Computer Architecture - CS 301

Rishit Saiya - 180010027, Assignment - 5

October 3, 2020

1

A hazard is defined as the possibility of erroneous execution of an instruction in a pipeline. A data hazard represents the possibility of erroneous execution because of the unavailability of data, or the availability of incorrect data. The following are very general hazards in the dependencies:

- ***Data Hazard***
  RAW Hazard: In an in-order pipeline, RAW is the only data hazard that can happen. A preceding instruction is always ahead of a succeeding instruction in the pipeline in an in-order pipeline. A RAW (Read after Write) hazard is where we read after write inappropriately because of insufficient clock cycles time in between.
  For example:

  ```
  add r1, r2, r3
  sub r3, r1, r4
  ```

  Here the instruction 2 will read incorrect values.

  Also other several which don't obey in-order pipelining (which are used in modern processors) in which it is possible for later instructions to execute before earlier instruction have hazards like WAW, WAR, etc.

  WAW Hazard: A WAW (Write after Write) hazard is where we write after write inappropriately because of insufficient clock cycles time in between.
  For example:

  ```
  add r1, r2, r3
  sub r1, r4, r3
  ```

Here the instruction 2 cannot write the value of r1, before instruction 1 writes to it. This leads to WAW hazard.

WAR Hazard: A WAR (Write after Read) hazard is where we write after read inappropriately because of insufficient clock cycles time in between.
For example:

```
add r1, r2, r3
add r2, r5, r6
```

Here the instruction 2 cannot write the value of r1, before instruction 1 reads it. This leads to WAR hazard.

- ***Control Hazard***
  A Control Hazard represents the possibility of erroneous execution in a pipeline because instructions in the wrong path of a branch can possibly get executed and save their results in the memory, or in the registry file.

  For example:

  ```
  beq .foo
  mov r1, 4
  add r2, r4, r3
  ```

  Here the 2 instructions fetched immediately after a branch instruction might have been fetched incorrectly. Thus, these instructions are said to be on wrong path.

- ***Structural Hazard***
  A structural hazard may occur when two instructions have a conflict on the same set of resources in a cycle.

  For example:

  ```
  st r4, 20[r5]
  sub r8, r9, r10
  add r1, r2, 10[r3]
  ```

  Here the instruction 3 tries to read 10[r3] which is in MA unit in Cycle 4. But at the same time, instruction 1 tries to write 20[r5] in MA unit in Cycle 4. So, for the same cycle, we cannot have multiple executions in same unit and this leads to Structural Hazard.

Effect on Performance: A pipelining is designed to increase the efficiency and perform more instruction executions in same given amount of time. But we see that with Hazards, our performance is depleted because the results are not met with certainty and their might exist incorrect fetches from the memory or also contradictory references in instructions.

2

A RAR is not a valid hazard because we have 1 read and after that once more to the same register. In an advanced pipeline also they can overtake each other. As long as any W (write is not involved in dependency), it is definitely not a hazard.

For example:

```
add r2, r4, r3
add r5, r4, r3
```

Here both instructions can read from r4 and there will be no contradiction and hence, RAR is not a potential hazard.