
CS 224

Lab no. 4

Section no. 2

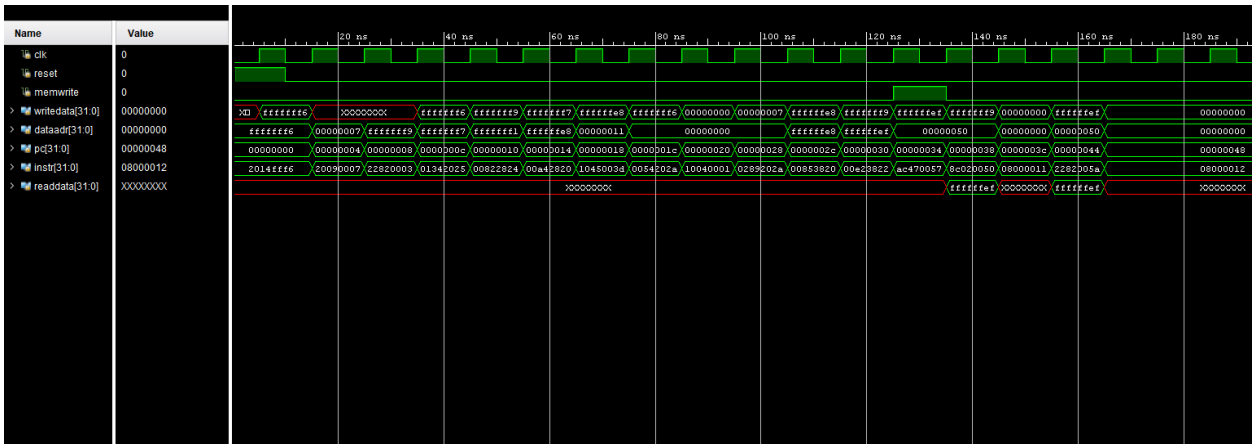
Ghulam Ahmed (22101001)

# Part 1. Setup and Test – Original10

a)

| Location | Machine Instruction (hex) | Assembly language |
|---|---|---|
| 00 | 0x2014fff6 | ADDI $s4, $zero, -10 |
| 04 | 0x20090007 | ADDI $t1, $zero, 7 |
| 08 | 0x22820003 | ADDI $v0, $s4, 3 |
| 0c | 0x01342025 | OR $a0, $t1, $s4 |
| 10 | 0x00822824 | AND $a1, $a0, $v0 |
| 14 | 0x00a42820 | ADD $a1, $a1, $a0 |
| 18 | 0x1045003d | BEQ $v0, $a1, 61 |
| 1c | 0x0054202a | SLT $a0, $v0, $s4 |
| 20 | 0x10040001 | BEQ $zero, $a0, 1 |
| 24 | 0x00002820 | ADD $a1, $zero, $zero |
| 28 | 0x0289202a | SLT $a0, $s4, $t1 |
| 2c | 0x00853820 | ADD $a3, $a0, $a1 |
| 30 | 0x00e23822 | SUB $a3, $a3, $v0 |
| 34 | 0xac470057 | SW $a3, 87($v0) |
| 38 | 0x8c020050 | LW $v0, 80($zero) |
| 3c | 0x08000011 | J 17 |
| 40 | 0x20020001 | ADDI $v0, $zero, 1 |
| 44 | 0x2282005a | ADDI $v0, $s4, 90 |
| 48 | 0x08000012 | J 18 |

## d) Waveform




## e) Observations

i) In an R-type instruction what does writedata correspond to?

Writedata is the value in rf[instr[20:16]].

ii) Why is writedata undefined for some of the early instructions in the program?

This is because the regfile is still waiting to be reset.

iii) In which instructions memwrite becomes 1?

The memwrite turns 1 for the instruction at pc 34 as we need to write to memory for that instruction.

iv) Why is dataaddr 0xfffffffe8 when PC is 0x14?

For the instruction at pc 14 we need to add the contents of $a1 and $a0 and then store it in $a1. $a0 currently holds -9 and $a1 holds -15. Therefore dataaddr would be -9 + -15 = -24, equivalent to 0xfffffffe8 in hex.

v) In the example program, when is the output of readdata defined and why?

For the instruction at pc 34, we need to store data memory at location [20] in the dmem module, therefore it has to be defined. Similarly for instruction at pc 38 we load from that location into readdata, therefore it is defined again.

## f) Modified ALU module

```
module alu(input  logic [31:0] a, b,
           input  logic [2:0]  alucont,
           output logic [31:0] result,
           output logic zero);

  always_comb
```

```
        case(alucont)
            3'b010: result = a + b;
            3'b110: result = a - b;
            3'b000: result = a & b;
            3'b001: result = a | b;
            3'b111: result = (a < b) ? 1 : 0;
            3'b011: result = a ^ b;
            default: result = {32{1'bx}};
        endcase

    assign zero = (result == 0) ? 1'b1 : 1'b0;
endmodule
```

# Part 2. Extend Original10

## Section 3 Instructions: spc & rol

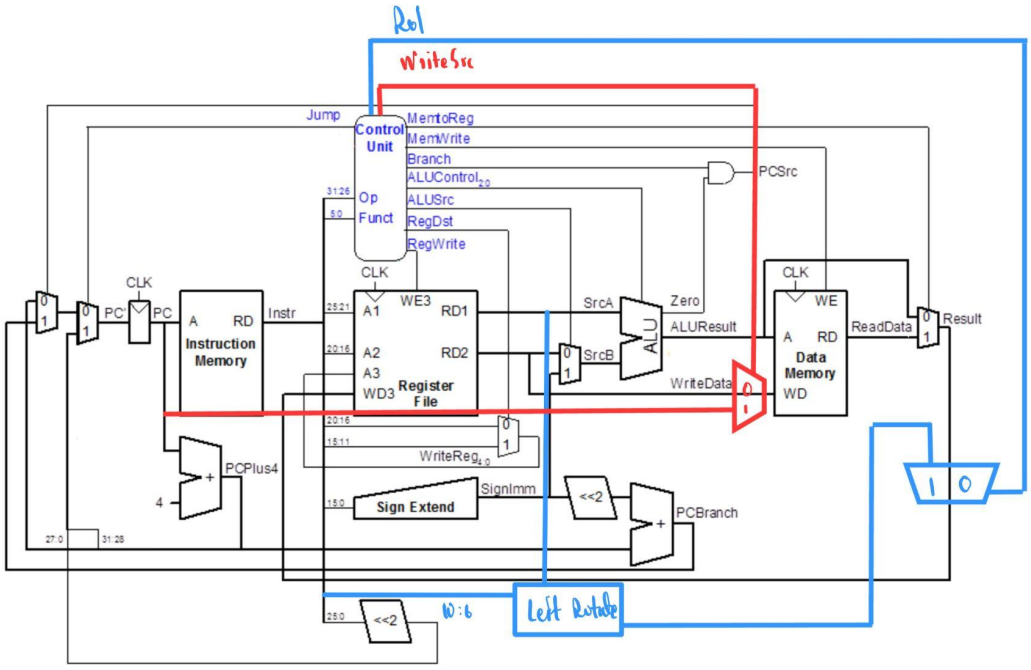
### a) RTL expression for spc

IM[PC]
M[R[rs] + SignExtImm] = PC
PC = PC + 4

### a) RTL expression for rol

IM[PC]
R[rd] = (R[rs] << shamt) | (R[rs] >> (32 - shamt))
PC = PC + 4

## b) Hardware diagram




## c) Tables

| Instruction | Opcode | RegWrite | RegDst | ALUSrc | Branch | MemWrite | MemToReg | ALUOp | Jump | WriteSrc |
|---|---|---|---|---|---|---|---|---|---|---|
| R-type | 000000 | 1 | 1 | 0 | 0 | 0 | 0 | 10 | 0 | 0 |
| lw | 100011 | 1 | 0 | 1 | 0 | 0 | 1 | 00 | 0 | 0 |
| sw | 101011 | 0 | X | 1 | 0 | 1 | X | 00 | 0 | 0 |
| beq | 000100 | 0 | X | 0 | 1 | 0 | X | 01 | 0 | 0 |
| addi | 001000 | 1 | 0 | 1 | 0 | 0 | 0 | 00 | 0 | 0 |
| j | 000010 | 0 | X | X | X | 0 | X | XX | 1 | 0 |
| rol | 000000 | 1 | 1 | X | 0 | 0 | X | XX | 0 | 0 |
| spc | 010000 | 0 | X | 1 | 0 | 1 | X | 00 | 0 | 1 |

| ALUOp | Funct | ALUControl | Rol |
|---|---|---|---|
| 00 | X | 010 (add) | 0 |
| 01 | X | 110 (subtract) | 0 |
| 1X | 100000 (add) | 010 (add) | 0 |
| 1X | 100010 (sub) | 110 (subtract) | 0 |
| 1X | 100100 (and) | 000 (and) | 0 |
| 1X | 100101 (or) | 001 (or) | 0 |
| 1X | 101010 (slt) | 111 (set less than) | 0 |
| 1X | 101011 (rol) | XXX | 1 |