**同济大学计算机系**

**计算机组成原理实验报告**

****

**学 号 1954263**

**姓 名 宋子铭**

**专 业 计算机科学与技术**

**授课老师 郝泳涛**

1. 实验内容

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。

我设计的是单周期CPU。

1. 设计思路和数据通路

**整体思路：**

该次实验，我先将各条指令理解清晰，分别找出其对应操作和所需部件并画出数据通路图，然后分析所有指令各部件的输入输出对应信号。依照整理的信号表格和各指令的数据通路，绘制总的数据通路。最后在vivado上编写并借助modelsim，mars汇编器进行调试，最后下板验证实验成功。

**2.1 54条指令数据通路**

1 add

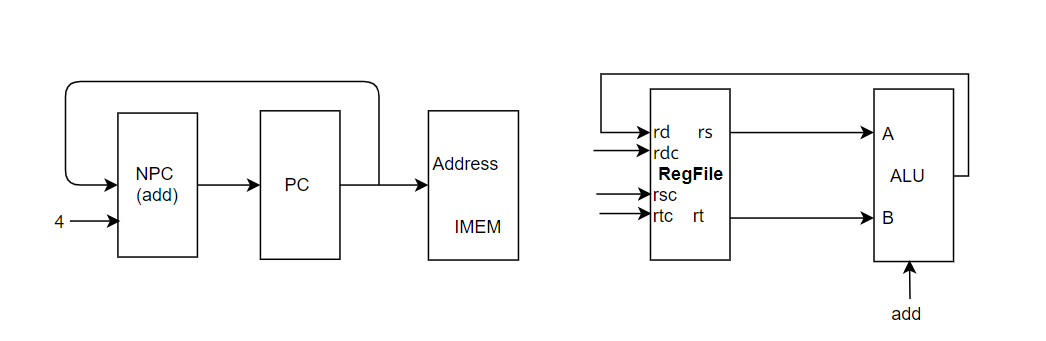
格式：add rd, rs, rt

操作：取指令，rd←rs+rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| add | NPC | PC | PC | ALU | rs | rt |



2 addu

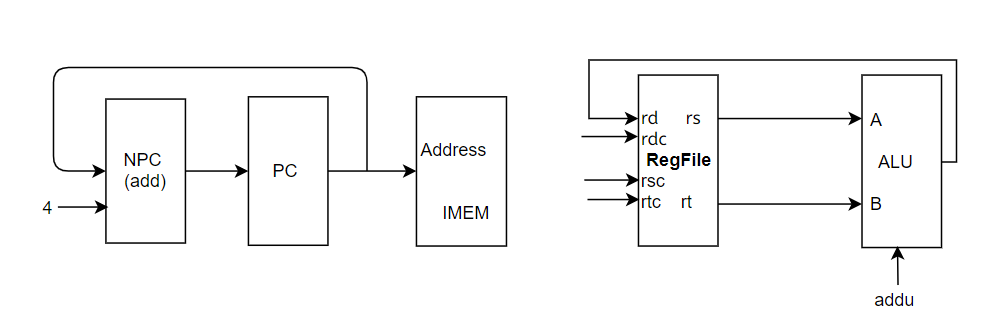
格式：addu rd, rs, rt

操作：取指令，rd←rs+rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| addu | NPC | PC | PC | ALU | rs | rt |



3 sub

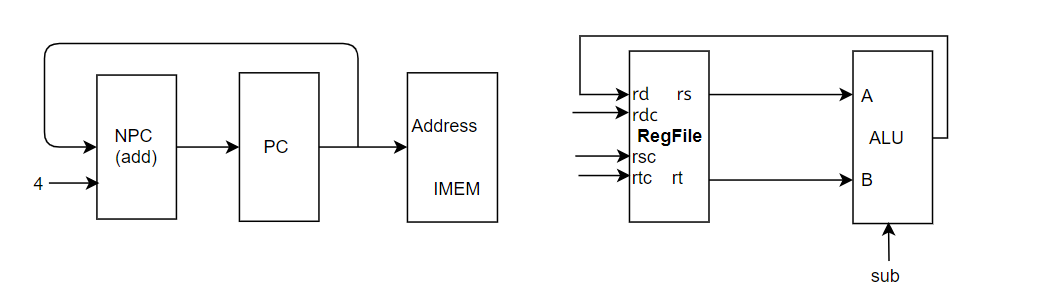
格式：sub rd, rs, rt

操作：取指令，rd←rs-rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| sub | NPC | PC | PC | ALU | rs | rt |



4 subu

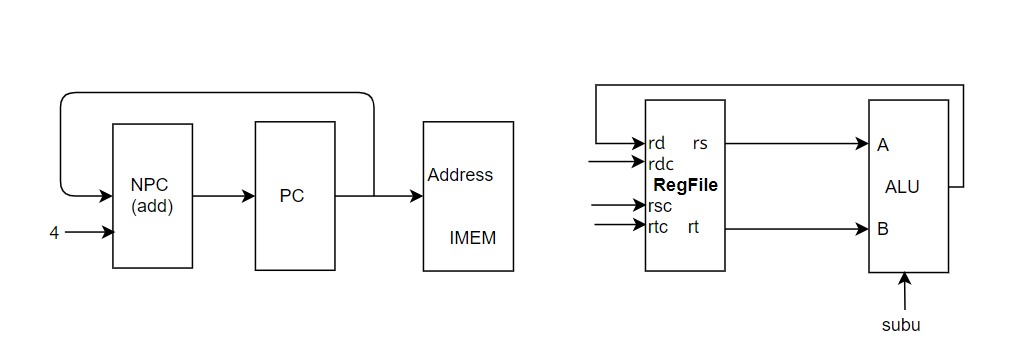
格式：subu rd, rs, rt

操作：取指令，rd←rs-rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| subu | NPC | PC | PC | ALU | rs | rt |



5 and

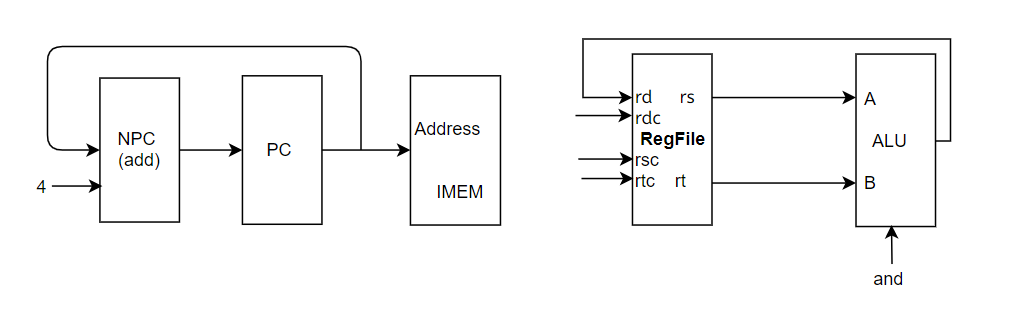
格式：and rd, rs, rt

操作：取指令，rd←rs&rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| and | NPC | PC | PC | ALU | rs | rt |



6 or

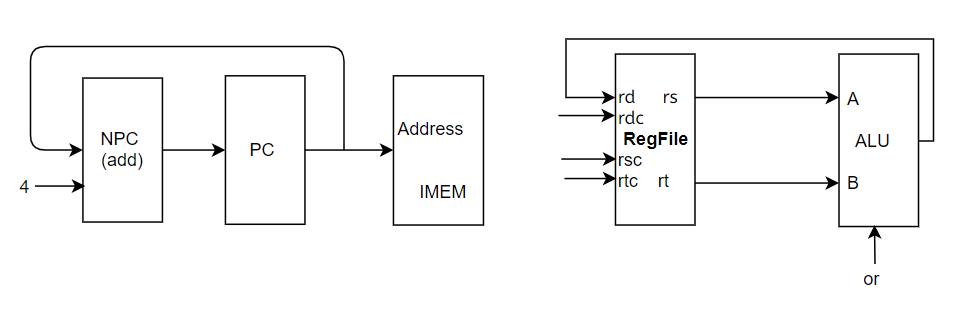
格式：or rd, rs, rt

操作：取指令，rd←rs|rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| or | NPC | PC | PC | ALU | rs | rt |



7 xor

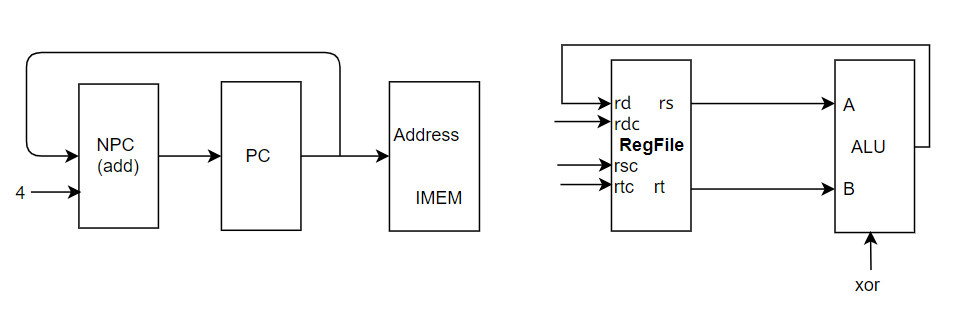
格式：xor rd, rs, rt

操作：取指令，rd←rs^rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| xor | NPC | PC | PC | ALU | rs | rt |



8 nor

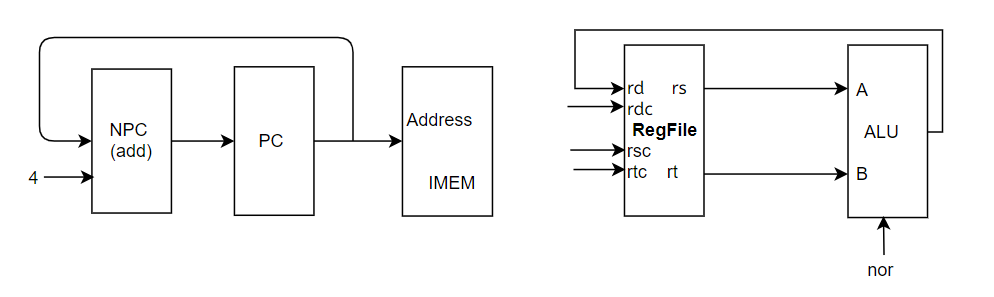
格式：nor rd, rs, rt

操作：取指令，rd←~(rs|rt), PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| nor | NPC | PC | PC | ALU | rs | rt |



9 slt

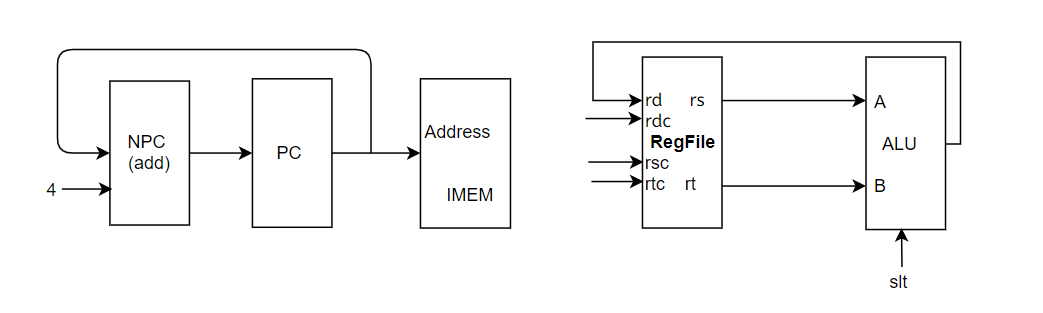
格式：slt rd, rs, rt

操作：取指令，rd←rs<rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| slt | NPC | PC | PC | ALU | rs | rt |



10 sltu

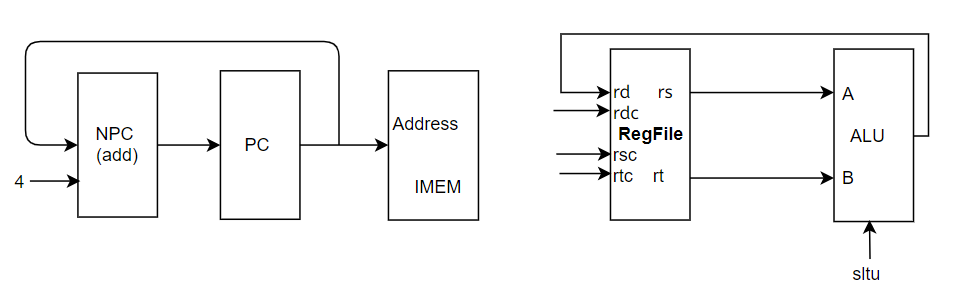
格式：sltu rd, rs, rt

操作：取指令，rd←rs<rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| sltu | NPC | PC | PC | ALU | rs | rt |



11 sll

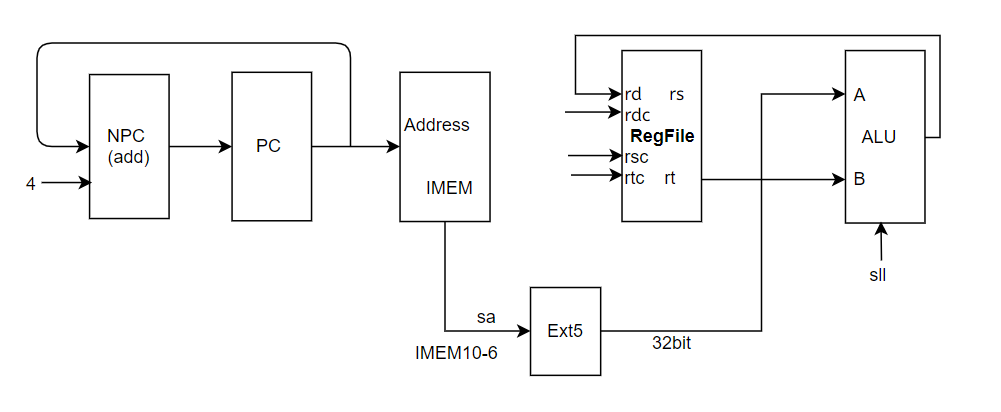
格式：sll rd, rt, sa

操作：取指令，rd←rt<<sa, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU,Ext5

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext5 |
|  |  |  |  | rd | A | B |  |
| sll | NPC | PC | PC | ALU | Ext5 | rt | sa(IM[10:6]) |



12 srl

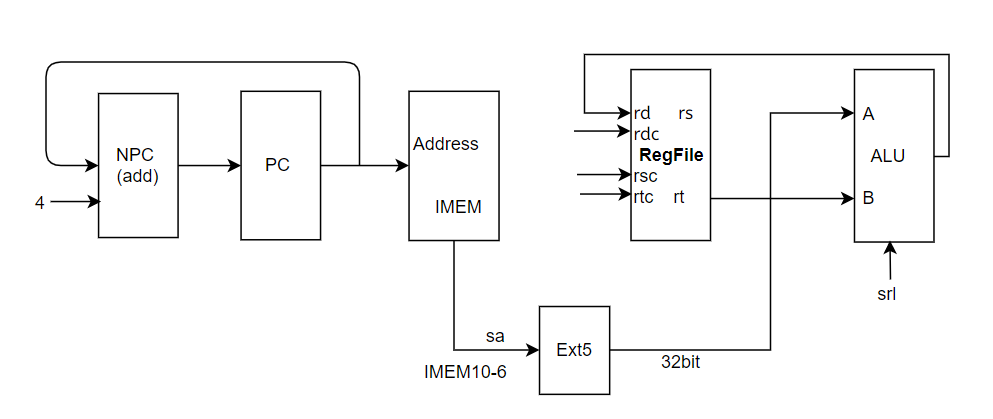
格式：srl rd, rt, sa

操作：取指令，rd←rt>>sa(逻辑), PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU,Ext5

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext5 |
|  |  |  |  | rd | A | B |  |
| srl | NPC | PC | PC | ALU | Ext5 | rt | sa(IM[10:6]) |



13 sra

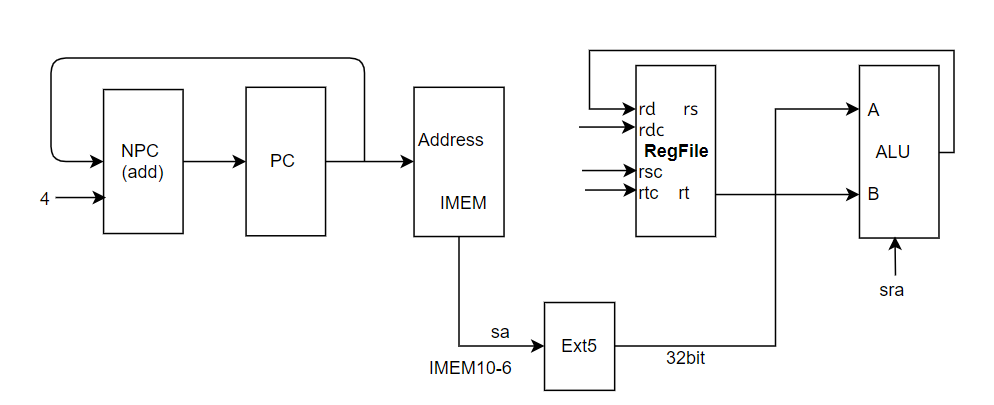
格式：sra rd, rt, sa

操作：取指令，rd←rt>>sa(算数), PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU,Ext5

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext5 |
|  |  |  |  | rd | A | B |  |
| srl | NPC | PC | PC | ALU | Ext5 | rt | sa(IM[10:6]) |



14 sllv

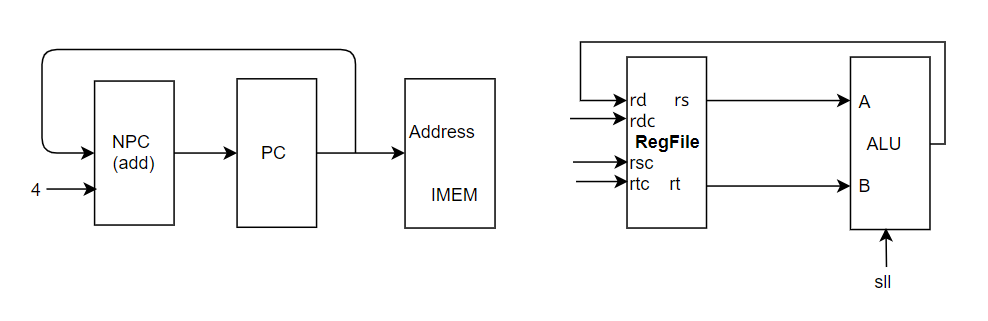
格式：sllv rd, rs, rt

操作：取指令，rd←rs<<rt, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| sllv | NPC | PC | PC | ALU | rs | rt |



15 srlv

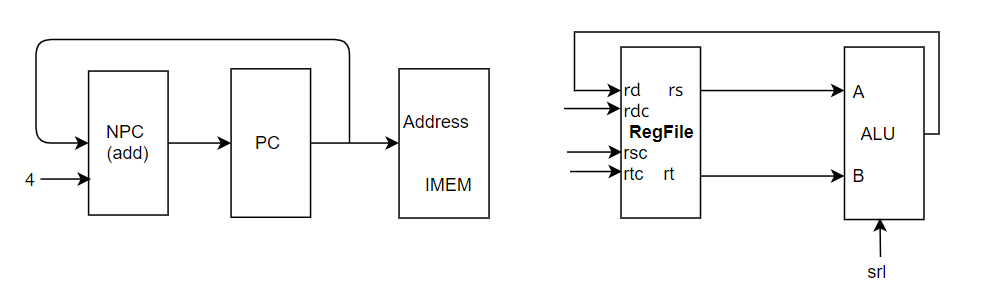
格式：srlv rd, rs, rt

操作：取指令，rd←rs>>rt(逻辑), PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| srlv | NPC | PC | PC | ALU | rs | rt |



16 srav

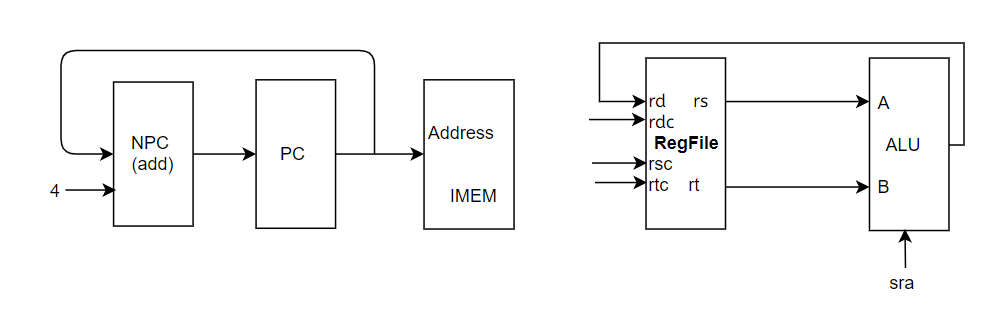
格式：srav rd, rs, rt

操作：取指令，rd←rs>>rt(算数), PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile, ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | |
|  |  |  |  | rd | A | B |
| srav | NPC | PC | PC | ALU | rs | rt |



17 jr

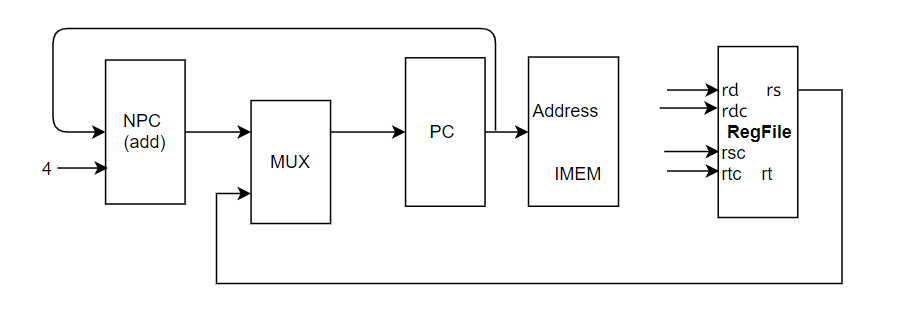
格式：jr rs

操作：取指令，PC←rs, PC←NPC(PC+4)

所需部件：PC, NPC, IMEM, Regfile,MUX

部件之间数据输入输出关系：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile |
|  |  |  |  | Rd |
| jr | rs | PC | PC |  |



18 addi

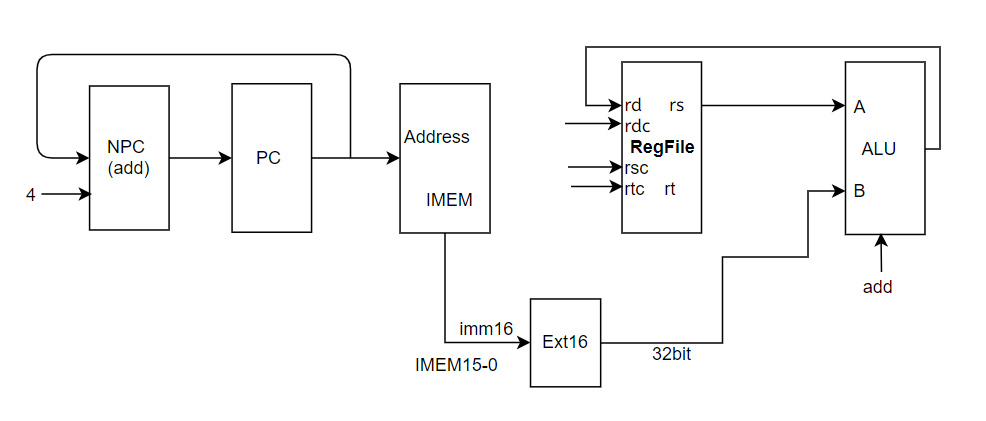
格式：addi rt, rs, imm16

操作：取指令、rt←rs+imm16(sign\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 |
|  |  |  |  | Rd | A | B |  |
| addi | NPC | PC | PC | ALU | rs | Ext16 | imm16 |



19 addiu

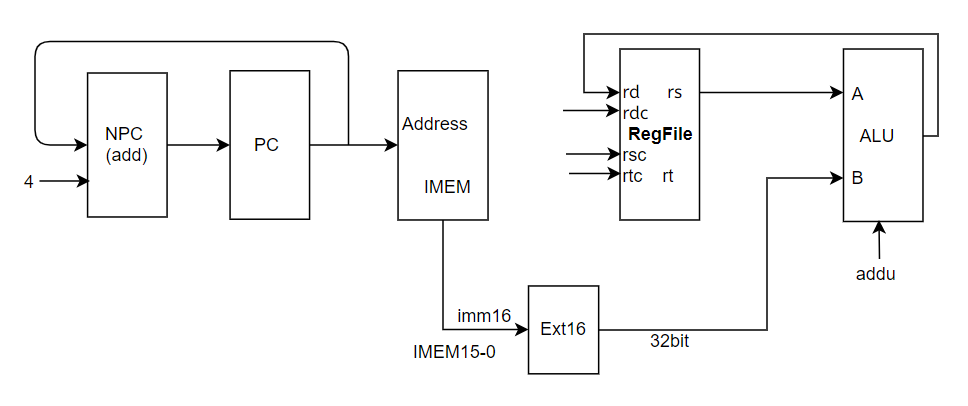
格式：addiu rt, rs, imm16

操作：取指令、rt←rs+imm16(sign\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 |
|  |  |  |  | Rd | A | B |  |
| addiu | NPC | PC | PC | ALU | rs | Ext16 | imm16 |



20 andi

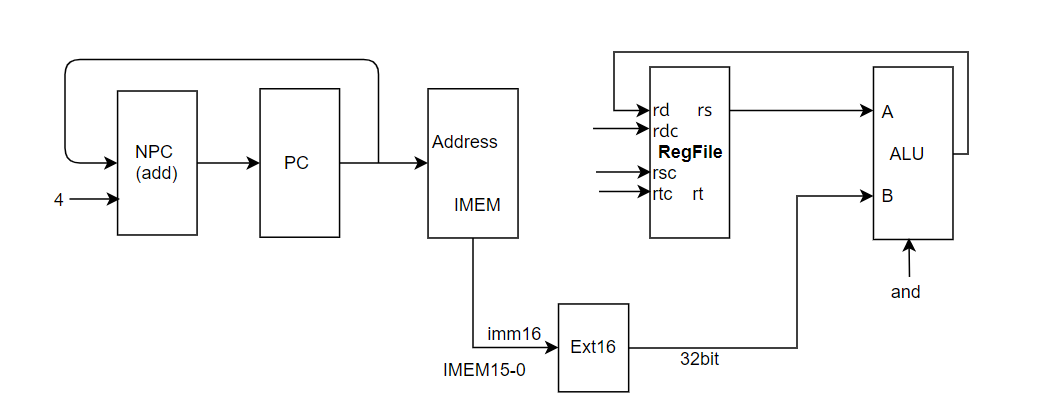
格式：andi rt, rs, imm16

操作：取指令、rt←rs&imm16(zero\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 |
|  |  |  |  | Rd | A | B |  |
| andi | NPC | PC | PC | ALU | rs | Ext16 | imm16 |



21 ori

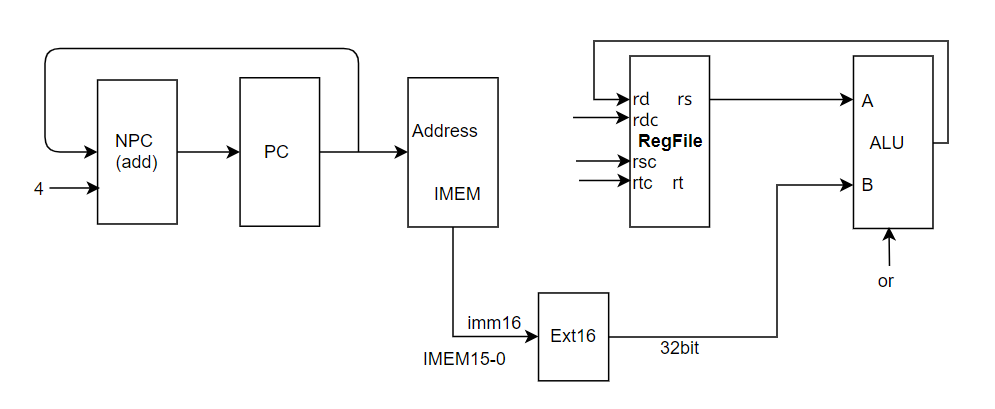
格式：ori rt, rs, imm16

操作：取指令、rt←rs|imm16(zero\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 |
|  |  |  |  | Rd | A | B |  |
| ori | NPC | PC | PC | ALU | rs | Ext16 | imm16 |



22 xori

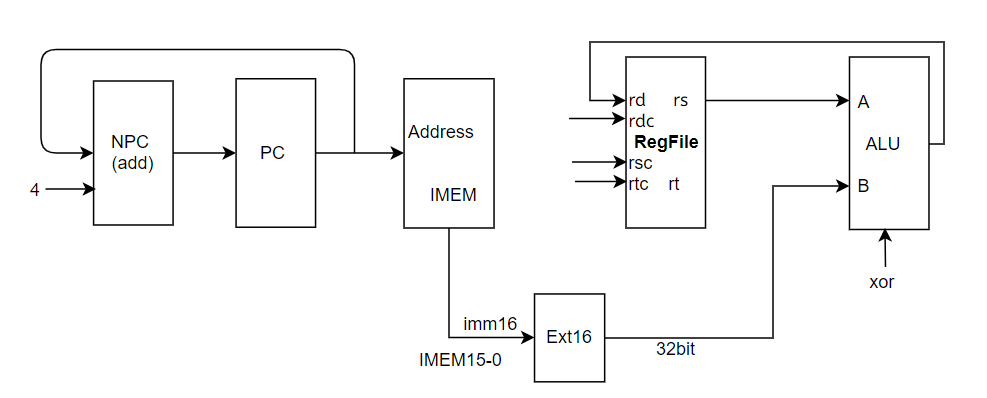
格式：xori rt, rs, imm16

操作：取指令、rt←rs^imm16(zero\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 |
|  |  |  |  | Rd | A | B |  |
| xori | NPC | PC | PC | ALU | rs | Ext16 | imm16 |



23 lw

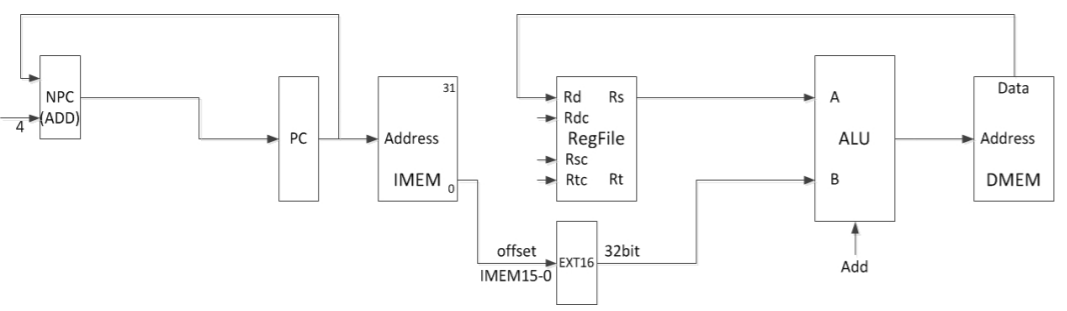
格式：lw rt, offset(base)

操作：取指令、rt←[rs+Sign\_ext\_offset] 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16、DMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 | DMEM | | |
|  |  |  |  | Rd | A | B |  | Addr | | Data |
| lw | NPC | PC | PC | DMEM(Data) | rs | Ext16 | offset | ALU |  | |



24 sw

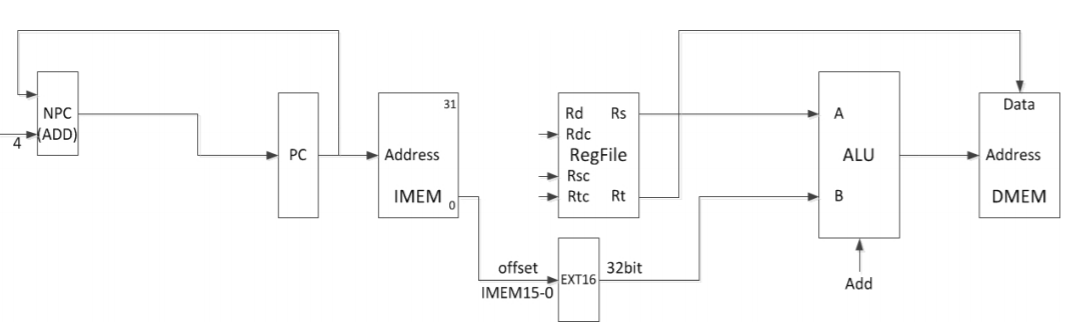
格式：sw rt, offset(base)

操作：取指令、[base+Sign\_ext\_offset]←rt, PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16、DMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 | DMEM | | |
|  |  |  |  | Rd | A | B |  | Addr | | Data |
| sw | NPC | PC | PC |  | rs | Ext16 | offset | ALU | rt | |



25 beq

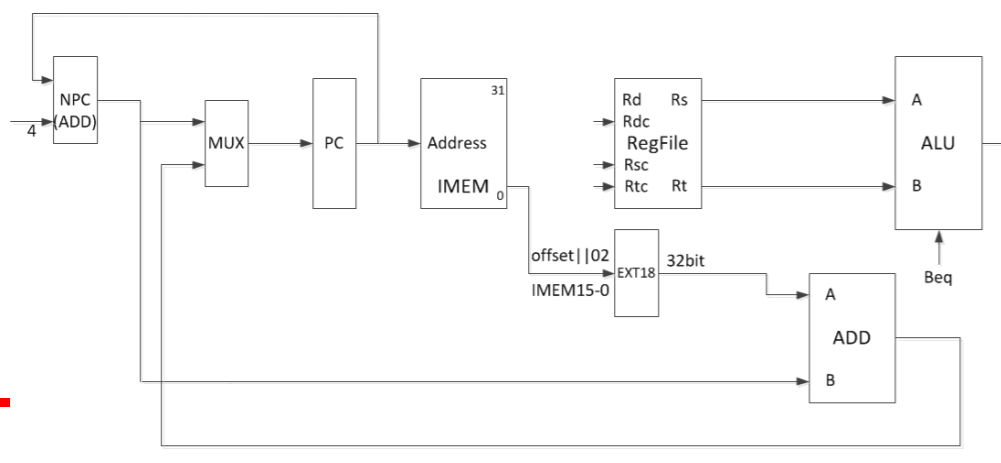
格式：beq rs,rt,offset

操作：取指令、rs=rt,PC←NPC+Sign\_ext(offset||02) 否则PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext18、ADD

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext18 | ADD | | |
|  |  |  |  | Rd | A | B |  | A | | B |
| beq | ADD | PC | PC |  | rs | rt | offset | NPC | Ext18 | |



26 bne

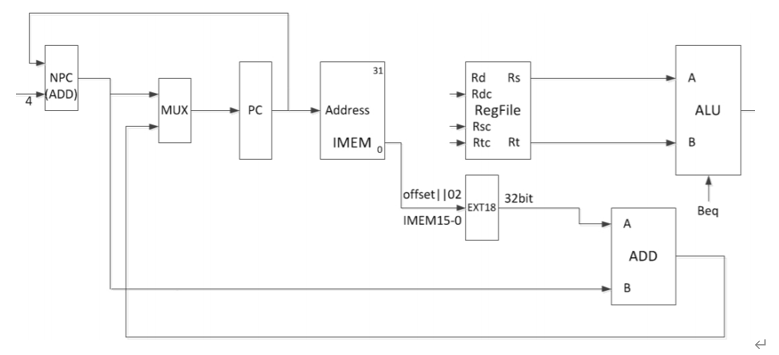
格式：bne rs,rt,offset

操作：取指令、rs≠rt,PC←NPC+Sign\_ext(offset||02 ) 否则PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext18、ADD

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext18 | ADD | | |
|  |  |  |  | Rd | A | B |  | A | | B |
| bne | ADD | PC | PC |  | rs | rt | offset | NPC | Ext18 | |



bne

27 slti

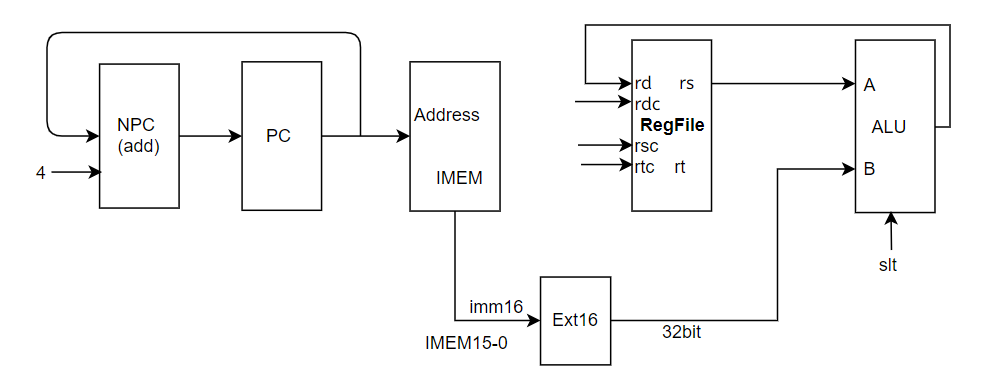
格式：slti rt, rs, imm16

操作：取指令、rt←rs<imm16(sign\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 |
|  |  |  |  | Rd | A | B |  |
| slti | NPC | PC | PC | ALU | rs | Ext16 | imm16 |



28 sltiu

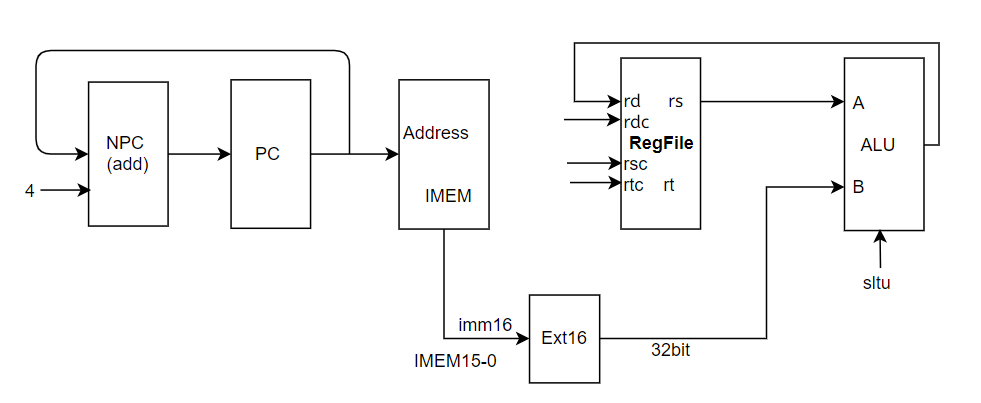
格式：sltiu rt, rs, imm16

操作：取指令、rt←rs<imm16(sign\_extend) 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 |
|  |  |  |  | Rd | A | B |  |
| sltiu | NPC | PC | PC | ALU | rs | Ext16 | imm16 |



29 lui

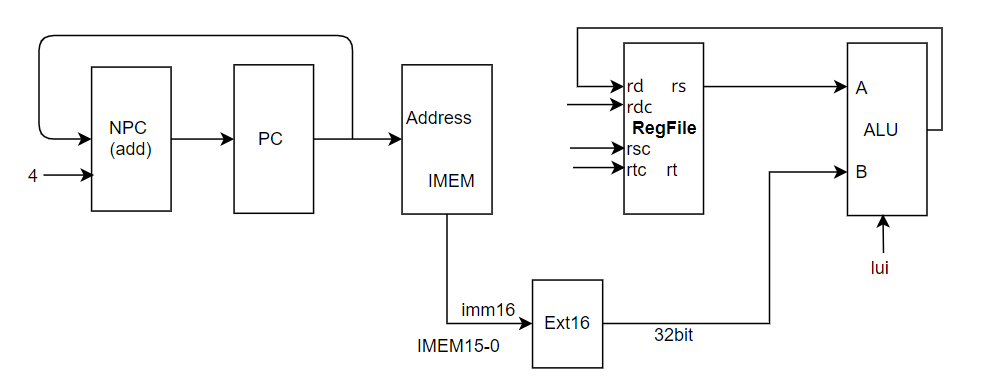
格式：lui rt, rs, imm16

操作：取指令、rt←imm16||016 、PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile、ALU、Ext16

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile | ALU | | Ext16 |
|  |  |  |  | Rd | A | B |  |
| lui | NPC | PC | PC | ALU | rs | Ext16 | imm16 |



30 j

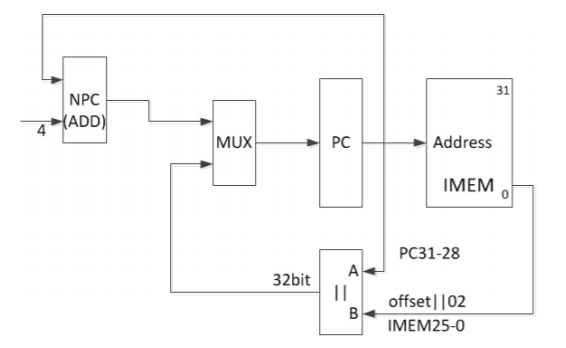
格式：j target

操作：取指令、PC←PC31-28||instr\_index||02 ，PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、||

部件之间数据输入输出关系：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | || | |
|  |  |  |  | A | B |
| j | || | PC | PC | PC31-28 | IMEM 25-0||02 |



31 jal

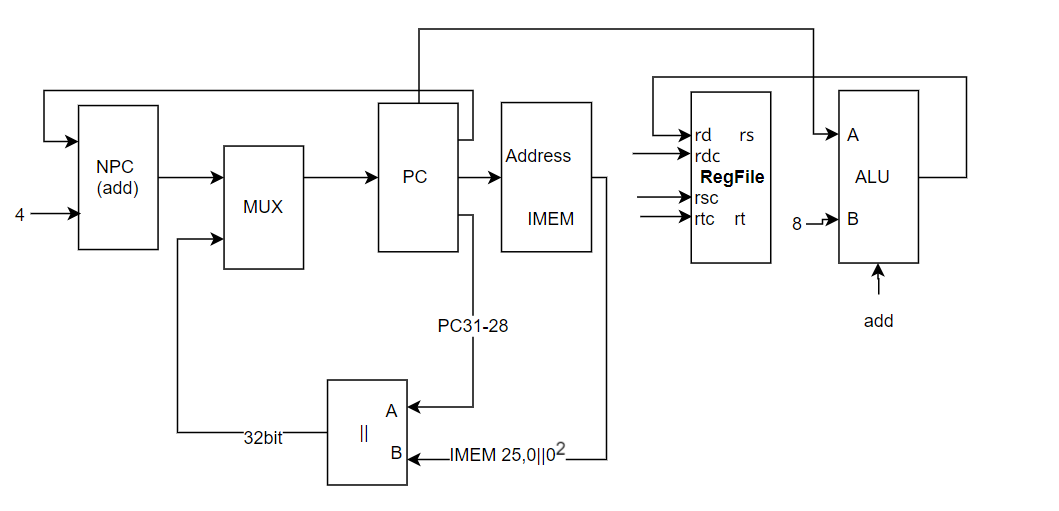
格式：jal target

操作：取指令、PC←PC31-28||instr\_index||02 ，PC←NPC(PC+4)，rd←PC+8

所需部件：PC、NPC、IMEM、||、ALU、

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | || | | ALU | |
|  |  |  |  | A | B | A | B |
| jal | || | PC | PC | PC31-28 | IMEM 25-0||02 | PC | 8 |



32 clz

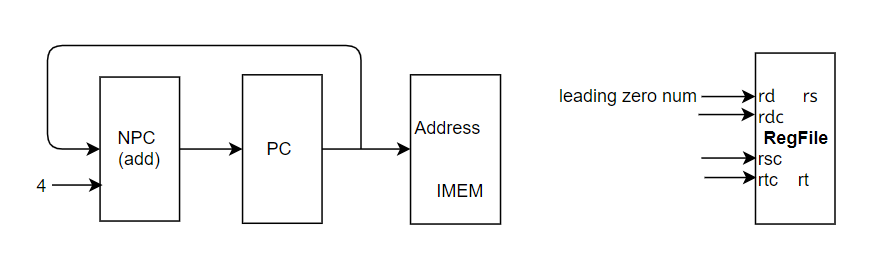
格式：clz rd, rs

操作：取指令、rd ←count\_leading\_zeros (rs), PC←NPC(PC+4)

所需部件：PC、NPC、ALU、IMEM、Regfile

部件之间数据输入输出关系：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Rgefile |
|  |  |  |  | Rd |
| clz | NPC | PC | PC | count\_leading\_zeros (rs) |



33 divu

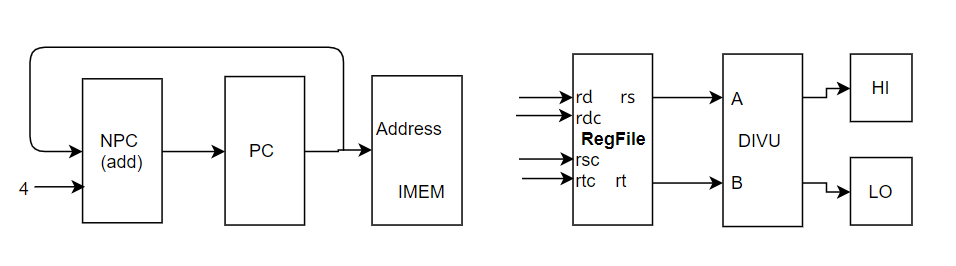
格式：divu rs, rt

操作：取指令、(HI，LO)←(unsign)rs/rt, PC←NPC(PC+4)

所需部件：PC、NPC、DIVU、IMEM、Regfile、HI、LO

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | DIVU | | HI | LO |
|  |  |  |  | dividend | divisor |  |  |
| divu | NPC | PC | PC | rs | rt | r | q |



34 eret

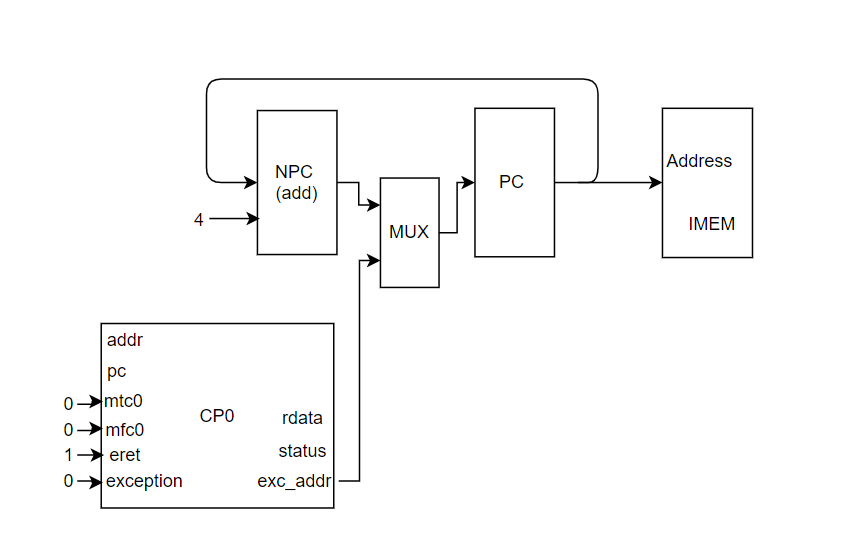
格式：eret

操作：取指令、PC←EPC，PC←NPC(PC+4)

所需部件：PC、NPC、CP0、IMEM

部件之间数据输入输出关系：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | CP0 |
|  |  |  |  | eret |
| eret | EPC | PC | PC | 1 |



35 jalr

格式：jalr rd rs

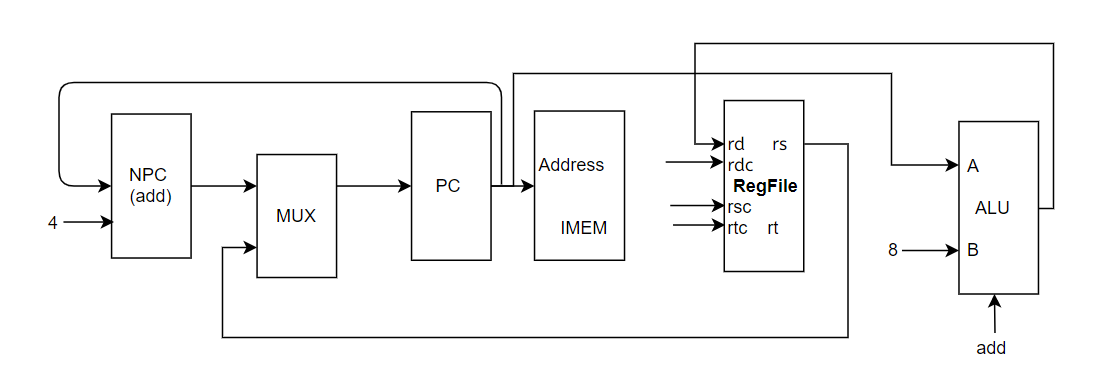
jalr rs (rd = 31 implied)

操作：取指令、rd ← PC+8 , PC ← rs，PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、Regfile

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Regfile | ALU | |
|  |  |  |  | rd | A | B |
| jalr | NPC | PC | PC | ALU.r | PC | 8 |



36 lb

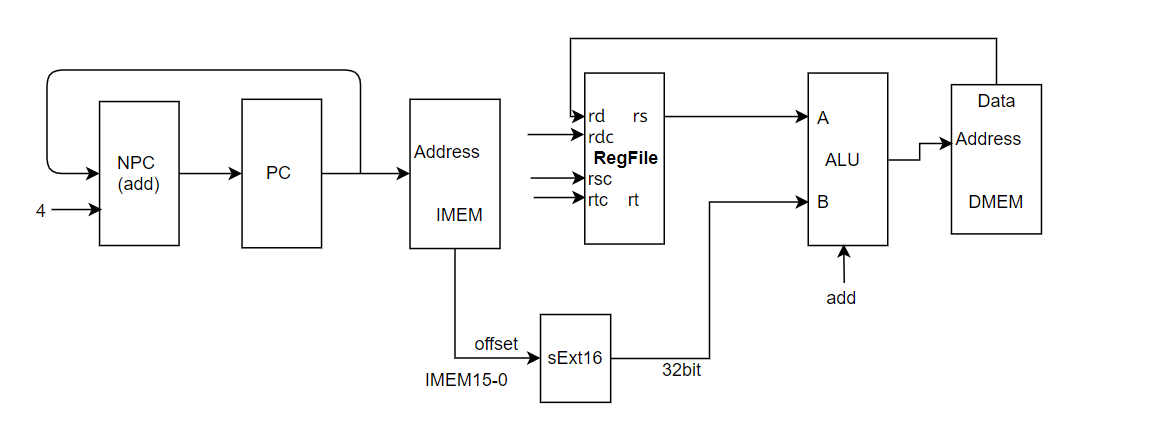
格式：lb rt,offset(base)

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

所需部件：PC、NPC、IMEM、Regfile、ALU、sExt16、DMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Regfile | ALU | | DMEM | Ext16 |
|  |  |  |  | rd | A | B | Addr |  |
| lb | NPC | PC | PC | DMEM.data | rs(base) | sExt16 | ALU.r | offset  (IMEM15-0) |



37 lbu

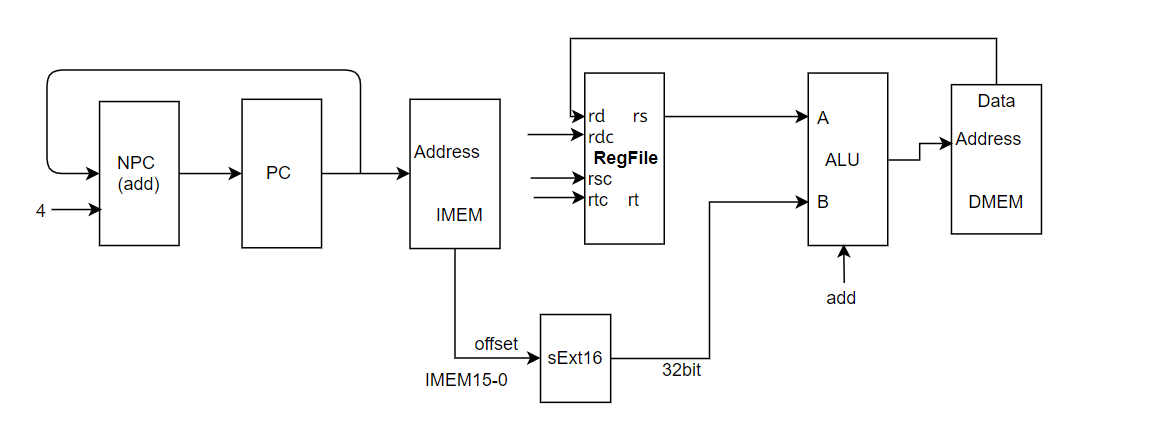
格式：lbu rt,offset(base)

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

所需部件：PC、NPC、IMEM、Regfile、ALU、sExt16、DMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Regfile | ALU | | DMEM | Ext16 |
|  |  |  |  | rd | A | B | Addr |  |
| lbu | NPC | PC | PC | DMEM.data | rs(base) | sExt16 | ALU.r | offset  (IMEM15-0) |



38 lhu

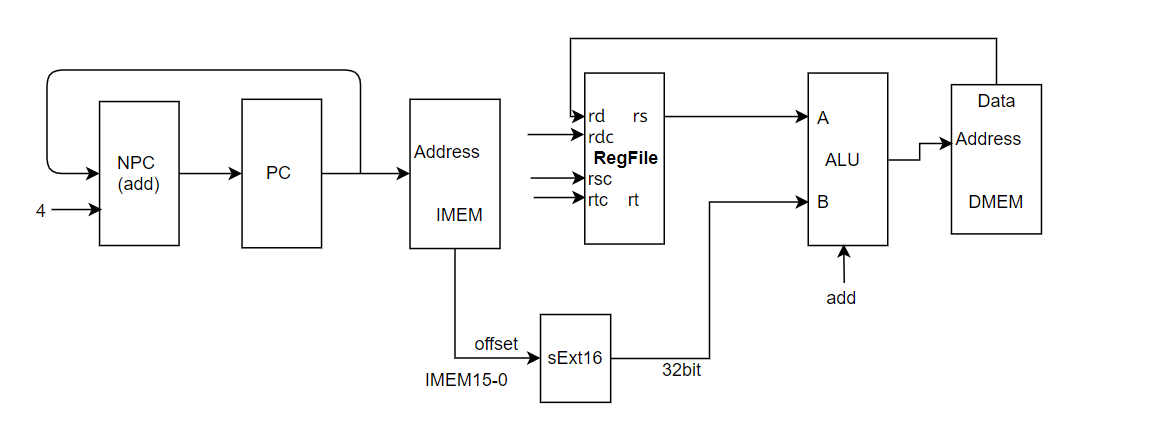
格式：lhu rt,offset(base)

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

所需部件：PC、NPC、IMEM、Regfile、ALU、sExt16、DMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Regfile | ALU | | DMEM | Ext16 |
|  |  |  |  | rd | A | B | Addr |  |
| lhu | NPC | PC | PC | DMEM.data | rs(base) | sExt16 | ALU.r | offset  (IMEM15-0) |



39 sb

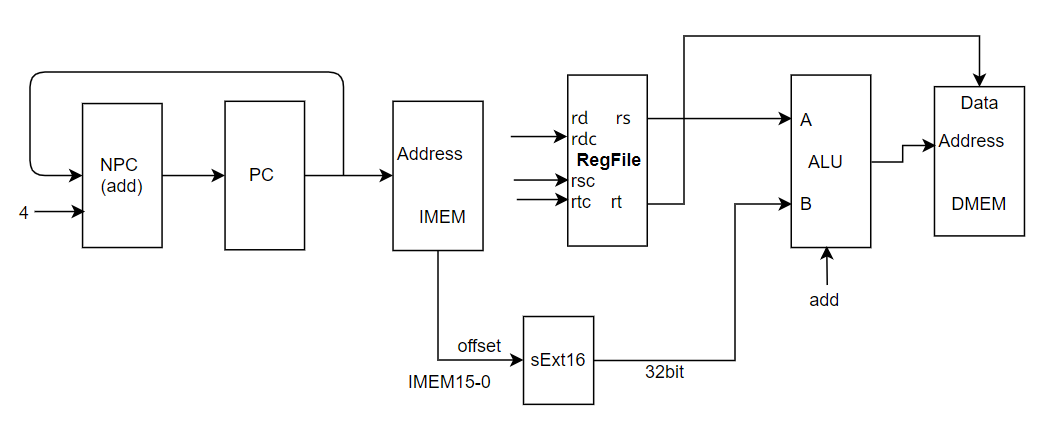
格式：sb rt,offset(base)

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

所需部件：PC、NPC、IMEM、Regfile、ALU、sExt16、DMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Regfile | ALU | | DMEM | | Ext16 |
|  |  |  |  | rt | A | B | Addr | data |  |
| sb | NPC | PC | PC | DMEM.data | rs(base) | sExt16 | ALU.r | RF.rt | offset  (IMEM15-0) |



40 sh

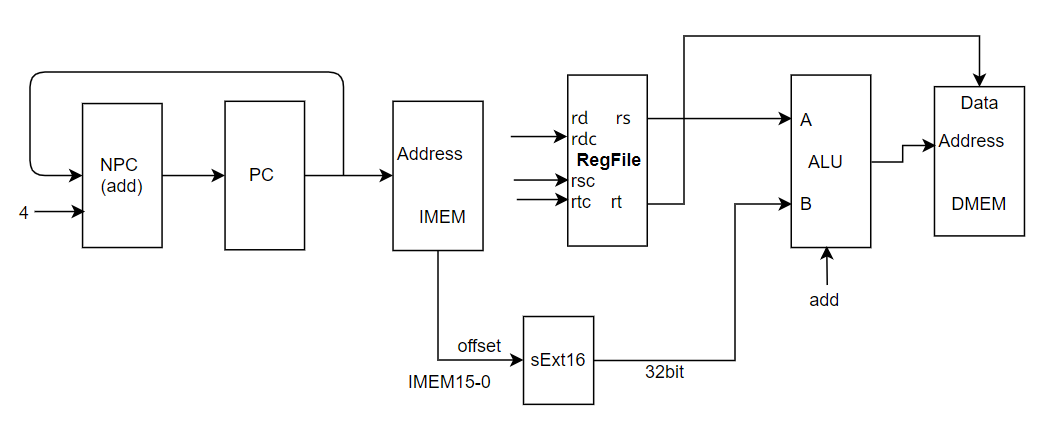
格式：sh rt,offset(base)

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

所需部件：PC、NPC、IMEM、Regfile、ALU、sExt16、DMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Regfile | ALU | | DMEM | | Ext16 |
|  |  |  |  | rt | A | B | Addr | data |  |
| sh | NPC | PC | PC | DMEM.data | rs(base) | sExt16 | ALU.r | RF.rt | offset  (IMEM15-0) |



41 lh

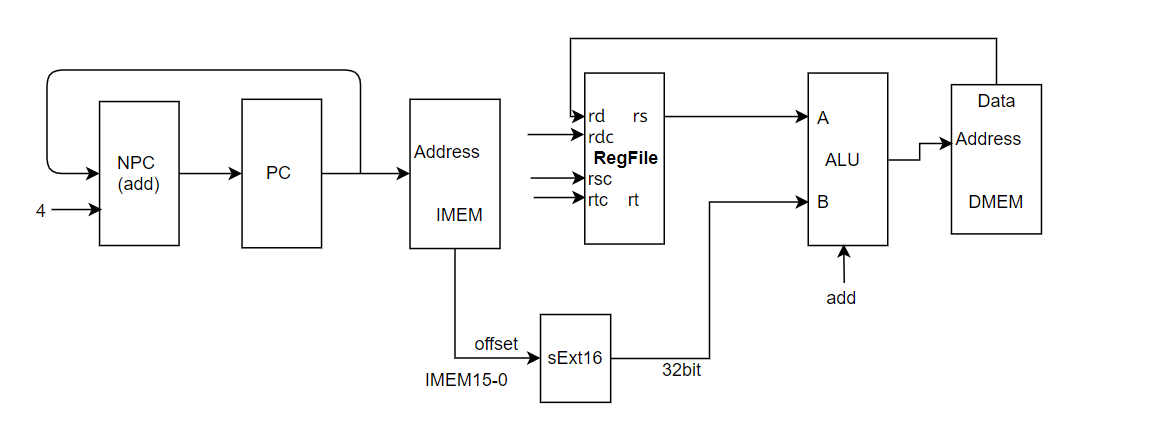
格式：lh rt,offset(base)

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

所需部件：PC、NPC、IMEM、Regfile、ALU、uExt16、DMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Regfile | ALU | | DMEM | Ext16 |
|  |  |  |  | rd | A | B | Addr |  |
| lh | NPC | PC | PC | DMEM.data | rs(base) | uExt16 | ALU.r | offset  (IMEM15-0) |



42 mfc0

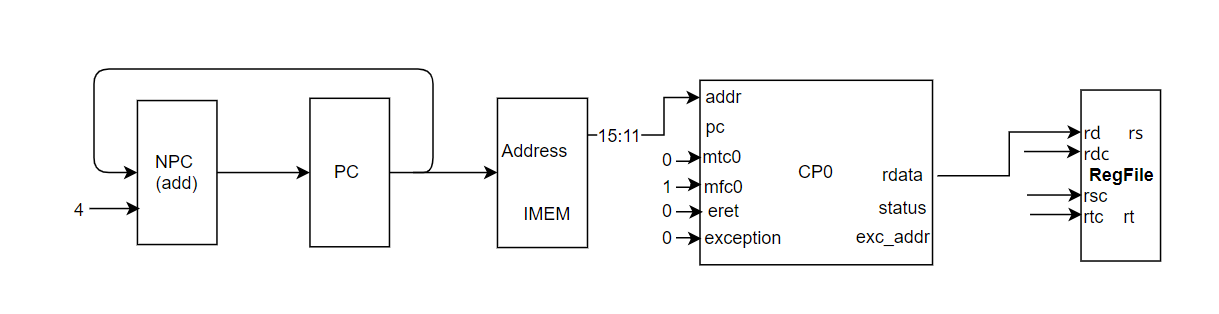
格式：mfc0 rt, rd

操作：取指令、rt←CP0 rd，PC←NPC(PC+4)

所需部件：PC、NPC、CP0、IMEM、Regfile

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | CP0 | | Regfile |
|  |  |  |  | mfc0 | addr(Rd) | rd |
| mfc0 | NPC | PC | PC | 1 | IMEM 15-11 | CP0.rdata |



43 mfhi

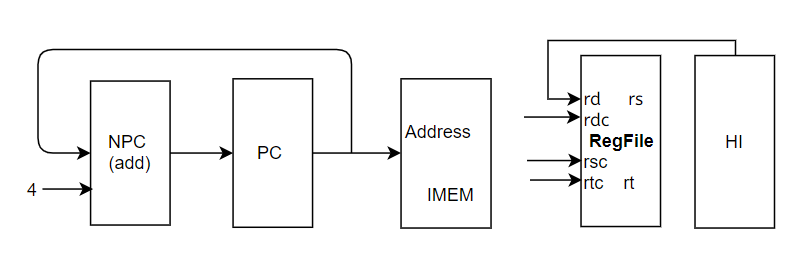
格式：mfhi rd

操作：取指令、rd← HI，PC←NPC(PC+4)

所需部件：PC、NPC、HI、IMEM、Regfile

部件之间数据输入输出关系：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Regfile |
|  |  |  |  | rd |
| mfhi | NPC | PC | PC | HI |



44 mflo

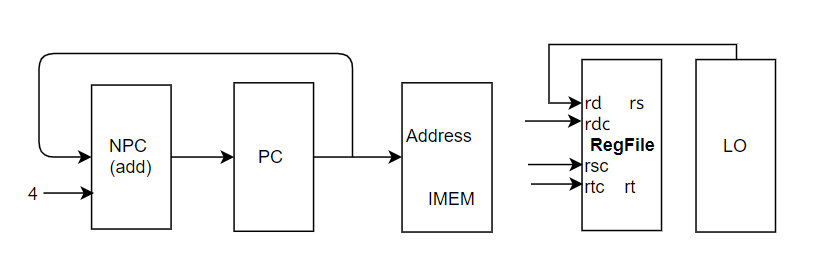
格式：mflo rd

操作：取指令、rd← LO，PC←NPC(PC+4)

所需部件：PC、NPC、LO、IMEM、Regfile

部件之间数据输入输出关系：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | Regfile |
|  |  |  |  | rd |
| mflo | NPC | PC | PC | LO |



45 mtc0

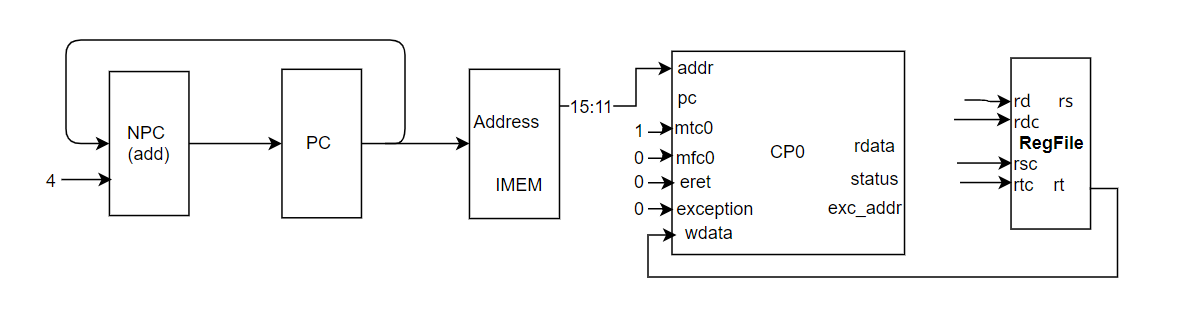
格式：mtc0 rt, rd

操作：取指令、CP0 rd←rt，PC←NPC(PC+4)

所需部件：PC、NPC、CP0、IMEM、Regfile

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | CP0 | | |
|  |  |  |  | mtc0 | addr(Rd) | wdata |
| mtc0 | NPC | PC | PC | 1 | IMEM 15-11 | RF.rt |



46 mthi

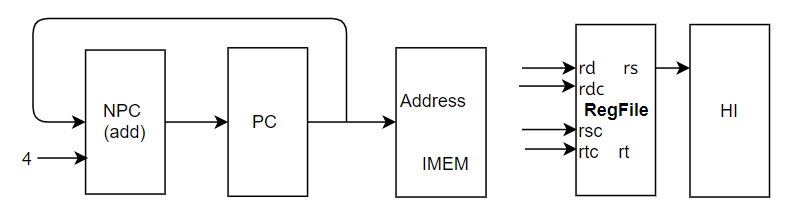
格式：mthi rd

操作：取指令、HI←rs，PC←NPC(PC+4)

所需部件：PC、NPC、HI、IMEM、Regfile

部件之间数据输入输出关系：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | HI |
| mthi | NPC | PC | PC | RF.rs |



47 mtlo

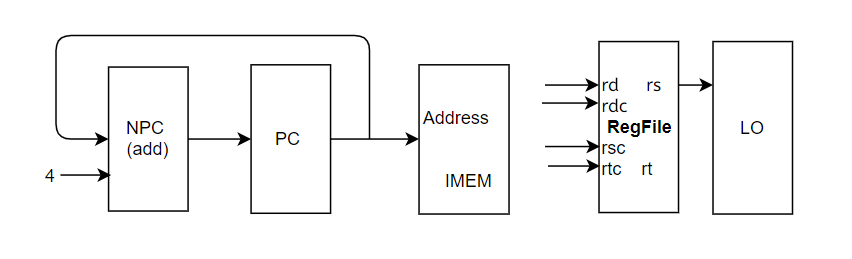
格式：mtlo rd

操作：取指令、LO←rs，PC←NPC(PC+4)

所需部件：PC、NPC、LO、IMEM、Regfile

部件之间数据输入输出关系：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | LO |
| mtlo | NPC | PC | PC | RF.rs |



48 mul

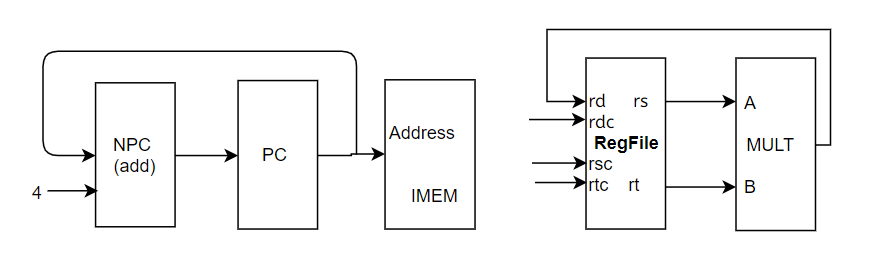
格式：mul rd, rs, rt

操作：取指令、rd←(sign)rs\*rt, PC←NPC(PC+4)

所需部件：PC、NPC、MULT、IMEM、Regfile

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | MULT | | Regfile |
|  |  |  |  | a | b | rd |
| div | NPC | PC | PC | rs | rt | z(lo) |



49 multu

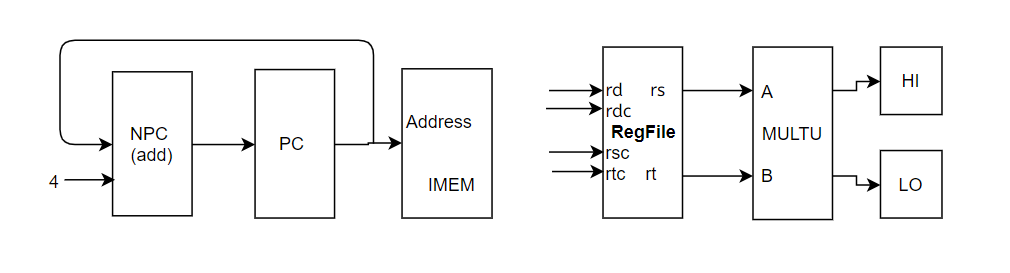
格式：multu rs, rt

操作：取指令、(HI，LO)←(unsign)rs\*rt, PC←NPC(PC+4)

所需部件：PC、NPC、MULTU、IMEM、Regfile、HI、LO

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | MULTU | | HI | LO |
|  |  |  |  | a | b |  |  |
| multu | NPC | PC | PC | rs | rt | z(hi) | z(lo) |



50 syscall

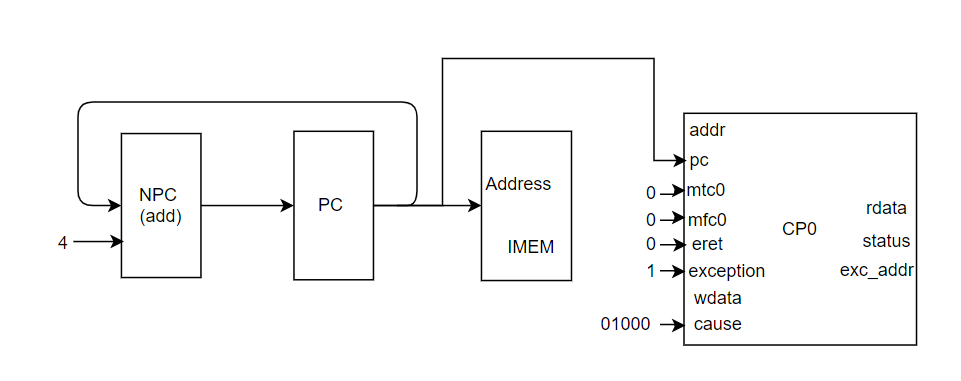
格式：syscall

操作：取指令、EPC←PC，cause← 01000，status<<5，PC←NPC(PC+4)

所需部件：PC、NPC、CP0、IMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | CP0 | | |
|  |  |  |  | exception | cause | EPC |
| syscall | NPC | PC | PC | 1 | 01000 | PC |



51 teq

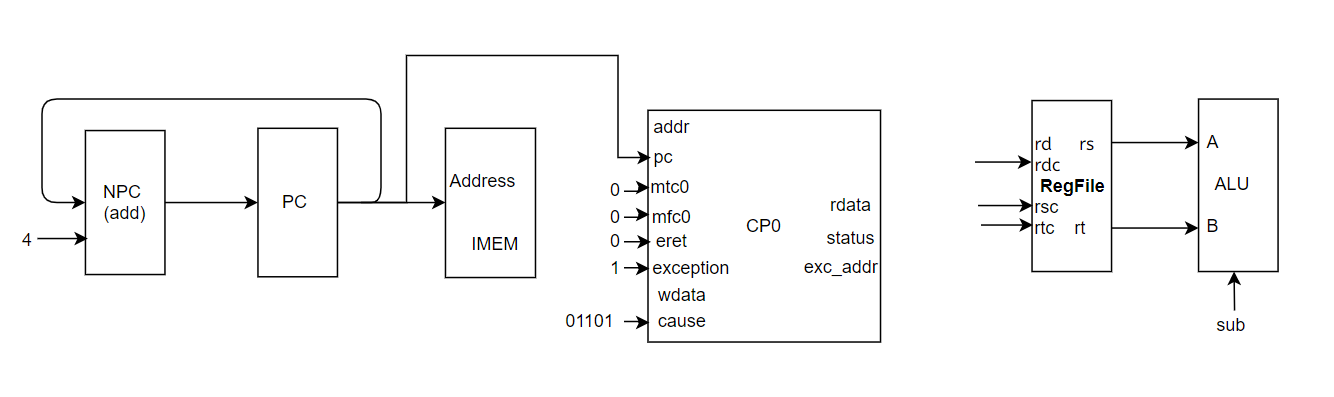
格式：teq rs, rt

操作：取指令、rs-rt，EPC←PC , cause← 01101，status<<5，PC←NPC(PC+4)

所需部件：PC、NPC、CP0、IMEM、Regfile、ALU

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | CP0 | | | ALU | |
|  |  |  |  | exception | cause | EPC | A | B |
| teq | NPC | PC | PC | 1 | 01101 | PC | rs | rt |



52 bgez

格式：bgez rs,offset

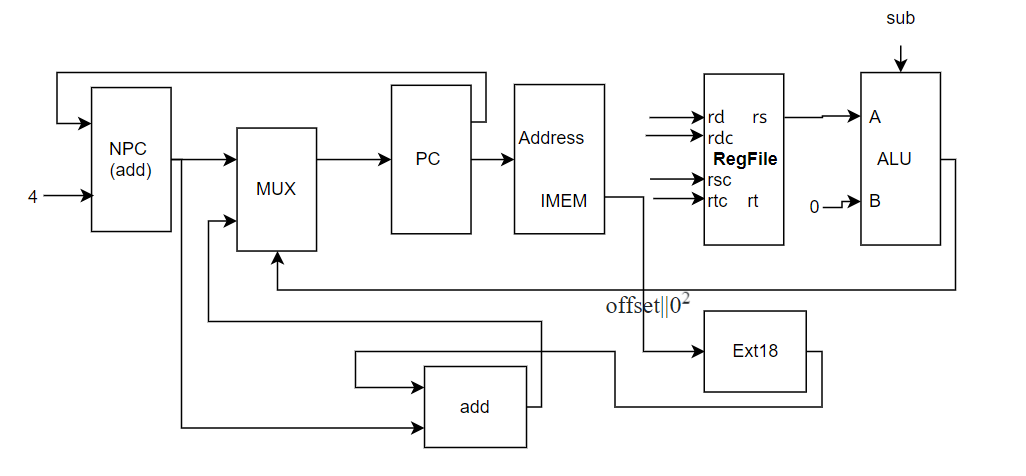
操作：取指令、if(rs>=0)PC←PC+Sign\_ext(offset||02)

else PC←NPC(PC+4)

所需部件：PC、NPC、IMEM、sExt18

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | sExt18 | ALU | |
|  |  |  |  |  | A | B |
| bgez | ALU.r | PC | PC | offset||02 | rs | 0 |



53 break

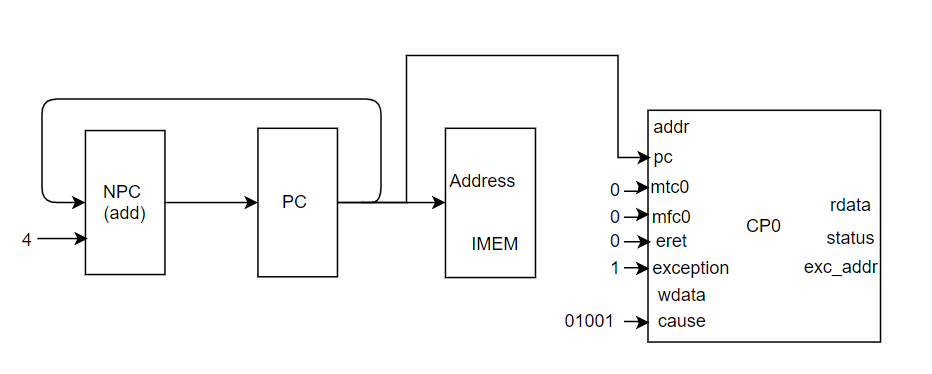
格式：break

操作：取指令、EPC←PC , cause← 01001，status<<5，PC←NPC(PC+4)

所需部件：PC、NPC、CP0、IMEM

部件之间数据输入输出关系：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | CP0 | | |
|  |  |  |  | exception | cause | EPC |
| break | NPC | PC | PC | 1 | 01001 | PC |



54 div

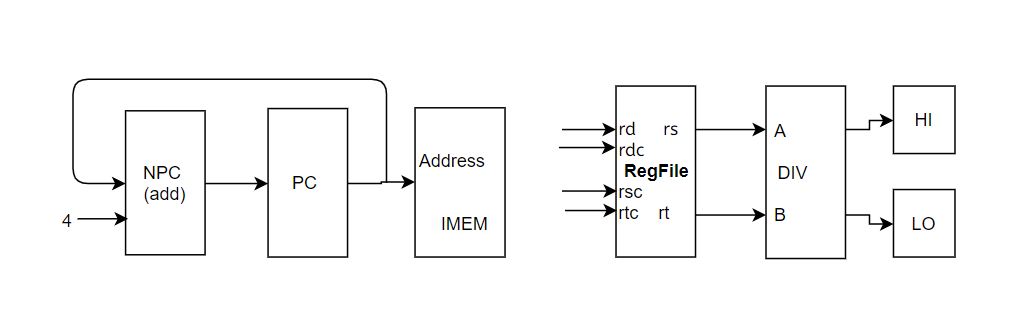
格式：div rs, rt

操作：取指令、(HI，LO)←(sign)rs/rt, PC←NPC(PC+4)

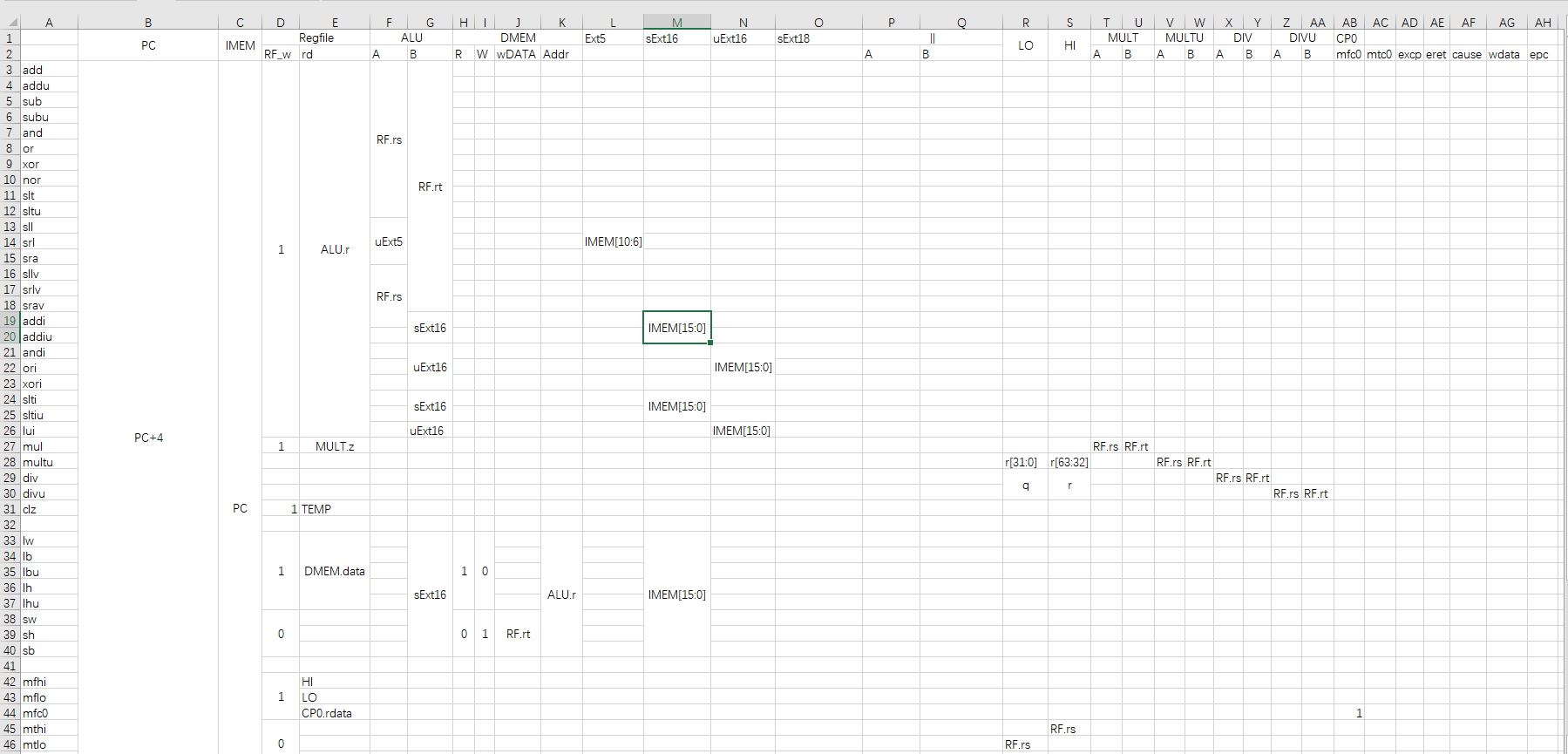
所需部件：PC、NPC、DIV、IMEM、Regfile、HI、LO

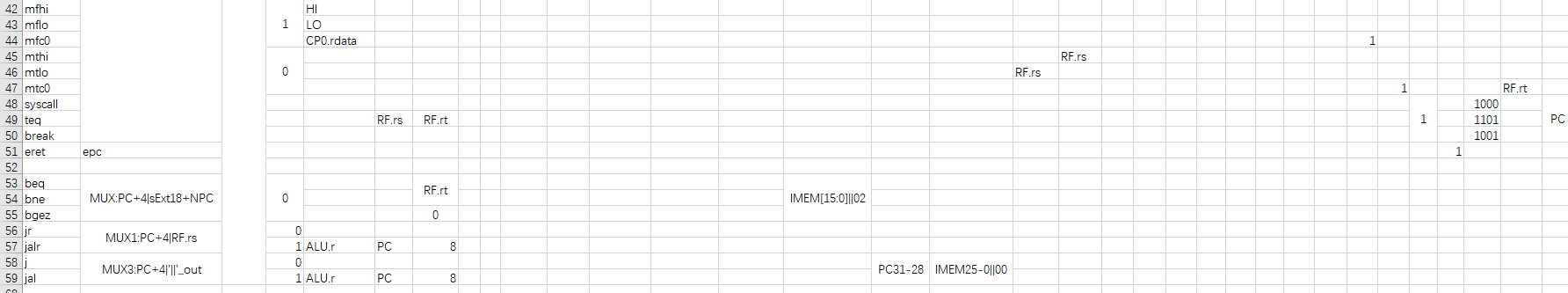
部件之间数据输入输出关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | PC | NPC | IMEM | DIV | | HI | LO |
|  |  |  |  | dividend | divisor |  |  |
| div | NPC | PC | PC | rs | rt | r | q |



**2.2输入输出关系表**

****

****

**2.3总数据通路图**

1. 模块建模

**3.1运算器模块，组合逻辑进行各类运算**

`define ALU\_addu 4'b0000

`define ALU\_add 4'b0010

`define ALU\_subu 4'b0001

`define ALU\_sub 4'b0011

`define ALU\_and 4'b0100

`define ALU\_or 4'b0101

`define ALU\_xor 4'b0110

`define ALU\_nor 4'b0111

`define ALU\_lui 4'b1000

`define ALU\_slt 4'b1011

`define ALU\_sltu 4'b1010

`define ALU\_sra 4'b1100

`define ALU\_sll 4'b1110

`define ALU\_srl 4'b1101

module alu(

input [31:0] a,

input [31:0] b,

input [3:0] aluc, //control

output[31:0] r, //result

output zero,

output carry,

output negative,

output overflow

);

reg [32:0] result;

wire signed [31:0] sa=a,sb=b;

always@(\*)begin

case(aluc)

`ALU\_addu:

result = a+b;

`ALU\_add:

result = sa+sb;

`ALU\_subu:

result = a-b;

`ALU\_sub:

result = sa-sb;

`ALU\_and:

result = a&b;

`ALU\_or:

result = a|b;

`ALU\_xor:

result = a^b;

`ALU\_nor:

result = ~(a|b);

`ALU\_lui:

result = {b[15:0], 16'b0};

`ALU\_slt:

result = sa<sb? 1:0;

`ALU\_sltu:

result = a<b? 1:0;

`ALU\_sra: begin

if (a==0)

{result[31:0],result[32]} = {b,1'b0};

else

{result[31:0],result[32]} = sb>>>(a-1);

end

`ALU\_srl: begin

if(a==0)

{result[31:0],result[32]} = {b,1'b0};

else

{result[31:0],result[32]} = b>>(a-1);

end

`ALU\_sll:

result = b<<a;

default:

result = a+b;

endcase

end

assign r = result[31:0];

assign carry = result[32];

assign zero = (r==32'b0) ?1'b1:1'b0;

assign negative = result[31];

assign overflow = result[32];

endmodule

**3.2 CPU模块，调用控制器、运算器，程序计数器，寄存器，实现31条指令。**

module cpu(

input clk,

input reset,

input [31:0]inst,

input [31:0] dm\_data,

output [31:0] Rt,

output [31:0] alu\_r,

output [31:0] pc,

output dm\_ena,

output dm\_w,

output dm\_r

);

wire [31:0] sExt18;

wire [31:0] sExt16,uExt16;

wire [8:0] MUX\_OUT;

wire [31:0] PC,NPC,PC\_TMP;

wire [31:0] PC\_IN;

wire [3:0] ALUC;//alu control

wire ZF,CF,NF,OF;

wire [31:0] ALU\_a,ALU\_b,ALU\_b\_TMP;

wire RF\_W; //regfiles write

wire [4:0] Rdc,Rsc,Rtc;

wire [31:0] Rs; //Rs

wire [31:0] RF\_WD;

assign sExt18 = {{14{inst[15]}},{inst[15:0],2'h0}};

assign sExt16 = {{16{inst[15]}},inst[15:0]};//

assign uExt16 = {16'h0,inst[15:0]};

assign pc = PC;

assign NPC = PC + 4;

assign PC\_TMP = MUX\_OUT[2]? sExt18 + NPC : NPC;

assign PC\_IN = MUX\_OUT[1]? Rs : (MUX\_OUT[0]? {PC[31:28],inst[25:0],2'b00} : PC\_TMP);

assign ALU\_a = MUX\_OUT[4]? PC : (MUX\_OUT[3]? {27'b0,inst[10:6]}: Rs);

assign ALU\_b\_TMP = (MUX\_OUT[5]||MUX\_OUT[6])? (MUX\_OUT[5]? sExt16 : uExt16): Rt;

assign ALU\_b = MUX\_OUT[4]? 32'h4 : ALU\_b\_TMP;//\*

assign Rsc = inst[25:21];

assign Rtc = inst[20:16];

assign Rdc = MUX\_OUT[8]? inst[20:16] :(MUX\_OUT[4]? 5'd31:inst[15:11]);

assign RF\_WD = MUX\_OUT[7]? dm\_data : alu\_r ;

pcreg pc\_uut(

.clk(clk), .reset(reset),.ena(1'b1),

.pc\_in(PC\_IN),.pc\_out(PC)

);

regfile cpu\_ref(

.clk(clk),.rst(reset),.ena(1'b1),.we(RF\_W),

.raddr1(Rsc),.rdata1(Rs),

.raddr2(Rtc),.rdata2(Rt),

.waddr(Rdc),.wdata(RF\_WD)

);

alu alu\_uut(

.a(ALU\_a),.b(ALU\_b),.aluc(ALUC),

.r(alu\_r),

.zero(ZF),.carry(CF),.negative(NF),.overflow(OF)

);

cu cu\_uut(

.inst(inst),.zf(ZF),.aluc(ALUC),

.mux(MUX\_OUT),.rf\_we(RF\_W),

.dm\_ena(dm\_ena),.dm\_w(dm\_w),.dm\_r(dm\_r)

);

endmodule

**3.3 控制器模块，实现对CPU中各控制信号的选择控制**

module cu(

input [31:0]inst,

output [8:0]mux,

input zf,

output [3:0] aluc,

output rf\_we,

output dm\_ena,

output dm\_w,

output dm\_r

);

wire add\_op, addu\_op, sub\_op, subu\_op, and\_op, or\_op, xor\_op, nor\_op;

wire slt\_op, sltu\_op, sll\_op, srl\_op, sra\_op, sllv\_op, srlv\_op, srav\_op, jr\_op;

assign add\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b100000)?1'b1:1'b0;

assign addu\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b100001)?1'b1:1'b0;

assign sub\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b100010)?1'b1:1'b0;

assign subu\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b100011)?1'b1:1'b0;

assign and\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b100100)?1'b1:1'b0;

assign or\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b100101)?1'b1:1'b0;

assign xor\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b100110)?1'b1:1'b0;

assign nor\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b100111)?1'b1:1'b0;

assign slt\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b101010)?1'b1:1'b0;

assign sltu\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b101011)?1'b1:1'b0;

assign sll\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b000000)?1'b1:1'b0;

assign srl\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b000010)?1'b1:1'b0;

assign sra\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b000011)?1'b1:1'b0;

assign sllv\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b000100)?1'b1:1'b0;

assign srlv\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b000110)?1'b1:1'b0;

assign srav\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b000111)?1'b1:1'b0;

assign jr\_op = (inst[31:26]==6'b000000&&inst[5:0]==6'b001000)?1'b1:1'b0;

wire addi\_op, addiu\_op, andi\_op, ori\_op, xori\_op;

wire lw\_op, sw\_op, beq\_op, bne\_op,slti\_op, sltiu\_op, lui\_op;

assign addi\_op = (inst[31:26]==6'b001000)?1'b1:1'b0;

assign addiu\_op = (inst[31:26]==6'b001001)?1'b1:1'b0;

assign andi\_op = (inst[31:26]==6'b001100)?1'b1:1'b0;

assign ori\_op = (inst[31:26]==6'b001101)?1'b1:1'b0;

assign xori\_op = (inst[31:26]==6'b001110)?1'b1:1'b0;

assign lw\_op = (inst[31:26]==6'b100011)?1'b1:1'b0;

assign sw\_op = (inst[31:26]==6'b101011)?1'b1:1'b0;

assign beq\_op = (inst[31:26]==6'b000100)?1'b1:1'b0;

assign bne\_op = (inst[31:26]==6'b000101)?1'b1:1'b0;

assign slti\_op = (inst[31:26]==6'b001010)?1'b1:1'b0;

assign sltiu\_op = (inst[31:26]==6'b001011)?1'b1:1'b0;

assign lui\_op = (inst[31:26]==6'b001111)?1'b1:1'b0;

wire j\_op, jal\_op;

assign j\_op = (inst[31:26]==6'b000010)?1'b1:1'b0;

assign jal\_op = (inst[31:26]==6'b000011)?1'b1:1'b0;

/\*------------------------------PC-----------------------------\*/

assign mux[0]=jal\_op||j\_op;// 1 is jal | j "||", 0 is others

assign mux[1]=jr\_op;//1 is jr

assign mux[2]=(beq\_op & zf)||( bne\_op & !zf);// 0 is pc+4, 1 is ext18 + npc

/\*------------------------------ALU-----------------------------\*/

assign aluc[3] = slt\_op || sltu\_op || sllv\_op || srlv\_op || srav\_op || sll\_op || srl\_op || sra\_op || slti\_op || sltiu\_op || lui\_op ;

assign aluc[2] = and\_op || or\_op || xor\_op || nor\_op || sllv\_op || srlv\_op || srav\_op || sll\_op || srl\_op || sra\_op || andi\_op || ori\_op || xori\_op ;

assign aluc[1] = add\_op || sub\_op || xor\_op || nor\_op || slt\_op || sltu\_op || sllv\_op || sll\_op || addi\_op || xori\_op || slti\_op || sltiu\_op ;

assign aluc[0] = sub\_op || subu\_op || or\_op || nor\_op || slt\_op || srlv\_op || srl\_op || ori\_op || slti\_op || beq\_op || bne\_op ;

//select aluc\_a;

assign mux[3]= sll\_op || srl\_op || sra\_op;// 1 is sll | srl | sra, 0 is others

assign mux[4]= jal\_op;//1 is jal ,0 is !(sll | srl | sra | jal)

//select aluc\_b;

assign mux[5]= addi\_op || addiu\_op || sw\_op || lw\_op ||slti\_op || sltiu\_op ;//1 is sExt16

//assign mux[5]= addi\_op || addiu\_op || lw\_op ||slti\_op || sltiu\_op ;//1 is sExt16

assign mux[6]= andi\_op || ori\_op || xori\_op || lui\_op;//1 is uExt16

//assign mux[6]= andi\_op || ori\_op || xori\_op || lui\_op || sw\_op;//1 is uExt16

/\*------------------------------RegFile-----------------------------\*/

assign mux[7]= lw\_op;//1 is lw ,0 is others

assign mux[8]= addi\_op || addiu\_op || lw\_op || slti\_op || sltiu\_op || andi\_op || ori\_op || xori\_op || lui\_op;

assign rf\_we = !(jr\_op || beq\_op || bne\_op|| sw\_op || j\_op );

/\*------------------------------DMEM-----------------------------\*/

assign dm\_ena = lw\_op || sw\_op;

assign dm\_w = sw\_op;

assign dm\_r = lw\_op;

endmodule

**3.4 数据寄存器模块**

module dmem(

input clk,

input dm\_ena,

input dm\_w, //write

input dm\_r, //read

input [9:0] addr,

input [31:0] dm\_in,

output [31:0] dm\_out

);

reg [31:0] dmem\_reg[1023:0];

assign dm\_out=dm\_ena? (dm\_r? dmem\_reg[addr]: 32'h00000000):32'hz;

always@(posedge clk or negedge dm\_ena)

if(dm\_ena && dm\_w)

dmem\_reg[addr]<=dm\_in;

else;

endmodule

**3.5 程序计数器模块**

module pcreg(

input clk,

input reset,

input ena,

input [31:0]pc\_in,

output [31:0]pc\_out

);

reg [31:0] pc\_reg;

always @(negedge clk or posedge reset)

if(reset)

pc\_reg <= 32'h00400000;

else

pc\_reg <= pc\_in;

assign pc\_out = ena? pc\_reg : 32'hz;

endmodule

**3.6寄存器模块**

module regfile(

input clk,

input rst,

input ena,

input we,

input [4:0] raddr1,

input [4:0] raddr2,

input [4:0] waddr,

input [31:0] wdata,

output [31:0] rdata1,

output [31:0] rdata2

);

reg [31:0]array\_reg[31:0];

integer i=0;

assign rdata1 = ena?array\_reg[raddr1]:32'bz;

assign rdata2 = ena?array\_reg[raddr2]:32'bz;

always @(posedge clk or posedge rst)

begin

if(rst==1'b1)

for(i=0;i<32;i=i+1)

array\_reg[i] <= 32'b0;

else if ((ena&we)&& (waddr != 0))

array\_reg[waddr] <= wdata;

else;

end

endmodule

**3.6顶层模块，调用cpu、指令寄存器、数据寄存器，实现哈佛结构**

module sccomp\_dataflow(

input clk,

input reset,

output [31:0] inst,

output [31:0] pc

);

wire [31:0] imm;

wire [31:0]im\_addr;

wire [31:0] Rt;

wire [31:0] alu\_r;

wire [31:0]dm\_addr;

wire dm\_ena;

wire dm\_w;

wire dm\_r;

wire [31:0] dm\_data;

assign inst = imm;

assign im\_addr = pc- 32'h00400000;

assign dm\_addr = alu\_r-32'h10010000;

//imem imem(im\_addr[12:2],imm);

pretest\_imem imem(1'b1,im\_addr[11:2],imm);

cpu sccpu (clk,reset,imm,dm\_data, //input

Rt,alu\_r,pc,dm\_ena,dm\_w,dm\_r); //output

dmem dmem\_uut (clk,dm\_ena,dm\_w,dm\_r,

dm\_addr[11:2],Rt,dm\_data);

endmodule

1. 测试模块建模

**4.1用mars中得到的指令初始化得到的指令存储器模块**

module pretest\_imem(

input im\_r, //read ena

input [9:0] addr,

output reg [31:0] data\_out

);

reg [31:0]mem [1023:0];

initial

begin

//文件名与路径可修改

$readmemh("C:/Users/1234/Desktop/test/random/random\_code",mem);

end

always @ (\*)

if(!im\_r)

data\_out <= 32'hz;

else

data\_out <= mem[addr];

endmodule

4.2 cpu测试模块

调用顶层module sccomp\_dataflow，将结果输入至文件

`timescale 1ns / 1ps

module cpu\_tb;

reg clk;

reg rst;

wire [31:0]inst;

wire [31:0]pc;

sccomp\_dataflow uut(clk,rst,inst,pc);

integer file\_output;

integer counter = 0;

initial begin

file\_output = $fopen("C:/Users/1234/Desktop/test/random/random\_my\_result.txt");

clk=0;

rst=1;

//wait 100ns for global reset to finish

#50

rst = 0;

end

always #3 clk=~clk;

always @(posedge clk)

begin

if(rst==0)

begin

if(counter == 500)

$fclose(file\_output);

else begin

counter=counter+1;

$fdisplay(file\_output, "pc: %h", pc);

$fdisplay(file\_output, "instr: %h", inst);

$fdisplay(file\_output,"regfile0: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[0]);

$fdisplay(file\_output,"regfile1: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[1]);

$fdisplay(file\_output,"regfile2: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[2]);

$fdisplay(file\_output,"regfile3: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[3]);

$fdisplay(file\_output,"regfile4: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[4]);

$fdisplay(file\_output,"regfile5: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[5]);

$fdisplay(file\_output,"regfile6: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[6]);

$fdisplay(file\_output,"regfile7: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[7]);

$fdisplay(file\_output,"regfile8: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[8]);

$fdisplay(file\_output,"regfile9: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[9]);

$fdisplay(file\_output,"regfile10: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[10]);

$fdisplay(file\_output,"regfile11: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[11]);

$fdisplay(file\_output,"regfile12: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[12]);

$fdisplay(file\_output,"regfile13: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[13]);

$fdisplay(file\_output,"regfile14: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[14]);

$fdisplay(file\_output,"regfile15: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[15]);

$fdisplay(file\_output,"regfile16: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[16]);

$fdisplay(file\_output,"regfile17: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[17]);

$fdisplay(file\_output,"regfile18: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[18]);

$fdisplay(file\_output,"regfile19: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[19]);

$fdisplay(file\_output,"regfile20: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[20]);

$fdisplay(file\_output,"regfile21: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[21]);

$fdisplay(file\_output,"regfile22: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[22]);

$fdisplay(file\_output,"regfile23: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[23]);

$fdisplay(file\_output,"regfile24: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[24]);

$fdisplay(file\_output,"regfile25: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[25]);

$fdisplay(file\_output,"regfile26: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[26]);

$fdisplay(file\_output,"regfile27: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[27]);

$fdisplay(file\_output,"regfile28: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[28]);

$fdisplay(file\_output,"regfile29: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[29]);

$fdisplay(file\_output,"regfile30: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[30]);

$fdisplay(file\_output,"regfile31: %h",cpu\_tb.uut.sccpu.cpu\_ref.array\_reg[31]);

// $fdisplay(file\_output,"we: %h",cpu\_tb.uut.sccpu.cpu\_ref.we);

// $fdisplay(file\_output,"wdata: %h",cpu\_tb.uut.sccpu.cpu\_ref.wdata);

// $fdisplay(file\_output,"waddr: %h",cpu\_tb.uut.sccpu.cpu\_ref.waddr);

end

end

end

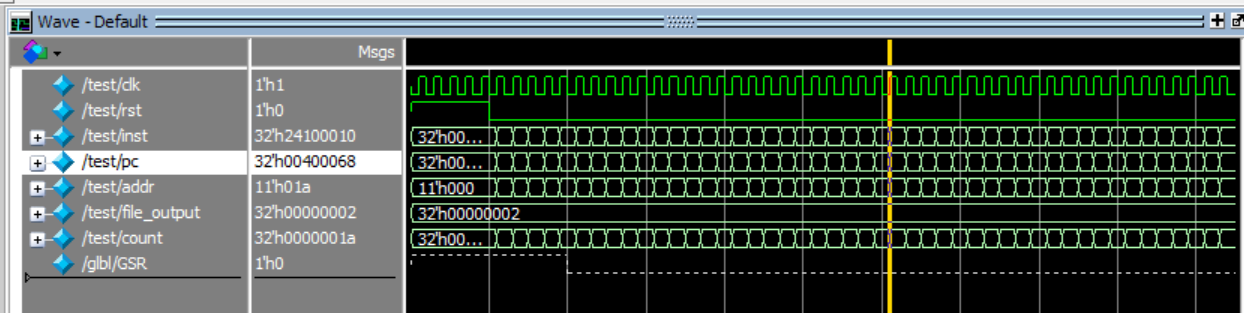
endmodule

1. 实验结果

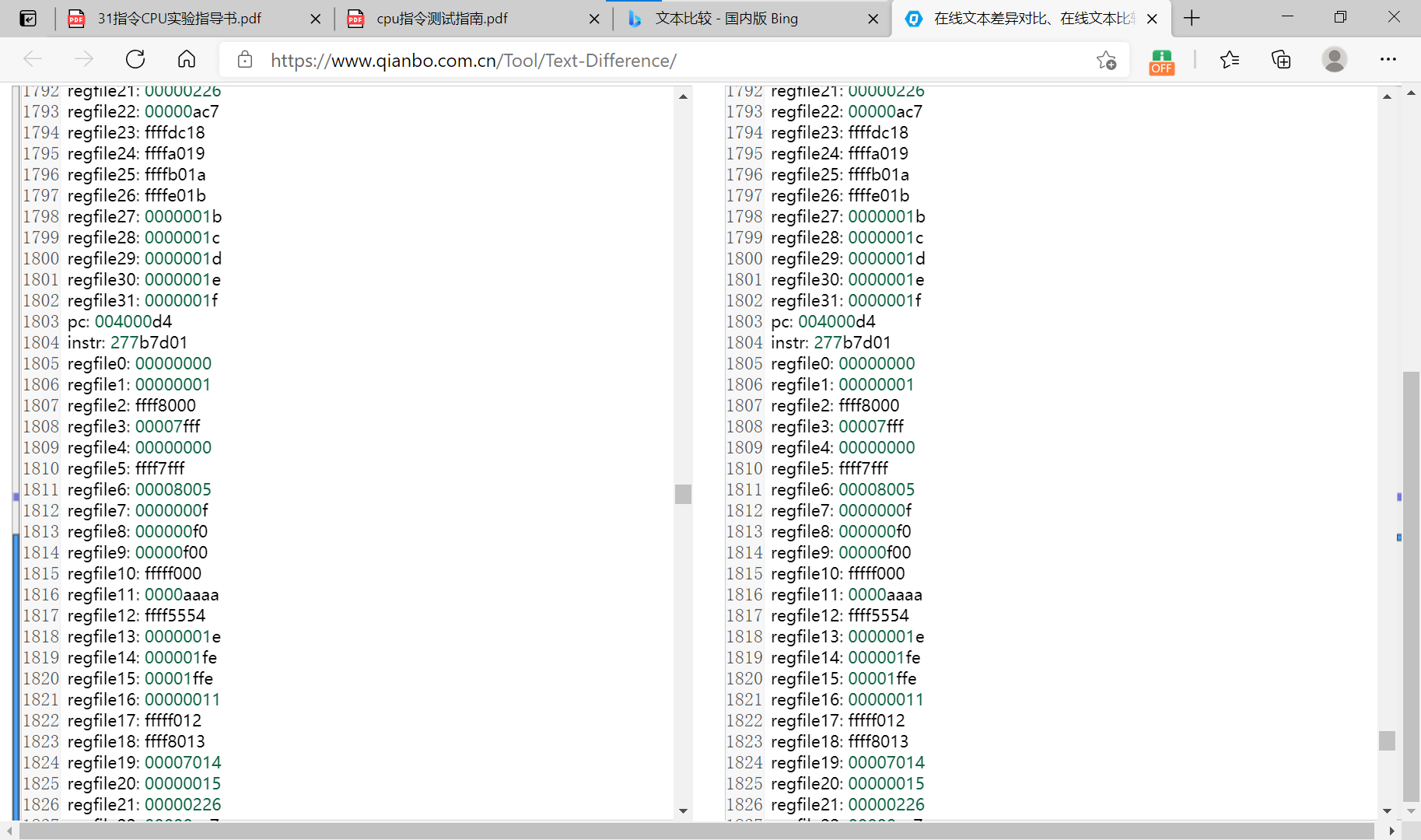
1.前仿真测试

与Mars4\_5.jr生成的result.txt文件进行一一比对，31个文件不超出标准输出的部分文本对比无误。

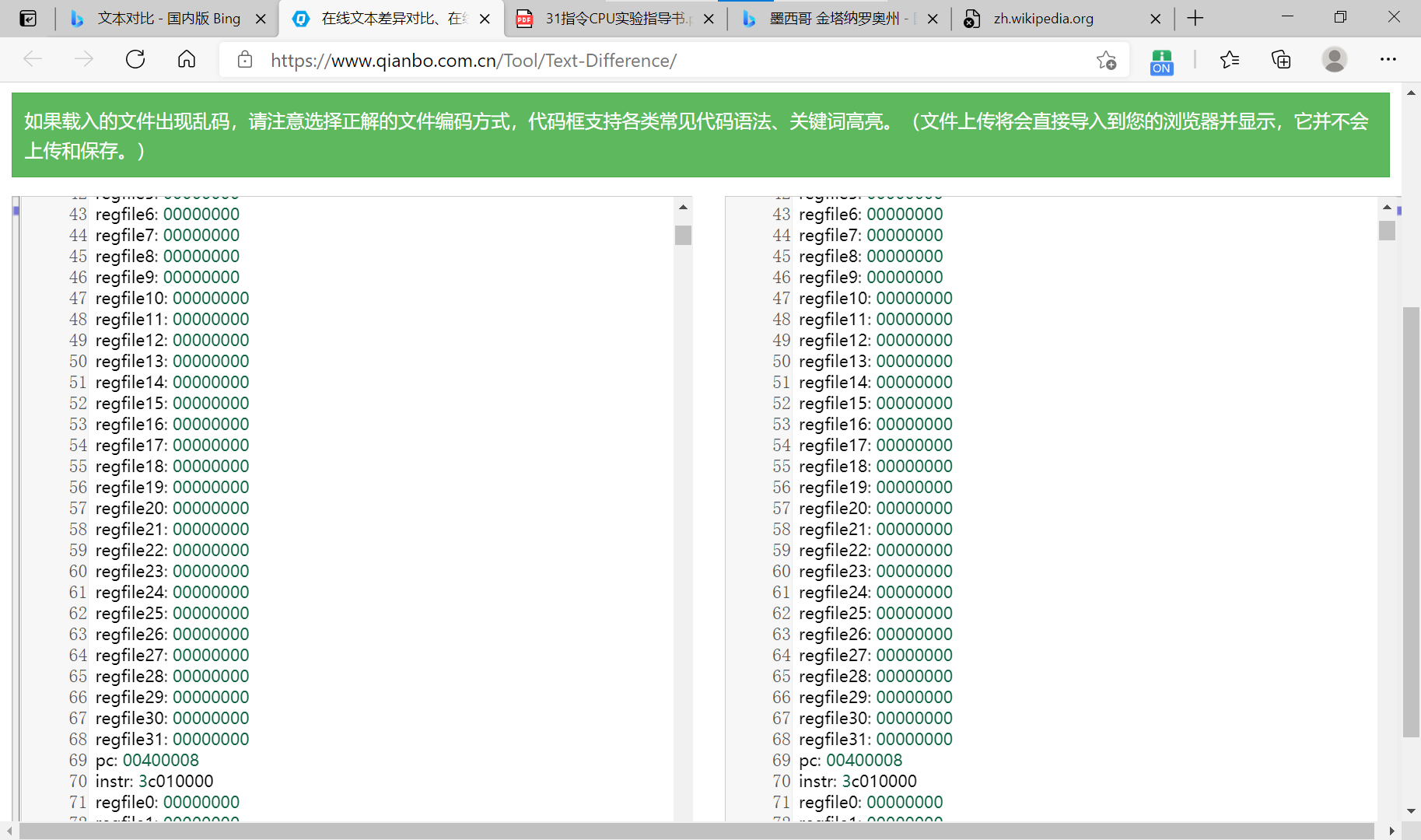
以addiu为例。modelsim下针对\_1\_addi.txt转换得到的16进制文件的实验结果：



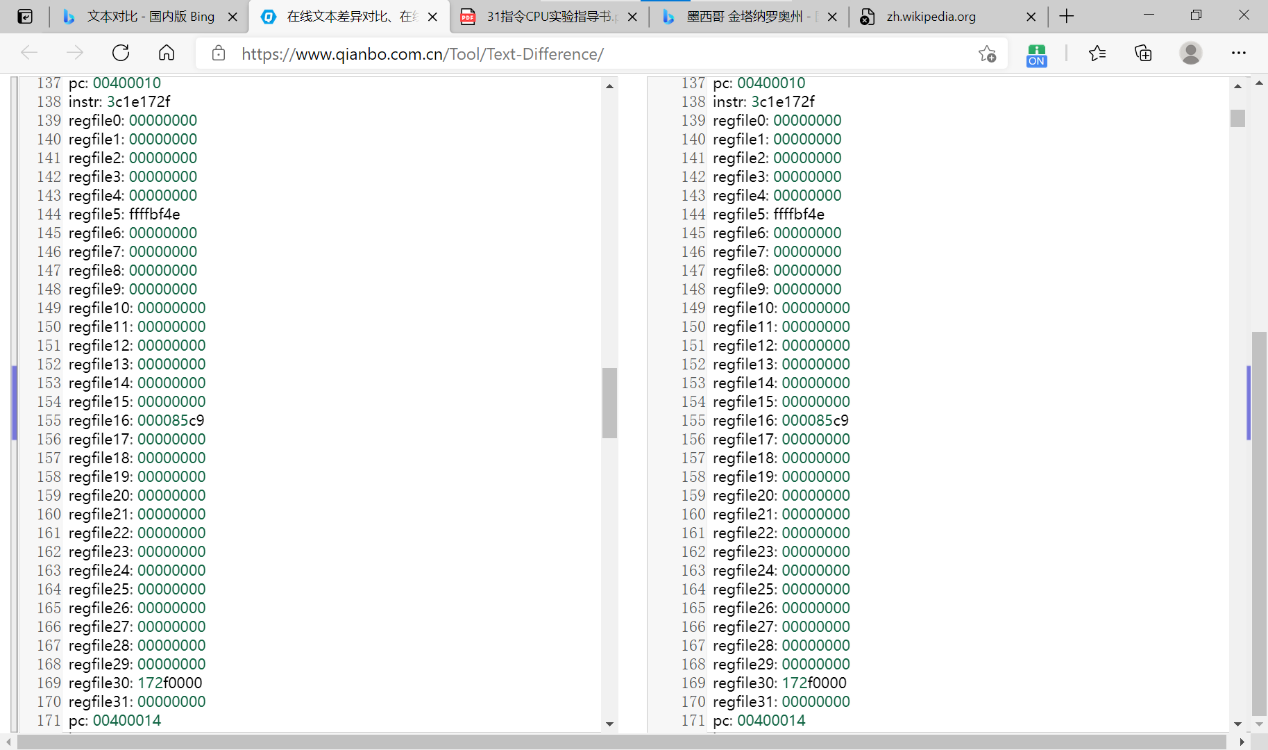
不超出标准输出的部分文本对比无误



新建了测试边界的汇编程序，以addi的边界为例，实验所设计的cpu与Mars4\_5.jr生成的result.txt结果一致。

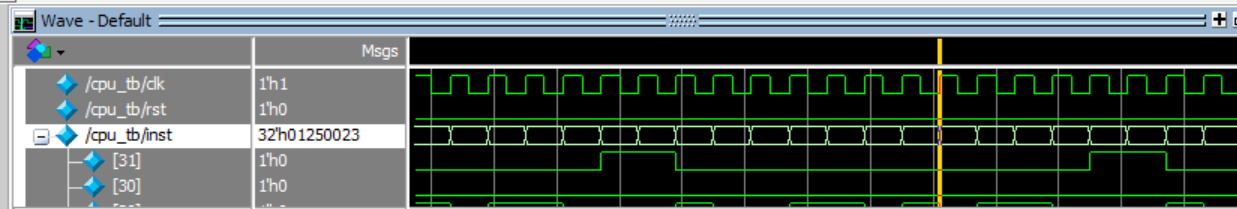


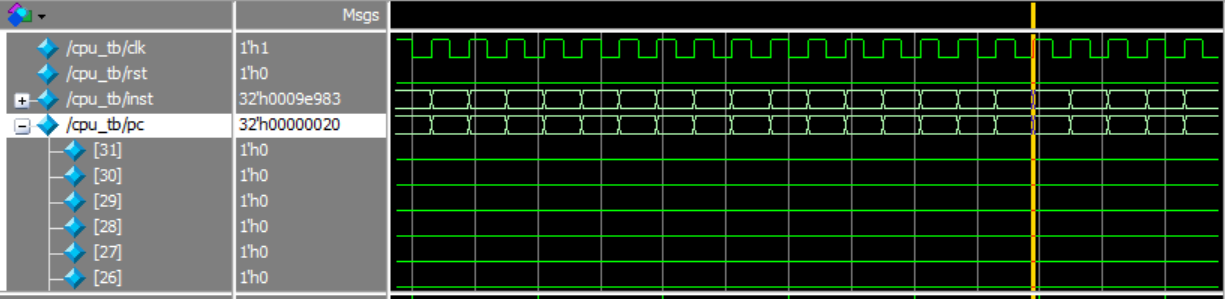
我自己编了几组数据进行测试，运行结果均正确，cpu能够通过随机数据测试。



但因网站测试需要在时序上略有调整

2.后仿真测试





结果通过

3.下板

显示pc结果：

结果显示，下板成功