计算机系统结构实验

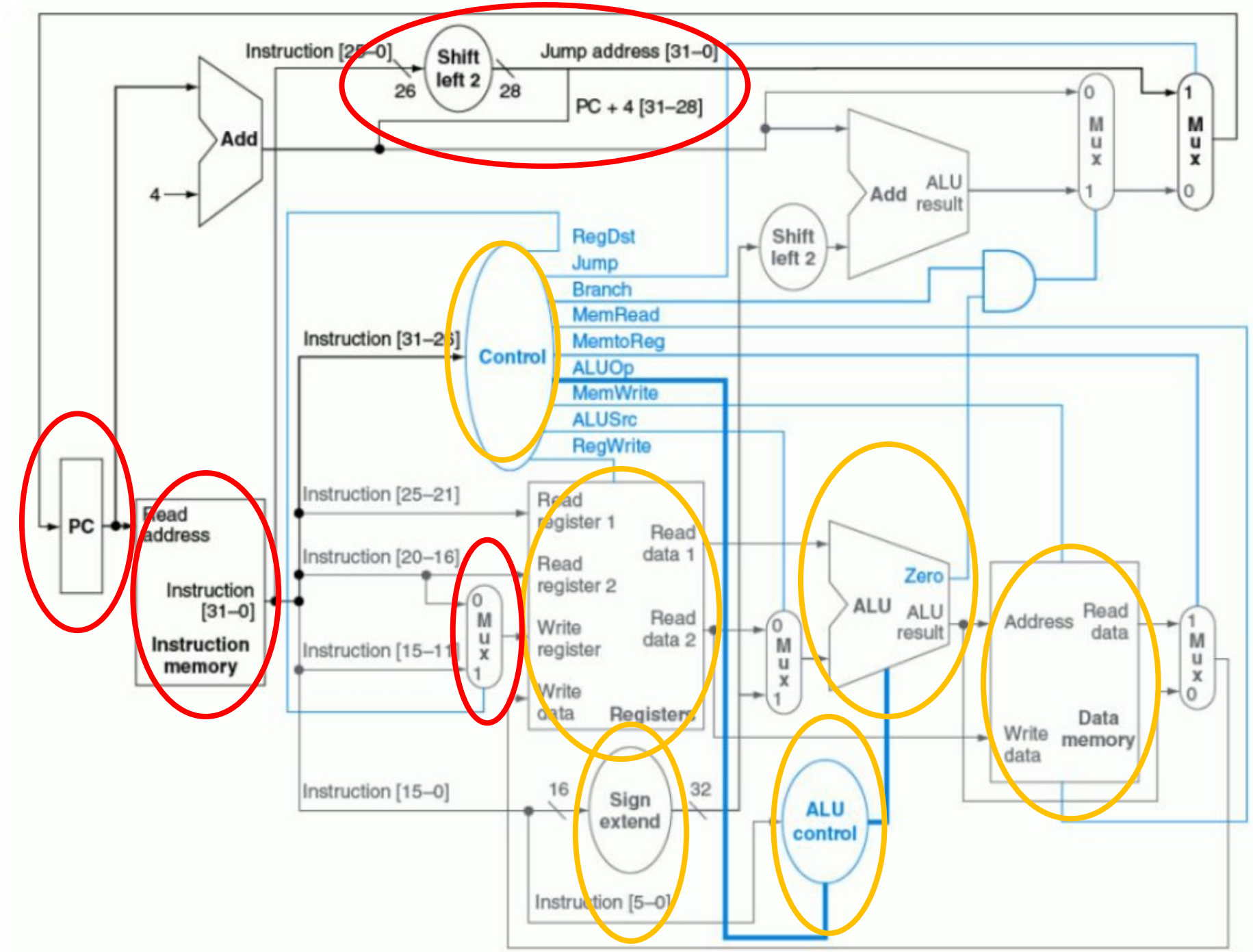
LAB05

2023.04.12

LAB05检查节点

1. 16条MIPS指令(lw, sw, beq, add, sub, and, or, slt, j，addi, andi, ori, sll, srl, jal, jr) CPU的实现与调试，按照助教给出的汇编乘法进行应用，乘法结果正确，得5分；
2. 利用switch、led、七段数码管等来观察指令运行结果是否预期，得5分。

LAB 05逻辑架构


Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC + 4 [31–28]
Add
4
M u x
0
1
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Instruction [31–26]
Control
Shift left 2
Add
ALU result
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign extend
32
ALU
Zero
ALU result
ALU control
Address
Read data
Write data
Data memory

子模块1：程序计数器PC

上升沿reset时PC为4，否则PC每次+4

```
module PC(
    input [31:0] PC,
    input reset,
    input clk,
    output  reg [31:0] newPC
    );
    always @ (posedge clk)
    begin
        if(reset)
            newPC <= 4;
        else
            newPC <= PC + 4;
    end
endmodule
```

子模块2：指令存储器InstMem

功能：时钟上升沿时，reset时，寄存器清零；非reset时，读取寄存器内容

对32位instruction做初始化时要求其**从文件中读取内容**
考虑用$readmemb:
$readmemb("<数据文件名>",<存储器名>)
如果文件file1.txt中存入二进制数据：1111 1010 0101 1x1z 1_1_ 1_111
或者
1111
1010
0101
1x1z
1_1_
1_111
存在一行每个用空格隔开，跟分行存，输出结果是一样的，但是若在一行中不用空格隔开会出错，编译器会试图把一整行数据存在一个存储单元中。

（接上页）readmemb的指令调用方式和结果

```
`timescale 10ns/1ns
module test;
reg[3:0] memory[0:7];//申请八个四位的存储单元
reg[4:0] n;
initial
    begin
        $readmemb("file1.txt",memory); //读取file1.txt中的数字到memory
    for(n=0;n<=7;n=n+1)   //把八个存储单元的数字都读取出来，若存的数不到八个单元输出x态，程序结果中会看到
        $display("%b",memory[n]);
    end
endmodule
```

编译，仿真，运行之后的输出结果：

```
# 1111
# 1010
# 0101
# 1x1z          不定态和高阻态输出依旧为不定态和高阻态
# 0011          文件中村的是1_1_，忽略下划线
# 1111          忽略下划线
# xxxx          文件中只有六个数据，剩下两个输出为不定态x
# xxxx
```

如果是读取十六进制的内容，
那么就用$readmemh

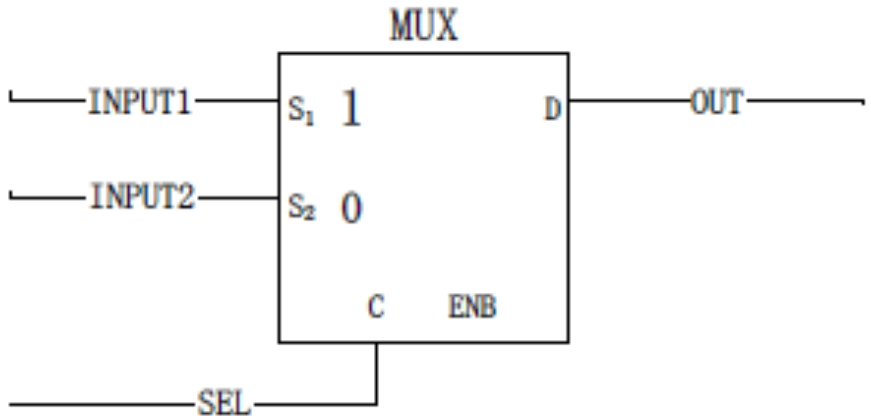
本LAB里readmemb的调用方法

按byte寻址，一个byte是2bits

```
initial begin
    $readmemb("Instruction.txt",instrMEM);
end
always @ (posedge clk)
// triggered at the positive edge of the clk
begin
    if(reset)
        instr <= instrMEM[0];
    else
        instr <= instrMEM[readAddr >> 2];
end
```

子模块3：多路选择器MUX

实现逻辑：assign data =select ? data1：data0



- 其中data1和data0既有无符号数拓展（前面补0）
- 也包括有符号数拓展（前面补最高位）

子模块4：地址更新PCupdate

对PC的修改有：beq、j、jr
Beq的逻辑： if($1==$2)
goto PC+4+40

J的逻辑：PC跳address/4
Jr的逻辑：PC<=rs

```
wire [27:0] shtarget;
assign shtarget = target << 2;
wire [31:0] jumptarget;
assign jumptarget = {newPC[31:28],shtarget};
wire [31:0] branchtarget;
assign branchtarget = newPC + offset;
wire [31:0] nonjumptarget;
assign nonjumptarget = (PCsrc) ? (branchtarget) : (newPC);
wire [31:0] nonretaddr;
assign nonretaddr = (Jump) ? (jumptarget) : (nonjumptarget);
assign nextPC = (ret) ? (retAddr) :(nonretaddr);
```

几个需要更新的模块

1. **Register模块**：添加readmemh从register.txt里面读取十六进制数据；添加reset时的register初始化
2. **DataMemory模块：**添加readmemh从data.txt里面读取十六进制数据；按字读取，每次要读4个字节
3. **Ctr控制模块：**查给的html里面的opcode，增加case描述即可
4. **ALU模块：**增加sll、srl、jr的描述

```
always@ (memRead or addr)
    begin
        readData = memFile[addr >> 2];
    end

always@ (negedge clk)
    begin
    if (memWrite)
        memFile[addr >> 2] = writeData;
    end
```

另一个更新的模块：ALUctr

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	and	0000
R-type	10	OR	100101	or	0001
R-type	10	set on less than	101010	set on less than	0111

1. 想要支持更多指令的区分，扩展ALUop从2位到3位，重新设计ALUop（**自行设计，只要能区分开不同指令就行，所有R-type的ALUop是一样的**）

2. ALUout也要补充nor、sll、slt、jr的编码（自己设计能区分即可）

3. 除了R-type，其他funct都是xxxxxx，这个也可以去查html

顶模块Top：

各模块实例化，将图中展示的逻辑相关联