

COMPUTER ARCHITECTURE

ENCS4370

Course Project

Instructor: Ayman Hroub

Section: 3

Team Members:

Momen Assaf 1191529

Sami Abu Rmaileh 1192325

Yazan Yousef1191706

2/7/2023

**Contents**

[Table of Figures: 3](#_heading=h.30j0zll)

[**Theory:** 4](#_heading=h.1fob9te)

[Datapath: 4](#_heading=h.3znysh7)

[Pipelining Architecture: 4](#_heading=h.2et92p0)

[**Objectives:** 4](#_heading=h.tyjcwt)

[Instruction Fetch (IF): 4](#_heading=h.3dy6vkm)

[Instruction Decode (ID): 4](#_heading=h.1t3h5sf)

[Execution (Ex): 4](#_heading=h.4d34og8)

[Memory (Mem): 5](#_heading=h.2s8eyo1)

[Write Back (WB): 5](#_heading=h.17dp8vu)

[**Project Specification:** 6](#_heading=h.26in1rg)

[Instruction set Architecture: 6](#_heading=h.lnxbz9)

[**Instruction Types and Formats** 6](#_heading=h.35nkun2)

[R-type instruction: 6](#_heading=h.1ksv4uv)

[I-Type (Immediate Type) 7](#_heading=h.2jxsxqh)

[J-Type (Jump Type) 7](#_heading=h.3j2qqm3)

[S-Type (Shift Type) 7](#_heading=h.4i7ojhp)

[**Instructions’ Encoding** 8](#_heading=h.1ci93xb)

[Control Unit 9](#_heading=h.2bn6wsx)

[**Design and Implementation:** 10](#_heading=h.qsh70q)

[Components: 10](#_heading=h.3as4poj)

[Register file: 10](#_heading=h.1pxezwc)

[Instruction Memory: 10](#_heading=h.49x2ik5)

[ALU: 11](#_heading=h.2p2csry)

[Data Memory: 11](#_heading=h.3o7alnk)

[Next PC: 12](#_heading=h.23ckvvd)

[**Total data path design** 12](#_heading=h.ihv636)

[**Test Cases:** 12](#_heading=h.1hmsyys)

# Table of Figures:

[Figure 1 Pipelined cycle data path 5](#_heading=h.3rdcrjn)

[Figure 2 R-type instruction 6](#_heading=h.44sinio)

[Figure 3 I-Type (Immediate Type) 7](#_heading=h.z337ya)

[Figure 4 J-Type (Jump Type) 7](#_heading=h.1y810tw)

[Figure 5 S-Type (Shift Type) 7](#_heading=h.2xcytpi)

[Figure 6 Instructions supported. 8](#_heading=h.3whwml4)

[Figure 7 Register file.](about:blank) 10

[Figure 8 Instruction Memory.](about:blank) 10

[Figure 9 Alu operation](about:blank) 11

[Figure 10 ALU. 11](#_heading=h.147n2zr)

[Figure 11 Data Memory.](about:blank) 11

[Figure 12 Next PC.](about:blank) 12

[Figure 13 Data path 12](#_heading=h.32hioqz)

[Figure 14 Fetch Cycle 12](#_heading=h.41mghml)

[Figure 15 Decode Cycle 1](#_heading=h.2grqrue)3 Figure 16 Execute Cycle 13 Figure 17 Memory Cycle 14

Figure 18 WB Cycle 14

# **Theory:**

## Datapath:

Is a collection of functional units that perform data processing tasks. A computer system's CPU (central processing unit) is made up of data pathways and a control unit. Joining more than one data path can also result in a longer data path. Multiplexers are used to connect one to another.

## Pipelining Architecture:

Pipelining is a technique that allows numerous instructions to be executed at the same time. The pipeline is separated into stages, which connect to form a pipe-like structure. Instructions enter at one end and exit at the other. Pipelining boosts total instruction throughput.

# **Objectives:**

To design and verify a simple RISC processor in Verilog. 5-stage pipelined processor (fetch, decode, execute, memory access, and write back).

## Instruction Fetch (IF):

This level oversees retrieving instructions from memory. It contains the program counter (PC), which stores the address of the next instruction to be retrieved. After each fetch, the PC is normally increased by the size of the instruction. The retrieved instruction is subsequently sent to the decoding step.

## Instruction Decode (ID):

The fetched instruction is decoded at this stage to identify the following stages in the execution. It entails assessing the instruction to determine the operations to be done, data sources and destinations, and any control signals necessary for the instruction to be executed. The control unit creates the control signals required to coordinate the succeeding steps.

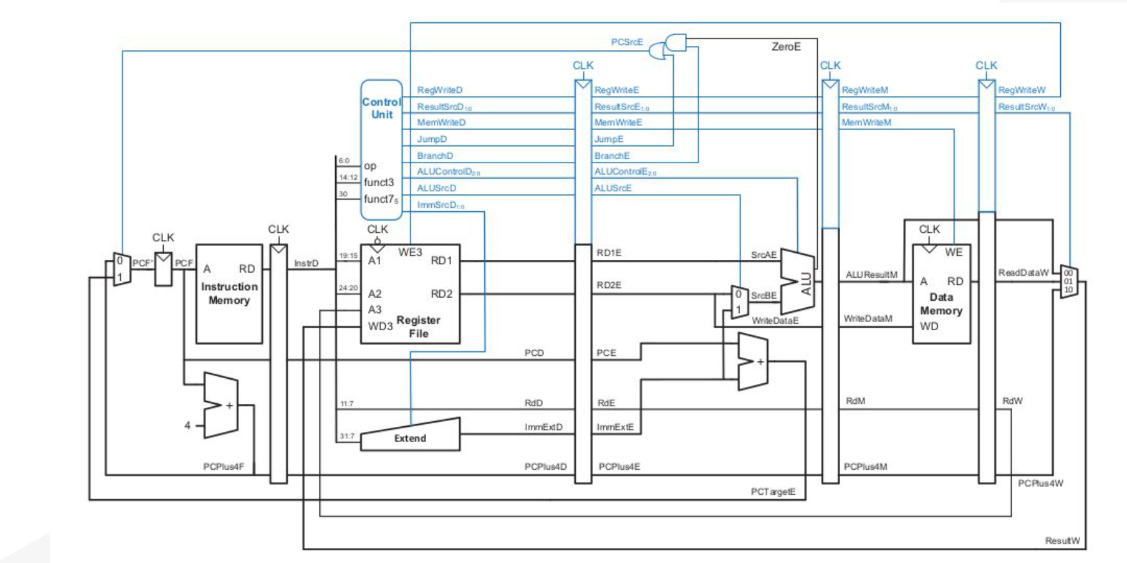
## Execution (Ex):

The execution step is responsible for carrying out the computation or action provided by the decoded instruction. This stage may include a variety of activities such as arithmetic computations, logical operations, and memory operations. The operands for the operation are retrieved from registers or memory, and the ALU executes the stated computation. The outcome is then passed on to the next level.

## Memory (Mem):

Certain operations, such as load and store instructions, need data memory access. Data memory operations are carried out at this level. The relevant data is obtained from memory for load instructions, whereas data is written to memory for store instructions. The information might be gathered through registries or the outcome of the preceding stage.

## Write Back (WB):

The pipeline's last step oversees sending back the results of the executed instruction to the proper destination. The outcome of the execution or memory stage is normally recorded back to the register file. This stage changes the registers with the calculated values so that later instructions may access them.

*Figure 1 Pipelined cycle data path*

From the above figure we see that A pipeline in a processor is a mechanism that allows numerous instructions to be performed simultaneously, increasing instruction throughput and overall speed. It splits the instruction execution process into numerous phases, each of which oversees a distinct duty. As one instruction advances through the pipeline, another instruction may arrive and begin execution in a different step, resulting in several instructions being executed at the same time.

# **Project Specification:**

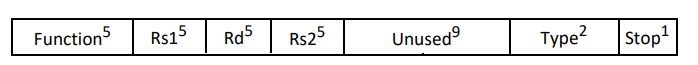
## Instruction set Architecture:

* The instruction size is 32 bits, so the word size is 4 bytes.
* 32 32-bit general-purpose registers: from R0 to R31.
* A special purpose register for the program counter (PC).
* It has a stack called control stack which saves the return addresses.
* Stack pointer (SP), another special purpose register to point to the top of the control stack. SP holds the address of the empty element on the top of the stack. For simplicity, you can assume a separate on-chip memory for the stack, and the initial value of SP is zero.
* Four instruction types (R-type, I-type, J-type, and S-type).
* The processor’s ALU has an output signal called “zero” signal, which is asserted when the result of the last ALU operation is zero.
* Separate data and instructions memories

# **Instruction Types and Formats**

As previously stated, this ISA contains four instruction formats: R-type, I-type, J-type, and S-type. These four categories share the following fields:

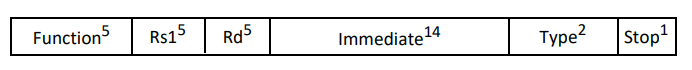
# R-type instruction:



*Figure 2 R-type instruction*

* 5-bit Rs1: first source register.
* 5-bit Rd: destination register.
* 5-bit Rs2: second source register.
* 9-bit unused.
* 2-bit instruction type.
* Stop bit.
* 5-bit function

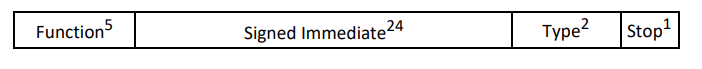
# I-Type (Immediate Type)



*Figure 3 I-Type (Immediate Type)*

* 5-bit Rs1: first source register
* 5-bit Rd: destination register
* 14-bit immediate: unsigned for logic instructions and signed otherwise.
* Stop bit.
* 2-bit instruction type.
* 5-bit function.

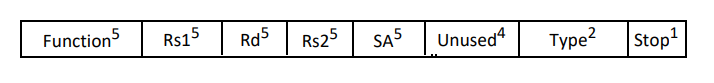
# J-Type (Jump Type)



*Figure 4 J-Type (Jump Type)*

* 24-bit signed immediate.
* 5-bit function.
* 2-bit instruction type.
* Stop bit.

# S-Type (Shift Type)

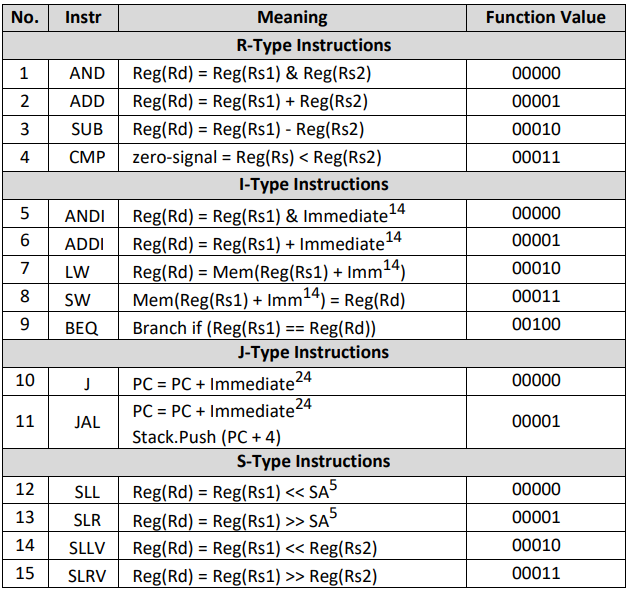


*Figure 5 S-Type (Shift Type)*

* 4-bit unused
* 5-bit SA
* 5-bit Rs2
* 5-bit Rd
* 5-bit Rs1
* 5-bit function.
* 2-bit instruction type.
* Stop bit.

# **Instructions’ Encoding**

To keep things simple, we must simply implement a subset of this processor's ISA. The table below summarizes the many instructions you must follow. It displays their type, function value, and meaning in RTN (Register Transfer Notation). Although the instruction set has been decreased, it is still sufficient to construct effective applications.



*Figure 6 Instructions supported.*

# Control Unit

*Table 1 Control Unit*

| OP | PC\_src | ExtOp | ALUSrc | Mem\_W | Mem\_R | WB | RegW |
| --- | --- | --- | --- | --- | --- | --- | --- |
| R-Type | 0 | X | 0 | 0 | 0 | 0 | 1 |
| ANDI | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| ADDI | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| LW | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| SW | 0 | 1 | 1 | 1 | 0 | X | 0 |
| BEQ | 0 or 2 | X | 0 | 0 | 0 | X | 0 |
| J | 1 | X | X | 0 | 0 | X | 0 |
| JAL | 1 | X | X | 0 | 0 | X | 0 |
| S-Type | 0 | X | 2 | 0 | 0 | 0 | 1 |

*Pc\_src[1] = J*

*Pc\_src[2] = Branch.zero*

*Extop = (ADDI + LW + SW)*

*ALUSrc[0] = R\_Type*

*ALUSrc[2] = S-type*

*Mem\_W = SW*

*Mem\_R = LW*

*WB = LW*

*RegW = ~(SW + BEQ + J +JAL)*

# **Design and Implementation:**

## Components:

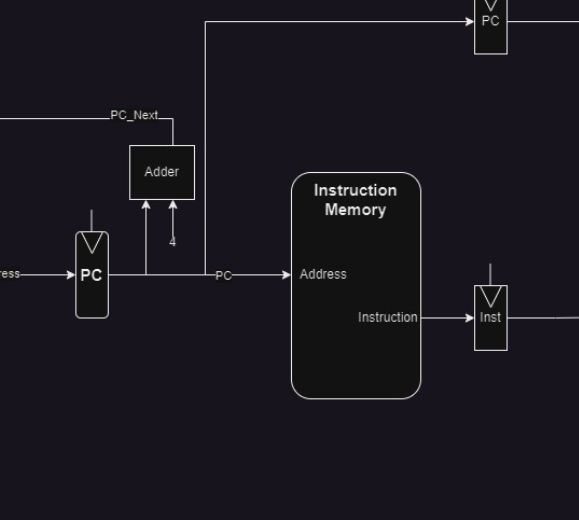
## Register file:

A Register File is an important component of computer architecture that stores data in a series of registers. It enables the processor to swiftly read and write to these registers, which are required for arithmetic, logical, and data transfer tasks.A computer screen shot of a computer

Description automatically generated

In this scenario, we have an 8-register Register File, with each register having a width of 32 bits. The number and breadth of registers might vary based on the system's design and requirements.

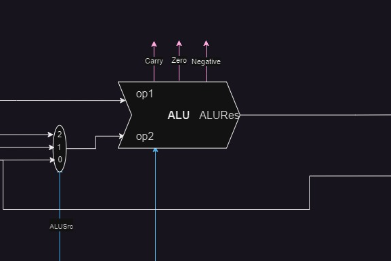
## Instruction Memory:

An Instruction Memory is a computer system component that holds software instructions. It is a memory unit that is specially intended to hold the instructions that the processor must execute.

The Direction Memory is arranged as a series of memory locations, each of which stores a single instruction. The size of each memory location is determined by the CPU architecture's instruction format. In a 32-bit instruction set architecture, for example, each memory location will be 32 bits wide.

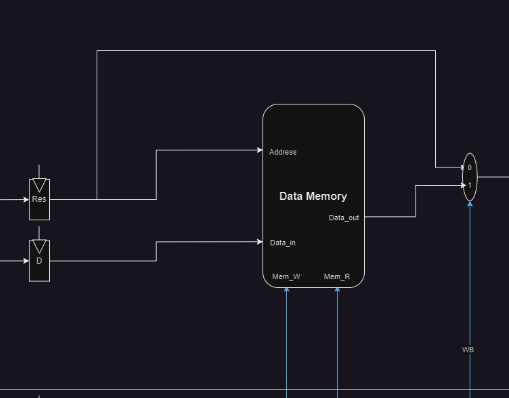
## ALU:

A digital combinational circuit that performs arithmetic and bitwise operations on binary integers in computing. There are floating-point units that works with floating-point numbers. In our CPU it will get 2 inputs of 32 bits and one output of 32 bits with ALU control signal to choose which operation to perform and some flags.



*Figure 10 ALU.*

## Data Memory:

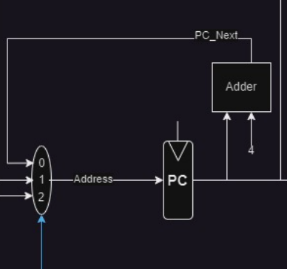
Data Memory is a component of a computer system that stores data utilized by the CPU during program execution. It is also known as Data Cache or Data RAM. It gives the processor a place to read and write data values.

Data Memory is structured as a collection of memory regions, each of which may hold a specific quantity of data. The architecture and design needs of the system define the size of each memory location. 8 bits (1 byte), 16 bits (2 bytes), 32 bits (4 bytes), and even greater amounts are common.

The CPU can save and retrieve data values from the Data Memory using memory addresses. MemRead: enables output on Data out Address selects the word to put on Data out.

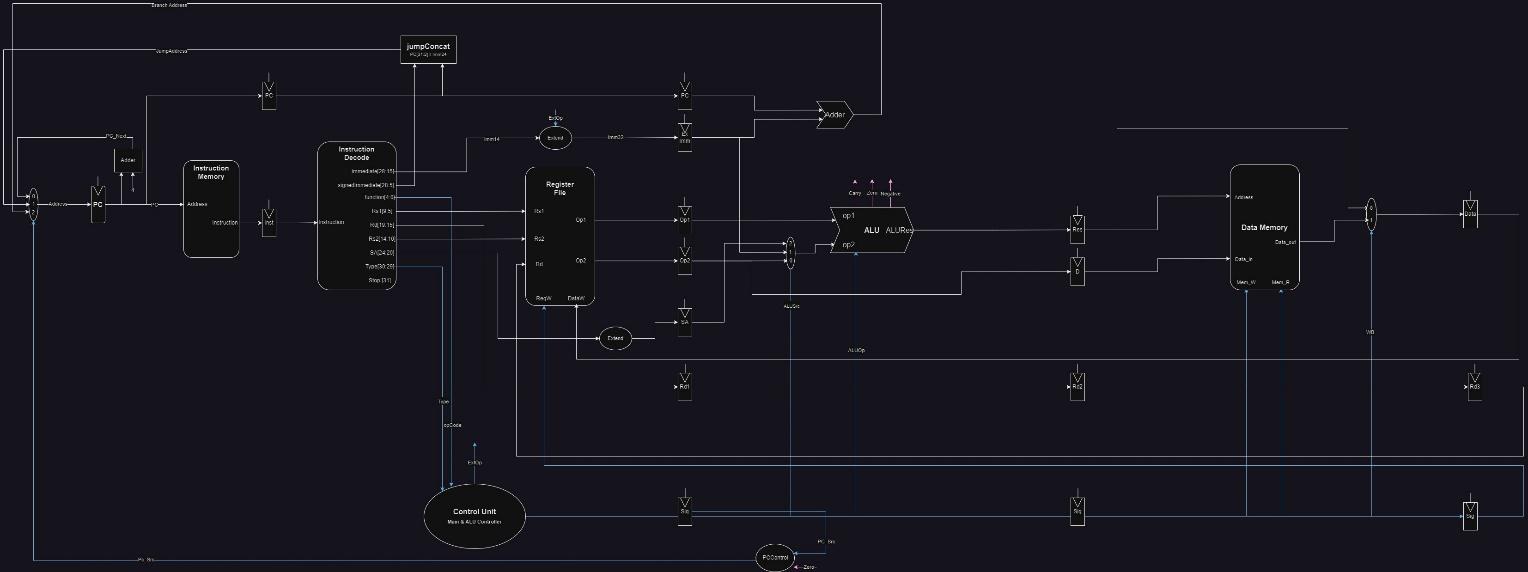
MemWrite: enables writing of Data in Address selects the memory word to be written.

## Next PC:

circuit that determines and calculates the next program counter for the processer and makes aware of some conditions like Branch or jump conditions.

responsible for increasing the PC by 4 at each clock cycle. linked to an added that adds 4 to the input, which contains the address of the next instruction to be.

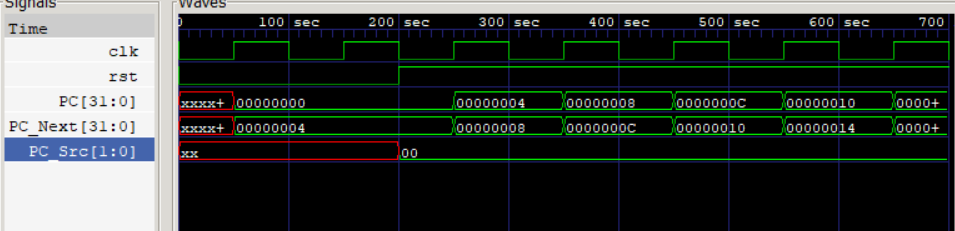
# **Total data path design**



*Figure 13 Data path*

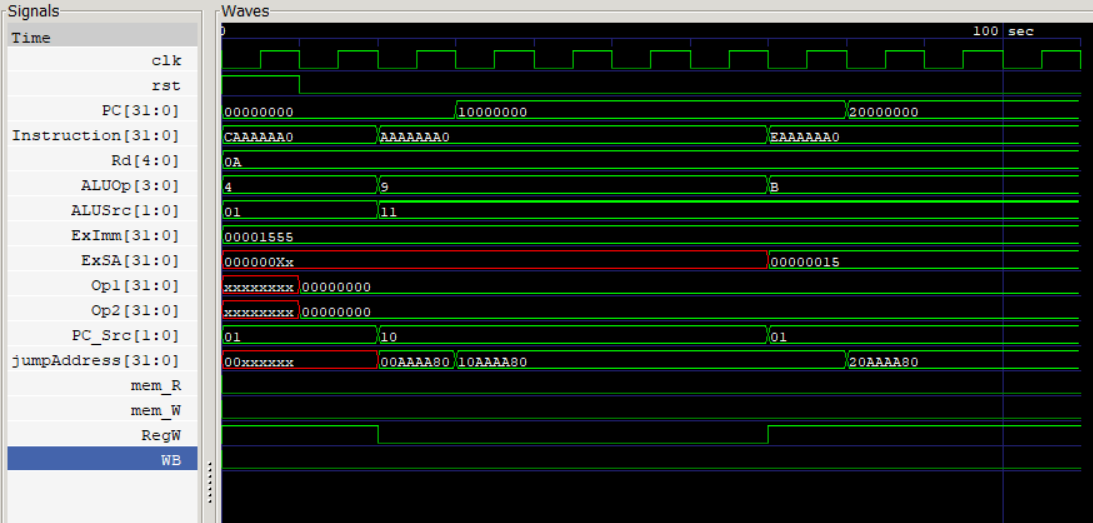
# **Test Cases:**

1. Fetch Cycle



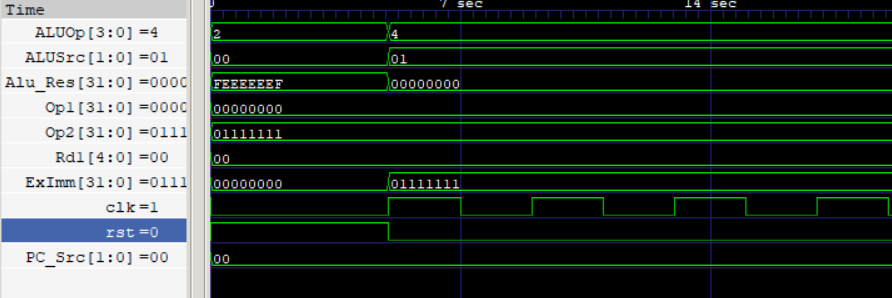
*Figure 14 Fetch Cycle*

1. Decode Cycle



*Figure 15 Decode Cycle*

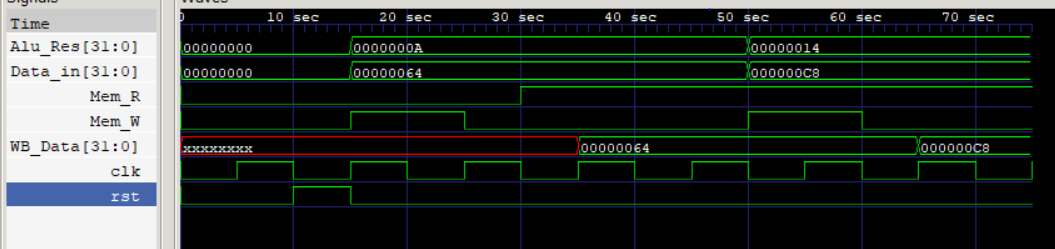
1. Execute Cycle



*Figure 16 Execute Cycle*

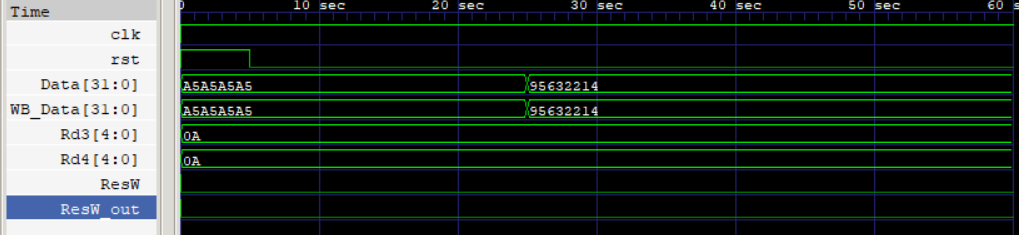
# 

1. Memory Cycle



*Figure 17 Memory Cycle*

1. WriteBackCycle



*Figure 17 WriteBack Cycle*

# **Recourses:**

1. Computer Architecture: A Quantitative Approach John L. Hennessy and David A. Patterson, Morgan Kaufmann,6 th Edition, 2019
2. Modern Processor Design
3. [277-148793598984-88.pdf (digitalxplore.org)](https://www.digitalxplore.org/up_proc/pdf/277-148793598984-88.pdf)
4. [Classic RISC pipeline - Wikipedia](https://en.wikipedia.org/wiki/Classic_RISC_pipeline)
5. [Introduction of ALU and Data Path - GeeksforGeeks](https://www.geeksforgeeks.org/introduction-of-alu-and-data-path/)