How Data Hazards can be removed effectively

Muhammad Zeeshan, Saadia Anayat, Rabia and Nabila Rehman

Abstract—For fast Processing of instructions in computer architecture, the most frequently used technique is Pipelining technique, the Pipelining is consider an important implementation technique used in computer hardware for multi-processing of instructions. Although multiple instructions can be executed at the same time with the help of pipelining, but sometimes multi-processing create a critical situation that altered the normal CPU executions in expected way, sometime it may cause processing delay and produce incorrect computational results than expected. This situation is known as hazard. Pipelining processing increase the processing speed of the CPU but these Hazards that accrue due to multi-processing may sometime decrease the CPU processing. Hazards can be needed to handle properly at the beginning otherwise it causes serious damage to pipelining processing or overall performance of computation can be effected. Data hazard is one from three types of pipeline hazards. It may result in Race condition if we ignore a data hazard, so it is essential to resolve data hazards properly. In this paper, we tries to present some ideas to deal with data hazards are presented i.e. introduce idea how data hazards are harmful for processing and what is the cause of data hazards, why data hazard accord, how we remove data hazards effectively. While pipelining is very useful but there are several complications and serious issue that may occurred related to pipelining i.e. load delay, branch delay data dependence, etc. Performance of pipelining technique is relay on data dependency between instructions and Data dependency some time generates pipeline hazards between instructions. In order to effectively deal with data hazards we also discuss pipelining complications like data dependence. Data dependence is a state where one instruction is relay on the data of preceding statement. We clearly impose these conditions of wrong result complication, data hazards and data dependence.

Index Terms— Data Hazard, Pipelining, Race Condition, Forwarding

1 Introduction

Pipelining technique ensures fast processing of instructions because there is a continuous and overlapped movement of multiple instructions so execution is done in fewer cycles. Pipeline architecture is a series of sub stages and these stages are connected with each other where some work is done at each stage, each stage is responsible for predefine tasks to be executed and the whole work is not done until it has passed through all the stages of pipeline. Several problems can affect the performance of pipelining technique such as hazard which is any source of potential damage or risk. It is a situation or condition that creates or increases the chances of loss. In the same way, pipelining hazard can cause loss in performance efficiency. Condition that can result in incorrect execution of instructions is known as pipeline hazard. As we know that instructions in the pipeline are being executed in parallel and when an instruction or set of instruction is rely upon the result of previous instruction that is under process and not yet complete by the CUP, Although there exist three type of pipeline hazards one of them is known as data hazards which can be defined as hazards that occur when some time an instructions access some invalid data value of previous instruction, previous instruction cannot update that time or update letter.

1.1 Data Hazard Examples:

Following is a simple example to explain data hazard b/w two instructions.

ADD **R**5, R3, R4

SUB R7, **R**5, R6

There exists a data hazard b/w these two instructions. The Register **R5** is processes in clock cycle 6 there for instruction SUB cannot proceed beyond stage 3 (Instruction de decode/operand fetch) until ADD instruction leaves the pipeline. This example refers to RAW or read after write data hazard where 2nd instruction tries to read the sources (register) before 1st instruction writes to it. SUB instruction depends upon the result of R2 and it cannot proceed further or access until ADD instruction completes its cycle. Data hazards can negatively influence pipelining performance if not handled properly. Performance would be slow down and processing will become more time consuming that is why it is very important to deal with data hazards properly. Ignoring potential data hazard can result in race condition also known as race hazard which refers to a situation or Condition where several instructions access and manipulates the same data concurrently and the output of manipulation depends upon the order in which the access takes place. For the protection against race condition, proper synchronization is required. In order to handle or remove data hazards properly we need to find out the reasons for the occurrence of hazards only then we can deal with data hazards properly. In this paper, we will discuss about complications related to pipelining, pipeline data hazards, Impact of data hazards on pipelining performance, reasons behind occurrence of data hazards and how we can effectively remove data hazards. This paper is divided into different sections. After the brief introduction a review of pipelining and data hazard related work is given in section 2. In section 3, reasons for occurrence of data

- ***Muhammad Zeeshan*** *is with Department of Computer Science & Information Technology as a student of MSCS at VU, Lahore, Pakistan. He is currently an employ of FCSC Peshawar, under Ministry of Defence Islamabad,Pakistan E-mail:ms150400558@vu.edu.pk*
- ***Saadia Anayat*** *is with Department of Computer Science & Information Technology as a student of MSCS at VU, Lahore, Pakistan. E-mail:ms150401012@vu.edu.pk*
- ***Rabia*** *is with Department of Computer Science & Information Technology as a student of MSCS at VU, Lahore, Pakistan. E-mail:ms150400912@vu.edu.pk*
- ***Nabila Rehman*** *is with Department of Computer Science & Information Technology as a student of MSCS at VU, Lahore, Pakistan. She is currently working as a Computer Science Teacher in Punjab School Education Department Dist. Khanewal E-mail:ms140400059@vu.edu.pk*

hazards are highlighted along with the ways to remove or resolve those data hazards effectively. Impact of data hazards on pipelining performance is also discussed in the form of comparison of pipelining with and without data hazards to clarify that how much it is important to remove data hazard for fast execution of pipeline. Last section 4, contains conclusion and references.

2 Related Work

Pipelining is a mechanism where multiple workloads is distributing over multiple stages and perform exactly in pipeline order. One is work is in next stage and till not completed next work is enter into expectation pipeline nest stage. In the pipelining is multiple instructions can be executed at the same time by performed in multiple stages [12] explains that pipeline execution of instruction is a technique in which there is overlapping of multiple instructions at the same time, one instruction is process at next pipeline till not completed next instruction enters into first pipeline and that process increases instruction execution throughput, in the parallel execution of instruction the result of previous executed instruction is used into next one instruction- pipelining concepts increase system throughput but it may also some time become the reason of wrong result and affect performance-no of instructions processed per second and results in fast processing but there are some risky situations or hazards which not only affect performance but also may cause incorrect computation results.

2.1 Pipeline example:

Because pipeline is also use in natural things so we take the example of natural work, we took the example of Laundry. We have four workloads: A, X, Y, and Z. there are four laundry operations: Wash, Dry, Fold and place into Drawers. Every operation takes 30 minute to complete its work.

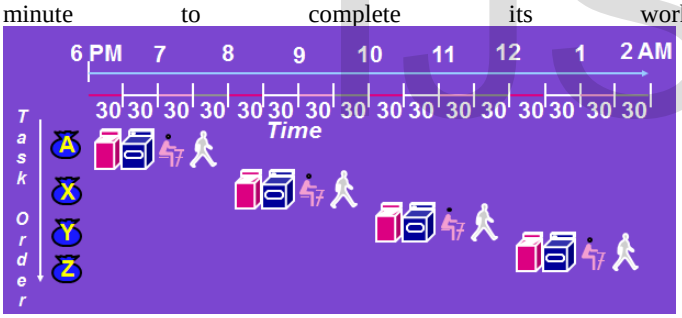

Diagram 1: Sequential Laundry

If laundry work is take complete into sequential order first work load complete its work in two hours. As laundry start at 6PM so at time of 8PM next work load start and it also takes two hours to complete its four tasks like wash, dry, folding and place into drawer. Every stage takes 30 minute so four work load complete at 2AM mean whole laundry work takes 8 hours. Now we perform that work into pipeline order.

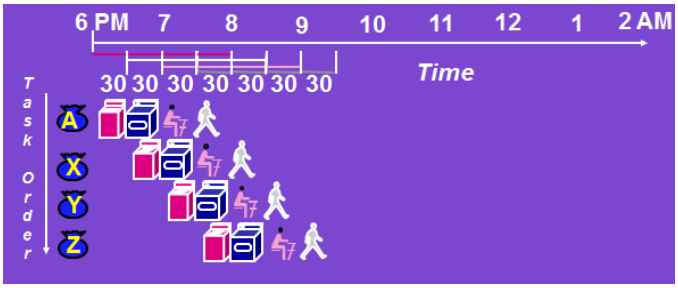

Diagram 2: pipelined laundry

In the pipelined laundry four tasks of single work load are performed into pipelined order one tasks is done and entered into next stage, 2nd load start its 1st task like load A starts its 1st tasks of wash at 6PM it complete at 6:30PM and entre into next stage of dry that time next load X is entered into pipeline and use wash. When load a complete 2nd task of dry it enter into next stage so load X enters into dry phase so that process works in pipeline order. In pipelined order whole laundry takes 3.5 hours for four work load. So increases the throughput. In the pipeline execution may sometime occurs unacceptable condition named hazard, Data hazards are one of the three types of hazards that occur in pipelining i.e. structural hazards, control hazards and data hazards. During its clock cycle the hazards prevent the next instructions in the pipeline from being executed and minimize the ideal speedup of performance achieved by pipeline execution technique. Newly research on pipeline introduce there are three major execution stages where data hazard occur and those stages are as follows

2.2 RAW (Read after Write) dependence

In the pipeline the unrealistic situation where execution of instruction refers to a result of the next instruction, that is under execution or updated later. If there are two instructions and Instruction I occur before instruction II then RAW can refer to a situation where instruction II tries to read source register before instruction I writes to it although the source register of one instruction is destination register of other register.

For Example: instruction_{I} tries to read operand before instruction_{II} writes it;

$I: ADD\ r10, r2, r3$

$II: SUB\ r6, r10, r3$

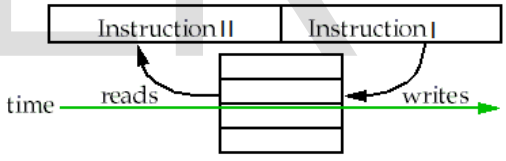

Diagram 3: Read after Write

2.3 WAR (Write after Read) anti-dependence

It is a situation that occurs when instruction II tries to write a destination before instruction I reads it. It occurs due to concurrent execution of instructions. It is also called Name dependence (renaming). For example

$I: SUB\ r6, r10, r3$

$II: ADD\ r10, r2, r3$

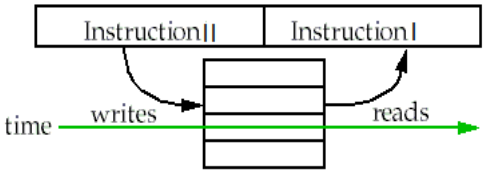

Diagram 4: Write after Read

2.4 WAW (Write after Write)

It is a situation that occurs when an instruction tries to write an operand before the first instruction writes it. This kind of hazard can also occur due to concurrent execution of instructions. For example

$$I: SUB\ r10, r4, r3$$

$$II: ADD\ r10, r2, r3$$

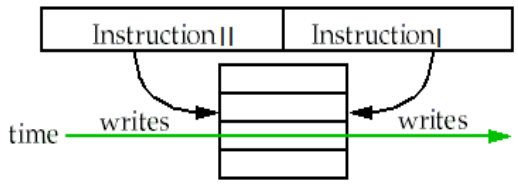

Diagram 5: Write after Write

In this example it may not calculate inconsistence result but it is a write after write hazard.

2.5 RAR (Read after read)

This situation occurs when one instruction read and operand before that other instruction read it. But that situation is not a hazard because register value is not changed.

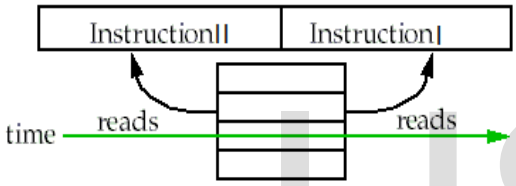

Diagram 6:Read after Read

According to [13] incorrectly read, written or overwritten data give way to data hazards occur. According to their work related to pipelining hazards many other hazard occur when instruction depends/use the result of other instruction that are under processing of still under process at other processor or other pipeline stage, Due to scheduling failure instruction try to access data that is not available that time i.e. sometime instruction is access information too early before data is available in the destination register or some time instruction access data too late after it had been overwritten in the destination register. Data hazards influence the performance of pipelining. Pipelining performance is dependent upon data dependency b/w instructions. These dependency b/w instructions impose data hazards which can put negative impact on overall performance of the system.

2.5 Impact of Data Hazards on Pipelining Performance

A miner delay in one pipeline or one instruction effect on overall performance of the system, because complete pipeline procedure relay on the result of the other instruction as will if one is late so other instruction may wait for its results. Pipelining performance can be negatively affected due to data hazards because data hazards occur when data is read, written or overwritten incorrectly this in-correction in data can cause delay in processing and produces incorrect results. Thus overall performance efficiency becomes slow due to hazard. With pipelining, we cannot start instruction one clock earlier since it is already in pipeline mean instruction is under process in other pipeline as will. When data hazard occurs then an instruction cannot access required data because the previous instruction has not computed or stored it, sometime data hazard occur in very critical situation one running instruction is not till commit next one is try to use this result in its execution. More there are chances of data hazards more negative impact on performance will be also think able because pipeline become useless if it may not give desire performance. Many instructions that a processor is able to run or execute parallel with pipelining, instructions are IF (Instruction fetch), ID (Instruction decode), OF (operand fetch), Execution of instruction or ALU (Arithmetic Logic Unit) operation, Memory access and Register write Register Rewrite. These all stages are interconnected with one to the next stage to make complete pipe, performance of each stage is very important for other stage because each stage is depend on other result and execute in parallel. In the pipeline procedure instructions comes at one end of pipeline and process through the each stage of the pipeline and exit at other the end. So if any kind of data hazard or any other kind of hazard exists or occurs during pipeline execution then it would affect the normal flow of instructions processing and it may cause in incorrect computations of final result.

2.6 Pipelining complications:

According to [4], there can be several complications related to pipelining and those complications are listed here.

2.6.1 Data dependence

It is occurs when an instruction are relay on other instruction results like in one stage instruction uses the result of an instruction that is under processing in previous stage, that is call Data dependence.

2.6.2 Branch delay

In pipelining some time branches cause many problems for processors that work in pipelined order. Because before testing the branch condition it is too much difficult to predict whether a branch will be taken on that stage or not. If branch is not taken we can insert stalls or NOP instructions after the branch instruction so it one cycle delay is pass-through.

2.6.3 Slow Stage

it is a big complication in pipelining if one stage is processing or executing instruction too much slow speed that effect all other faster stage of the pipeline.

2.6.4 Load delay

The time that is used by loading the value into register that is currently used by the next instruction. These pipelining complications decrease overall performance, causes reduction or delay in processing or may cause production of incorrect results.

3 Analysis

We will discuss the reasons behind the occurrence of data hazards and effective ways to remove data hazards effectively. We will also compare pipelining with and without data hazards to analyze that why it is needed to remove data hazards and what is the impact of data hazards on pipelining ideal performance. Because due to data hazards processing of instructions cannot be done correctly and consumes more time. Data hazards can occur due to several reasons and we need to deal with each kind of reason that can cause data hazards in order to remove data hazards effectively. If reasons of their occurrence are not known then data hazards can be hard to solve and even lead to serious damage or performance issues.

3.1 REASONS FOR DATA HAZARD OCCURENCE

Data hazards can occur due to several reasons. Few reasons can be as follows

1. Data hazards occur due to dependency b/w instructions known as data dependence**:-** It is occurs when an instruction are relay on other instruction results like in one stage instruction uses the result of an instruction that is under processing in previous stage, that is call Data dependence.

2. Data hazards occur when data is read, written or overwritten incorrectly.

3. Data hazards occur when instructions try to access data at wrong time before it is available of after it had been overwritten

4. Data hazard can also occur when two or more simultaneous instructions conflict

5. Data hazards can occur due to concurrent execution of instructions.

6. Data hazards occur due to unavailability of required piece of data due to failure of scheduling.

7. Data hazards can occur on instruction decode stage one tries to read a register value and exactly same time other instruction at Write back stage tries to write that register.

8. Data hazards occur because data is not available when and where it is required.

9. Data hazards also occur in pipelining because of execution of multiple instructions at the same time. Due to overlapping of instructions there are more chances of occurrence of hazards in pipeline as compare to non-pipelining where only one instruction is executed at a time. Once an instruction execution is completed then the other instruction execution starts.

   We need to handle data hazards effectively because ignoring data hazards may lead to race condition or race hazard.

3.2 RESOLVING PIPELINING COMPLICATIONS (DATA HAZARDS)

In order to resolve or remove data hazards effectively we can follow these effective ways.

1. Data hazards occur due to data dependence b/w instruction so in order to resolve data hazards we must eliminate data dependency

2. To elimination of hazards it is requires that some of the instructions that are in pipeline and ready for execution are allowed to proceed while other are delayed that are depend or produce unacceptable results.

3. Other way to eliminate data hazards is uses of pipeline stalls between some instructions. To block some instruction enter to pipeline while other later instruction complete, Stalls, NOP are inserted between two pipeline stages. Inserting stalls will remove dependency b/w the instructions but stalls can decrease the ideal performance.

4. Data hazards can be eliminated by using forwarding or bypassing.

5. Proper synchronization of instructions is required to deal with data hazards effectively

6. Data hazards can be removed easily by reordering of instruction but there is need to be careful about instruction re-ordering technique because reordering of instruction to avoid one type of hazard can become cause of other hazards as will.

7. False data dependencies b/w instructions can be removed by using register renaming technique.

3.3 Explanation with Examples:

In this stage we will discuss how to effectively remove data hazards in pipelined instruction with the help of examples.

1. Instructions Reordering:

Re-ordering technique is very simple and efficient way to remove some Data hazards that occur during execution instruction in pipelined. In many cases occur where reordering may affect the execution because some time order of the execution matter. Following is an example of instruction containing data hazard or data dependency

Data Hazard Example:

$$\mathrm{I}: \mathbf{R}8 \leftarrow \mathrm{R}7 \times \mathrm{R}7$$

$$\mathrm{II}: \mathrm{R}7 \leftarrow \mathbf{R}8 + \mathrm{R}3$$

$$\mathrm{III}: \mathrm{R}4 \leftarrow \mathrm{R}5 - \mathrm{R}6$$

There exists dependency b/w instruction I and instruction II because value of register R1 become invalid during pipelined execution and this dependency can be eliminated very efficiently by re-ordering technique as follows.

Re-ordered Instructions:

$$\mathrm{I}: \mathbf{R}8 \leftarrow \mathrm{R}7 \times \mathrm{R}7$$

$$\mathrm{II}: \mathrm{R}4 \leftarrow \mathrm{R}5 - \mathrm{R}6$$

$$\mathrm{III}: \mathrm{R}7 \leftarrow \mathbf{R}8 + \mathrm{R}3$$

Now each instruction can proceed independently without any dependency.

Data Hazard Example:

Here is another example that has data hazard we try to remove that hazard with re-ordering.

$$\mathrm{I}: \mathrm{LW}\ \mathbf{R}4,\ 0\ (\mathrm{R}6)$$

$$\mathrm{II}: \mathrm{ADD}\ \mathrm{R}1,\ \mathbf{R}4,\ \mathrm{R}3$$

$$\mathrm{III}: \mathrm{LW}\ \mathrm{R}2,\ 4(\mathrm{R}6)$$

In this example here exist dependence b/w first of two instructions because both are depended on register R4. Re-ordering can be done as follows to remove data hazard in the instructions.

Re-ordered Instructions:

$$\mathrm{I}: \mathrm{LW}\ \mathbf{R}4,\ 0\ (\mathrm{R}6)$$

$$\mathrm{II}: \mathrm{LW}\ \mathrm{R}2,\ 4(\mathrm{R}6)$$

$$\mathrm{III}: \mathrm{ADD}\ \mathrm{R}1,\ \mathbf{R}4,\ \mathrm{R}3$$

Later instruction execute between depended instructions now the instructions are free from data hazards or any kind of dependency.

2. Data Forwarding or bypassing:

Data forwarding also known as bypassing is an efficient way to solve data hazards in pipelined instruction execution. In the data forwarding or bypassing technique, normal processor is updated with special hardware. This technique the result of one stage is forward before complete execution of particular instruction. Where the result is require it is directly pass to that pipeline stage. In the forwarding there is no restriction from which stage data is transfer to next every next stage. For example data is passed from EX/MEM pipeline register to ALU stage of other pipeline or data is forward through instruction decode stage to instruction execution stage directly.

Example:

$$\text{I}: \text{ADD } \mathbf{R3},\ \text{R2},\ \text{R1}$$

$$\text{II}: \text{SUB R4}, \mathbf{R3}, \text{R5}$$

In this example here is a dependency between ADD and SUB instruction. In the 1st instruction register R3 is written and I the 2nd instruction register R3 is read here hazard occur if instruction II is complete before 1st instruction.

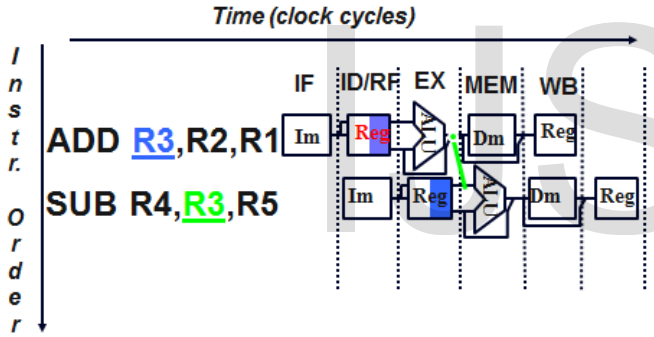

Diagram 7: Data Hazard Solution - Forwarding

First instruction will be in executing stage when second instruction will be decoded. Although the destination operand and source operand of both instruction is same. So remove this hazard data is forward from the EX/MEM pipeline register of ADD instruction to Sub instruction ALU stage.

3. Pipeline Stalls

Data hazards can also be removed by inserting stalls or alter the normal flow of execution. Stalls are inserted to skip one stall cycle and instruction is waiting until other same instruction or depended instruction complete or data hazard is chance is leave. The simplest way to fix the hazard is to stall the pipeline. Stalling involves blocking flow of instructions until the result is ready to be used.

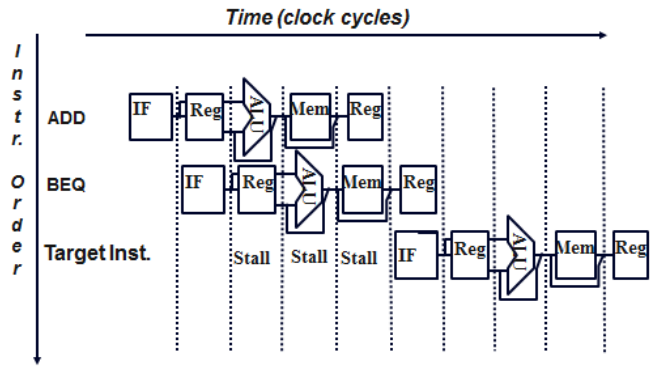

Diagram 8: Pipeline Stall

4. Compiler based Scheduling:

Compiler based scheduling is also a technique that is used to remove hazard efficiently. Compiler first detects the Hazards then analyzes the instruction where dependencies or data hazard occur, NOP or bubbles are inserted by the compiler between these instructions that has dependencies.

Example:

Here is an example of compiler based scheduling technique.

$$\text{I}: \text{LW } \mathbf{R}2,\ 0\ (\text{R4})$$

$$\text{II}: \text{MUL } \mathbf{R}2,\ \text{R2},\ \text{R2}$$

$$\text{III}: \text{SW } \mathbf{R}2,\ 0\ (\text{R4})$$

In this example many dependencies exist, compiler inserts bubbles or no operation instruction as follows.

$$\text{I}: \text{LW } \mathbf{R}2,\ 0\ (\text{R4})$$

$$\text{NOP}$$

$$\text{II}: \text{MUL } \mathbf{R}2,\ \text{R2},\ \text{R2}$$

$$\text{NOP}$$

$$\text{NOP}$$

$$\text{III}: \text{SW } \mathbf{R}2,\ 0\ (\text{R4})$$

So by inserting NOPs data hazards can be solved easily.

5. Register Renaming:

Pipelined issue of Data hazards can be solved through register renaming that is used to remove false data dependencies b/w instructions in running state. Register operands of instructions are renamed. Register renaming can reduce the impact of WAR and WAW dependencies. WAR and WAW both are data hazards.

3.4 DETECTION AND CORRECTION OF DATA HAZARDS

Detection of data hazard is not a complex as it is complex to remove or correction. The simplest technique is when the destination register and the source register of another instruction are same so there is a hazard or dependencies. The easy way to handle this situation is that the dependent instruction wait for other instruction on which it depends gets completed first. Stall or NOP is inserted between depended instructions. Data forwarding is another

technique that before completion of instruction Data is forwarded to next instruction from the stage where it is available. Hardware based solution to hazards is more efficient because compiler based solution to hazards is complex, expensive and inefficient.

3.5 COMPARISON

If we compare pipelining execution technique with data hazards and pipelining execution technique without data hazards then it will be clear to say that data hazards negatively impact pipelining performance and it is necessary to detect and correct data hazards for smooth and effective execution of pipeline. Comparison is given in the form of table as follows.

Table 1: Comparison of Pipelined execution with and without data hazards

With data hazards	Without data hazards
Data dependency b/w instructions	No data dependency
Data is read, written or overwritten incorrectly	Data is read, written and overwritten correctly.
Slow processing	Fast processing
Race condition can occur	No race condition
Performance reduction	Ideal Performance
Data is unavailable when and where it is needed.	Data is available when and where it is needed.

4 CONCLUSION

The purpose of our research paper is to give a complete analysis of the pipelining instruction processing. In the related work part we try to give complete discretion of the topic and with the help of example we explain the pipelined and without pipelined execution of work. Pipelining give us a way to efficient use of processor in various instruction, where number of instruction is large. Large instruction is quickly execute a pipelined order but there are many issues behind pipelined processing that effect the execution speed of the pipelined processor. I.e. data hazards, structural hazards, control hazards disturb the smooth execution of pipeline. Data hazards can disturb the normal executions of pipelining and can be considered as an obstacle that negatively affects pipelining performance. So if these data hazard cannot be handled properly it posts many performance issues on execution speed. Because pipelined executions mean speedup the processor if that goal is not complete then pipelining is failed. For that purpose we need to first detect data hazard because without detection no other process or correcting technique is used. There are many technique that we discuss in analysis, In order to remove data hazards. We can delay instruction for some time and insert stall/NOP between instructions that has any dependencies and reordering is best way to remove data hazards. Next technique that we discuss is data forwarding to next stage each stage of the pipeline without completion of first instruction. Other technique of hazard removing is hardware based; processor up gradation, a new type of hardware is added to processor that is called Compiler based solution. But it is complex and expensive than other technique like forwarding or bypassing. Stalling is simple way to fix data hazards but it can waste processing time by nothing while waiting for the result and some processing power wasted due to stalls but there are many ways available that reduce the stalls. Although it removes dependency b/w instructions and solves data hazards but ideal performance of pipelining may slow down. So we need to be careful about dealing with hazards because removing one kind of hazard may cause another hazard. Use of ineffective way to deal with hazards can also cause problems.

Acknowledgements

Our special praise to **Hazrat Muhammad (PBUH)** from the deepest core of my heart is forever a model of guidance and knowledge for the whole mankind. The very special entity Allah has brought into our lives, whose saying learns from cradle to grave, awakened the strong desire in us to undertake this course of study write up of this manuscript.

We would like to express our special thanks of gratitude to our course teacher **Dr. M. Ashraf Chughtai**. Who gave us the golden opportunity to do this wonderful paper on the topic "**How Data Hazards can be removed effectively**", which also helped us in doing a lot of Research and we came to know about so many new things. His knowledge about research helped us much in planning this paper. His critical suggestions helped shaped and consequently complete this paper. We really thankful to them

REFERENCE

[1] J. Flynn, Michael. Computer architecture: Pipelined and parallel processor design", Jones & Bartlett Learning, 1995.

[2] Godse, A.P. Godse, D.A. *"Computer Architecture- Advanced treatments for pipelining*", 2010.

[3] Matravers, J. *"Introduction to computer systems architecture and programming"* 2011.

[4] Abd-El-Barr, Mostafa, El-Rewini, Hesham. *"Advanced computer architecture and parallel processing* "2005.

[5] Hwang, Kai. Jotwani, Naresh. *"Advanced computer architecture": A beginner guide* 2nd Edition 2006.

[6] Stalling, Williams. "*Computer organization and architecture: dealing with pipeline hazards*", 8th Edition, 2006.

[7] Burrell, Mark. "*Fundamentals of computer architecture –Pipeline processing"* 2004.

[8] John P. Shen and Mikko H. Lipasti, Modern Processor Design: Fundamentals of Superscalar Processors, (2004), ISBN 0070570647.

[9] J. L. Hennessy et al. Computer Architecture: A Quantitative Approach, "Pipelining basics," Morgan Kaufmann May 2002.

[10] P. Dubey, M. Flynn. "Optimal Pipelining in J. Parallel and Distributed Computing".1990.

[11] V. Zyuban, D. Brooks, V. Srinivasan, M. Gschwind, P. Bose, P. Strenski, P. Emma Integrated "*Analysis of Power and Performance for Pipelined Microprocessors*". IEEE Transactions on Computer 2004.

[12] Patterson, David: Hennessy, John (2009). *Computer Organization and Design* 4th Edition, "*Pipelining Data Hazards*" Morgan Kaufmann ISBN 978-0-12-374493-7.

[13] Patterson, David: Hennessy, John (2011). *Computer Architecture: A Quantitative Approach* 5th Edition. *"Pipelining Performance Issues"* Morgan Kaufmann ISBN 978-0-12-383872-8.