

COMPUTER ARCHITECTURE

ENCS4370

Course Project

**Easy Park**

Instructor: Ayman Hroub

Section: 3

Team Members:

Momen Assaf 1191529

Sami Abu Rmaileh 1192325

Yazan Yousef1191706

2/7/2023

**Contents**

[**Table of fingers:** 3](#_Toc139129619)

[**Theory:** 3](#_Toc139129620)

[Datapath: 3](#_Toc139129621)

[Pipelining Architecture: 3](#_Toc139129622)

[**Objectives:** 4](#_Toc139129623)

[Instruction Fetch (IF): 4](#_Toc139129624)

[Instruction Decode (ID): 4](#_Toc139129625)

[Execution (Ex): 4](#_Toc139129626)

[Memory (Mem): 4](#_Toc139129627)

[Write Back (WB): 4](#_Toc139129628)

[**Project Specification:** 5](#_Toc139129629)

[Instruction set Architecture: 5](#_Toc139129630)

[**Instruction Types and Formats** 6](#_Toc139129631)

[R-type instruction: 6](#_Toc139129632)

[I-Type (Immediate Type) 6](#_Toc139129633)

[J-Type (Jump Type) 7](#_Toc139129634)

[S-Type (Shift Type) 7](#_Toc139129635)

[**Instructions’ Encoding** 8](#_Toc139129636)

# **Table of fingers:**

[Figure 1 Pipelined cycle data path 4](#_Toc139129572)

[Figure 2 R-type instruction 5](#_Toc139129573)

[Figure 3 I-Type (Immediate Type) 6](#_Toc139129574)

[Figure 4 J-Type (Jump Type) 6](#_Toc139129575)

[Figure 5 S-Type (Shift Type) 6](#_Toc139129576)

[Figure 6 Instructions supported. 7](#_Toc139129577)

# **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):

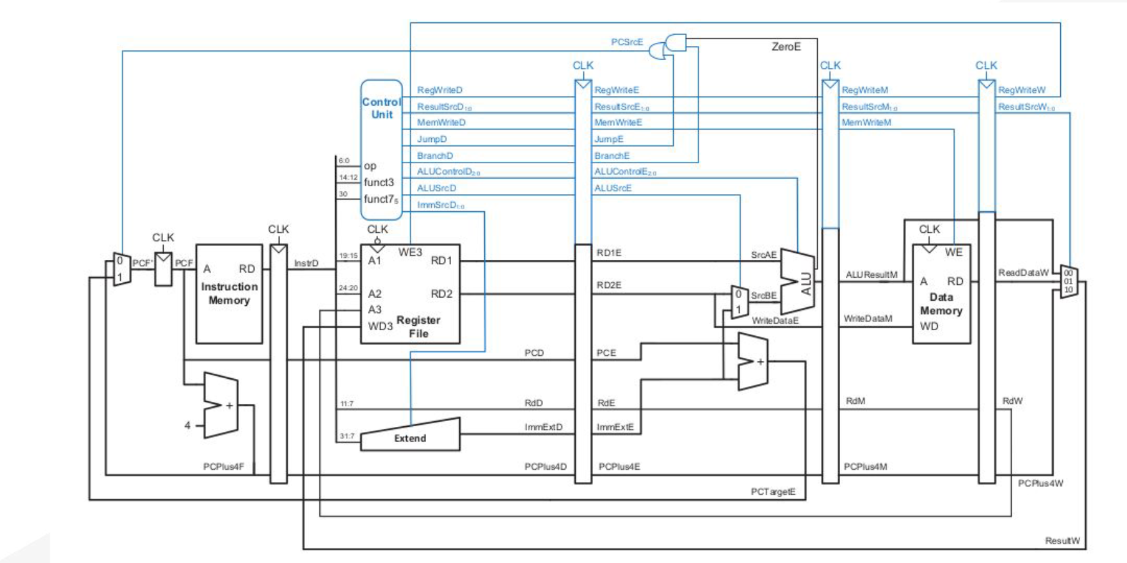
The pipeline's last step is in charge of sending back the results of the executed instruction to the proper destination. The outcome of the execution or memory stage is normally recorded back to the register file. This stage changes the registers with the calculated values so that later instructions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

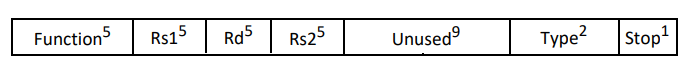


Figure 2 R-type instruction

* 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)

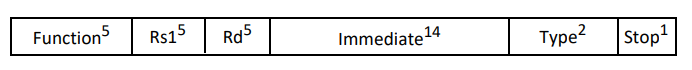


Figure 3 I-Type (Immediate Type)

* 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)

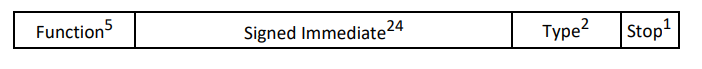


Figure 4 J-Type (Jump Type)

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

# S-Type (Shift Type)

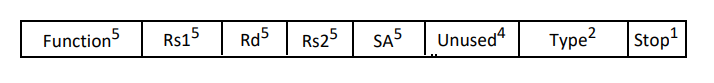


Figure 5 S-Type (Shift Type)

* 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.

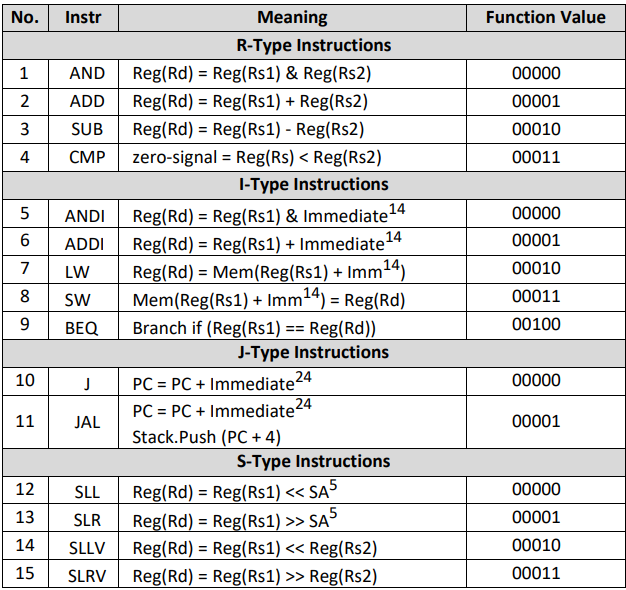
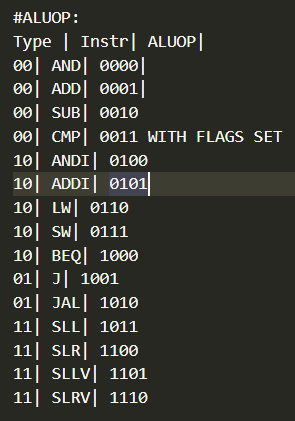


Figure 6 Instructions supported.

## Control Unit

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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)