Varun Khadayate A016 03-11-2020

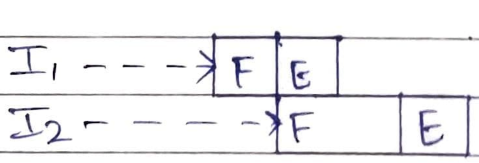
COA

Assignment-IV

Q2. It assumes the pipeline is running absolutely smooth & every instruction requires exactly same time in each functional stage of the pipeline and therefore after the first instruction, every instruction is completing its execution in exactly one clock period. However, in the practical situation, it is not the case. There are many resources bottle necks & operational difficulties that dont allow the pipeline to operate in ideal condition. All such practical issues are resource bottlenecks that make the instruction pipeline to deviate from its ideal characteristics are called as Hazards. Hazards introduce inefficiency & therefore increase the execution time of instruction in certain cases. Therefore, hazards in general degrade the instruction pipeline performance. The hazards can be classified into 3 types:-

- Structural Hazard
- Data Hazard
- Control Hazard.

- Structural Hazard



Execution Unit is idle
its a stall.

Structural Hazards are the hazards introduced in an instruction pipeline by virtue of structural problems. Consider a situation where the fetch and execute stages of the pipeline are of dissimilar function duration. Such situation is called as pipeline stall which pushes the timings forward making instruction to take longer time to execute & therfore degrading the performance. Alternatively, if the current instruction that is under execution requires a memory reference & at the same time fetch stage also requires access to the memory for fetching the next instruction in the program. In these case both fetch & execute stage are required for the memory reference over the buses. However, the buses can given only one of them at a time. This forces the 2 operations to be performed one after the other due to the resource bottleneck on the buses. The result is that the instruction execution takes longer time & pipelined performance is degraded.

- Data Hazard.

Data Hazards are introduced into instruction pipeline due to the consecutive instruction being dependent. The 2 consecutive instructions are said to be dependent if atleast one of the source or destination operand reffered in these 2 instructions are common. There are 3 types of dependencies observed in data hazards:

1. Read After Write [RAW] dependency: The destination operand of the ~~source~~ previous instruction is one of the source operands in the subsequent instruction ~~being dependent. The 2 consecutive instruction are said to be dependent if at least one of the source or destination operand reffered to in these 2 instruction are~~

1. Read And Write [RAW] Dependency: In this type of data dependency hazard, the destination operand of the previous instruction is one of the source operands in subsequent instruction. This type of dependencies always lead to the data hazard in instruction pipeline. This happens because the subsequent instruction cannot fetch its sources operands unless the previous instruction is restored to its destination operand & generally results in one clock period stall as a subsequent instruction pipeline.

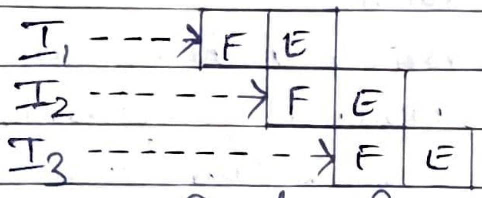
2. Write After Read [WAR] Dependency: In this type, one of the source operands of previous instruction has stored its destination operand ~~and~~ of subsequent instruction. This happens because the subsequent instruction cannot fetch its source operand unless the previous instruction has stored its destination operand & generally results in one clock-period stall as a subsequent instruction execution waits. This type of instruction doesnot lead to data hazard in instruction pipeline.

3. Write After Write [WAW] Dependency: In this, the destination operands of the previous instruction are same as destination operand in the subsequent instruction.

- Control Hazard

These are caused due to the control transfers encountered in the user programs. Generally caused by conditional & unconditional branch instruction & subroutine calls in user program. Can also be caused by system events such as interrupts, exceptions & task switches. They are hazards that contribute max^m to pipeline stalling.

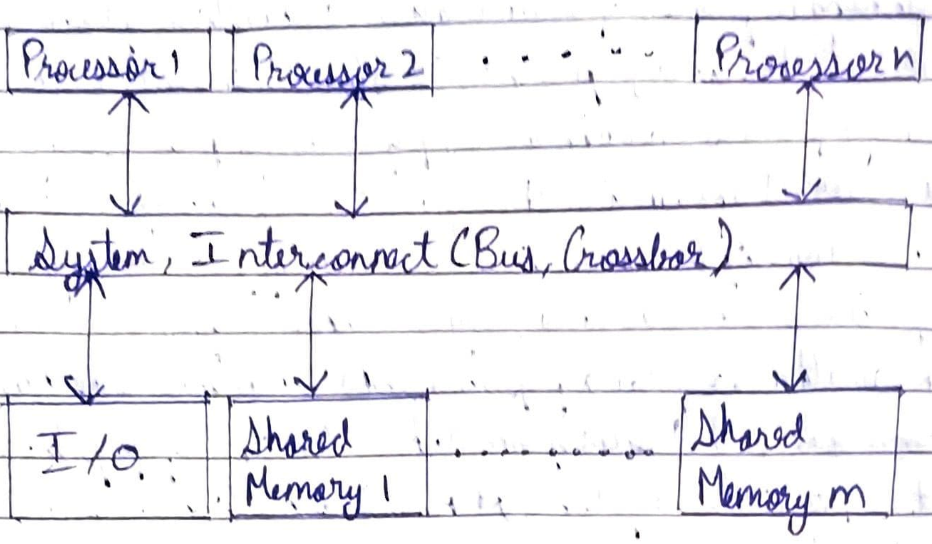
Q1. Pipelining is process of accumulation instruction from the prepro-cessor or through a pipeline. It stores & executes instruction in orderly process. Also known as pipelining processing. It increases the overall instruction throughout. Pipelining is a technique where multiple instruction are overlapped during execution. Pipeline is divided into stages & these stages are connected with one another to form a pipeline like structure. Instruction enter from one end & exit from other. In case of pipelined processor, it is possible to fetch the next instruction at same time while the prev. instruction is being executed.



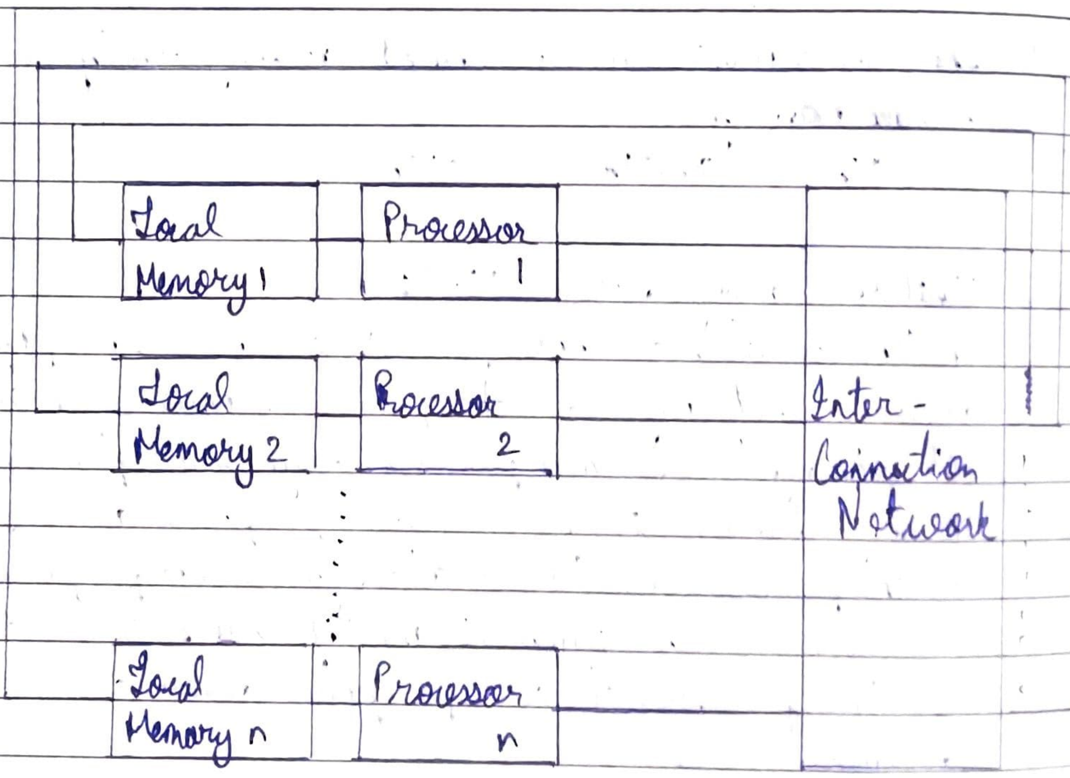
Pipeline Processor in CPU.

Q3. There are 3 most common shared memory multiprocessor models namely:
- UMA - NUMA - COMA

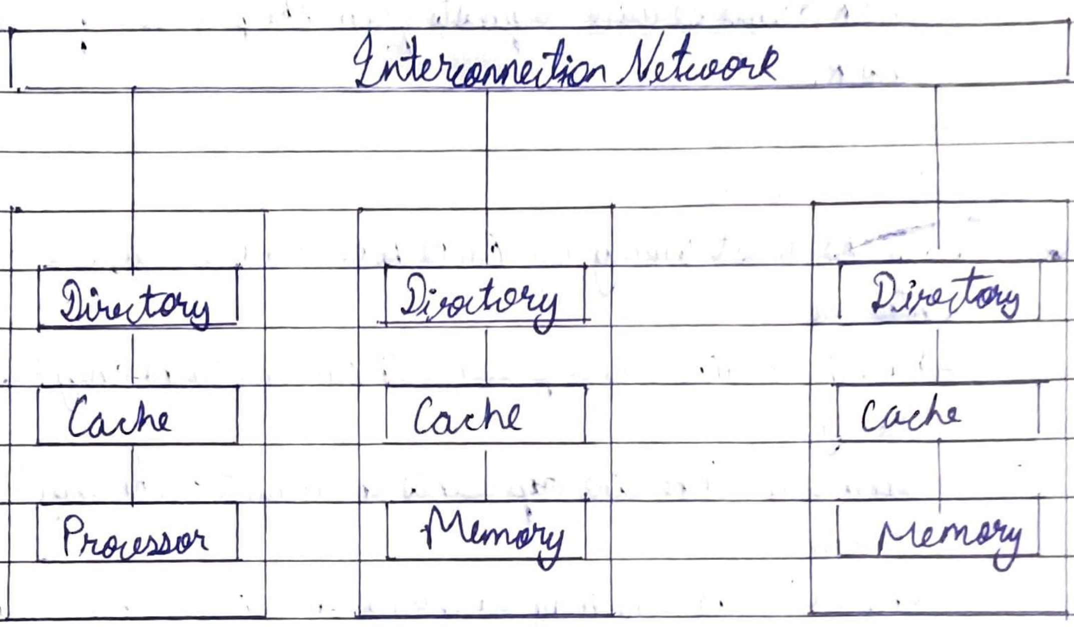
- Uniform Memory Access [UMA]:
In this model, all the processors share the physical memory uniformly. All the processors have equal access to all memory words. Each processor may have a private cache memory. Same rule is followed for peripheral devices. When all the processors have equal access to all the peripheral devices, the system is called symmetric multiprocessor. When only a few or one processor can access peripheral device are called as assymetric processor.



- Non-Uniform Memory Access [NUMA]
In this model, the access time varies with the location of memory word. Here, the shared memory is physically distributed among all the processors called local memories. The collection of all local memories forms a global address space which can be accessed by all processors.



- Cache Only Memory Architecture [COMA]
This model is a special case of NUMA model. Here, all the distributed main memory are converted to ~~ch~~ cache memories.



Q4. 1. MSI Protocol
This is basic cache coherence protocol used in multi-processor system. The letters of protocol name identify possible states in which a cache can be; So, for each block can have one of following states:
- Modified : The block has been modified ~~a~~ cache. So, a cache with a block in 'M' state has responsibility to ~~to~~ write the block to backing store when it is evicted.
- Shared : This block is not modified and is present in atleast one cache. The cache can evict the data without writing it to backing store.
- Invalid : This block is invalid & must be fetched from another cache if it is to be stored in the cache.

2. MOSI Protocol

This is an extention to MSI protocal. It adds the following state in MSI protocol:

- Owned: It indicates that the present processor owns this block & will service requests from the processor for the block.

3. MESI Protocol

It is the most widely used Cache Coherence Protocal. The states are:

- Modified: Cache line is present in current cache only & is dirty.
- Exclusive: Cache line ~~r~~presents in current cache only & is clean.
- Shared: Cache line may be stored in other caches of the machine.
- Invalid: Cache line is invalid.

4. MOESI Protocol

This a full cache coherence protocal that encompasses all of the possible states comonly used in other protocols. The states are:

- Modified: A cache line in this state holds the most recent, correct copy of the data while the copy in the main memory is ~~ino~~ incorrect and no other processor holds a copy.
- Owned: A cache line in this state holds the most recent, correct copy of the ~~st~~ data.
- Exclusive: A cache line in this holds the most recent, correct copy of the data.
- Shared: A cache line in this state holds the most recent, correct copy of the data.
- Invalid: Cache line does not hold a valid copy of data.