同济大学计算机系

计算机组成原理实验报告

多周期 54 条指令 CPU

1. 实验介绍

在本次实验中，我们将使用 Verilog HDL 语言实现 54 条 MIPS 指令的 CPU 的设计和仿真，设计的 CPU 可以是单周期的，也可以是多周期。

2. 实验目标

- 深入了解 CPU 的原理。
- 画出实现 54 条指令的 CPU 的通路图。
- 学习使用 Verilog HDL 语言设计实现 54 条指令的 CPU。

3. 实验原理

1）54 指令 CPU 需要实现的 54 条 MIPS 指令，见下表

指令	指令说明	指令格式	OP 31-26	FUNCT 5-0	指令码 16 进制
addi	加立即数	addi rt, rs, immediate	001000		20000000
addiu	加立即数（无符号）	addiu rd, rs, immediate	001001		24000000
andi	立即数与	andi rt, rs, immediate	001100		30000000
ori	或立即数	ori rt, rs, immediate	001101		34000000
sltiu	小于立即数置 1（无符号）	sltiu rt, rs, immediate	001011		2C000000
lui	立即数加载高位	lui rt, immediate	001111		3C000000
xori	异或（立即数）	xori rt, rs, immediate	001110		38000000
slti	小于置 1（立即数）	slti rt, rs, immediate	001010		28000000
addu	加（无符号）	addu rd, rs, rt	000000	100001	00000021
and	与	and rd, rs, rt	000000	100100	00000024
beq	相等时分支	beq rs, rt, offset	000100		10000000
bne	不等时分支	bne rs, rt, offset	000101		14000000
j	跳转	j target	000010		08000000
jal	跳转并链接	jal target	000011		0C000000
jr	跳转至寄存器所指地址	jr rs	000000	001000	00000009
lw	取字	lw rt, offset(base)	100011		8C000000
xor	异或	xor rd, rs, rt	000000	100110	00000026
nor	或非	nor rd, rs, rt	000000	100111	00000027
or	或	or rd, rs, rt	000000	100101	00000025

sll	逻辑左移	sll rd, rt, sa	000000	000000	00000000
sllv	逻辑左移（位数可变）	sllv rd, rt, rs	000000	000100	00000004
sltu	小于置 1（无符号）	sltu rd, rs, rt	000000	101011	0000002B
sra	算数右移	sra rd, rt, sa	000000	000011	00000003
srl	逻辑右移	srl rd, rt, sa	000000	000010	00000002
subu	减（无符号）	sub rd, rs, rt	000000	100010	00000022
sw	存字	sw rt, offset(base)	101011		AC000000
add	加	add rd, rs, rt	000000	100000	00000020
sub	减	sub rd, rs, rt	000000	100010	00000022
slt	小于置 1	slt rd, rs, rt	000000	101010	0000002A
srlv	逻辑右移（位数可变）	srlv rd, rt, rs	000000	000110	00000006
srav	算数右移（位数可变）	srav rd, rt, rs	000000	000111	00000007
clz	前导零计数	clz rd, rs	011100	100000	70000020
divu	除（无符号）	divu rs, rt	000000	011011	0000001B
eret	异常返回	eret	010000	011000	42000018
jalr	跳转至寄存器所指地址，返回地址保存在	jalr rs	000000	001001	00000008
lb	取字节	lb rt, offset(base)	100000		80000000
lbu	取字节（无符号）	lbu rt, offset(base)	100100		90000000
lhu	取半字（无符号）	lhu rt, offset(base)	100101		94000000
sb	存字节	sb rt, offset(base)	101000		A0000000
sh	存半字	sh rt, offset(base)	101001		A4000000
lh	取半字	lh rt, offset(base)	100001		84000000
mfc0	读 CP0 寄存器	mfc0 rt, rd	010000	000000	40000000
mfhi	读 Hi 寄存器	mfhi rd	000000	010000	00000010
mflo	读 Lo 寄存器	mflo rd	000000	010010	00000012
mtc0	写 CP0 寄存器	mtc0 rt, rd	010000	000000	40800000
mthi	写 Hi 寄存器	mthi rd	000000	010001	00000011
mtlo	写 Lo 寄存器	mtlo rd	000000	010011	00000013
mul	乘	mul rd, rs, rt	011100	000010	70000002
multu	乘（无符号）	multu rs, rt	000000	011001	00000019
syscall	系统调用	syscall	000000	001100	0000000C
teq	相等异常	teq rs, rt	000000	110100	00000034
bgez	大于等于 0 时分支	bgez rs, offset	000001		04010000
break	断点	break	000000	001101	0000000D
div	除	div rs, rt	000000	011010	0000001A

- 阅读每条指令，对每条指令所需执行的功能与过程都有充分的了解
- 确定每条指令在执行过程中所用到的部件
- 使用表格列出指令所用部件，并在表格中填入每个部件的数据输入来源
- 根据表格所涉及部件和部件的数据输入来源，画出整个数据通路

2）多周期

多周期 CPU 的中心思想是把一条指令的执行分成若干个小周期，根据每条指令的复杂程度，使用不同数量的小周期去执行，许多个小周期加在一起相当于单周期 CPU 中的一个周期。

CPU 在处理指令时，一般需要经过以下几个阶段：

取指令(IF)：根据程序计数器 pc 中的指令地址，从存储器中取出一条指令，同时，pc 根据指令字长度自动递增产生下一条指令所需要的指令地址，但遇到“地址转移”指令时，则控制器把“转移地址”送入 pc，当然得到的“地址”需要做些变换才送入 pc。

指令译码(ID)：对取指令操作中得到的指令进行分析并译码，确定这条指令需要完成的操作，从而产生相应的操作控制信号，用于驱动执行状态中的各种操作。

指令执行(EXE)：根据指令译码得到的操作控制信号，具体地执行指令动作，然后转移到结果写回状态。

存储器访问(MEM)：所有需要访问存储器的操作都将在这个步骤中执行，该步骤给出存储器的数据地址，把数据写入到存储器中数据地址所指定的存储单元或者从存储器中得到数据地址单元中的数据。

结果写回(WB)：指令执行的结果或者访问存储器中得到的数据写回相应的目的寄存器中。

实验中就按照这五个阶段进行设计。

为设计方便，以及保证 CPU 的扩展性，以及方便设计流水线，每条指令都需要五个小周期。

4.数据通路

1）单独数据通路：

1 ADD

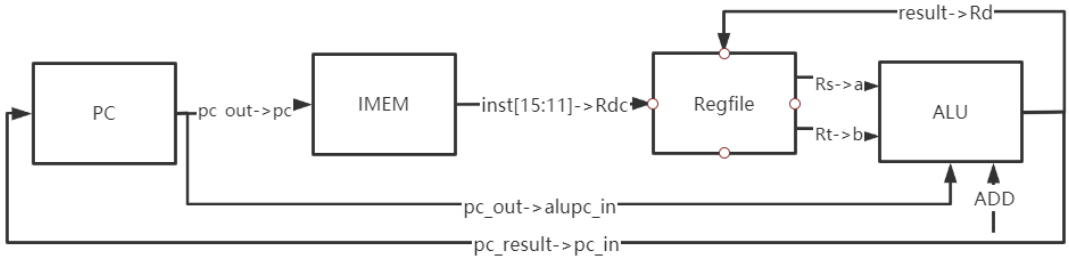
格式：ADD rd, rs, rt

操作：取指令，rd←rs+rt, PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
ADD	ALU	PC	ALU	15-11	Rs	Rt	PC



2 ADDU

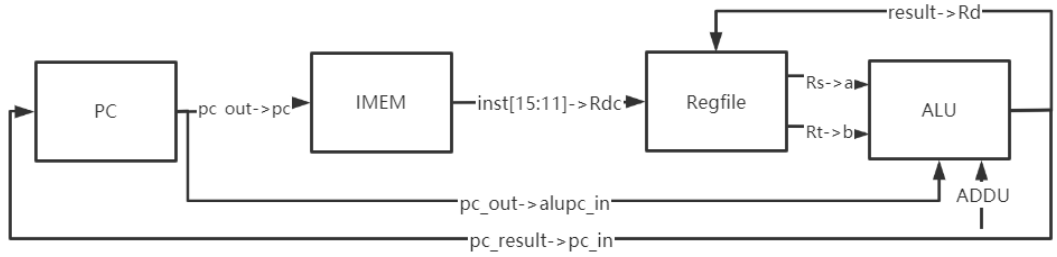
格式：ADDU rd, rs, rt

操作：取指令，rd←rs+rt, PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
ADDU	ALU	PC	ALU	15-11	Rs	Rt	PC



3 SUB

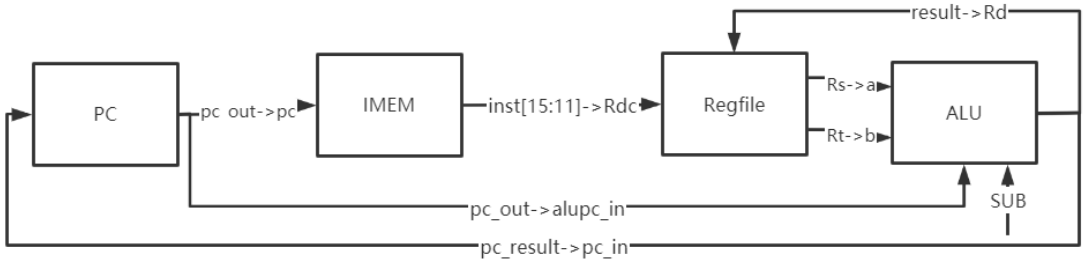
格式：SUB rd, rs, rt

操作：取指令，rd←rs-rt, PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SUB	ALU	PC	ALU	15-11	Rs	Rt	PC



4 SUBU

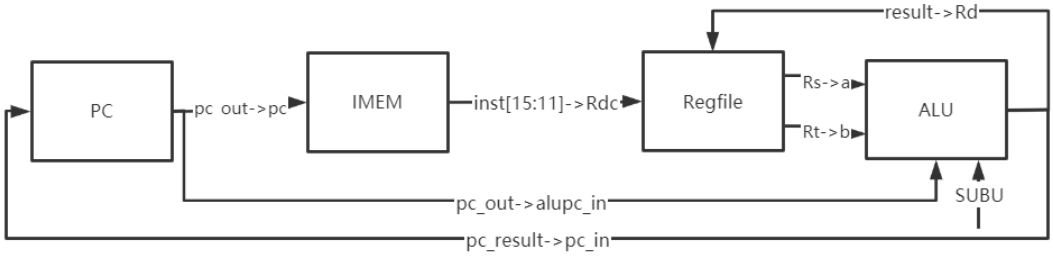
格式：SUBU rd, rs, rt

操作：取指令，rd←rs-rt, PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SUBU	ALU	PC	ALU	15-11	Rs	Rt	PC



5 AND

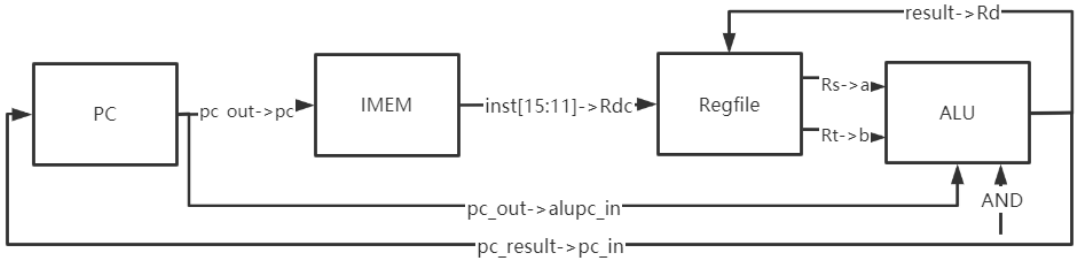
格式：AND rd, rs, rt

操作：取指令，rd←rs & rt, PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
AND	ALU	PC	ALU	15-11	Rs	Rt	PC



6 OR

格式：OR rd, rs, rt

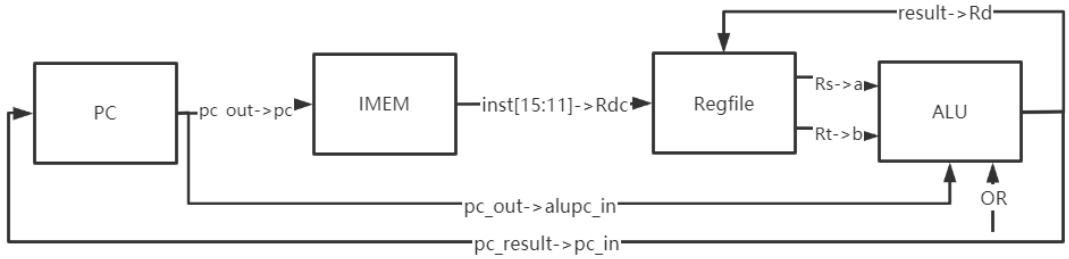
操作：取指令，rd←rs | rt, PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile	ALU

			Rd	Rdc	A	B	pc_in
OR	ALU	PC	ALU	15-11	Rs	Rt	PC



7 XOR

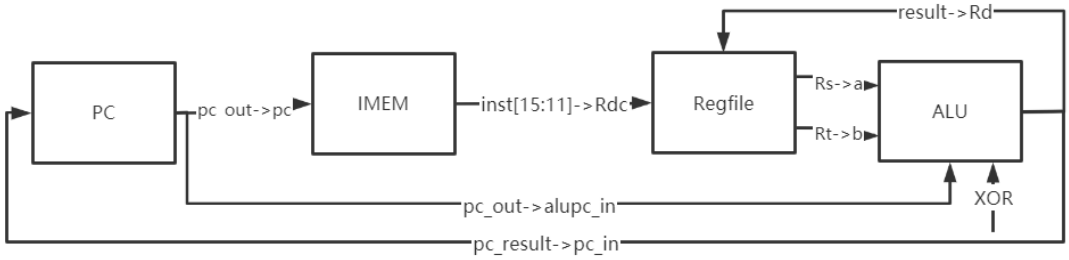
格式：XOR rd, rs, rt

操作：取指令，rd←rs ^ rt, PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
XOR	ALU	PC	ALU	15-11	Rs	Rt	PC



8 NOR

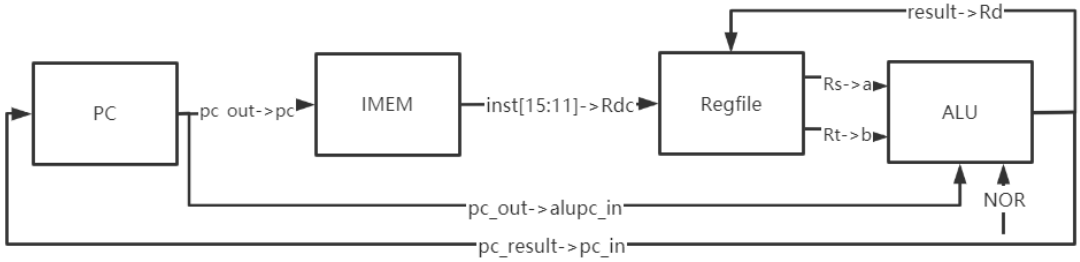
格式：NOR rd, rs, rt

操作：取指令，rd←~(rs | rt), PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Rgefile		ALU		
			Rd	Rdc	A	B	pc_in
NOR	ALU	PC	ALU	15-11	Rs	Rt	PC



9 SLT

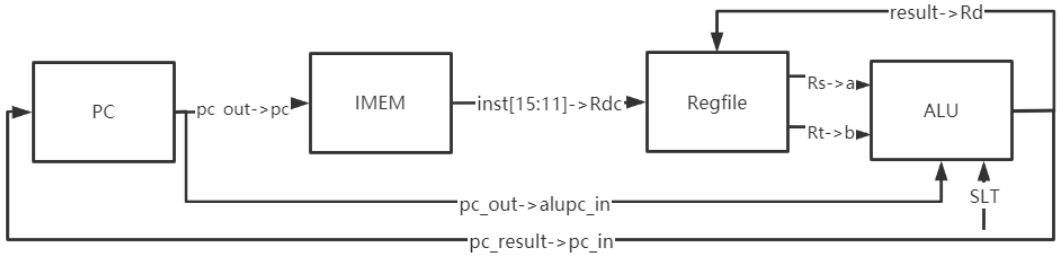
格式：SLT rd, rs, rt

操作：取指令，rd← rs < rt , PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SLT	ALU	PC	ALU	15-11	Rs	Rt	PC



10 SLTU

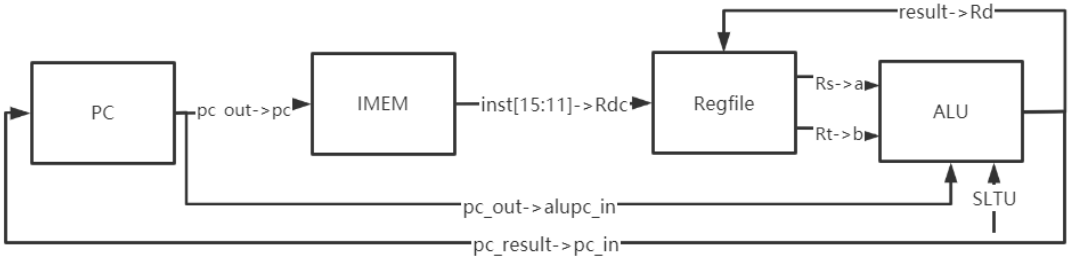
格式：SLTU rd, rs, rt

操作：取指令，rd← rs < rt , PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SLTU	ALU	PC	ALU	15-11	Rs	Rt	PC



11 SLL

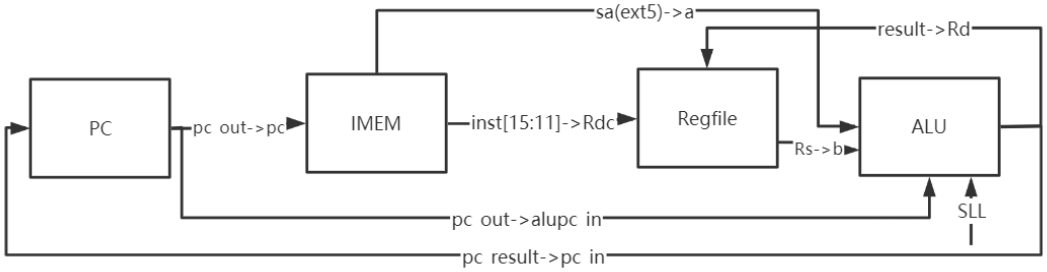
格式：SLL rd, rt, sa

操作：取指令，rd←rt << sa , PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SLL	ALU	PC	ALU	15-11	Sa(ext5)	Rt	PC



12 SRL

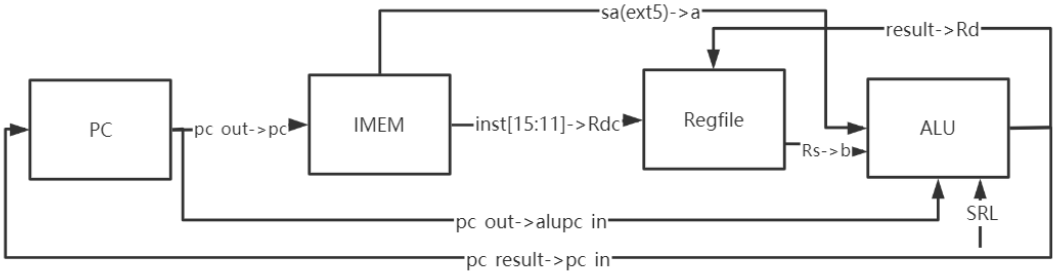
格式：SRL rd, rt, sa

操作：取指令，rd←rt >> sa (logical) , PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SRL	ALU	PC	ALU	15-11	Sa(ext5)	Rt	PC



13 SRA

格式：SRA rd, rt, sa

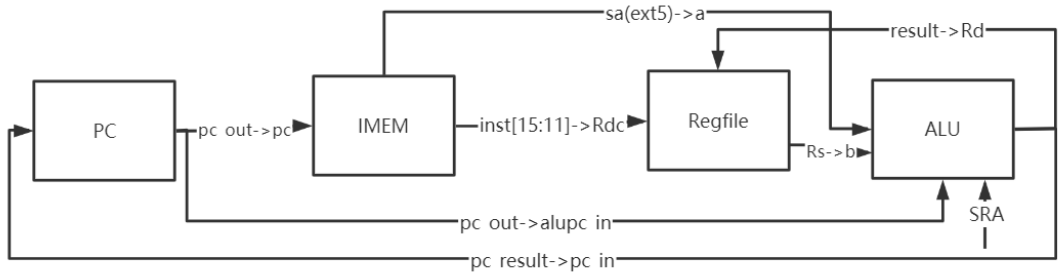
操作：取指令，rd←rt >> sa (arithmetic) , PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile	ALU

			Rd	Rdc	A	B	pc_in
SRA	ALU	PC	ALU	15-11	Sa(ext5)	Rt	PC



14 SLLV

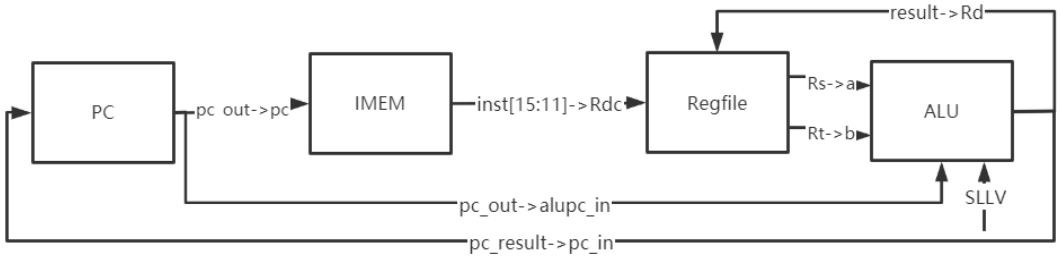
格式：SLLV rd, rt, rs,

操作：取指令，rd←rt << rs ，PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SLLV	ALU	PC	ALU	15-11	Rs	Rt	PC



15 SRLV

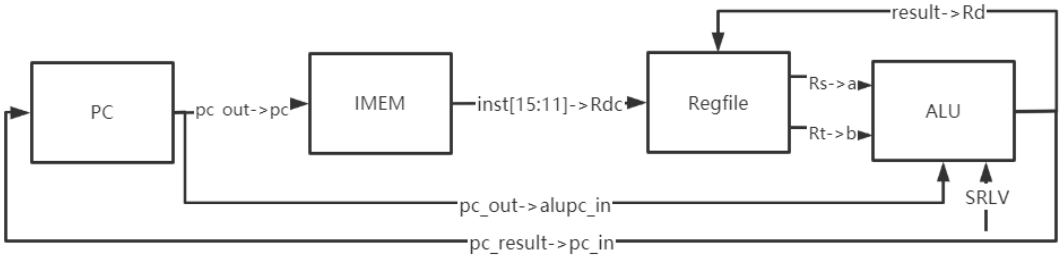
格式：SRLV rd, rt,rs

操作：取指令，rd←rt >> rs (logical) ，PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SRLV	ALU	PC	ALU	15-11	Rs	Rt	PC



16 SRAV

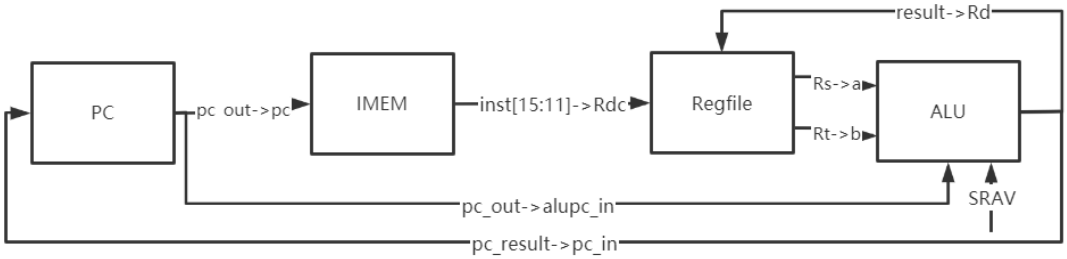
格式：SRAV rd, rt,rs

操作：取指令，rd←rt >> rs (arithmetic) , PC←PC+4

所需部件：PC, IMEM, Regfile, ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SRAV	ALU	PC	ALU	15-11	Rs	Rt	PC



17 JR

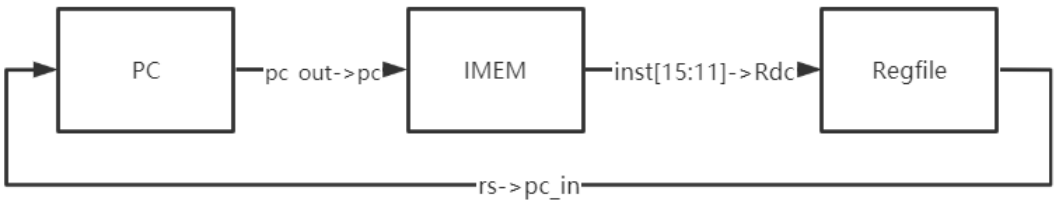
格式：JR rs

操作：取指令，PC←rs， PC←PC+4

所需部件：PC, IMEM, Regfile

输入输出关系：

	PC	IMEM	Regfile(Rdc)
JR	Rs	PC	15-11



18 ADDI

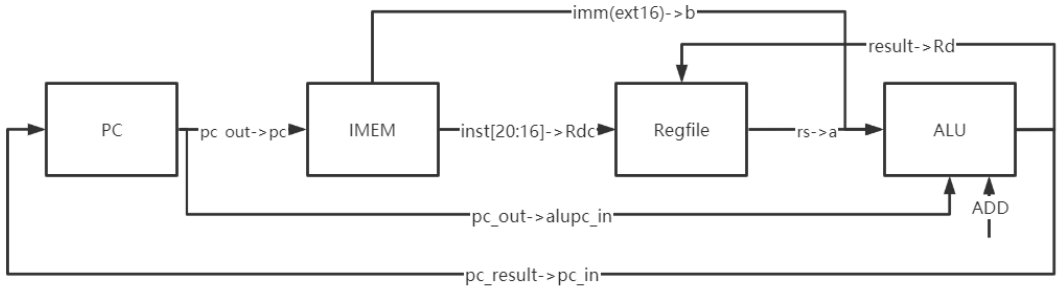
格式：ADDI rt, rs, imm16

操作：取指令、rt←rs+imm16(sign_extend) 、PC←PC+4

所需部件:: PC、IMEM、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
ADDI	ALU	PC	ALU	20-16	Rs	Imm(Ext16)	PC



19 ADDIU

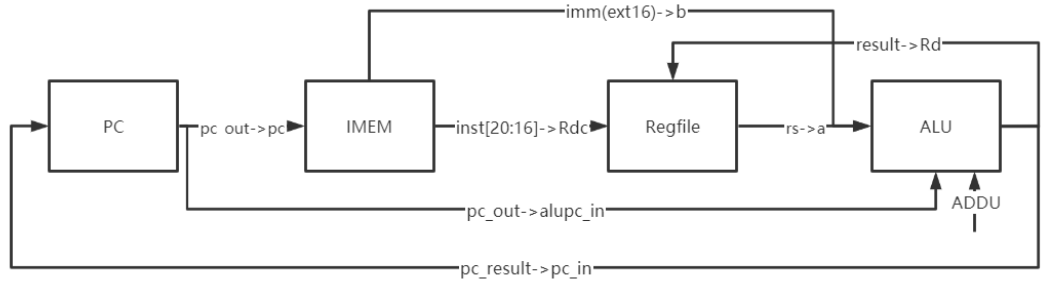
格式：ADDIU rt, rs, imm16

操作：取指令、rt←rs+imm16(sign_extend) 、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
ADDIU	ALU	PC	ALU	20-16	Rs	Imm(Ext16)	PC



20 ANDI

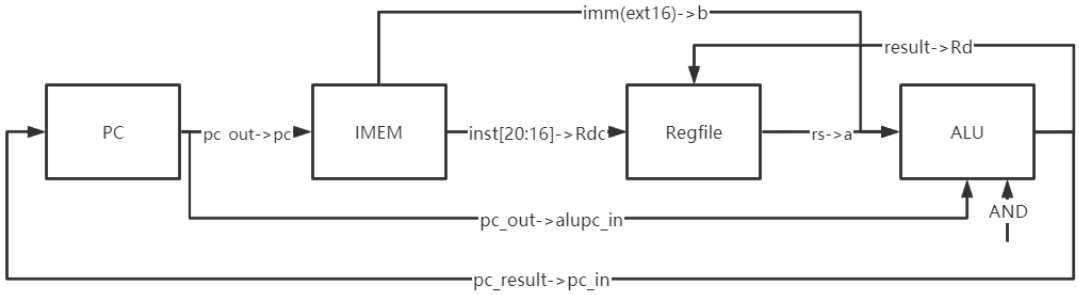
格式：ANDI rt, rs, imm16

操作：取指令、rt←rs&imm16(zero_extend) 、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
ANDI	ALU	PC	ALU	20-16	Rs	Imm(Ext16)	PC



21 ORI

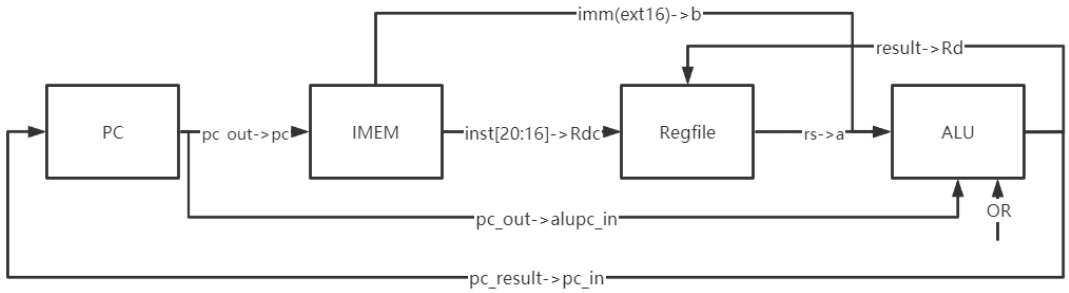
格式：ORI rt, rs, imm16

操作：取指令、rt←rs|imm16(zero_extend) 、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
ORI	ALU	PC	ALU	20-16	Rs	Imm(Ext16)	PC



22 XORI

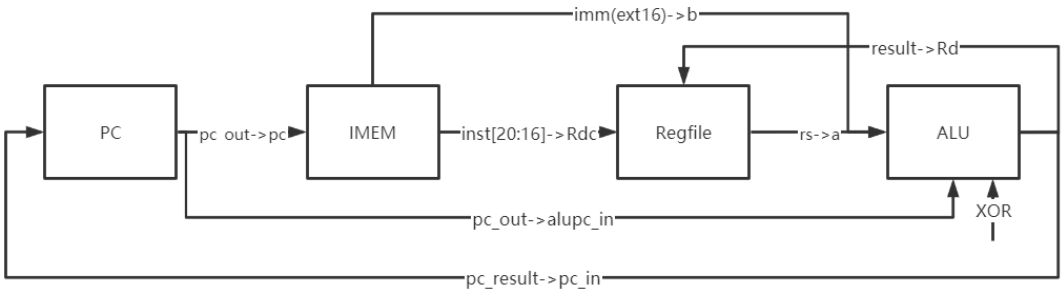
格式：XORI rt, rs, imm16

操作：取指令、rt←rs^imm16(zero_extend) 、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
XORI	ALU	PC	ALU	20-16	Rs	Imm(Ext16)	PC



23 LW

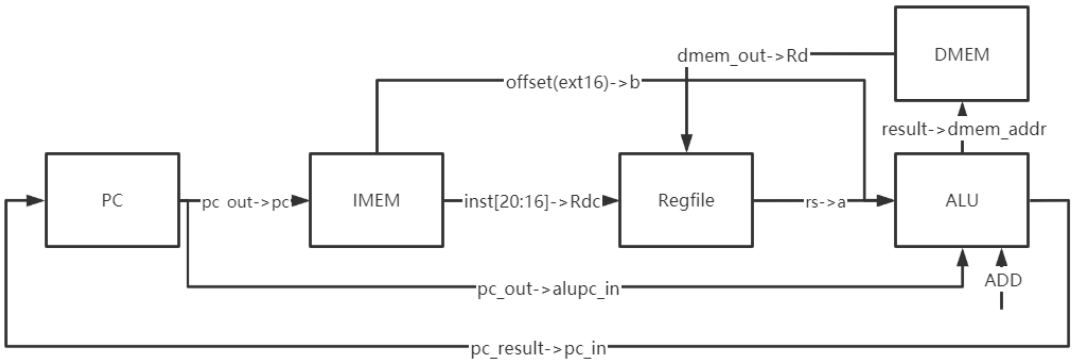
格式：LW rt, offset(base)

操作：取指令、rt←memory[rs + offset]、PC←NPC(PC+4)

所需部件：PC、IMEM、Regfile、ALU、DMEM

输入输出关系：

	PC	IMEM	Regfile		ALU			DMEM	
			Rd	Rdc	A	B	pc_in	Addr	Data
LW	ALU	PC	DMEM	20-16	Rs(base)	Offset(Ext16	PC	ALU	



24 SW

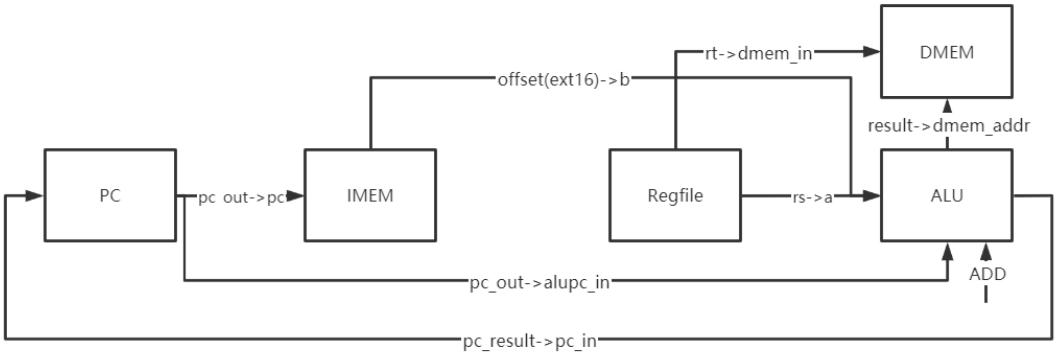
格式：SW rt, offset(base)

操作：取指令、memory[base + offset]←rt、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU、DMEM

输入输出关系：

	PC	IMEM	Regfile		ALU			DMEM	
			Rd	Rdc	A	B	pc_in	Addr	Data
SW	ALU	PC			Rs(base)	Offset(Ext16)	PC	ALU	Rt



25 BEQ

格式：BEQ rs, rt, offset

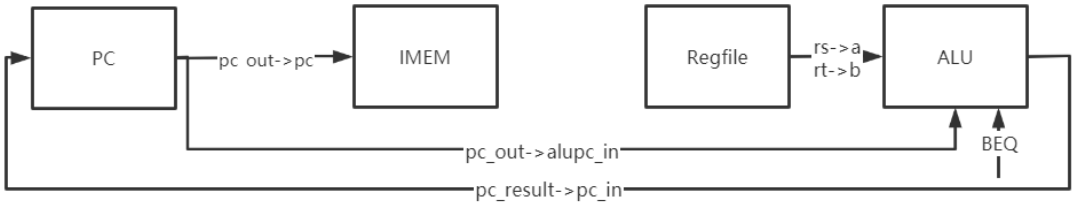
操作：if (rs=rt)PC←NPC + Sign_ext(offset||0^2)

else PC←PC+4

所需部件：PC、IMEM、Regfile、ALU

输入输出关系：

	PC	IMEM	Rgefile		ALU		
			Rd	Rdc	A	B	pc_in
BEQ	ALU	PC			Rs	Rt	PC



26 BNE

格式：BNE rs, rt, offset

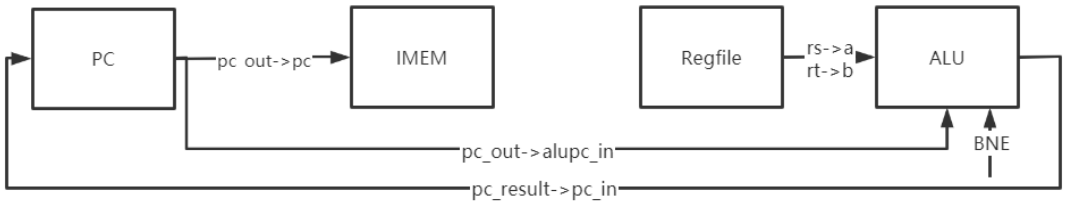
操作：if (rs≠rt)PC←NPC + Sign_ext(offset||02)

else PC←PC+4

所需部件：PC、IMEM、Regfile、ALU、ADD

输入输出关系：

	PC	IMEM	Rgefile		ALU		
			Rd	Rdc	A	B	pc_in
BNE	ALU	PC			Rs	Rt	PC



27 SLTI

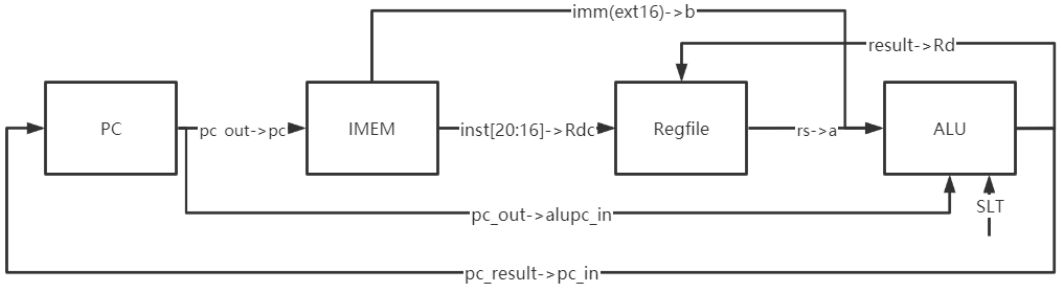
格式：SLTI rt, rs, imm16

操作：取指令、rt←rs << imm16(sign_extend)、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SLTI	ALU	PC	ALU	20-16	Rs	Imm(Ext16)	PC



28 SLTIU

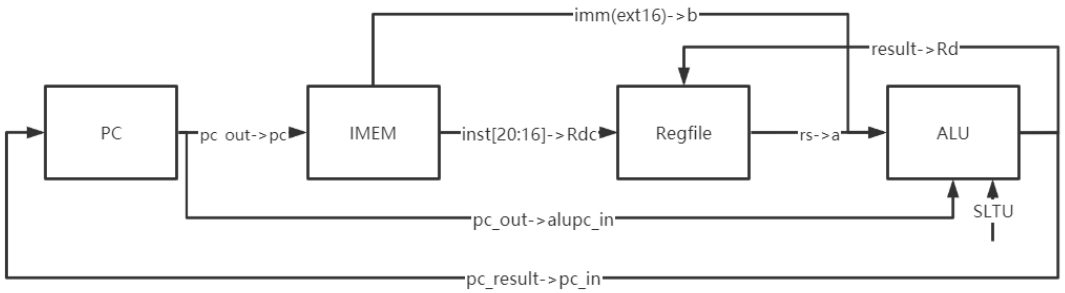
格式：SLTIU rt, rs, imm16

操作：取指令、rt←rs << imm16(sign_extend)、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
SLTIU	ALU	PC	ALU	20-16	Rs	Imm(Ext16)	PC



29 LUI

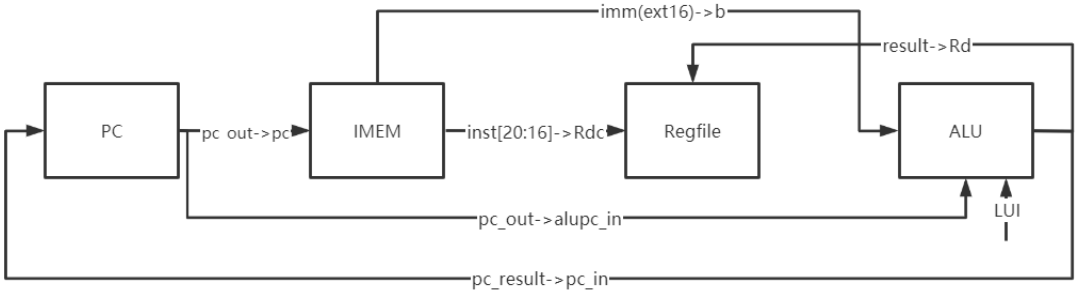
格式：LUI rt, imm16

操作：取指令、rt←imm16||0^{16} 、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU	
			Rd	Rdc	B	pc_in
LUI	ALU	PC	ALU	20-16	Imm(Ext16)	PC



30 J

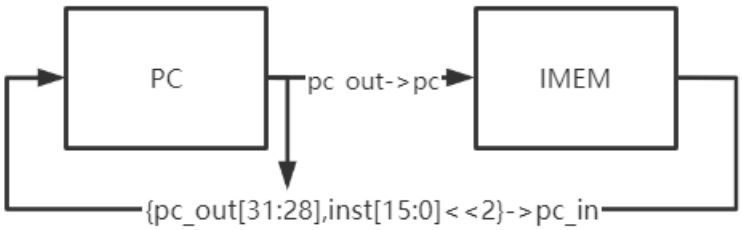
格式：J target

操作：取指令、PC ← PC_{31-28}||instr_index||0^2 , PC←PC+4

所需部件：PC、IMEM

输入输出关系：

	PC	IMEM
J	PC_{31-28}||instr_index||0^2	PC



31 JAL

格式：JAL target

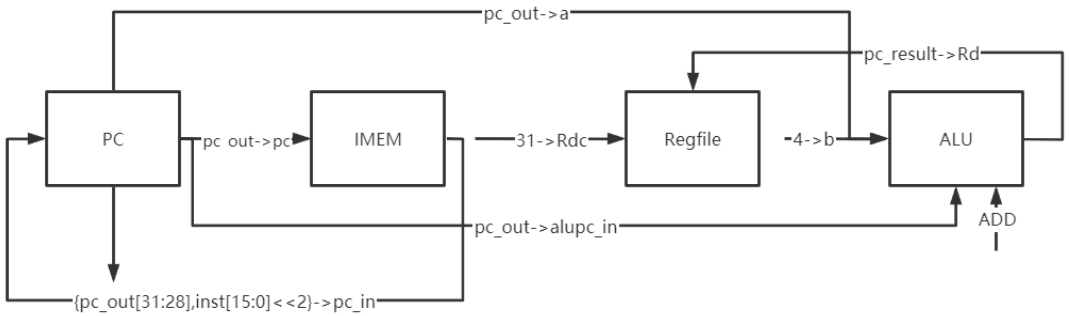
操作：取指令、 R[31] ← PC + 4,PC ← PC_{31-28}||instr_index||0^2 , PC←PC+4

所需部件:: PC、IMEM、Regfile

输入输出关系：

	PC	IMEM	Rgefile	ALU

			Rd	Rdc	A	B	pc_in
JAL	PC_{31-28}||instr_index||0^2	PC	ALU	31	PC	4	PC



32 BREAK

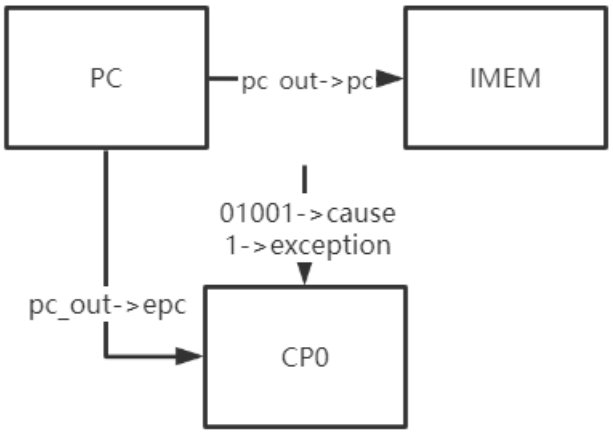
格式：BREAK MIPS32

操作：取指令、EPC←PC, cause←01001, status<<5

所需部件：PC、IMEM、CP0

输入输出关系：

	IMEM	EPC	cause	exception
BREAK	PC	PC	01001	1



33 SYSCALL

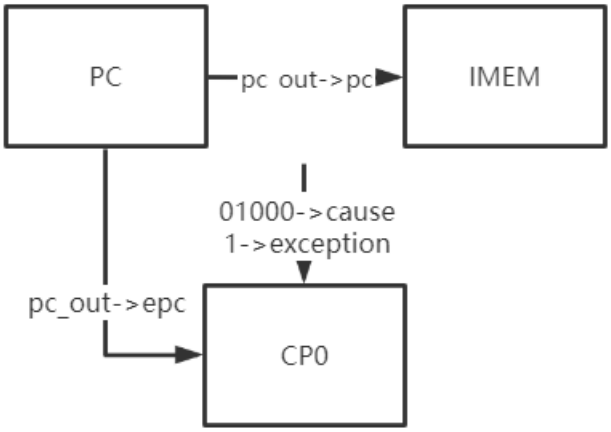
格式：SYSCALL

操作：取指令、EPC←PC, cause←01000, status<<5

所需部件：PC、IMEM、CP0

输入输出关系：

	IMEM	EPC	cause	exception
SYSCALL	PC	PC	01000	1



34 TEQ

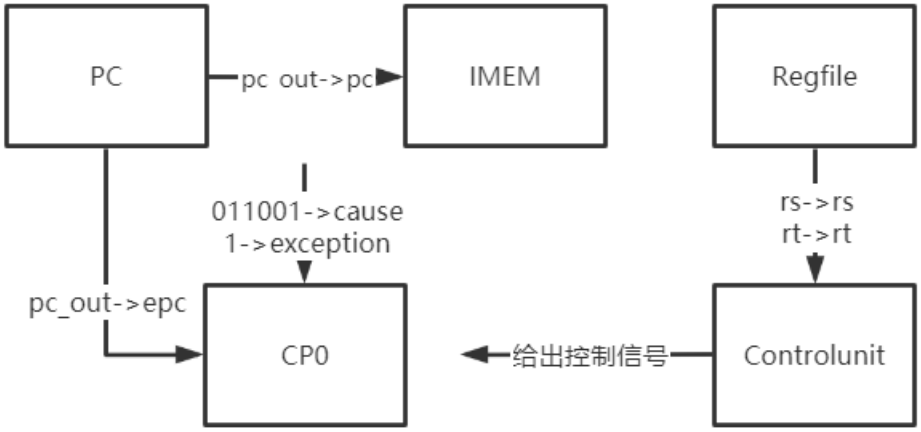
格式：TEQ rs,rt

操作取指令、EPC←PC、cause←01101, status<<5

所需部件：PC、CP0、IMEM、Regfile

输入输出关系：

	IMEM	EPC	cause	exception	Controlunit
TEQ	PC	PC	01101	1	Rs、Rt



35 ERET

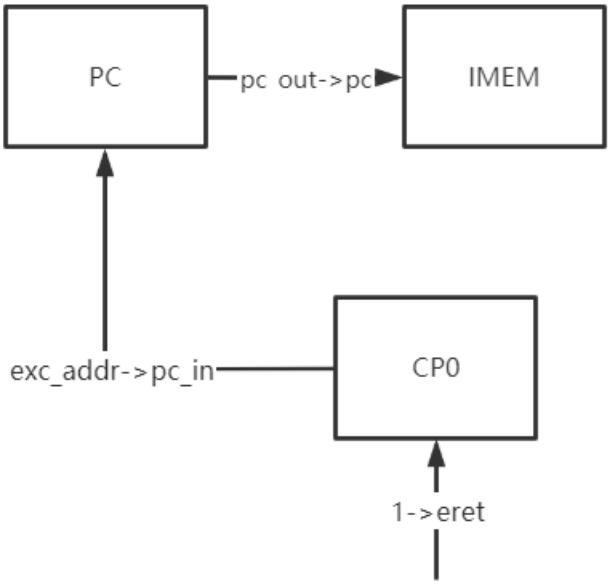
格式：ERET

操作：取指令、eret←1、PC←EPC

所需部件: PC、IMEM、CP0

输入输出关系：

	PC	IMEM	Eret
ERET	EPC	PC	1



36 MFC0

格式：MFC0 rt,rd

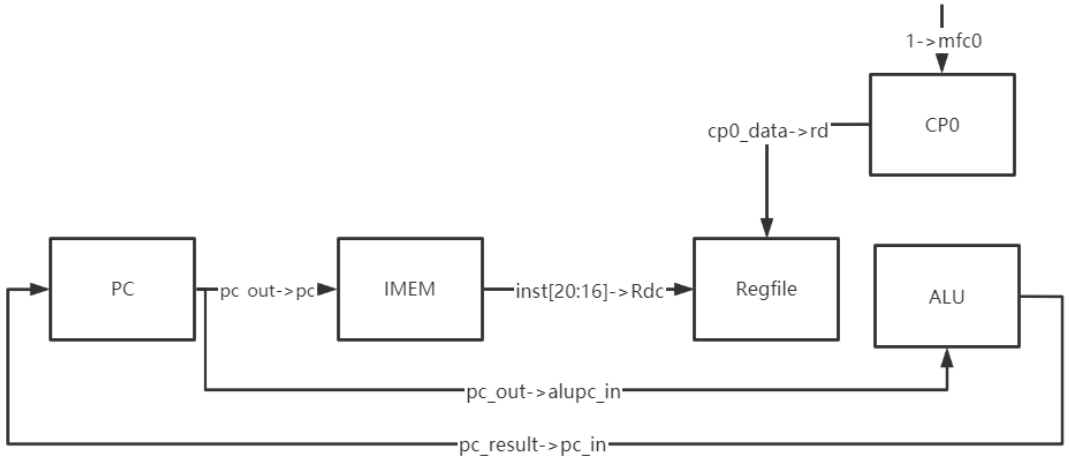
MFC0 rt,rd,sel

操作：取指令、 R[rt] ← CP0 R[rd], PC←PC+4

所需部件：PC、NPC、CP0、Rregfile

输入输出关系：

	PC	IMEM	Regfile		ALU			CP0
			Rd	Rdc	A	B	pc_in	Mfc0
MFC0	ALU	PC	Cp0_data	20-16			PC	1



37 MTC0

格式：MTC0 rt,rd

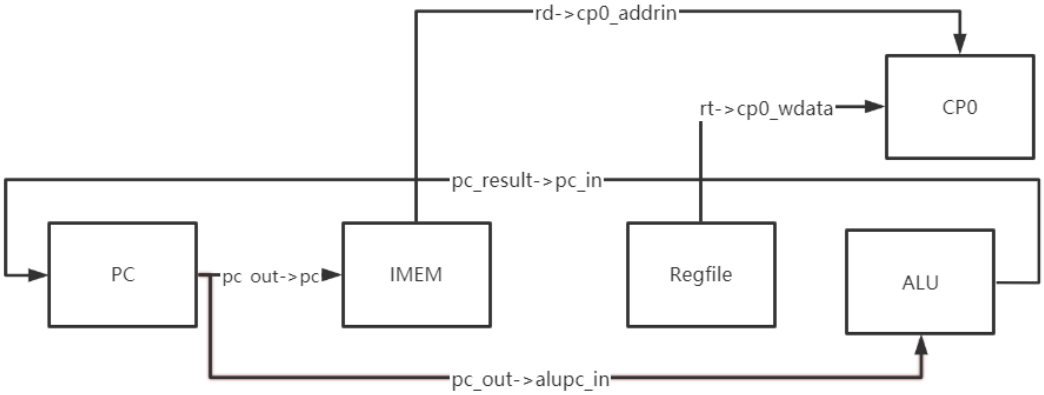
MTC0 rt,rd,sel

操作：取指令、 CP0 R[rd] ←R[rt], PC←PC+4

所需部件：PC、IMEM、CP0、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU			CP0	
			Rd	Rdc	A	B	pc_in	Addr	Data
MTC0	ALU	PC					PC	rd	Rt



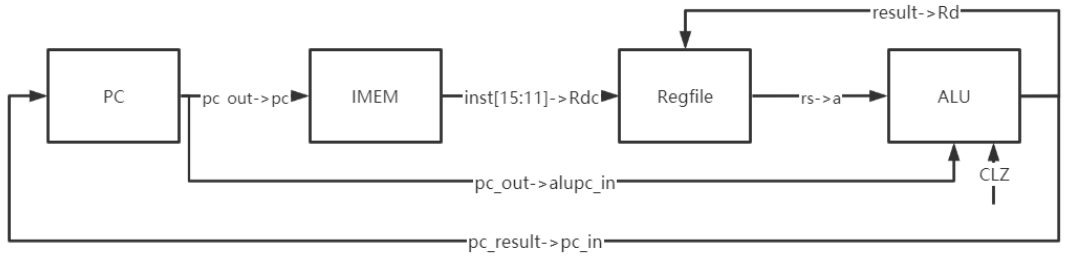
38、Clz rd rs

所需的操作: 取指令、R[rd]←count_leading_zeros R[rs], PC←PC+4

所需器件：PC、ALU、IMEM、Regfile

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
CLZ	ALU	PC	ALU	15-11	Rs		PC



39、DIVU rd rs

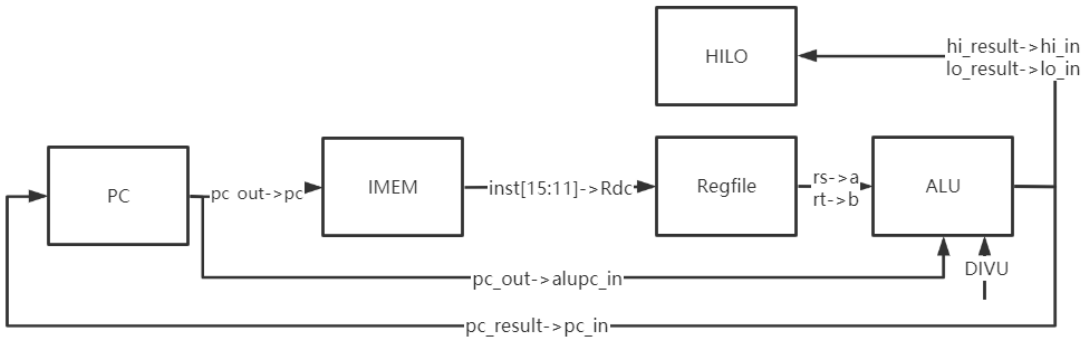
所需的操作: 取指令、(HI,LO)←R[rd]/R[rt], PC←PC+4

所需器件：PC、Regfile、ALU、IMEM、LO、HI

输入输出关系：

	PC	Regfile	IMEM	ALU			LO	HI
				A	B	pc_in		

DIVU	ALU		pc	rs	rt	PC	q	r



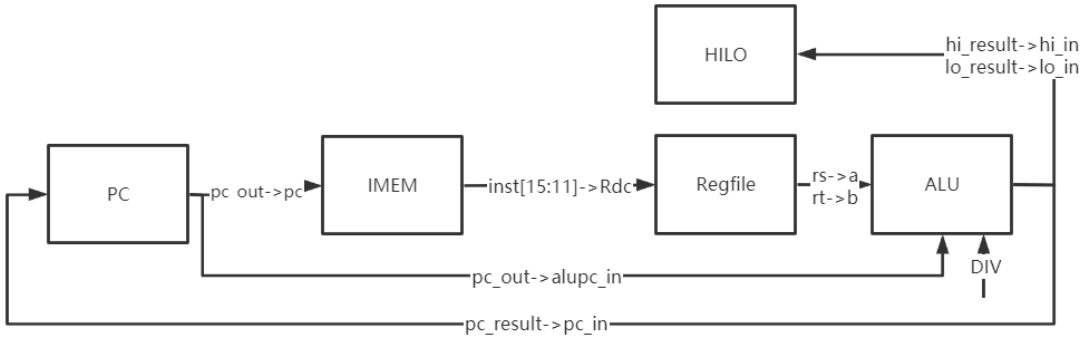
40、DIV rd rs

所需的操作：取指令、(HI,LO)←R[rd]/R[rt], PC←PC+4

所需器件：PC、Rregfile、IMEM、ALU、LO、HI

输入输出关系：

	PC	Regfile	IMEM	ALU			LO	HI
				A	B	pc_in		
DIVU	ALU		pc	rs	rt	PC	q	r



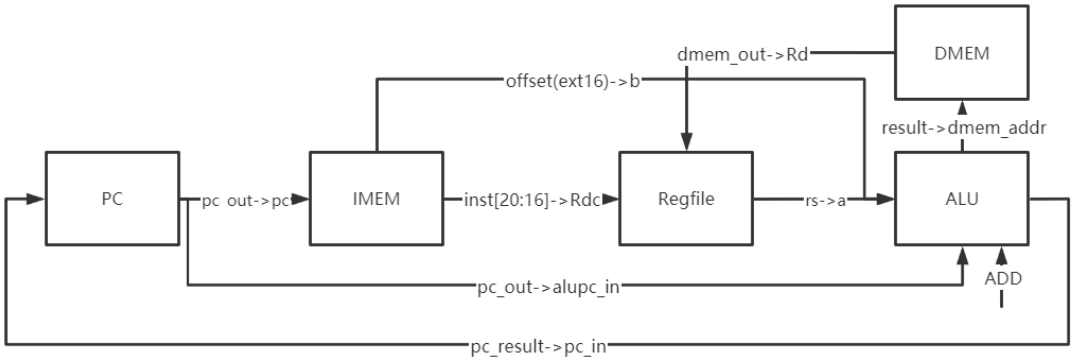
41、Lb rt,offset(base);

所需的操作：取指令、R[rt]←memory[R[base] + offset]、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU、DMEM

输入输出关系：

	PC	IMEM	Regfile		ALU			DMEM	
			Rd	Rdc	A	B	pc_in	Addr	Data
Lb	ALU	PC	DMEM	20-16	Rs(base)	Offset(Ext16)	PC	ALU	



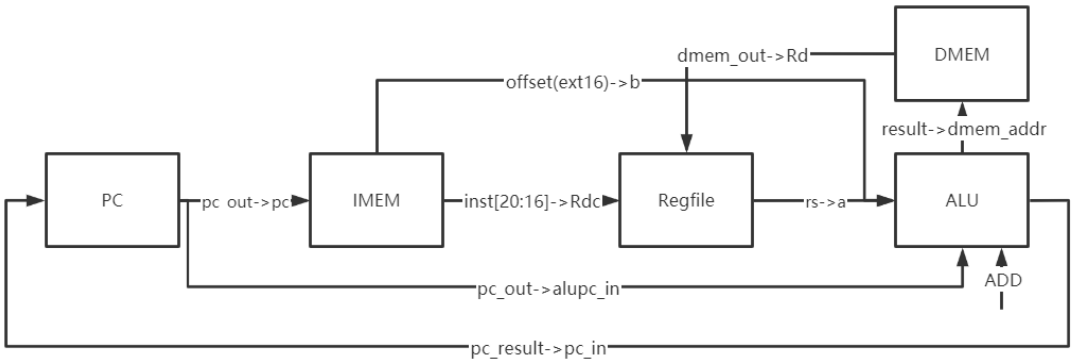
42、Lbu rt,offset(base);

所需的操作：取指令、R[rt]←memory[R[base] + offset]、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU、DMEM

输入输出关系：

	PC	IMEM	Regfile		ALU			DMEM	
			Rd	Rdc	A	B	pc_in	Addr	Data
Lbu	ALU	PC	DMEM	20-16	Rs(base)	Offset(Ext16)	PC	ALU	



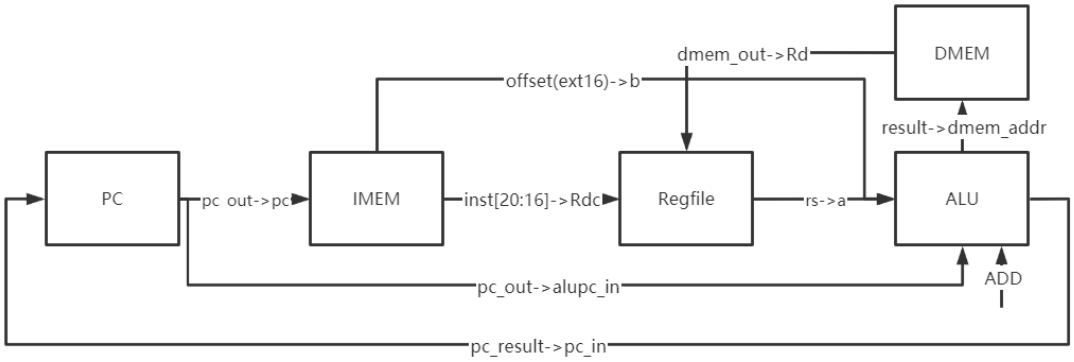
43、Lhu rt,offset(base);

所需的操作：取指令、R[rt]←memory[R[base] + offset]、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU、DMEM

输入输出关系：

	PC	IMEM	Regfile		ALU			DMEM	
			Rd	Rdc	A	B	pc_in	Addr	Data
Lhu	ALU	PC	DMEM	20-16	Rs(base)	Offset(Ext16)	PC	ALU	



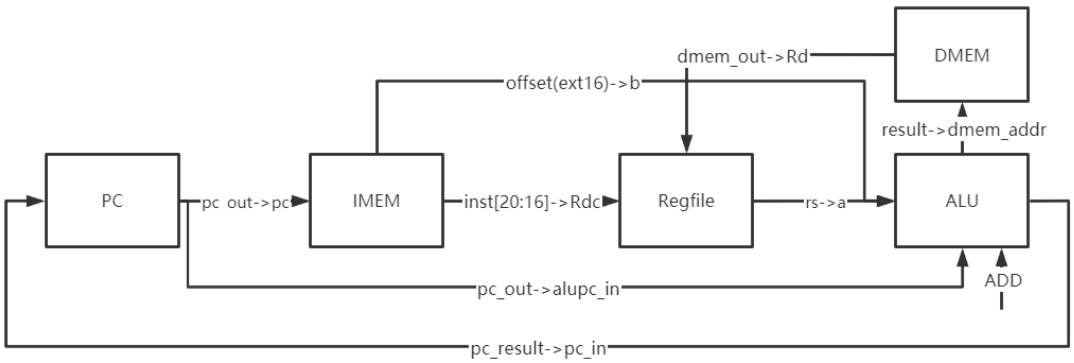
44、Lh　rt,offset(base);

所需的操作：取指令、R[rt]←memory[R[base] + offset]、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU、DMEM

输入输出关系：

	PC	IMEM	Regfile		ALU			DMEM	
			Rd	Rdc	A	B	pc_in	Addr	Data
Lh	ALU	PC	DMEM	20-16	Rs(base)	Offset(Ext16)	PC	ALU	



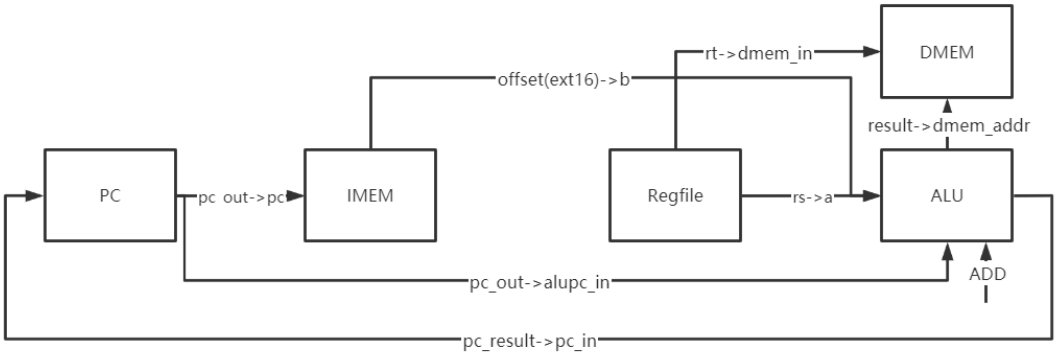
45、Sb　rt,offset(base);

所需的操作：取指令、memory[R[base] + offset] ← R[rt]、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU、DMEM

输入输出关系：

	PC	IMEM	Regfile		ALU			DMEM	
			Rd	Rdc	A	B	pc_in	Addr	Data
Sb	ALU	PC			Rs(base)	Offset(Ext16)	PC	ALU	Rt



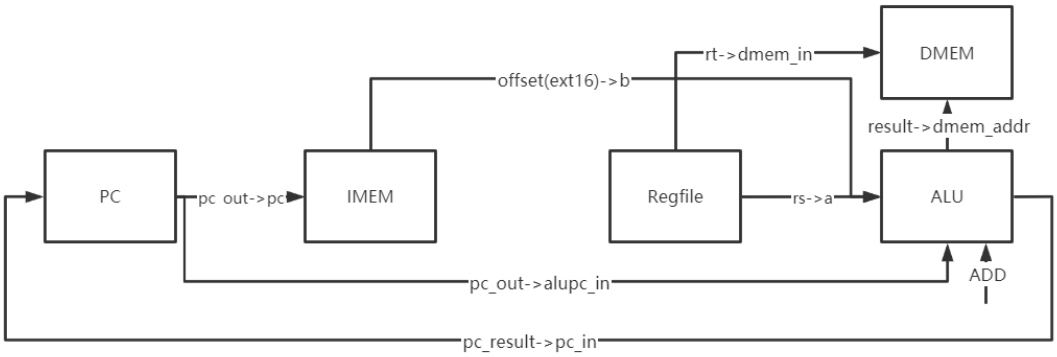
46、Sh rt,offset(base);

所需的操作：取指令、memory[R[base] + offset] ← R[rt]、PC←PC+4

所需部件：PC、IMEM、Regfile、ALU、DMEM

输入输出关系：

	PC	IMEM	Regfile		ALU			DMEM	
			Rd	Rdc	A	B	pc_in	Addr	Data
Sh	ALU	PC			Rs(base)	Offset(Ext16)	PC	ALU	Rt



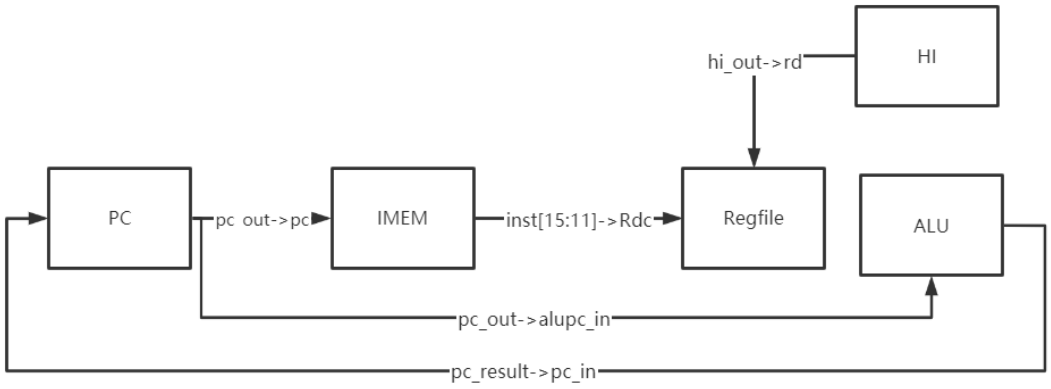
47、mfhi rd

所需的操作：取指令、 R[rd] ← HI、PC←PC+4

所需器件：PC、ALU、IMEM、Regfile、HI

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
MFHI	ALU	PC	HI	15-11			PC



48、mflo rd

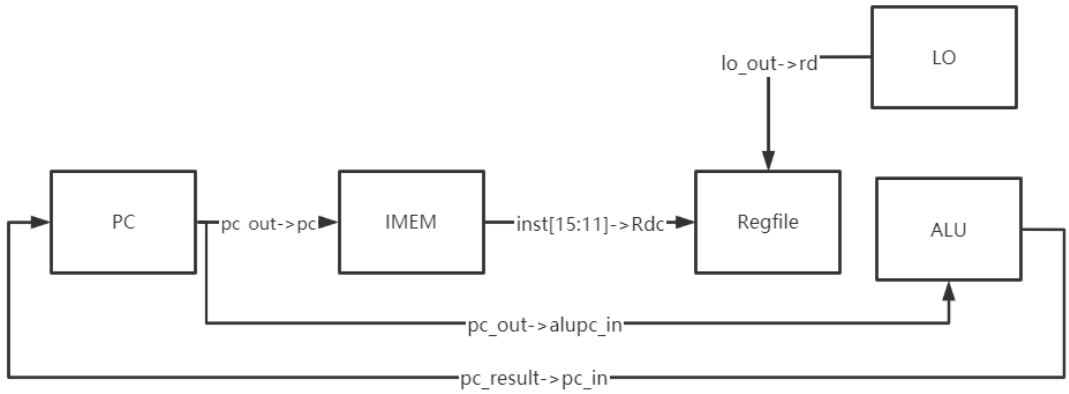
所需的操作：取指令、 R[rd] ← LO、PC←PC+4

所需器件：PC、NPC、IMEM、RegFile、LO

所需器件：PC、ALU、IMEM、Regfile、LO

输入输出关系：

	PC	IMEM	Regfile		ALU		
			Rd	Rdc	A	B	pc_in
MFLO	ALU	PC	LO	15-11			PC



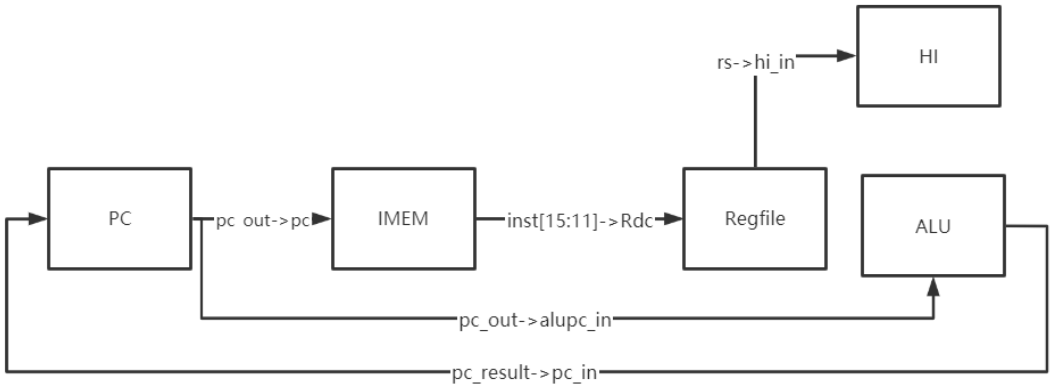
49、mthi rd

所需的操作：取指令、 HI←R[rs]、PC←PC+4

所需部件：PC、IMEM、HI、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU			HI
			Rd	Rdc	A	B	pc_in	
MTHI	ALU	PC					PC	Rs



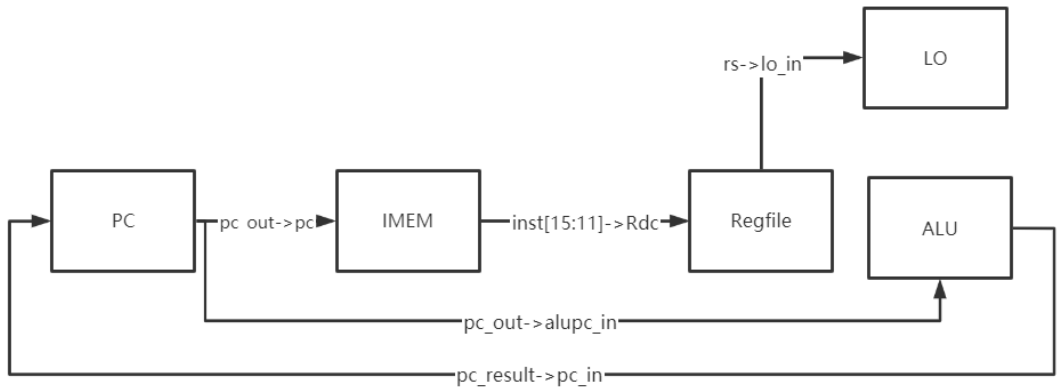
50、mtlo rd

所需的操作：取指令、 LO←R[rs]、PC←PC+4

所需部件：PC、IMEM、LO、Regfile、ALU

输入输出关系：

	PC	IMEM	Regfile		ALU			LO
			Rd	Rdc	A	B	pc_in	
MTHI	ALU	PC					PC	Rs



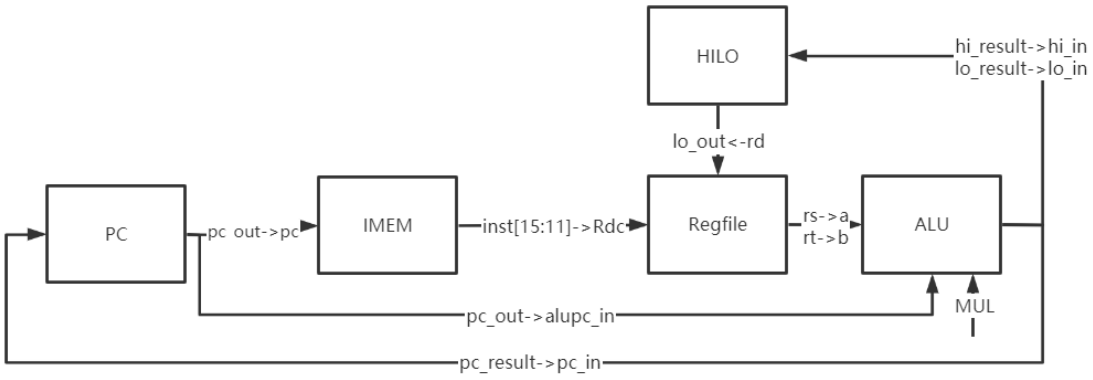
51、MUL rd rs

所需的操作：取指令、(HI,LO)←R[rs] * R[rt], PC←PC+4

所需部件：PC, IMEM, Regfile, ALU, HI, LO

输入输出关系：

	PC	IMEM	Regfile		ALU			HI	LO
			Rd	Rdc	A	B	pc_in		
MUL	ALU	PC	LO	15-11	Rs	Rt	PC	ALU	ALU



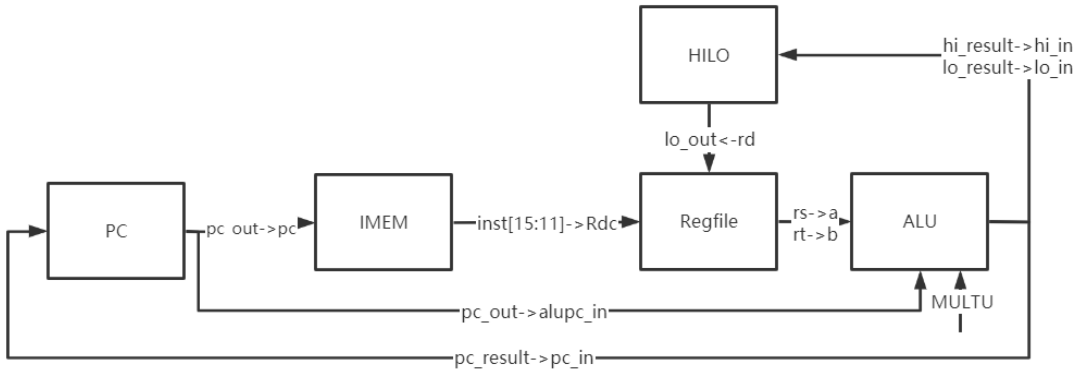
52、MULTU rd rs

所需的操作: 取指令、(HI,LO)←R[rs] * R[rt], PC←PC+4

所需部件：PC, IMEM, Regfile, ALU, HI, LO

输入输出关系：

	PC	IMEM	Regfile		ALU			HI	LO
			Rd	Rdc	A	B	pc_in		
MUL	ALU	PC	LO	15-11	Rs	Rt	PC	ALU	ALU



53、Bgez rs,rt,offset ;

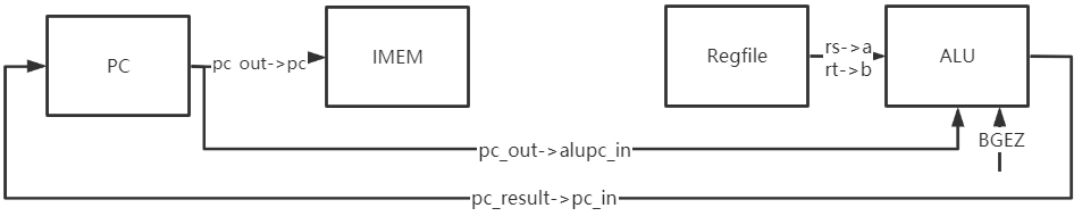
所需的操作: if (rs>0)PC←NPC + Sign_ext(offset||02)

else PC← NPC(PC+4)

所需部件：PC、IMEM、Regfile、ALU、ADD

输入输出关系：

	PC	IMEM	Rgefile		ALU		
			Rd	Rdc	A	B	pc_in
BNE	ALU	PC			Rs	Rt	PC



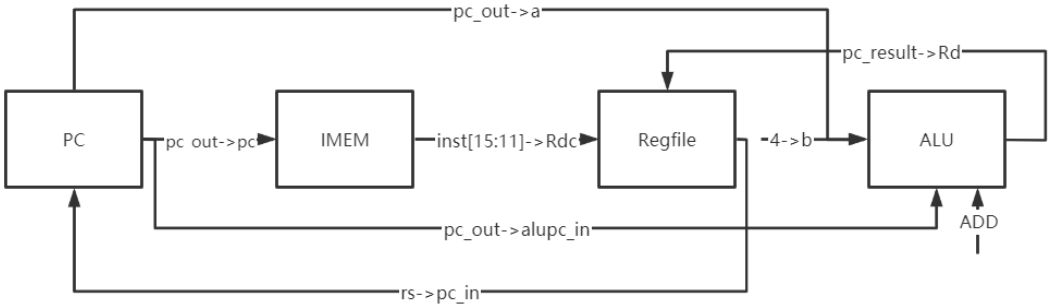
54、Jalr rd rs

所需的操作：取指令、R[rd]←PC + 4,PC←R[rs]

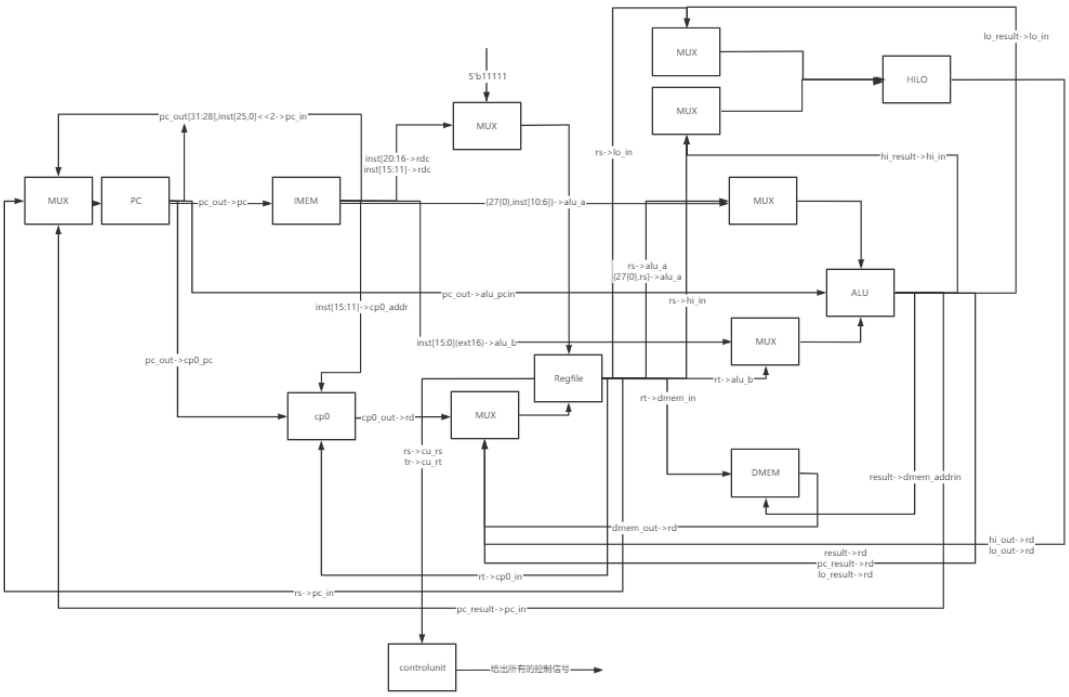
所需器件：PC、ALU、IMEM、Regfile

输入输出关系：

	PC	IMEM	RegFile		ALU		
			Rd	Rdc	A	B	Pc_in
Jalr	Rs	PC	ALU	15-11	PC	4	PC



2）整体数据通路



5.模块建模

1）sccomp_dataflow：

顶层模块，调用 cpu、imem、dmem;

```
`timescale 1ns / 1ps
module sccomp_dataflow(
    input clk_in, //时钟
    input reset,  //复位
    input pc_reset,
    output [31:0] inst, //32 位指令
    output [31:0] pc,  //指令地址
    output imemsrc
    );
    wire dmemsrc;
    wire [1:0]dmem_inchoice;
    wire [2:0]dmem_outchoice;
    wire [31:0]addr;
    wire [31:0]data_in;
    wire [31:0]data_out;

    imem imem(~clk_in, imemsrc, ((pc - 32'h00400000)/4), inst);//输出 PC，输出指令
```

```
    cpu sccpu(clk_in, reset, pc_reset, inst, pc, imemsrc,      dmemsrc,
dmem_inchoice, dmem_outchoice, addr, data_in, data_out); //CPU 相关功能

    dmem dmem(~clk_in, dmemsrc, dmem_inchoice, dmem_outchoice, addr -
32'h10010000, data_in, data_out);//
endmodule
```

2）imem：

存放指令，通过 pc 和取指信号来取指令；

```
module imem(
    input clk,//时钟
    input imemsrc,//可取指
    input [31:0] pc,//pc 信号
    output reg [31:0] inst//指令
    );
    reg [31:0] temp;//中间变量存储地址输出
    reg [31:0] ins[0:1023];//指令寄存器模块

    initial
    begin
        $readmemh("D:/cputest/cp0.hex.txt", ins);
    end

    always @(pc or imemsrc)
    begin
        // Read Instruction
        if (imemsrc) begin
            temp = ins[pc];
        end
    end

    always @(posedge clk)
    begin
        if(imemsrc)
            inst <= temp;
    end
endmodule
```

3）dmem：

数据存储器，可读写数据

```
`timescale 1ns / 1ps
module dmem(
    input clk,//时钟
    input dmemsrc,//可写
    input [1:0]dmem_inchoice,
    input [2:0]dmem_outchoice,
    input [31:0] addr,//地址
    input [31:0] data_in,//输入
    output reg [31:0] data_out//输出
    );
    wire [7:0]Addr;
    assign Addr = addr[7:0];
    wire [7:0]lh_lb;
    reg [7:0] num [0:1023];//寄存器
    assign lh_lb = num[Addr];
    always@(posedge clk)
    begin
        case(dmem_outchoice)
        3'b000:data_out = {num[Addr],num[Addr+1],num[Addr+2],num[Addr+3]};
        3'b001:data_out = {{16{lh_lb[7]}},num[Addr],num[Addr+1]};
        3'b010:data_out = {{16{1'b0}},num[Addr],num[Addr+1]};
        3'b011:data_out = {{24{lh_lb[7]}},num[Addr]};
        3'b100:data_out = {{24{1'b0}},num[Addr]};
        endcase
        if(dmemsrc==1'b1) //可写
        begin
            case (dmem_inchoice)
            2'b10:num[Addr]<=data_in[7:0];
            2'b01:
            begin
                num[Addr]<=data_in[15:8];
                num[Addr+1]<=data_in[7:0];
            end
            2'b00:
            begin
                num[Addr]<=data_in[31:24];
                num[Addr+1]<=data_in[23:16];
```

```
                num[Addr+2]<=data_in[15:8];
                num[Addr+3]<=data_in[7:0];
            end
            endcase
        end
    else
        num[Addr]<=num[Addr];
    end
endmodule
```

4）cpu：

cpu 主模块，对外和 imem、dmem 联系，对内负责协调 controlunit、pcreg、alu、regfile、hi_lo、cp0 等部件；

```
`timescale 1ns / 1ps
module cpu(
    input clk,//时钟
    input reset,//重置
    input pc_reset,
    input  [31:0] inst,//指令
    output [31:0] pc,//指令地址
    output imemsrc,//是否取指

    output dmemsrc,
    output [1:0]dmem_inchoice,
    output [2:0]dmem_outchoice,
    output [31:0]dmem_addrin,
    output [31:0]dmem_in,
    input [31:0]dmem_out
    );
    //alu 部分
    wire [31:0]alu_a;
    wire [31:0]alu_b;
    wire [31:0]imm;
    wire [31:0]alupc_in;
    wire alusrc;
    wire [4:0]aluchoice;
    wire [31:0]hi_result;
    wire [31:0]lo_result;
    wire [31:0]result;
```

```
wire [31:0]pc_result;

//refile 部分
wire [4:0]rf_addrin;
wire [31:0]rf_in;
wire regfilesrc;
wire [31:0]rs;
wire [31:0]rt;

//hi_lo 部分
wire [31:0]hi_in;
wire [31:0]lo_in;
wire hi_losrc;
wire [31:0]hi_out;
wire [31:0]lo_out;

//pcreg 部分
wire [31:0]pc_in;
wire pcsrc;
wire [31:0]pc_out;

//cp0
wire mfc0;
wire mtc0;
wire [31:0]cp0_pc;
wire [4:0]cp0_addr;
wire [31:0]cp0_in;
wire exception;
wire [4:0]cause;
wire [31:0]cp0_out;
wire [31:0]exc_addr;

//选择器部分
wire [1:0]alu_achoice;
wire [1:0]alu_bchoice;
wire [1:0]rf_addrinchoice;
wire [1:0]pc_inchoice;
wire [2:0]rf_inchoice;
wire [1:0]_dmem_inchoice;
```

```
    wire [2:0]_dmem_outchoice;
    wire hi_inchoice;
    wire lo_inchoice;

    //cpu 输出
    wire _imemsrc;
    assign imemsrc = _imemsrc;
    assign pc = pc_out;
    assign dmem_addrin = result;
    assign dmem_in = rt;

    wire _dmemsrc;
    assign dmemsrc = _dmemsrc;
    assign dmem_inchoice = _dmem_inchoice;
    assign dmem_outchoice = _dmem_outchoice;
    //指令译码
    controlunit cpu_controlunit(clk, pc_reset, inst, alusrc, aluchoice,
regfilesrc, hi_losrc, _imemsrc, pcsrc,
    alu_achoice, alu_bchoice, rf_addrinchoice, pc_inchoice, rf_inchoice,
_dmemsrc, _dmem_inchoice, _dmem_outchoice,
    mfc0, mtc0, exception, eret, cause, hi_inchoice, lo_inchoice, rs, rt);
    //选择器
    assign alu_a = alu_achoice==2'b00 ? rs : (alu_achoice==2'b01 ?
{{27{1'b0}},rs} : {{27{1'b0}},inst[10:6]});
    assign alu_b = alu_bchoice==2'b00 ? rt : (alu_bchoice==2'b01 ?
{{16{inst[15]}},inst[15:0]} : {{16{1'b0}},inst[15:0]});
    assign imm = {{(32 - 18){inst[15]}},inst[15:0],2'b00};
    assign alupc_in = pc_out;

    assign rf_addrin = rf_addrinchoice==2'b00 ? inst[15:11] :
(rf_addrinchoice==2'b01 ? inst[20:16] : 5'b11111);
    assign rf_in = rf_inchoice==3'b000 ? result : (rf_inchoice==3'b001 ?
pc_result : (rf_inchoice==3'b010 ? dmem_out :
    (rf_inchoice==3'b011 ? lo_result : (rf_inchoice==3'b100 ? cp0_out :
(rf_inchoice==3'b101 ? hi_out : lo_out)))));

    assign hi_in = hi_inchoice ? rs : hi_result;
    assign lo_in = lo_inchoice ? rs : lo_result;
```

```
    assign pc_in = pc_inchoice==2'b00 ? pc_result : (pc_inchoice==2'b01 ?
{pc_out[31:28],inst[25:0]<<2} : rs);

    assign cp0_pc = pc_out;
    assign cp0_addr = inst[15:11];
    assign cp0_in = rt;

    //部件
    alu      cpu_alu    (alu_a, alu_b, imm, alupc_in, alusrc, aluchoice,
hi_result, lo_result, result, pc_result);
    regfile  cpu_ref    (~clk, reset, inst[25:21], inst[20:16], rf_addrin,
rf_in, regfilesrc,                  rs, rt);
    hi_lo    cpu_hi_lo  (~clk, reset, hi_in, lo_in, hi_losrc,
hi_out, lo_out);
    pcreg    cpu_pcreg  (clk, pc_reset, pc_in, pcsrc,
pc_out);
    cp0      cpu_cp0    (~clk, reset, mfc0, mtc0, cp0_pc, cp0_addr, cp0_in,
exception, eret, cause, cp0_out, exc_addr);
endmodule
```

5）controlunit：

控制器，复杂译码并传递控制信号；

```
`timescale 1ns / 1ps
module controlunit(
    input clk,
    input reset,
    input [31:0] inst,
    output reg alusrc,
    output reg [4:0] aluchoice,
    output reg regfilesrc,
    output reg hi_losrc,
    output reg imemsrc,
    output reg pcsrc,
    output reg [1:0]alu_achoice,
    output reg [1:0]alu_bchoice,
    output reg [1:0]rf_addrinchoice,
    output reg [1:0]pc_inchoice,
    output reg [2:0]rf_inchoice,
```

```
output reg dmemsrc,
output reg [1:0]dmem_inchoice,
output reg [2:0]dmem_outchoice,

output reg mfc0src,
output reg mtc0src,
output reg exception,
output reg _eret,
output reg [4:0]cause,

output reg hi_inchoice,
output reg lo_inchoice,

input [31:0]rs,
input [31:0]rt
);
wire [16:0]moreop;
assign moreop = {inst[31:21],inst[5:0]};
wire [11:0]opcode;
assign opcode = {inst[31:26],inst[5:0]};
wire [5:0]halfop;
assign halfop = {inst[31:26]};

// 存储前一个状态和后一个状态
reg[2:0] presentState;
reg[2:0] nextState;
parameter [2:0]
sIF = 3'b000,
sID = 3'b001,
sEXE = 3'b010,
sMEM = 3'b011,
sWB = 3'b100;

parameter [5:0]
addi = 6'b001000,
addiu = 6'b001001,
andi = 6'b001100,
ori = 6'b001101,
sltiu = 6'b001011,
```

```
lui = 6'b001111,
xori = 6'b001110,
slti = 6'b001010,
beq = 12'b000100,
bne = 6'b000101,
bgez = 6'b000001,

j = 6'b000010,
jal = 6'b000011,

lw = 6'b100011,
sw = 6'b101011,
lb = 6'b100000,
lbu = 6'b100100,
lhu = 6'b100101,
sb = 6'b101000,
sh = 6'b101001,
lh = 6'b100001;

parameter [11:0]
addu = 12'b000000_100001,
_and = 12'b000000_100100,
_xor = 12'b000000_100110,
_nor = 12'b000000_100111,
_or = 12'b000000_100101,
sll = 12'b000000_000000,
sllv = 12'b000000_000100,
sltu = 12'b000000_101011,
sra = 12'b000000_000011,
srl = 12'b000000_000010,
subu = 12'b000000_100011,
add = 12'b000000_100000,
sub = 12'b000000_100010,
slt = 12'b000000_101010,
srlv = 12'b000000_000110,
srav = 12'b000000_000111,
clz = 12'b011100_100000,
divu = 12'b000000_011011,
mul = 12'b011100_000010,
```

```
multu = 12'b000000_011001,
teq = 12'b000000_110100,
div = 12'b000000_011010,

jr = 12'b000000_001000,
jalr = 12'b000000_001001,

mfhi = 12'b000000_010000,
mflo = 12'b000000_010010,
mthi = 12'b000000_010001,
mtlo = 12'b000000_010011,

eret = 12'b010000_011000,
syscall = 12'b000000_001100,
_break = 12'b000000_001101;

parameter [16:0]
mfc0 = 17'b010000_00000_000000,
mtc0 = 17'b010000_00100_000000;

//跳到下一个状态
always @(posedge clk)
begin
    if (reset)
        presentState <= sIF;
    else
    begin
        presentState <= nextState;
    end
end

//根据 opcode 和当前状态计算出下一个状态
always @(presentState or opcode)
begin
    case (presentState)
        sIF: nextState <= sID;
        sID: nextState <= sEXE;
        sEXE: nextState <= sMEM;
        sMEM: nextState <= sWB;
```

```
        sWB: nextState <= sIF;
    endcase
end

//根据 opcode 和当前状态确认当前控制信号
always @(presentState or opcode or posedge clk)
begin;
    //是否使用 alu
    if(presentState == sEXE)
        alusrc = 1;
    else
        alusrc = 0;
    //alu 的操作
    case(opcode)
        addu,addiu:aluchoice = 5'b00000;
        add,addi:aluchoice = 5'b00001;
        subu:aluchoice = 5'b00010;
        sub:aluchoice = 5'b00011;
        andi,_and:aluchoice = 5'b00100;
        ori,_or:aluchoice = 5'b00101;
        xori,_xor:aluchoice = 5'b00110;
        _nor:aluchoice = 5'b00111;
        lui:aluchoice = 5'b01000;
        sltu,sltiu:aluchoice = 5'b01001;
        slt,slti:aluchoice = 5'b01010;
        sra,srav:aluchoice = 5'b01011;
        srl,srlv:aluchoice = 5'b01100;
        sll,sllv:aluchoice = 5'b01101;
        beq:aluchoice = 5'b01110;
        bne:aluchoice = 5'b01111;
        bgez:aluchoice = 5'b10000;
        div:aluchoice = 5'b10001;
        divu:aluchoice = 5'b10010;
        mul:aluchoice = 5'b10011;
        multu:aluchoice = 5'b10100;
        clz:aluchoice = 5'b10101;
        teq:aluchoice = 5'b10110;//暂不处理
        default: aluchoice = 5'b00000;
    endcase
```

```
        case(halfop)
            addiu:aluchoice = 5'b00000;
            addi, lw, sw, lb, lbu, lhu, sb, sh, lh:aluchoice = 5'b00001;
            andi:aluchoice = 5'b00100;
            ori:aluchoice = 5'b00101;
            xori:aluchoice = 5'b00110;
            lui:aluchoice = 5'b01000;
            sltiu:aluchoice = 5'b01001;
            slti:aluchoice = 5'b01010;
            beq:aluchoice = 5'b01110;
            bne:aluchoice = 5'b01111;
            bgez:aluchoice = 5'b10000;
            default: aluchoice <= aluchoice;
        endcase
        //是否使用 regfile
        if (presentState == sWB && halfop != sw && halfop != sh && halfop != sb
&& halfop != beq && halfop != bne)
            regfilesrc = 1;
        else
            regfilesrc = 0;
        //是否使用 hi_lo
        if (presentState == sWB)
            hi_losrc = 1;
        else
            hi_losrc = 0;
        //是否取指
        if (presentState == sIF)
            imemsrc <= 1;
        else
            imemsrc <= 0;
        //是否使用 pcsrc
        if(presentState == sWB)
            pcsrc <= 1;
        else
            pcsrc <= 0;
        //alu_a 选择
        case(opcode)
            sll, srl, sra:alu_achoice = 2'b10;
            sllv, srlv, srav:alu_achoice = 2'b01;
```

```
            default: alu_achoice = 2'b00;
        endcase
        //alu_b 选择
        case(halfop)
            andi, ori, xori, lw, sw, lb, lbu, lhu, sb, sh, lh:alu_bchoice =
2'b10;
            addi,addiu, sltiu, lui, slti:alu_bchoice = 2'b01;
            default: alu_bchoice = 2'b00;
        endcase
        //rf_addrin 选择
        case(halfop)
            addi,addiu, andi, ori, sltiu, lui, xori, slti, lw, sw, lb, lbu,
lhu, sb, sh, lh:rf_addrinchoice = 2'b01;
            jal:rf_addrinchoice = 2'b10;
            default: rf_addrinchoice = 2'b00;
        endcase
        if(moreop == mfc0)
            rf_addrinchoice <= 2'b01;
        else
            rf_addrinchoice <= rf_addrinchoice;
        //pc_in 选择
        if(halfop == j || halfop == jal)
            pc_inchoice <= 2'b01;
        else if(opcode == jr || opcode == jalr)
            pc_inchoice <= 2'b10;
        else
            pc_inchoice <= 2'b00;
        //rf_in 选择
        if(halfop == jal || opcode == jalr)
            rf_inchoice <= 3'b001;
        else if(halfop == lw || halfop == lb || halfop == lbu || halfop == lhu
|| halfop == lh)
            rf_inchoice <= 3'b010;
        else if(opcode == mul || opcode == multu)
            rf_inchoice <= 3'b011;
        else if(moreop == mfc0)
            rf_inchoice <= 3'b100;
        else if(opcode == mfhi)
            rf_inchoice <= 3'b101;
```

```
        else if(opcode == mflo)
            rf_inchoice <= 3'b110;
        else
            rf_inchoice <= 0;
        //
        if(presentState == sMEM && (halfop == sw || halfop == sh || halfop ==
sb))
            dmemsrc <= 1;
        else
            dmemsrc <= 0;
        //
        case(halfop)
            sw:dmem_inchoice <= 2'b00;
            sh:dmem_inchoice <= 2'b01;
            sb:dmem_inchoice <= 2'b10;
            default: dmem_inchoice <= 2'b00;
        endcase
        //
        case(halfop)
            lw:dmem_outchoice <= 3'b000;
            lh:dmem_outchoice <= 3'b001;
            lhu:dmem_outchoice <= 3'b010;
            lb:dmem_outchoice <= 3'b011;
            lbu:dmem_outchoice <= 3'b100;
            default: dmem_outchoice <= 3'b000;
        endcase
        //
        case(opcode)
            teq:
            begin
                if(rs == rt)
                begin
                    exception <= 1;
                    _eret <= 0;
                    cause <= 5'b01101;
                end
            end
            _break:
            begin
```

```
        exception <= 1;
        _eret <= 0;
        cause <= 5'b01001;
    end
    eret:
    begin
        exception <= 0;
        _eret <= 1;
    end
    syscall:
    begin
        exception <= 1;
        _eret <= 0;
        cause <= 5'b01000;
    end
    default:
    begin
        exception <= 0;
        _eret <= 0;
        cause <= 5'b00000;
    end
endcase
if(moreop == mfc0)
begin
    mfc0src <= 1;
    mtc0src <= 0;
end
else if(moreop == mtc0)
begin
    mfc0src <= 0;
    mtc0src <= 1;
end
else
begin
    mfc0src <= 0;
    mtc0src <= 0;
end
if(opcode == mthi)
begin
```

```
            hi_inchoice <= 1;
            lo_inchoice <= 0;
        end
        else if(opcode == mtlo)
        begin
            hi_inchoice <= 0;
            lo_inchoice <= 1;
        end
        else
        begin
            hi_inchoice <= 0;
            lo_inchoice <= 0;
        end
    end
endmodule
```

6）alu：

运算器，负责各类运算，以及 pc 的更新；

```
`timescale 1ns / 1ps
module alu(
    input [31:0] A,
    input [31:0] B,
    input [31:0] imm,
    input [31:0] pc_in,
    input alusrc,
    input [4:0] aluchoice,
    output reg [31:0] hi_result,
    output reg [31:0] lo_result,
    output reg [31:0] result,
    output reg [31:0] pc_result
    );
    integer i; // CLZ 用
    integer j;
    integer max;

    always @(*)
    begin
        if(alusrc)
        begin
```

```
        pc_result <= pc_in + 32'b100;
        case(aluchoice)
            5'b00000: result <= A + B;
//ADDU
            5'b00001: result <= $signed(A) + $signed(B);                //ADD
            5'b00010: result <= A - B;
//SUBU
            5'b00011: result <= $signed(A) - $signed(B);               //SUB
            5'b00100: result <= A & B;                                   //AND
            5'b00101: result <= A | B;                                   //OR
            5'b00110: result <= A ^ B;                                   //XOR
            5'b00111: result <= ~(A|B);                                  //NOR
            5'b01000: result <= {B[15:0], 16'b0};                        //LUI
            5'b01001: result <= (A < B) ? 1 : 0;
//SLTU
            5'b01010: result <= ($signed(A)<$signed(B)) ? 1 : 0;       //SLT
            5'b01011: result <= $signed(B) >>> A;                        //SRA
向右算术移位
            5'b01100: result <= B >> A;                                  //SRL
向右逻辑移位
            5'b01101: result <= B << A;                                  //SLL
SLR
            5'b01110: pc_result <= (B == A) ? pc_in + imm + 32'b100 : pc_in +
32'b100;//BEQ 跳转地址
            5'b01111: pc_result <= (B != A) ? pc_in + imm + 32'b100 : pc_in +
32'b100;//BNE 跳转地址
            5'b10000: pc_result <= ($signed(A)>=0) ? pc_in + imm + 32'b100 :
pc_in + 32'b100; //BGEZ 生成跳转地址
            5'b10001: {hi_result, lo_result} <= {$signed(A) % $signed(B),
$signed(A) / $signed(B)};//DIV
            5'b10010: {hi_result, lo_result} <= {A % B, A / B};//DIVU
            5'b10011: {hi_result, lo_result} <= $signed(A) * $signed(B);// MUL
            5'b10100: {hi_result, lo_result} <= A * B; //MULTU
            5'b10101://CLZ
            begin
                j = 0;
                max = 0;
                for(i = 31; i >= 0;i = i-1)
                begin
```

```
                    if(A[i]==1'b1)
                        j = 1;
                    if(!j)
                        max = max + 1;
                end
                result <= max;
            end
        endcase
        end
    end
endmodule
```

7）regfile：

cpu 的寄存器，可读写、存储数据；

```
`timescale 1ns / 1ps
module regfile(
    input clk, //下降沿写入数据
    input rst, //高电平时全部寄存器置零
    input [4:0] raddr1, //所需读取的寄存器的地址
    input [4:0] raddr2, //所需读取的寄存器的地址
    input [4:0] waddr, //写寄存器的地址
    input [31:0] wdata, //写寄存器数据，数据在 clk 下降沿时被写入
    input regfilesrc,
    output [31:0] rdata1, //raddr1 所对应寄存器的输出数据
    output [31:0] rdata2 //raddr2 所对应寄存器的输出数据
    );
    reg [31:0] array_reg [31:0]; //寄存器

    //写寄存器
    integer i;
    always @(posedge clk or posedge rst)
    begin
        if(rst)
        begin
            for(i=0;i<32;i=i+1)
               array_reg[i] <= 0;
        end
        else if((waddr!=0)&&regfilesrc==1)
        begin
```

```
            array_reg[waddr] <= wdata;
        end
        else
        begin
            array_reg[waddr] <= array_reg[waddr];
        end
    end

    //输出数据
    assign rdata1 = array_reg[raddr1];
    assign rdata2 = array_reg[raddr2];
endmodule
```

8）hi_lo：

hi、lo 寄存器；

```
`timescale 1ns / 1ps
module hi_lo(
    input clk,
    input rst,
    input [31:0] Write_hi,
    input [31:0] Write_lo,
    input hi_losrc,
    output [31:0] Rd_hi,
    output [31:0] Rd_lo
    );
    reg [31:0] hi;
    reg [31:0] lo;
    assign Rd_hi = hi;
    assign Rd_lo = lo;

    always @(posedge clk)
    begin
        if(rst)
        begin
            hi <= 32'b0;
            lo <= 32'b0;
        end
        else
        begin
```

```
            if(hi_losrc)
            begin
                hi <= Write_hi;
                lo <= Write_lo;
            end
            else
            begin
                hi <= hi;
                lo <= lo;
            end
        end
    end
endmodule
```

9）pcreg：

pc 寄存器；

```
module pcreg(
    input clk,
    input rst,  //高电平时将 PC 寄存器清零
    input [31:0] data_in,
    input pcsrc,
    output reg [31:0] data_out
    );
    always @(posedge rst or posedge clk)
    begin
    if(rst)
        data_out <= 32'b0000_0000_0100_0000_0000_0000_0000_0000;
    else
    begin
        if(pcsrc)
            data_out <= data_in;
        else
            data_out <= data_out;
    end
    end
endmodule
```

10）cp0：

协处理器模块，负责 cpu 的中断及异常处理等；

```
module cp0(
    input clk,
    input rst,
    input mfc0,
    input mtc0,
    input [31:0]pc,
    input [4:0]Rd,//mfc0 读地址
    input [31:0]wdata,//mtc0 写数据
    input exception,//异常发生
    input eret,
    input [4:0]cause,//传递什么数据
    output [31:0]rdata,//mfc0 读数据
    output [31:0]exc_addr//返回给 pc 的地址
);
    integer i;
    reg[31:0]cp0[31:0];
    reg[31:0] status_temp;
    assign rdata = mfc0 ? cp0[Rd] : 32'hz;
    assign exc_addr = (eret==1) ? cp0[14]:32'h00400004;
    always@(posedge clk)
    begin
        if(rst)
        begin
            for(i=0;i<32;i=i+1)
                cp0[i]<=0;
        end
        else
        begin
        if(mtc0)
            cp0[Rd] <= wdata;
        else
            if(exception)
            begin
                status_temp<=cp0[12];
                    if(eret==1'b0)
                    begin
                        cp0[12]<=cp0[12]<<5;
                        cp0[13]<={25'b0,cause,2'b0};
                        cp0[14]<=pc;
```

```
                    end
                    else
                        cp0[12]<=status_temp;
            end
        end
    end
endmodule
```

6.测试模块建模

```
`timescale 1ns / 1ps

module cpu_tb();
    reg clk;
    reg rst;
    reg pc_rst;
    wire [31:0] inst;
    wire [31:0] pc;
    wire imemsrc;

    //看控制信号
    wire [2:0]presentstate = uut.sccpu.cpu_controlunit.presentState;
    wire alusrc = uut.sccpu.cpu_controlunit.alusrc;
    wire [4:0]aluchoice = uut.sccpu.cpu_controlunit.aluchoice;
    wire regfilesrc = uut.sccpu.cpu_controlunit.regfilesrc;
    wire hi_losrc = uut.sccpu.cpu_controlunit.hi_losrc;
    wire _imemsrc = uut.sccpu.cpu_controlunit.imemsrc;
    wire [1:0]alu_achoice = uut.sccpu.cpu_controlunit.alu_achoice;
    wire alu_bchoice = uut.sccpu.cpu_controlunit.alu_bchoice;
    wire rf_addrinchoice = uut.sccpu.cpu_controlunit.rf_addrinchoice;
    wire pcsrc = uut.sccpu.cpu_controlunit.pcsrc;
    wire [1:0]pc_inchoice = uut.sccpu.cpu_controlunit.pc_inchoice;
    wire [2:0]rf_inchoice = uut.sccpu.cpu_controlunit.rf_inchoice;

    //看 dmem
    wire dmemsrc = uut.sccpu.cpu_controlunit.dmemsrc;
    wire [1:0]dmem_inchoice = uut.sccpu.cpu_controlunit.dmem_inchoice;
    wire [2:0]dmem_outchocie = uut.sccpu.cpu_controlunit.dmem_outchoice;
```

```
//看 cp0
wire mfc0src = uut.sccpu.cpu_controlunit.mfc0src;
wire mtc0src = uut.sccpu.cpu_controlunit.mtc0src;
wire exception = uut.sccpu.cpu_controlunit.exception;
wire _eret = uut.sccpu.cpu_controlunit._eret;
wire [4:0]cause = uut.sccpu.cpu_controlunit.cause;
wire cp0_12 = uut.sccpu.cpu_cp0.cp0[12];
wire cp0_13 = uut.sccpu.cpu_cp0.cp0[13];
wire cp0_14 = uut.sccpu.cpu_cp0.cp0[14];
wire [31:0]exc_addr = uut.sccpu.cpu_cp0.exc_addr;

sccomp_dataflow uut(
   .clk_in(clk),
   .reset(rst),
   .pc_reset(pc_rst),
   .inst(inst),
   .pc(pc),
   .imemsrc(imemsrc)
    );
integer file_output;
integer counter;
initial
begin
   file_output = $fopen("D:/cpuresult/cp0test1.txt");//可根据需要调整
   clk = 0;
   rst = 1;
   pc_rst = 1;
   counter = 0;
   #5;
   rst = 0;
   # 20
   pc_rst = 0;
end

always
begin
#5;
clk = ~clk;
if (clk == 1'b1)
```

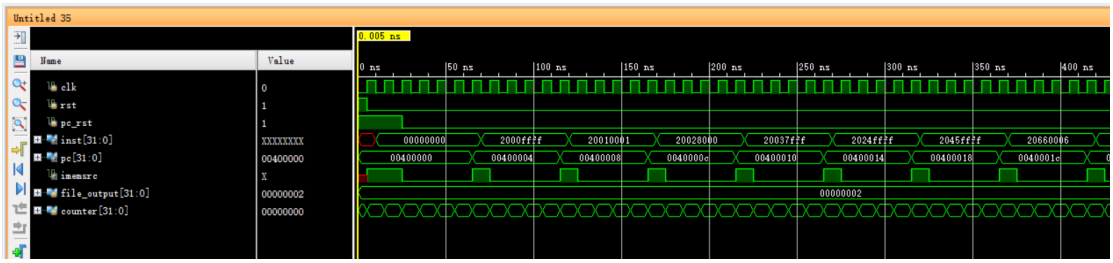
```
    begin
    if (counter == 4000)
    begin
       $fclose(file_output);
    end
    else begin
        counter = counter + 1;
//      if(counter % 5 == 2 && counter > 5)
        if(counter % 5 == 2 && counter > 5)
        begin
        $fdisplay(file_output,"pc:%h",pc-32'h00400000);
        $fdisplay(file_output,"instr:%h",uut.inst);
        $fdisplay(file_output,"regfile0: %h",uut.sccpu.cpu_ref.array_reg[0]);
        $fdisplay(file_output,"regfile1: %h",uut.sccpu.cpu_ref.array_reg[1]);
        $fdisplay(file_output,"regfile2: %h",uut.sccpu.cpu_ref.array_reg[2]);
        $fdisplay(file_output,"regfile3: %h",uut.sccpu.cpu_ref.array_reg[3]);
        $fdisplay(file_output,"regfile4: %h",uut.sccpu.cpu_ref.array_reg[4]);
        $fdisplay(file_output,"regfile5: %h",uut.sccpu.cpu_ref.array_reg[5]);
        $fdisplay(file_output,"regfile6: %h",uut.sccpu.cpu_ref.array_reg[6]);
        $fdisplay(file_output,"regfile7: %h",uut.sccpu.cpu_ref.array_reg[7]);
        $fdisplay(file_output,"regfile8: %h",uut.sccpu.cpu_ref.array_reg[8]);
        $fdisplay(file_output,"regfile9: %h",uut.sccpu.cpu_ref.array_reg[9]);
        $fdisplay(file_output,"regfile10: %h",uut.sccpu.cpu_ref.array_reg[10]);
        $fdisplay(file_output,"regfile11: %h",uut.sccpu.cpu_ref.array_reg[11]);
        $fdisplay(file_output,"regfile12: %h",uut.sccpu.cpu_ref.array_reg[12]);
        $fdisplay(file_output,"regfile13: %h",uut.sccpu.cpu_ref.array_reg[13]);
        $fdisplay(file_output,"regfile14: %h",uut.sccpu.cpu_ref.array_reg[14]);
        $fdisplay(file_output,"regfile15: %h",uut.sccpu.cpu_ref.array_reg[15]);
        $fdisplay(file_output,"regfile16: %h",uut.sccpu.cpu_ref.array_reg[16]);
        $fdisplay(file_output,"regfile17: %h",uut.sccpu.cpu_ref.array_reg[17]);
        $fdisplay(file_output,"regfile18: %h",uut.sccpu.cpu_ref.array_reg[18]);
        $fdisplay(file_output,"regfile19: %h",uut.sccpu.cpu_ref.array_reg[19]);
        $fdisplay(file_output,"regfile20: %h",uut.sccpu.cpu_ref.array_reg[20]);
        $fdisplay(file_output,"regfile21: %h",uut.sccpu.cpu_ref.array_reg[21]);
        $fdisplay(file_output,"regfile22: %h",uut.sccpu.cpu_ref.array_reg[22]);
        $fdisplay(file_output,"regfile23: %h",uut.sccpu.cpu_ref.array_reg[23]);
        $fdisplay(file_output,"regfile24: %h",uut.sccpu.cpu_ref.array_reg[24]);
        $fdisplay(file_output,"regfile25: %h",uut.sccpu.cpu_ref.array_reg[25]);
        $fdisplay(file_output,"regfile26: %h",uut.sccpu.cpu_ref.array_reg[26]);
```

```
            $fdisplay(file_output,"regfile27: %h",uut.sccpu.cpu_ref.array_reg[27]);
            $fdisplay(file_output,"regfile28: %h",uut.sccpu.cpu_ref.array_reg[28]);
            $fdisplay(file_output,"regfile29: %h",uut.sccpu.cpu_ref.array_reg[29]);
            $fdisplay(file_output,"regfile30: %h",uut.sccpu.cpu_ref.array_reg[30]);
            $fdisplay(file_output,"regfile31: %h",uut.sccpu.cpu_ref.array_reg[31]);
            end
          end
      end
      end
endmodule
```

在测试时，只需要修改 cpu_tb 以及 imem 的相应文件路径即可；用$readmemh 初始化 imem，读取相应的指令，进行测试。

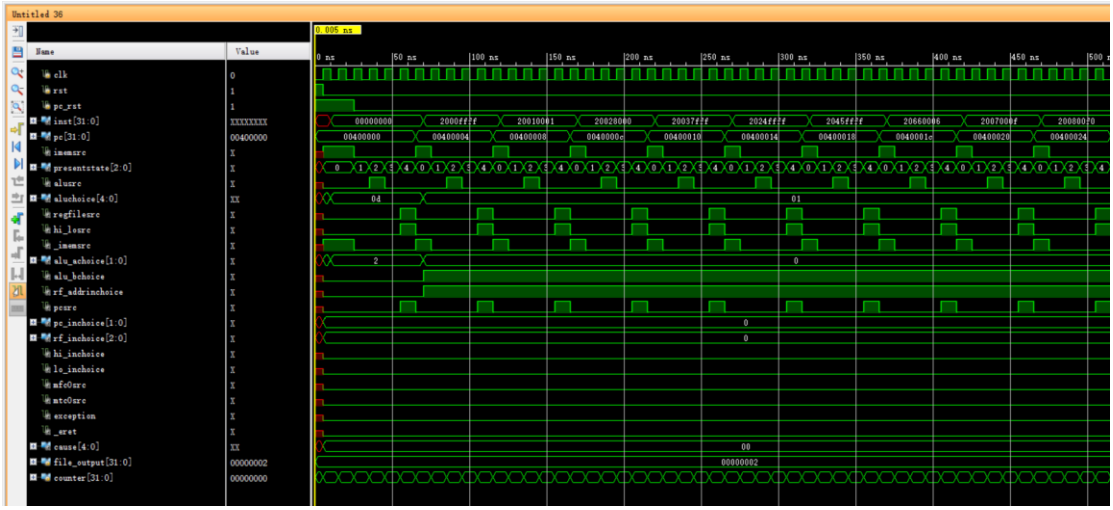
7. 实验结果

1）pc 更新以及指令正常测试：



pc 更新及指令获取正常。

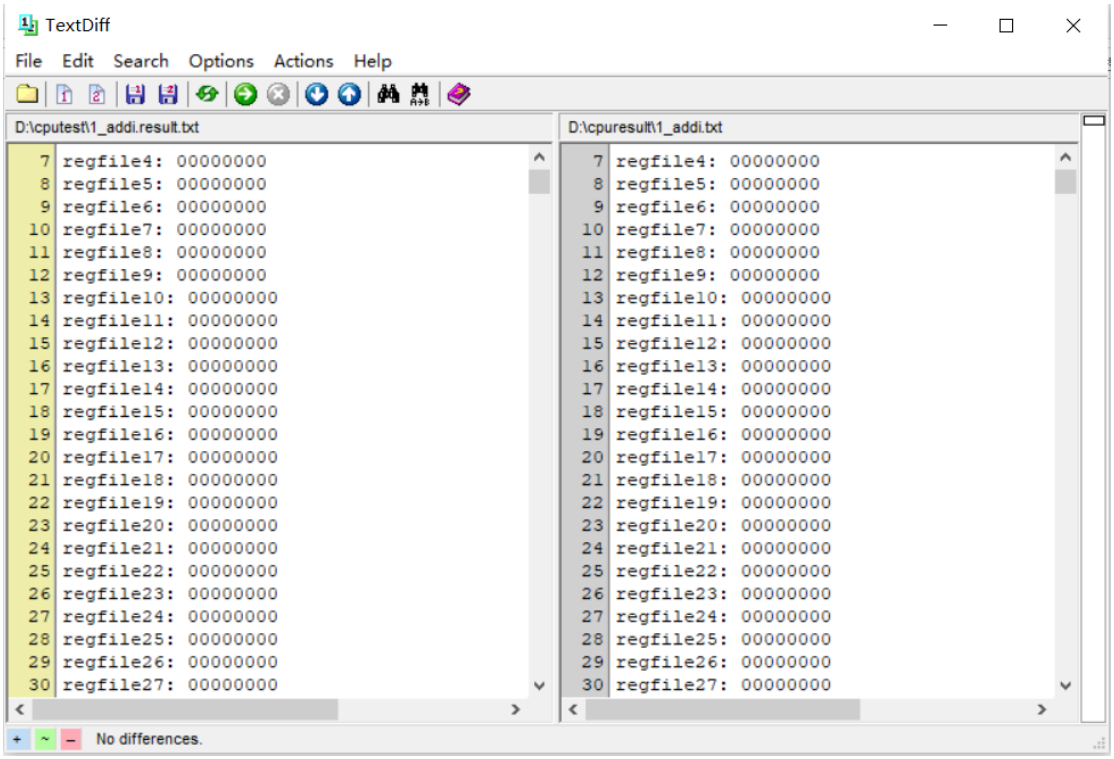
2）controlunit 信号测试：



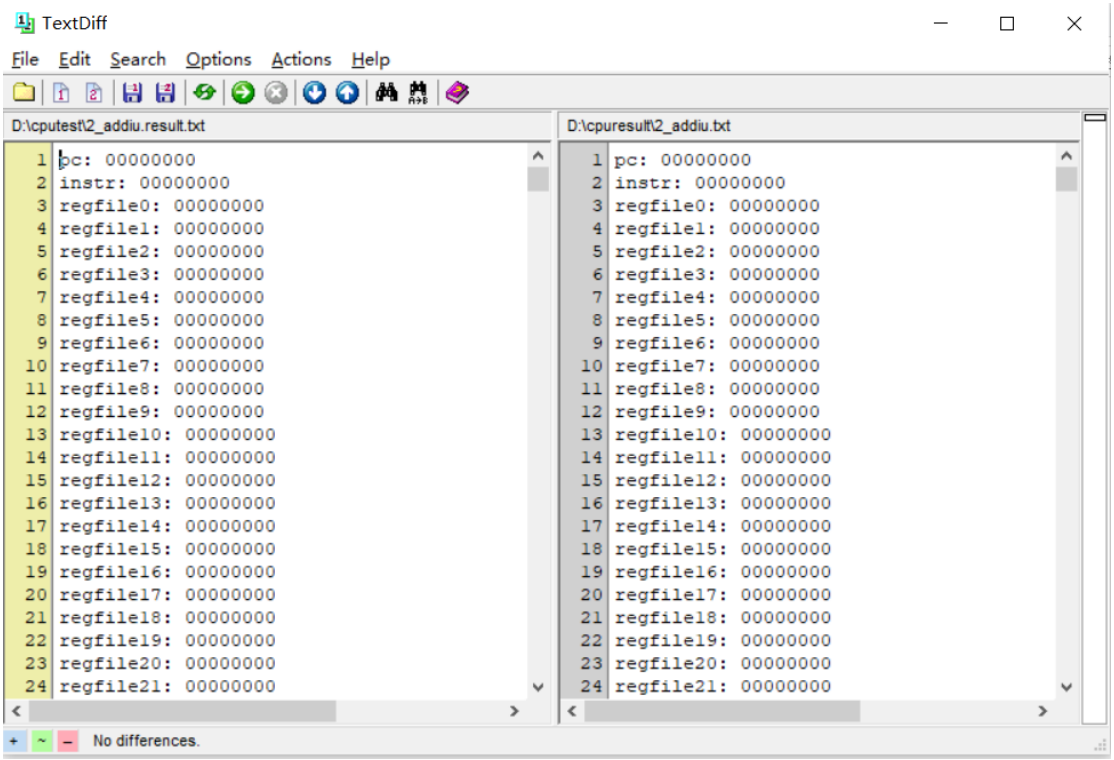
控制信号正常。

3）54 条指令：

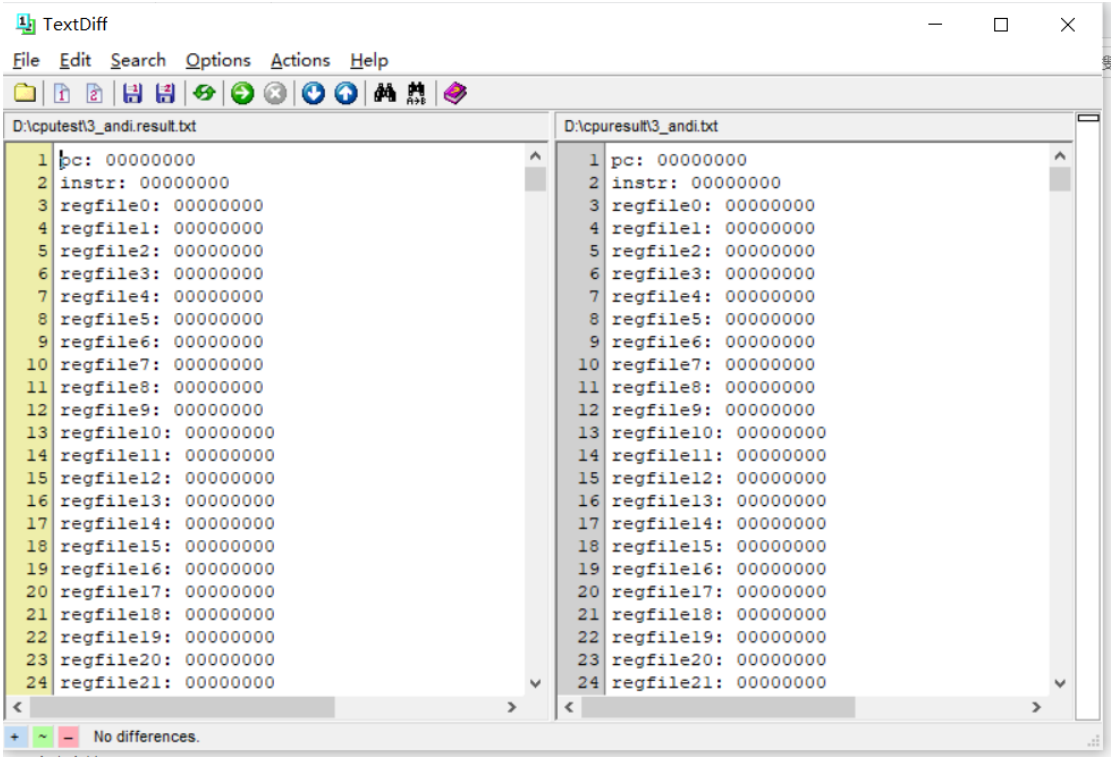
Addi:



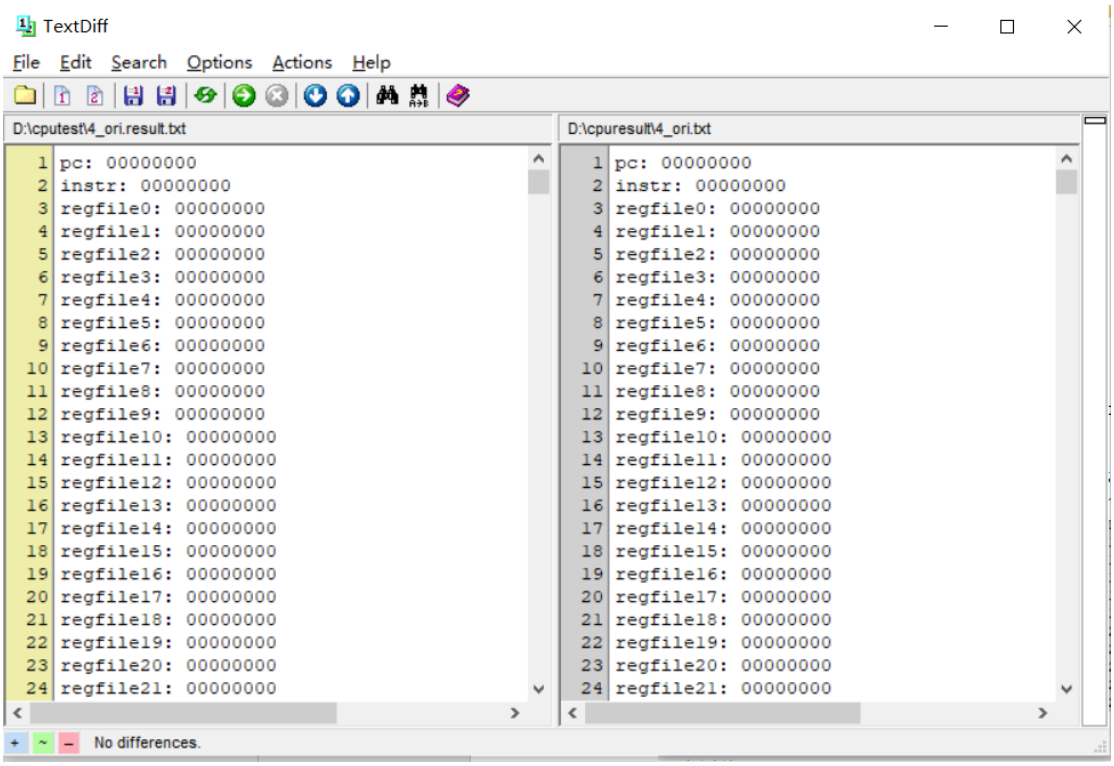
Addiu:



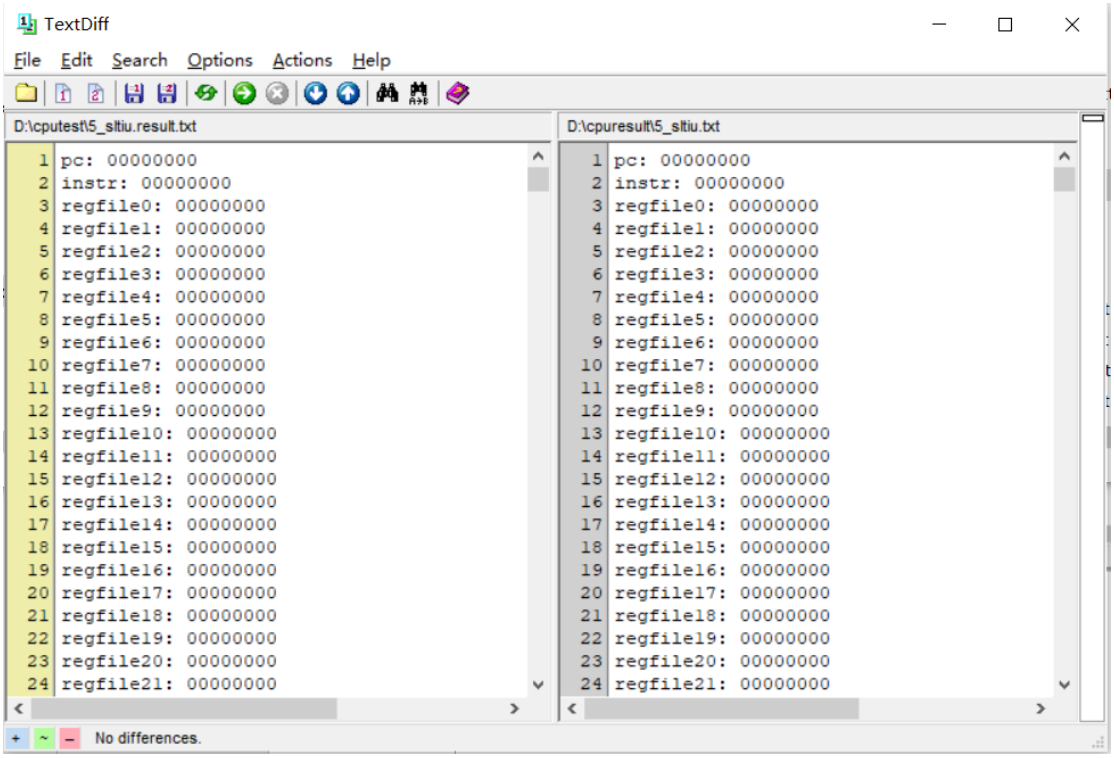
Andi:



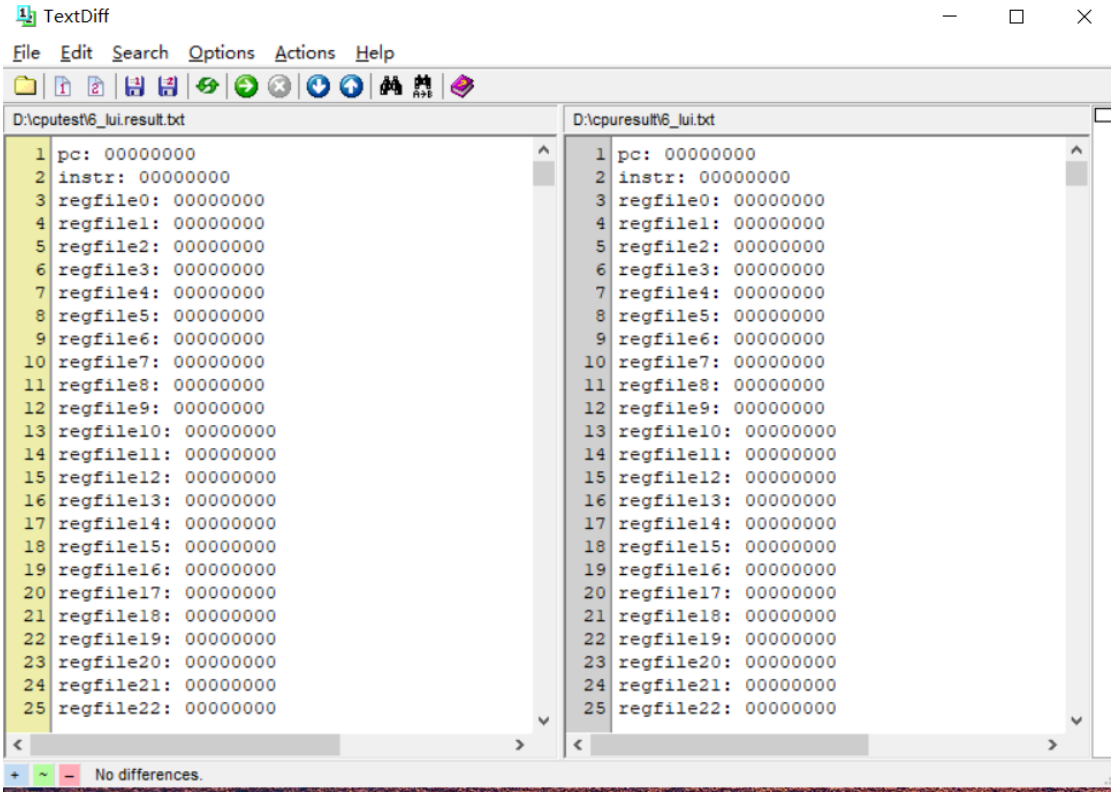
Ori:



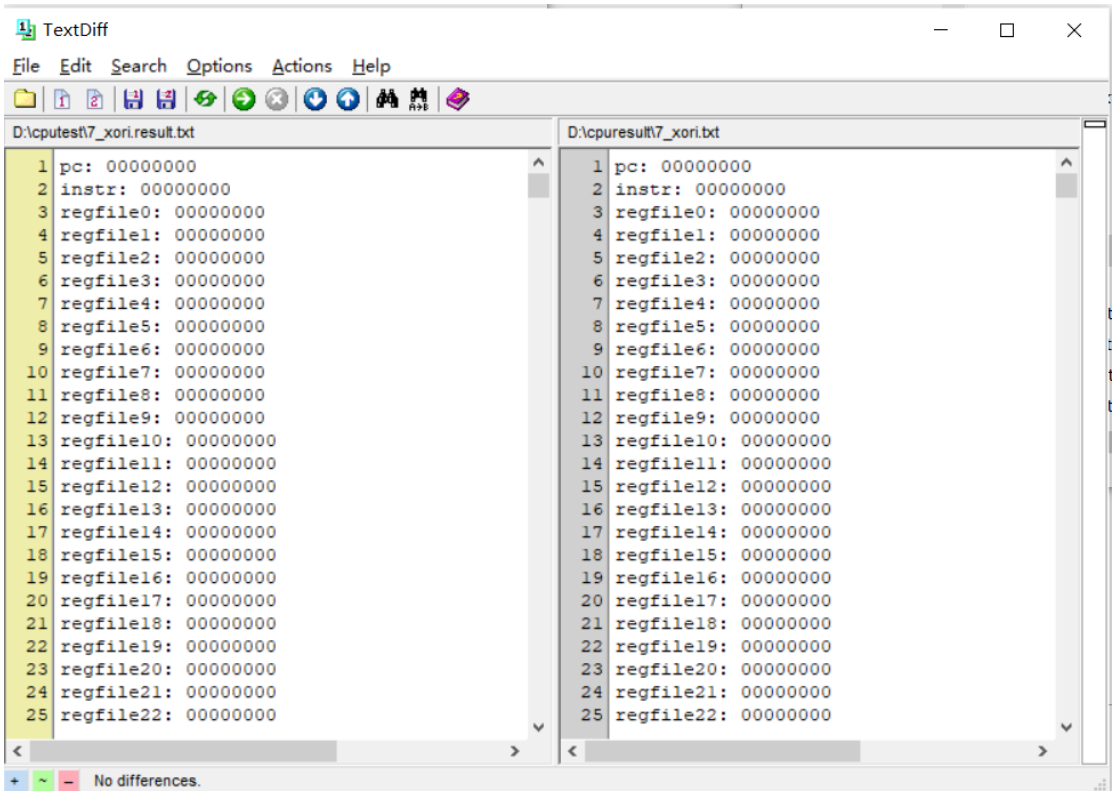
Sltiu:


TextDiff
File Edit Search Options Actions Help
D:\cputest\5_sltiu.result.txt
D:\cpuresult\5_sltiu.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
No differences.

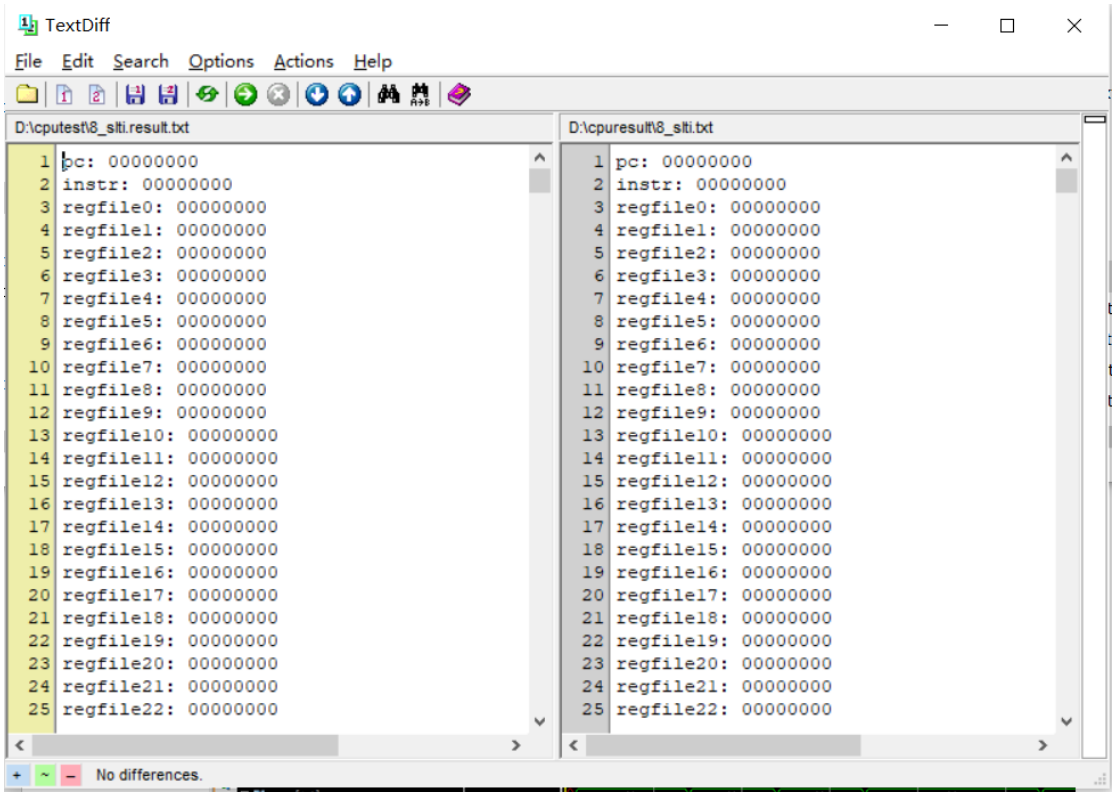
Lui：


TextDiff
File Edit Search Options Actions Help
D:\cputest\6_lui.result.txt
D:\cpuresult\6_lui.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

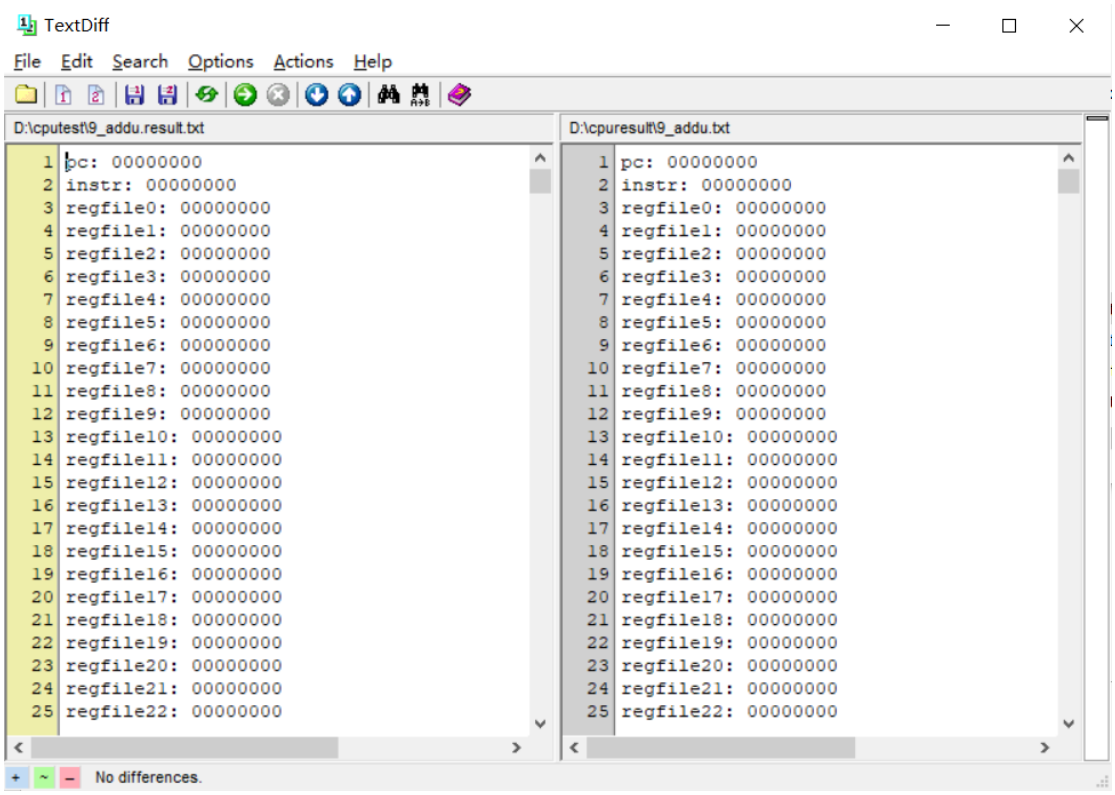
Xori：


TextDiff
File Edit Search Options Actions Help
D:\cputest\7_xori.result.txt
D:\cpuresult\7_xori.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

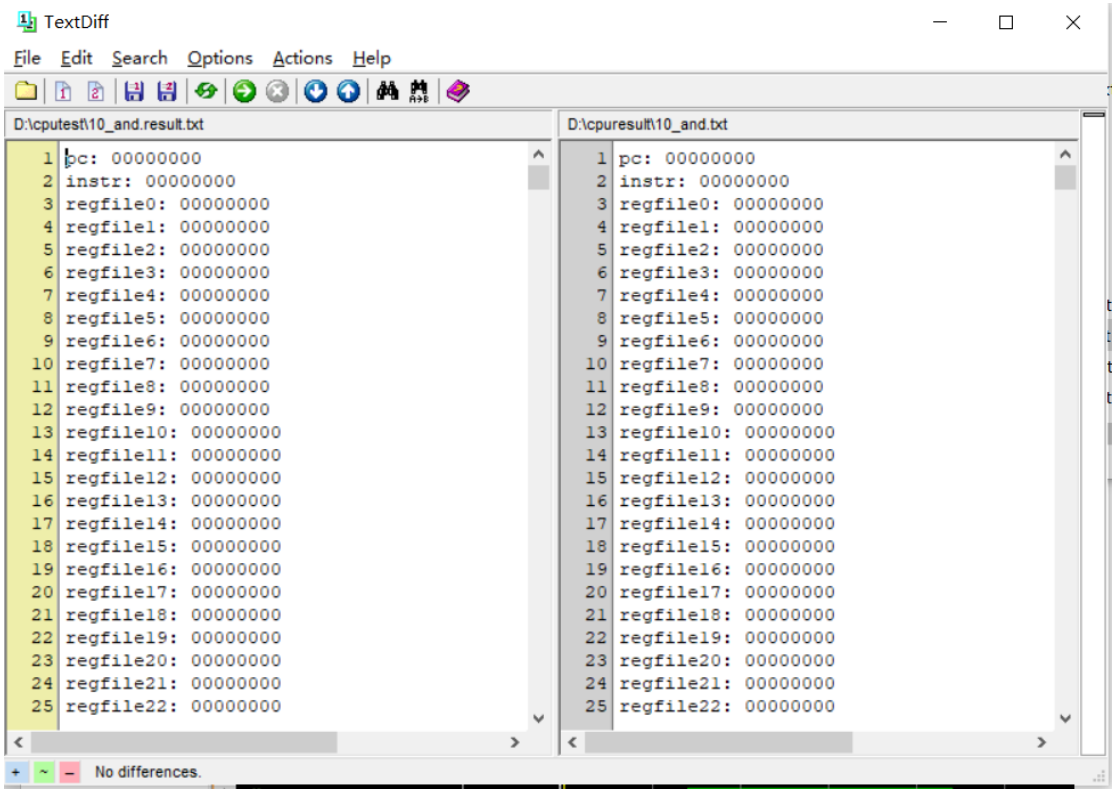
Slti:


TextDiff
File Edit Search Options Actions Help
D:\cputest\8_slti.result.txt
D:\cpuresult\8_slti.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

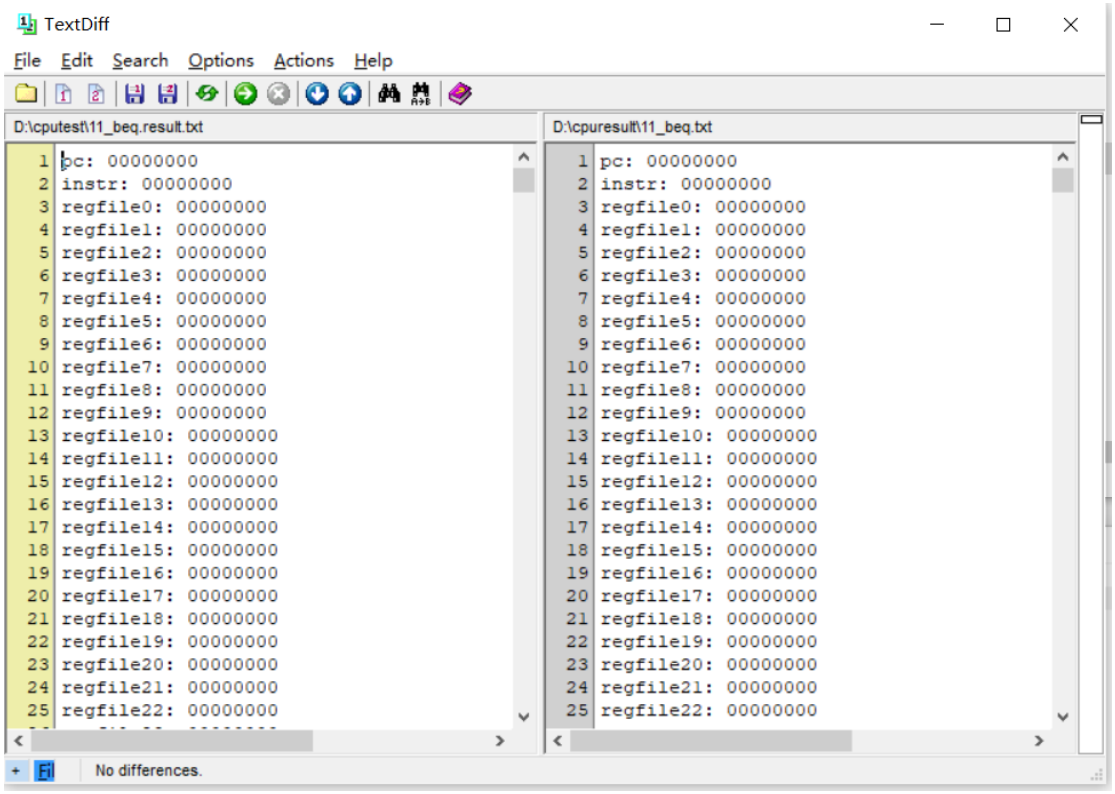
Addu:


TextDiff
File Edit Search Options Actions Help
D:\cputest\9_addu.result.txt
D:\cpuresult\9_addu.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

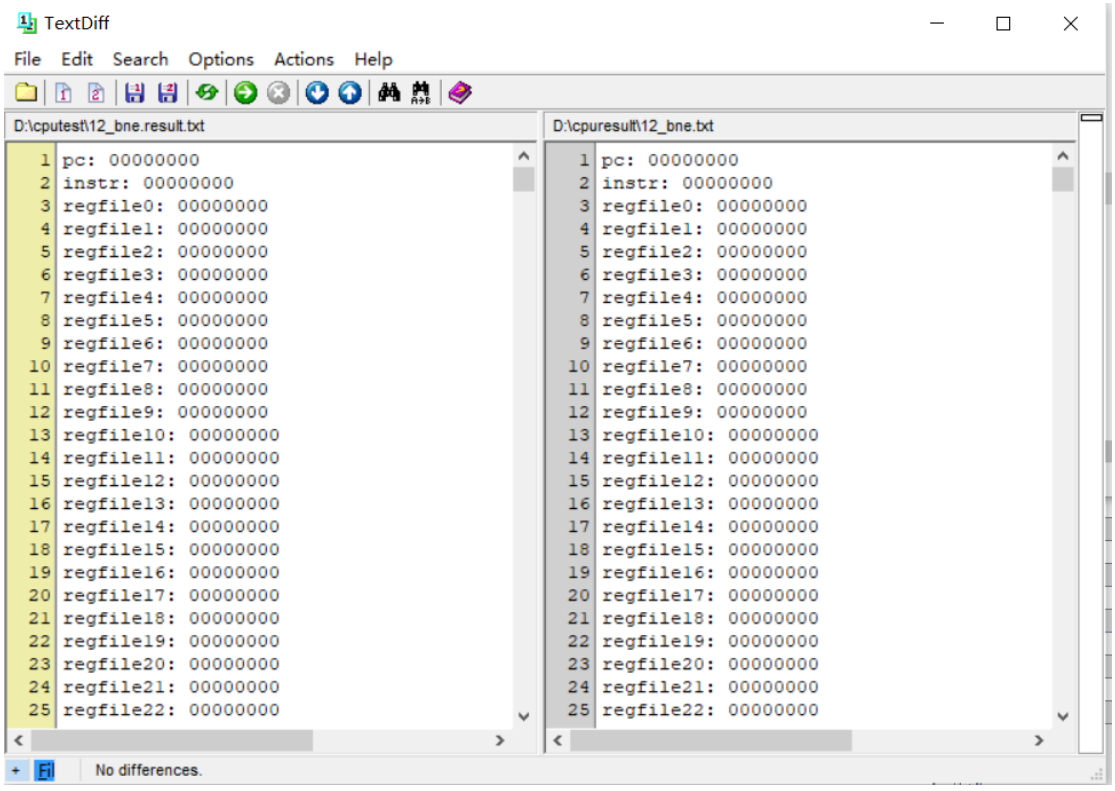
And:


TextDiff
File Edit Search Options Actions Help
D:\cputest\10_and.result.txt
D:\cpuresult\10_and.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

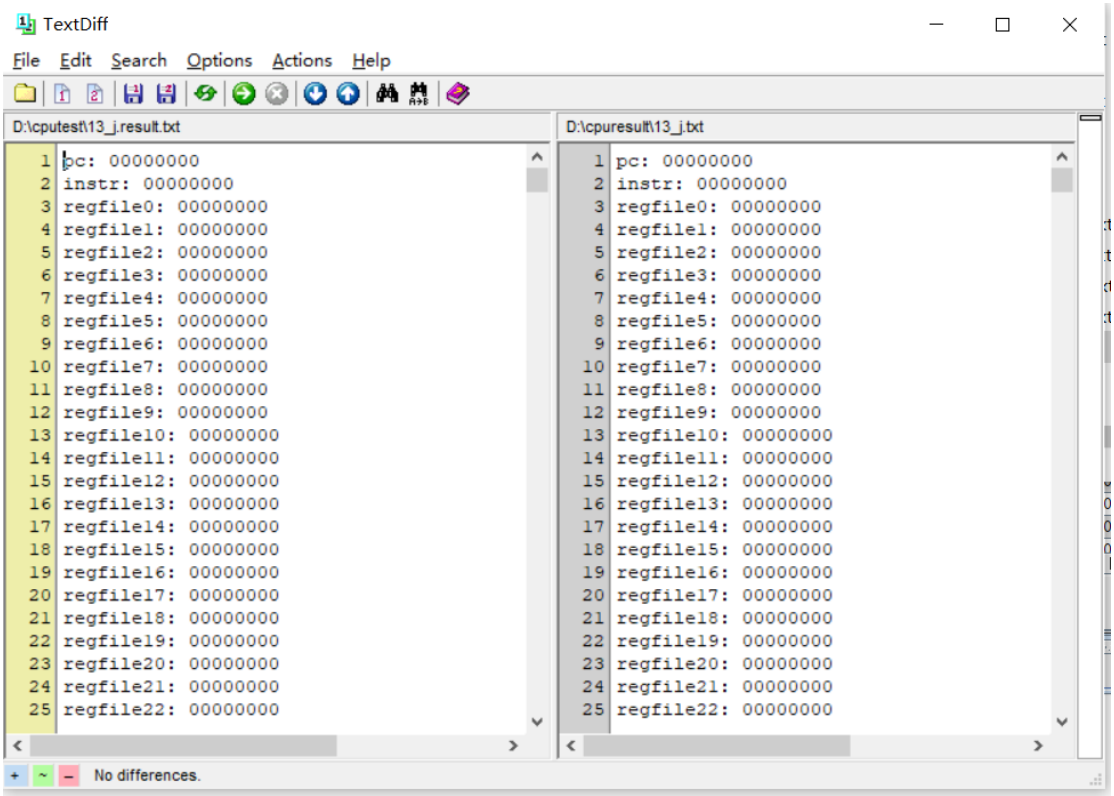
Beq:


TextDiff
File Edit Search Options Actions Help
D:\cputest\11_beq.result.txt
D:\cpuresult\11_beq.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

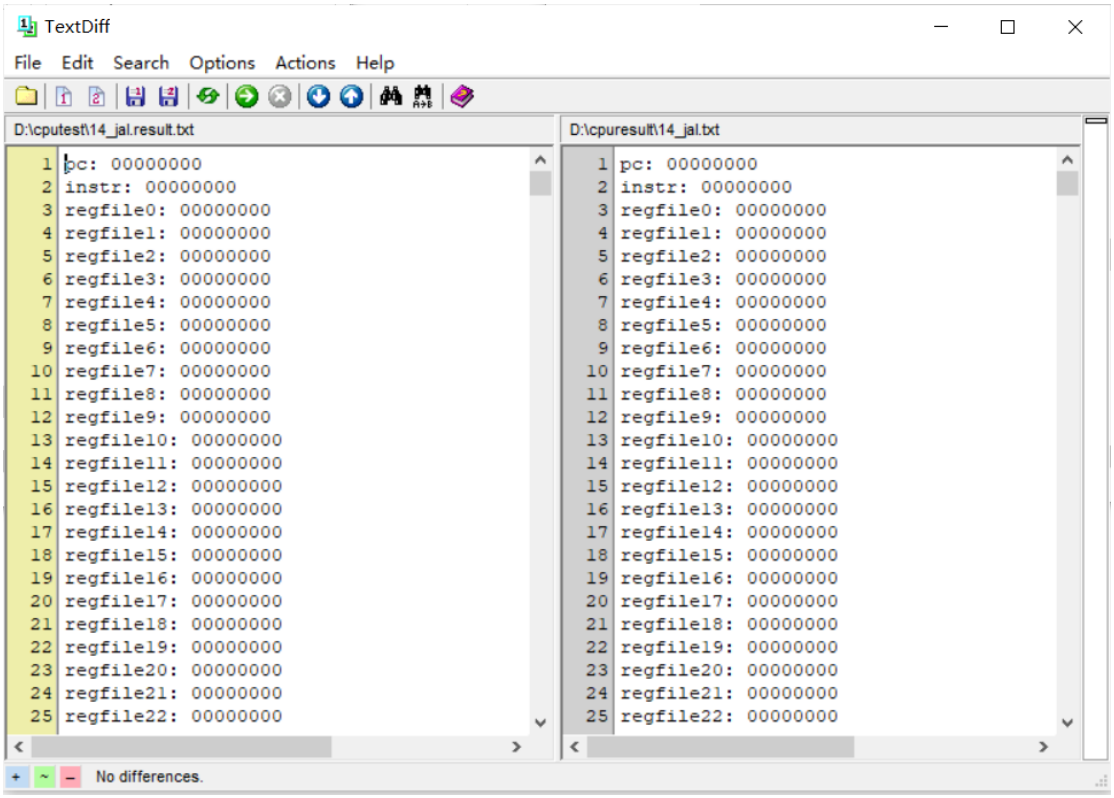
Bne：


TextDiff
File Edit Search Options Actions Help
D:\cputest\12_bne.result.txt
D:\cpuresult\12_bne.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

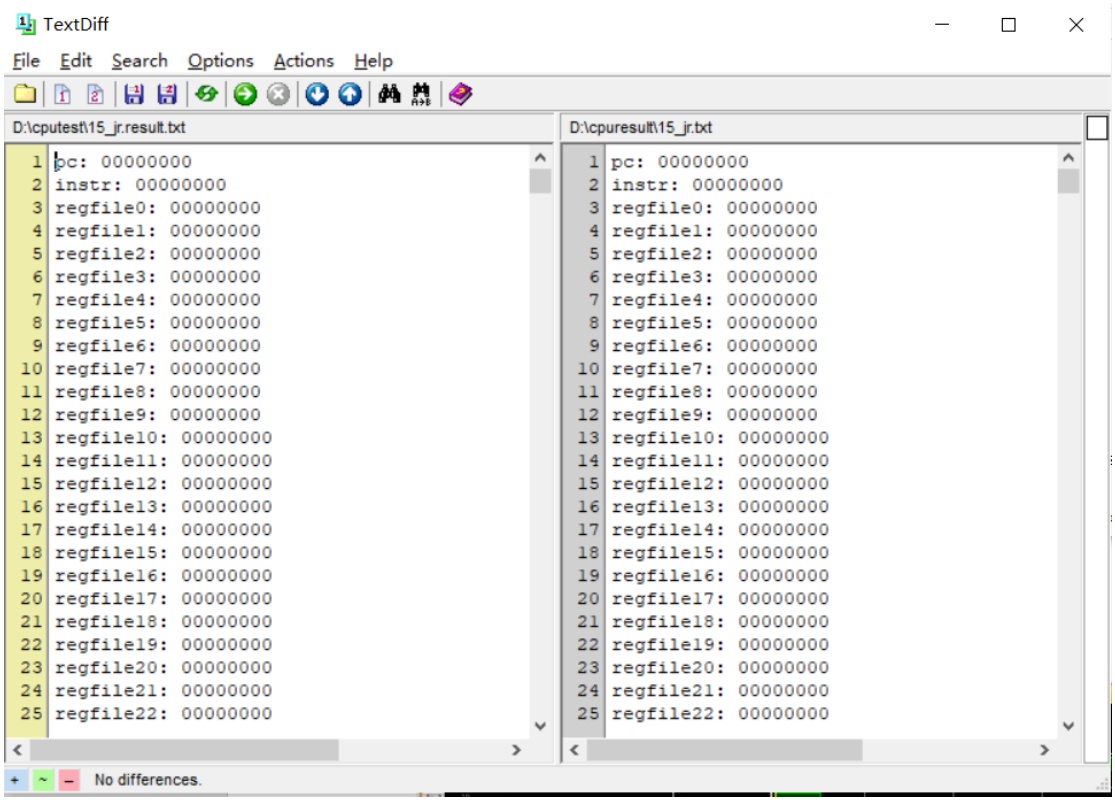
J：


TextDiff
File Edit Search Options Actions Help
D:\cputest\13_j.result.txt
D:\cpuresult\13_j.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

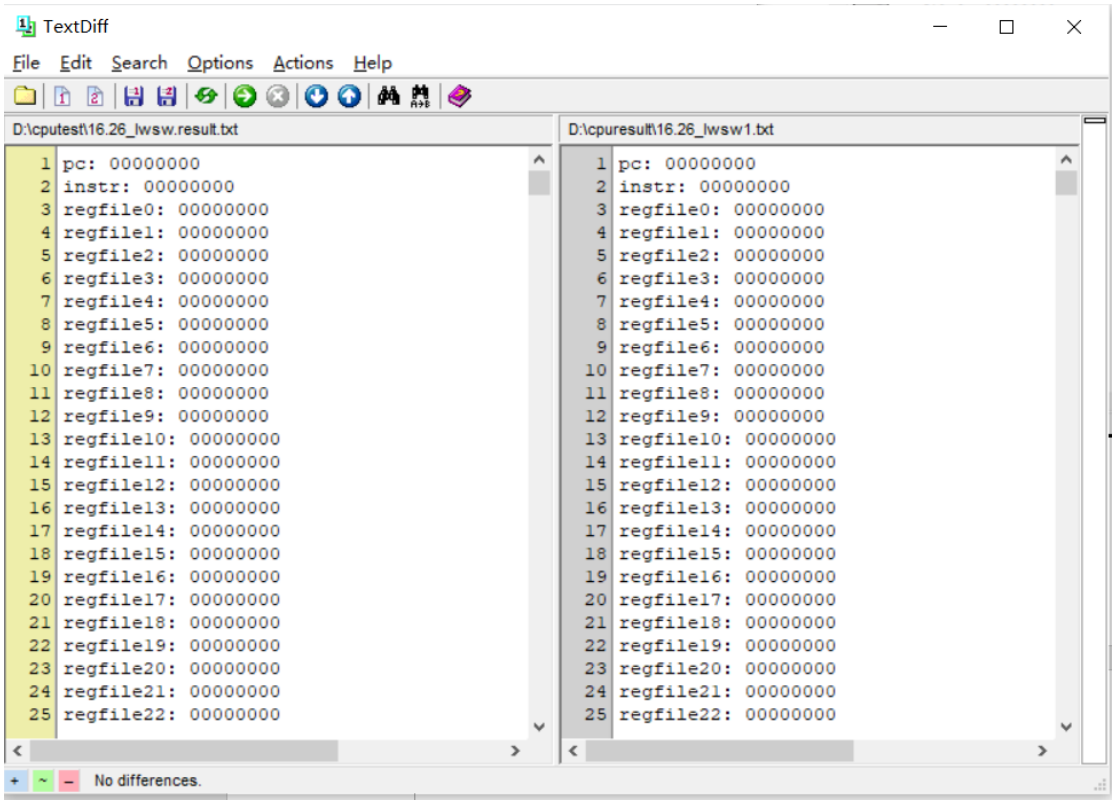
Jal:


TextDiff
File Edit Search Options Actions Help
D:\cputest\14_jal.result.txt
D:\cpuresult\14_jal.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

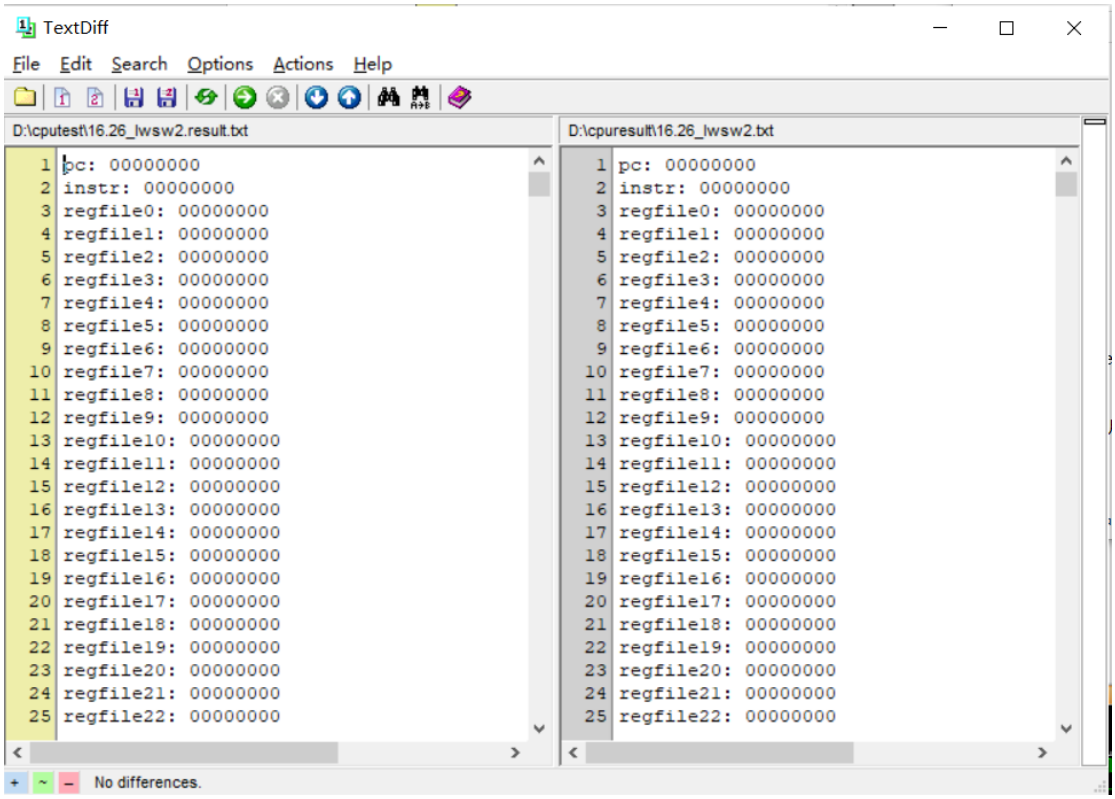
Jr:


TextDiff
File Edit Search Options Actions Help
D:\cputest\15_jr.result.txt
D:\cpuresult\15_jr.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

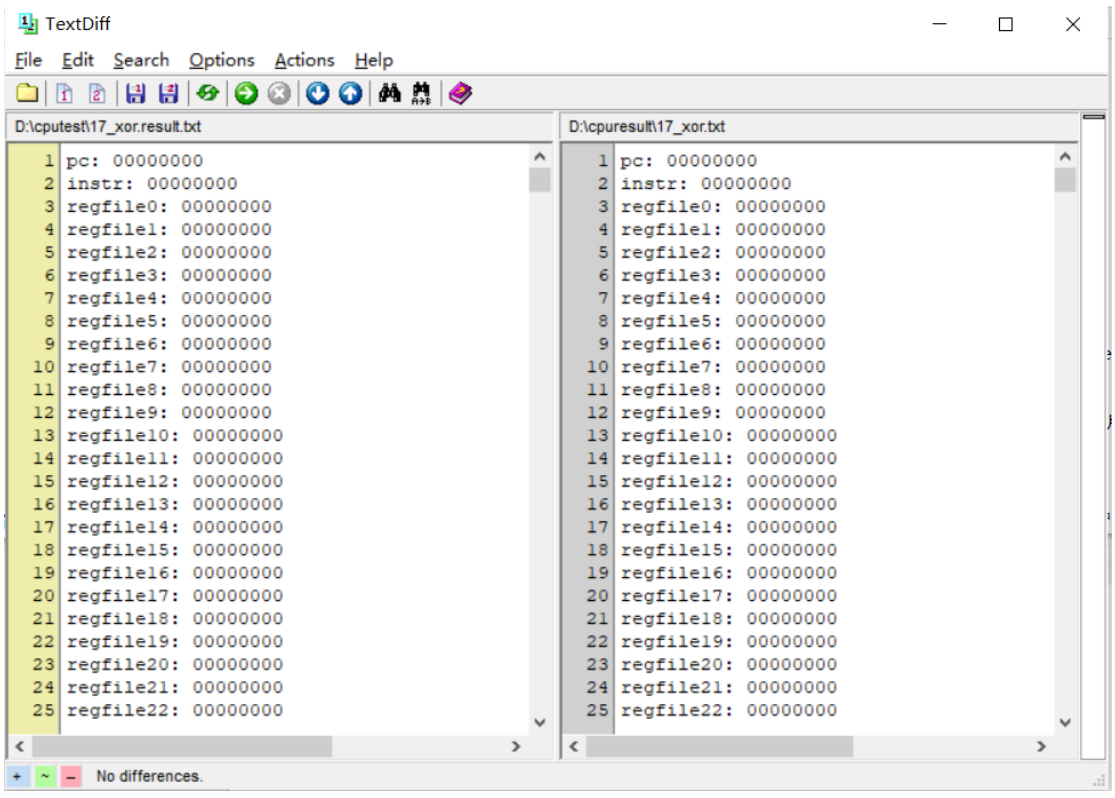
Lwsw1：


TextDiff
File Edit Search Options Actions Help
D:\cputest\16.26_lwsw.result.txt
D:\cpuresult\16.26_lwsw1.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

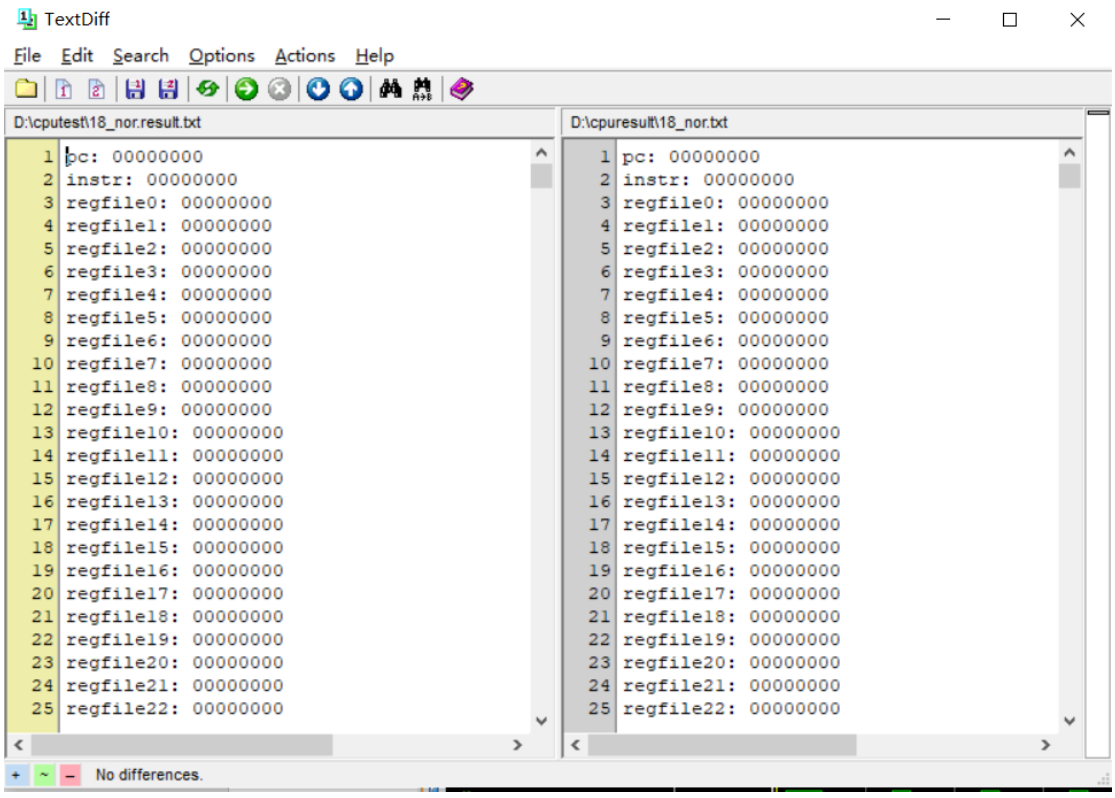
Lwsw2：


TextDiff
File Edit Search Options Actions Help
D:\cputest\16.26_lwsw2.result.txt
D:\cpuresult\16.26_lwsw2.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

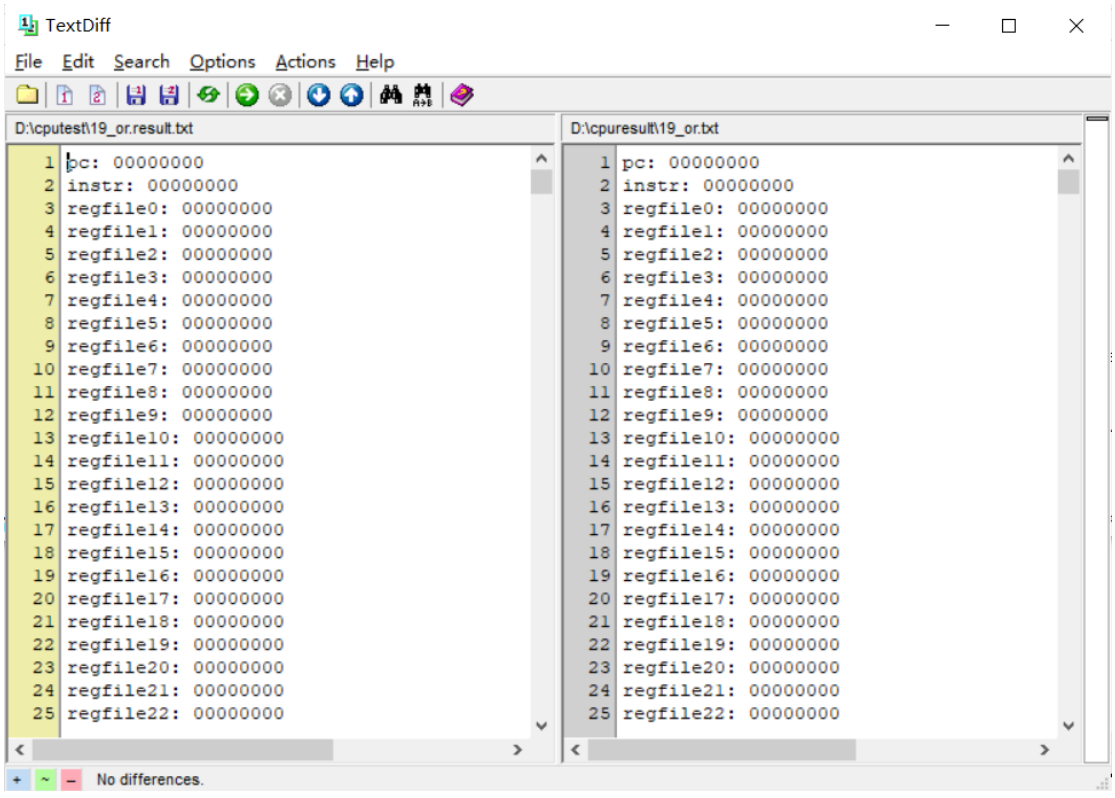
Xor：


TextDiff
File Edit Search Options Actions Help
D:\cputest\17_xor.result.txt
D:\cpuresult\17_xor.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

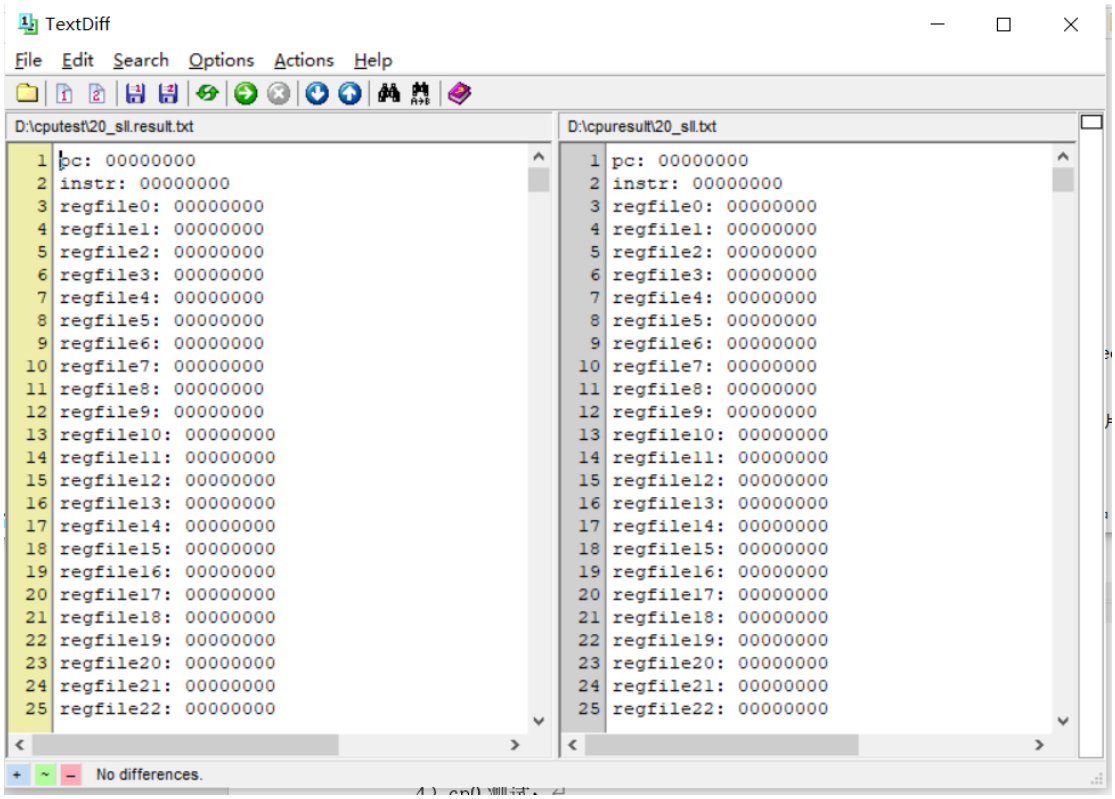
Nor：


TextDiff
File Edit Search Options Actions Help
D:\cputest\18_nor.result.txt
D:\cpuresult\18_nor.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

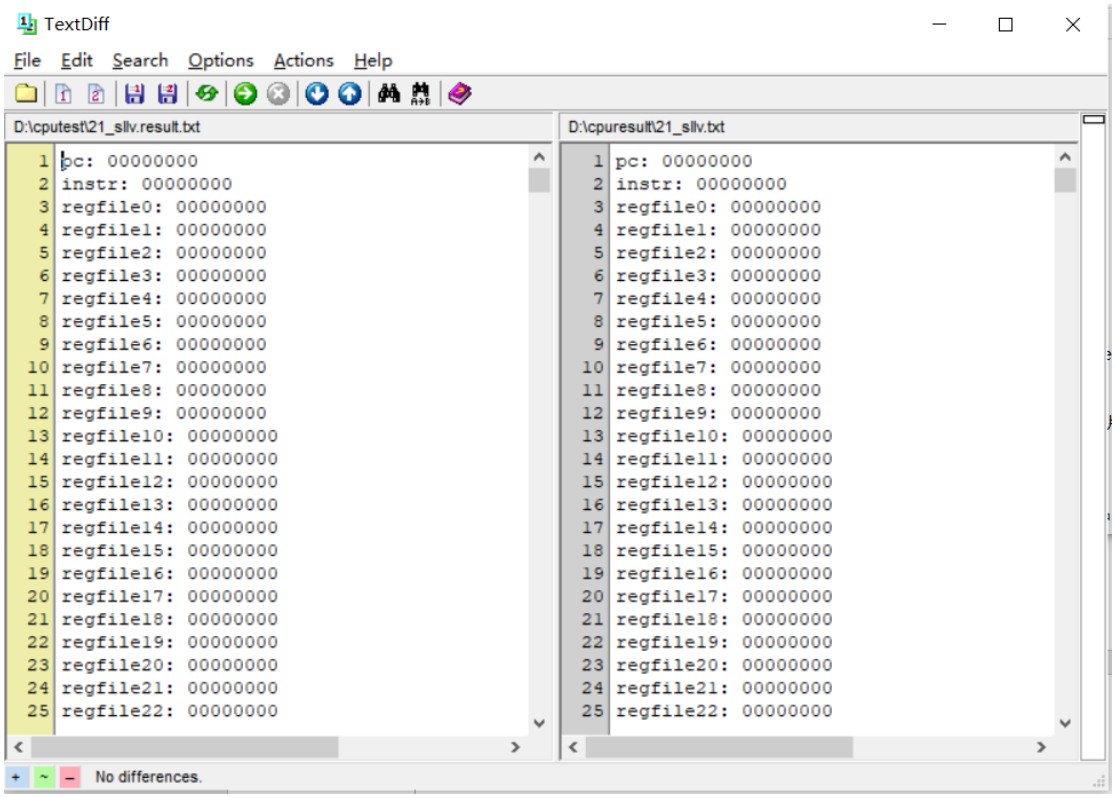
Or:


TextDiff
File Edit Search Options Actions Help
D:\cputest\19_or.result.txt
D:\cpuresult\19_or.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

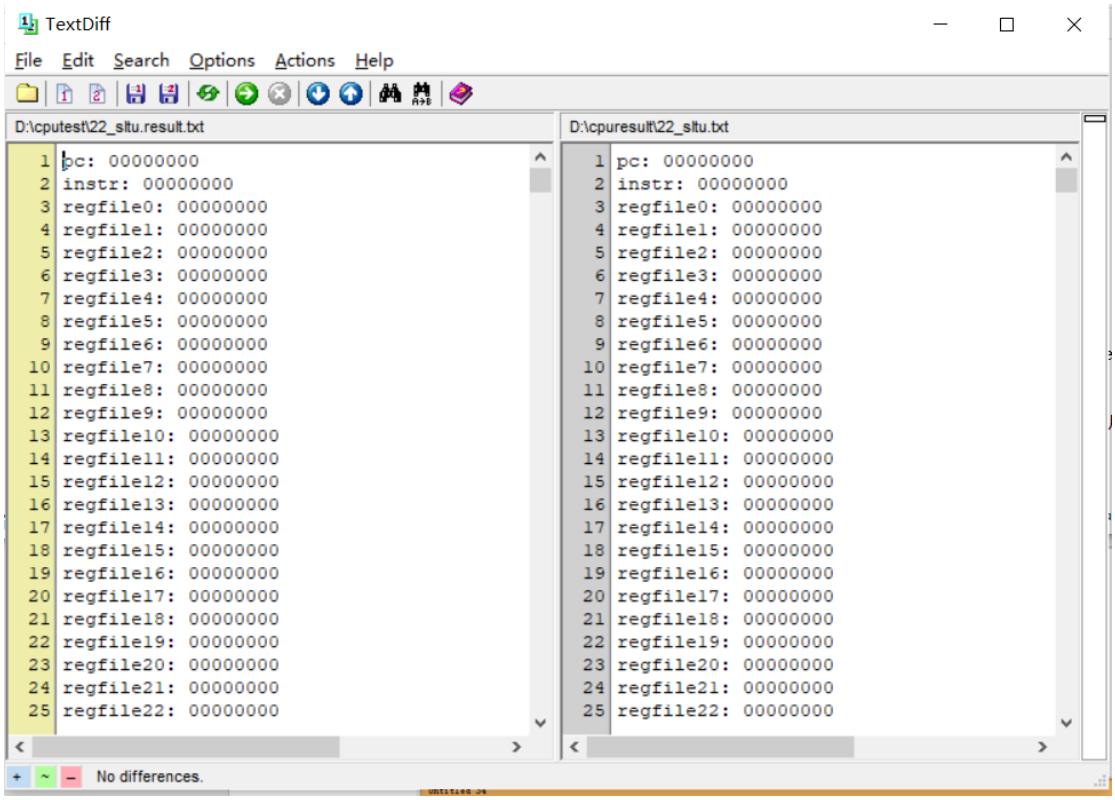
Sll:


TextDiff
File Edit Search Options Actions Help
D:\cputest\20_sll.result.txt
D:\cpuresult\20_sll.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

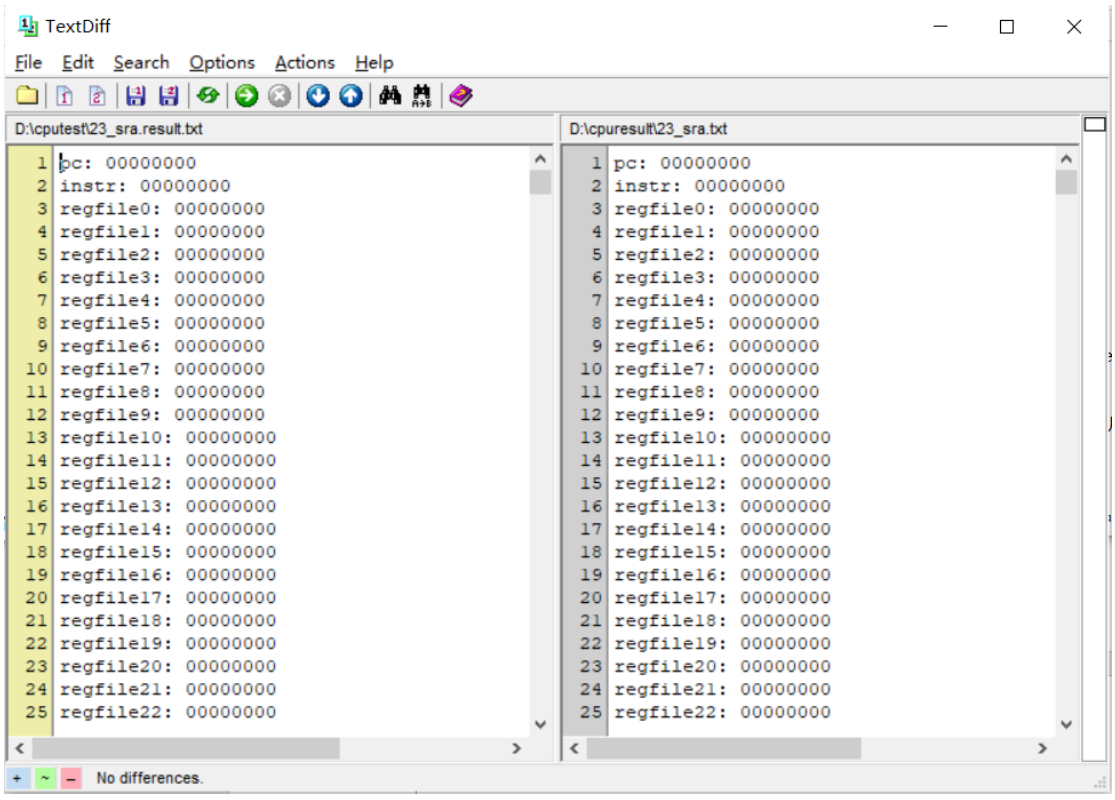
Sllv:


TextDiff
File Edit Search Options Actions Help
D:\cputest\21_sllv.result.txt
D:\cpuresult\21_sllv.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

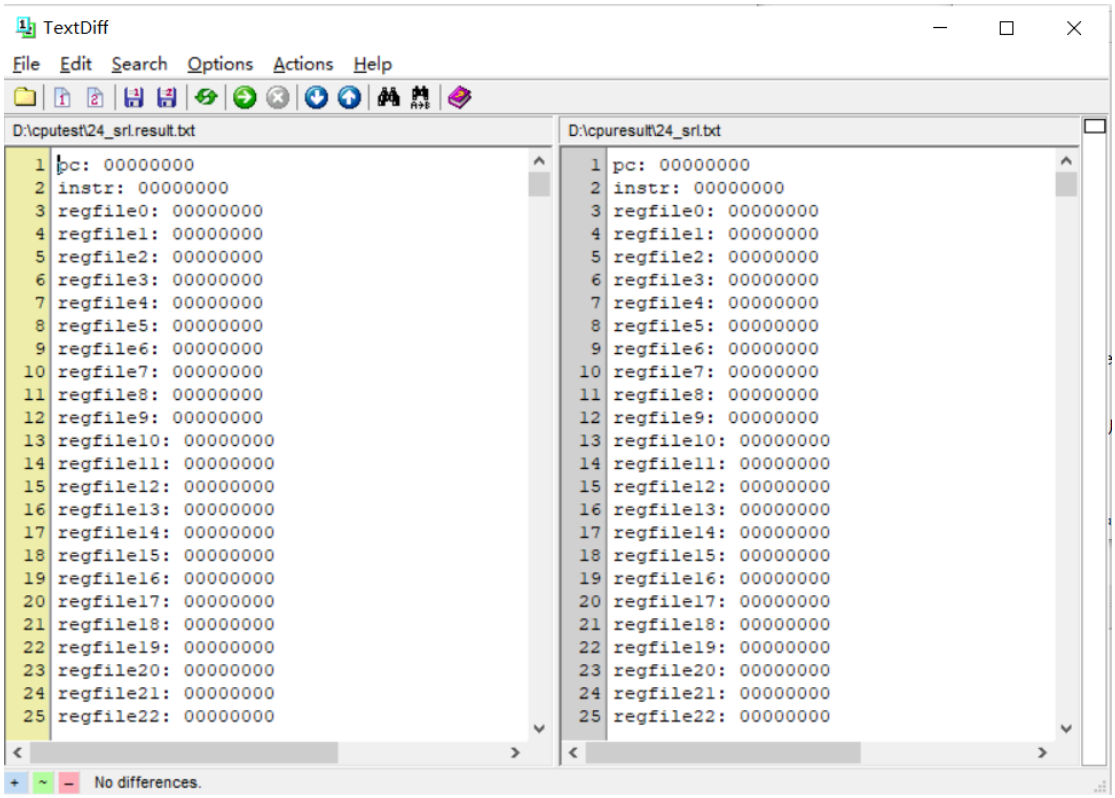
Sltu:


TextDiff
File Edit Search Options Actions Help
D:\cputest\22_sltu.result.txt
D:\cpuresult\22_sltu.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

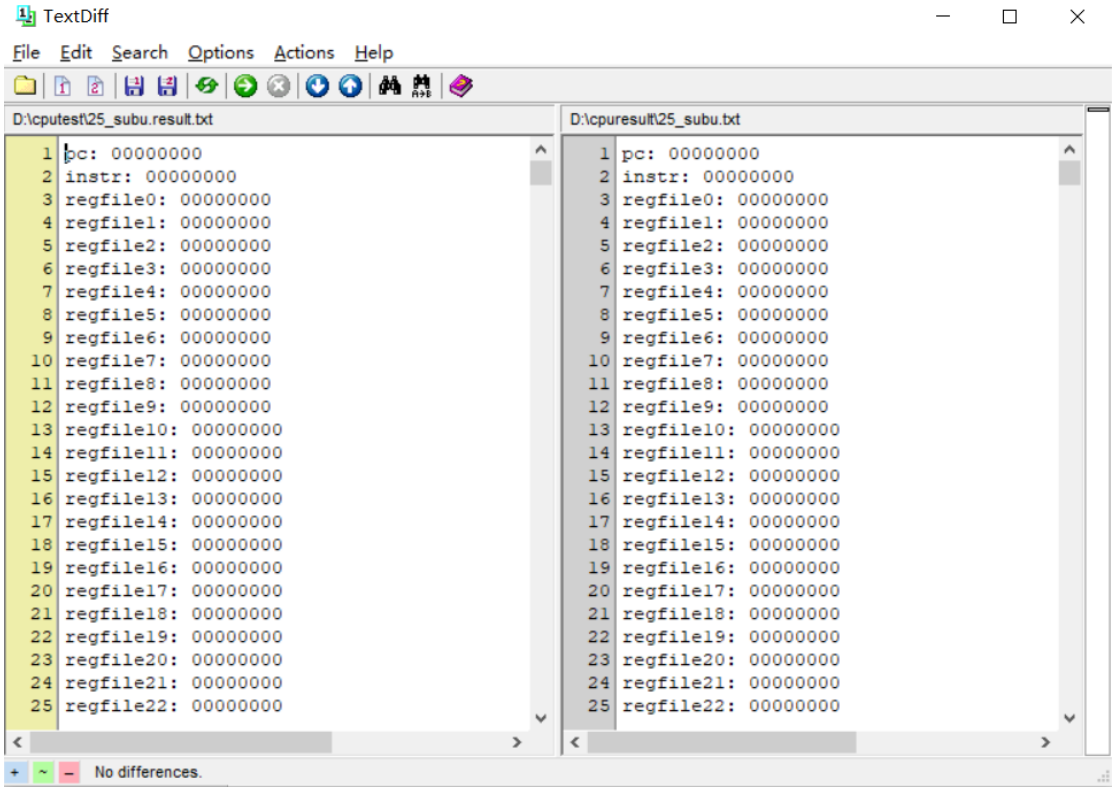
Sra:


TextDiff
File Edit Search Options Actions Help
D:\cputest\23_sra.result.txt
D:\cpuresult\23_sra.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

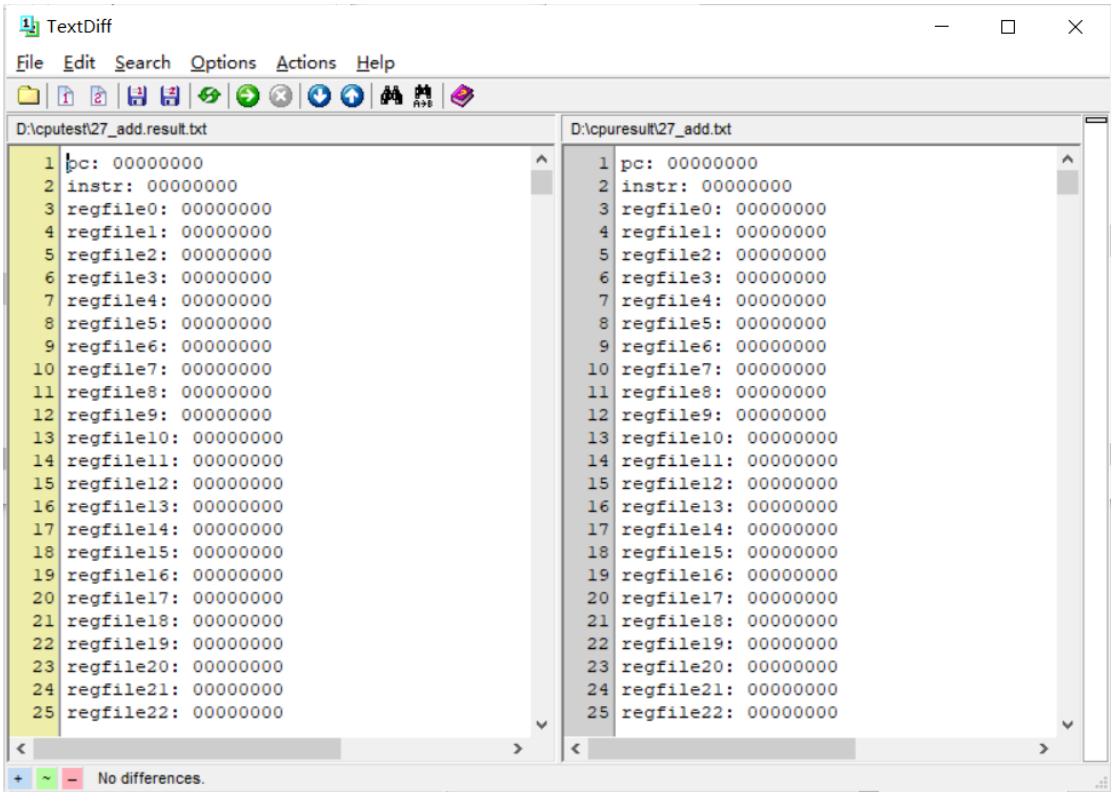
Srl:


TextDiff
File Edit Search Options Actions Help
D:\cputest\24_srl.result.txt
D:\cpuresult\24_srl.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

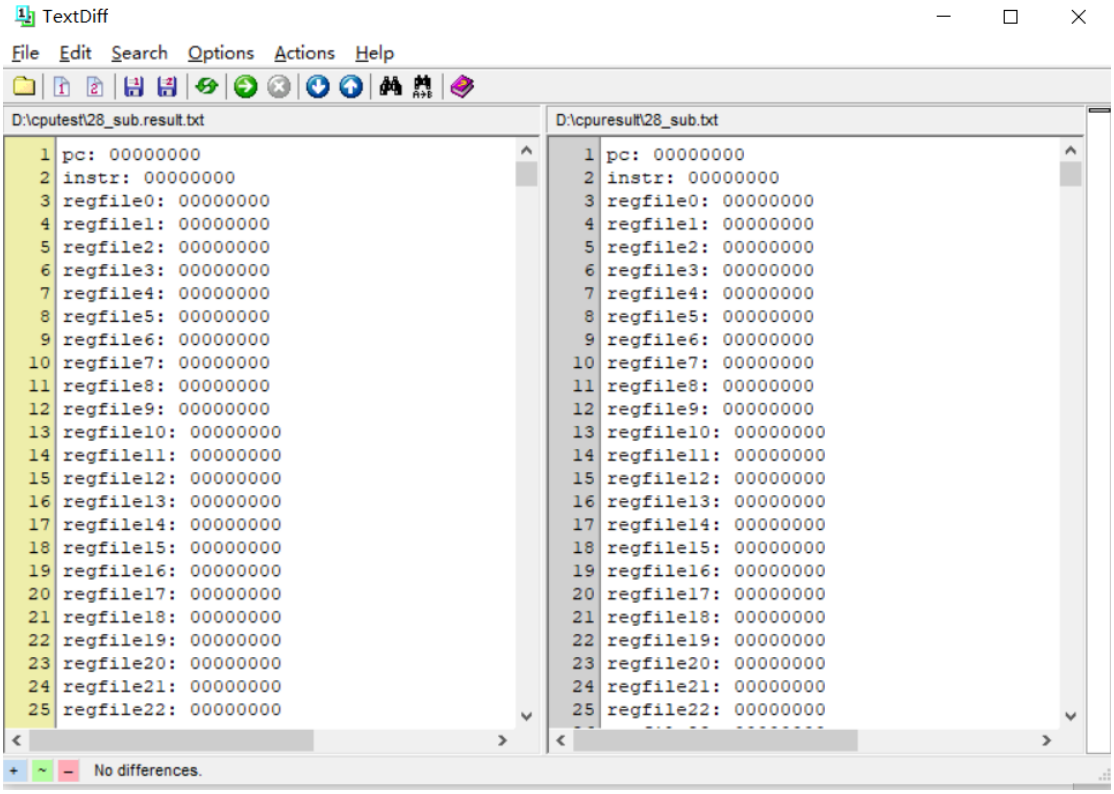
Subu:


TextDiff
File Edit Search Options Actions Help
D:\cputest\25_subu.result.txt
D:\cpuresult\25_subu.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

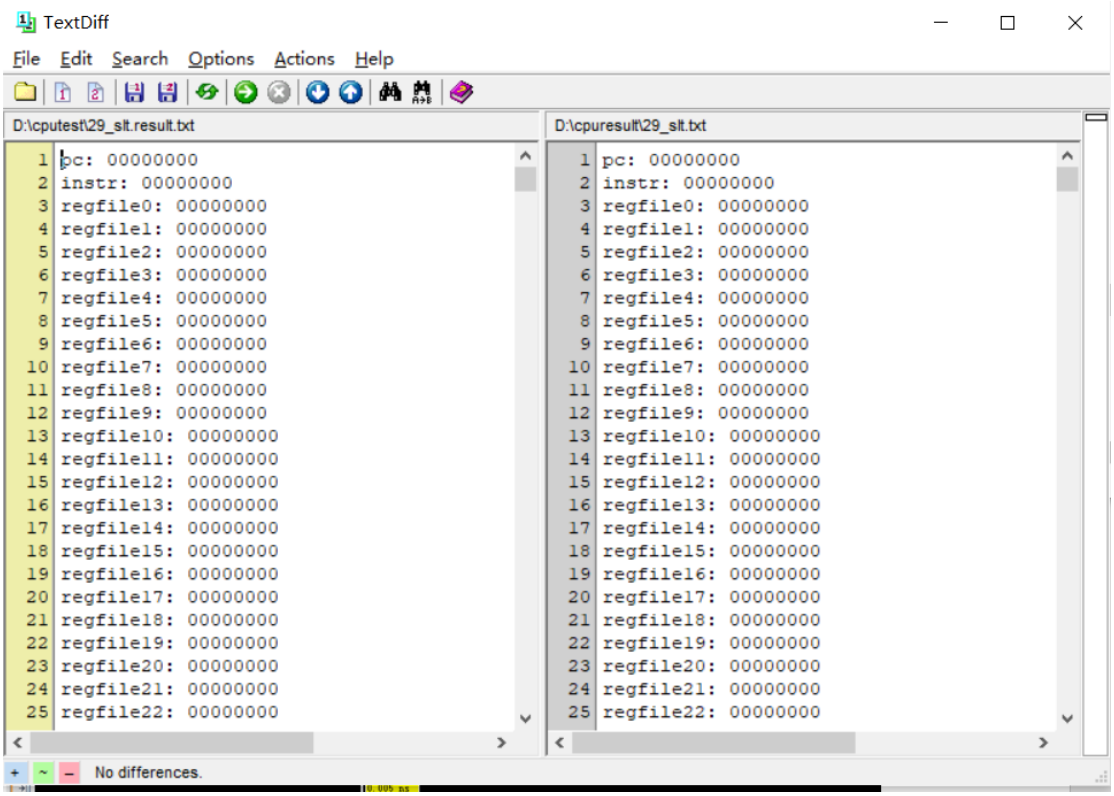
Add:


TextDiff
File Edit Search Options Actions Help
D:\cputest\27_add.result.txt
D:\cpuresult\27_add.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

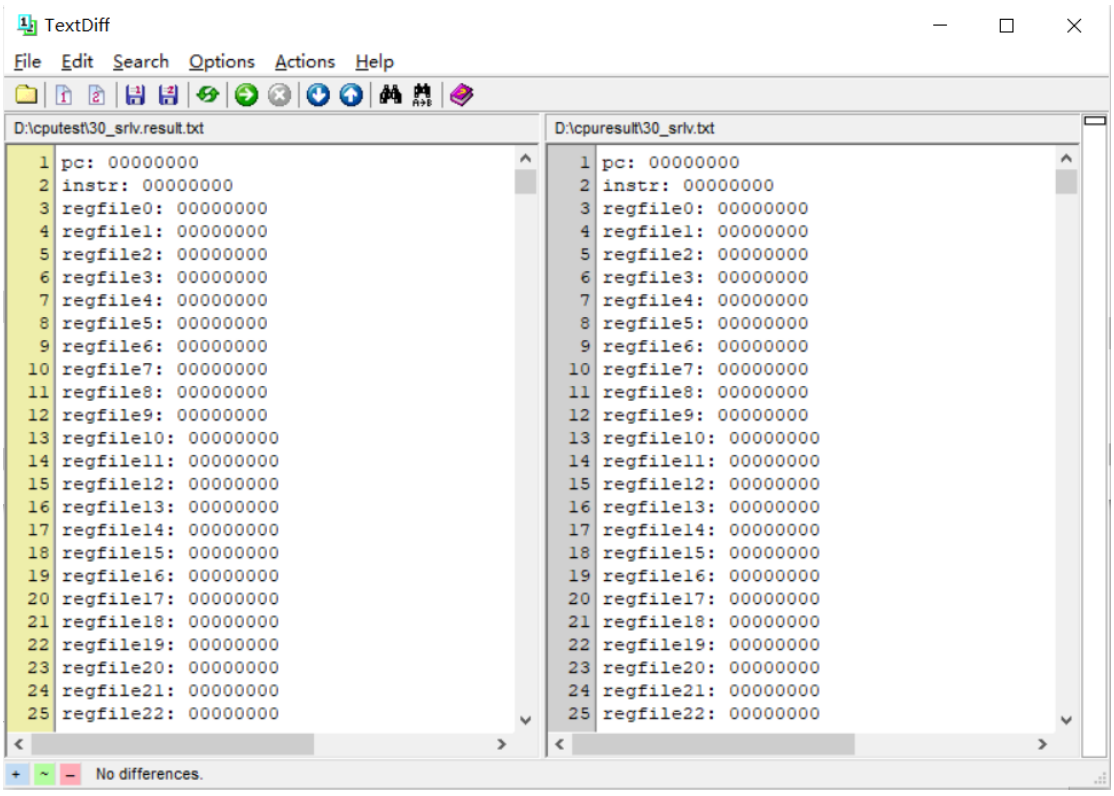
Sub:


TextDiff
File Edit Search Options Actions Help
D:\cputest\28_sub.result.txt
D:\cpuresult\28_sub.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

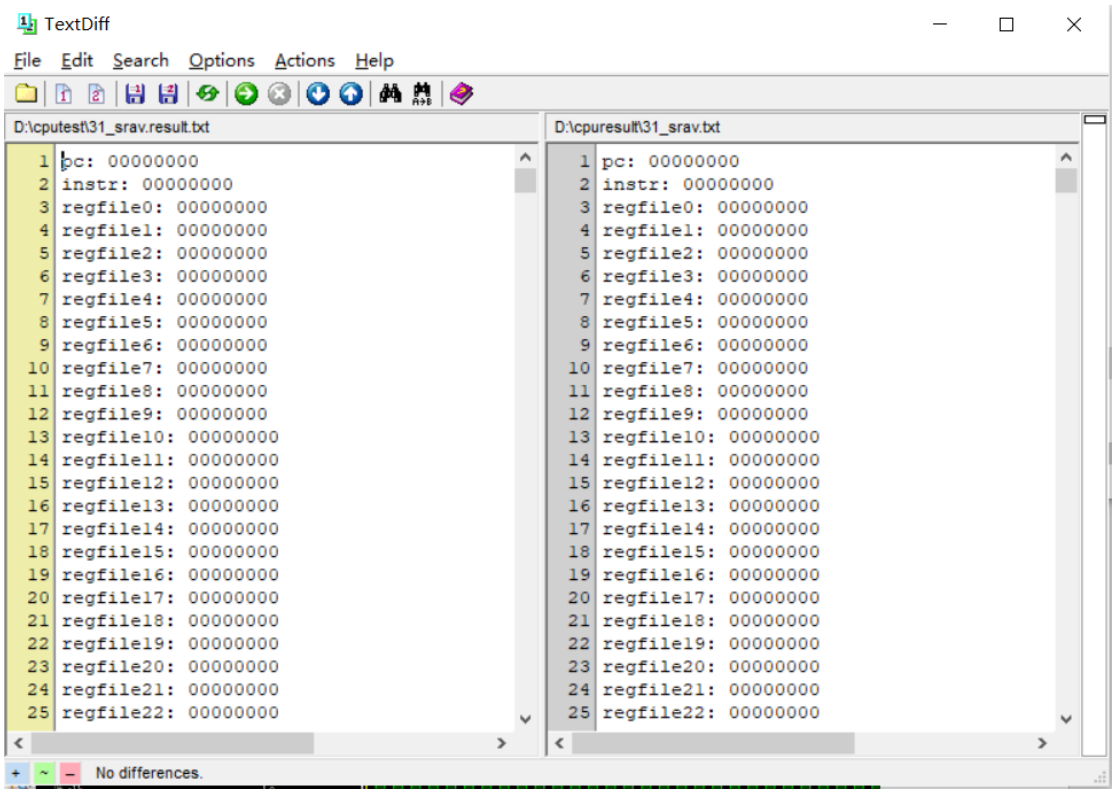
Slt:


TextDiff
File Edit Search Options Actions Help
D:\cputest\29_slt.result.txt
D:\cpuresult\29_slt.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

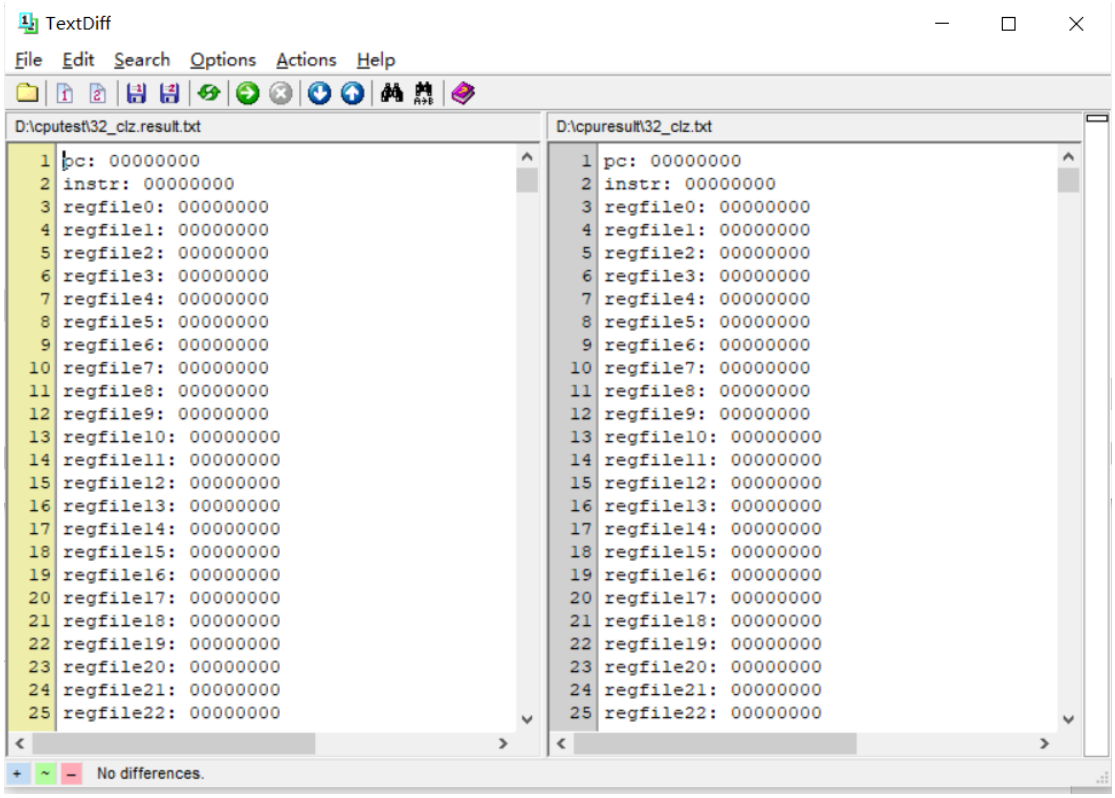
Srlv:


TextDiff
File Edit Search Options Actions Help
D:\cputest\30_srlv.result.txt
D:\cpuresult\30_srlv.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

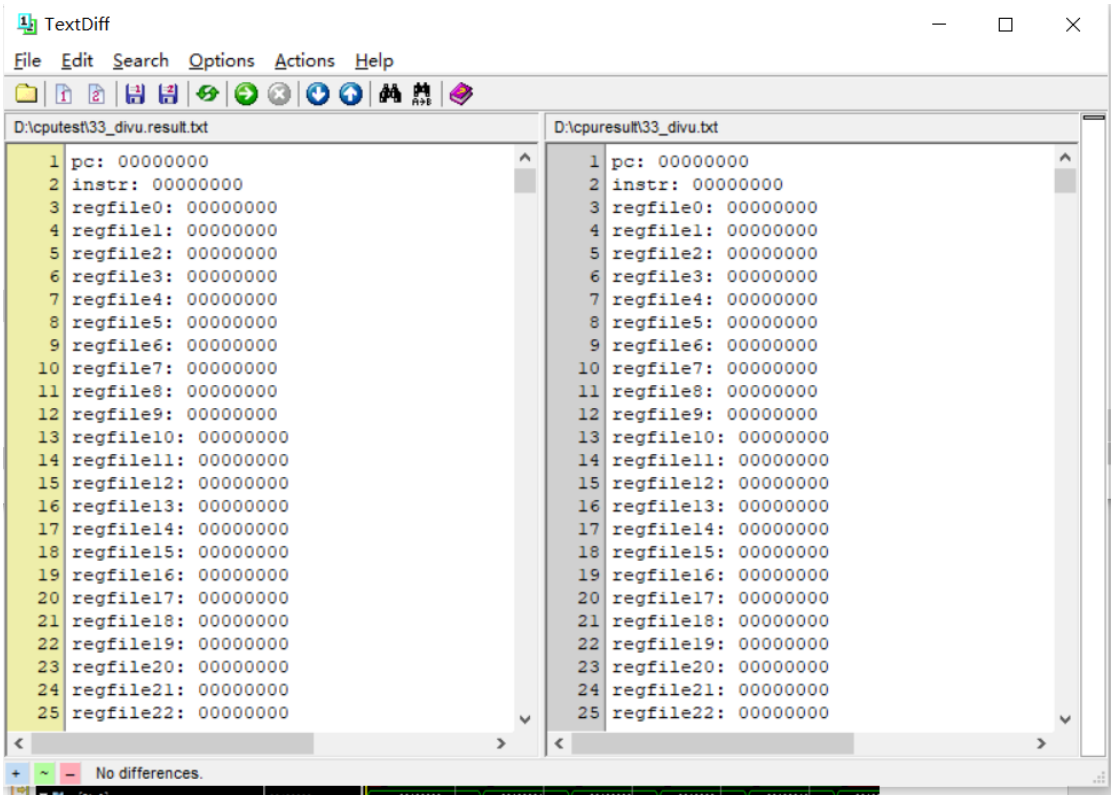
Srav:


TextDiff
File Edit Search Options Actions Help
D:\cputest\31_srav.result.txt
D:\cpuresult\31_srav.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

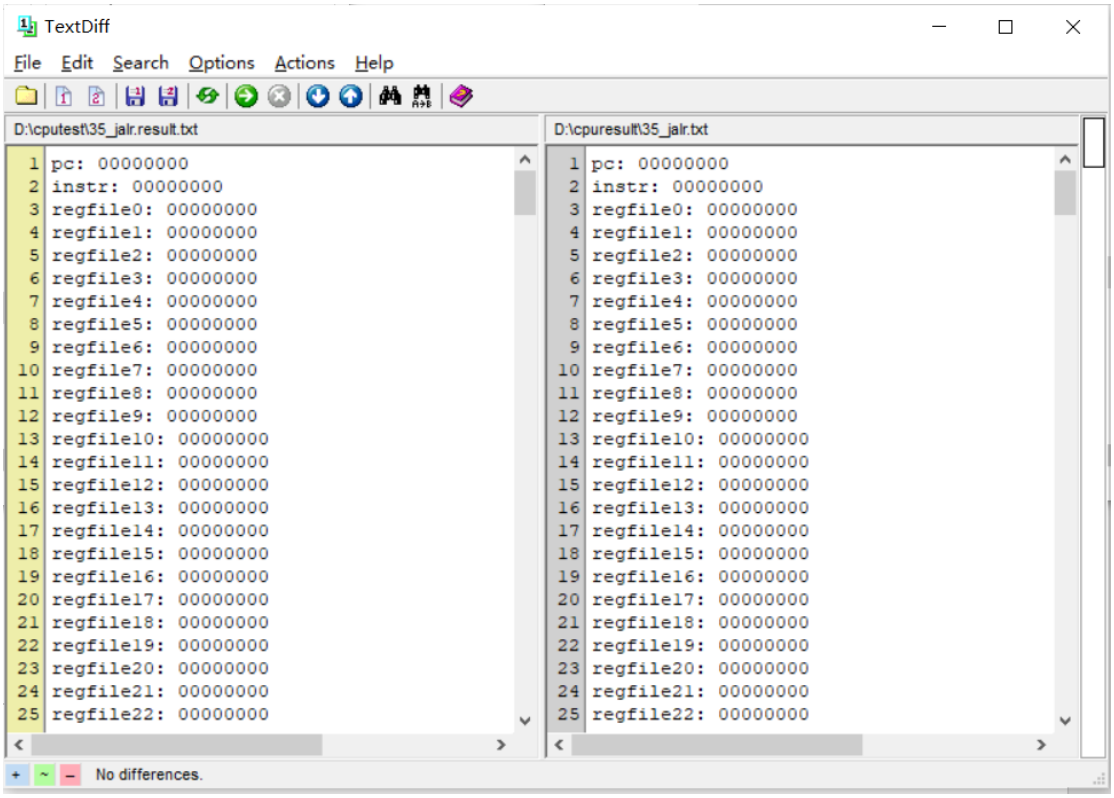
Clz:


TextDiff
File Edit Search Options Actions Help
D:\cputest\32_clz.result.txt
D:\cpuresult\32_clz.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

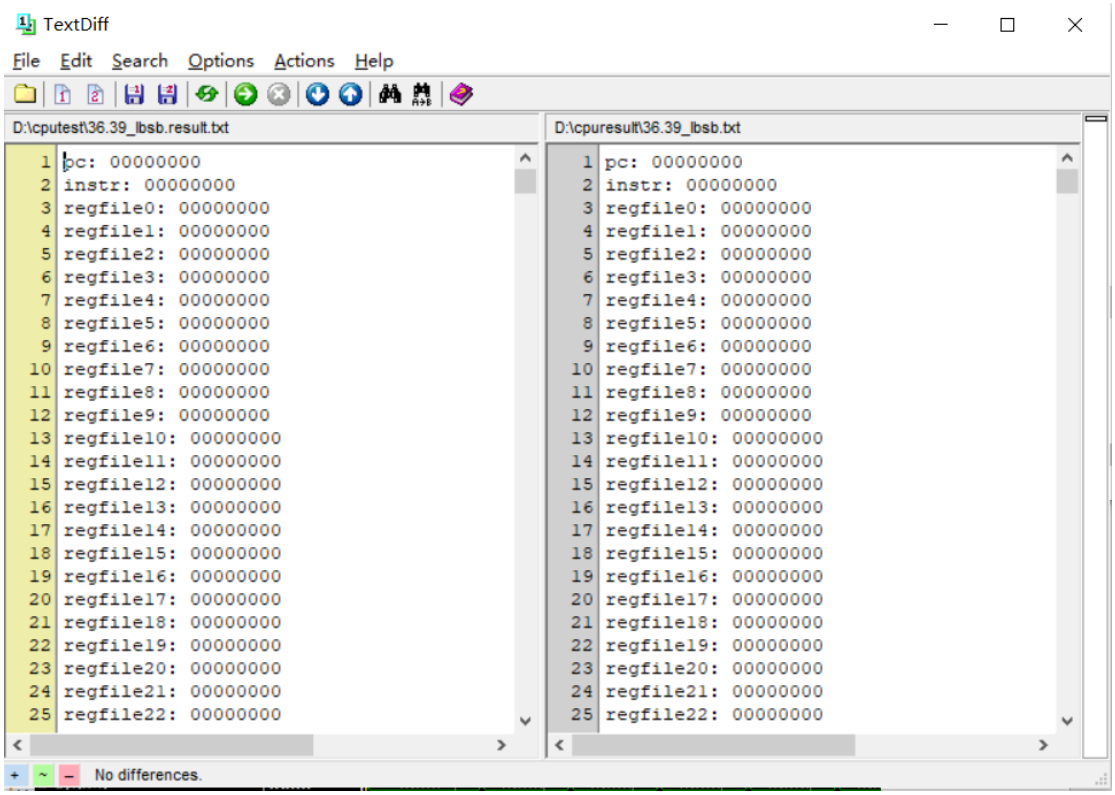
Divu:


TextDiff
File Edit Search Options Actions Help
D:\cputest\33_divu.result.txt
D:\cpuresult\33_divu.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

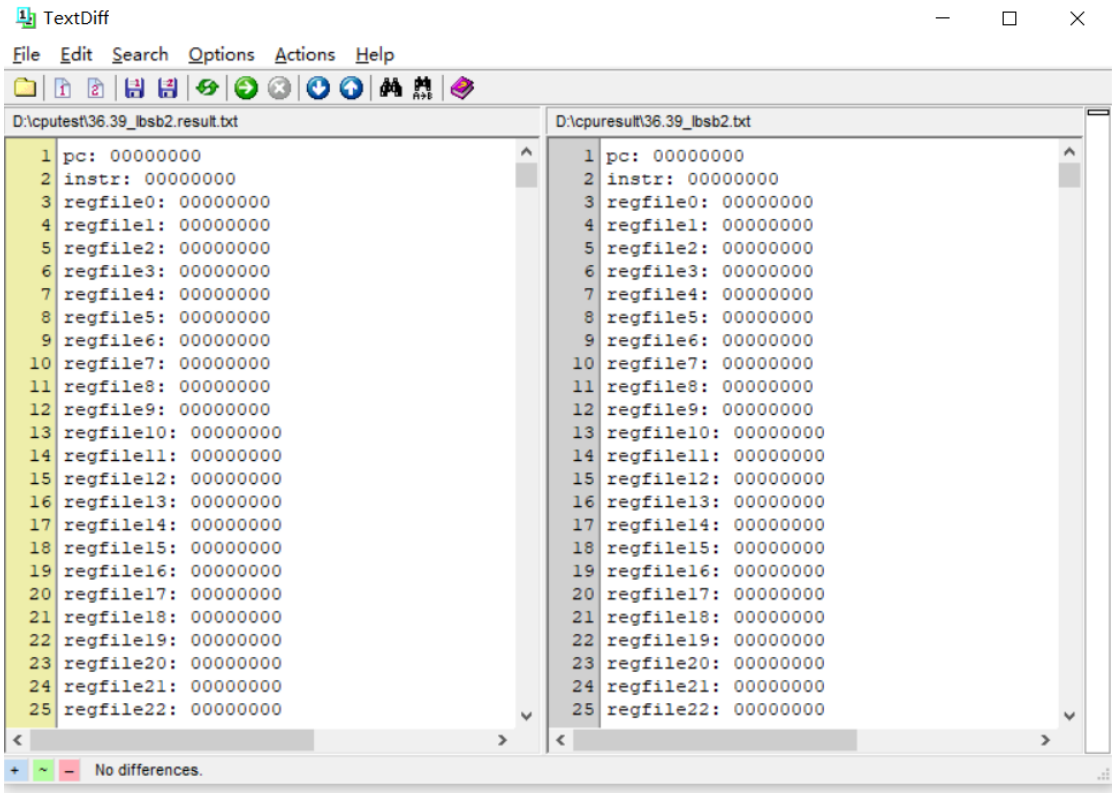
Jalr:


TextDiff
File Edit Search Options Actions Help
D:\cputest\35_jalr.result.txt
D:\cpuresult\35_jalr.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

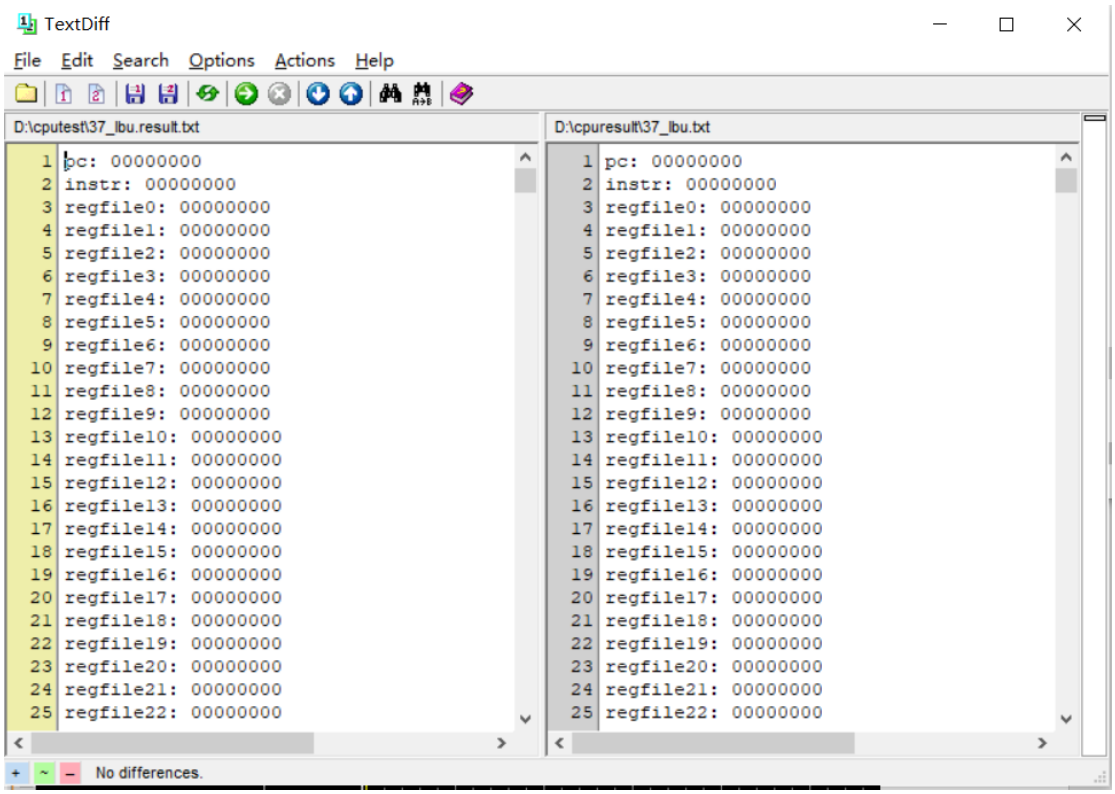
Lbsb:


TextDiff
File Edit Search Options Actions Help
D:\cputest\36.39_lbsb.result.txt
D:\cpuresult\36.39_lbsb.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

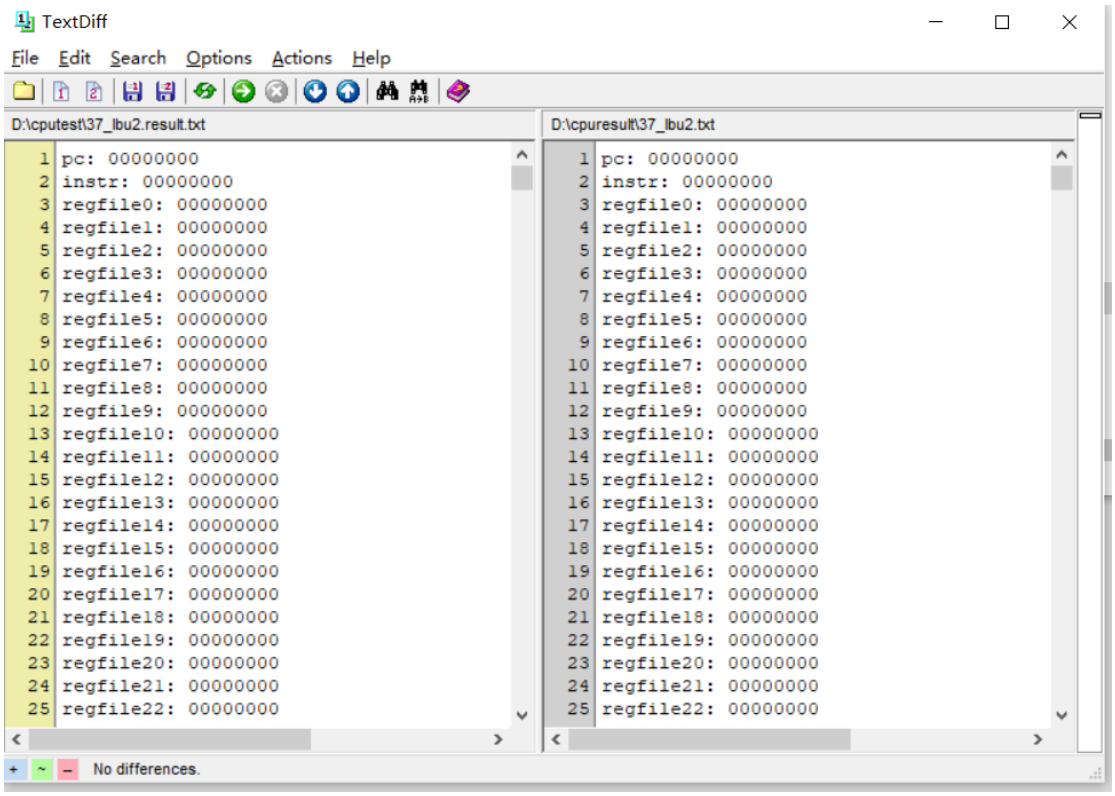
Lbsb2：


TextDiff
File Edit Search Options Actions Help
D:\cputest\36.39_lbsb2.result.txt
D:\cpuresult\36.39_lbsb2.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

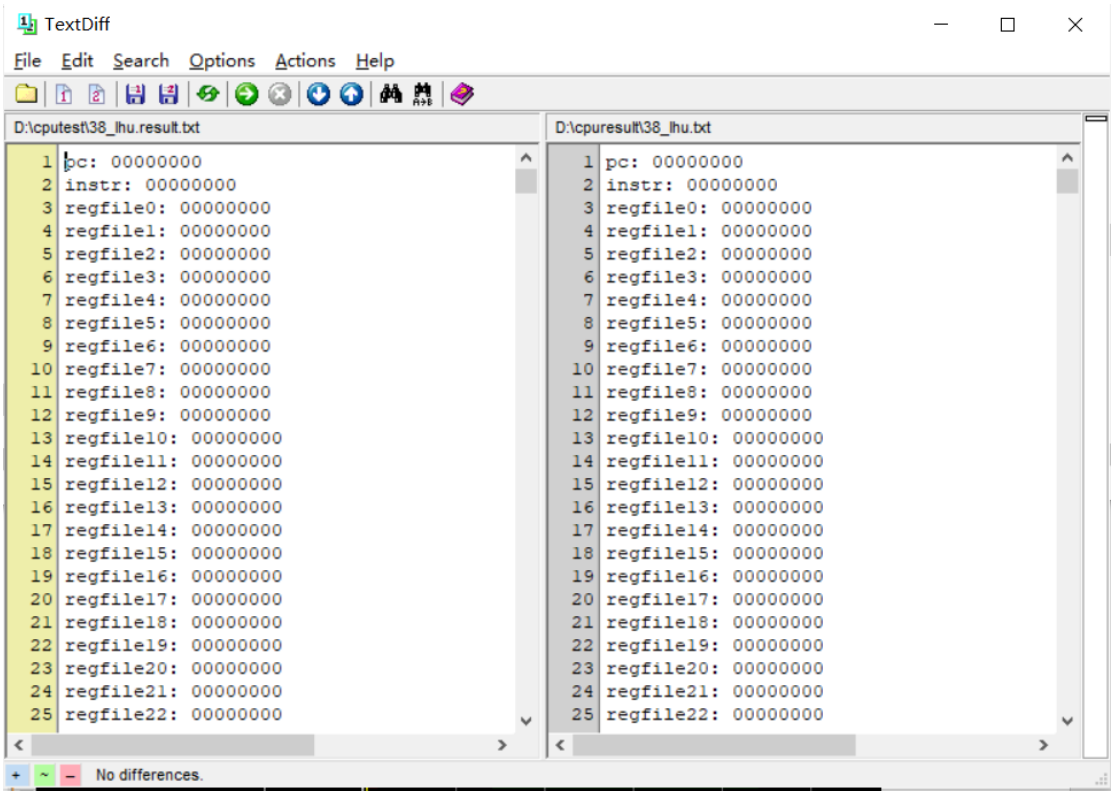
Lbu：


TextDiff
File Edit Search Options Actions Help
D:\cputest\37_lbu.result.txt
D:\cpuresult\37_lbu.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

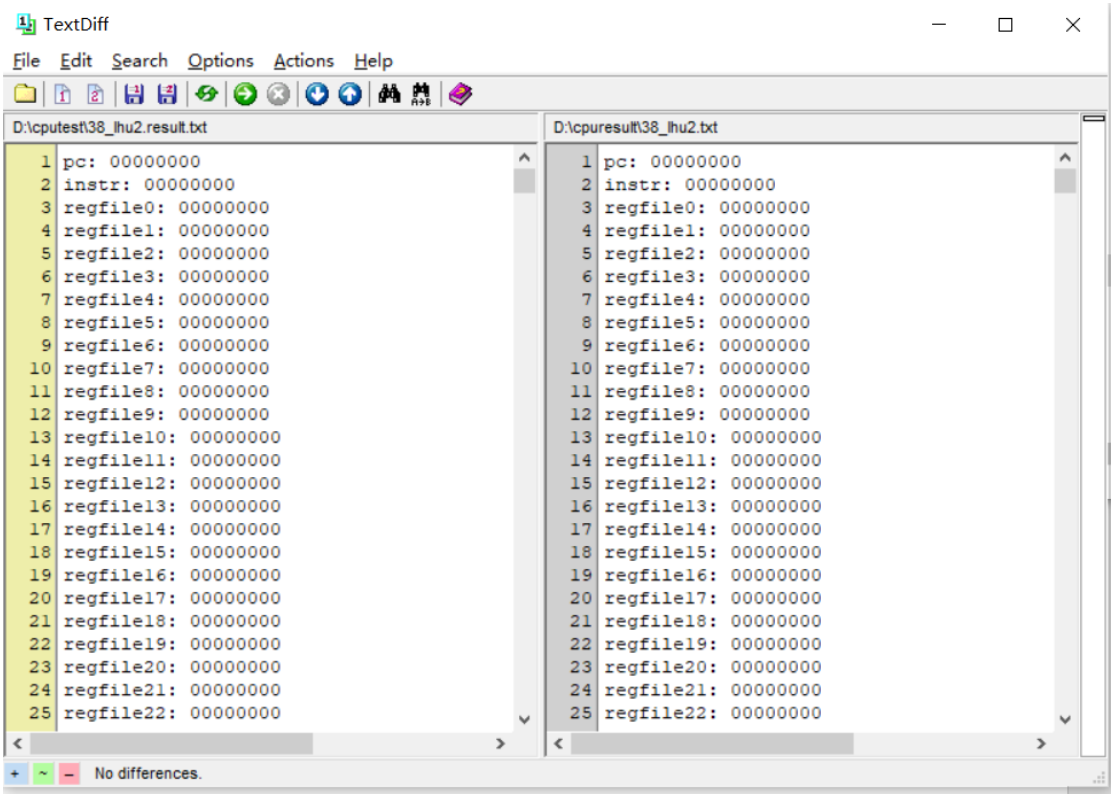
Lbu2：


TextDiff
File Edit Search Options Actions Help
D:\cputest\37_lbu2.result.txt
D:\cpuresult\37_lbu2.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

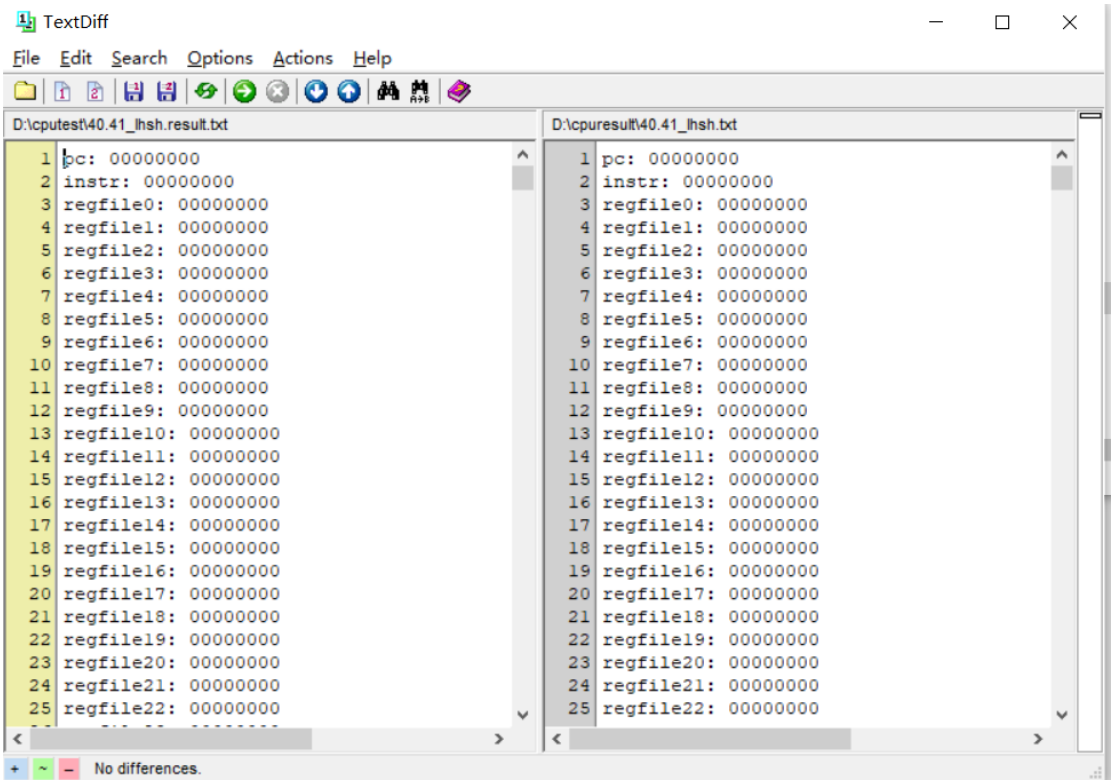
Lhu：


TextDiff
File Edit Search Options Actions Help
D:\cputest\38_lhu.result.txt
D:\cpuresult\38_lhu.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

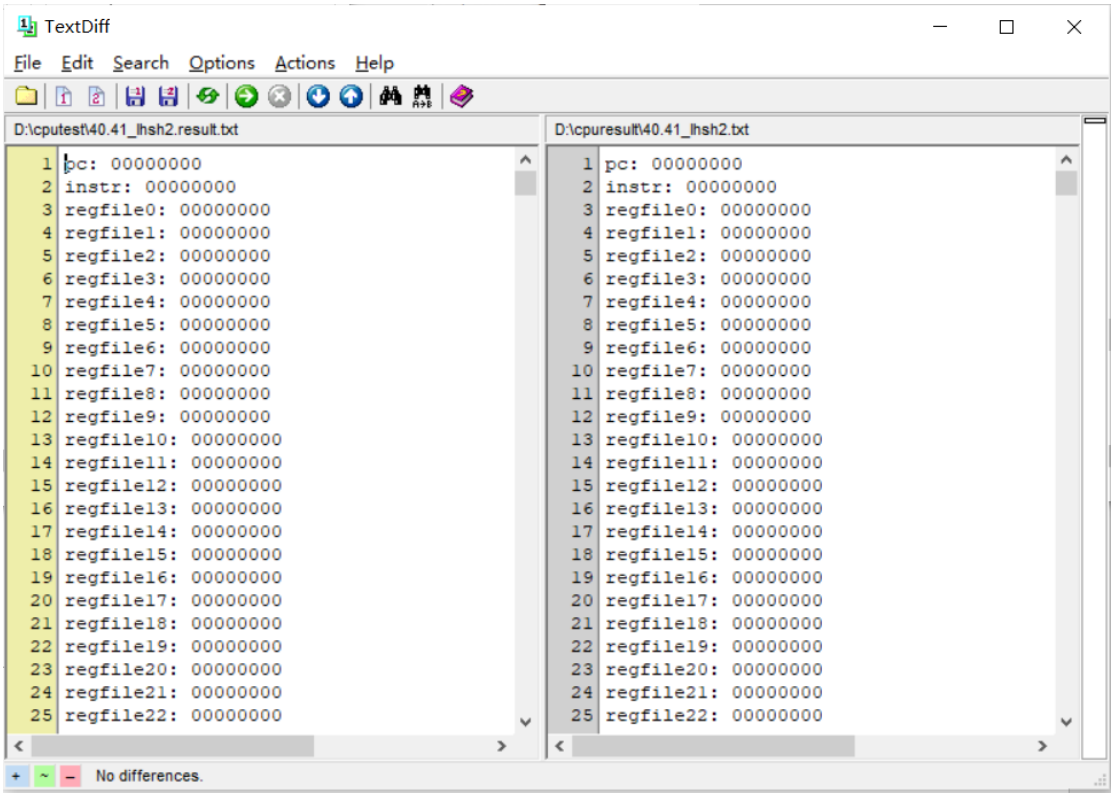
Lhu2：


TextDiff
File Edit Search Options Actions Help
D:\cputest\38_lhu2.result.txt
D:\cpuresult\38_lhu2.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

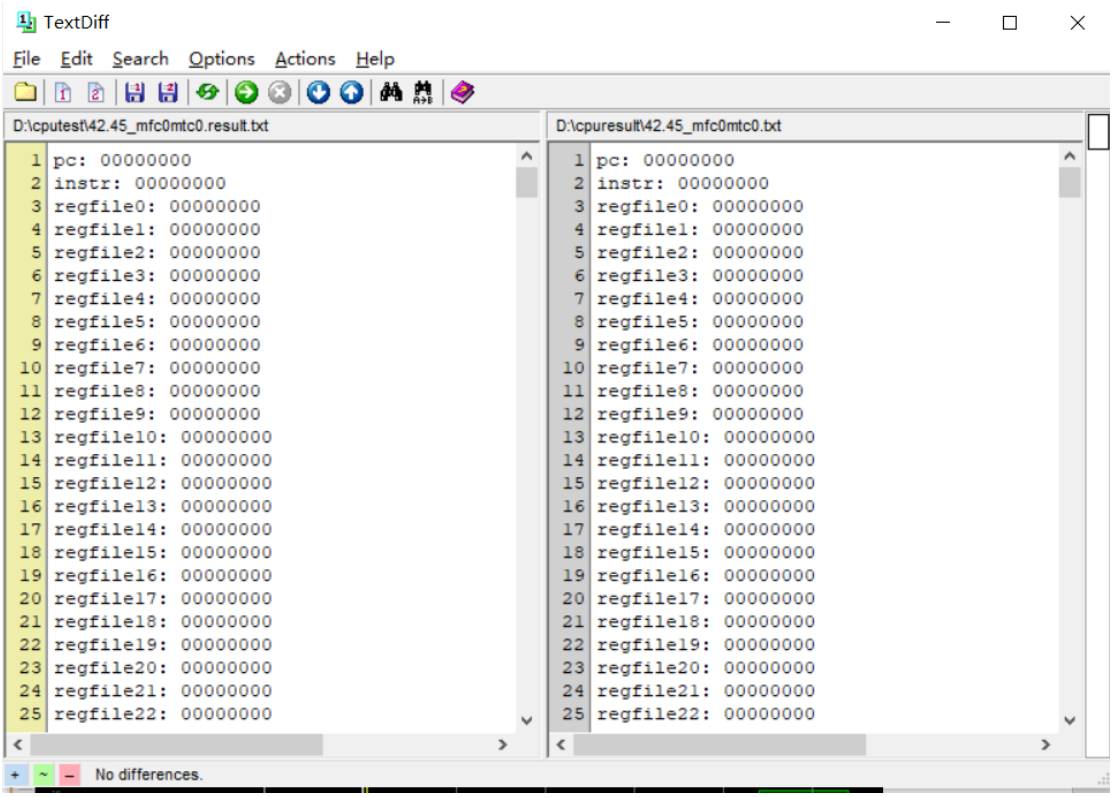
Lhsh：



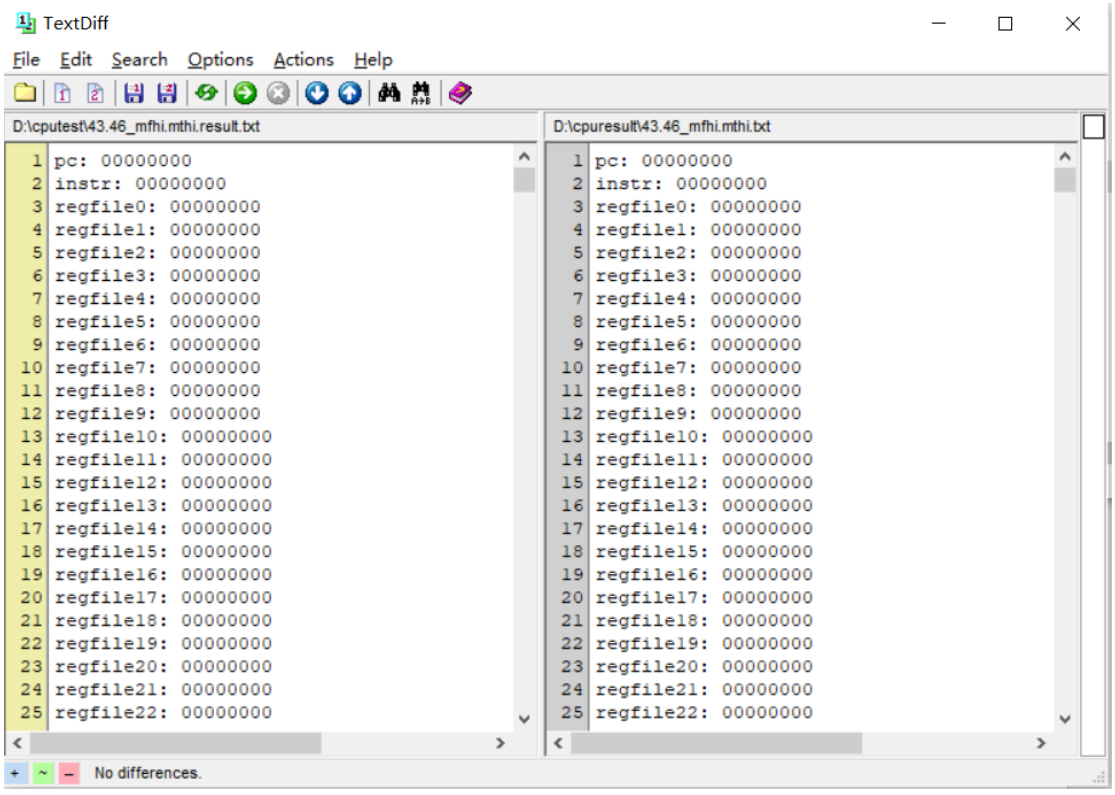
Lhsh2:



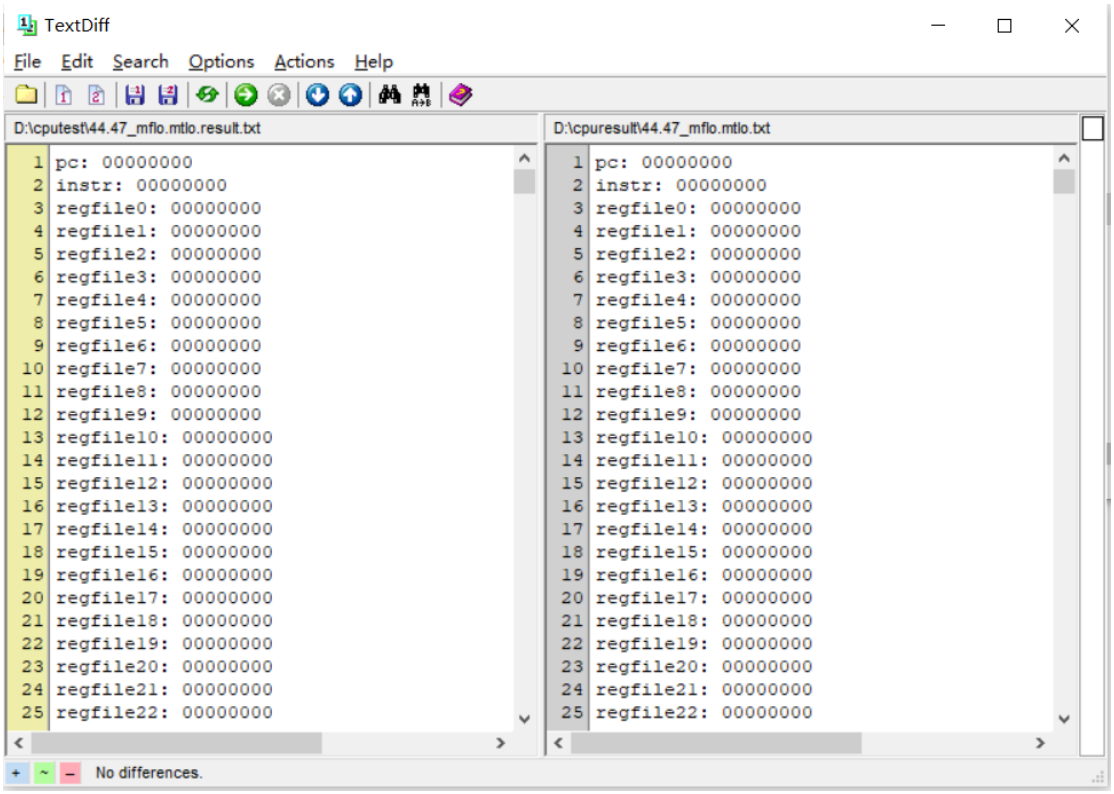
MfcOmtcO:



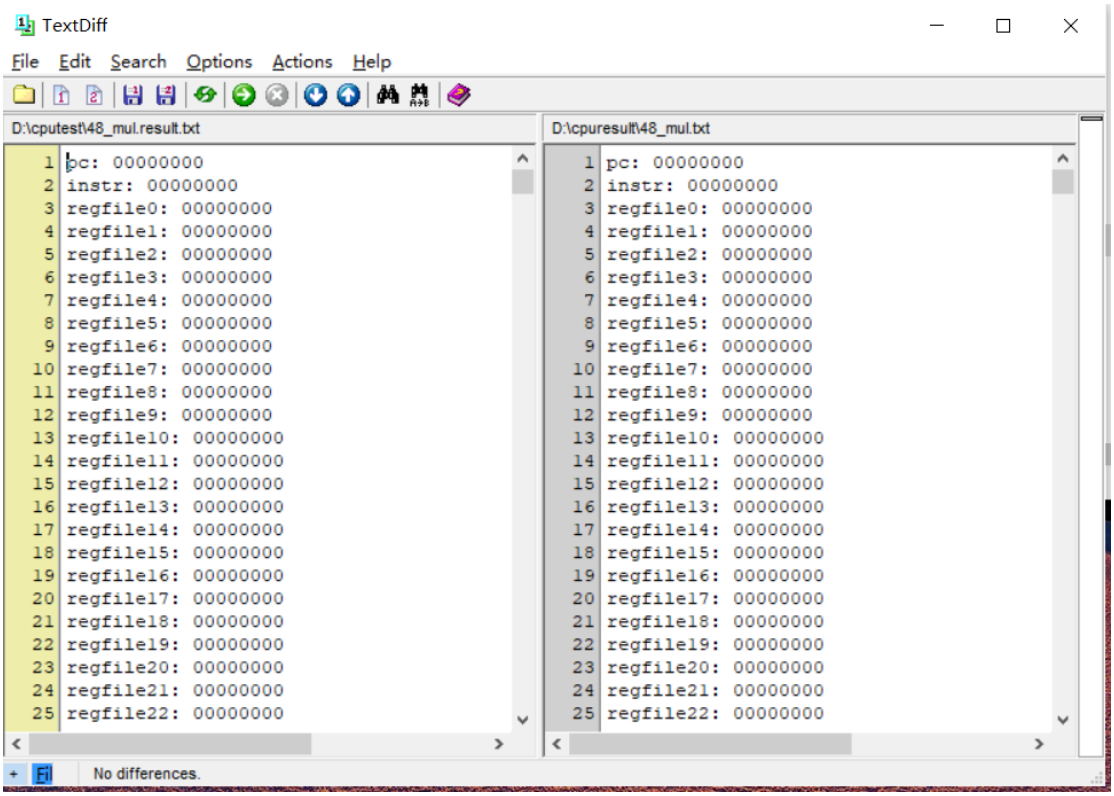
Mfhimthi:



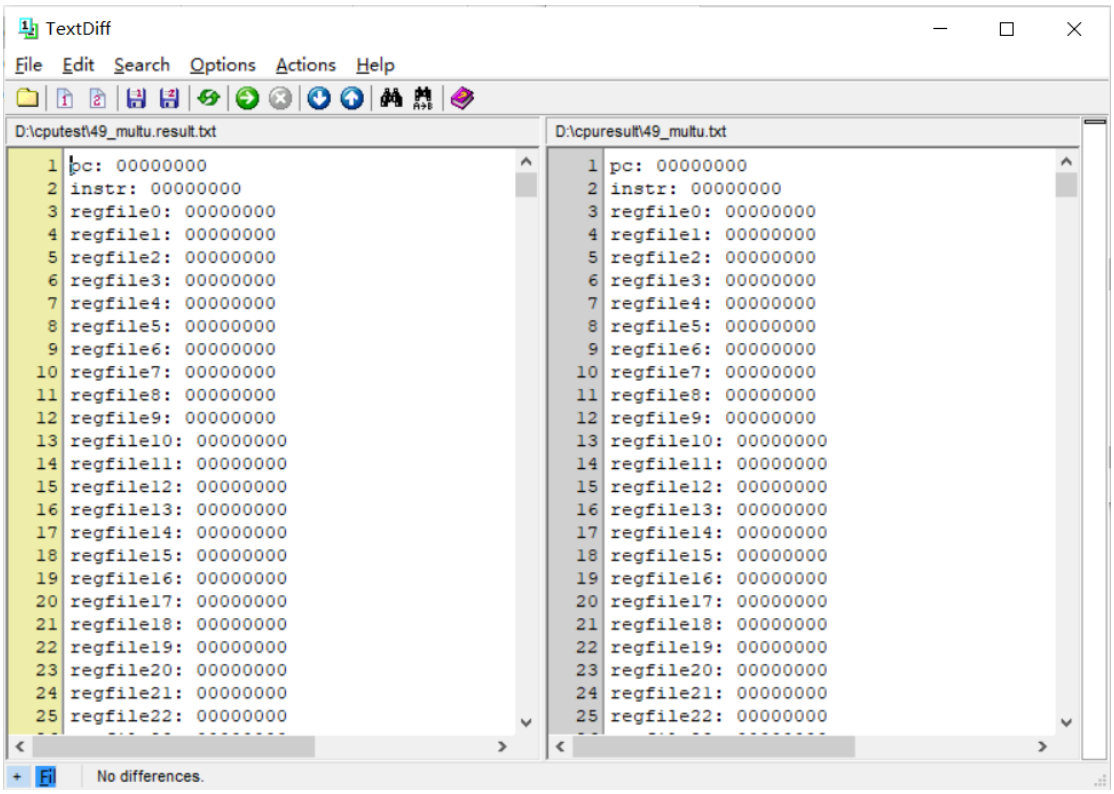
Mflomtlo:


TextDiff
File Edit Search Options Actions Help
D:\cputest\44.47_mflo.mtlo.result.txt
D:\cpuresult\44.47_mflo.mtlo.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

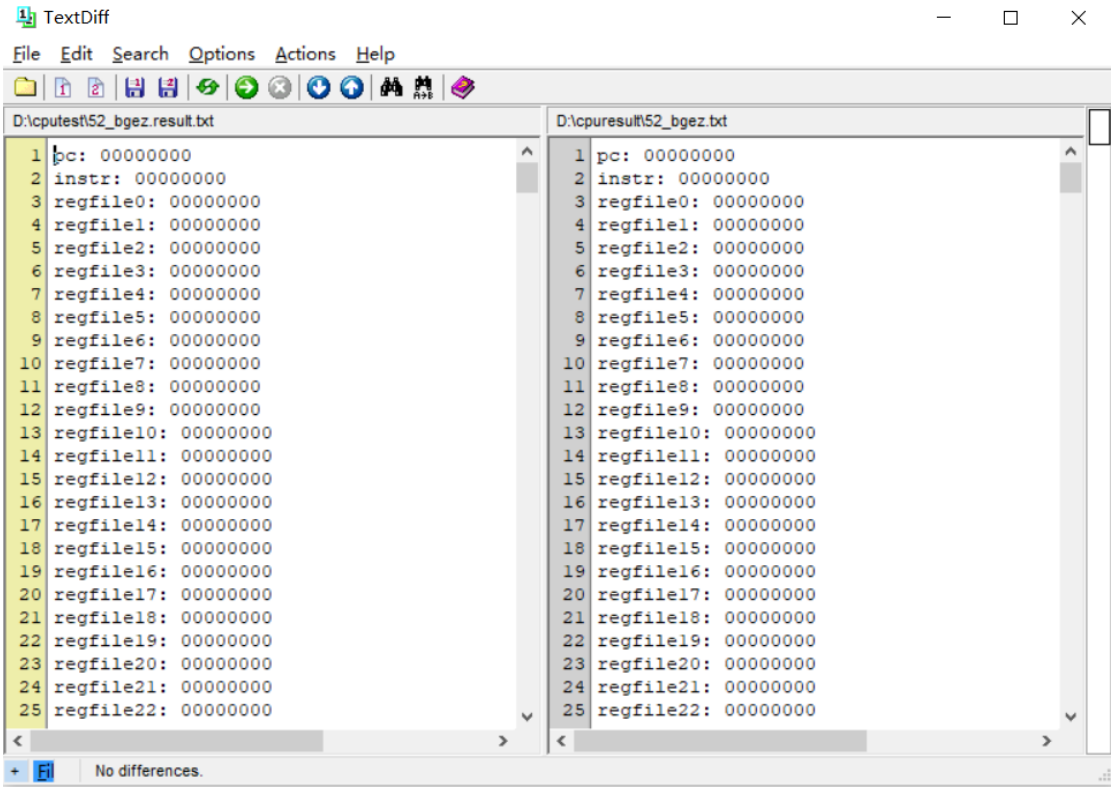
Mul:


TextDiff
File Edit Search Options Actions Help
D:\cputest\48_mul.result.txt
D:\cpuresult\48_mul.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

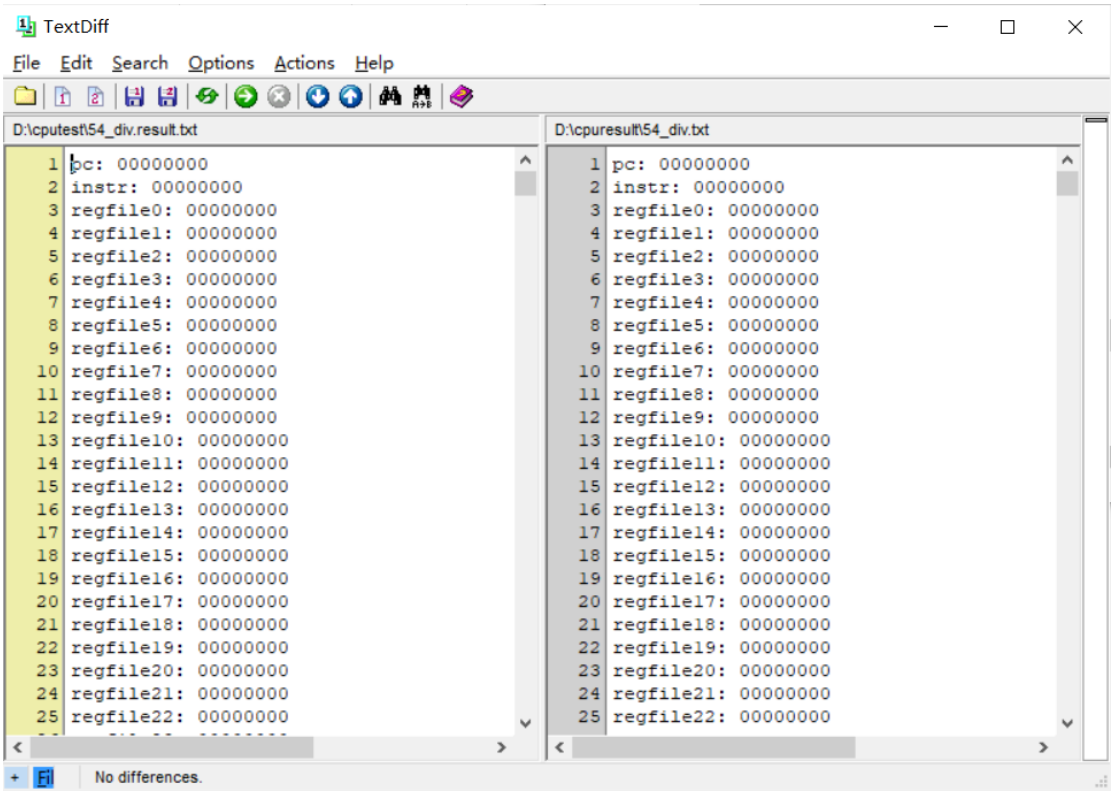
Multu:


TextDiff
File Edit Search Options Actions Help
D:\cputest\49_multu.result.txt
D:\cpuresult\49_multu.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

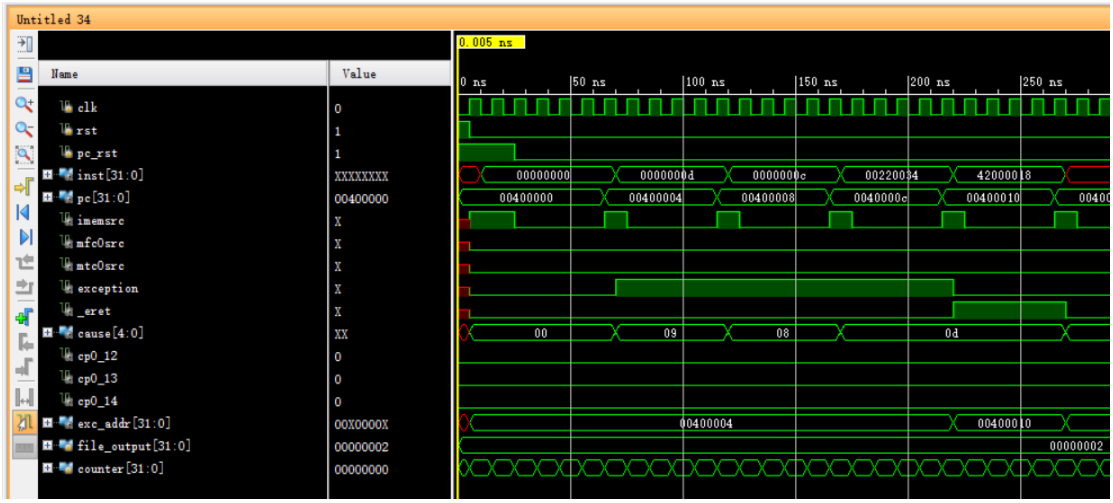
Bgez:


TextDiff
File Edit Search Options Actions Help
D:\cputest\52_bgez.result.txt
D:\cpuresult\52_bgez.txt
pc: 00000000
instr: 00000000
regfile0: 00000000
regfile1: 00000000
regfile2: 00000000
regfile3: 00000000
regfile4: 00000000
regfile5: 00000000
regfile6: 00000000
regfile7: 00000000
regfile8: 00000000
regfile9: 00000000
regfile10: 00000000
regfile11: 00000000
regfile12: 00000000
regfile13: 00000000
regfile14: 00000000
regfile15: 00000000
regfile16: 00000000
regfile17: 00000000
regfile18: 00000000
regfile19: 00000000
regfile20: 00000000
regfile21: 00000000
regfile22: 00000000
No differences.

Div:



4）cp0 测试：



cp0 相关信号正常。

9. 心得体会及建议

心得体会：

对于此次实验，第一点体会就是对 Verilog 语言的理解更深刻了。和 C 等高级语言相比，Verilog 语言更侧重语言与底层电路逻辑的对应，也就是说在写 Verilog 代码时本质是在设计底层电路。以设计电路的思路去写代码，能尽可能地去避免一些细节上的错误。

第二点体会就是加深了对 CPU 的基本原理与结构的理解。之前的计算机组成原理理论课，更多的是对于 CPU、内存等理论的掌握,但是,这样也只是能看懂数据的流通,指令的过程等,对 CPU 的深层理解还不足。而通过这次设计 54 条指令 CPU 的实验，将理论和实践结合起来，对 CPU 有了更加深入的理解。如何分析每一条指令，拆分模块，把每个模块合起来，控制器如何设计，如何逐步添加指令等；还有如何进行优化,怎样让 CPU 的结构更加清晰,怎样调整控制信号,怎样简化代码。这些都让我对 CPU 的原理、对每个部件的功能、对指令的流程、对模块的设计等有了更深的理解。

第三点体会就是加深了对 CPU、计算机底层硬件以及系统结构的兴趣。通过在实验中对每条指令的分析与实现,激发了我对 CPU 设计的兴趣。我也希望能继续深入学习相关的知识，一方面为探索自己的乐趣，另一方面为我国的 CPU 发展尽一份力。

建议：

本次 CPU 实验难度比较适中,并且资料给的也很充足,包括指令的分析方法、测试方法、测试用例等。因此实现起来也比较顺畅,没有出现特别难以处理的问题。如果要提建议的话，我认为：

一是希望可以对整体的设计过程进行更多的讲解，这样可以帮助同学们更快的上手设计，让大家的思路不会跑偏。

二是希望可以提供更多的测试用例，比如开放网上测试等，因为目前提供的测试用例还是主要针对每一条指令；即使有指令间的组合，也是几条指令组合，并没有把大量的指令组合起来，可能会有某些特殊情况考虑不周全，导致 CPU 设计出现 bug。