



C o m p u t e r O r g a n i z a t i o n



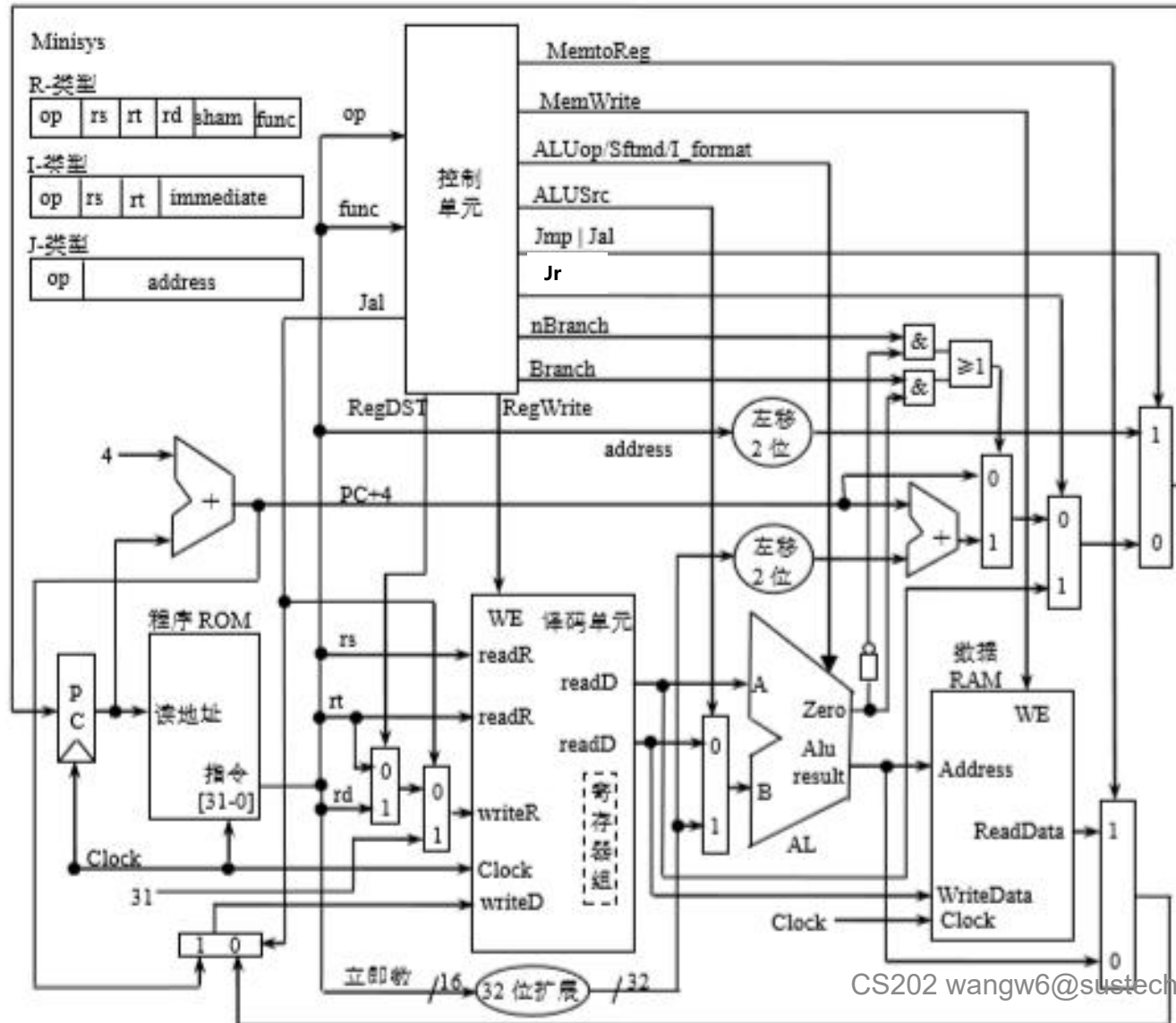
Lab12 CPU(4) I/O, Controller+



Topics

- CPU (4)
 - CPU with I/O
 - I/O
 - Controller+

3 Single Cycle CPU continued



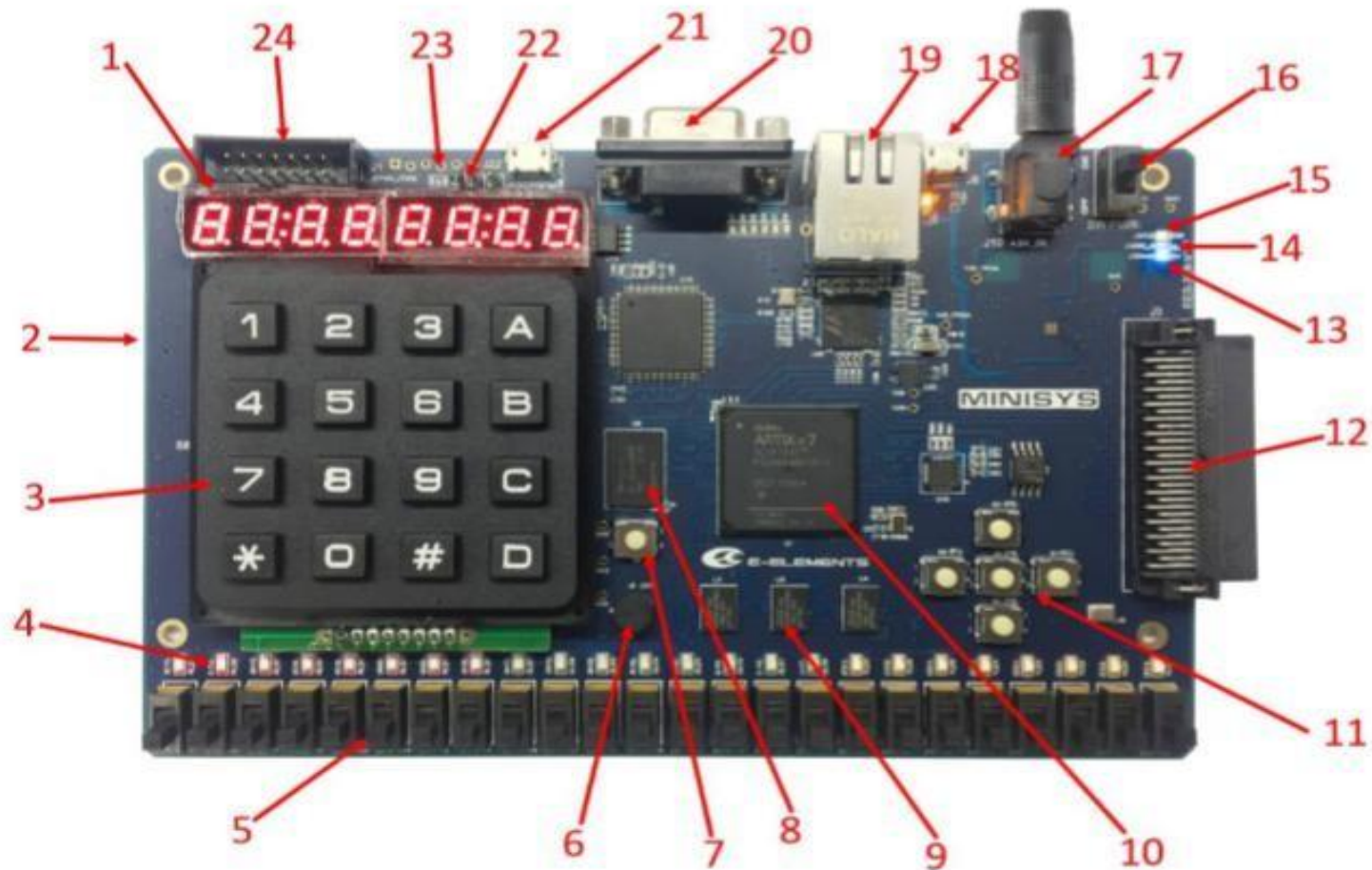
Build a CPU top module

1) **Instantiating** the sub-modules: **clock**, **Decoder**, execution unit/**ALU**, **IFetch**, **Controller** and **Data-Memory**.

2) Complete the inter-module connection inside the CPU and the **binding** to the CPU **port**.

Q. How to test the CPU ?
how to determine the
program, the data, how to
check the result?

I/O Interface



5

A Simple Design on the I/O Interface

This part mainly accomplishes the following work:

1. Add I/O function
2. 16-bit LED design
3. 16-bit DIP Switch design

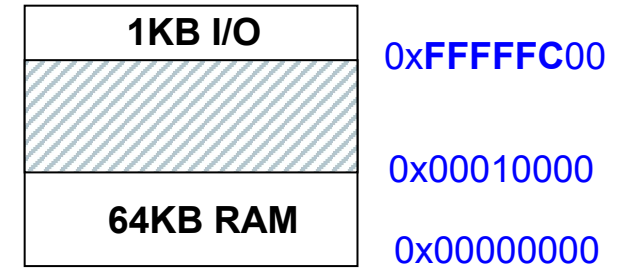
It is just a way for IO address implementation, NOT the ONLY choice.

I/O Share Part of the Data Bus Address

The space of **32** bits address bus is **4GB**(0x0000_0000~0xFFFF_FFFF)

1024 bytes(0xFFFF_FC00~0xFFFF_FFFF) is designed to be allocated for the **I/O**.

Chip **Select** and **address** are specified by specifying **10** IO port lines.



Here is an example for **24 LED lights** and **24 DIP switches** on Minisys board, both of them are divided into two groups, all the ports in one group share the same address.

1. The CS(Chip Select) signal of the LED light is **ledCtrl**
2. The CS(Chip Select) signal of the DIP switch is **switchCtrl**

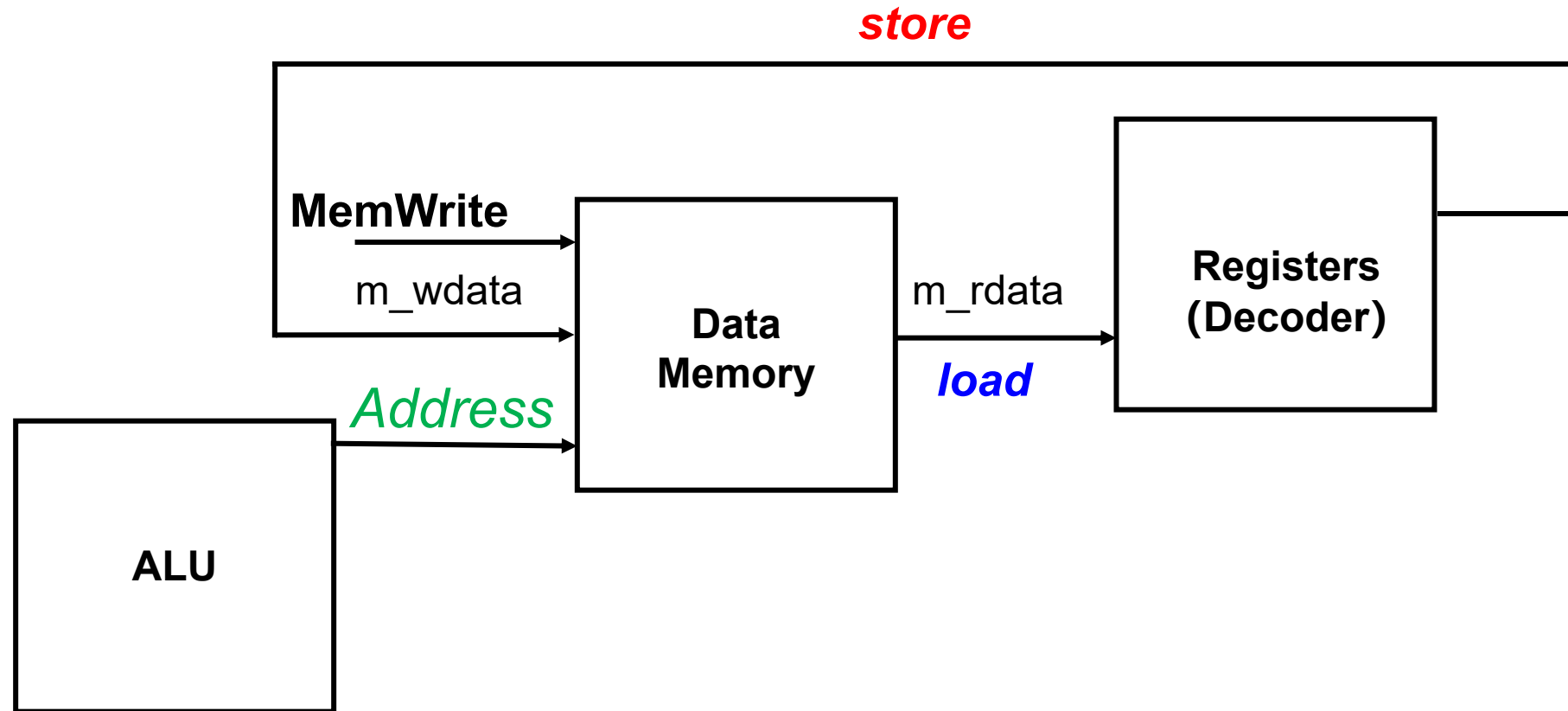
Range	LED(1~16)	LED(17~24)	Switch(1~16)	Switch(17~24)
Address	0xFFFFFC60	0xFFFFFC62	0xFFFFFC70	0xFFFFFC72

Note:

1. In the computer field, there are usually two schemes for I/O address space design: I/O and memory **unified addressing** or **I/O independent addressing**. However there is no dedicated I/ O instruction in current Minisys-1. Here, both LW and SW instructions are used for RAM access and I/O access, which means Minisys-1 can only use I/O unified addressing.

2. It is just a way for IO address implementation, but not the only choice.

Corresponding Operation of LW/SW

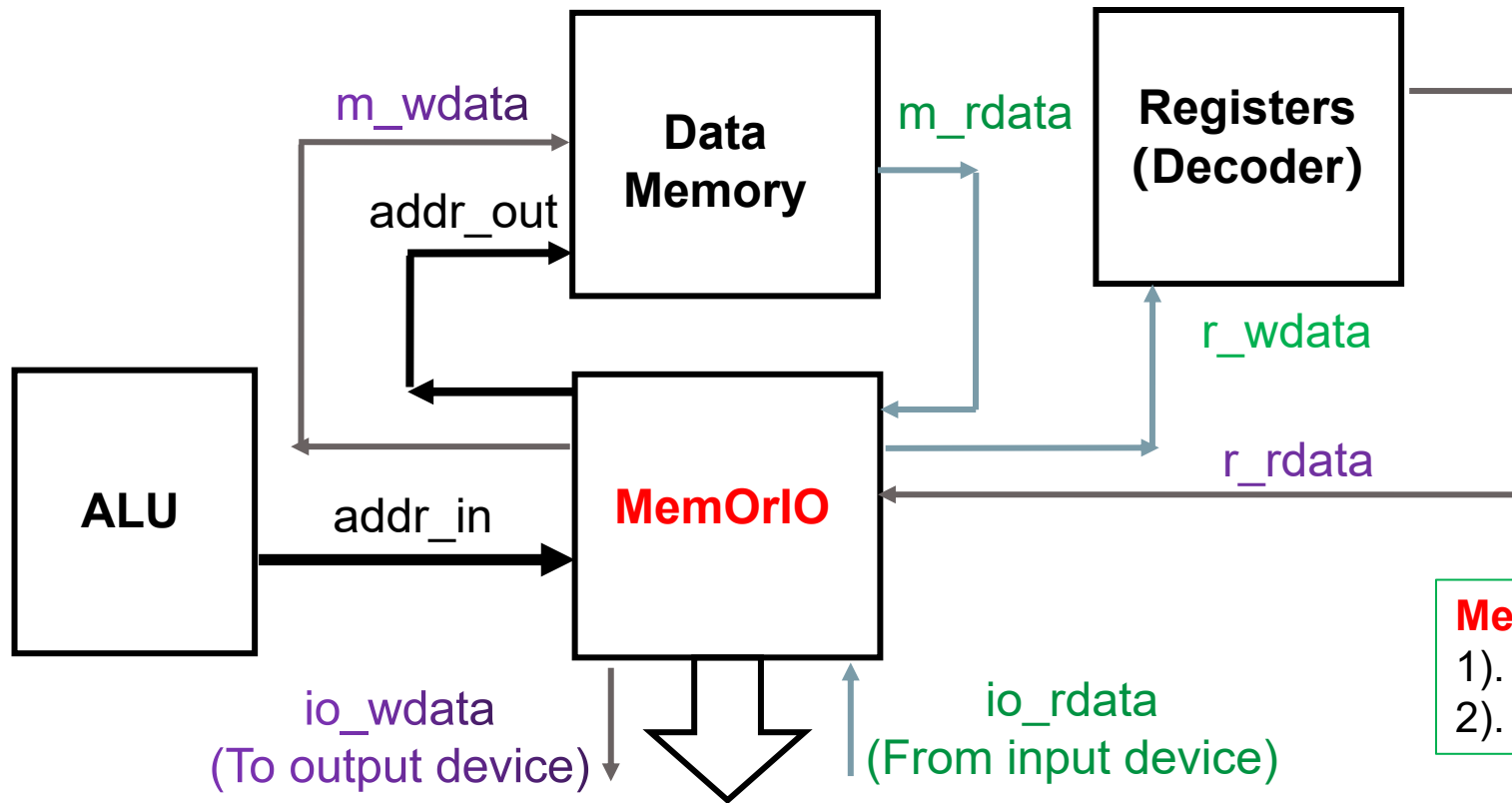


NOTE:

- 1) There is no specific instruction in Minisys to read data from input ports and write data to output ports.
- 2) To implement the read/write process on I/O, it needs to **share the load/store instructions** in Minisys.

MemOrIO

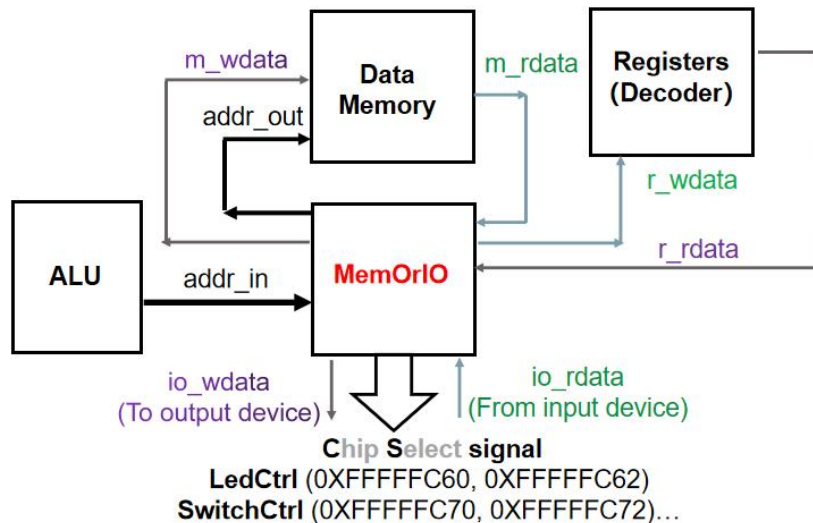
8



MemOrIO determine:
1). The source of `r_rdata`
2). The destination of `r_wdata`

Chip Select signal
LedCtrl (0xFFFFFC60, 0xFFFFFC62)
SwitchCtrl (0xFFFFFC70, 0xFFFFFC72)...

MemOrIO continued



```

module MemOrIO( mRead, mWrite, ioRead, ioWrite, addr_in, addr_out,
m_rdata, io_rdata, r_wdata, r_rdata, write_data, LEDCtrl, SwitchCtrl);

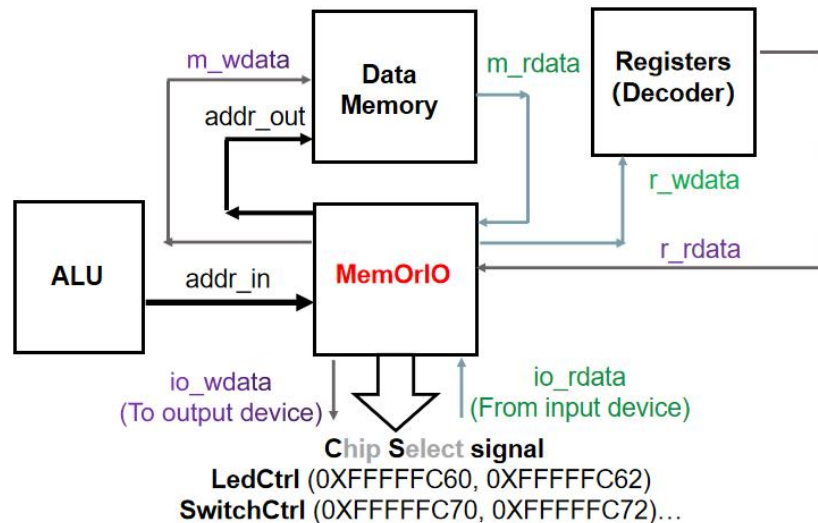
input mRead;           // read memory, from Controller
input mWrite;          // write memory, from Controller
input ioRead;          // read IO, from Controller
input ioWrite;         // write IO, from Controller

input[31:0] addr_in;    // from alu_result in ALU
output[31:0] addr_out;  // address to Data-Memory

input[31:0] m_rdata;    // data read from Data-Memory
input[15:0] io_rdata;   // data read from IO, 16 bits
output[31:0] r_wdata;   // data to Decoder(register file)

input[31:0] r_rdata;    // data read from Decoder(register file)
output reg[31:0] write_data; // data to memory or I/O (m_wdata, io_wdata)
output LEDCtrl;         // LED Chip Select
output SwitchCtrl;      // Switch Chip Select
    
```

MemOrIO continued



```
assign addr_out= addr_in;
```

```
// The data write to register file may be from memory or io.
```

```
// While the data is from io, it should be the lower 16bit of r_wdata.
```

```
assign r_wdata = ? ? ?
```

```
// Chip select signal of Led and Switch are all active high;
```

```
assign LEDCtrl= ? ? ?
```

```
assign SwitchCtrl= ? ? ?
```

```
always @* begin
```

```
    if((mWrite==1)|| (ioWrite==1))
```

```
        //write_data could go to either memory or IO. where is it from?
```

```
        write_data = ? ? ?
```

```
    else
```

```
        write_data = 32'hZZZZZZZZ;
```

```
end
```

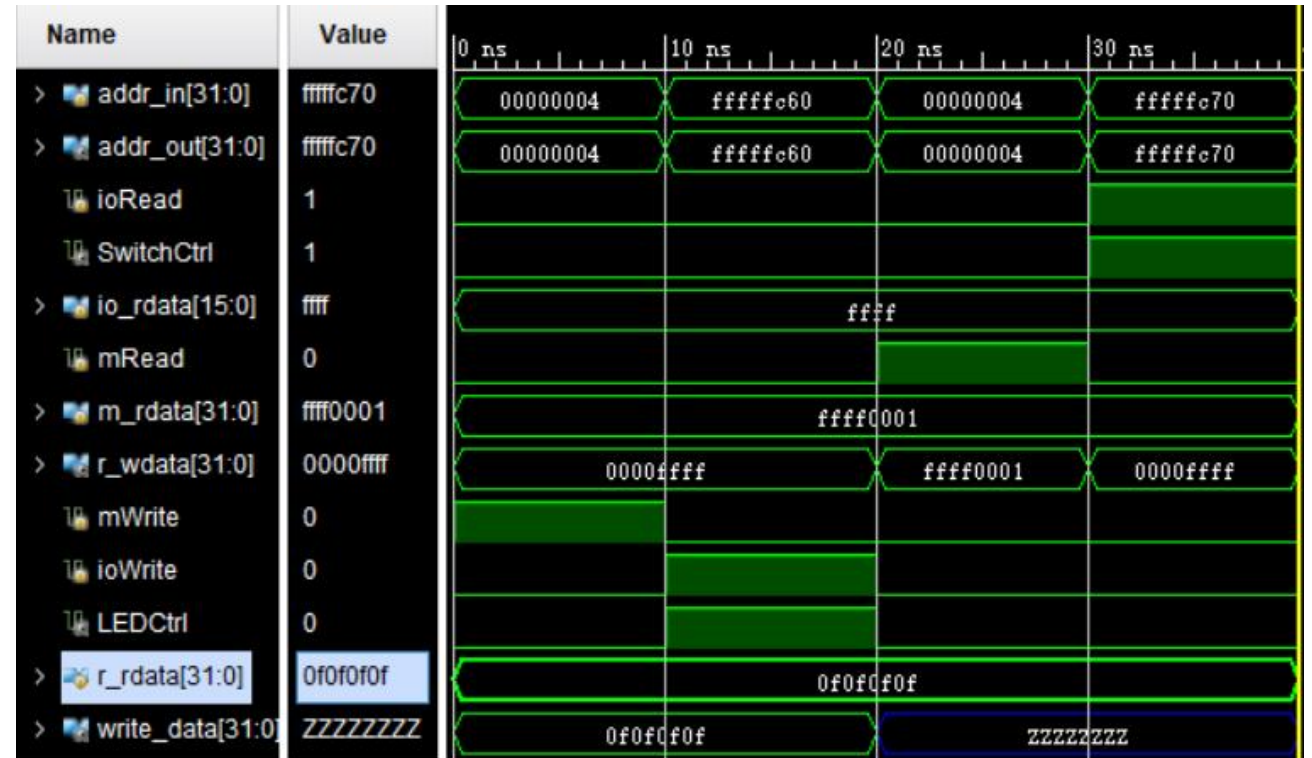
```
endmodule
```

The Function Verification of MemOrIO

// a reference for the testbench of MemOrIO

```
module MemOrIO_tb( );
    reg mRead,mWrite,ioRead,ioWrite;
    reg[31:0] addr_in,m_rdata,r_rdata;
    reg[15:0] io_rdata;
    wire LEDCtrl,SwitchCtrl;
    wire [31:0] addr_out,r_wdata,write_data;
```

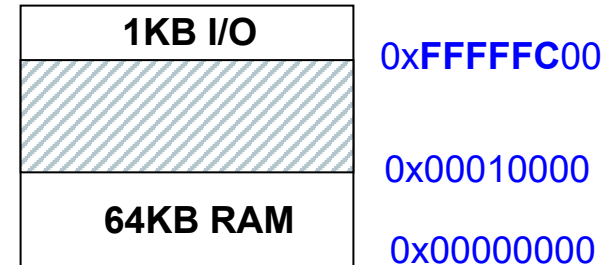
```
    MemoryOrIO umio(addr_out, addr_in,
        mRead, mWrite, ioRead, ioWrite,
        m_rdata, io_rdata, r_rdata, r_wdata, write_data,
        LEDCtrl, SwitchCtrl );
```



```
initial begin // r_rdata -> m_wdata(write_data)
    m_rdata = 32'h0xffff_0001; io_rdata = 32'h0xffff; r_rdata = 32'h0x0f0f_0f0f; addr_in = 32'h4;{mRead,mWrite,ioRead,ioWrite}= 4'b01_00;
    #10 addr_in = 32'hffff_fc60; {mRead,mWrite,ioRead,ioWrite}= 4'b00_01; // r_rdata -> io_wdata(write_data)
    #10 addr_in = 32'h0000_0004; {mRead,mWrite,ioRead,ioWrite}= 4'b10_00; // m_rdata -> r_wdata
    #10 addr_in = 32'hffff_fc70; {mRead,mWrite,ioRead,ioWrite}= 4'b00_10; // io_rdata -> r_wdata(write_data)
    #10 $finish;
end
endmodule
```

Controller +

Add new ports to Controller for IO reading and writing support.



```
module control32(Opcode,Function_opcode,Jr,Branch,nBranch,Jump,Jal,
  Alu_resultHigh,
  RegDST, MemorIOtoReg, RegWrite,
  MemRead, MemWrite,
  IORead, IOWrite,
  ALUSrc,ALUOp,Sftmd,I_format);
  ...
  input[21:0] Alu_resultHigh; // From the execution unit Alu_Result[31..10]
  output MemorIOtoReg; // 1 indicates that data needs to be read from memory or I/O to the register
  output RegWrite; // 1 indicates that the instruction needs to write to the register
  output MemRead; // 1 indicates that the instruction needs to read from the memory
  output MemWrite; // 1 indicates that the instruction needs to write to the memory
  output IORead; // 1 indicates I/O read
  output IOWrite; // 1 indicates I/O write
  ...
```

Controller + continued

- 1) **Modify** the logic of the '**MemWrite**'
- 2) **Add** '**MemRead**', '**IORead**' and '**IOWrite**' signals
- 3) **Change** '**MemtoReg**' to '**MemorIotoReg**'.

```
// The real address of LW and SW is Alu_Result, the signal comes from the execution unit
// From the execution unit Alu_Result[31..10], used to help determine whether to process Mem or IO
input[21:0] Alu_resultHigh;

output    MemorIotoReg;    //1 indicates that read data from memory or I/O to write to the register
output    MemRead;        // 1 indicates that reading from the memory to get data
output    IORead;         // 1 indicates I/O read
output    IOWrite;        // 1 indicates I/O write

assign RegWrite = (R_format || Lw || Jal || I_format) && !(Jr) ;    // Write memory or write IO
assign MemWrite = ((sw==1) && (Alu_resultHigh[21:0] != 22'h3FFFFFF)) ? 1'b1:1'b0;
assign MemRead = ? ? ?    // Read memory
assign IORead = ? ? ?    // Read input port
assign IOWrite = ? ? ?    // Write output port

// Read operations require reading data from memory or I/O to write to the register
assign MemorIotoReg = IORead || MemRead;
```

Practice

Complete the following modules, do the function verification:

- 1. MemoryOrIO
- 2. Controller+
- 3. Single cycle CPU with I/O process