**CS224**

**Lab 4**

**Section 3**

**Murat Furkan Uğurlu**

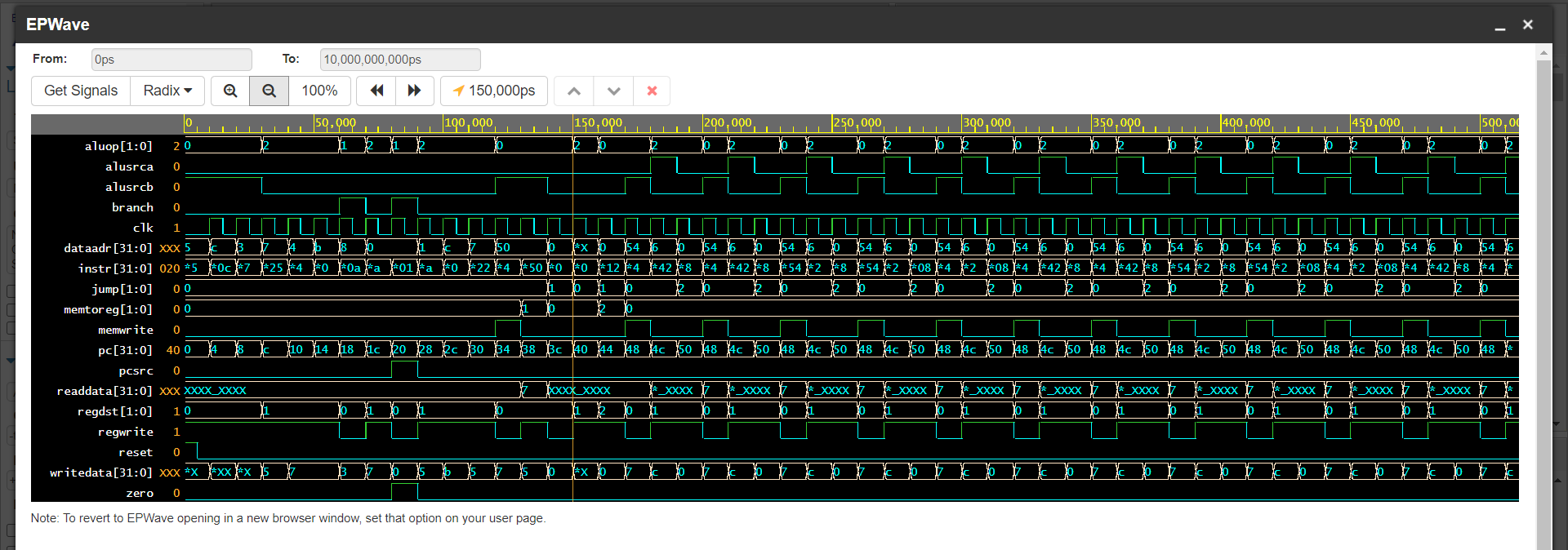
**21802062**

**PART I**

**A)**

|  |  |  |
| --- | --- | --- |
| **Address** | **Machine Instruction** | **Assembly Language** |
| 00 | 0x20020005 | addi $v0, $zero, 5 |
| 04 | 0x2003000c | addi $v1, $zero, 12 |
| 08 | 0x2067fff7 | addi $a3, $v1, -9 |
| 0c | 0x00e22025 | or $a0, $a3, $v0 |
| 10 | 0x00642824 | and $a1, $v1, $a0 |
| 14 | 0x00a42820 | add $a1, $a1, $a0 |
| 18 | 0x10a7000a | beq $a1, $a3, 10 |
| 1c | 0x0064202a | slt $a0, $v1, $a0 |
| 20 | 0x10800001 | beq $a0, $zero, 1 |
| 24 | 0x20050000 | addi $a1, $zero, 0 |
| 28 | 0x00e2202a | slt $a0, $a3, $v0 |
| 2c | 0x00853820 | add $a3 $a0 $a1 |
| 30 | 0x00e23822 | sub $a3 $a3 $v0 |
| 34 | 0xac670044 | sw $a3, 68($v1) |
| 38 | 0x8c020050 | lw $v0, 80($zero) |
| 3c | 0x08000010 | j 0x0000010 |
| 40 | 0x001f6020 | add $t4, $zero, $ra |
| 44 | 0x0c000012 | jal 0x0000012 |
| 48 | 0xac020054 | sw $v0, 84($zero) |
| 4c | 0x00039042 | srl $s2, $v1, 1 |
| 50 | 0x03E00008 | jr $ra |

**E)**

****

**F)**

I. It is the data to be written into data memory.

II. Since early instructions do not write data to data memory (because they are I type instructions), it is undefined.

III. Since readData is used for only load and store operations, it is not defined most of the time.

IV. It corresponds to ALU output, and it is written into register file.

V. When $ra register is used in an arithmetic operation, dataadr becomes undefined because $ra register holds an address which is not in the number range for arithmetic operations.

**G)**

I. No, I do not need. If ALUSrca is set to zero in controller, then shift amount will come from register.

II. I would modify alu module since shift operation is executed there. I would put another case for the operation shift left (the operation would be b << a).

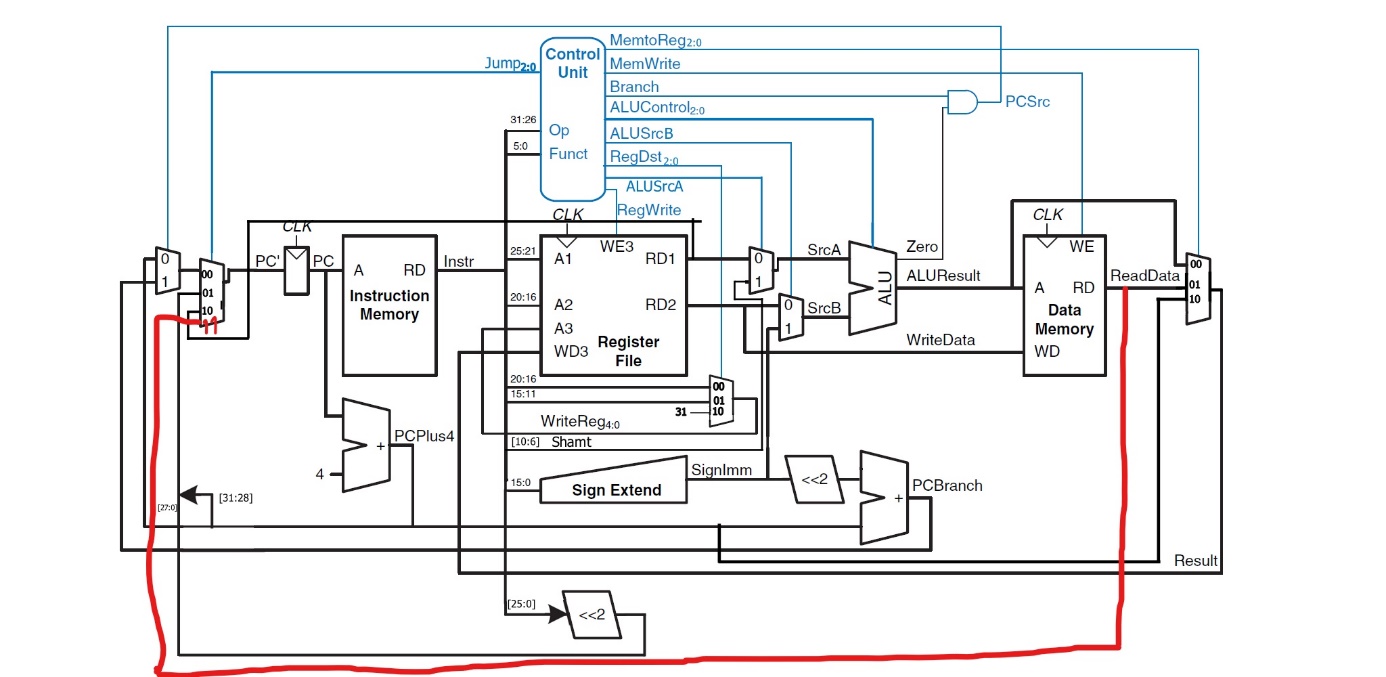
**PART II**

A) IM[PC]

R[rt] 🡸 PC + 4

PC 🡸 M [ R[rs] + SignExtImm ]

B)



C)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instruction | Opcode | RegWrite | RegDst | ALUSrcA | ALUSrcB | Branch | MemWrite | MemToReg | ALUOp | Jump |
| R-type | 000000 | 1 | 01 | 0 | 0 | 0 | 0 | 00 | 10 | 00 |
| srl | 000000 | 1 | 01 | 1 | 0 | 0 | 0 | 00 | 10 | 00 |
| lw | 100011 | 1 | 00 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| sw | 101011 | 0 | X | 0 | 1 | 0 | 1 | XX | 00 | 00 |
| beq | 000100 | 0 | X | 0 | 0 | 1 | 0 | 01 | 01 | 00 |
| addi | 001000 | 1 | 00 | 0 | 1 | 0 | 0 | 00 | 00 | 00 |
| j | 000010 | 0 | X | X | X | X | 0 | XX | XX | 01 |
| jal | 000011 | 1 | 10 | X | X | X | 0 | 10 | XX | 01 |
| jr | 000000 | 1 | 01 | 0 | 0 | 0 | 0 | 00 | 10 | 10 |
| jalm | 101010 | 1 | 00 | 0 | 1 | 0 | 0 | 10 | 00 | 11 |