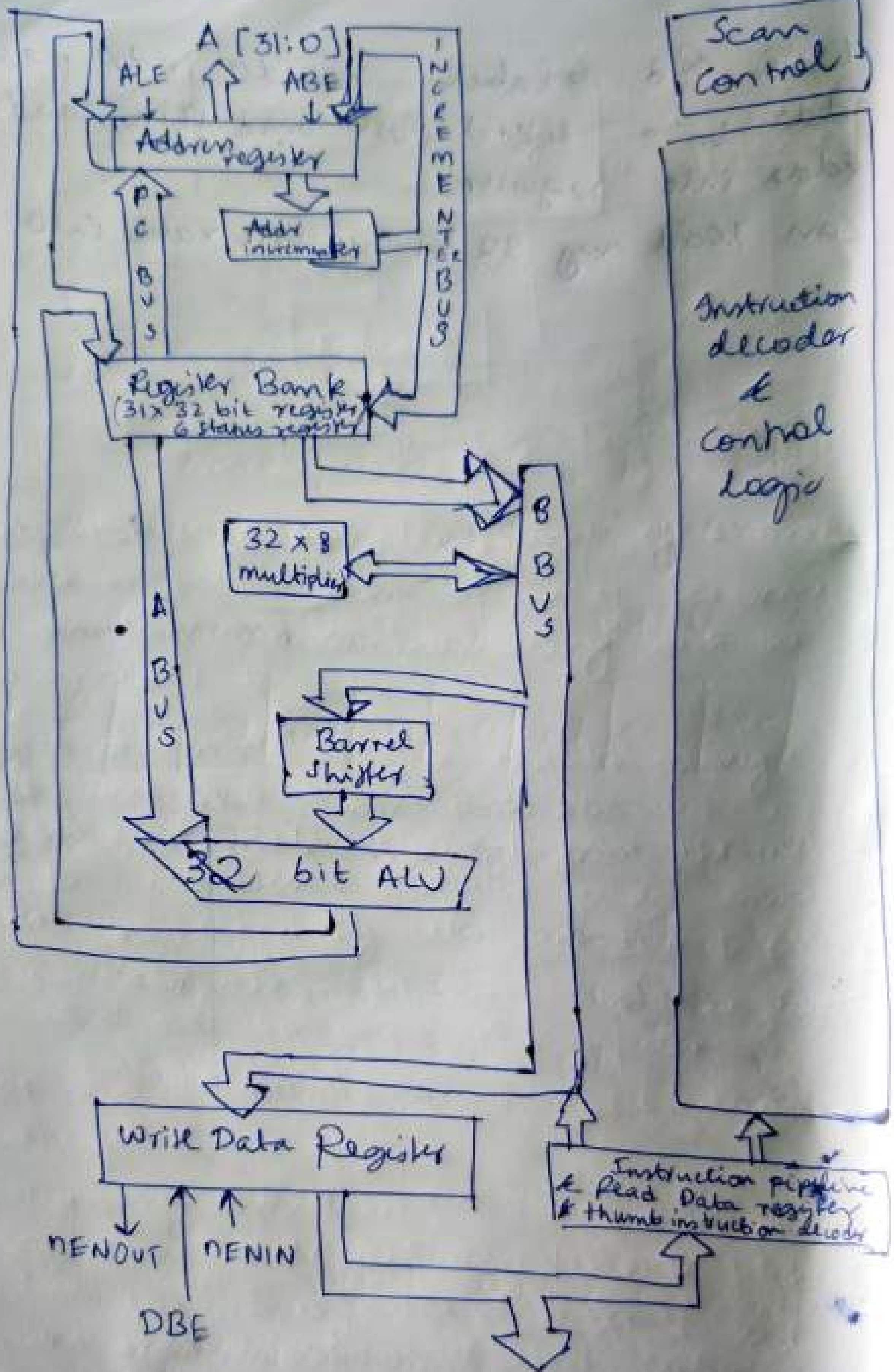


Fig: Havard machine

Von Neumann	Havard
1. The CPU can be either reading an instruction or reading / writing data from/to memory. Both can't occur at same time since instructions and data use same bus system	1. CPU can both read an instruction and perform a data memory access at same time, even without a cache.
2. Its much slower as it has single communication pathway	2. Its much faster for a given circuit complexity because instruction fetches and data access do not contend for a single memory pathway
3. It doesnot have distinct code and data address spaces.	3. Machine has distinct code and data address spaces
4. It is not possible to have two separate memory systems for a Von-Neumann architecture	4. It is possible to have two separate memory systems for a Havard architecture
5. It usually have a single unified cache which stores both instruction and data	5. It usually have a multiple cache, which stores both instructions and data separately
6. It needs extra memory all the time	6. It may not need external memory at all

I.


A [31:0]
ALE
ABE
Scan Control
Address register
INCREMENTER BUS
PC BUS
Addr incrementer
Register Bank
(31x32 bit register
6 status register)
Instruction decoder & control logic
B BUS
32 x 8 multiplier
A BUS
Barrel shifter
32 bit ALU
Write Data Register
Instruction pipeline
& Read Data register
& thumb instruction decoder
MENOUT
MENIN
DBE

- ARM processor has a 32 bit address bus. Hence it can access a total of 2^{32} = 4GB memory address space. It is based on von Neumann model, this space contains both program and data.

- PC (R15) gives a 32 bit address of instruction to be fetched. This address is put on the address bus & instruction is fetched from memory through the 32 bit data bus. PC is periodically incremented after every instruction is fetched.

- ARM 7 processor has a 32 bit data bus. This bus is used to both instructions as well as data. Both, instructions and data are 32 bits in size.

- If an instruction is fetched from memory then it goes to instruction decoder. The instruction decoder decodes the instruction and generates control signals for its ~~instruction~~ execution. ARM uses a hardwired control unit for decoding instructions.

- If data is fetched from memory then it gets stored to register file (R0 - R15). All registers are 32 bit each. If data is fetched is lesser in size than it is sign extended into 32 bits before being placed into register file.

- ARM7 processor has 32 bit ALU. It can perform 32 bit Arithmetic and logic operations. The operand for ALU can be obtained from registers (R0 - R15) only and status flags are updated correspondingly.

- ~~ALU~~ ARM 7 processor has barrel shifter. This is used to pre-shift operands before given to ALU

- Register file has 16 registers named R0 to R15. All are 32 bit registers and can be used as GPRs. Amongst them are some special registers like PC (R15), LR (R14) & SP (R13). Some of these registers are banked that means their use changes as the operating mode changes

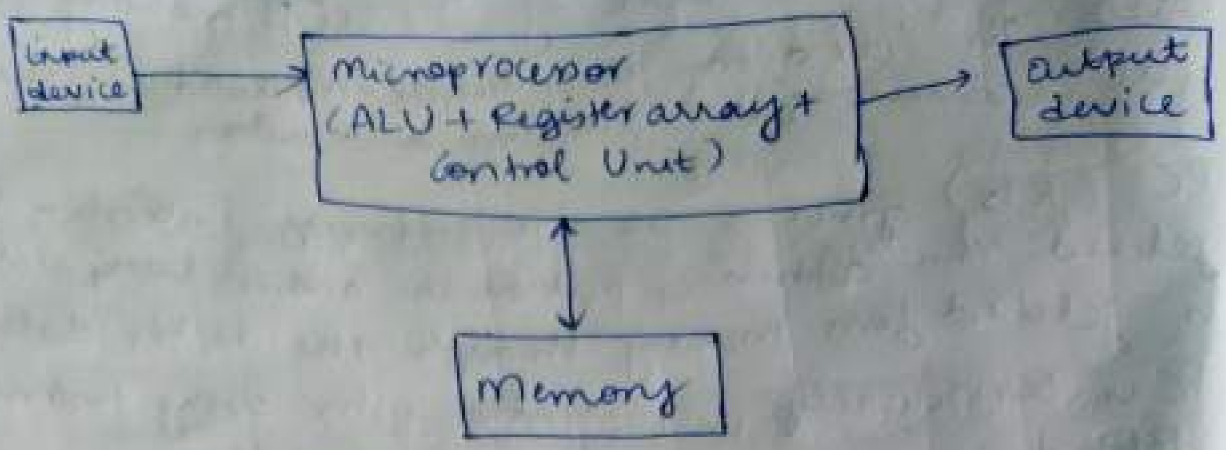


Fig: Block diagram of a Basic microcompu

Microprocessor is a controlling unit of a micro-comp ut fabricated on a small chip capable of performing ALU operations and communicating with the other devices connected to it.

Microprocessor consist of ALU, register array & a control unit. ALU performs arithmetic and logical operations on the data received from the memory or an ~~fo~~ input device Register array consist of registers identified by letters like B, C, D, E, H, L & accumulator. The control unit controls the flow of data and instructions within the computer.

Microprocessor follows a sequence - fetch, decode and the execute

Initially the instructions are stored in memory in sequential order. The microprocessor fetches those instructions from memory, then decodes it and executes those instructions till STOP instruction is reached. Later, it sends the result in binary to output port. Between these processes, the register stores the temporarily data and ALU performs the computing functions.

9.

Parameter	8085 microprocessor	8086 microprocessor
1. Data bus size.	8 bit	16 bit
2. Address Bus size	16 bit	20 bit.
3. clock speed	3mHz	Varies in range 5.8 - 10mHz
4. Duty cycle for clock	50%	33%
5. Flags	It has 5 flags (sign, zero, auxillary carry, parity, carry)	It has 9 flags (overflag direction, interrupt, Trap, sign, zero, auxillary carry, parity carry)
6. Pipelining support	does not support	support
7. memory segmentation supports	does not support	support
8. No of Transistors	nearly 6500	nearly 29000
9. Processor type	~~Not present~~ Accumulator based	~~Present~~ General purpose register based
10. Presence of minimum and maximum code	Not present	Present
11. No of processors	only one processor is used	more than one processor is used. Additional processor (external) can also be employed.
12. memory size	64KB	1MB
13. Instruction	No multiplication & division instruction	multiplication & division operations are present
14. Instruction queue support	does not support	support

10.

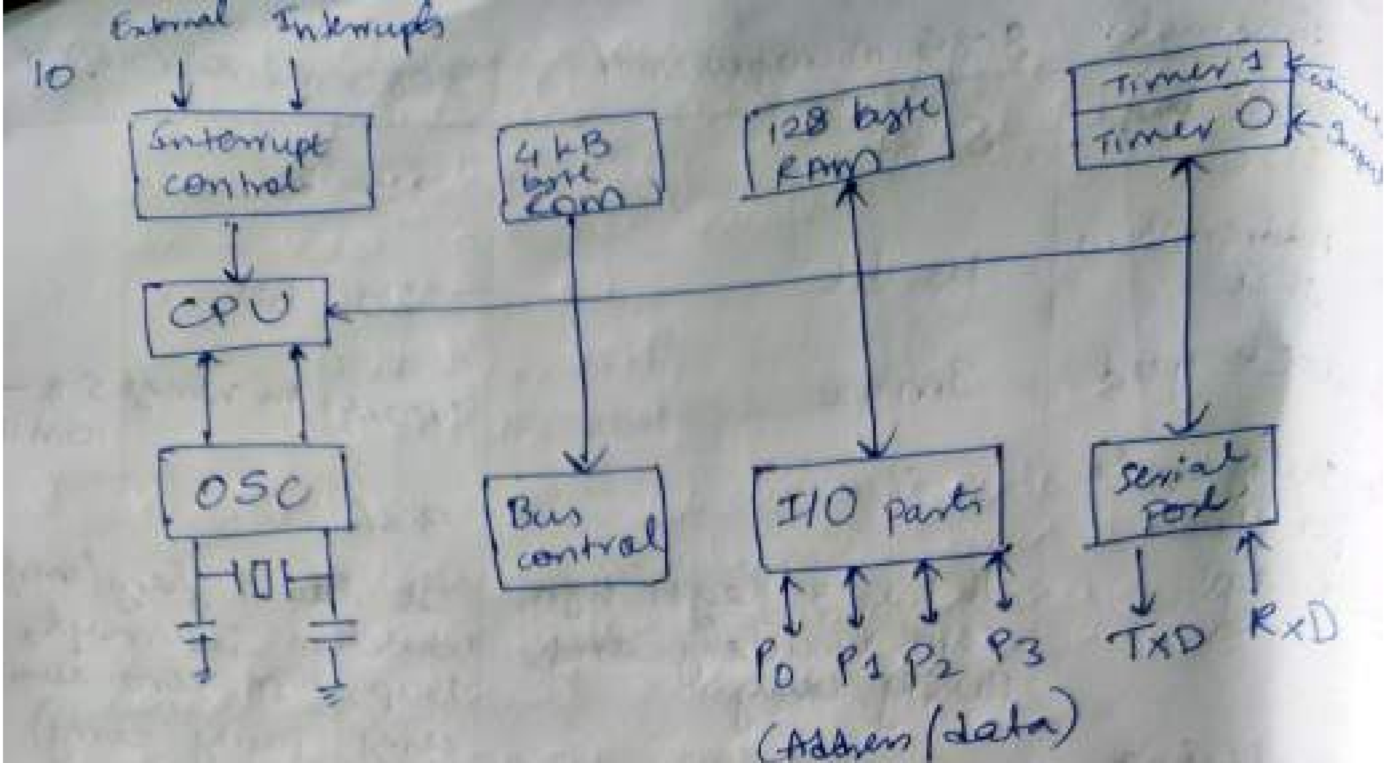


Fig: Microcontroller

Memory:

A microcontroller needs program memory to store program/instructions to perform defined tasks. This memory is termed as ROM. Furthermore µC also requires data memory to store the operands/data on a temporary basis. The memory is known as RAM. The 8051 µC is built with 4KB onchip ROM & 128 bytes RAM.

Address Bus:

Bus of µC can be defined as grp of wire which can act as a medium for the transfer of data. There are two buses present in 8051 µC.

- Data bus → used to transfer the data from one component of µC to the other
- Address bus → used to address memory location, is a 16 bit wide. Furthermore, the address bus can also be used to transfer data from the CPU to memory. Address bus is unidirectional

Interrupts

- Temporarily suspend the ongoing program
- Pass the control to a subroutine
- execute subroutine
- Resume the ongoing /main program

I/O ports

8051 needs to be connected to peripheral devices in order to control their operations. The I/O ports are responsible for connection of µC to its peripheral devices. There are 4 8-bit I/O ports present in this µC.

Other features

- 2 16 bit timers & counters
- a data pointer and a program counter of 16 bit each
- 128 user defined flags
- 4 register banks
- 31 general purpose registers which are 8-bit each.

II. Addressing mode.

Refers to the way in which the operand of an instruction is specified. Addressing mode specifies a rule for interpreting or modifying the address field of instruction before the operand is actually executed

There are 6 types of addressing modes

1. Immediate addressing mode

Data is provided in instruction itself. Data is provided immediately after opcode.

Eg:-
```
MOV A, #0AFH;
MOV R3, #4JH;
MOV DPTR, #FE00H;
```

2. Register addressing mode

The source or destination data should be present in a register (R0 to R7)

Eg:-
```
MOV A, R5;
MOV R2, #45H;
MOV R0, A;
```

3. Direct addressing mode

Source or destination address is specified by using the data in instruction. Only the internal data memory can be used in this mode

eg:- MOV A, R0;
mov R2, 45H;
mov R0, 05H;

4. Register indirect addressing mode

Source or destination address is given to register. By using register indirect addressing mode the internal or external addresses can be accessed

Eg:- MOV 05E5H, @R0,
mov @R1, 80H

5. Indexed addressing mode

Source memory can only be accessed from program memory only. Destination operand is always register A

Eg:- MOVC A, @A+DPTR;

6. Implied addressing mode

In this mode there will be a single operand. These types of instruction can work on specific registers only. These types of instructions are also known as register specific instruction

Eg:- RLA;
SWAPA;

12. In RISC, the instruction set size is small while in CISC the instruction set size is large

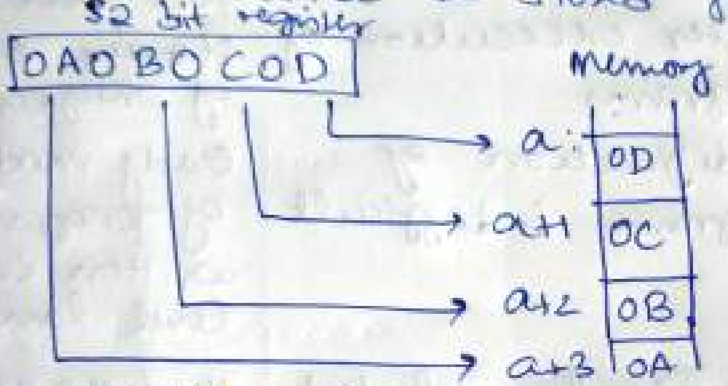
- RISC uses fixed format (32 bits) and mostly register-based instructions whereas CISC uses variable format ranges from 16-64 bits per instruct-ion
- RISC uses a single clock and limited addressing mode (ie, ~~35~~ 3-5). On the other hand, CISC uses multi-clock 12 to 24 addressing mode
- The no of general purpose registers that RISC uses range from 32-192. On contrary, CISC architecture uses 8-24 GPRs.
- Register to register memory mechanism is used in RISC with indepedent LOAD & STORE instruction. In contrast, CISC uses memory to memory mechanism for performing operations, jurthermore incorporated LOAD & STORE instructions
- RISC has split data and instruction cache design. As against, CISC uses unified cache for data and instructions, although latest designs also use split caches
- Most of the CPU control in RISC is hardwired without having a control memory. Conversely, CISC is microcode and uses control memory (ROM), but modern CISC also uses hardwired control

13. Code density is the amount of space that an executable program takes up in memory. Code density is important in devices that contain a limited amount of memory. The ARM instruction set is designed so that a program can achieve maximum performance with minimum no of instructions. most ARM9TDMI instructions are executed in single cycle.
The simpler thumb instruction set offers much

increased code density reducing code size and memory requirement.
Code can switch b/w ARM & thumb instruction sets on any procedure call.

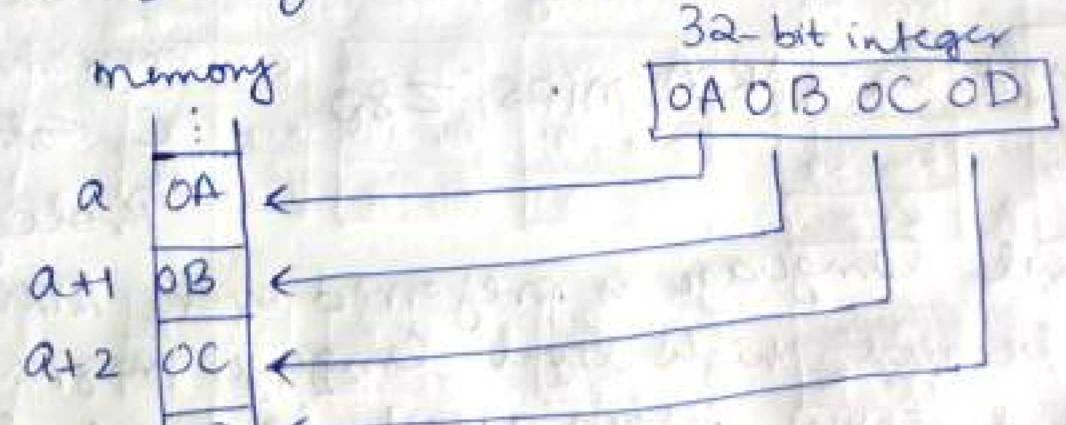
14. Endiannen refers to order of stored and reading multi-byte words in memory. Endiannen determines if the least significant byte of a word that we want in memory will go to lowest or the highest address of assigned memory space.

Little Endian -
Last bit will be stored first



Little endian

Big endian
First byte of multibyte data is stored first



Big-endian

15. Parameter	Assembly level language	High level languag
1. Abstraction	Negligible abstraction with computer language	Strong abstraction with computer language
2. Use of interpreters	It does not make use of compiler & interpreter	It uses compiler as well as interpreter to convert instruction into machine code
3. Flexibility	Difficult to use as it requires elaborate technical details at each step	Readable and machine friendly language that can be easily interpreted and executed
4 Execution	Faster execution of programs	Slower execution of programs
5. Modification	modification of program is difficult	Easy modification of programs written in high level language
6. Hardware	It is closely related to hardware and hence used to write hardware programs	It has no correspondence with hardware and used only to write software application programs
7 Example	ARM, MIPS, Z80	C, Fortan, Lisp, Prolog

High level language is preferred over Assembly level language because, high level language programmer does not need to know details about hardware like registers in processor as compared to assembly programmer, code of assembly language is difficult to understand and debug than a high level and the high-level language programs run independently of processor type.

ARM 7 supports Van Neumann architecture. It has 3-stage pipelining.

Instructions are either 32 bits long (in ARM state) or 16 bits long (in Thumb state)

Data types.

- ARM7TDMI supports byte (8bit), halfword (16 bit) and word (32 bit) datatypes. Words ~~can~~ must be aligned to 4-byte boundaries and half words to 2-byte boundaries
- ARM7TDMI has total of 37 registers - 31 general purpose 32-bit registers and 6 status registers but these can't be seen all at once.
- Processor state & operating mode dictate which register are available to programmer
- Register 13 is stack pointer, Register 15 holds PC. R14 is used as subroutine link register and receives a copy of R15 when a branch & link instruction is executed. It maybe treated as GPR at all other times.
- ARM7TDMI contains a CPSR & 5 SPSR for use of exception handlers. These registers hold info about most recently performed ALU operation, control the enabling and disabling of interrupts and set the processor operating mode

Operating modes: 7 modes of operation

- User - normal ARM prg execution state
- FIQ - designed to support a data transfer or channel process
- IRQ - used for General purpose interrupt handling
- Supervisor - protected mode for operating mode
- Abort - entered after a data or instruction prefetch abort
- System - a privileged user mode for OS

Exception priorities

- Highest priority

1. Reset 2. Data abot 3. FIQ 4. IRQ 5. Prefetch abort

Lowest priority

6. Undefined instr, software interrupt

		Fast Interrupt	Supervisory	Abort	Regular interrupt	undefined
	R0					
	R1					
	R2					
	R3					
	R4					
	R5					
	R6					
	R7					
	R8	R8_fiq				
	R9	R9_fiq				
	R10	R10-fiq				
	R11	R11-fiq				
	R12	R12_fiq				
SP	R13	R13_fiq	R13_SVC	R13_abt	R13_irq	R13_und
LR	R14	R14_fiq	R14_SVC	R14_abt	R14_irq	R14_und
PC	R15	PC R15	R15(PC)	R15(PC)	R15_(PC)	R15(PC)
	CPSR	CPSR	CPSR	CPSR	CPSR	CPSR
		SPSR	SPSR	SPSR	SPSR	SPSR
		fiq	SVC	abt	irq	und

17. V4T

- halfword & signed
- byte support
- system mode
- Thumb instr set

Architecture V5TE

- Improved Arm/thumb
- Inter networking
- CLZ
- structured architecture
- DSP multiply-ACC instr

Architecture V6

- SIMD Instr
- multiprocessing
- V6 memory arch
- Unaligned data support

Extensions

Thumb-2 (V6T2)

Trust zone (V6Z)

Multicore (V6K)

Thumbonly (V6-M

ARM11 36j(F)-1

Architecture V7

V7-A (applications)

- Advanced
- OS support

V7-R (real-time)

- hardware divide

V7-M (microcontroller

- hardware divid
- Thumb 2 only

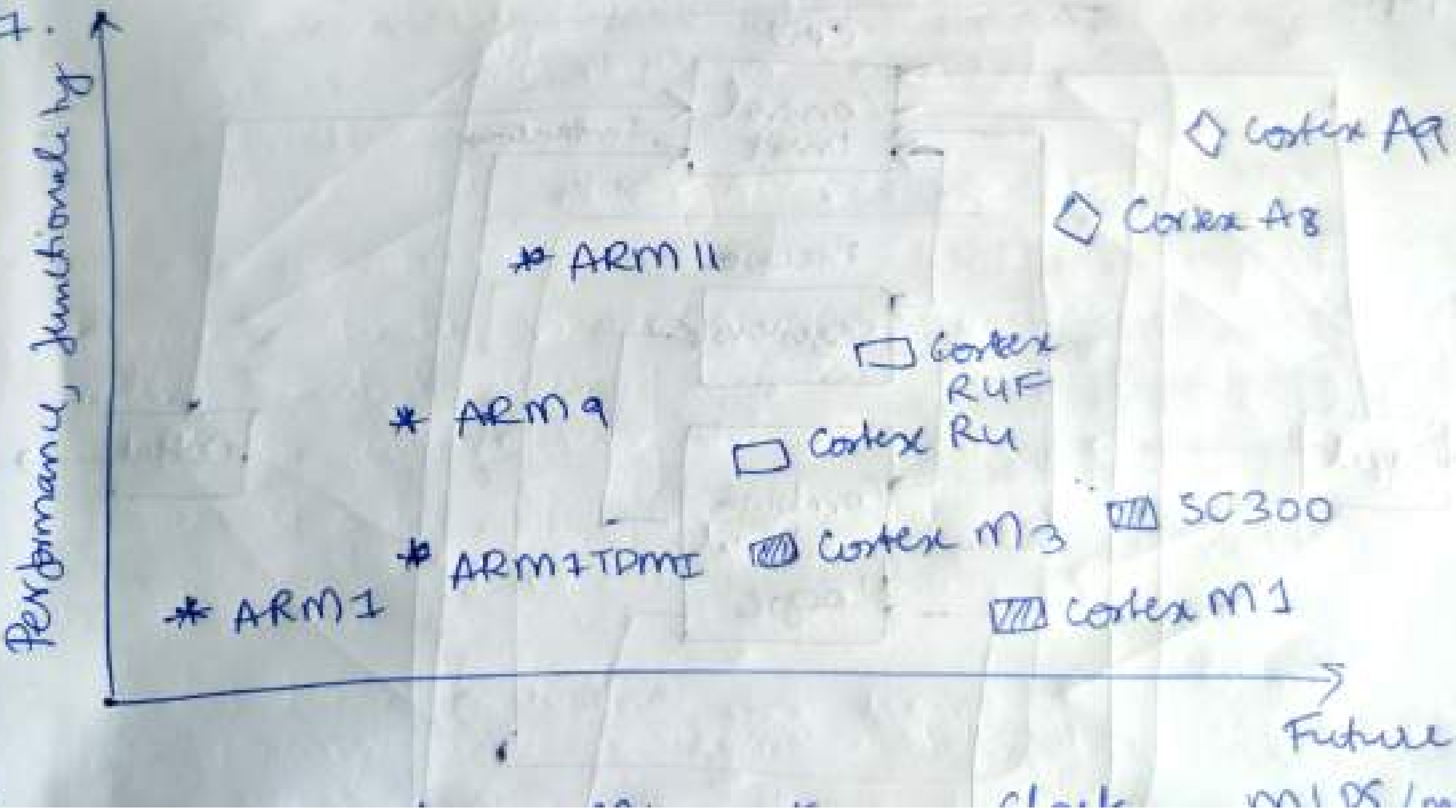
Thumb 2

NEON

Trust zone

Virtualization

17.


Performance, functionality
Future
* ARM1
* ARM7TDMI
* ARM9
* ARM11
□ Cortex R4
□ Cortex R4F
▨ Cortex M3
▨ SC300
▨ Cortex M1
◇ Cortex A8
◇ Cortex A9

18. Hardware extensions are standard components placed next to ARM. It improves performance, manage resource & extra functionality and are designed to provide flexibility in handling particular applications.

Cache and TCM:

memory management (MPU & MMU) prevents apps from in access to hardware co-processor interface.

ARM core extensions with co-processor - coprocessor can be attached to ARM processor. Extends the processing feature of a core by extending the instruction set or by providing configuration features.

More than one co-processor can add to ARM core via co-processor interface. coprocessors can access through group of dedicated ARM instr that provide Load-store type interface.

Eg:- coprocessor 15, the ARM processor uses coprocessor 15 registers to control the cache, TCMs & my management

19.

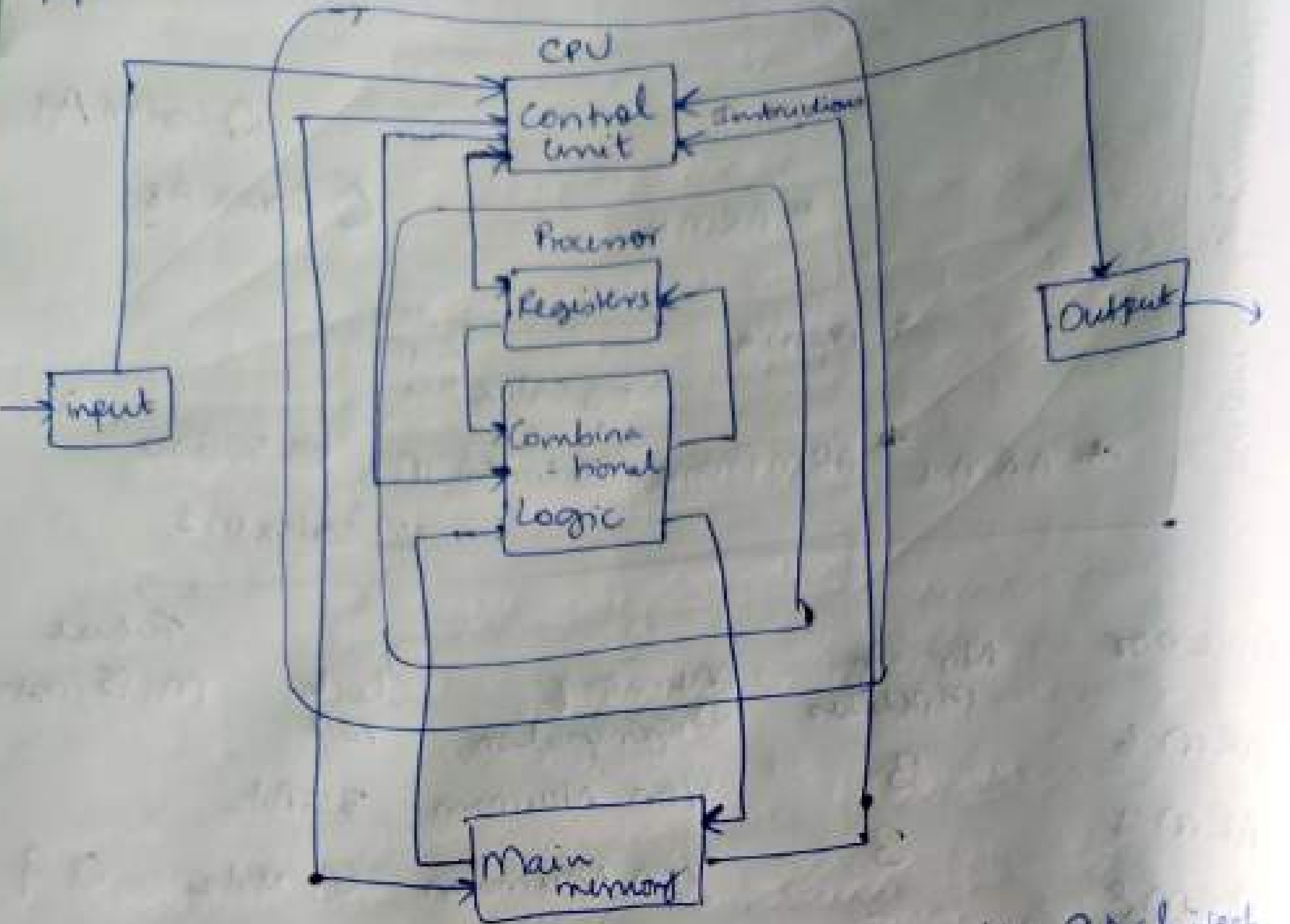


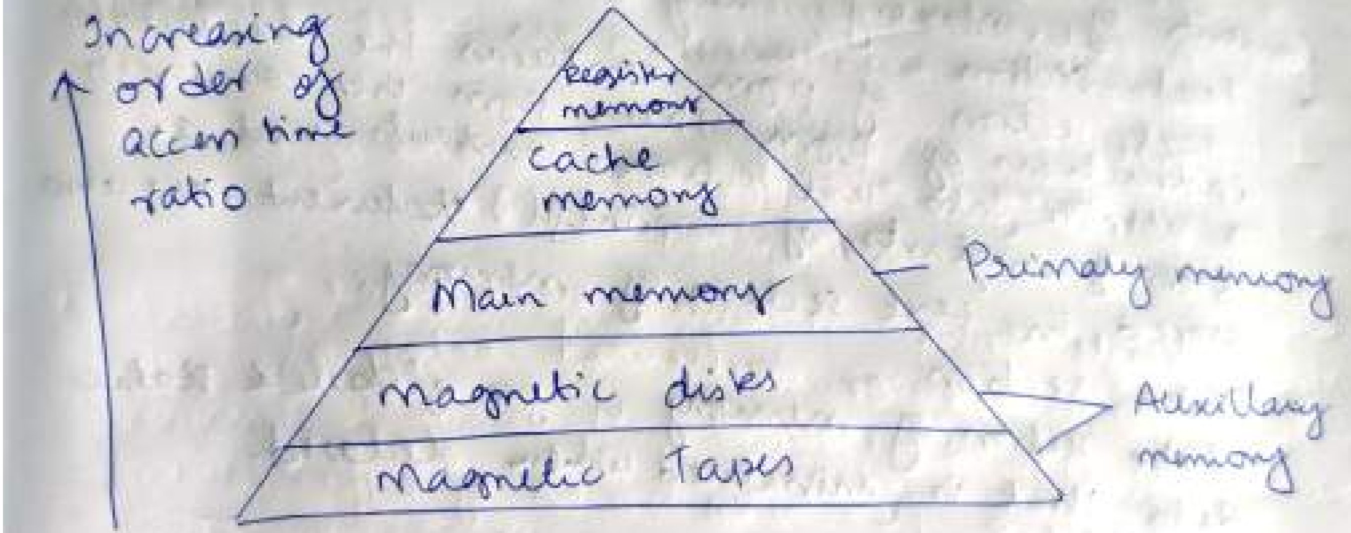
Fig: Bird eye view of computer architecture

20. Heirarchy of memory in computer

- Register:
Usually, the register is a static RAM (SRAM). processor of computer which is used for holding the data word which is typically 64 or 128 bits. The PC register is most important as well as found in all processors. Most of the processors use a status word register as well as an accumulator. A status word register is used for decision making & the accumulator is used to store data like mathematical operation.

- Cache memory
It holds chunk of data which are frequently used from main memory.

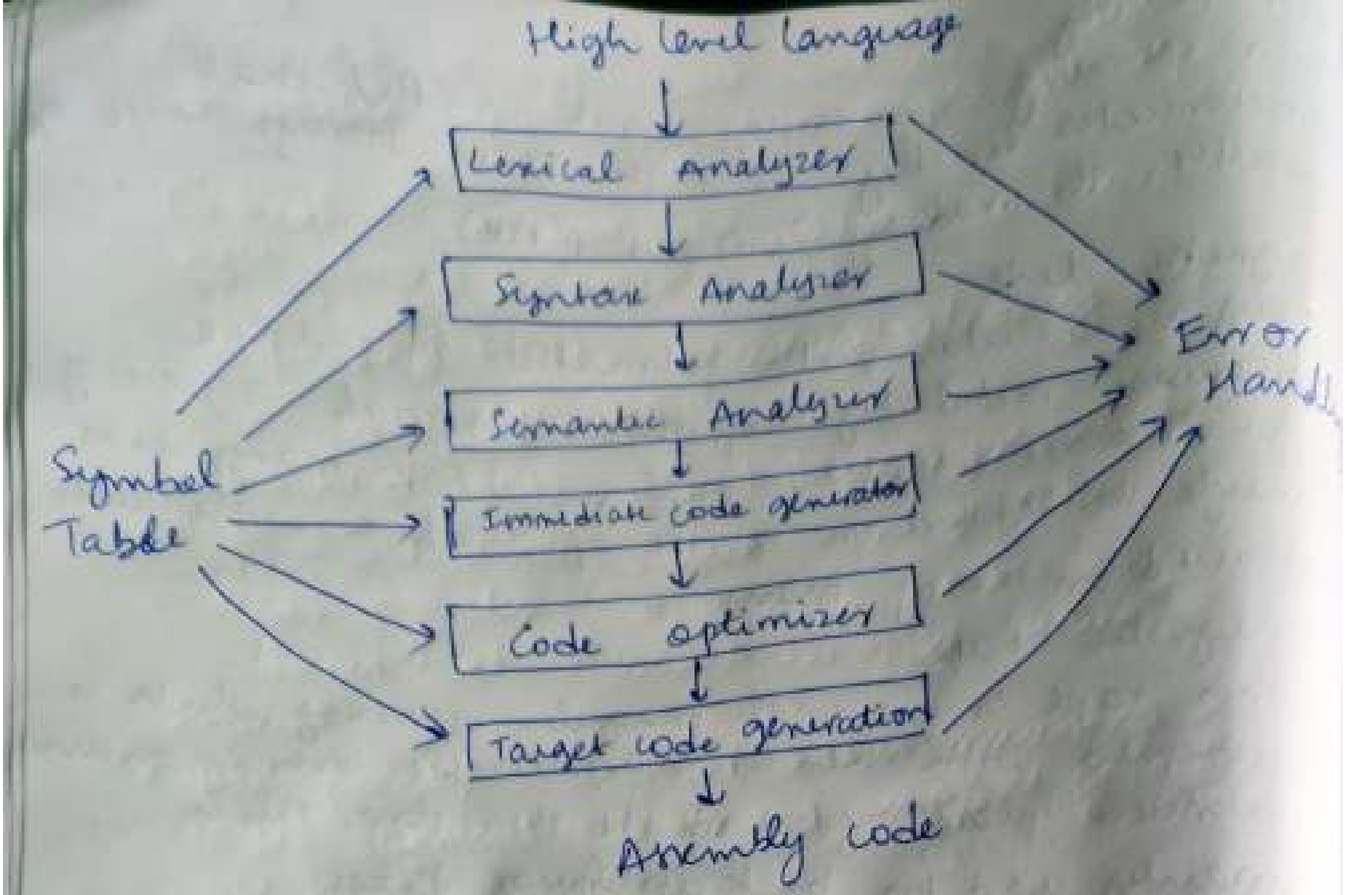
- Main memory
  Main memory is memory unit in CPU that communicates directly, it is the main storage unit of computer. Its made of ROM and RAM
- Magnetic disk
  They are circular plates fabricated of plastic otherwise metal by magnetized material. 2 faces of the disk are utilised as well as disk many disks maybe stacked on one spindle by read or write heads obtained on every plane
- Magnetic tape
  This tape is normal magnetic recording which is designed with slender magnetizable covering on an extended plastic film of the thin strip This is mainly used to back up huge data



21. 1. Lexical Analyzer
    2. Syntax Analyzer
    3. Semantic Analyzer
    4. Immediate Code generator

Lexical analyzer divides program into tokens. Syntax analyzer recognises sentences in program using syntax of language & semantic analyzer recognises senkures of each construct.

5. code optimizer 6. Code generator



Contents of memory are as follows

- Text section - contains the excecutable
  Text section of pgm. Thus, pgm istr counter in
  instructions of pgm. Thus, pgm istr counter in
  a pointer into text section. Constants, such as
  string constants of a printf() statement are also
  stored in text section of memory.

- Data section
  data section of memory is for global & static
  data that is initialized when declared

- BSS section
  gbs is for storing global and static variables
  BSS is initialized to 0 when pgm starts

- Stack section:
  The stack stores various pointer values which are
  needed for excecution of pgm and also is the default
  storage location for variables which are local to
  a fn i.e declared within body of fn.

- Registers
  In addition to computers' main memory, it is also
  possible to temporarily store a few variables directly in
  mem locations called registers

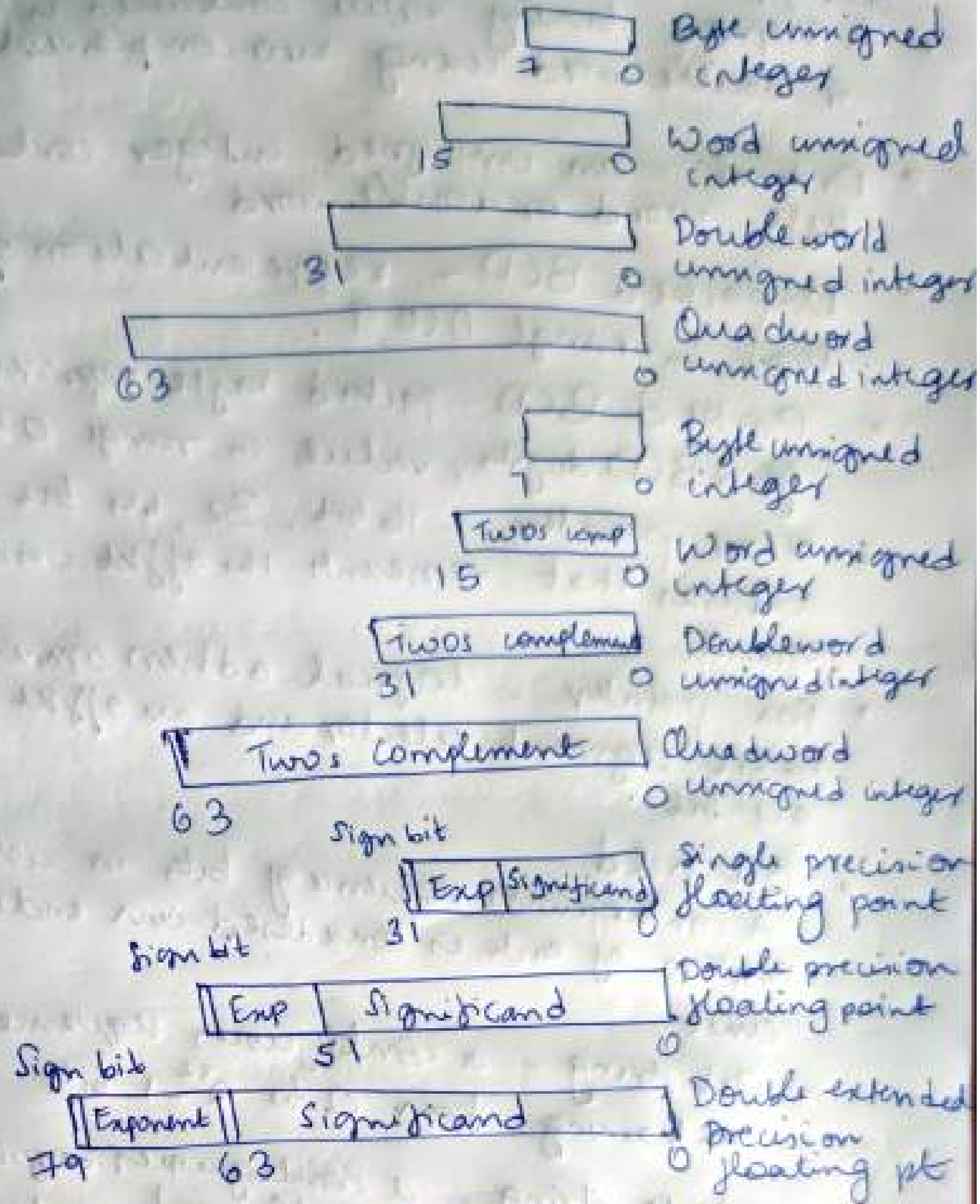


Fig: x86 Numeric data Formats

x86 Data types

- x86 can deal with data types of 8, 16, 32, 64, 128 bits in length
- are refered to as general data types
- TO allow max flexibility in data structures & efficient memory utilization, words donot need to be aligned at even numbered addreses and doublewords do not need ~~need~~ to aligned at addreses evenly divisible by 4
- x86 also supports specific data types that are recognised & operated on by particular instr.
- general - byte, 16, 32, 64, 128 bits location with arbitary binary contents

Integer

- A signed binary value contained in byte, word or double word using two complement representation
- Ordinal - an unsigned integer contained in byte, word or double word
- Unpacked BCD - representation of BCD digit in range 0 to 9.
- Packed BCD - packed byte representation of 2 BCD digits, value in range 0 to 99.
- Near pointer - 16 bit, 32, 64 bit effective address that represents the offset within a segment.
- Far pointer - logical address consisting of 16 bit segment selector and an offset of 16, 32, 64 bit.
- Bit field

  A contiguous sequence of bits in which the position of each is considered an independent unit.
- Bit string - a contiguous sequence of bits, containing from 0 to $2^{32}-1$ bits
- Xbyte string - a contiguous sequence of bytes, words or doubleword containing from 0 to $2^{32}-1$ bytes
- Floating pt - refers to a set of types that are used by floating pt - pt unit & operated on by floating pt instr