MERL-UIT
PAKISTAN

RV32I SINGLE CYCLE PROCESSOR

RISC-V Data Path

CREATED BY

ZAIN ALI
(19B-020-EE)

Table of Contents

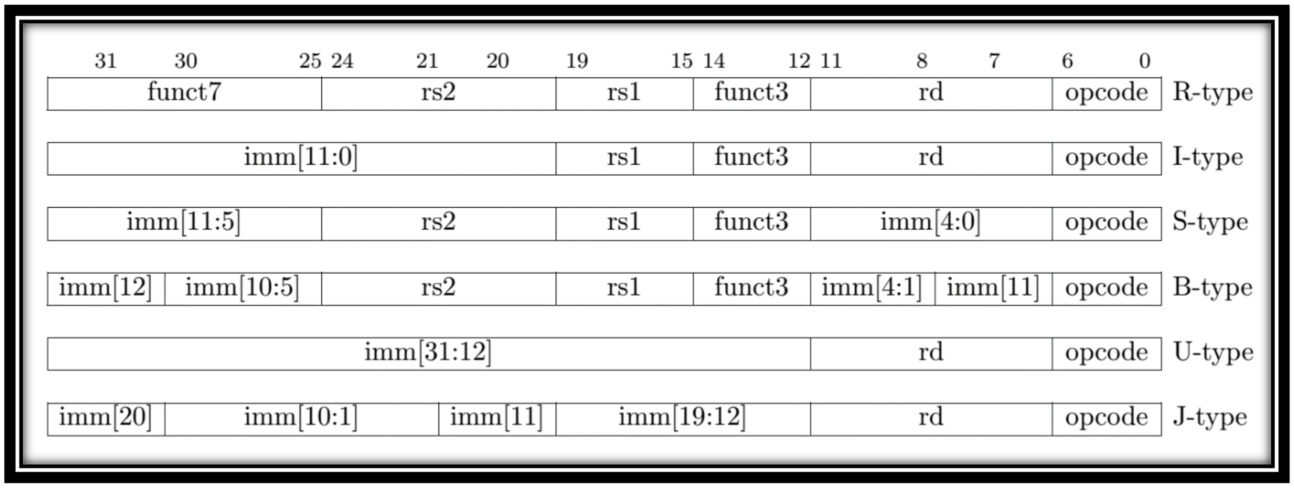
INTRODUCTION 3
METHODOLOGY 3
IMPLEMENTATION 4
RESULTS 4
VERIFICATION 4
TEST PROGRAMS 5
Fibonacci Series 5
Table of 5 6
CONCLUSION 6
REFERENCES 7

INTRODUCTION

We will first design a simpler processor that executes each instruction in only one clock cycle time. This is not efficient from performance point of view, A single cycle cpu executes each instruction in one cycle. in other words, one cycle is needed to execute any instruction. in other words, our cpi (cycle per instructions) is 1. Each cycle requires some constant amount of time. this means we will spend the same amount of time to execute every instruction [one cycle], since a clock cycle time (i.e. clock rate) must be chosen such that the longest instruction can be executed in one clock cycle and that makes shorter instructions execute in one unnecessarily long cycle. the big advantage of single cycle cpu's is that they are easy to implement.

METHODOLOGY

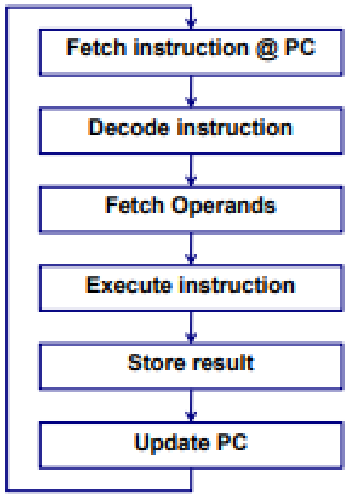
The base RISC-V instruction set is composed of just 47 instructions. Eight are system instructions that perform system calls and access performance counters. The remaining 39 instructions fall into the categories of computational instructions, control flow instructions, and memory access instructions Certain related instructions get represented in machine language with the same OP field (6-bit) value, but a different FUNC field (6-bit) value. So, the CPU will only be able to distinguish between them based on the FUNC field. For example, add and sub instructions are encoded with the same OP field value (000000) but different FUNC field values. add and sub use the same CPU resource (called the ALU). For add, the ALU will be instructed to add, while for sub, the ALU will be instructed to subtract, all based on the FUNC field value.



31	30 25	24 21	20	19 15	14 12	11 8	7	6 0	
funct7		rs2		rs1	funct3	rd		opcode	R-type
imm[11:0]				rs1	funct3	rd		opcode	I-type
imm[11:5]		rs2		rs1	funct3	imm[4:0]		opcode	S-type
imm[12]	imm[10:5]	rs2		rs1	funct3	imm[4:1]	imm[11]	opcode	B-type
imm[31:12]						rd		opcode	U-type
imm[20]	imm[10:1]		imm[11]	imm[19:12]		rd		opcode	J-type

IMPLEMENTATION

In a basic single-cycle implementation all operations take the same amount of time a single cycle. All instructions will execute in the same amount of time; this will determine the clock cycle time for our performance equations. Firstly, the program counter or PC register holds the address of the current instruction and updated at the end of every clock cycle. Secondly, we have instruction memory. Instruction memory is a state element that provides read-access to the instructions of a program. Thirdly, the control unit is responsible for setting all the control signals so that each instruction is executed properly. Type decode specify the types of instruction. Control Decoder depending upon the type of instruction that specifies a particular arithmetic operation. Fourthly, immediate generation that uses complete instruction and PC to generate the relevant immediate then register file stores thirty-two 32-bit values. The ALU performs the desired operation. Its result is stored in the destination register. Built a 32-bit ALU with 4-bit ALU operation Select. It performs the arithmetic or logical operation specified by the instruction. Add RAM and set its data bits to 32 bits and address width to 12 bits. Memory requires 32 (address) bits and 12 (write data) bits and just one output 32 bits. Finally, built a branch circuit to control branch instructions if branch function is true so to fetch the next instruction sequentially. Similar to branch, the jump instruction requires operand, immediate and perform add function so that why required 32-bit adder for jalr. Connect the wiring using the splitters, multiplexers, constants, tunnels, and clocks.



RESULTS

VERIFICATION

Troubleshooting of circuit is not such a complicated We can verify the circuit by checking by in six steps. Fetch instruction, Decode instruction, Fetch Operands, Execute instruction, Store result, Update PC verification will be much easier. If we check this process step by step, use Venus Online RV32I Simulator. This Simulator helps grasp the working behind the instruction much faster. To load the machine code on the instruction register, copy the code on Venus, then copy the machine code using the dump feature and create the abdcode.mem extension file using All file saving option on notepad

TEST PROGRAMS

Many programs are based on a particular type while some are based on a particular concept. some of them Discussed below:

"Fibonacci Series".

```
main:

li t0,0                # 1st Number

li t1,1                # 2nd Number

jal ra,swapping

swapping:

# Result = 1st Number + 2nd Number

add t2,t0,t1

# Updating 1st Number

sw t1,0x0(t3)

lw t0,0x0(t3)

# Updating 2nd Number

sw t2,0x0(t4)

lw t1,0x0(t4)

ret
```

"Table of 5".

```
addi x10 x0 10
loop1:                  # fuction_1
addi x1,x1,1
addi x5,x0,5
addi x6,x6,1
bne x6 x10 main
main:                   # fuction_2
slli x4 x6 2
add x7 x4 x1
bne x6 x10 loop1
```

CONCLUSION

In this project, we modeled a simplified single-cycle processor on Logisim software. We simulated our modules by using Venus Online RV32I Simulator and verified that our model did behave as expected. The processors can perform basic R, I, S, SB, U, UJ-type instructions correctly. The RV32I architecture is easy to understand, Implementing software-based instruction can be applied to hardware.

REFERENCES

- https://www.d.umn.edu/~gshute/mips/single-cycle.html
- https://inst.eecs.berkeley.edu/~cs61c/su19/disc/discussion7/disc7.pdf