Ex. No.:	SIMULATION OF COMPUTER COMPONENTS
Date:	

Aim:

To study the simulation of computer components.

Procedure:

1. Motherboard:

The motherboard is a paramount computer component of the central part where everything else is connecting to a motherboard is affable sized circuit board that allows other elements to interact it has orts which are fairing outside a pc are so that you can plug to a monitor, charge computer or plug in a mouse, it also has slots for expansion so that you can install additional accessory ports if you wish to do so, the mother board store low level data like the system time even when a pc is switched off.

2. Power Supply:

The power supply is the device that powers all other mechanism of the pc, it generally plugs into the motherboard, it can connect to either a plug for an outlet (desktop) or an internal battery laptop.

3. Input and Output Devices:

Depending on particular pc, a variable of devices can be connected to send data into it or out from it, common place input data include Mia/laptop and touchpads web icons and ergodic keyboard white output devices are printers, monitors & speaker, removable media like SD cards and flash devices can also be utilized for transferring data between PCs can visit the website for more information.

4. CPU [Central Processing Unit]:

The CPU is the pc's brain since it works the hardest. A CPU does all the calculations needed for a system and varies in speed. The CPU generates that least, and that why a far is installed inside the pc. More powerful CPUs are required for instance computer work or work that necessitate programming multifaceted software or editing high-definition video.

5. RAM (Random Access Memory):

RAM is the shortest-term memory when you open Microsoft word, the computer places it in RAM, and when closing the window, that RAM is released. The more powerful CPUs are required for intense computer work that necessitate programming multifaceted software or editing high-definition video.

6. Hard Disk Drive/Solid State Drive:

Since RAM is short-term, your pc needs a place for strong data permanently, this is what a hard drive is for it has many spinning platers with an arm that write data to the disk. Hard disk is slow and are replaced by they quicker sorted-state drives or SD card with SD-card reader, which consist of flash memory like flash drives or smart phones.

7. Video Cards:

Video cards handle the outputs of images to display they have their own RAM for doing performing the function. Several types of video cards can be bought with different power capabilities and prices.

8. Optical Drives:

They are less regular than they used to be, but several machines still save optical drive for reading DVD or CDs. They can be used for watching movies or listening to music, copying information on a blank disc or installation software to the disc.

Result:

Thus, the study of various components of a pc are listed out.

Ex. No.:	**SIMULATION OF PIPLINE**
Date:	

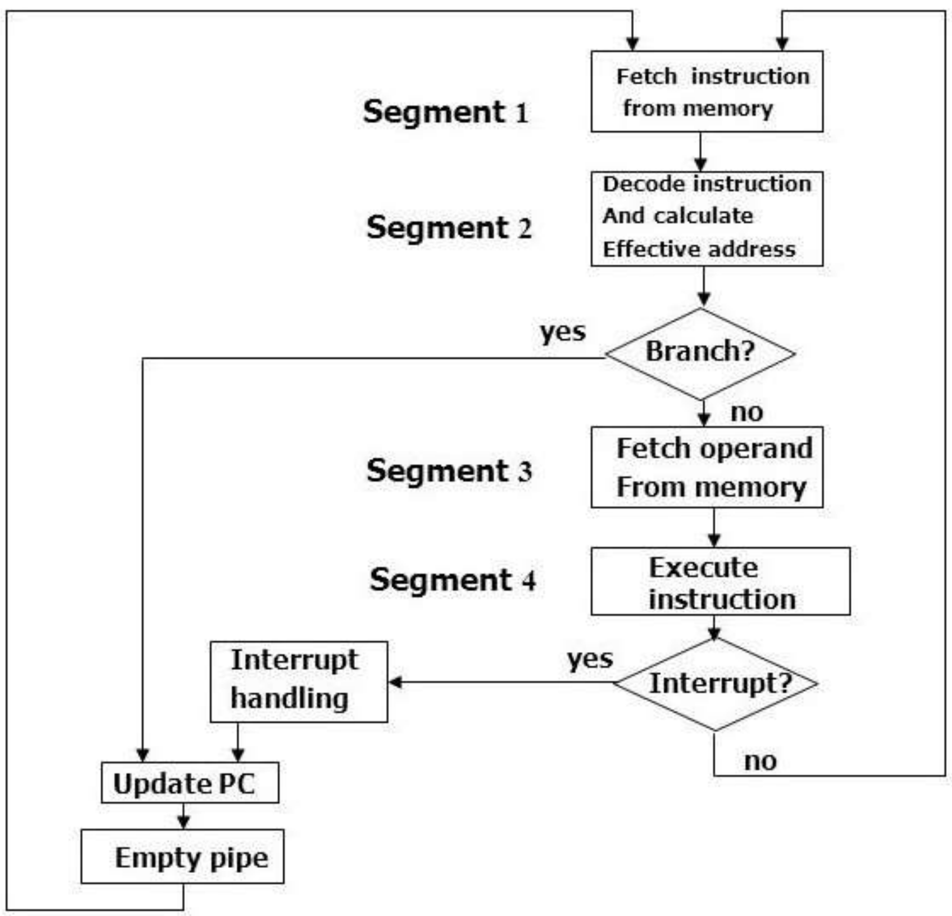
Aim:

To study pipeline feature and instructions pipeline.

Procedure:

- Pipeline processing can occur not only in the data stream but in the instruction stream as well.
- Most of the digital computer with complex instructions require instructs pipeline to carry out operations like fetch, decode and execute instruction. Sequence steps.

1. Fetch instruction from memory.
2. Decode the instruction
3. Calculate the effective address
4. Fetch the operands
5. Execute the instruction
6. Store the secret in the proper place.



Segment 1:

The instruction fetch segment can be implemented using first in first out (FIFO) buffer.

Segment 2:

The instruction fetched from memory is decoded in the second segment and eventually, the effective address is calculated in a separate arithmetic circuit.

Segment 3:

An operand from memory is fetched in the third segments.

Segments 4:

The instruction is finally executed in the last segment of the pipeline organization.

Stage 1 2 3 4 5 6 7 8 910 11 12 13

1 F1 DA FO EX

2 FA DA FO EX

3 FA DA FO EX

4 FA …. FA DA FO EX

5 FA DA FO EX

6 FA DA FO EX

Ex. No.:	SIMULATION OF THREAD LEVELPARALLELISM
Date:	

Aim:

To study simulation of thread level parallelism.

Procedure:

A program can be quickly executed by introducing concurrency using threads.

Each thread can execute part of the same task or it can allocate the same task for different clients in a client server architecture.

1. Header File:

Include the header file p thread.h

#include<p thread.h>

2. The ID For a Thread

Each thread has an object of type p thread t associated with it that tells its ID.

p thread_t id [2];

3. Creating a thread:

A thread is created and starts using the function p thread_create(). It takes fur parameter.

- ID _parameter – reference (or pointer) to the ID of the thread.
- Starting routine – void*_this name of the function that the thread starts to execute if the function's return type is void.
- Argument – void*_this is the argument that the starting routine tasks. If the task multiple arguments, a struct is used.

4. Existing a Thread:

P thread_exit() is used to exit a thread.

//Global Variable;

Int i=1;

5. Waiting for a Thread:

A parent thread is made to wait for a child using p thread join(). The two parameters of the functions are.

Thread ID_p thread_t – The ID of the thread that the parent waits for.

Reference to return value – void*_the value returned by the existing thread is caught by this pointer.

```
Int*ptr;
pthread_join(id, & ptr)
```

Ex.No:	IMPLEMENTATION OF LOGIC GATES
Date:	

Aim:

To Implement Basic Logic Gates in logicaly software.

AND Gate:

An AND Gate is a digital logic gate that has two or more inputs and one output. The output of an AND gates is True only when all of the inputs are true.

OR Gate:

An OR Gate is a digital logic gate that has two or more inputs and one output. The output of an OR gate is true if any of the inputs are true

NOT Gate:

A NOT Gate is a digital logic gate that has one input and one output. The output of NOT gate is logical opposite of its input.

NAND Gate:

A NAND Gate is a digital logic gate that is a combination of an AND gate followed by a NOT gate. The output of a NAND gate is false only when all of the input are true.

NOR Gate:

A NOR Gate is a digital logic that is a combination of an OR gate followed by a NOT gate. The output of a NOR gate is true only when all of its inputs are false.

XOR Gate:

A XOR Gate is digital logic gate that has two inputs and one output of an XOR gate is true only when its inputs are different.

XNOR Gate:

A XNOR Gate is digital logic gate that has two inputs and one output of an XNOR gate is true only when both of its inputs are the same.

Result:

All the logic gates are implemented and verified using truth table.

Ex.No:	IMPLEMENTATION OF HALF AND FULL ADDER
Date:	

Aim:

To implement half and full adder using logicly software.

Half Adder:

A Half Adder in a digital circuit that adds two single bit binary number and output**.** The sum and carry as separate without taking into account any carry from previous additions.

Full Adder:

A Full Adder is composed of two half adders and an additional OR gate. It is used in many digital circuit such as microprocessor and calculator. It can also be combined with others full address to create larger binary adders for performing arithmetic operations or larger binary numbers.

Result:

The Logic gates were designed using logicly software and the truth table was verified.

Ex. No:	IMPLEMENTATION OF HALF AND FULL SUBTRACTOR
Date:	

Aim:

To implement Half and Full Subtractor using logicly software.

Half Subtractor:

A half subtractor is a digital circuit that performs binary subtraction on two single bit inputs produce a difference and a borrow output. It is composed of a XOR gate and a NAND gate. It is used in many digital circuit, such as microprocessor and calculator.

Full Subtractor:

A full subtractor is a digital circuit that performs subtraction on the inputs two single bit input and a barrow input and produce a difference and a barrow output. It is combined of two half subtractor and additional OR gate. It is used in many digital circuit such as microprocessor and calculator.

Result:

All the logic gates are designed using logicly software.

Ex. No.:	**SIMULATION OF CACHE MEMORY**
Date:	

Aim:

To study the simulation of cache memory.

Procedure:

1. Define Cache Parameters:

- Cache Size (S)
- Block/Line Size (B)
- Associativity (A)
- Replacement Policy (e.g., LRU, FIFO)
- Initialization (clear cache, load initial data, etc.)

2. Represent Memory Hierarchy:

- Consider the main memory and cache hierarchy.
- Break down main memory into blocks corresponding to cache lines.

3. Initialize Cache:

- Set all valid bits to false or initialize according to your model.
- If necessary, preload initial data into the cache.

4. Simulate Memory Access:

- For each memory access, determine the cache set using the index bits.
- Check if the desired data is present in the cache (cache hit).
- If hit, update cache state (e.g., LRU information).
- If miss, decide which block to replace (based on replacement policy).
- If replacement is required, load the new block into the cache.
- Update cache state accordingly.

5. Track Cache Hit/Miss Statistics:

- Maintain counters for cache hits and misses.
- Record additional information for analysis, such as access patterns.

6. Repeat for Each Memory Access:

- Iterate through the memory access sequence, applying the simulation steps for each access.

7. Analyze Results:

- Calculate hit rate, miss rate, and other relevant metrics.
- Evaluate the impact of different cache parameters and policies on performance.
- Consider the effects of changes in cache size, associativity, and block size.

8. Optimize and Experiment:

- Experiment with different cache configurations to find optimal settings.
- Explore the impact of varying cache parameters on performance.

9. Documentation and Reporting:

- Document the simulation process, including assumptions and settings.
- Present results and insights gained from the simulation.