18 Lecture - CS501

Important Subjective

1. **What is pipelining, and how does it work?**
   Answer: Pipelining is a technique used in computer architecture to increase the processing speed of a CPU. It divides the processing of an instruction into smaller sequential stages, allowing multiple instructions to be processed simultaneously. Each stage in the pipeline performs a specific task, and the output of one stage becomes the input for the next.

   What is a pipeline stage, and how many stages are typically used in a pipeline?
   Answer: A pipeline stage is a specific step in the pipelining process, and typically five stages are used in a pipeline. These stages are instruction fetch, instruction decode, execute, memory access, and write-back.

   What is a pipeline hazard, and how can it be resolved?
   Answer: A pipeline hazard is a delay in the pipeline caused by an instruction that depends on a previous instruction. It can be resolved by inserting pipeline stalls, forwarding data, or reordering instructions.

   What is pipeline flushing, and why is it necessary?
   Answer: Pipeline flushing is the process of discarding instructions in the pipeline when a pipeline hazard occurs. It is necessary to prevent incorrect results and maintain the correct order of instruction execution.

   What is a data hazard, and how can it be resolved?
   Answer: A data hazard is a type of pipeline hazard that occurs when a later instruction depends on the result of a previous instruction. It can be resolved by forwarding data or inserting pipeline stalls.

   What is branch prediction, and how does it work?
   Answer: Branch prediction is a technique used to improve the performance of pipelined processors by predicting the outcome of a branch instruction before it is executed. It works by analyzing the program's behavior and history to predict whether a branch is taken or not taken.

   What is instruction-level parallelism, and how is it achieved?
   Answer: Instruction-level parallelism is the ability to execute multiple instructions simultaneously. It is achieved through pipelining, superscalar execution, and out-of-order execution.

   What is pipeline efficiency, and how is it calculated?
   Answer: Pipeline efficiency is the ratio of the number of instructions executed in a unit of time to the number of cycles required to execute one instruction. It is calculated as (number of instructions executed / number of cycles) x 100%.

   What is pipeline depth, and how does it affect performance?
   Answer: Pipeline depth is the number of pipeline stages used in a pipeline. It affects

performance by increasing the latency of the pipeline and introducing additional overhead.

What is superscalar execution, and how does it differ from pipelining?
Answer: Superscalar execution is a technique used to achieve instruction-level parallelism by executing multiple instructions simultaneously. It differs from pipelining in that it can execute multiple instructions from the same program simultaneously, while pipelining can only execute multiple instructions from different programs simultaneously.