Lab – CSR Support

In this lab, we are going to support privileged architecture in our pipelined processor. For this purpose, we are going to partially support the machine mode of the RISC-V specification in our processor. This will involve adding support for some new instructions in our datapath. We are also going to create a new register file which is going to contain the machine mode CSR registers which can be accessed by these new instructions.

CSR Register File

First, we are going to create a new registerfile which will contain our CSR registers. For this lab, we are only going to implement **`mip, mie, mstatus, mcause, mtvec`** and **`mepc`** in our register file. These registers have their 12-bit address defined as part of the RISC-V specification. The register file will have the address of the CSR register, the value of PC during the Memory-Writeback stage, the CSR register write control, the CSR register read control, the interrupt/exception pins and the CSR register write data at its input. The register file will have the CSR read data and the exception PC at its output. This can be illustrated by Figure 9.1.

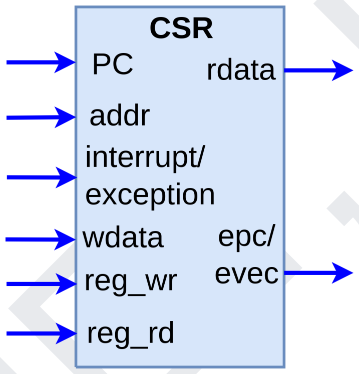

Figure 9.1. CSR Register File.

Listing 9.1 illustrates the implementation of the CSR Register file read and write operations. In the illustrated code we can see that the read and write operation will depend on the 12-bit address of the CSR registers. For checking the values address of the CSR registers refer to Table 2.5 of the RISC-V privileged specifications manual.

```
// CSR read operation
always_comb begin
   csr_rdata  = '0;
   if(exe2csr_ctrl.csr_reg_rd) begin
       case (exe2csr_data.csr_addr)
           CSR_ADDR_MSTATUS         : csr_rdata    = csr_mstatus_ff;
           CSR_ADDR_MIE             : csr_rdata    = csr_mie_ff;
           ...
           CSR_ADDR_MEPC            : csr_rdata    = csr_mepc_ff;
       endcase // exu2csr_data.csr_addr
   end
end
```

```
// CSR write operation
always_comb begin
   csr_mstatus_wr_flag         = 1'b0;
   csr_mie_wr_flag             = 1'b0;
   ...
   csr_mepc_wr_flag            = 1'b0;
   if (exe2csr_ctrl.csr_wr_req) begin
       case (exe2csr_data.csr_addr)
           CSR_ADDR_MSTATUS        : csr_mstatus_wr_flag  = 1'b1;
           CSR_ADDR_MIE            : csr_mie_wr_flag      = 1'b1;
           ...
           CSR_ADDR_MEPC           : csr_mepc_wr_flag     = 1'b1;
       endcase // exu2csr_data.csr_addr
   end // exe2csr_ctrl.csr_wr_req
end

// Update the mip (machine interrupt pending) CSR
always_ff @(negedge rst_n, posedge clk) begin
   if (~rst_n) begin
       csr_mip_ff <= {`XLEN{1'b0}};
   end else if (csr_mip_wr_flag) begin
       csr_mip_ff <= csr_wdata;
   end
end
...
// Update the mtvec CSR
always_ff @(negedge rst_n, posedge clk) begin
   if (~rst_n) begin
       csr_mtvec_ff <= {`XLEN{1'b0}};
   end else if (csr_mtvec_wr_flag) begin
       csr_mtvec_ff <= csr_wdata;
   end
end
```

Listing 9.1. CSR Register File Read and Write operations.

Tasks

- Implement the CSR register file while complying with the specifications manual and simulate it to check the read and write operation.

Implementing the CSR Instructions

In this lab, we are going to implement the ***CSRRW*** and ***mret*** instructions in our pipeline. The `CSRRW` is the CSR Read Write instruction which reads the old value of the CSR register and saves it in the destination register. Then the value of the source register is written to the CSR register. Figure 9.2 illustrates the instruction format for the CSR instruction.

31 20	15	12	7	0
csr	rs1	func3	rd	opcode

Figure 9.2. CSRRW Instruction Format.

Implementation of CSRRW

For implementing these instructions, we will be required to integrate the CSR register file in our datapath. We will also modify our controller for adding two new control signals for the read and write operation of the CSR register file. The address of the CSR register file will be given by the immediate value generator and the data to be written to the CSR register will be given by the output of the rs1 forwarding mux. The read data output of the CSR register file will be connected to the writeback mux. These changes have been illustrated in Figure 9.3.

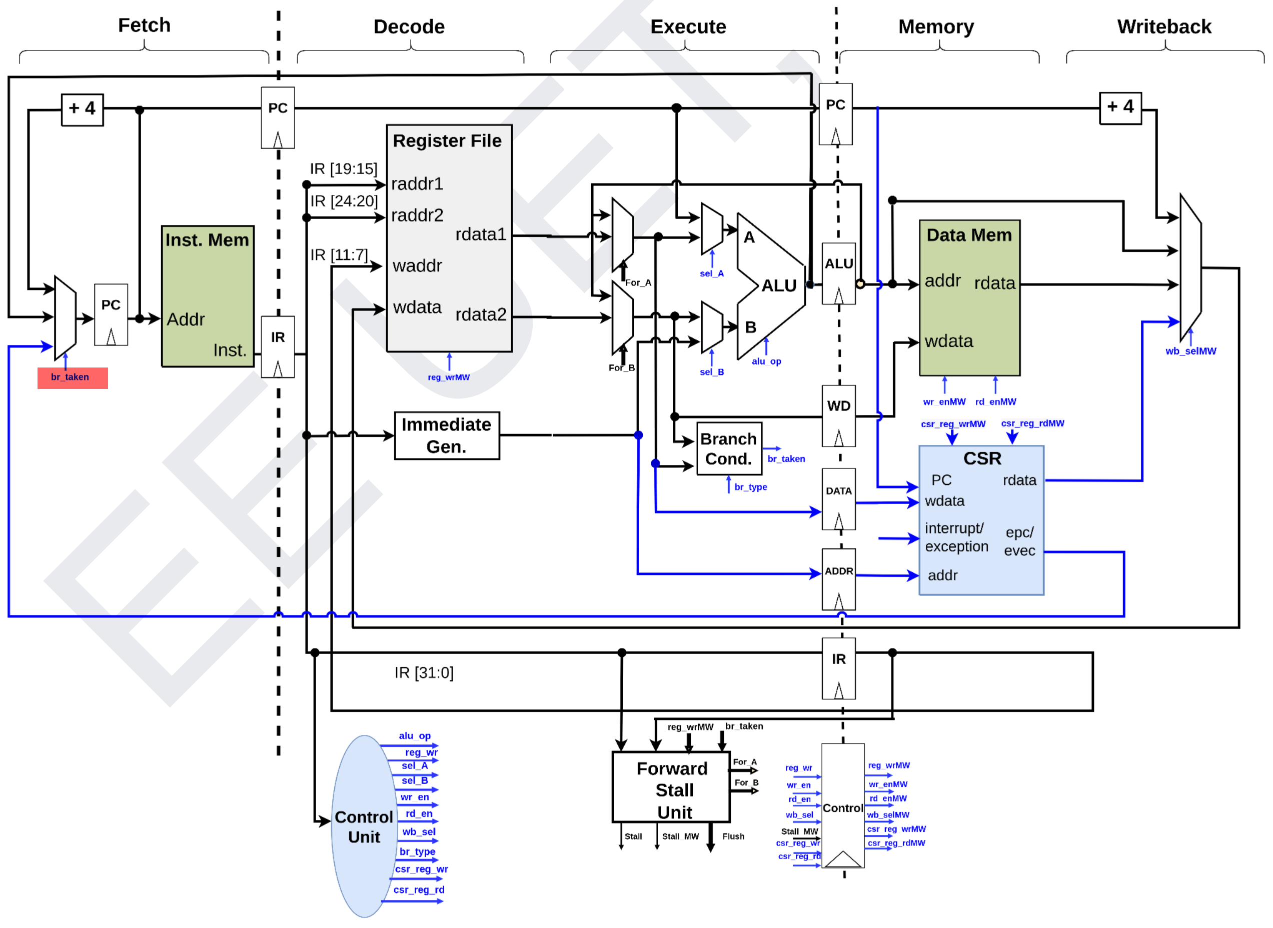

Figure 9.3. Integration of CSRRW instruction in the Datapath.

Tasks

- Integrate the CSR register file and implement the ***CSRRW*** instruction in your processor.
- Write a simple assembly code to test whether the CSR instruction is working correctly or not. A sample code has been provided in Listing 9.2.

```
li x10,1
li x12,10
csrrw x11,mtvec,x10
csrrw x13,mtvec,x12
```

Listing 9.2. Sample CSR Instruction Test assembly

Implementation of MRET

Now we will also modify our controller for adding a new multiplexor and new control signals for detecting the ***mret*** instruction. The CSR register file receives this control signal from the pipeline register. In case the ***mret*** instruction is received by the CSR register file the value of **`mepc`** register will be loaded in the **`epc/evec`** output of the register file and **`epc_taken`** flag of the mux will be asserted.

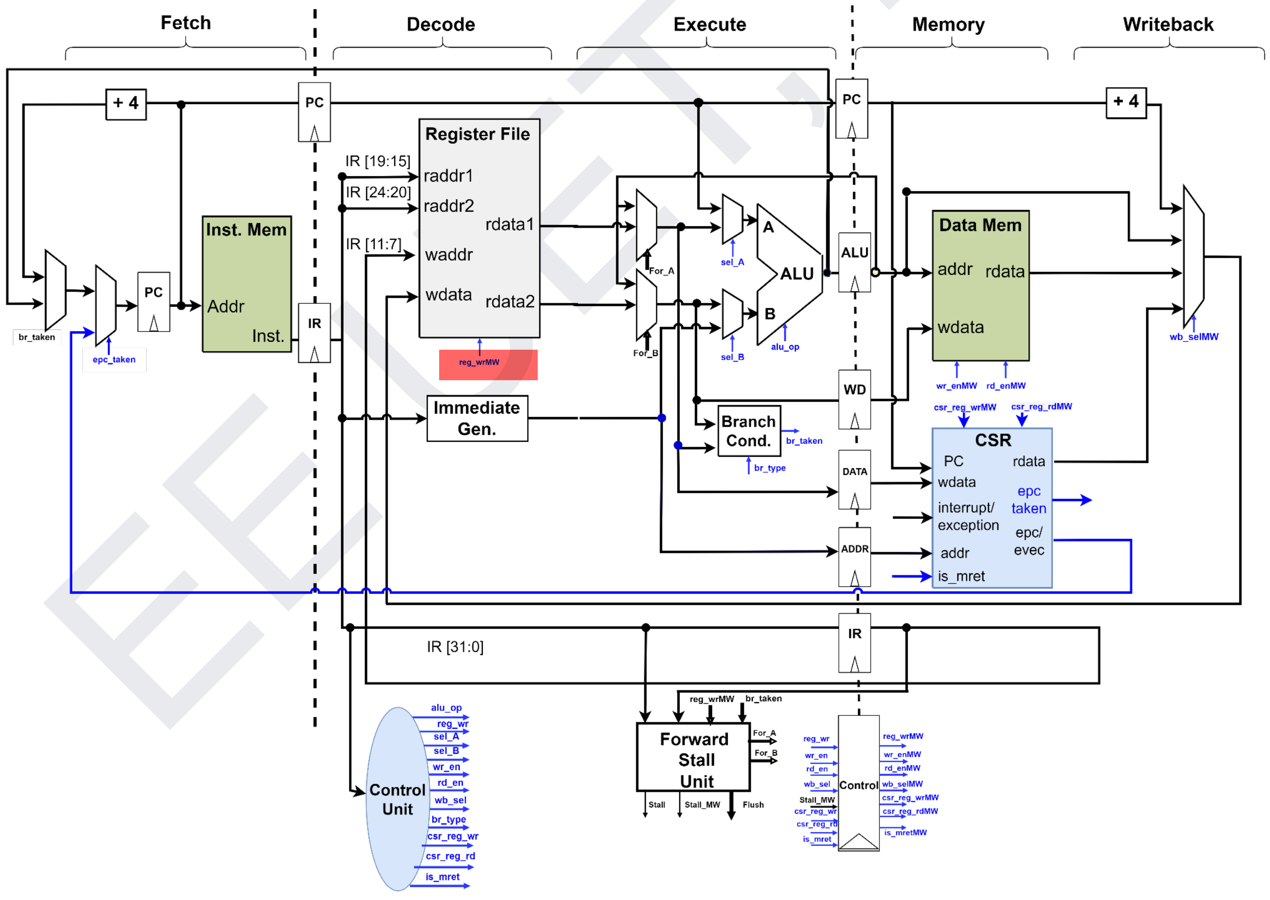

Figure 9.4. Integration of MRET instruction in the Datapath.

Tasks

- Implement the `MRET` instruction in your processor.

Interrupt Handling

Whenever an interrupt arrives, the interrupt will be registered at the positive edge of the clock in the corresponding bit of the **`mip`** register based on the source of the interrupt. If the corresponding bit of the **`mie`** register is high and the MIE bit of the **`mstatus`** register is high then we say that the Exception/interrupt has occurred and the value of PC in Memory-Writeback stage will be saved in the **`mepc`** register.

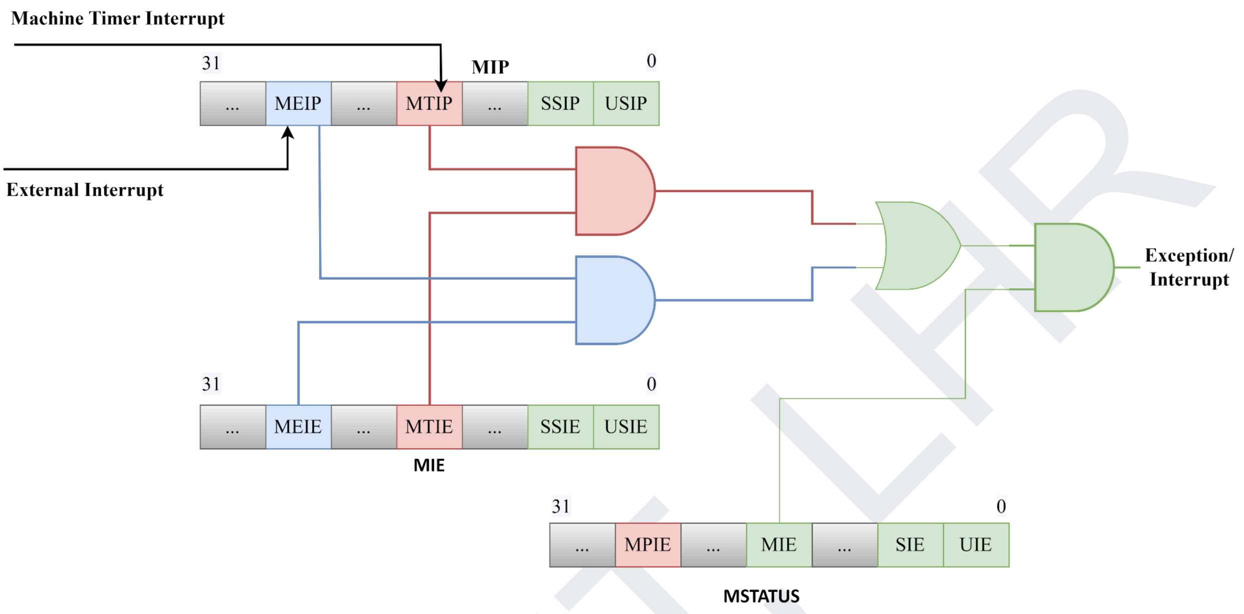

Figure 9.5. Interrupt Handling in CSR Register File.

When the Exception/interrupt has occurred **`epc_taken`** flag in datapath will be asserted and the cause of the interrupt will be saved in the **`mcause`** register by the hardware. Based on the mode of the **`mtvec`** register, the **`epc/evec`** address will be calculated as the ***`base_address`*** and ***`base_address+cause*4`*** in non-vector mode and vector mode respectively. Please note that the **`mie, mstatus`** and **`mtvec`** register will be configured by software using ***`csrrw`*** instruction.

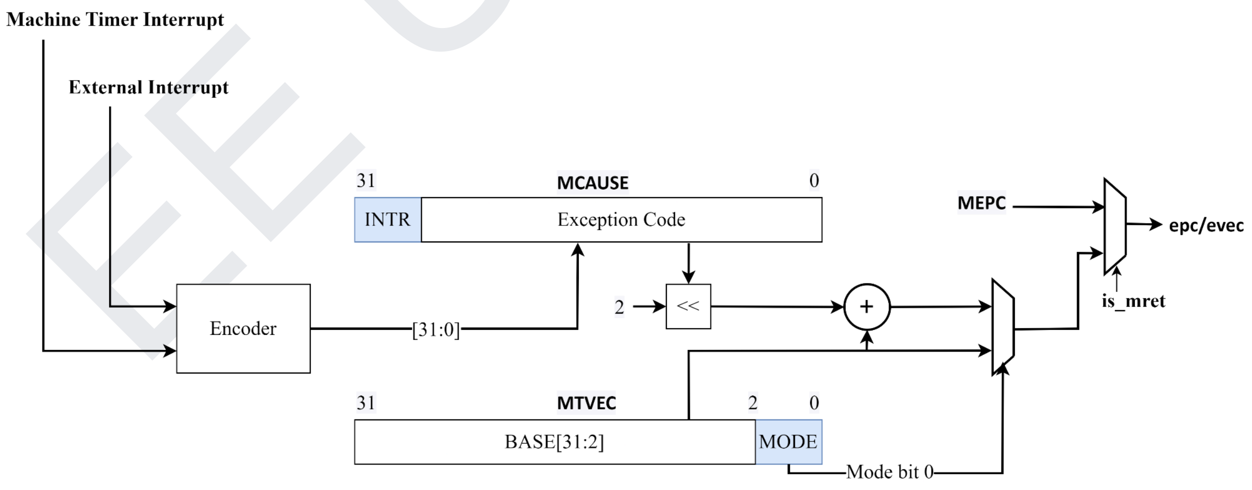

Figure 9.6. Address Calculation in CSR Register File.

Tasks

- Modify the CSR register file for handling single interrupt. Generate the interrupts using testbenches.
- Write a simple assembly code to test whether the processor is handling interrupt. Consult the RISC-V specifications for configuring the CSR registers. A sample code is shown in Listing 9.3.

```
j main
j interrupt_handler

main:
    li x11,(calculate your own value)
    li x12,(calculate your own value)
    li x13,(calculate your own value)
    csrrw x0,mie,x11
    csrrw x0,mstatus,x12
    csrrw x0,mtvec,x13

loop:
    addi x14,x14,1
    j loop

interrupt_handler:
    csrrw x0,mie,x0
    li x2,0xFFFFFFFF
    xor x3,x3,x2
    csrrw x0,mie,x11
    mret
```

Listing 9.3. Sample Interrupt Handling Test assembly