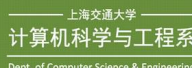




计算机组成实验指导书-LAB5

 上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5 作者 CETC	CSE-COA-LAB-005 修改日期 03/21/2020	0.5	1 of 13 公开

1. 概述

1.1 实验名称

类 MIPS 单周期处理器的设计与实现

1.2 实验目的

完成单周期的类 MIPS 处理器

设计支持 16 条 MIPS 指令（add, sub, and, or, addi, andi, ori, slt, lw, sw, beq, j, jal, jr, sll, srl,）的单周期 CPU

1.3 实验内容

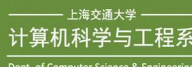
1. Instruction memory 等模块的设计
2. 单周期 CPU 的实现与调试
3. 功能仿真
4. 上板验证

1.4 实验预计时间

240~480 分钟

1.5 实验报告与验收办法

- 1) 实验报告和工程文件在第十二周星期五晚上 23 点前提交
- 2) 无法验收，但报告要体现原需检查指令运行的仿真结果

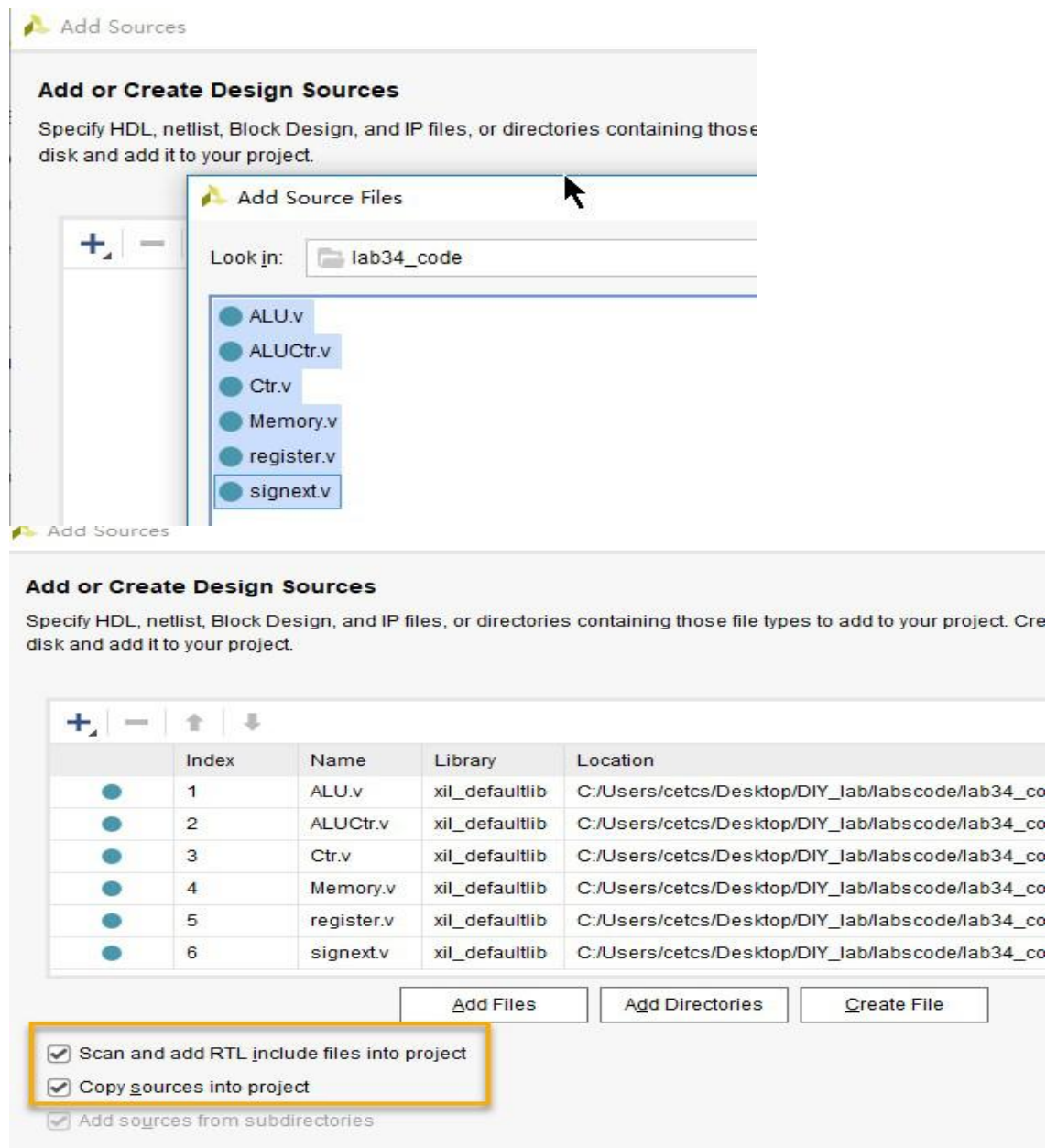
 上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5 作者 CETC	CSE-COA-LAB-005 修改日期 03/21/2020	0.5	2 of 13 公开

2. 新建工程，导入文件

2.1 实验描述

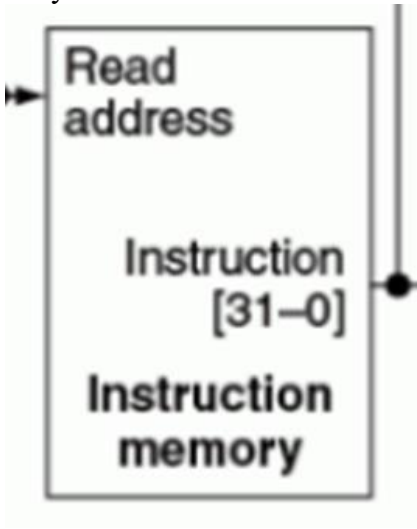
2.1.1 新建工程

1. 启动 Vivado
2. 新建工程 lab5
3. 将此前两次实验中的模块添加到 lab5 工程目录下



<div>上海交通大学</div> <div>计算机科学与工程系</div> <div>Dept. of Computer Science & Engineering</div>	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5	CSE-COA-LAB-005	0.5	3 of 13
	作者	修改日期	公开	
	CETC	03/21/2020		

4. 下图为单周期处理器结构图的指令存储器部分，可设计成模块文件来描述，可取名 InstMemory



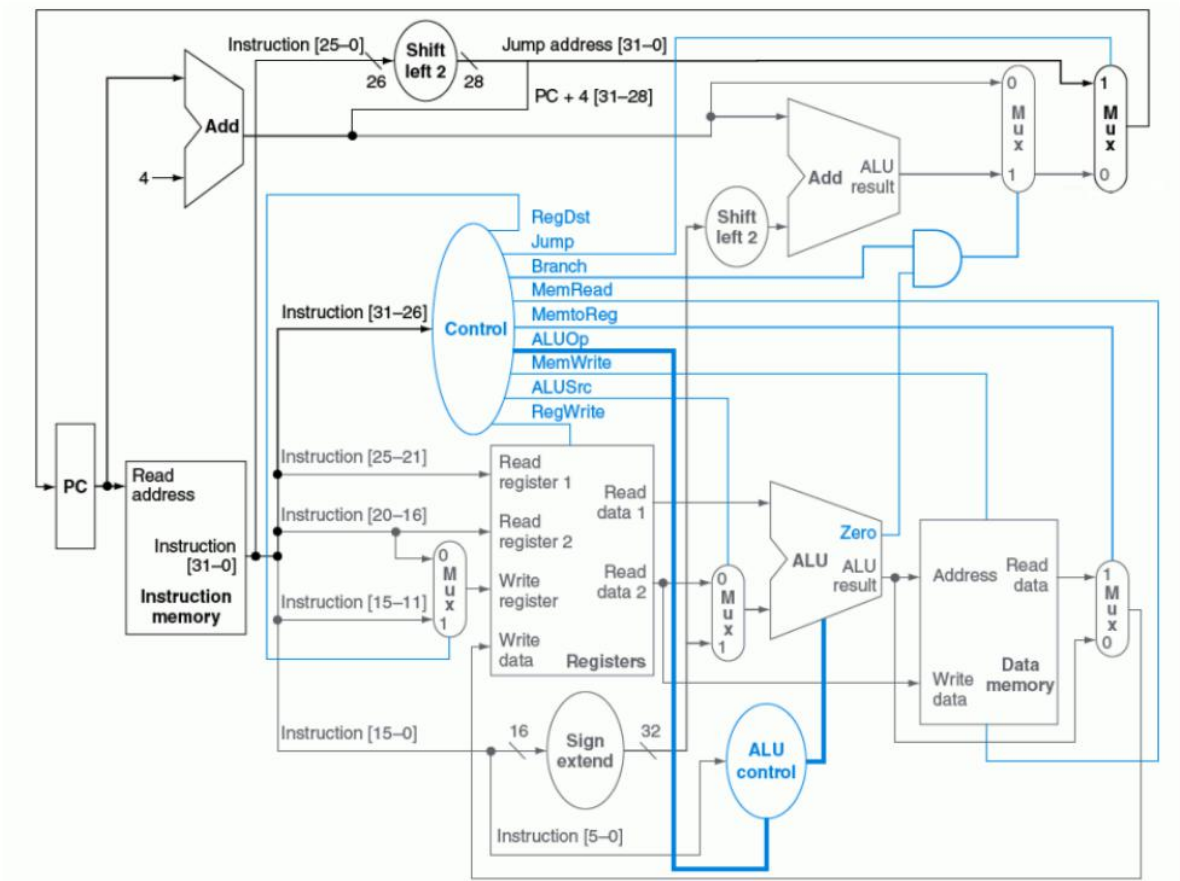
<div>上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering</div>	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5	CSE-COA-LAB-005	0.5	4 of 13
	作者	修改日期	公开	
	CETC	03/21/2020		

3. 顶层模块设计

3.1 实验描述

整体调试单周期类 MIPS 处理器。

3.1.1 模块描述



类 MIPS 单周期处理器原理图

3.1.2 新建顶层模块文件，可命名 Top

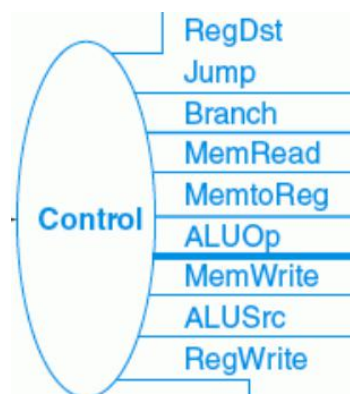
3.1.3 定义信号线

为 Top 模块内的每一根连接的信号线命名，并在 Top 模块中定义它们

<div>上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering</div>	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5	CSE-COA-LAB-005	0.5	5 of 13
	作者 CETC	修改日期 03/21/2020	公开	

例如，主控制模块输出端口上的连线：

```
47 wire REG_DST,  
48     JUMP,  
49     BRANCH,  
50     MEM_READ,  
51     MEM_TO_REG,  
52     MEM_WRITE;  
53 wire[1:0] ALU_OP;  
54 wire ALU_SRC,  
55     REG_WRITE;
```



3.1.4 程序计数器 PC

程序计数器是这个简单 CPU 能够跑起来的关键。定义一个 32 位 reg 类型的程序计数器 PC，在时钟上升沿(下降沿已经被我们用作寄存器的写了)做 $PC \leq PC + 4$ 。

PS: 1) 简单的讲，在组合逻辑中用阻塞赋值“=”，时序逻辑中用非阻塞赋值“<=”。两者综合出来的电路不一样，具体区别查阅参考书。时序逻辑和组合逻辑不要放在同一个 always 块中。

2) PC 也可以考虑设计成模块来实现

3.1.5 RESET

PC 置 0x00000000，各寄存器清零，这是 reset 要做的工作。同步或异步，边沿或电平，同学们可以自由实现。

寄存器清零，所以要适当修改上次实验的 register 模块，给模块添加 reset 信号。

PS: 添加 reset 要注意，写在原来“写”的 always 块中。假如新加一个 always 块，当两个“写”always 同时满足时，就混乱不知赋什么值了。

3.1.6 模块实例化，连接模块


实例化前两次实验中设计的模块，实例化的过程中连接模块的端口。实例化有以下两种方法：

1. 严格按照模块定义时的端口顺序来连接，不用表明原模块定义时规定的端口名：

模块 模块名(连接端口 1 信号名, 连接端口信号名 2...)

2. 在连接时用“.”符号，表明原模块是定义时规定的端口名：

模块 模块名(.端口 1 名(信号 1), .端口 2 名(信号 2))

 上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5	CSE-COA-LAB-005	0.5	6 of 13
	作者 CETC	修改日期 03/21/2020	公开	

实验中推荐用第 2 种实例化方法。

以主控制模块为例，以下代码为实例化 Ctr（mainCtr 就是模块名），并连接其端口。其中 INST 是定义好的指令存储器输出的连接信号，其它信号线在 3.1.3 中已完成定义。

```
100      Ctr mainCtr(  
101          .opcode(INST[31:26]),  
102          .regDst(REG_DST),  
103          .jump(JUMP),  
104          .branch(BRANCH),  
105          .memRead(MEM_READ),  
106          .memToReg(MEM_TO_REG),  
107          .aLUOp(ALU_OP),  
108          .memWrite(MEM_WRITE),  
109          .aLUSrc(ALU_SRC),  
110          .regWrite(REG_WRITE));  
...
```

实例化 Ctr

```
21 module Ctr(opcode, regDst, jump, branch, memRead, memToReg, aLUOp, memWrite, aLUSrc, regWrite);
```

Ctr 模块定义

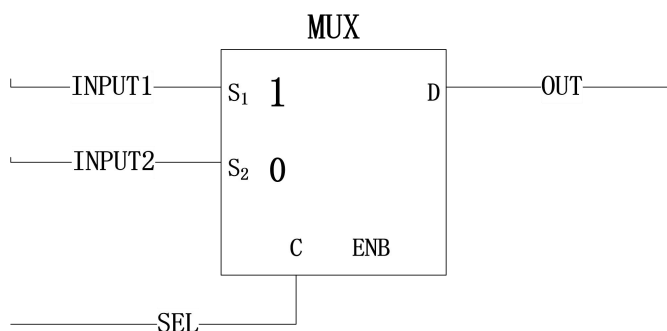
3.1.7 连接其它信号线

1. MUX


Mux 已经在前几次实验中提到，实现很简单，一个三目运算符就行

Assign $OUT = SEL ? INPUT1 : INPUT2;$

OUT, SEL, INPUT1, INPUT2 都是预先定义好的信号



2. 左移两位，用移位运算符：左移("<<"), 右移(">>")

 上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5	CSE-COA-LAB-005	0.5	7 of 13
	作者 CETC	修改日期 03/21/2020	公开	


3. 加法器，直接用无符号加法运算

注：verilog 中寄存器类型被解释成无符号数，整数类型(integer)被解释成二进制补码形式的有符号数。因此要综合成无符号算术算符需要使用寄存器类型，而要得到有符号算术算符就需要使用整数。网线类型(wire)被解释成无符号数。

4. 与门，使用位运算符&(位与)。注意&和&&的区别

5. 考虑到本实验的指令（含类型）较之前的实验有所增加，之前的相关模块可能要适当修改下

6. 程序计数器(PC)也可考虑通过模块设计来实现，当然多路选择器(MUX)也可设计成模块实现

 上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5 作者 CETC	CSE-COA-LAB-005 修改日期 03/21/2020	0.5	8 of 13 公开

4. 仿真

1. 编写二进制测试程序

请仔细编写自己的测试汇编。下面提供一个简易汇编器供参考。

一些相关的基本知识：

指令格式：

R	opcode	rs	rt	rd	shamt	funct
	31 26	25 21	20 16	15 11	10 6	5 0
I	opcode	rs	rt	immediate		
	31 26	25 21	20 16	15		0
J	opcode	address				
	31 26	25				0

Mips 基本指令格式

汇编格式：注意汇编中寄存器的顺序跟指令格式中的不一样

```
add $1,$2,$3      : $1=$2 + $3
sub $1,$2,$3      : $1=$2 - $3
and $1,$2,$3      : $1=$2 & $3
or $1,$2,$3       : $1=$2 | $3
slt $1,$2,$3      : if($2<$3) $1=1 else $1=0
lw $1,10($2)      : $1=memory[$2+10]
sw $1,10($2)      : memory[$2+10]=$1
beq $1,$2,10       : if($1==$2) goto PC+4+40
j 10000           : goto 10000
```

[10 是 PC+4 后的指令间隔数，故为 PC+4+40]

2. Verilog 的系统任务 \$readmemb 和 \$readmemh，放在 initial 初始化块中。

Verilog 中这两个系统任务用来从文件中读取数据到存储器中，格式如下：

1) 相对路径：

```
$readmemh("Instruction", InstMemFile);
```

PS:文件 Instruction 必须放在本工程当前目录里

上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5	CSE-COA-LAB-005	0.5	9 of 13
	作者	修改日期	公开	
	CETC	03/21/2020		

1
2
3
4
5
6
7
8
9
1
1
1
1
1
1
1
1
1
2
2
2
2
2
2
2
2
2
2
3
3
3
3
3
3
3
3
3
3
4
4
4
4

```
$readmemh("D:/Archlab/Lab05/Lab005.srsc /Src/Data", MemFile,10'h0);
```

这里给出 `mem_data` 和 `mem_inst` 两个样例文件，分别用来装入 data memory 和 instruction memory。

mem_data

1	00000000
2	00000001
3	00000002
4	00000003
5	00000004
6	00000005
7	00000006
8	00000007
9	00000008

• • • • •

mem inst

```

1      0000100000000000000000000000000000000000 // j a
2      0000000000000000000000000000000000000000 // nop
3      0000000000000000000000000000000000000000 // nop
4      0000000000000000000000000000000000000000 // nop
5      1000110000000000100000000100010100        // lw $t1, 276($0)
6      1000110000000000100000000100001000        // lw $t2, 264($0)
7      000000000000100010000110000001000000        // add $t3, $t1, $t2
8      000000000000100010001000000001000100        // sub $t4, $t1, $t2

```

• • • • •



3. 其他常用系统任务(可以不用)

\$monitor 提供监控和输出参数列表中的表达式或变量值的功能。

如\$monitor(“rxd=%b”, rxb);

\$time 返回当前仿真时刻值。

如\$monitor(\$time);

 上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5	CSE-COA-LAB-005	0.5	10 of 13
 CETC	作者	修改日期		
		03/21/2020	公开	

4. 整个处理器设计基本完成后，编辑激励文件，进行仿真测试：

- 1) 初始化数据存储器、指令存储器和寄存器三大存储模块，这里仅以初始化数据存储器为例明，其它类推。如下图，表明系统任务 \$readmemh 将 Data 文件中的数据读入到 DataMem 数组中。

```
29  
30 reg [31:0] DataMem [0:127];  
31 //initial the instruction and data memory  
32 initial  
33 begin  
34 $readmemh("Data",DataMem); //相对路径格式  
35 end  
36
```

- 2) 添加时钟激励和其它输入信号并给定初始值等

- 3) 按需要添加寄存器模块中的 regfile 数组以及其它需要观察的变量或信号到仿真波形窗口，查看各个相关数值的变化情况。

- 4) 添加 register 模块中的 regfile 寄存器数组到仿真波形窗口,观察各个寄存器的变化情况。

- 5) 也可在 Console 窗口中先输入 restart 命令，在输入 run 2000ns 命令，以便重新进行仿真。将仿真波形与你设计的 MIPS 指令运行结果进行对比.....

5. 下面给出一个仿真样例：

上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5	CSE-COA-LAB-005	0.5	11 of 13
	作者	修改日期	公开	
	CETC	03/21/2020		




上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5	CSE-COA-LAB-005	0.5	12 of 13
	作者 CETC	修改日期 03/21/2020	公开	

5. 上板调试（暂不做）

注：

1. 利用 switch、led 或者是 pushbutton 等 来观察指令运行结果是否预期
2. 可参考 lab2 相关外设上板验证的方法。

6. 实验报告

 上海交通大学 计算机科学与工程系 Dept. of Computer Science & Engineering	标题	文档编号	版本	页
	计算机系统结构实验指导书 LAB5 作者 CETC	CSE-COA-LAB-005 修改日期 03/21/2020	0.5	13 of 13 公开