BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
Second Semester 2019-2020
CS F342 Computer Architecture
Topic: Design of pipelined MIPS Datapath with Hazards

PROJECT - 1

A *pipeline processor* is comprised of a sequential, linear list of *segments*, where each segment performs one computational task or group of tasks

There are three things that one must observe about the pipeline. First, the work (in a computer, the ISA) is divided up into pieces that more or less fit into the segments alloted for them. Second, this implies that in order for the pipeline to work efficiently and smoothly, the work partitions must each take about the same time to complete. Otherwise, the longest partition requiring time T would hold up the pipeline, and every segment would have to take time T to complete its work. For fast segments, this would mean much idle time. Third, in order for the pipeline to work smoothly, there must be few (if any) exceptions or hazards that cause errors or delays within the pipeline. Otherwise, the instruction will have to be reloaded and the pipeline restarted with the same instruction that causes the exception. There are additional problems we need to discuss about pipeline processors, which we will consider shortly.

Pipeline processors have several problems associated with controlling smooth, efficient execution of instructions on the pipeline. These problems are generally called *hazards,* and include the following three types:

- *Structural Hazards* occur when different instructions collide while trying to access the same piece of hardware in the same segment of a pipeline. This type of hazard can be alleviated by having redundant hardware for the segments wherein the collision occurs. Occasionally, it is possible to insert stalls or reorder instructions to omit this type of hazard.
- *Data Hazards* occur when an instruction depends on the result of a previous instruction still in the pipeline, which result has not yet been computed. The simplest remedy inserts stalls in the execution sequence, which reduces the pipeline's efficiency. The solution to data dependencies is twofold. First, one can *forward* the ALU result to the write back or data fetch stages. Second, in selected instances, it is possible to restructure the code to eliminate some data dependencies.
- *Control Hazards* can result from branch instructions. Here, the branch target address might not be ready in time for the branch to be taken, which results in *stalls* (dead segments) in the pipeline that have to be inserted as local wait events, until processing can resume after the branch target is executed. Control hazards can be mitigated through accurate branch prediction (which is difficult), and by *delayed branch* strategies.

Design a 5-stage pipelined MIPS datapath that detects and remedies hazards for the following set of instructions: -

```
sub   $2,   $1,   $3
and   $8,   $2,   $5
or    $9,   $6,   $2
add $5, $5, $6
sub $4, $3, $6
beq $5, $6
```