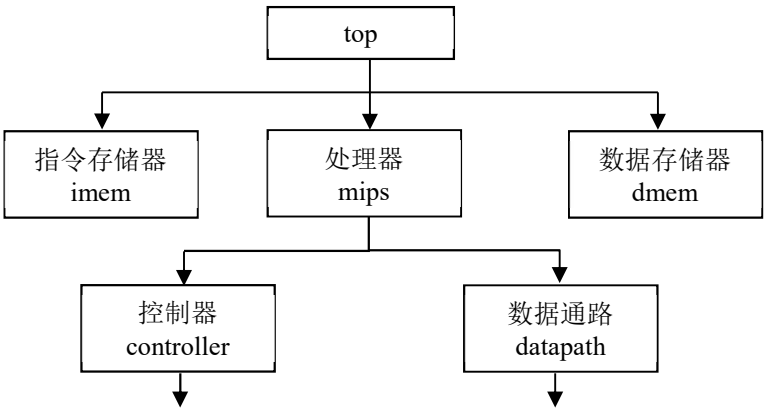
实验8　MIPs 单周期处理器设计与测试

预习内容

1. 补充完成下图所示的处理器层次化结构图



2. 表 1 是基准测试程序中的指令，其机器码被存在 memfile.dat 中。请填写表 1 中每条指令执行的结果。

表 1 基准测试程序指令

标号	汇编指令	执行结果描述	指令地址	机器码
main	addi $s2,$0,5	初始化$s2=0x00008000	0	20020005
	addi $s3,$0,12		4	2003000c
	addi $s7,$s3,-9		8	2067fff7
	or $s4,$s7,$s2		c	00e22025
	and $s5, $s3, $s4		10	00642824
	add $s5, $s5, $s4		14	00a42820
	beq $s5,$s7,end		18	10a7000a
	slt $s4,$s3,$s4		1c	0064202a
	beq $s4,$0,around		20	10800001
	addi $s5, $0, 0		24	20050000
around	slt $s4, $s7, $s2		28	00e2202a
	add $s7,$s4,$s5		2c	00853820
	sub $s7,$s7,$s2		30	00e23822
	sw $s7,68($0)		34	ac670044
	lw $s2,80($0)		38	8c020050
	j end		3c	08000011
	addi $s2,$0,1		40	20020001
end	sw $s2,84($0)		44	ac020054

3. 根据第 7 章 7.3 的原理，填写完成表 2 中基准测试程序执行前 16 个周期时，每条指令执行后相关信号的值。

表 2 基准测试程序前 16 个周期的执行结果

clk	reset	pc	instr	SrcA	SrcB	ALU Control	ALU Reslut	PCSrc	Write Data	mem write	Read Data
1	1	00	`addi $s2,$0,5`	0	5	010	5	0	0	0	x
2	0	04	`addi $s3,$0,12`	0	12	010	12	0	0	0	x
3	0	08	`addi $s7,$s3,-9`	c	-9	010	3	0	0	0	x
4	0	0C									
5											
6											
7											
8											
9											
10											
11											
13											
13											
14											
15											
16											

实验内容

一、基础部分

1. 参考教材 7.6 节的代码，结合第 5 章作业所描述的 ALU，用 Verilog HDL 设计单周期 MIPs 处理器；

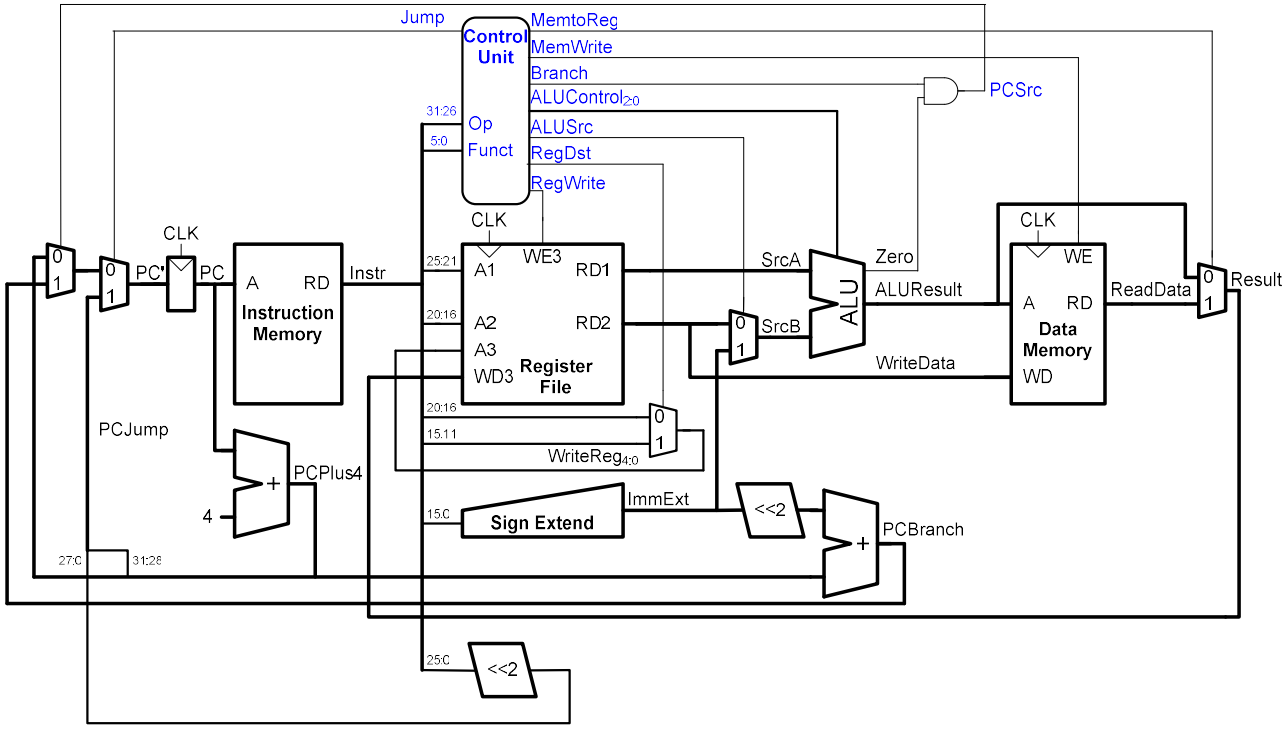
2. 使用基准测试代码和数据文件测试所设计的 MIPs 处理器；

二、提高部分

1. 修改单周期 MIPS 处理器以实现两条新指令 **`ori`** 和 **`bne`**；

2. 修改测试代码以测试新的指令是否能正确工作。

特别提示：实验开始前，请认真学习教材 7.3 节有关 MIPs 单周期处理器内容，电路原理图如图所示。



实验步骤

一、基础部分

1. MIPS 单周期处理器设计

- 学习教材 7.6 节中 System Verilog 描述的 MIPS 单周期处理器 mips.sv 及其相关模块（除 alu 外），该处理器可以正确执行以下指令：`add`, `sub`, `and`, `or`, `slt`, `lw`, `sw`, `beq`, `addi`, `j`。
- 参照 7.6 节的 mips.sv，用 Verilog HDL 完成自己的 mips.v。mips.v 中有 2 个子模块：controller 和 datapath，可以正确执行以下指令：`add`, `sub`, `and`, `or`, `slt`, `lw`, `sw`, `beq`, `addi`, `j`。
- controller 有子模块 maindec 和 aludec，maindec 模块产生除 ALU 之外的所有控制信号，aludec 模块产生 ALU 控制信号 alucontrol[2:0]。
- datapath 模块由多个小模块构成，请对照上面的原理图，理解每个模块在 datapath 中的位置、作用和与其它模块之间的接口。注意，需要将第 5 章作业的 ALU 复制到你的工程文件夹中，将模块名修改成 alu，并确保输入和输出端口的顺序与 datapath 调用的顺序一致。
- 顶层模块 `top`, 包括处理器 mips、指令存储器 imem 和数据存储器 dmem， 2 个存储器的容量均为 64-word × 32-bit。 top、imem 和 dmem 模块的代码如下：

```
//顶层模块
module top  (input        clk, reset,
             output [31:0]  writedata, dataadr,
             output        memwrite);
  wire [31:0] pc, instr, readdata;
  // instantiate processor and memories
  mips mips (clk, reset, pc, instr, memwrite, dataadr, writedata, readdata);
  imem imem (pc[7:2], instr);
  dmem dmem (clk, memwrite, dataadr, writedata, readdata);
endmodule

//数据存储器模块
module dmem  (input       clk, we,
                  input [31:0]   a, wd,
                  output [31:0]  rd);
  reg [31:0] RAM[63:0];
  assign rd = RAM[a[31:2]];  // word aligned
  always @ (posedge clk)
    if (we)  RAM[a[31:2]] <= wd;
endmodule

//程序存储器模块
module imem (input [5:0]    a,
             output [31:0]  rd);
  reg [31:0] RAM[63:0];
  initial
    begin
      $readmemh ("memfile.dat", RAM);
    end
  assign rd = RAM[a];  // word aligned
endmodule
```

- 编译、综合电路。注意如果 warning 数太多要仔细检查代码，否则无法得出测试结果.

2. MIPs 处理器测试

- testbench 代码如下，请认真学习并理解这段代码。

```
module testbench();
  reg clk;
  reg reset;
  wire [31:0] writedata, dataadr;
  wire memwrite;
  // instantiate device to be tested
  top dut (clk, reset, writedata, dataadr, memwrite);
  // initialize test
  initial
    begin
      reset <= 1; # 22; reset <= 0;
    end
  // generate clock to sequence tests
  always
    begin
      clk <= 1; # 5; clk <= 0; # 5;
    end
  // check results
  always @ (negedge clk)
    begin
      if (memwrite) begin
        if (dataadr === 84 & writedata === 7) begin
          $display ("Simulation succeeded");
          $stop;
        end else if (dataadr !== 80) begin
          $display ("Simulation failed");
          $stop;
        end
      end
    end
endmodule
```

- 基准测试程序的机器码存在memfile.dat数据文件中。
  初始化时，指令存储器模块imem中的“ $readmemh ("memfile.dat", RAM);” 语句将把memfile.dat读入到imem中。
  testbench 和基准测试程序数据文件 memfile 可以从课程 QQ 群中下载。

- 用 ModelSim 测试 MIPs 处理器。确保加入处理器的所有.v 文件（testbench, top, mips 等），并在 waves window 中加入表 2 中列出的所有信号。

  注意，很多信号并不在 top 模块，需要从底层模块层次化引出到 top 模块；表 2 中列出的信号名可能与你代码中信号名不完全相同，请对照上面的电路图确定是哪些信号。

- 运行仿真。如果代码正确，测试代码将会输出“Simulation succeeded”； 如果代码有问题，将会输出“Simulation failed”。检查输出波形与表 2 中的信号值是否一样，如果不一样，有可能是电路连接有误，或者是没有把全部.v 文件加入。如果要进一步查找错误，需要检查更多的内部信号。

- 实验报告中输出波形必须按以下顺序显示信号：

  `clk, reset, pc, instr, branch, srca, srcb, aluout, zero, pcsrc, writedata,`

memwrite, readdata

所有的信号值必须用十六进制表示，并且清晰可读，否则将会扣分。

二、提高部分

1. 改进 MIPS 处理器

增加 2 条新指令： **ori** and **bne**。

- **修改** MIPS 处理器的原理图，可以在上面的原理图上直接画出你的改进。
- 按照要求修改主译码器，把修改内容填入后面的表 3 和表 4 中。
- 根据需要修改 Verilog 代码。

2. 测试改进后的 MIPs 处理器

- 使用下面的 MIPs 汇编指令，测试改进后的处理器，以验证新加入的指令是否能正确执行。

  这些 MIPs 汇编指令的机器码存到文件名为 memfile2.dat 中。

```
# Test MIPS instructions.
# Assembly Code
main:       ori  $t0, $0,  0x8000
            addi $t1, $0,  -32768
            ori  $t2, $t0, 0x8001
            beq  $t0, $t1, there
            slt  $t3, $t1, $t0
            bne  $t3, $0,  here
            j    there
here:       sub  $t2, $t2, $t0
            ori  $t0, $t0, 0xFF
there:      add  $t3, $t3, $t2
            sub  $t0, $t2, $t0
            sw   $t0, 82($t3)
```

- 修改 testbench，当汇编程序执行结束得到正确的数据和地址时输出“simulation succeeded”，如果结果错误，输出“simulation failed”。
- 运行仿真，实验报告中输出波形必须按以下顺序显示信号：

  clk, reset, pc, instr,branch, srca, srcb, aluout, zero, pcsrc, writedata, memwrite, readdata

  注意：所有的信号值必须用十六进制表示，并且清晰可读，否则将会扣分。

实验报告要求：

请严格按照以下顺序提交下述各项内容，否则将会扣分：

一、MIPS 单周期处理器

1. MIPS 单周期处理器的 Verilog 代码，包括 ALU。必须有清楚的注释，否则将会被扣分；
2. 编译报告截图（检查编译是否通过，注意如果 warning 数太多要仔细检查代码，否则无法得出测试结果）；
3. 所有含有子模块的模块的 RTL Viewer 截图（检查层次化调用是否正确）；
4. ModelSim 仿真输出波形，必须按以下顺序显示信号：

   `clk, reset, pc, instr,branch, srca, srcb, aluout, zero, pcsrc, writedata, memwrite, readdata`

   所有的信号值必须用十六进制表示，并且清晰可读（必要时可用多个图），否则将会扣分。

二、 MIPS 单周期处理器改进

1. 加入 `ori` 和 `bne` 指令后的处理器原理图（如果有修改）；
2. 主译码器和 ALU 译码器功能表（表 3、表 4）；
3. 改进后的 MIPS 处理器 Verilog 代码和注释（只需要给出有改进的模块和新加入的模块）；
4. 参考教材附录 B 和 6.2 节、6.3 节内容，填写表 5 中每条指令的执行结果。
5. 修改后的 testbench 代码。
6. ModelSim 仿真输出波形，必须按以下顺序显示信号：

   `clk, reset, pc, instr,branch, srca, srcb, aluout, zero, pcsrc, writedata, memwrite, readdata`

   所有的信号值必须用十六进制表示，并且清晰可读（必要时可用多个图），否则将会扣分。

表 3 功能扩展（主译码器）

Instruction	$Op_{5:0}$	RegWrite	RegDst	AluSrc	Branch	MemWrite	MemtoReg	$ALUOp_{1:0}$	Jump			
R-type	000000	1	1	0	0	0	0	10	0			
lw	100011	1	0	1	0	0	1	00	0			
sw	101011	0	X	1	0	1	X	00	0			
beq	000100	0	X	0	1	0	X	01	0			
addi	001000	1	0	1	0	0	0	00	0			
j	000010	0	X	X	X	0	X	XX	1			
ori	001101											
bne	000101											

表 4 功能扩展（ALU 译码器）

$ALUOp_{1:0}$	Meaning
00	Add
01	Subtract
10	Look at funct field
11	

表 5 MIPs 扩展测试指令

标号	汇编指令	执行结果描述	地址	机器码
main	ori $t0,$0,0x8000	初始化$t0=0x00008000	0	34088000
	addi $t1,$0,-32768		4	20098000
	ori $t2,$t0,0x8001		8	350a8001
	beq $t0, $t1, there		c	11090005
	slt $t3, $t1, $t0		10	0128582a
	bne $t3, $0, here		14	15600001
	j there		18	08100009
here	sub $t2, $t2, $t0		1c	01485022
	ori $t0, $t0, 0xFF		20	350800ff
there	add $t3, $t3, $t2		24	016a5820
	sub $t0, $t2, $t0		28	01484022
	sw $t0, 82($t3)		2c	ad680052