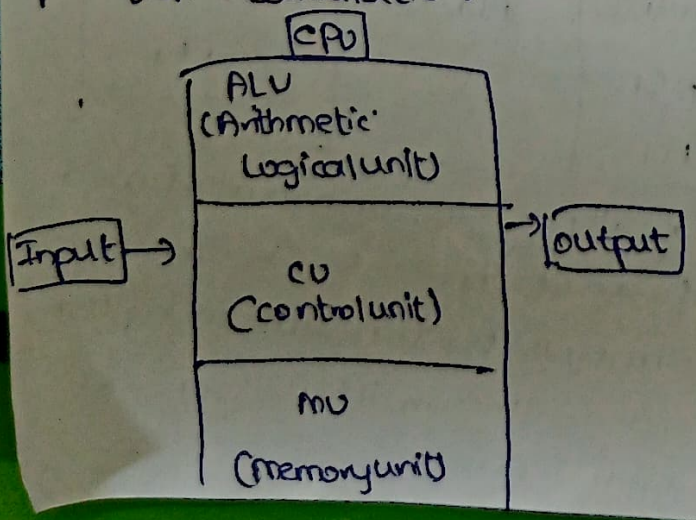
UNIT-01

① Explain about basic functional blocks of Computer.

A:) Functional Blork of Computer:

- A computer organization describes the functions and design of the various units of digital system.
- A general-purpose computer system is the best-known example of a digital system. other examples include telephone switching exchanges, digital voltmetres, digital counters, electonic calculators and digital displays.
- Computer architecture deels with the specification of instruction set and the hardware units that implement the instructions.
- Computer hardware consists of electronic circuits, displays, magnetic and optic storage media and also communication facilities.
- Functions units are a part of CPU that performs the opreations and calculations called computer program
- Functional units of a computer system are parts of CPU that perform the opreations & Calculations.



Input Unit

- Inputs units are used by the Computer to read the data. The most commonly used input devices are keyboards, mouse, joysticks, trackballs, microphones etc..
- However, the most well-known input device is a keyboard. whenever a key is pressed, the correspond letter or digit is automatically translated into its correspond binary code & transmitted over a cable to either the memory or processor.

Central processing Unit:

- CPU commonly known as CPU can be reffered as an electronic circuitry within a computer that carries out the instructions given by a computer prgm by performing the basic arithmetic, logical, control & input/output opreations specified by instructions.

Memory Unit

- The memory unit can be reffered to as the storage area in which program are kept which are running, & that contain data needed by running pgms.
- The memory unit can be categorized in two ways namely, primary memory and secondary memory
- It enables a process to access running execution applications and seriuces that are temporarily stored in specific memory location.

Arithmetic & logical Unit

- Most of-all the arithmetic & logical opreations of a computer are executed in ALU of the processor.

It performs arithmetic opreations the add, sub, mul, div and also logical opreations like AND, OR, NOT opreation

Control Unit

- The control unit is a component of a computer's central processing unit that coordinates the opreations of processor. It tells the computer's memory logic unit & input & output devices how to respond program's instruct
- The control unit is known as the nerve center of computer system.
- Example of Addition of two operands by instruction given as Add LOCA, R0.

output Unit

- The primary function of output unit is to send the processed results to the user. output device displays information in a way that user can understand.
- The most common example of an output device is a monitor

---

② Explain About Instruction Set Architecture (ISA)?

A:) The ISA is part of processor that is visible to programmer or compile writer. The ISA serves as boundary b/w software & hardware. we will briefly describe the instruction sets found in many of microprocessor used today. The ISA of processor can be described using 5 categories:

Opreand Stroage in CPU:

where are the opreands kept other than in memory?

Number of explicit named opreands

How many opreands are named in a typical instruction.

operand location:

Can any ALU instruction operand be located in memory? or must all operands be kept internally in CPU?

opreations

what opreations are provided in ISA.

Type & size of operands

what is the type & size of each operand & how is it specified?

The most common 3 types of ISAs are:

① The stack - The operand are implicity on top of stack.

② Accumulator - one operand is implicity the accumulator.

③ General Purpose Register (GPR) - All operands are explicitely mentioned, they are either registers or memory location

Let's look at the assembly code of

C = A + B;

In all 3 architectures:

Stack	Accumulator	GPR
Push A	LOAD A	Load R1, A
Push B	ADD B	ADD R1, B
ADD	STOREC	STORE R1, C
POP C	-	-

Note all processors can be neatly tagged into one of above categories. The i8086 has many instructions that use implicit opreands although it has a general register set.

The i8051 is another example, it has 4 banks of GPRs but most instructions must have the A register as one of its operands.

Stack

Advantages: Simple model of expression evalution. short instructions

DisAdvantages: A stack can't be randomly accessed this makes it hard to generate efficient code. The stack itself is accessed every operation and becomes a bottleneck.

Accumulator

Advantages: short instructions

DisAdvantages: The accumulator is only temporary storage so memory traffic is highest for this approach.

GPR

Advantages: Makes code generation easy. Data can be stored for long periods in registers.

DisAdvantages: All operands must be named leading to longer instructions.

---

(3) outline the instruction execution cycle and explain it.

A:) A program residing in the memory unit of a computer consits of a sequence of instructions. These instruction are executed by processor by going through a cycle for each instruction

(1) Fetch instruction from memory
(2) Decode the instruction
(3) Read the effective address from memory.
(4) Excute the instruction.

Instruction cycle

An instruction cycle, also known as fetch-decode-excute cycle is basic opreational process of computer. This process is repeated continuosly by CPU from boot up to shut down of computer.

(1) Fetch the instruction

The instruction is fetched from memory address that is stored in PC (Program Counter) & stored in instruction register IR. At the end of fetch opreation, PC is incremented by 1 and it then points to next instruction to be executed.

(2) Decode the instruction

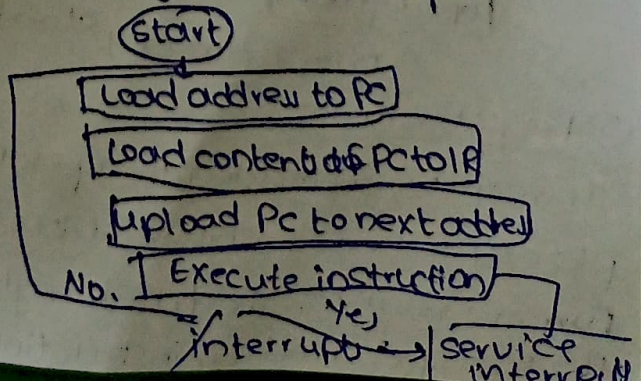
The instruction in the IR is executed by the decoder.

(3) Read the Effective address

In instruction has an indirect address, the effective address is read from the memory. otherwise opreands are directly read in case of immediate Operand instruction.

(4) Execute the instruction:

The Control unit passes the information in form of control signals to functional unit of CPU. The result generated is stored in main memory or sent to output device.



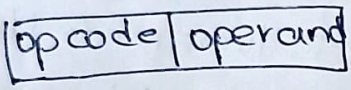
③ classify the types of Addressing modes in detail.

A:) Addressing Modes:

- The addressing mode specifies a rule of interpreting (or) modifying the address field of instruction before the operand is actually executed
- Addressing modes for i8086 instructions are divided into 2 categories

Addressing modes for data

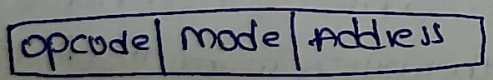
Addressing modes for branch



① Implied Addressing Mode:

In implied addressing the operand is specified in instruction itself. In this mode the data is 8 bits or 16 bits long and data is part of instruction

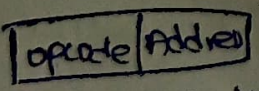
- zero address instruction are designed with implied addressing mode



Instruction format with mode field

② Immediate Addressing mode:

In this mode data is present in address field of instruction designed like one address instruction format

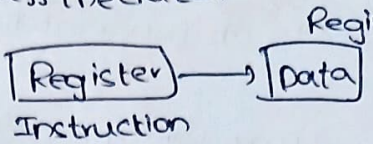


Data is directly stored here

③ Register Mode:

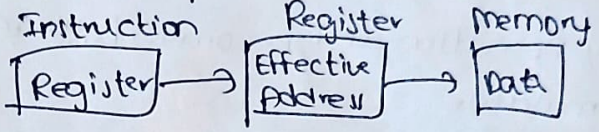
In register addressing the operand is placed in one of 8 bit or 16 bit general Purpose register. The data is register that is specified by instruction.

one register refrence is required to access the data.



④ Register Indirect Mode:

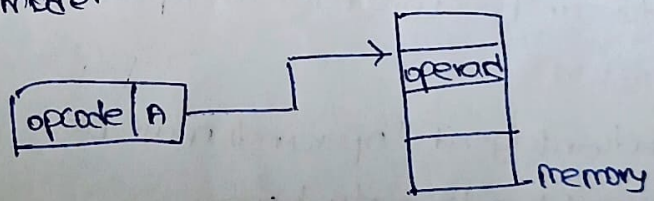
In this addressing the operand's offset is placed in any one of registers Bx, Bp, SI, DI as specified in Input.



⑤ Direct Addressing Mode:

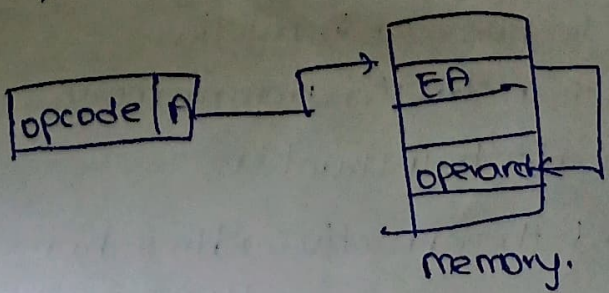
In this addressing mode,

- The address field of instruction contains the effective address of operand.
- Only one refrence to memory is required to fetch the operand
- It is also called as absolute addressing mode.



⑥ Indirect Addressing Mode:

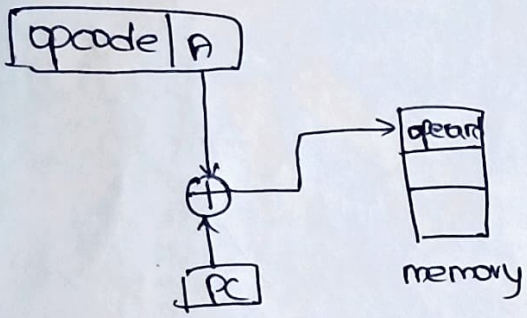
- The address field of instruction specifies the address of memory location that contains the effective address of operand.
- Two references to memory are required to fetch the operand.



⑦ Relative Addressing Mode:

- Effective address of the operand is obtained by adding the content of program counter with the address of part of instruction.

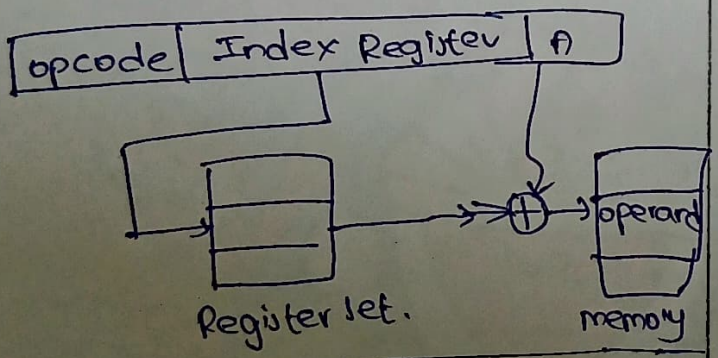
Effective Address = content of program Counter + Address part of instruction.



⑧ Indexed Addressing Mode

- Effective address of operand is obtained by adding the content of index register with the address part of instruction.

Effective Address = content of Index Register + Address part of Instruction.



Unit-2

① Construct ripple carry adder, look-ahead adder with example.

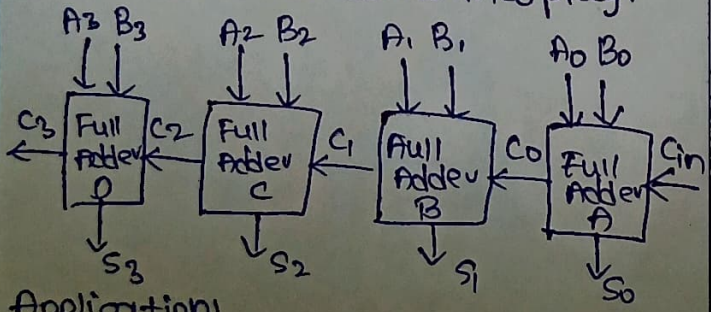
A:) Ripple Carry Adder

Is a combinational logic circuit. It is used for purpose of adding two n-bit binary numbers. It requires n full adders in its circuit for adding two n-bit binary numbers. It is also known as n-bit parallel adder.

Ripple Carry Adder:

In Ripple Carry Adder,

- Each full Adder has to wait for its carry in from its previous stage full adder.
- Thus nth full adder has to wait until all full adders have completed their opreations.
- This causes a delay and makes ripple carry adder extremely slow.
- This situation becomes worst When the value of n becomes very large.
- To overcome this disadvantage, carry Look Ahead adder comes into play.



Applications

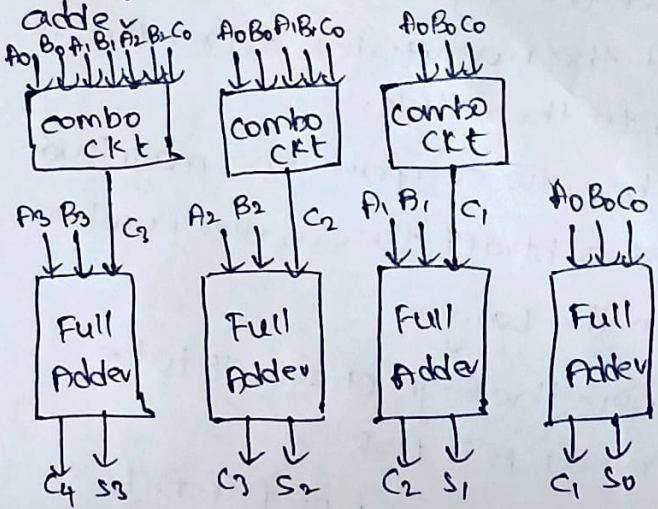
- It is used to add n-bit i/p sequences
- It is applicable in digital signal processing & microprocessor.

Carry Look Ahead Adder

- carry Look Ahead adder is an improved version of ripple carry adder
- It generates the carry-in of each full adder simultaneously without causing any delay.
- The time complexity of carry look ahead adder = O(log n)

Logic Diagram:

The logic Diagram for carry look a-head adder



working:

The working of carry look ahead adder is based on principle the carry-in of any stage full adder is independent of carry bits generated during intermediate stage. The carry of any stage full adder depends only on following two parameters.

(i) Bits being added in previous stages
(ii) Carry-in provided in beginning.

Now,

The above two parametres are always known from beginning. So the carry in of any stage full adder can be evaluated at any instant of time. Thus any full adder need not wait until its carry in generated by previous stage full adder.

② Slove integer addition with example

A:) In addition opreation the digits are added bit by bit from right to left and the carries are passed to next digit forward the left.

•subtraction operation can also be done using addition the appropriate operated is negated before adding.

Addition algorithm:

when the right of A and B are same, add the magnitude & attach the sign of A to the result.

• otherwise compare the magnitude and subtract the smaller number from the larger.

• choose the sign of result to be same as A if A>B

• The complement of A if A<B.

• If A=B subtract B from A and make the sign +ve.

operation	Add magnitude	Subtract magnitude		
		A>B	A<B	A=B
(+A)+(+B)	+(A+B)			
(-A)+(-B)	-(A+B)			
(+A)+(-B)		+(A-B)	-(B-A)	+(A-B)
(-A)+(+B)		-(A-B)	+(B-A)	+(A-B)

Example:

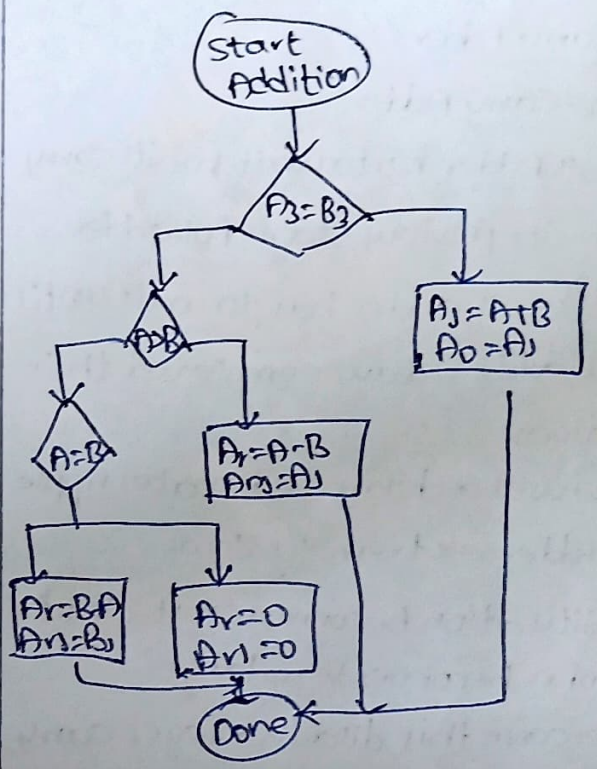
Adding two magnitudes when the result is the same sign of both operands

$$\begin{array}{r} -3 \quad 1011 \\ +2 \quad 0010 \\ \hline (-1) \quad 1001 \end{array}$$

• Adding two magnitude when the result is the different sign of both operands

$$\begin{array}{r} -3 \quad 1011 \\ +2 \quad 0010 \\ \hline (-1) \quad 1001 \end{array}$$

Flowchart



Example:

Adding two magnitudes when the result is the same sign of both operands

$$\begin{array}{r} +3 \quad 00\ 11 \\ +2 \quad 00\ 10 \\ \hline +5 \quad 01\ 01 \end{array}$$

1+1=10

0+0=0

③ Apply the integer subtraction with example:

A:-) A-B

A - minuent B - subtratand

- when the sign of A & B are differe add the magnitude and attach the sign of A of the result.
- otherwise compare the magnitude and subtract the smaller number from the larger.
- choose the sign of result to be sign of A if A>B.
- OR complement of A if A<B
- If A=B subtract B from A and make the result +ve.

Signed Bit operation

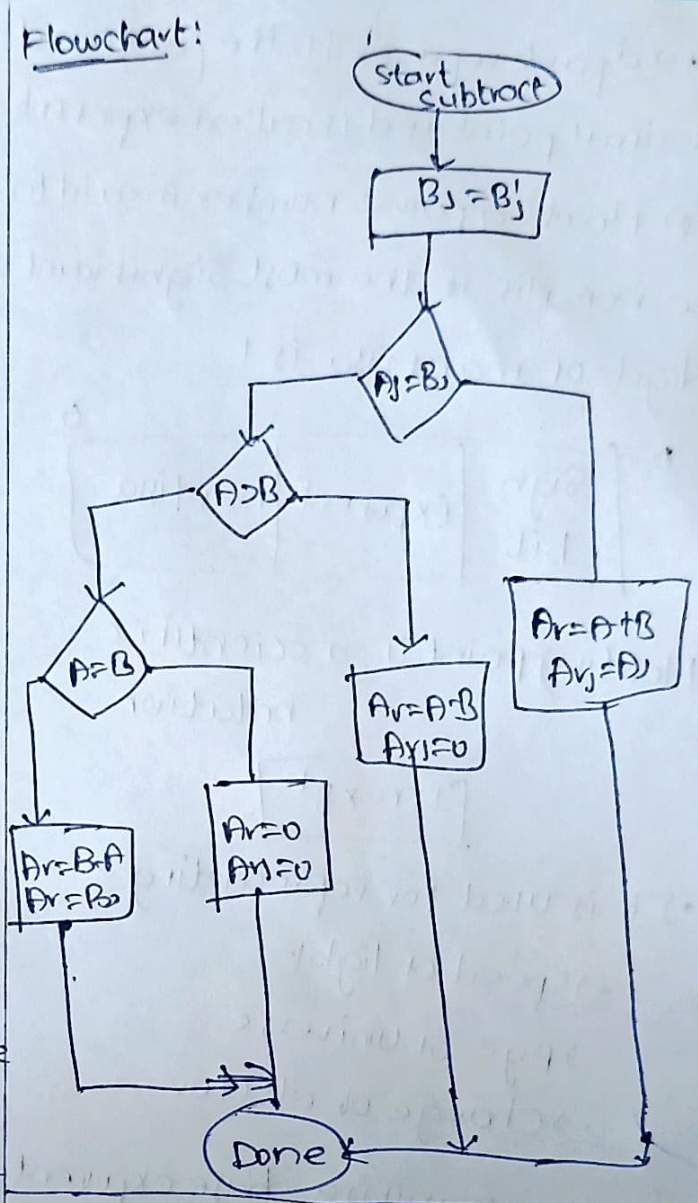
Opreation	Add magnitude	Subtract magnitude		
		A>B	A<B	A=B
(+A)-(-B)	+(A+B)			
(-A)-(+B)	-(A+B)			
(+A)-(+B)		+(A-B)	-(B-A)	+(A-B)
(-A)-(-B)		-(A-B)	+(B-A)	+(A-B)

Example:

```
  +3   0011
- +2   0010
-------------
  +1   0001
-------------

  -3   0011
- +2   0010
-------------
  -5   0101
-------------
```

Flowchart:



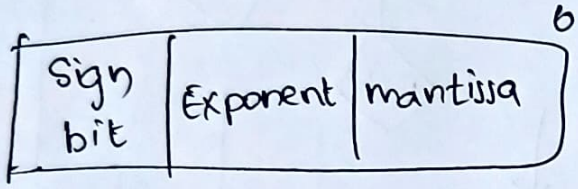
④ Floating point arithmetic

- This representation does not reserve any specific number of bits for integer part or the fractional part
- It reserves certain number of bits for the number called mantissa and certain number of bits with in that for exponent.
- The floating point representation has 2 parts:

→ 1st part is signed fixed point number called the mantissa.

⇒ 2nd part represents the position of decimal point and is called exponent

• A floating point number is said to be normalis if the most significant digit of manfijsa is 1



• Floating point is a scientific notation

$$Im \times B^E$$

• It is used for representing
  → speed of light
  → Age of universe
  → Charge of elector

Number	manifija	Baye	exponent
9×10^8	9	10	8
110×2^7	110	2	7
4364.784	4364784	10	-3

Unit-3

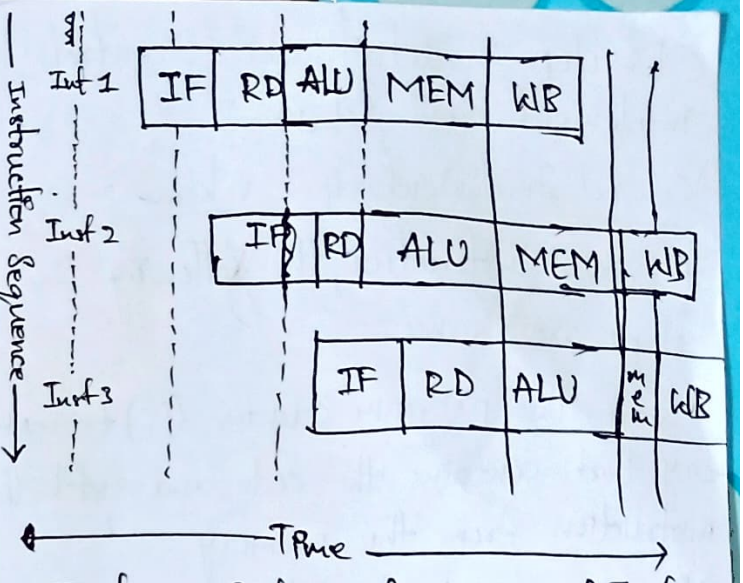
① Explain in detail in pipeline and its execution.

A) Pipelining: Pipelining is an implementation technique in which multiple instruction are executed simultaneously by overlapping them in execution to save time and resource. The previous instruction will be in the execution phase when the current instruction is fetched from the memory.

Need for Pipelining: Without a pipeline, a computer processor fetches the first instruction from memory, performs the operation mentioned in it, and then goes to fetch the next instruction from memory. While fetching the instruction, the arithmetic unit of the processor is idle. It must wait until it is loaded with next instruction.

Stages in MIPS pipelining:

The following are the various stages in pipelining: ① Instruction fetch (IF): Fetch instruction from memory ② Instruction Decode (RD): Read registers while decoding the instruction. The format of MIPS instructions allows reading and decoding to occur simultaneously ③ Execute: Execute the operation (or) calculate an address. This involves ALU operations. ④ Memory access (MEM): Access an operand in data memory. ⑤ Write Back (WB): Write the result into a register.



5 stage pipelining of MIPS architecture.

The pipelining speed can be manipulated using the expression:

$$\text{Time b/w instructions}_{\text{pipelined}} = \frac{\text{Time b/w inst.}_{\text{nonpipelined}}}{\text{No. of pipe stages}}$$

Pipelining improves performance by increasing instruction throughout. It is not decreasing the execution time of an individual instruction, but increases the no. of instructions that complete its execution for a give time period.

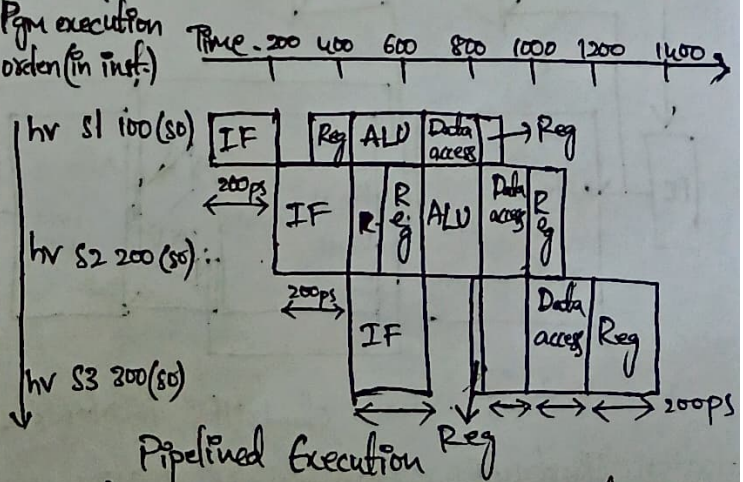


Fig. shows the comparison of execution of instructions with and without pipelining on same hardware components.

② Develop in detail about control implementation scheme?

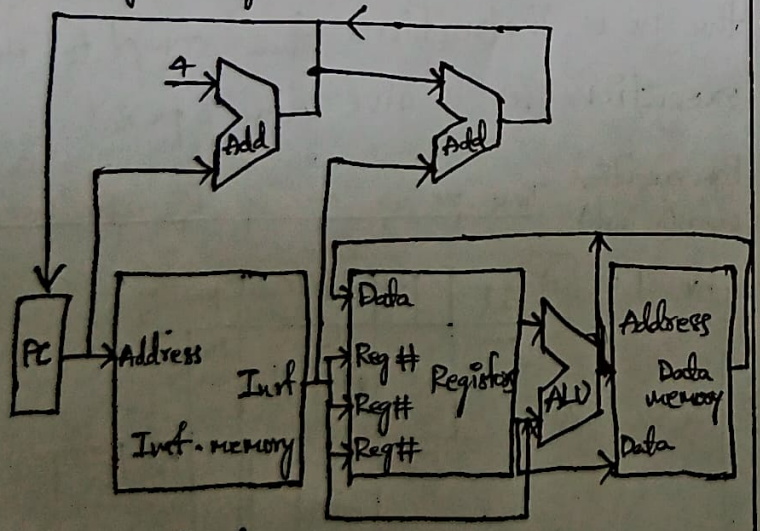
A) Control Implementation Scheme:

For any instruction, the following 2 steps are same:

⇒ Send the program counter (PC) to memory that contains the code and fetch the instruction from the memory.

⇒ Read one (or) two registers, using fields of the instruction to select the registers to read.

Load instruction needs to read only one register, but most other instructions require reading two registers. The remaining actions required to complete the instruction depend on the inst. class. This is due to the simplicity and regularity of MIPS instruction set.



An Abstract view of MIPS implementation Instruction Formats of MIPS.

Field	0	rs	rt	rd	shamt	funct
Bit Positions	31:26	25:21	20:16	15:11	10:6	5:0

R-Format Instruction.

Instruction format for R-format instruction have an op code of 0. These inst. have 3 register operands: sources: rs, rt, destination: rd. This inst. type is used to implement are add, sub, and, or, slt. The shamt field is for shifting operation.

Field	35(or)43	rs	rt	address
Bit Positions	31:26	25:21	20:16	15:0

Load (or) Store inst.

Instruction format for load specified by op code = 35 ten and store is specified by op code 43 ten instructions. The register rs is the base register that is added to the 16-bit address field to form the memory address.

Field	4	rs	rt	address
Bit Positions	31:26	25:21	20:16	15:0

Branch Instructions.

Inst. format for branch equal (op code = 4). The registers rs and rt are the source registers that are compared for equality. The 16-bit address field is sign extended, shifted, and added to the PC to computer the branch target address.

⇒ All inst. classes, except jump, use the arithmetic-logical unit (ALU) after reading the registers.

⇒ The memory reference instructions use the ALU for an address calculation, the arithmetic-logical inst. for the operation execution, and branches for comparison.

⇒ After using the ALU, the actions required to complete various inst. classes differ.

⇒ A memory reference inst. will be need to access the memory either to read data for a load (or) write data for a store. [redacted]

③ Organize the MIPS Architecture ?.

A) MIPS (Million Instructions Per Second) is a simple, streamlined, highly scalable RISC architecture with adopted by the industries.

⇒ The features that makes its widely useable are:

① Simple load & store with large no. of register
② The number & the character of the inst.
③ Better pipelining efficiency with visible pipeline delay slots.
④ Efficiency with compilers.

Implementation of MIPS.

MIPS has 32 General Purpose Registers (GPR) (or) Integer registers (64 bit) holding integer data. Floating point registers (FPR) are also available in MIPS capable of holding both single precision (32 bit) and double precision data (64 bit). The following are the data types available for MIPS:

Size	Name	Registers.
8 Bits	Byte	Integer register.
Bits	Half word	"
Bits	word	Floating point register
Bits	Double word	"

With these resources the MIPS performs the following operations:

⇒ Memory referencing: load word (lw) and store word (sw)
⇒ Arithmetic-logical inst: add, sub, and, or, slt
⇒ Branch inst: equal (beq) and jump (j)

① Set the program counter (PC) to the address of the code and fetch the inst. from that memory.

② Read one (or) two registers, using fields of instruction to select the registers to read

Program Counter (PC): This register contains the address of inst. currently getting executed. The PC is incremented to read the next inst. to be executed.

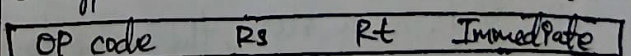
⇒ The operand in the inst. are fetched from the registers. In case of branch inst., the result of the branch operation is used to determine the next inst. to be executed. The multiplexer (MUX1), selects one input control line from multiple inputs. This acts as a data selector. This helps to control several units depending on type of inst.

MIPS instruction format.

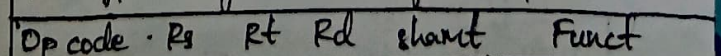
There are only 3 inst. formats in MIPS. The inst. belongs to any one of following type:

⇒ Arithmetic / logical / shift / comparison.
⇒ Control inst. (branch and jump)
⇒ Load / store.
⇒ Other (exception register movement)

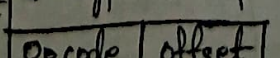
I-type: load and store instructions.



R-type: Register to Register operations.



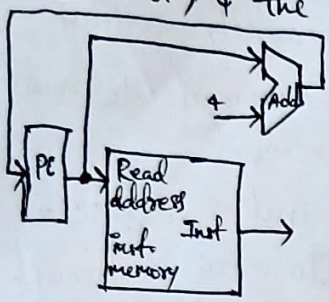
J-type: Jump instructions.



⇒ The multiplexor is controlled by the AND gate that operates on the zero output of the ALU and a control signal that indicates that the instruction is a branch.

④ Explain in detail Instruction Fetch?

A) Instruction Fetch: The fundamental operation in Inst. Fetch is to send the address in the PC to the inst. memory & obtain the specified inst, & the increment the PC.



Instruction Fetch

R type inst:

⇒ They all read two registers, perform an ALU operation on the contents of the registers and write the result.

⇒ This inst. class includes add, sub, and, or, and slt.

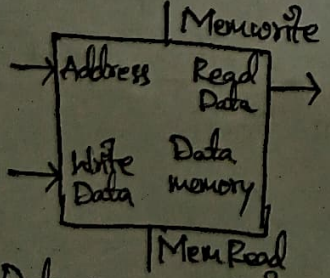
⇒ The processor have general-purpose registers are stored in a structure called a register file.

⇒ The R-format always performs ALU operation that has three register operands..

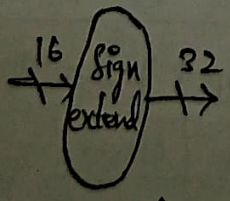
Load & Store Instructions.

⇒ The load & store inst. compute a memory address by adding the base register.

⇒ If the inst. is a store, the value to be stored must also be read from register



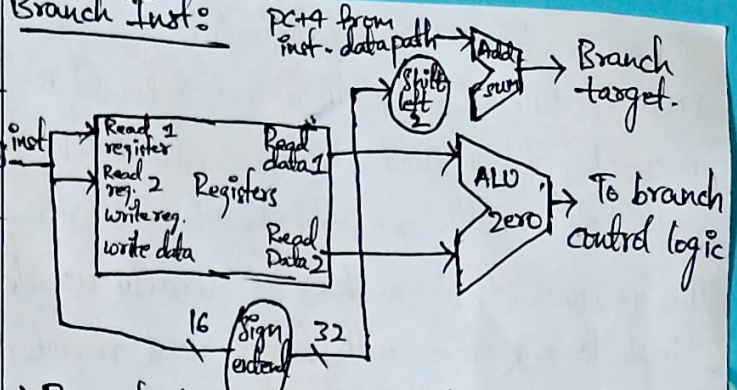
Data memory Unit



Sign-extension Unit

⇒ The processor has a sign extension unit to sign extend the 16-bit offset field in the inst. to a 32-bit signed value.

Branch Inst:



⇒ Branch target is the address specified in a branch, which is used to update the PC if the branch is taken. In the MIPS architecture the branch target is computed as sum of the offset field of inst. and address of the inst. following the branch.

⇒ J. The beq inst. has 3 operands, two registers that are compared for equality, and a 16-bit offset to compute the branch target address. beq t1, t2, offset.

⇒ K. Thus, the branch data path must do two operations: compute the branch target address and compare the register contents.

⇒ L. Branch taken is where the branch condition is satisfied and the program counter (PC) loads the branch target.

⇒ M. Branch not taken is where the branch condition is false and the program counter (PC) loads the address of the inst.

⇒ N. The branch target is calculated by taking the address of the net inst. after the branch instruction.

⇒ O. The offset field is shifted left 2 bits to increase the effective range of the offset field by a factor of four.

⇒JJ. The unit labelled shift left adds 2 zeros to the low-order end of sign extended offset field.

*KK. The control logic decides whether the incremented PC (or) branch target should replace the PC.

⇒LL. The jump inst. operates by replacing the lower 28 bits of the PC with the lower 26 bits of the inst. shifted left by 2 bits.

⇒MM. Delayed branch is where the inst. immediately following the branch is always executed.

NN. MIPS architecture implements delayed branch the inst. immediately following the branch is always executed.

JJJ. When the condition is false, the execution looks like a normal branch.

*KKK. When the condition is true, a delayed branch first executes the instruction immediately following branch.

⇒KKK. Delayed branches facilitate pipelining.

⇒To implement branch instructions the data path must include an adder circuitry to complete branch target.

⇒To control Unit for this data path must take inputs and generate a write signal for each state element.

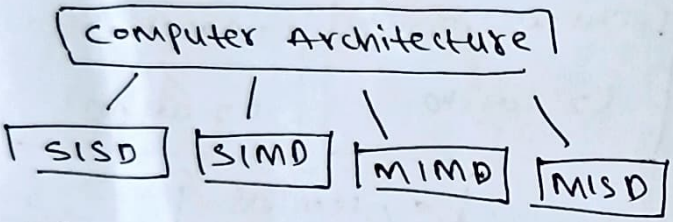
⇒The operations of arithmetic logical (or R-type) instructions and the memory instructions data path are almost similar.

⇒

UNIT-4

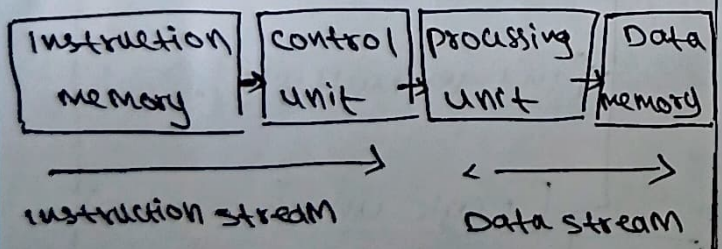
flynn's taxonomy

- single instruction single Data (SISD)
  - Traditional sequential computing systems.
- single instruction multiple Data (SIMD)
- multiple instructions multiple Data (MIMD)
- multiple instructions single Data (MISD)



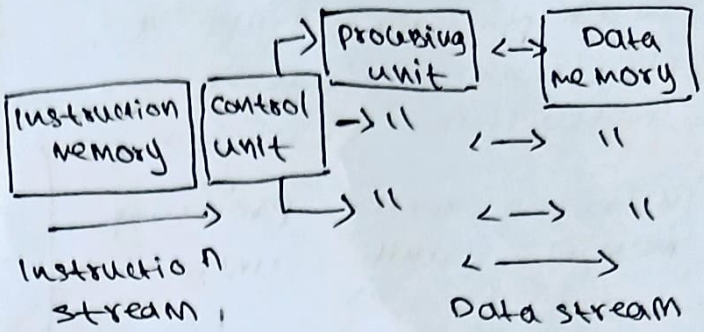
(i) SISD

- At one time, one instruction operates on one data.
- Traditional sequence operator
- single data stream is being processed by one instruction stream



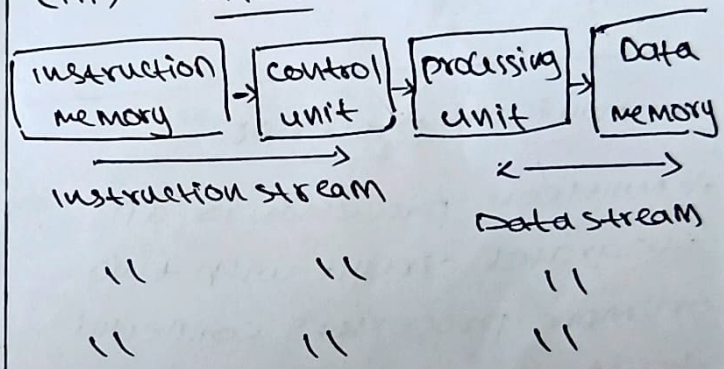
(ii) SIMD

- At one time, one instruction operates on many data.
  - Data parallel architecture
- Vector architecture has similar characteristics, but achieve the parallelism with pipeling.
- Array processors.



- SIMD (single-instruction stream, Multiple-Data streams)
- SIMD works best when dealing with array in for loops

(iii) MIMD

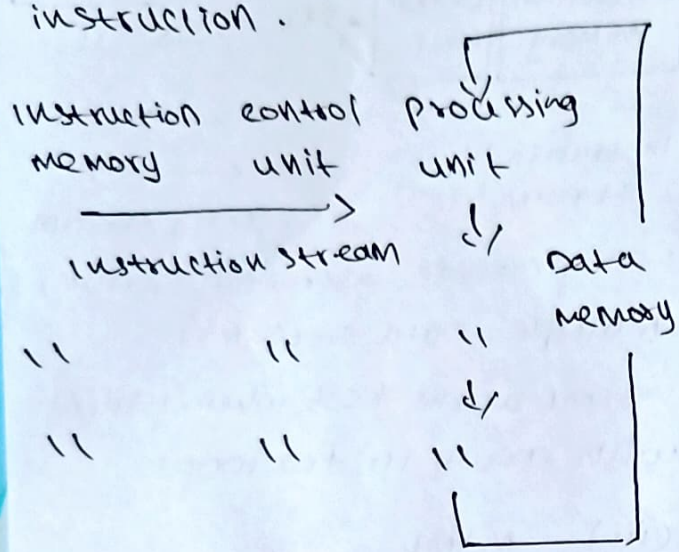


- MIMD (multiple-instruction streams, multiple-Data streams)
- most common and general parallel machine.
- each processor has a separate program.
- each instruction operates different data.

(iv) MISD

- systolic array is one example of a MISD architecture.
- systolic array is homogeneous network of tightly coupled data processing units (DPUs) called nodes

- Each processor executes a different sequence of instruction.

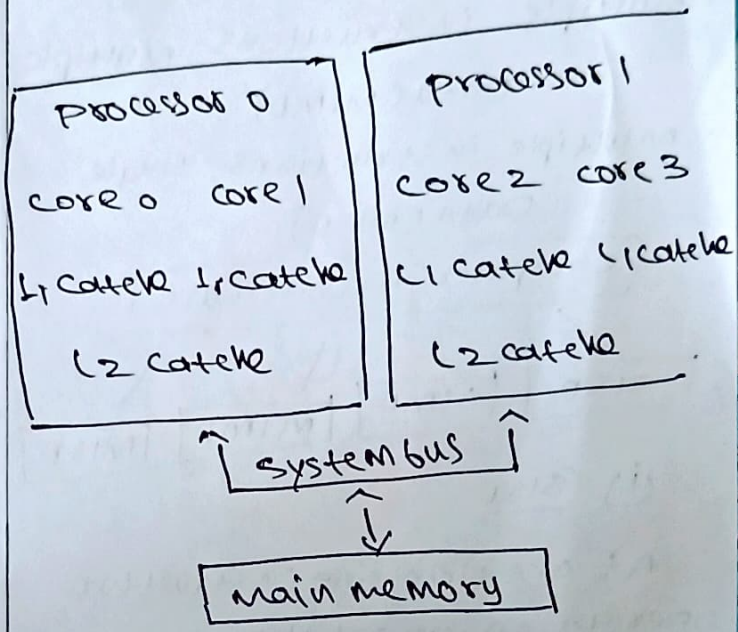


Multicore processor

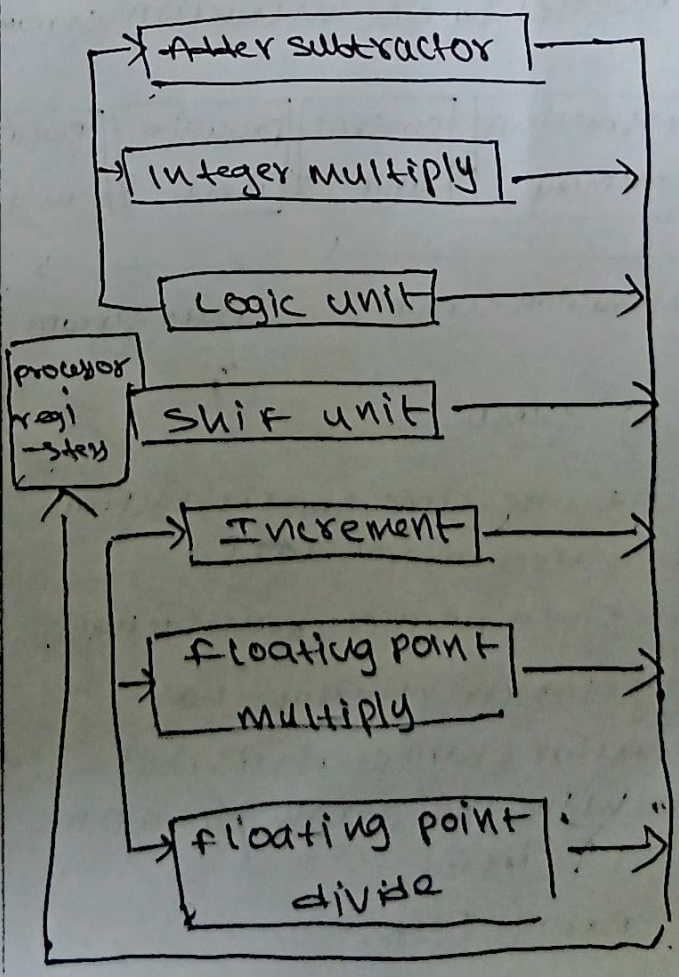
A multicore processor is an integrated circuit with two or more processors connected to it for faster simultaneous processing of several tasks, reduced power consumption and for greater performance.

Architecture of multicore processor

- A multicore processor design enables the communi cation b/w all available core and they divide and assign all processing duties approximately.
- The processed data from each core is transmitted back to the computer's main broad via a single common gateway over all of the processing operations have been finished.
- The methods beats a single core cpu in terms of total performance.



construct parallel processor



The following diagram shows one possible way of separating the execution unit into eight functional units operating in set.

- The adder and integer multiplier performs the arithmetic operation with integer number.
- The floating point operations are separated into three circuits operating.
- The logic, shift and increment operations can be performed concurrently on different data.
- All units are independent of each other.
- So one number can be shifted while another is being incremented.

Instruction level parallelism

- ILP is used to refers to the architecture in which multiple operations can be performed in a particular process with its own set of resources (address space, registers, identifiers).
- ILP processors have the same execution blw a RISC processors.
- The machines without ILP have compile blw which is hard to implement.

- A typical ILP allows multiple cycle operations to be pipelined.

Example

Let the sequence of instructions be

1. $y_1 = x_1 + 1000$ 2. $y_2 = x_3 + 100$
3. $z_1 = y_1 + 0010$ 4. $z_2 = y_2 + 0101$
5. $t_1 = t_1 + 1$ 6. $p = q * 1000$
7. $clr = clr + 0010$ 8. $r = r + 0001$

cycle operation
1 $y_1 = x_1 + 1000$

Cycle	Int ALU	Int ALU	Float ALU	float ALU
1	$t_1 = n+1$	clr = clr+1000	$y_1 =$ x+1010	$y_2 =$ $x_2 +$ 1100
2.	$r = r + 0001$			
3.	nop $z = y_1 + 0010$	$x_2 = y_2 + 0101$		

fig: ILP record of execution.

No operator in the fig are used to show idle time of processor since latency of floating point operators is 3. hence multiplication take 3 cycles and processor has to remain idle for that time period. But in IPL execution the processor can utilize those nop's to execute other operations while previous are still being executed.

Unit-5

① Utilize virtual memory and how it used efficiently in computer system.

A:) Virtual Memory:

Virtual memory is a storage scheme that provides user an illusion of having a very big main memory. This is done by treating a part of secondary memory as the main memory.

Instead of loading one big process in the main memory the os loads the different parts of more than one processors in the main memory.

By doing this the degree of multiprogramming will be increased in the CPU utilization will also be increased.

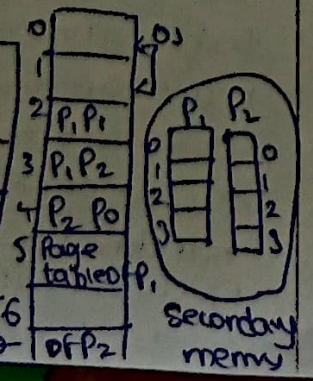
How virtual memory works?

whenever some pages needs to be loaded in the main memory for execution & the memory is not available for those many pages, then in that case instead of stopping the pages from entering in the main memory..

Os searches for the RAM area that are not referenced and copy into the secondary mly to make the space for new page in main memory.

Let us assume 2 processos, P_1 and P_2 Contains 4 pages each. each page size is 1KB. The main memory contains 8 frame of 1KB each. The OS resides in first two partitions

Frame	Present/ Absent	DBit	Reference Bit	Protection
	0	0	0	
2	1	0	1	
3	1	1	0	
	0	0	0	



Frame	Present/ Absent	DBit	Reference Bit	Protection
4	1	0	1	
	0	0	0	
	0	0	0	
	0	0	0	

CPU

5 7

Page table base(P_1) Page table base(P_2)

The page tables of both the pages are 1KB size each and therefore they can be fit in one frame each. The page tables of both the processes contain various information that are shown in above fig.

The CPU contains a register which contains the base address of page table that is 5 in case of P_1 and 7 in case of P_2. This page table base addresses will be added to page number of the logical address when it comes to accessing the actual corresponding entry.

Advantages:

- The degree of multiprogramming will be increased.
- User can run large application with less real RAM
- There is no need to by more memory RAMs.

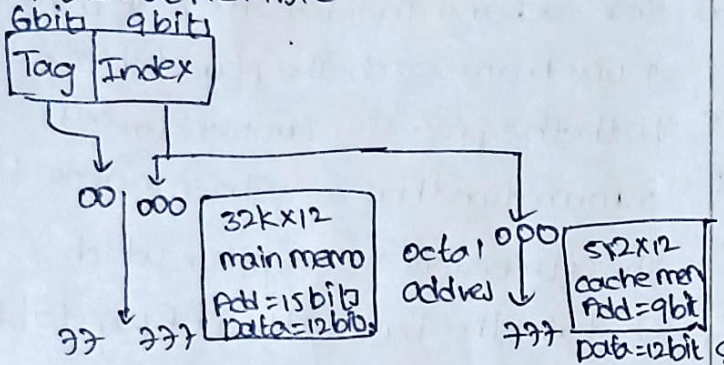
Disadvantages:

- The system becomes slower since swapping takes time.
- It takes more time in switching between applications
- The user will have the lesser hard disk space for its use.

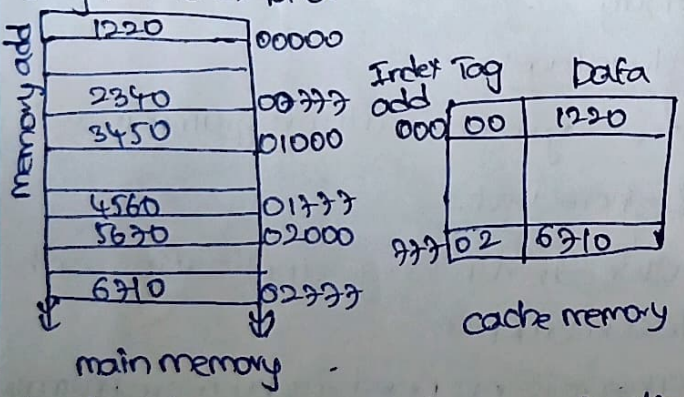
② Dissect about direct and set associative map techique in cache.

A:) Direct mapping:

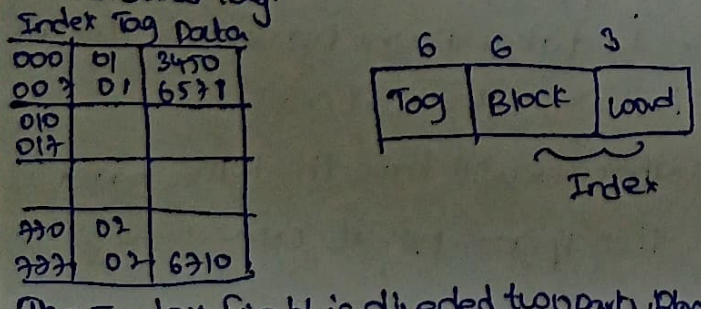
A particular block of main memory can be brought a particular block of cache memory so, it is not flexible.



The CPU address of 15 bits is divided two field, The nine least significant bits constitute the index field & remaining six bits from tag field. The main memory needs an address that includes both the tag & index bits



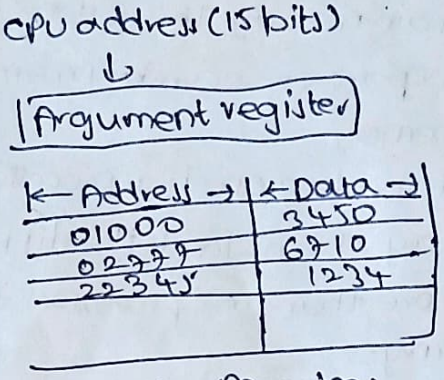
The direct mapping cache organization uses the n-bit address to access the main memory & the k-bit to access the cache. Each word in cache consists of data word and associated tag.



The Index field is divided two parts. Block field & the word field. In 512 word cache 64 blocks of 8 words. Since 64x8 = 512

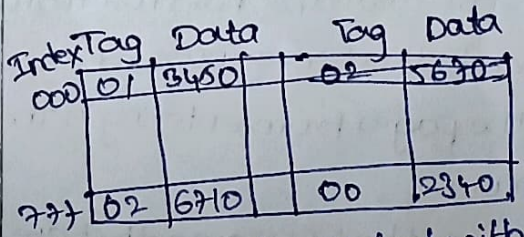
Associative mapping:

In mapping function, any block of main memory can potentially reside in any cache block position. This is much more flexible mapping method.



Set Associative Mapping:

In this method, blocks of cache are grouped into sets, and mapping allows a block of main memory to reside in any block of specific set. From flexibility point of view, it is in btw to other two methods.



The octal numbers listed with refrence to main memory contents. When the CPU generates a memory request, the index value of address is used to access the cache. The comparison logic dine by associative search of tags in set similar to a associt memory search thus name "set Associative".

Replacement politics

- When the cache is full & there is necessity to bring new data to cache, then a decision must be made as to which data from cache is to be removed.

In case of set associative memory:

- Random replacement
- First-in First out (FIFO)
- Least Recently Used (LRU)

③ Construct and explain Central Storage Unit.

A:) Main memory is the central storage unit in a computer system. It relatively large and fast memory used to store programs and data during the computer opreation. The principal technology used for main memory is based on semi conductor integrated circuits. Integrate circuit RAM chips are available in two possible opreating modes, static & dynamic.

- static RAM - consits of internal flip flops that store the binary information.
- Dynamic RAM - stores the binary informatio in form of electric charges that are applied to capacitors.

most of the main memory in general purpose computer is made up of RAM integrated circuit chips, but a portion of memory may be constructed with Rom chips.

- Read only memory - Store programs that are permarently resident in computer and for tables of constant that do not change in value once the production of computer is completed.
- Boot strap loader - function is start the computer software opreating when power is turned on.
- Boot strap program loads a portion of opreating system from disc to main memory & control is then transferred to opreating system.

Memory Address Map

The interconnection btw memory and processor is then established from knowledge of size of memory needed and type of RAM & Rom chips available. The addressing of memory can be establish by means of table that specify the memory address assigned to each chip.

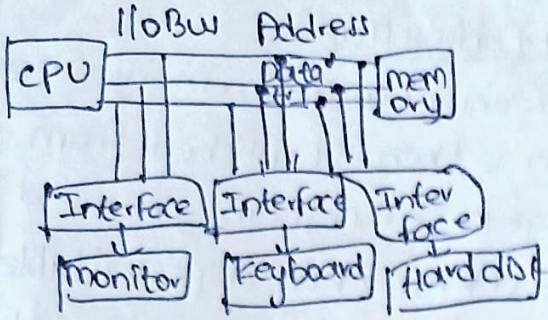
Memory connection to CPU:

RAM and Rom chips are connected to CPU through the data & address buses. The low order lines in address bus select the byte within the chips and other lines in address bus select a particular chip through its select inputs. Each RAM receives the seven low-order bits of address bus to select one of 128 possible bytes. The particular RAM chip selected is determined from lines 8 and 9 in the address bus. This is done through a 2x4 decoder whose outputs goto to CS1 input in each RAM chip.

---

④ Develop how I/o devices can be interfaced with a block diagram.

A:) I/o Interface is used as an method which helps in transfeering of informati btw the internal Storage devices memory and the external peripheal device. It is also called Input-output devices. Just like the external hard drives there is also availability of some peripheral devices which able to provide both input and output.



Functions of I/O Interface

- It is used to synchronize the operating speed of CPU with respect to input-output
- It selects the input-output device which is appropriate for interpetation of the input-output device.
- It is capable of providing signals like control and timing signals.
- In this data buffering can be possible through data bus.
- There are various error detectors.

SCSI (Small Computer System Interface)

- SCSI is a btw bus specification for connecting peripherals to a computer using parallel transmission interface.
- SCSI devices come in 2 types

① Single-ended devices:

Use one data lead & one group lead to establish single ended signal transmission over the bus.

② Differential devices:

Use two data leads, neither of which are at ground potential. These devices are generally more expensive but are resistant to effects of noise.

Common SCSI components.

There are several components used in SCSI storage systems.

- **Initiator**: An initiator issue request for service by SCSI devices & receive reports

- **Target**: SCSI target is typically a physical storage device. The target can be hard disk or an entire storage array.
- **Service delivery subsystem**

The mechanism that allows communication to occur btw the initiator & target. It usually takes from of cabling.

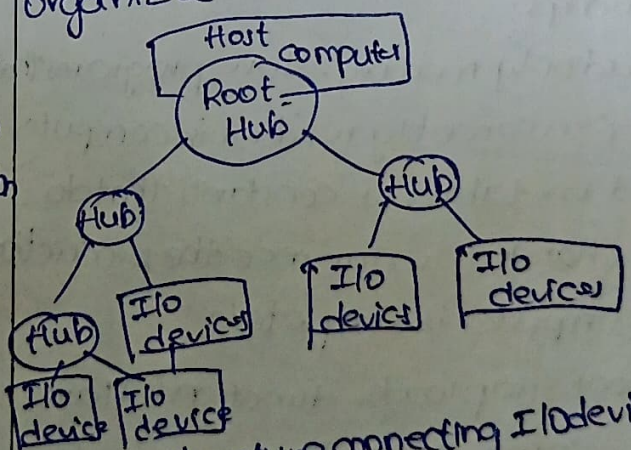
- **Expander**: only used with serial-attach SCSI (SAS). It allows multiple SAS devices to share a single initiator port.

USB (Universal Serial Bus)

- USB is standard interface for connecting a wide range of devices to computer such as keyboard, mouse, smartphones, speakers, cameras etc.
- USB was introduced for commerical use in the year 1995.

USB Architecture:

When multiple I/O devices are connected to computer through USB they allave organized in tree structure.



The tree structure connecting I/O devices to computer using USB hay nodes which are reffered to as a hub.

HUB is intermediatory connecting btw I/O devices & computer.

Types of USB connectors:

(i) USB Type A (ii) Type B (iii) micro USB (iv) mini (v) USB Type C (vi) USB 3.0 micro B.