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 Computer are listed out.

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
```

Result:

The program is executed successfully and the output is verified.

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.

Result: The program is executed successfully and the output is verified.

TASK OR THREAD CREATION:

Define tasks or threads that will be executed on each processor. Specify their characteristics, such as arrival times, execution times, and dependencies.

SIMULATION INITIALIZATION:

Set up the initial state of the system, including processor states, memory contents, and any initial tasks or threads.

SIMULATION EXECUTION:

Execute the simulation over time, advancing in discrete steps or cycles. Model the execution of tasks on processors, memory accesses, cache hits\misses, and communication through the interconnection network.

RESULT: The program is executed successfully and the output is verified.

DATA STRUCTURES:

 Implement data structures to represent vector registers, scalar registers, memory, and other components of the vector processor. Use appropriate data structures to efficiently model the vector operations.

INSTRUCTION SET ARCHITECTURE:

 Implement the instruction set of the vector processor. Define data types, vector instructions, and their corresponding opcodes. Map these instructions to corresponding functions or methods in your simulation.

PIPELINE MODEL:

 If the vector processor has a pipeline, model each stage of the pipeline. Implement functions for instruction fetch decode, execute, and write-back. Handle hazards and dependence between instructions.

MEMORY HIERARCHY:

 Simulate the memory hierarchy, including caches and main memory. Important cache management, including cache hits and misses.

VECTOR OPERATIONS:

 Implement the core vector operations based on the processor's capabilities. This includes vector addition, multiplication, and other vectorized computations

RESULT: The program is executed successfully and the output is verified.

DESIGN PARALLEL DATA STRUCTURES:

 Modify or design data structures that support parallel processing. Consider how data is distributed among processing units and how communication and synchronization between these units will occur.

IMPLEMENT PARALLELIZATION TECHNIQUES:

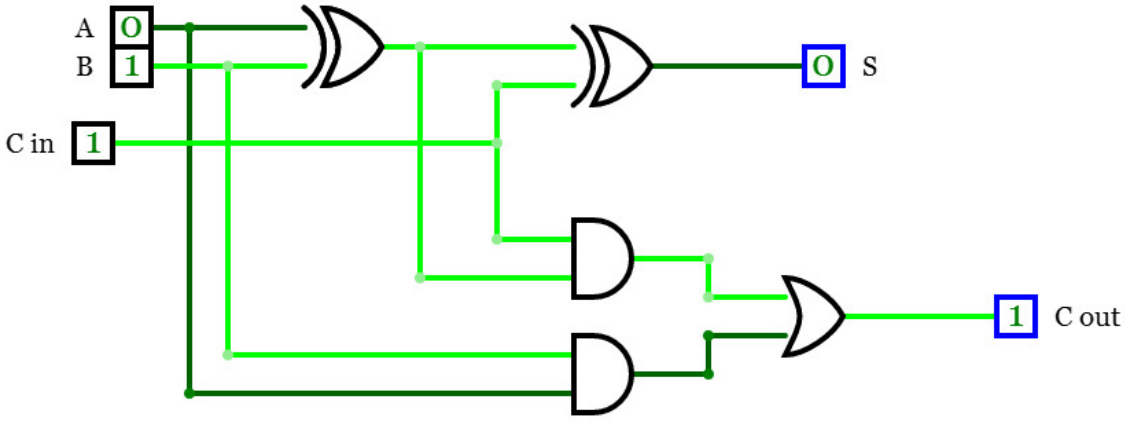
 Implement the identified parallelization techniques within you simulation environment. This might involve utilizing parallel programming constructs such as threads or processors, GPU acceleration, or specialized parallel processing libraries.

INSTRUMENTATION AND MEASUREMENT:

 Introduce instrumentation into your simulation to measure and analyze the performance of the parallelized algorithm. Track metrics such as execution time, speedup, and efficiency.

RESULT: The program is executed successfully and the output is verified.

CIRCUIT DIAGRAM:



RESULT: The one-bit adder is executed successfully and the output is verified.