Department of Electrical and Computer Engineering

ENCS4370| Computer Architecture

The 2nd Project: Single cycle Processer Build and Design

Prepared By:

Shahd Qatre 1200431

Sondos Shahin 1200166

Motaz Al shekh 1201737

Instructor:

Dr.Ayman Hroub & Dr. Aziz Qaruish

Section **:** 2&1

Date**:** 29/1/2024

Abstract

The goal of this project is to develop and test a small RISC CPU using Verilog. The CPU includes 32 general-purpose registers, a program counter (PC), a stack pointer (SP), a return address control stack, and a 32-bit instruction size. Together with an ALU, four different instruction kinds (R-type, I-type, J-type, and S-type) are supported. Instruction and data memory are kept separate. The RTL architecture has a datapath and a control path. There are three possible design options: single-cycle, multi-cycle, and five-stage pipelined processors. The team opted to construct a single processor. Code sequences run in the intended processor and a testbench are used for verification.

Catalog

Table Of Figure

Design Specification:

➢ Single cycle processer

The team chose to implement and design a single-cycle processor, which was chosen because it is easier than pipelines and multi to implement. There was not enough time . single-cycle processor, instruction execution is into one stages,take one clock cycle.

Components

Instruction Memory

The memories in the implementation are separated into two parts, instruction memory, and data memory. This was done to solve some conflicts, such as, one instruction might be fetching the instruction from the memory and the other instruction is loading/storing some data from/to the memory, so in order to obey the isolation principle, they need to be separated into different memory elements.

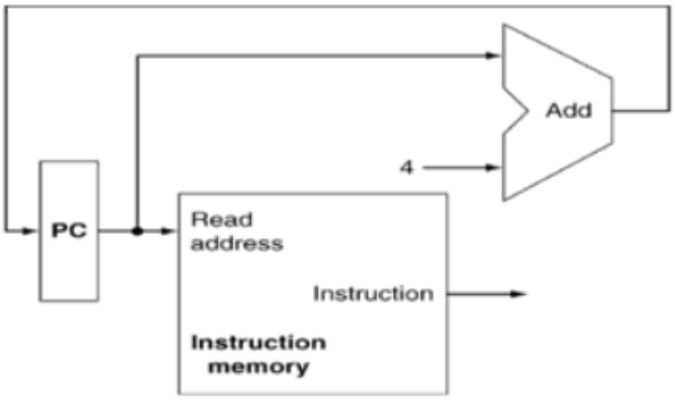

Figure 1-instruction memory

Data Memory

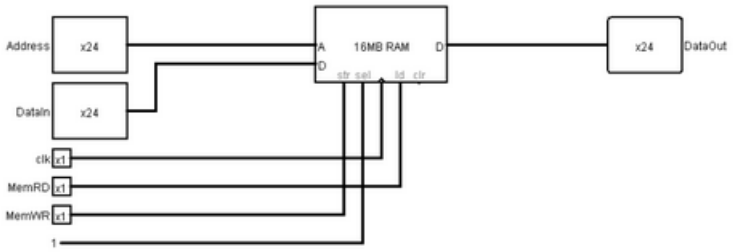

Figure 2-Data memory

Figure 2 shows the implementation of data memory using a Random Access Memory, to support read and write, which is done through and address bus to determine where to write or read, and a data bus to determine the data to write in case of writing to memory.

Register File

A register file is an array of processor registers in a central processing unit (CPU). The instruction set architecture of a CPU will almost always define a set of registers which are used to stage data between memory and the functional units on the chip. The register file is part of the architecture and visible to the programmer, as opposed to the concept of transparent caches. In simpler CPUs, these *architectural registers* correspond one-for-one to the entries in a physical register file (PRF) within the CPU. More complicated CPUs use register renaming, so that the mapping of which physical entry stores a particular architectural register changes dynamically during execution.

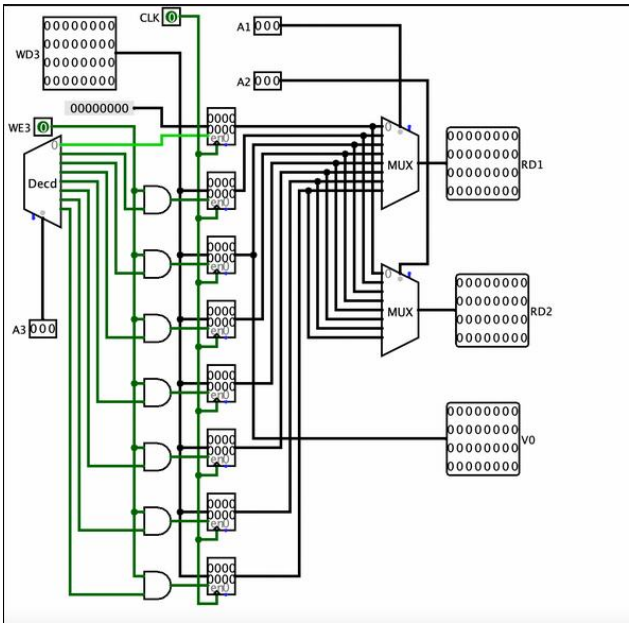

Figure 3-Register File

Extender

The extender was 16-bit immediate.

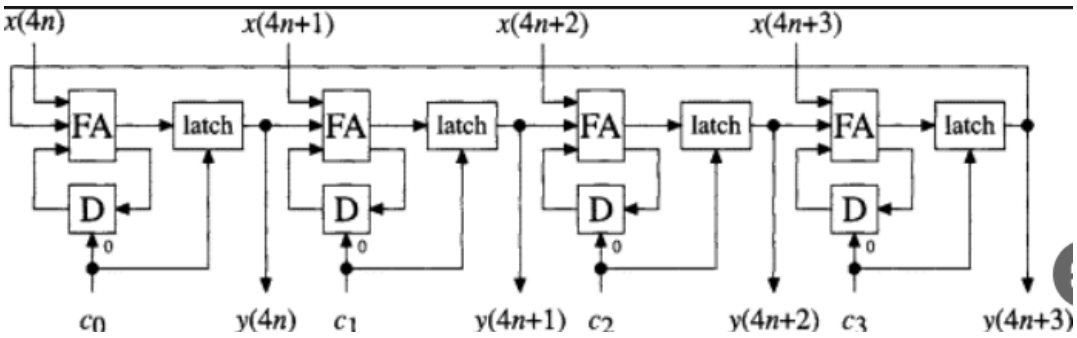

Figure 4-Extender

Arithmetic Logic Unit (ALU)

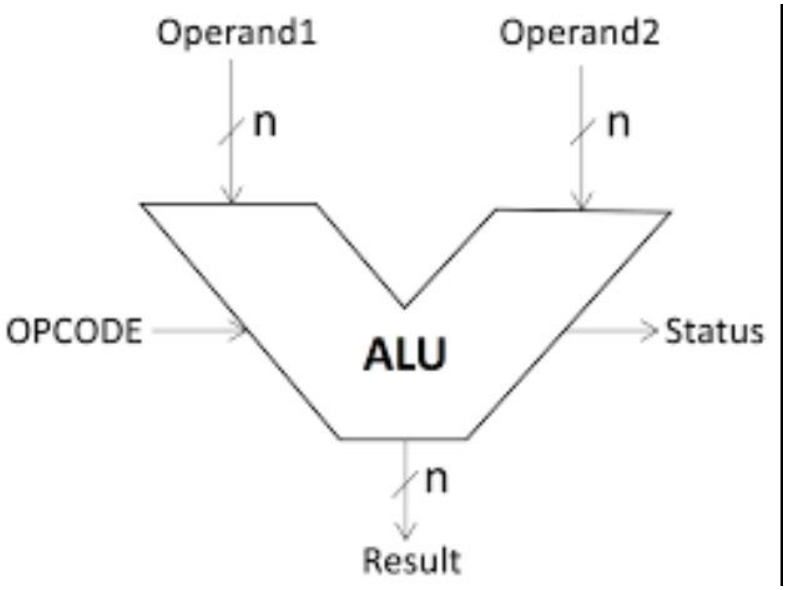

Figure 5-ALU

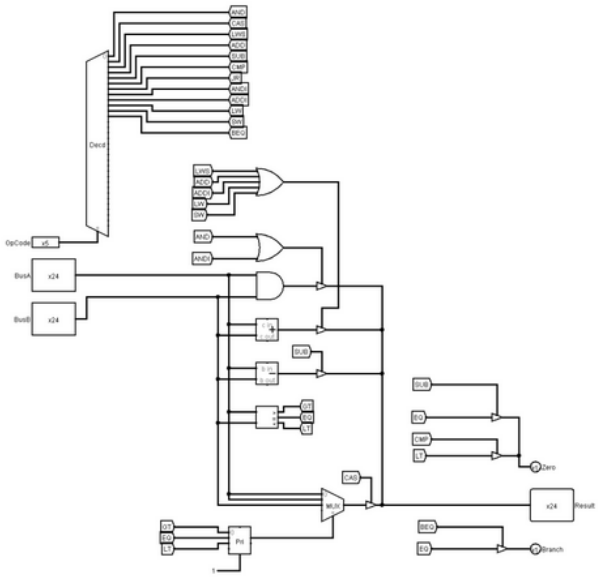

Figure 6-ALU implementation

Figure 6 shows the implementation of the ALU, where all components are working without enable, but the opcode decides which result to go out to the result bus.

Control Unit

Truth Table

Name	function	Regwr	AluOp	AluSrc	Ext	Wresult	Bw2	MemRd	MemWr	PC-CONTROL	DMadd	SP	DMdata
R-type instruction													
AND	6'b000000	1	0	0	X	0	x	x	x	0	x	x	X
ADD	6'b000001	1	1	0	X	0	x	x	x	0	x	X	X
SUB	6'b000010	1	2	0	x	0	x	x	x	0	x	x	X
I-type instruction													
ANDI	6'b000011	1	0	1	0	0	X	x	x	0	x	X	X
ADDI	6'b000100	1	1	1	0	0	x	x	x	0	x	x	X
LW	6'b000101	1	1	1	1	1	0	1	0	0	0	x	X
LW.poi	6'b000110	1	1	1	1	1	1	1	0	0	0	x	X
SW	6'b000111	0	1	1	1	X	x	0	1	0	0	x	0
BGT	6'b001000	0	1	1	1	X	x	X	x	1/0	x	x	X
BLT	6'b001001	0	1	1	1	X	X	x	x	1/0	X	x	X
BEQ	6'b001010	0	1	1	1	X	x	x	x	1/0	X	X	X
BNE	6'b001011	0	1	1	1	x	x	x	x	1/0	x	X	x
I-type instruction													
JMP	6'b001100	0	X	X	X	X	X	X	X	2	X	X	X
CALL	6'b001101	0	X	X	X	X	X	0	1	2	1	X	1
RET	6'b001110	0	X	X	X	X	X	1	0	3	1	X	X
S-type instruction													
PUSH	6'b001111	0	X	X	X	X	X	0	1	0	1	0	0
POP	6'b010000	0	X	X	X	X	X	1	0	0	1	1	X

Figure 7--controls-truth table

Boolean Expression

Regwr : AND+ADD+SUB+ANDI+ADDI+LW.LW.poi

AliOp: if = 2 = SUB

Else =ADD+ADDI+LW+LWpoi+SW+BGT+BLT+BEQ+BNQ

AluSrc: ANDI+ADDI+LW+LW.poi+SW+BGT+BLT+BEQ+BNE

EXT: LW+LW.poi+SW+BGT+BLT+BEQ+BNE

Wresult: LW+LW.poi

Bw2 : LW.poi

MemRd : LW.poi + RET + POP

MemWr : LW+LW.poi+SW+CALL

PC : IF = 3 = RET

ELSE If 2 = CALL+JMP

ELSE = BGT+BLT+BEQ+BNQ

DMadd : CALL+ RET+ PUSH+ POP

SP : POP

DMdata : CALL

Control unit data path

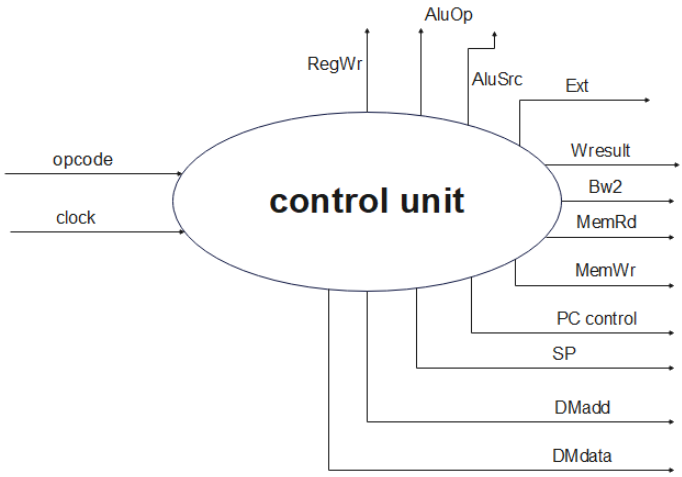

Figure 8- Control unit data path

complete description of the control signals

Regwr : We use it if we want to write or read on the register file.

AliOp:To specify the type of operation (such as addition, subtraction...).

AluSrc: To specify the input to the alu if it the output to the register file or the immedate .

EXT: choose if it sign or unsign .

Wresult:To specify if we need to write on the register file the data out or alu result.

Bw2 : specify what I need to write in bus2.

MemRd : specify if I need to read from memory.

MemWr : specify if I need to write from memory.

PC : To specify the input to the instruction memory (pc+4,bjt…..).

DMadd : To specify the data memory address (sp output or alu result).

SP : To specify the input to the stack

DMdata : To specify the data memory data(pc or rd).

➢ Single-Cycle Disadvantages & Advantages

◆ Uses the clock cycle inefficiently – the clock cycle must be timed to accommodate the slowest instruction.

✧ This would be especially problematic for more complex instructions like floating point multiply.

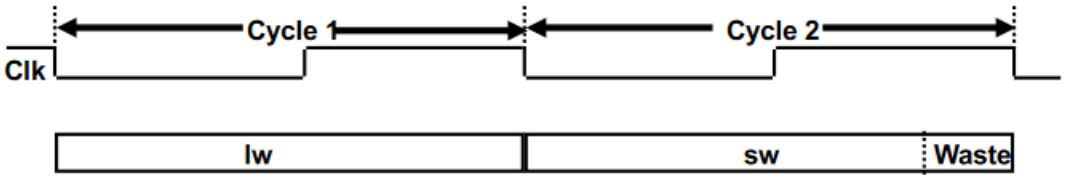

Figure 9-Adv & Dis

◆ May be wasteful of area. Some functional units (e.g., adders, memory) must be duplicated since they can not be shared during a clock cycle.

◆ However, the single-cycle implementation is simple and easy to understand.

➢ Single cycle datapath

Single Datapaths is equivalent to the original single-cycle datapath The data memory has only one Address input. The actual memory operation can be determined from the MemRead and MemWrite control signals. There are separate memories for instructions and data. There are 4 adders for PC-based computations and one ALU. The control signals are the same.

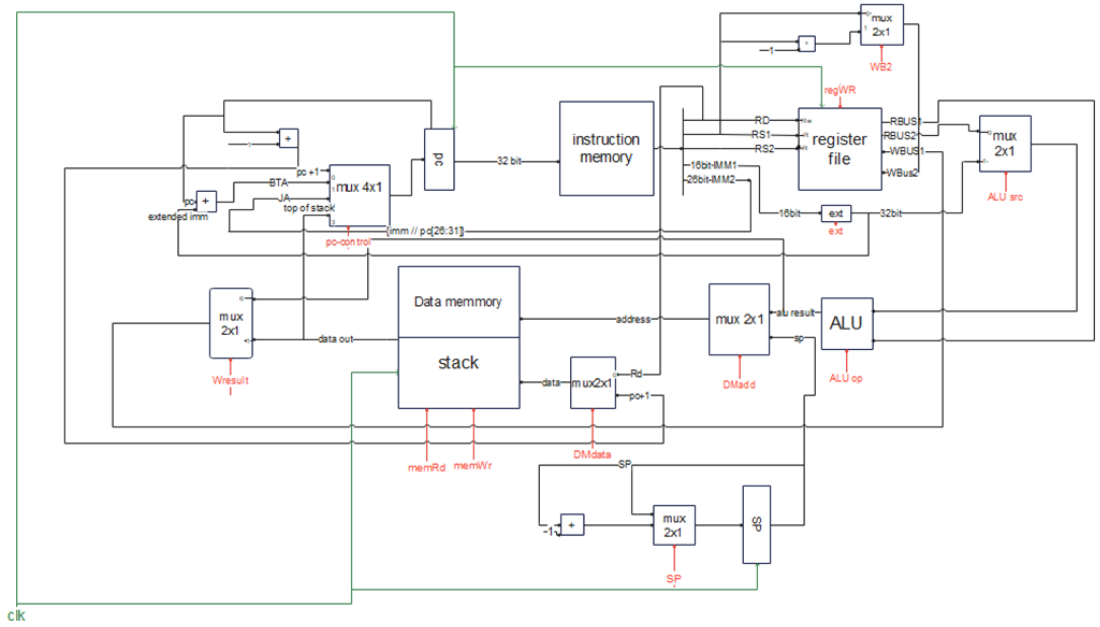

Figure 10-Data path

Testing

ALU RESULT

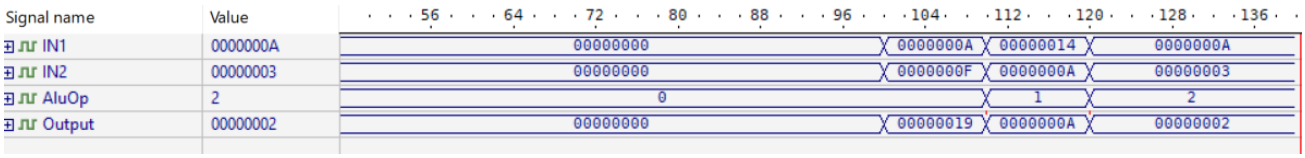

Figure 11-ALU result

The result here is true
Example 2 is and … A=1010 , 3 = 0010 …A and 3 = 0010=2
So the result is correct .

Contol unit

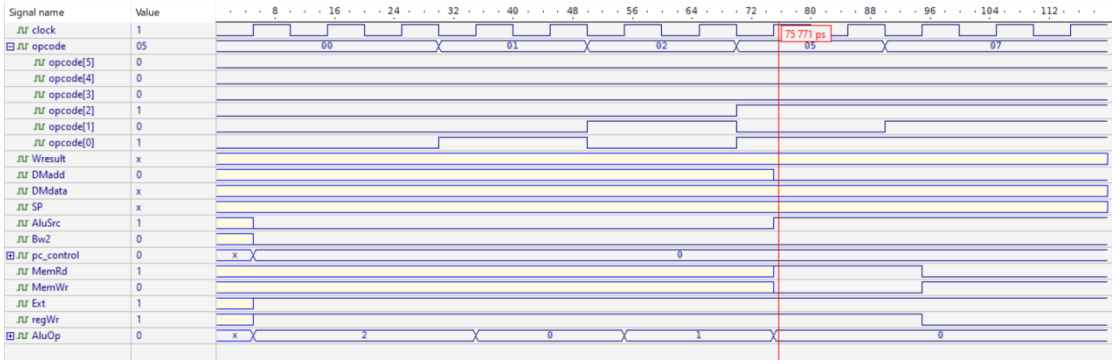

Figure 12-CONTROL unit result

The result here is true .
Example load if we see the result step by step , we will see that is correct result.

PC

Figure 13-PC result
Alse the result is corect
Example when pc = 3 add to the top of stack.

Conclusion

The design of processor includes multiple steps that must be precise and correct, to avoid conflictsor edge cases.

Testing must be made for all components and for all instructions, then a scenarios that manipulate data, and branches must be executed to test the performance and correctness of the design.