浙 江 大 學

本科生实验报告

课程名称 计算机体系结构

姓　　名 吴同

学　　院 计算机科学与技术学院

专　　业 计算机科学与技术

学　　号 **3170104848**

指导教师 陈文智

浙江大学实验报告

专业：计算机科学与技术
姓名：吴同
学号：3170104848
日期：2019 年 11 月 27 日
地点：曹西 301

课程名称：计算机体系结构　指导老师：陈文智　电子邮件：wutongcs@zju.edu.cn
实验名称：支持 31 条指令的流水线 CPU　实验类型：综合型　同组同学：徐欣苑

一、 实验目的和要求

1. 实验目的

- 理解 31 条指令的格式和执行效果
- 掌握执行 31 条指令的流水线 CPU 设计
- 掌握执行 31 条指令的流水线 CPU 的程序验证

2. 实验要求

- 设计执行 31 条指令的流水线 CPU 的数据通路和控制器
- 使用程序验证 CPU 并观察程序的执行

二、 实验内容和原理

1. 实验内容

本次实验在原有流水线 CPU 的基础上，增加以下指令。

inst	31...26	25...21	20...16	15...11	10...6	5...0	operation
addu	000000	rs	rt	rd	00000	100001	$rd = $rs + $rt; PC += 4
subu		rs	rt	rd	00000	100011	$rd = $rs - $rt; PC += 4
xor		rs	rt	rd	00000	100110	$rd = $rs ^ $rt; PC += 4
nor		rs	rt	rd	00000	100111	$rd = ~($rs | $rt); PC += 4
sltu		rs	rt	rd	00000	101010	$rd = ($rs < $rt) ? 1 : 0; PC += 4
sll		00000	rt	rd	sa	000000	$rd = $rt < < $sa; PC += 4
srl		00000	rt	rd	sa	000010	$rd = $rt > > $sa; PC += 4
sra		00000	rt	rd	sa	000011	$rd = $rt > > > $sa; PC += 4
sllv		rs	rt	rd	00000	000100	$rd = $rs < < $rt; PC += 4
srlv		rs	rt	rd	00000	000110	$rd = $rs > > $rt; PC += 4
srav		rs	rt	rd	00000	000111	$rd = $rs > > > $rt; PC += 4
addiu	001001	rs	rt	imm			$rt = $rs + (sign)imm; PC += 4
xori	001110	rs	rt	imm			$rt = $rs ^ (zero)imm; PC += 4
lui	001111	00000	rt	imm			$rt = $imm < < 16; PC += 4
slti	001010	rs	rt	imm			$rt = ($rs < (sign)$imm) ? 1 : 0; PC += 4
sltiu	001011	rs	rt	imm			$rt = ($rs < (zero)$imm) ? 1 : 0; PC += 4

2. 实验原理

本次实验对数据通路做出微小的修改，以支持 sll、srl、sra 指令。其他指令通过修改控制器的逻辑即可添加。

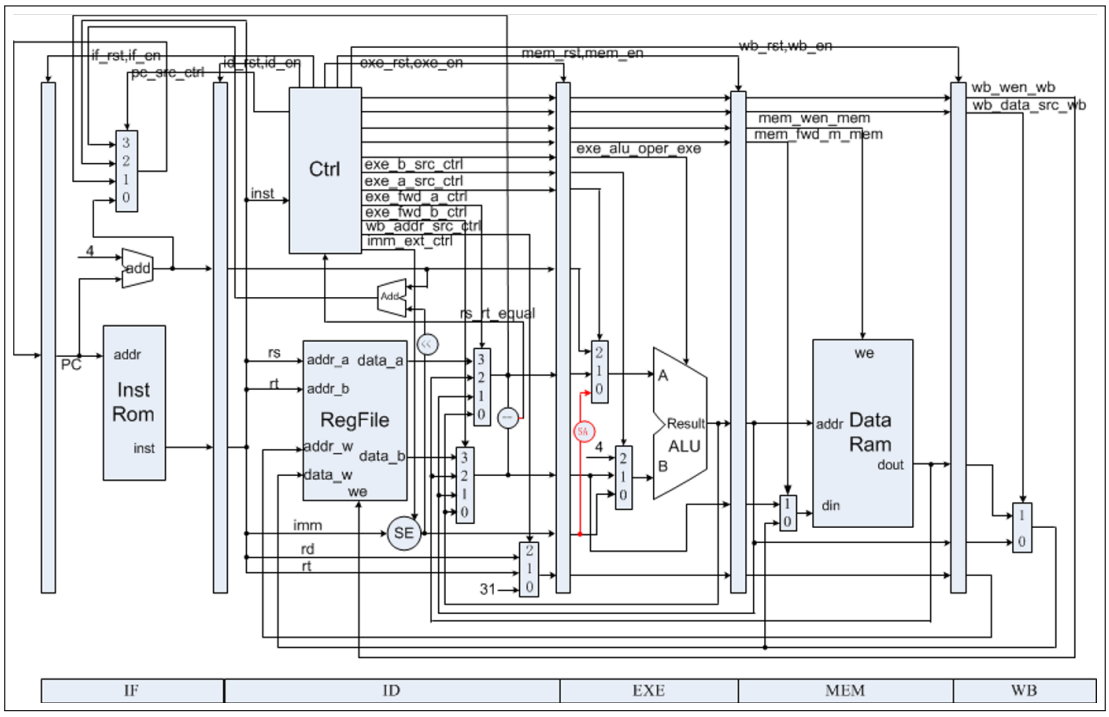

图 1: 支持 31 条指令的数据通路图

三、 实验过程记录

由于本实验中不对 overflow 进行处理，所以 addu、subu、sltu、addiu 的控制逻辑分别与 add、sub、slt、addi 相同。

xor 和 nor 的控制逻辑与 or 相似，xori 的控制逻辑与 ori 的相似，同时在 ALU 内增加相应的运算单元。

sll、srl、sra 三条移位指令的控制逻辑为：

```
R_FUNC_SLL: begin
   exe_alu_oper = EXE_ALU_SL;
   exe_a_src = EXE_A_SA;
   wb_addr_src = WB_ADDR_RD;
   wb_data_src = WB_DATA_ALU;
   wb_wen = 1;
   rt_used = 1;
end
R_FUNC_SRL: begin
   exe_alu_oper = EXE_ALU_SR;
   exe_a_src = EXE_A_SA;
   wb_addr_src = WB_ADDR_RD;
   wb_data_src = WB_DATA_ALU;
   wb_wen = 1;
   rt_used = 1;
end
R_FUNC_SRA: begin
```

```
      exe_alu_oper = EXE_ALU_SR;
      exe_a_src = EXE_A_SA;
      wb_addr_src = WB_ADDR_RD;
      wb_data_src = WB_DATA_ALU;
      wb_wen = 1;
      rt_used = 1;
      sign = 1;
   end
```

sllv、srlv、srav 三条指令的控制逻辑为：

```
   R_FUNC_SLLV: begin
      exe_alu_oper = EXE_ALU_SL;
      wb_addr_src = WB_ADDR_RD;
      wb_data_src = WB_DATA_ALU;
      wb_wen = 1;
      rs_used = 1;
      rt_used = 1;
   end
   R_FUNC_SRLV: begin
      exe_alu_oper = EXE_ALU_SR;
      wb_addr_src = WB_ADDR_RD;
      wb_data_src = WB_DATA_ALU;
      wb_wen = 1;
      rs_used = 1;
      rt_used = 1;
   end
   R_FUNC_SRAV: begin
      exe_alu_oper = EXE_ALU_SR;
      sign = 1;
      wb_addr_src = WB_ADDR_RD;
      wb_data_src = WB_DATA_ALU;
      wb_wen = 1;
      rs_used = 1;
      rt_used = 1;
   end
```

lui 指令的控制逻辑为：

```
   INST_LUI: begin
      exe_b_src = EXE_B_IMM;
      exe_alu_oper = EXE_ALU_LUI;
      wb_addr_src = WB_ADDR_RT;
      wb_data_src = WB_DATA_ALU;
      wb_wen = 1;
   end
```

slti、sltiu 指令的控制逻辑为：

```
   INST_SLTI: begin
      exe_alu_oper = EXE_ALU_SLT;
      wb_addr_src = WB_ADDR_RT;
      wb_data_src = WB_DATA_ALU;
      wb_wen = 1;
```

```
      rs_used = 1;
      imm_ext = 1;
      exe_b_src = EXE_B_IMM;
      sign = 1;
   end
   INST_SLTIU: begin
      exe_alu_oper = EXE_ALU_SLT;
      wb_addr_src = WB_ADDR_RT;
      wb_data_src = WB_DATA_ALU;
      wb_wen = 1;
      rs_used = 1;
      imm_ext = 1;
      exe_b_src = EXE_B_IMM;
   end
```

四、 实验结果分析

本次实验的测试程序进行了修改，修改后的程序如下：

```
lui $1, 0
ori $4, $1, 80
jal 0x001b
addi $5, $0, 4
sw $2, ($4)
lw $9, 0($4)
sub $8, $9, $4
addi $5, $0, 3
addi $5, $5, -1
ori $8, $5, 0xffff
xori $8, $8, 0x5555
addi $9, $0, -1
andi $10, $9, 0xffff
or $6, $10, $9
xor $8, $10, $9
and $7, $10, $6
beq $5, $0, 0x0003
nop
j 0x0008
nop
addi $5, $0, -1
sll $8, $5, 15
sll $8, $8, 16
sra $8, $8, 16
srl $8, $8, 15
j 0x0019
nopadd $8, $0, $0
lw $9, 0($4)
add $8, $8, $9
addi $5, $5, -1
bne $5, $0, 0xfffc
addi $4, $4, 4
jr $ra
```

```
34    sll $2, $8, 0
```

仿真 1800ns，即 90 个时钟周期，结果如下：

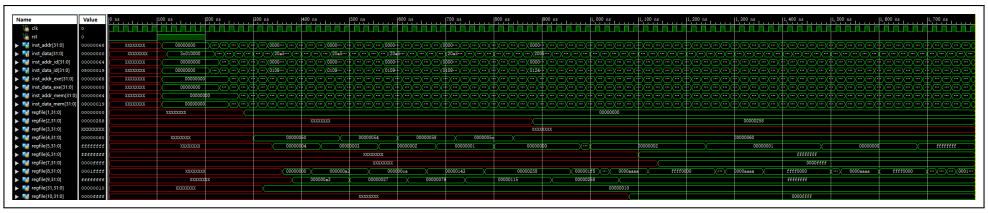

图 2: 仿真结果图 (0 ∼ 1800ns)

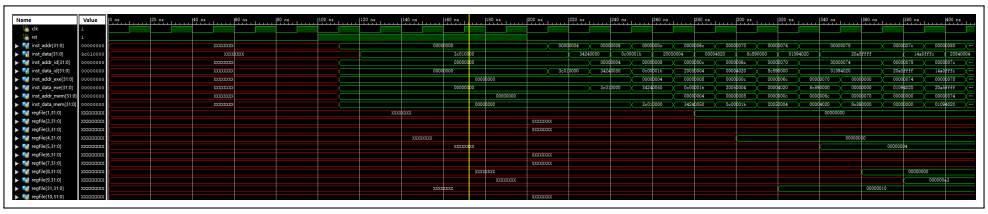

图 3: 仿真结果图 (0 ∼ 400ns)

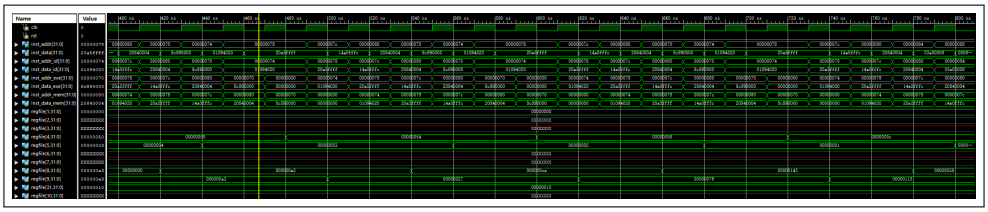

图 4: 仿真结果图 (400 ∼ 800ns)

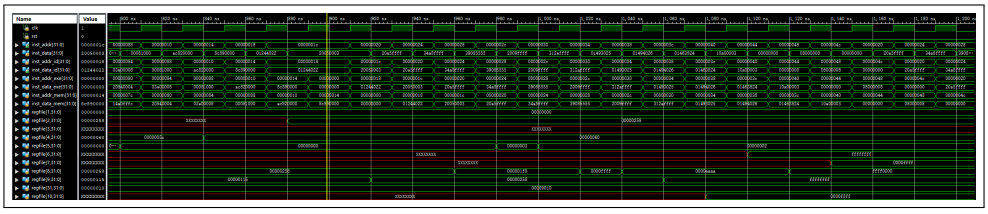

图 5: 仿真结果图 (800 ∼ 1200ns)

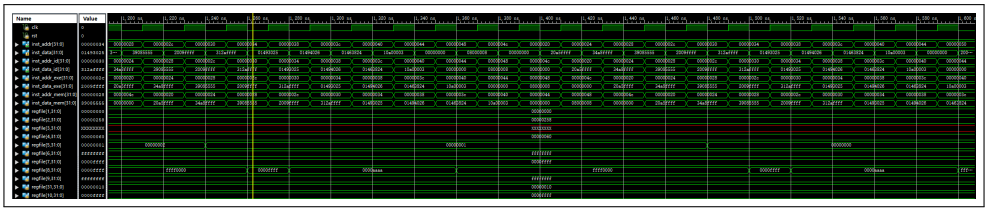

图 6: 仿真结果图 (1200 ∼ 1600ns)

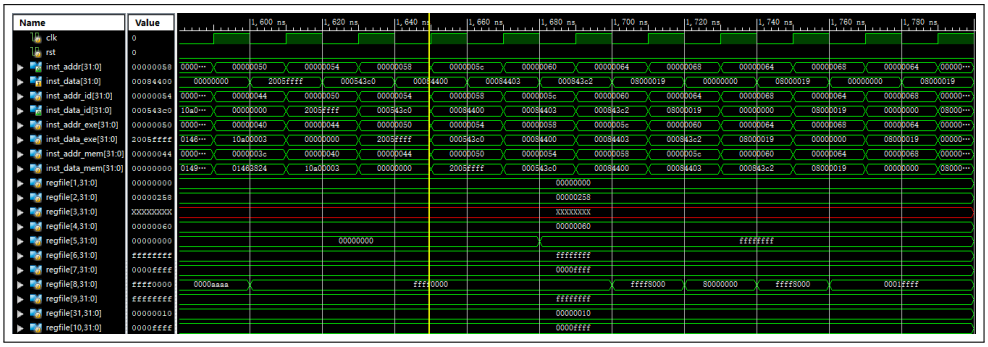

图 7: 仿真结果图 (1600 ∼ 1800ns)

仿真验证通过后，将 CPU 与外设进行综合，下载到 SWORD 板中，测试结果符合预期。

五、 讨论与心得

本次实验较为简单，在前序实验的基础上，增加 CPU 支持的指令。本次实验的数据通路只进行了微小的修改，控制器的修改也都是相似的逻辑。在修改 ALU 的过程中，我学会了 Verilog HDL 中的 unsigned 和 signed 类型转换，并学会了算术右移这一运算符。