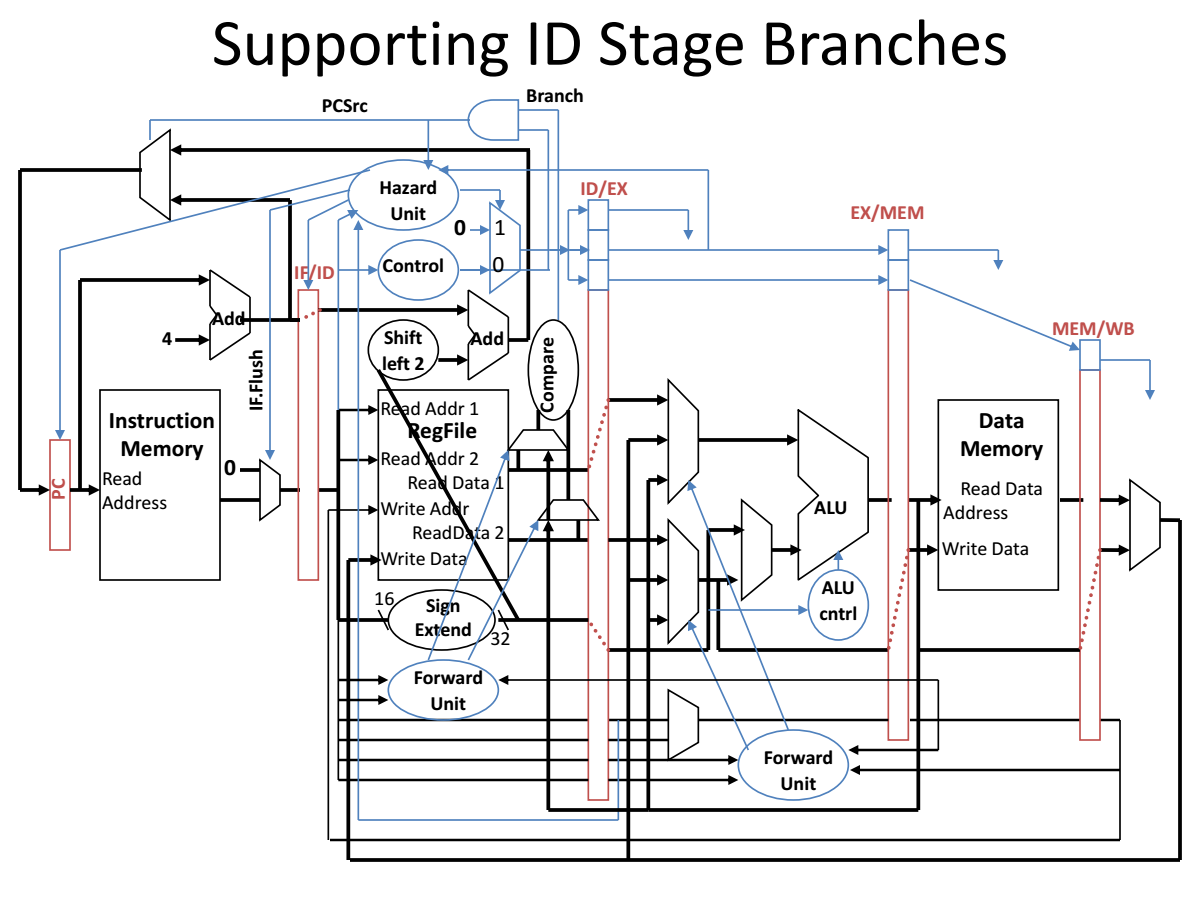
****

MIPS PIPLELINE

**PROCESSOR**

Designing for Hazards

**MIPS PIPELINE PROCESSOR *DESIGNING FOR DATA HAZARDS AND CONTROL HAZARDS***

*Group40*

**CONTENTS**

Online Resources

[**About the Author**](#_bookmark8)

[**Chapter 0 Reader’s and Instructor’s Guide 1**](#_bookmark9)

[PART ONE OVERVIEW 6](#_bookmark19)

[**Chapter 1 Introduction 6**](#_bookmark20)

* 1. [Organization and Architecture 7](#_bookmark22)
  2. [Structure and Function 8](#_bookmark25)
  3. [Key Terms and Review Questions 14](#_bookmark31)

[Chapter 2 Computer Evolution and Performance 15](#_bookmark33)

* 1. [A Brief History of Computers 16](#_bookmark36)
  2. [Designing for Performance 37](#_bookmark58)
  3. [Multicore, MICs, and GPGPUs 43](#_bookmark62)
  4. [The Evolution of the Intel x86 Architecture 44](#_bookmark64)
  5. [Embedded Systems and the ARM 45](#_bookmark67)
  6. [Performance Assessment 49](#_bookmark72)
  7. [Recommended Reading 59](#_bookmark81)
  8. [Key Terms, Review Questions, and Problems 60](#_bookmark83)

[PART TWO THE COMPUTER SYSTEM 65](#_bookmark86)

[**Chapter 3 A Top-Level View of Computer Function and Interconnection 65**](#_bookmark87)

* 1. [Computer Components 66](#_bookmark90)
  2. [Computer Function 68](#_bookmark93)
  3. [Interconnection Structures 84](#_bookmark107)
  4. [Bus Interconnection 85](#_bookmark110)
  5. [Point-To-Point Interconnect 93](#_bookmark117)
  6. [PCI Express 98](#_bookmark123)
  7. [Recommended Reading 108](#_bookmark133)
  8. [Key Terms, Review Questions, and Problems 108](#_bookmark134)

[Chapter 4 Cache Memory 112](#_bookmark137)

* 1. [Computer Memory System Overview 113](#_bookmark140)
  2. [Cache Memory Principles 120](#_bookmark147)
  3. [Elements of Cache Design 123](#_bookmark151)

**Online ResourceS**

|  |  |  |
| --- | --- | --- |
| **Site** | **Location** | **Description** |
| **Companion Website** | WilliamStallings.com/ ComputerOrganization | *Student Resources* link: Useful links and documents for students.  *Instructor Resources* link: Useful links and documents for instructors. |
| **Premium Content** | Click on *Premium Content* link at Companion Website or at  pearsonhighered.com/stallings and enter the student access code found on the card in the front of the book. | Online chapters, appendices, and other documents that supplement the book. |
| **Instructor Resource Center (IRC)** | Click on *Pearson Resources for Instructors* link at Companion Website or on *Instructor Resource* link at pearsonhighered.com/stallings. | Solutions manual, projects manual, slides, and other useful documents. |
| **Computer Science Student Resource Site** | ComputerScienceStudent.com | Useful links and documents for computer science students. |

**[PREFACE](#_bookmark0)**

We tried hardly to support all possible data hazards, control hazards and stalling conditions, and we hope that we succeed in that.

* + - **Data Hazards:** hazards occurs due to read after write dependencies.
    - **Control Hazards:** Control pipelined registers in case if the processor needs

Stalling or flushing the pipeline.

* + - **Branching:** Supporting static branch and assumed that the branch is not taken.
    - **Supported R-Formats Instructions :** add,sub,and,or,xor,nor,sll,srl,sra,slt,jr
    - **Supported I-Formats Instructions :** addi,andi,ori,xori,slti,lui,lw,sw,beq,bne
    - **Supported J-Formats Instructions :** j,jal
    - **Supporting signed and unsigned operands**
    - **I/O standards:** The only input to the MIPS Top Module is **Clk** signal.
    - **Test cases :** Support test cases with very hard situations to handle , you will

Find it in Test cases Document.

[**ABOUT THE AUTHOR**](#_bookmark0)

This project is made by a group of students(Group40) at 3rd year Computer and systems department students at Faculty of Engineering Ain Shams University.

This is a complete rewrite in Verilog HDL. The aim of this project is to simulate a pipelined MIPS processor dealing with various MIPS instructions.

This Project supporting data hazards, control hazards for LW, BEQ and R-Format instructions.

The HDL code supporting filling the memories (Instruction Memory and Data Memory) and register files from outside text files or filling it manually by the user by typing them inside their modules .These text files must be found inside the Project files directory under constant names.

There is also a GUI program with simple interface that can be simply used by any user.

GUI support converting ASSEMBLY language into machine code language, filling text files contain the data that will be stored in the Register file and the memories (Instruction Memory and Data Memory),and viewing the monitor of the internal test bench.

***Group Members***

*Abdelrahman Ibrahim Yassin*

*Abdelrahman Mahmoud*

*Abdallah Reda Abdallah*

*Diaa Ahmed Abdelzaher*

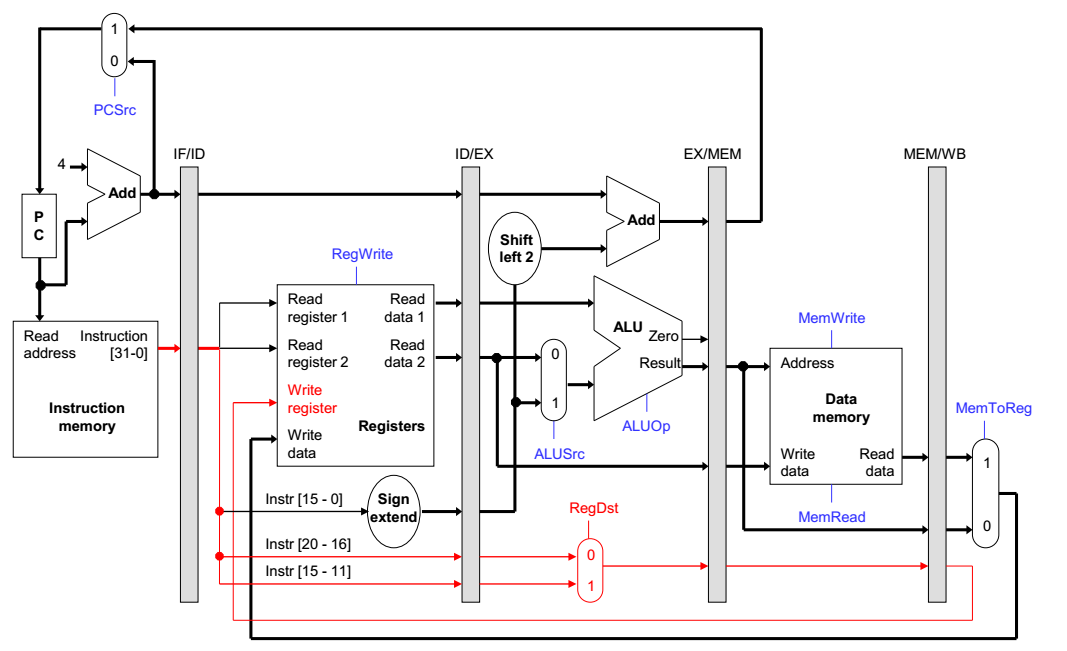
*Samuel Medhat* Farid

**Chapter 2: Pipelining**

* **Pipeline Register**
* We’ll add intermediate registers to our pipelined datapath too.
* There’s a lot of information to save, however. We’ll simplify our diagrams by drawing just one big pipeline register between each stage.
* The registers are named for the stages they connect.

IF/ID ID/EX EX/MEM MEM/WB

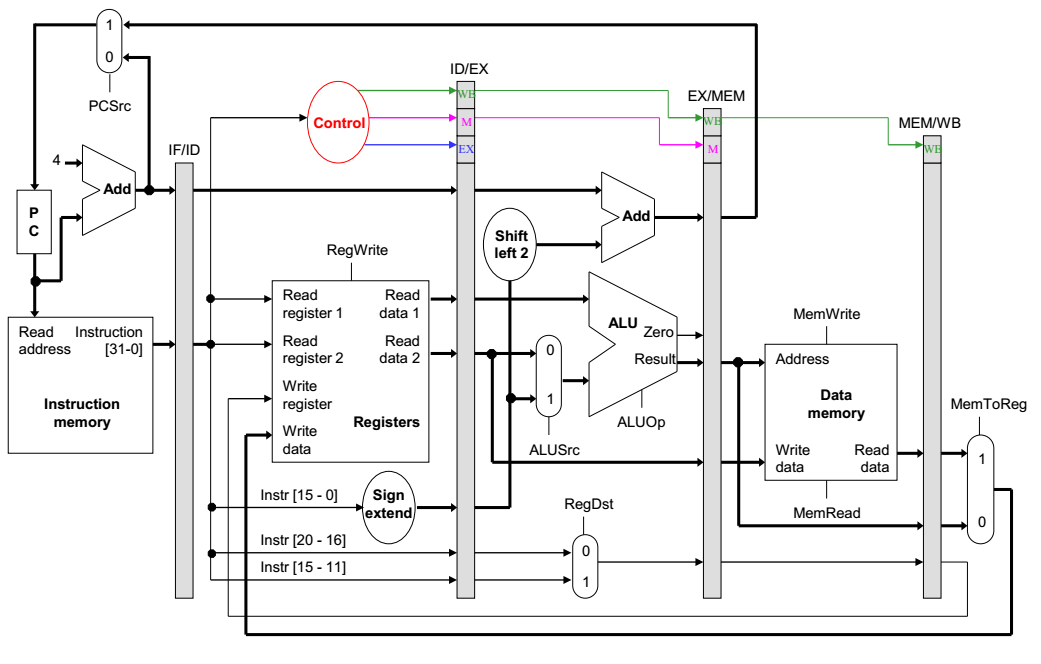
* No register is needed after the WB stage, because after WB the instruction is done.

****

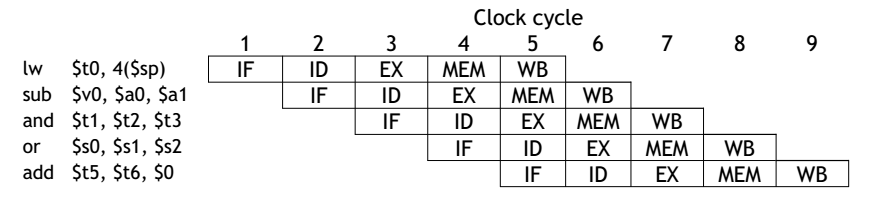
* **Control Lines**
* The control signals are generated in the same way as in the single-cycle processor—after an instruction is fetched, the processor decodes it and produces the appropriate control values.
* But just like before, some of the control signals will not be needed until some later stage and clock cycle.
* These signals must be propagated through the pipeline until they reach the appropriate stage. We can just pass them in the pipeline registers, along with the other data.
* Control signals can be categorized by the pipeline stage that uses them.

|  |  |  |  |
| --- | --- | --- | --- |
| Stage | Control signals needed | | |
| EX | ALUSrc | ALUOp | RegDst |
| MEM | MemRead | MemWrite | PCSrc |
| WB | RegWrite | MemToReg |  |

* **Here, the Pipelined data path and control**

****

* **So, the following instructions will be executed in pipeline like illustrated in the diagram**



* **Summary**
* The pipelined datapath combines ideas from the single and multicycle processors that we saw earlier.
  + It uses multiple memories and ALUs.
  + Instruction execution is split into several stages.
* Pipeline registers propagate data and control values to later stages.
* The MIPS instruction set architecture supports pipelining with uniform instruction formats and simple addressing modes.
* Next chapter , we’ll start talking about Hazards.