实 验 报 告

学号：2014K8009929022　姓名：孔静　专业：计算机科学与技术

实验序号：4　实验名称：多周期处理器设计（下）

一、　代码以及波形图

此次代码是在实验 3 的基础上增加指令，所以该部分就按增加的指令分写相应代码以及波形图。

图 1addiu/li 图 2addu/move 图 3beq/beqz 图 4bnez 图 5j 图 6jal 图 7jr 图 8lui 图 9sll(nop)图 10slti 图 11sltiu 图 12subu。

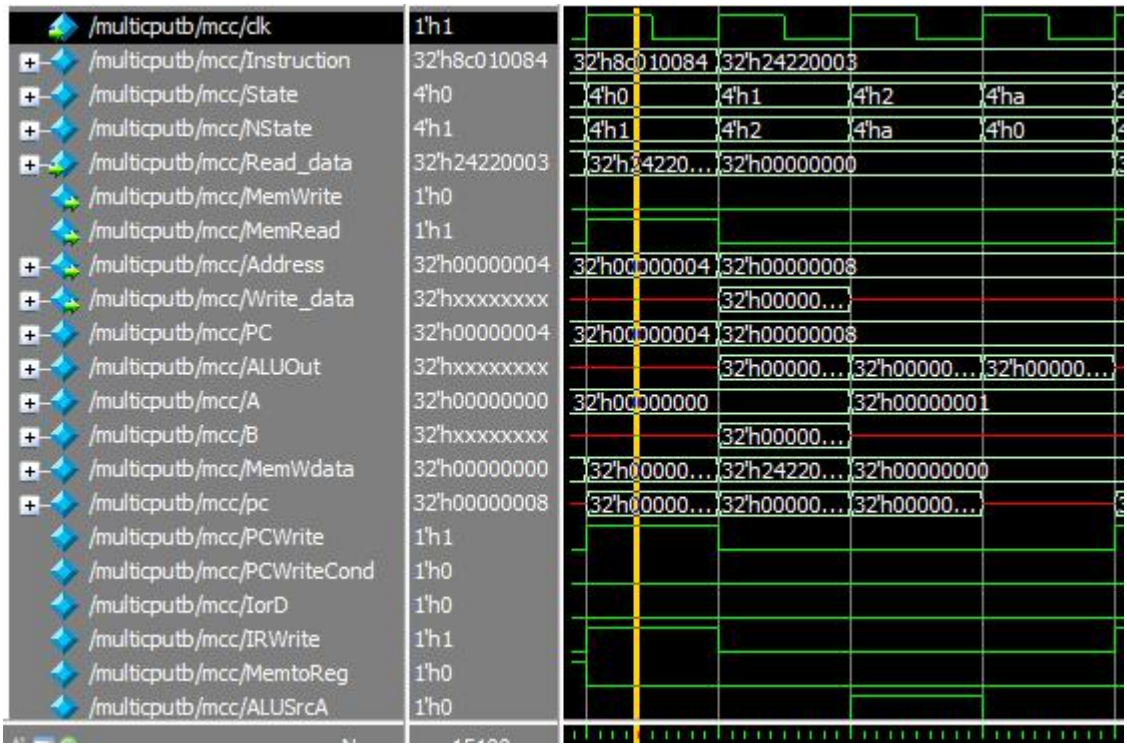
先上真值表部分。（其中 casefx 表示是对 instruction 末 6 位的 case，caseop 是对前 6 位的 case，S1/S2 是 instruction[15:0]的两种方式拓展，绿色的表示只有在那个指令下才是 1，蓝色均表示 1）

state	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
PCWrite	■									■				■		
PCWriteCond									■							
BNE									BNE							
MemRead	■			■												
MemWrite						■										
IRWrite	■															
RegWrite					■			■			■		■			
ALUOp[1]							■					■			■	■
ALUOp[0]									■							■
alu=	add	add	add	add	add	add	casefx	add	sub	add	add	caseop	add	add	casefx	caseop
PCSource[1]										■						
PCSource[0]									■							
JRPCSource														■		
pc=	Result	Result	Result	Result	Result	Result	Result	Result	ALUOut	PCI00	Result	Result	Result	rdata1	Result	Result
IorD				■		■										
address=	PC	PC	PC	ALUOut	PC	ALUOut	PC	PC	PC	PC	PC	PC	PC	PC	PC	PC
MemtoReg					■											
wdata=	ALUOut	ALUOut	ALUOut	ALUOut	MemW	ALUOut	ALUOut	ALUOut	ALUOut	ALUOut	ALUOut	ALUOut	ALUOut	ALUOut	ALUOut	ALUOut
RegDst								■								
JalRegDst													■			
waddr=	I20:16	I20:16	I20:16	I20:16	I20:16	I20:16	I20:16	I15:11	I20:16	I20:16	I20:16	I20:16	31	I20:16	I20:16	I20:16
ALUSrcA			■				■		■							■
SllALUSrcA															■	
C=	PC	PC	A	PC	PC	PC	A	PC	A	PC	PC	PC	PC	PC	I10:6	A
ALUSrcB[1]		■	■									■				■
ALUSrcB[0]	■	■														
JalALUSrcB		Jal														
D=	4	4/S2	S1	B	B	B	B	B	B	B	B	S1	B	B	B	S1

其次是状态机部分。（紫色表示有该状态，其上数字表示按顺序，例如 addiu 是 state0→state1→state2→state10。）

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
001001addiu	1	2	3								4						temp ← GPR[rs] + sign_extend(immediate) GPR[rt]← temp
000000addu100001	1	2					3	4									temp ← GPR[rs] + GPR[rt] GPR[rd] ← temp
000100beq	1	2							3								I: target_offset ← sign_extend(offset || 0 2) I+1: if GPR[rs] = GPR[rt] then PC ← PC + target_offset
000101bnez	1	2							3								I: target_offset ← sign_extend(offset || 0 2) I+1: if GPR[rs] ≠ GPR[rt] then PC ← PC + target_offset
000010j	1	2								3							I: I+1:PC ← PC GPRLEN..28 || instr_index || 0 2
000011jal	1	2								4			3				I: GPR[31]← PC + 8 I+1:PC ← PC GPRLEN..28 || instr_index || 0 2
000000jr001000	1	2					3							4			I: temp ← GPR[rs] I+1:PC ← temp
001111lui	1	2									4	3					GPR[rt] ← immediate || 0 16
000000sll000000	1	2					3	5							4		s ← sa temp ← GPR[rt] (31-s)..0 || 0 s GPR[rd]← temp
001010slti	1	2									4					3	if GPR[rs] < sign_extend(immediate) then GPR[rt] ← 0 GPRLEN-1 || 1 else GPR[rt] ← 0 GPRLEN
001011sltiu	1	2									4					3	if (0 || GPR[rs]) < (0 || sign_extend(immediate)) then GPR[rd] ← 0 GPRLEN-1 || 1 else GPR[rd] ← 0 GPRLEN
000000subu100011	1	2					3	4									temp ← GPR[rs] - GPR[rt] GPR[rd]← temp

图 1-addiu



state0→state1→state2→state10

相应代码只涉及了 state10，而 state10 并没有新增其他变量，略去相应代码。

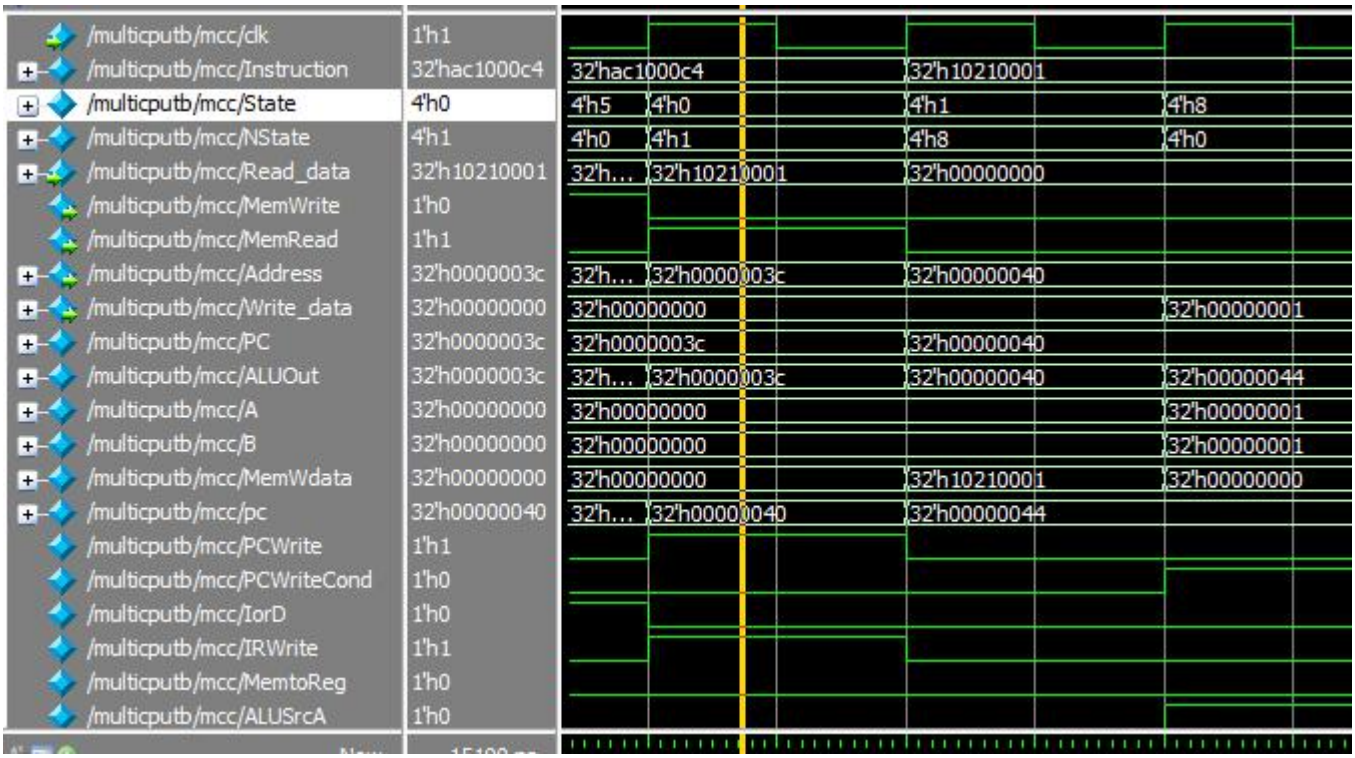
图 2-addu

state0→state1→state6→state7

addu 指令只涉及了实验 3 部分的内容，所以测试都没测试，无仿真波形，略去相应代码。

图 3-beq

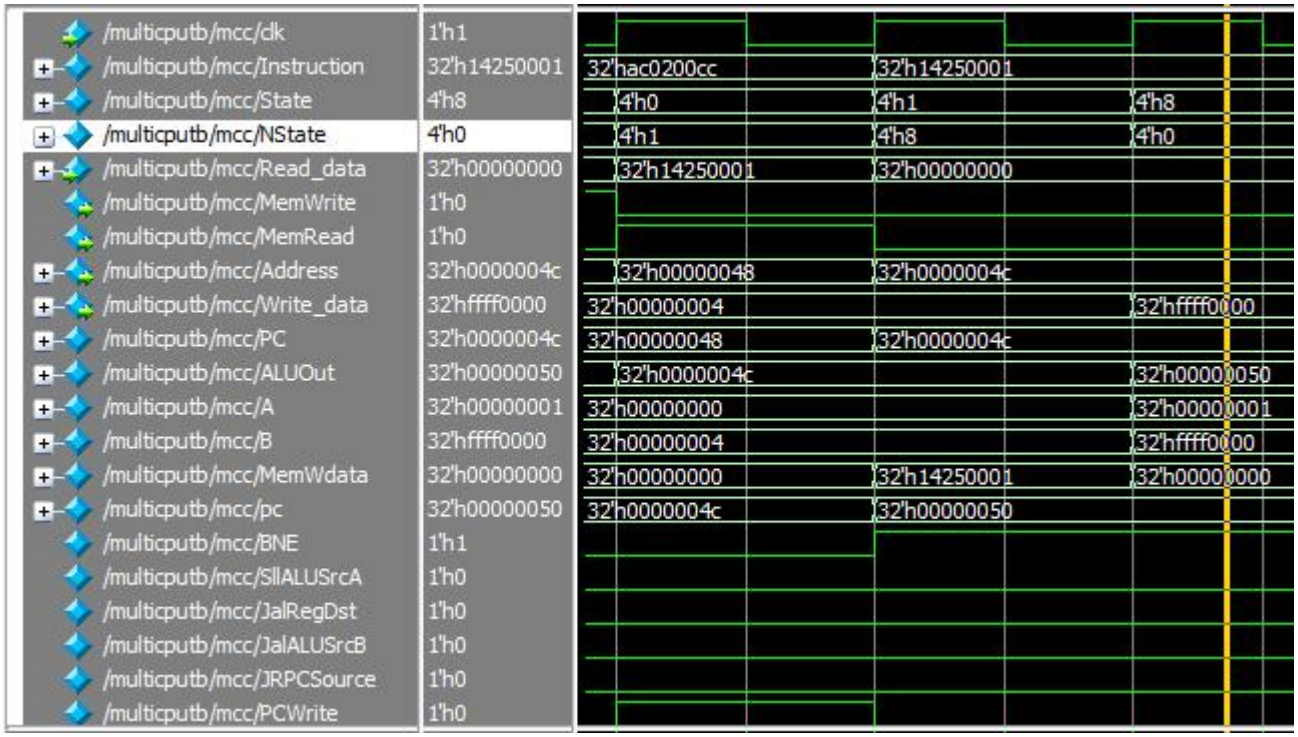
state0→state1→state8



代码部分写到 beq 的时候，仍然只涉及了实验 3 部分，虽然相应代码有所改动，但是因为 bne 的出现，故代码部分放置于 bne 部分中。

图 4-bne

state0→state1→state8



图中 BNE 在状态 8 的时候为 1，即我真值表状态 8 中的绿色部分。相应代码如下。

```
assign BNE=(Op==6'b000101)? 1:0;
```

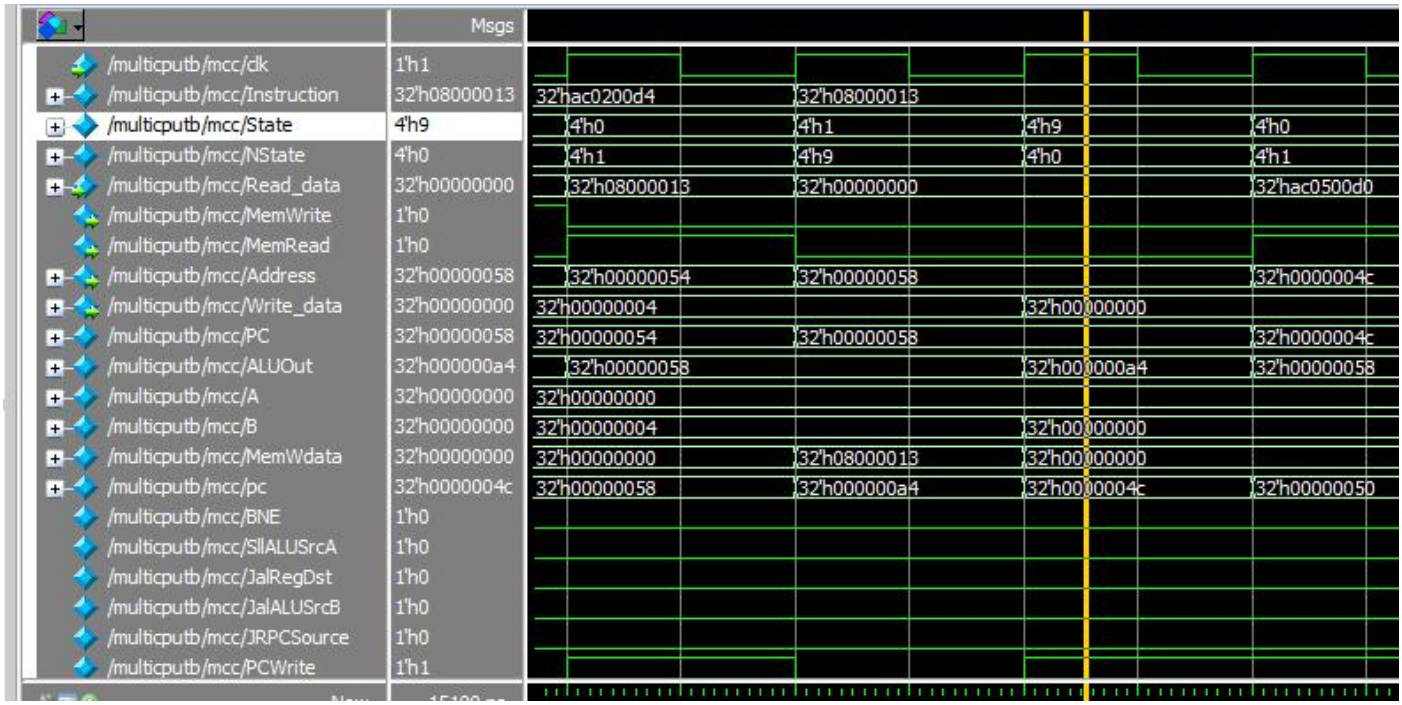
新增 BNE 变量，用以判断区分是 bne 还是 beq，因为只有这 2 个指令才涉及到 state8

```
if( PCWrite|(PCWriteCond&(Zero^BNE)) )//+ bne
  PC<=pc;
else PC<=PC;
```

在 bne 的情况下要 Zero=0 才对 PC 改写，beq 则相反，所以利用 BNE 变量与 Zero 进行异或。

图 5-j

state0→state1→state9

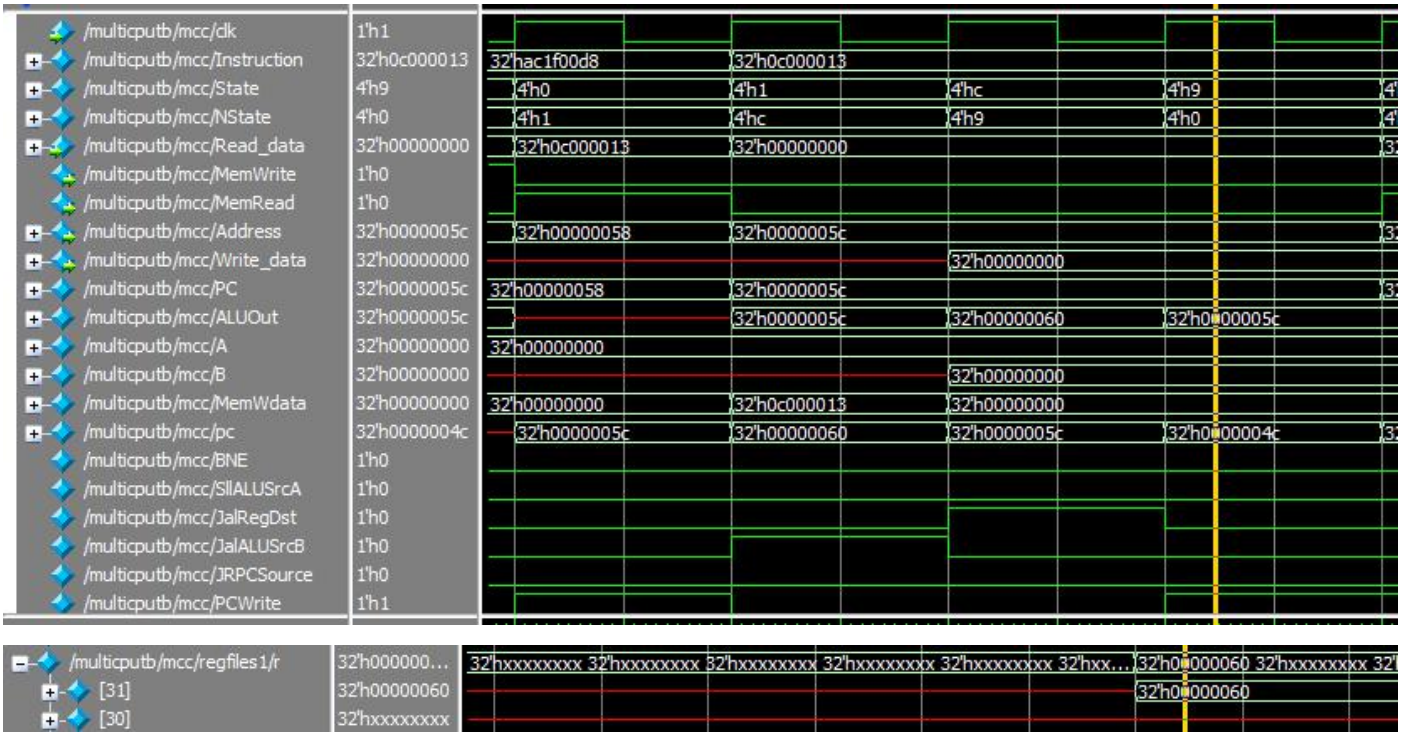


如图所示，在状态 9 的时候，pc（我代码中，若对 PC 进行改写，PC<=pc）已经变成了要跳转的地址 32'h0000004c，在下一阶段 PC 也变成了这个值。

代码部分，在实验 3 中按照几个 pdf 中的真值表状态转换表已经完成了，故略去代码。

图 6-jal

state0→state1→state12→state9



如图所示，在 state9 的时候，因为 state12 的操作，r[31]已经写入了 PC+4 的数据。
相应代码只涉及了 state12，新增代码如下。

```
assign JalRegDst=state[12];
assign waddr=(JalRegDst)? 5'd31:((RegDst)? Instruction[15:11]:Instruction[20:16]);
assign JalALUSrcB=state[1]&&(Op==6'b000011);
assign D= (JalALUSrcB)? 32'd4:
( (ALUSrcB[1])? ((ALUSrcB[0])? Signextend2:Signextend1):((ALUSrcB[0])? 32'd4:B) );
```

图 7-jr

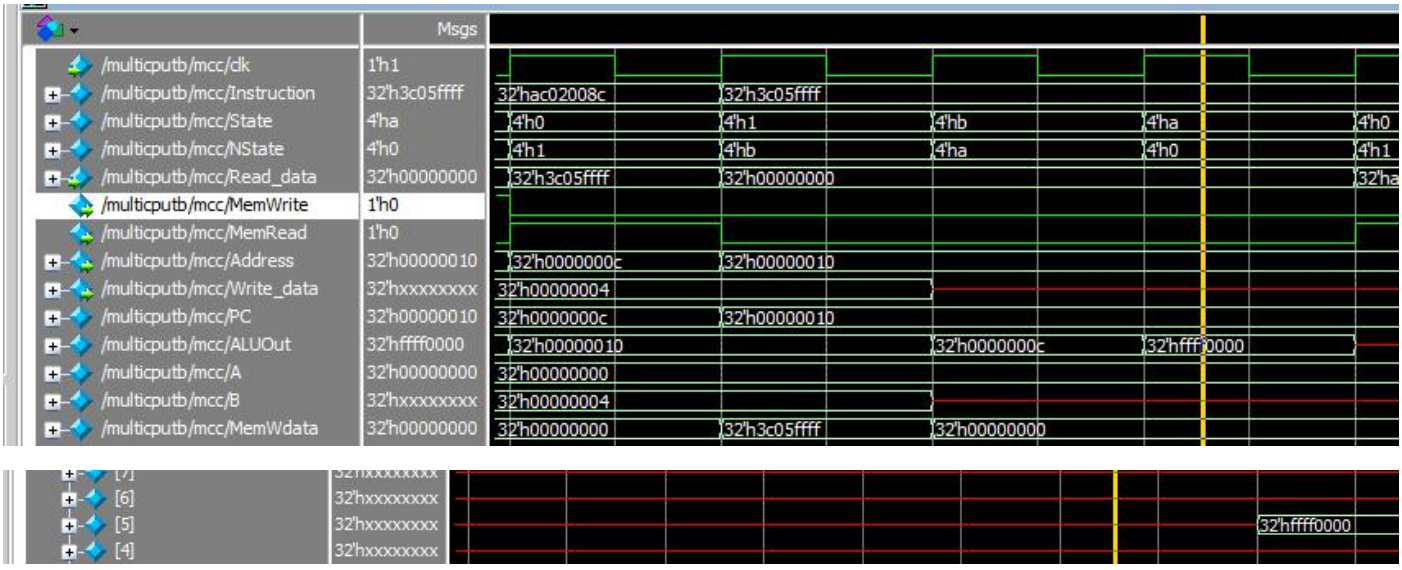
state0→state1→state6→state13

如图所示，在下一个 state0 的时候，因为 jr 指令已经跳到 r[31]中的数 32’h0000004c
相应代码只涉及了 state13，新增代码如下。

```
assign JRPCSource=state[13];
assign pc= (JRPCSource)? rdata1:
((PCSource[1])? {PC[31:28],Instruction[25:0],2'b00}:((PCSource[0])? ALUOut:Result));
```

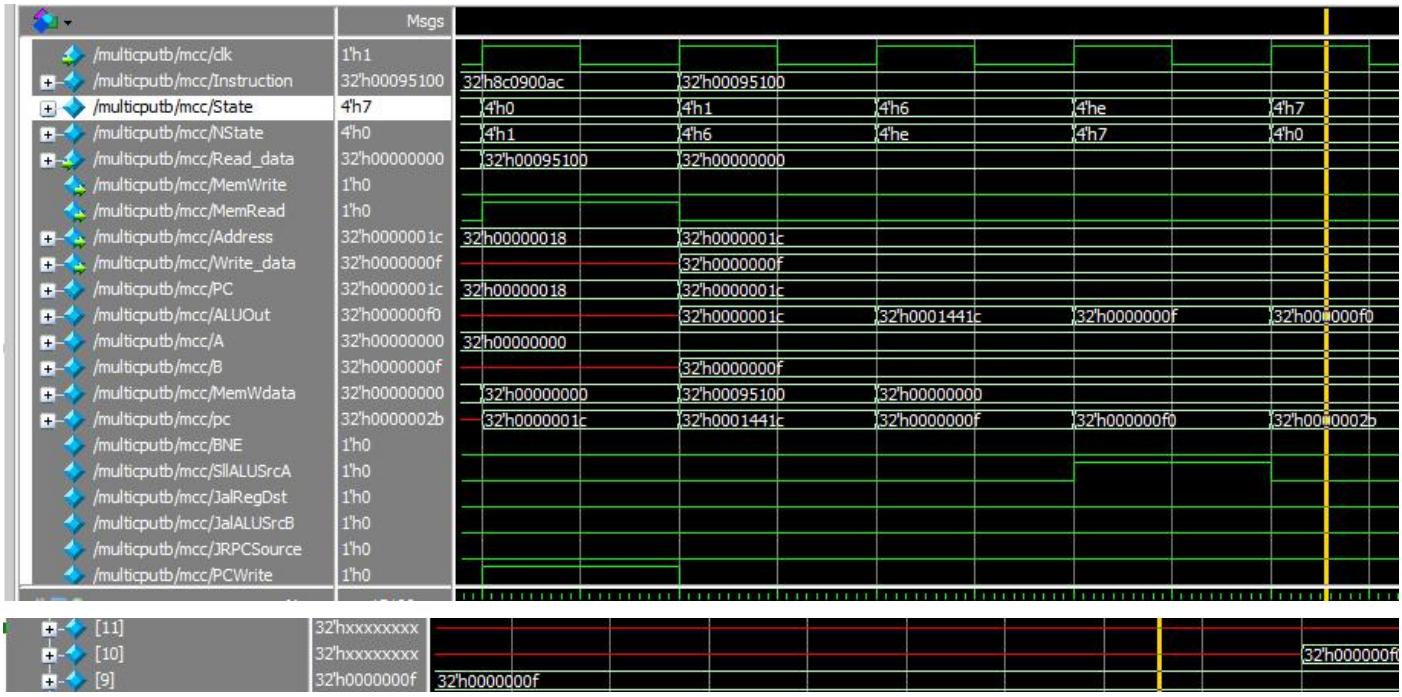
图 8-lui

state0→state1→state10→state11



如图所示，该指令向 r[5]中存了{Instruction[15:0],16'b0}
代码部分略去，涉及了 state10，和 state11。在 state11 中有关 alu，alu 部分代码放置于最后。

图 9-sll
state0→state1→state6→state14→state7

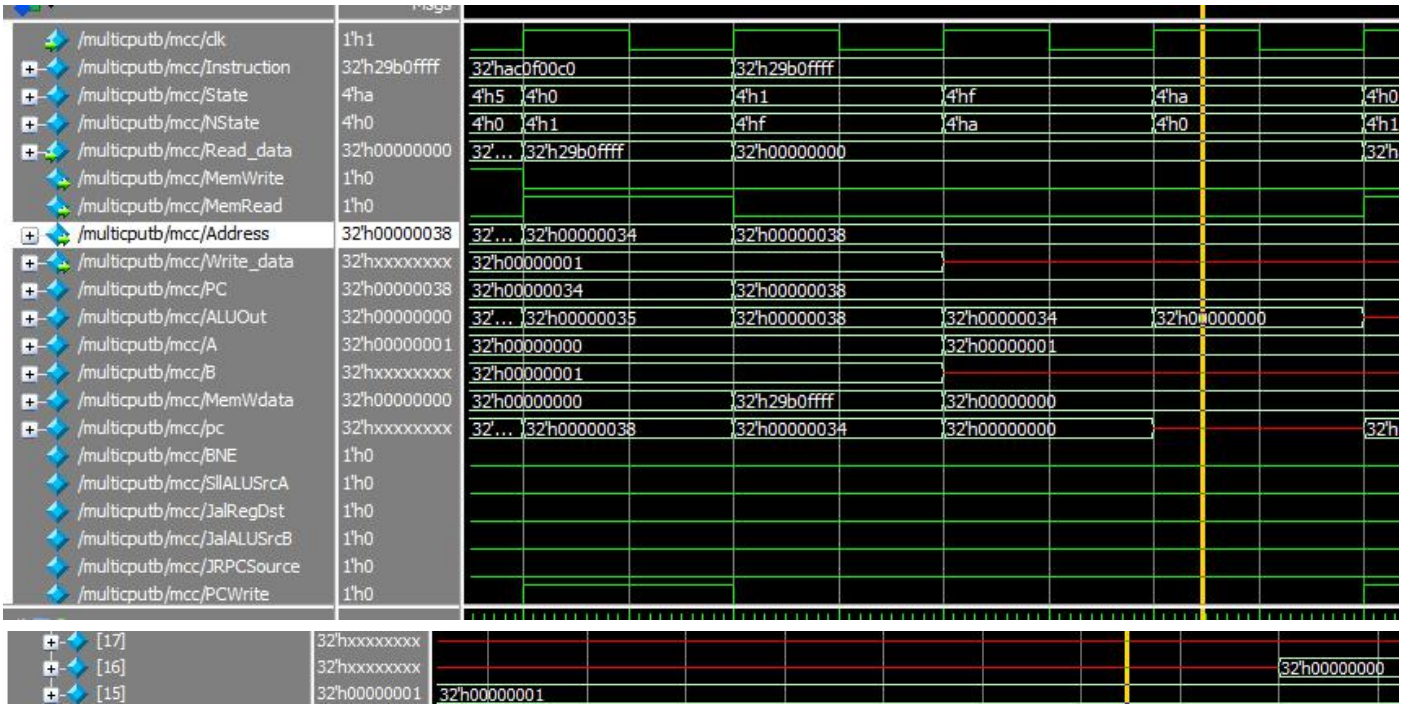


如图所示，该指令向 r[10]中存了 r[9]左移 Instruction[10:6]位
代码部分，涉及了 state14。在 state14 中有关 alu，alu 部分代码放置于最后。

```
assign SllALUSrcA=state[14];

assign C= (SllALUSrcA)? {27'b0,Instruction[10:6]}:((ALUSrcA)? A:PC);
```

图 10-slti
state0→state1→state15→state10



如图所示，该指令向 r[16]中存了判断 r[13]与-1 的比较结果。
代码部分略去，涉及了 state15，而 state15 涉及了 alu，该指令中用的是以前写过的 alu 功能 slt。

图 11-sltiu

state0→state1→state15→state10

如图所示，该指令向 r[15]中存了判断 r[13]与 ffff 的无符号拓展比较结果。

代码部分略去，涉及了 state15，而 state15 涉及了 alu，新增了 alu 功能，alu 代码放在最后。

图 12-subu

state0→state1→state6→state7

该代码与实验三的 sub 指令基本一致，且我们忽略了 overflow 的限制，所以无仿真波形。

Alu

```
assign C=(ALUop[2])? ~B:B;
assign cin=ALUop[2];

add32 part1(cout,result1,A,C,cin);

   //0-and,1-or,2-add,3-sll,4-sltiu,5-lui,6-sub,7-slt
   always @(A or B or result1 or cout or ALUop)
   begin
     case(ALUop)
       3'b000:result=A&B;
       3'b001:result=A|B;
       3'b010:result=result1;
       3'b011:result=B<<A[4:0];
       3'b100:result=(cout[6])? 32'b0:32'b1;
       3'b101:result=B<<16;
       3'b110:result=result1;
       3'b111:
         if( (~(cout[6]^cout[5])&&result1[31]) || ((cout[6]^cout[5])&&cout[6]) )
           result=32'b1;
         else
           result=32'b0;
       default:result=32'b0;
     endcase
   end
```

如图所示，在 3’b011 地方加入了 sll，直接对 B 进行左移 A[4:0]；3’b100 加入了 sltiu，保持了 aluop[2]仍为 1，使内部加法器做减法，并且由于是不会溢出的减法，所以直接判断符号位进位即可；3’b101 加入了 lui，直接对 B 进行

左移 16 位。

二、 问题合集

这次新增了十多条指令，增加了 6 个状态，3 个 alu 功能，略去各种粗心造成的写错问题，还有一些思考上的遗漏错误。

Problem1

```
//if(PCWrite|(PCWriteCond&Zero))
//if( PCWrite|(PCWriteCond&Zero)|(PCWriteCond&~Zero&(Op==6'b000101)) )
if( PCWrite|(PCWriteCond&(Zero^BNE)) )//+ bne
  PC<=pc;
else PC<=PC;
```

问题：写 bne 的时候，忽略了一种情况，在 bne 条件下 Zero=1 时，也会对 PC 进行改写。

解决：将 BNE 与 Zero 进行异或后再并上 PCWriteCond，就没有上述错误出现。

Problem2

```
assign C=(ALUop[2])? ~B:B;
assign cin=ALUop[2];

add32 part1(cout,result1,A,C,cin);

//0-and,1-or,2-add,3-sll,4-sltiu,5-lui,6-sub,7-slt
```

问题：之前在写 sltiu 的的时候，给他对应的 aluop 是 011，然后就不知道怎么处理减法了，写的比较繁复

解决：将指令位置互换即可，把该指令换到 100 或 101 的位置上。

Problem3

无图

问题：PC+8 的指令没有放入 alu，直接在代码部分写了 PC+8

解决：改写成进入 alu 做加法的 PC+8。（后来改成了+4，因为借用同学的 testbench 要求的是+4）

三、 对于此次实验的心得、感受和建议

增加了几条指令，不能像以前那样无脑按照真值表和状态表写了，每条指令都要自己耐心的去写应该需要哪些信号置 1，问题产生也不像之前，耐心看下代码逻辑就可以了，有时候出错了，是自己遗漏的方面，只能对着仿真代码每个信号去看，他理论上应该是什么，但实际上是什么，发现错误，然后去看相应代码。因此此次作业做的比之前的也费时许多，也认真许多。

另外感谢各路大腿借我用的 testbench，使我省了不少写 testbench 的功夫。就这样吧=。=希望下次作业不要死的太惨。