计算机组织与系统结构实习报告 Lab 2.2

学号：1600012830

姓名：盛朱恒

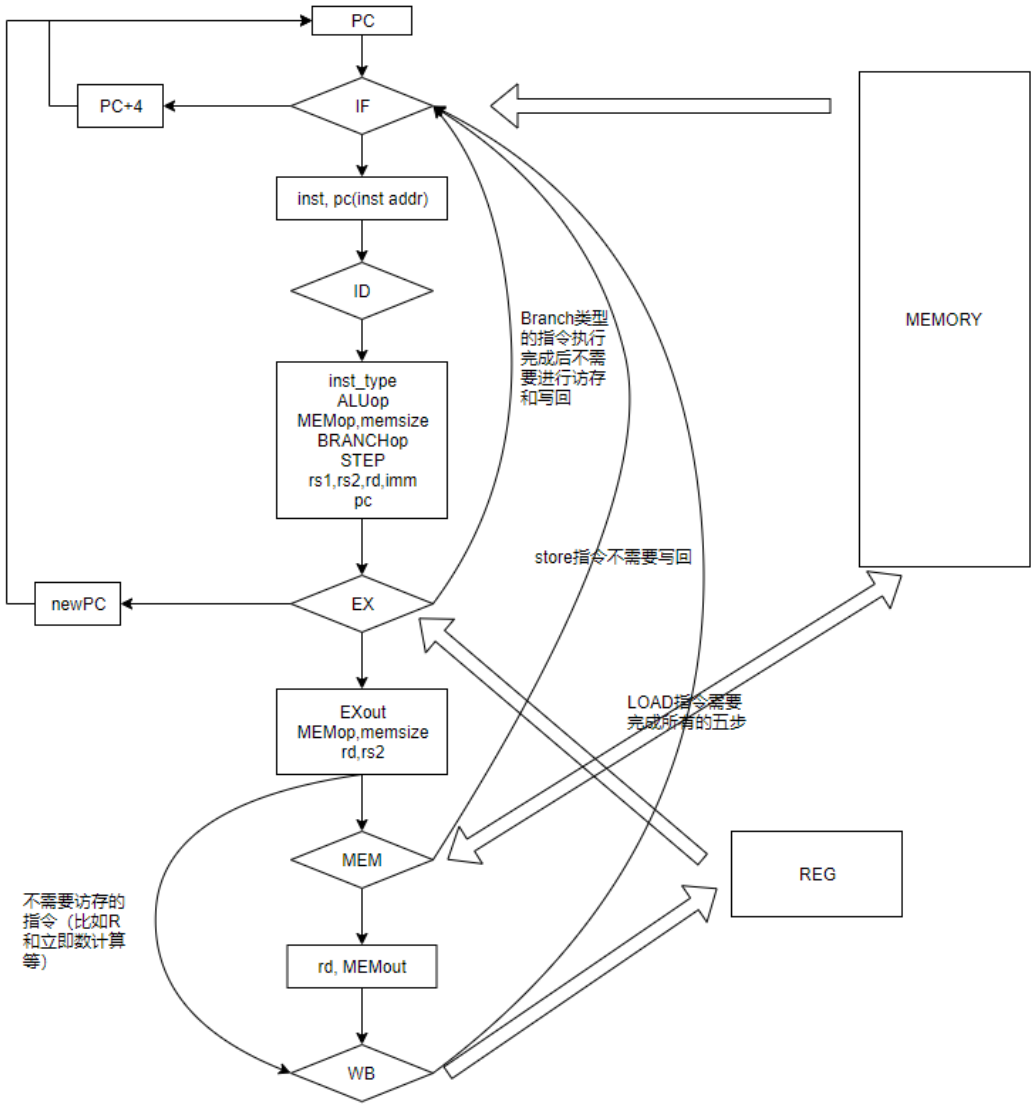
Part I：RISC-V 多周期模拟器 （50分）

1. 基于实现的RISC-V ISA，给出指令各阶段的寄存器传输级描述 （10分）。每类指令举1-2个例子即可。示例如下：

```
总分为5个阶段：IF取指令,ID译码,EX执行（包括修改PC,计算得到地址，计算得到结果）,MEM访问内存,WB写回寄存器rd
    add rd, rs1, rs2
    IF:inst=mem[PC], inst_addr=PC, PC=PC+4
    ID:opcode=0x33, func3=0x0, func7=0x00, rd=inst[11:7], rs1=inst[19:15], rs2=inst[24:20]
    --> aluop = ADD, inst_type = R, step = IF\ID\EX\WB, rd,rs1,rs2
    EX:EXout = reg[rs1] + reg[rs2]
    WB:reg[rd] = out;

    addw rd, rs1, rs2
    IF:inst= mem[PC], inst_addr = PC, PC=PC+4
    ID:opcode=0x3B, func3=0x0, func7=0x00, rd=inst[11:7], rs1=inst[19:15], rs2=inst[24:20]
    -->aluop = ADD, inst_type = RW, step = IF\ID\EX\WB, rd,rs1,rs2
    EX:EXout = reg[rs1] + reg[rs2]
    WB:reg[rd] = out;

    lw rd, imm(rs1)
    IF:inst= mem[PC], inst_addr = PC, PC=PC+4
    ID:opcode=0x03, func3=0x2, rd=inst[11:7], rs1=inst[19:15], imm=inst[31:20]
    -->memop=LOAD, memsize = 4, inst_type = LW, step = IF\ID\EX\MEM\WB, rd,rs1,imm
    EX:out(address) = reg[rs1] + imm
    MEM: out = memory[out]~memmory[out+3]
    WB:reg[rd] = out;

    addi rd, rs1, imm
    IF:inst= mem[PC], inst_addr = PC, PC=PC+4
    ID:opcode=0x13, func3=0x0, rd=inst[11:7], rs1=inst[19:15], imm=inst[31:20]
    -->aluop = ADD, inst_type = I, step = IF\ID\EX\WB, rd,rs1,imm
    EX:out = reg[rs1] + imm
    WB:reg[rd] = out;

    jalr rd, rs1, imm
    IF:inst= mem[PC], inst_addr = PC, PC=PC+4
    ID:opcode=0x67, func3=0x0, rd=inst[11:7], rs1=inst[19:15], imm=inst[31:20]
    -->inst_type = JALR, step = IF\ID\EX\WB, rd,rs1,imm
    EX:out = inst_addr + 4, PC = reg[rs1] +imm
```

```
    WB:reg[rd] = out;

    sw rs2, imm(rs1)
    IF:inst= mem[PC], inst_addr = PC, PC=PC+4
    ID:opcode=0x23, func3=0x2, rs1=inst[19:15], rs2=inst[24:20], imm =
inst[31:25]inst[11,7]
    -->memop=STORE, memsize = 4, inst_type = SW, step = IF\ID\EX\MEM,
rs2,rs1,imm
    EX:out = reg[rs1] + imm
    MEM:memory[out] = reg[rs2]

    beq rs1, rs2, imm
    IF:inst= mem[PC], inst_addr = PC, PC=PC+4
    ID:opcode=0x63, func3=0x0, rs1=inst[19:15], rs2=inst[24:20], imm =
inst[31]inst[7]inst[30:25]inst[11:8]
    -->inst_type = Branch, Bran = BEQ, step = IF\ID\EX, rs1, rs2
    EX:if(rsg[rs1]==reg[rs2]) PC = inst_addr + (imm << 1)

    auipc rd, imm
    IF:inst= mem[PC], inst_addr = PC, PC=PC+4
    ID:opcode=0x17,rd = inst[11:7], imm = inst[31:12]
    -->inst_type = AUIPC, step = IF\ID\EX\WB, rd,imm
    EX:out = inst_addr + (imm<<12)
    WB:reg[rd] = out;

    jal rd, imm
    IF:inst= mem[PC], inst_addr = PC, PC=PC+4
    ID:opcode=0x6F,rd = inst[11:7], imm =
inst[31]inst[19:12]inst[20]inst[30:21]
    -->inst_type = JAL, step = IF\ID\EX\WB, rd,imm
    EX:out = inst_addr + 4, PC = PC + (imm<<1)
    WB:reg[rd] = out;
```

2. 基于以上分析，给出多周期处理器的数据通路图和控制信号产生逻辑。不限形式，手绘也可。（10分）



控制信号逻辑：

1、PC输入IF，访问内存得到inst，记录下pc(inst addr),将PC+4

2、inst输入ID，得到inst_type, ALUop, MEMop,memsize,Branchop,STEP(用来表示需要进行哪几个步骤),寄存器rs1,rs2,rd，立即数（imm）；pc直接向后传递。（有些信号在一些指令下属于无关信号）

3、上述信号传入EX，MEMop,memsize,rd,rs2直接向后传递。BRANCH和JUMP指令直接更改PC=newPC；load/store指令中，EXout为address，写回类型得指令中，EXout为要写入reg[rd]得结果。

BRANCH指令EX后结束，LOAD/STORE指令进入MEM，其他指令进入WB。

4、进入MEM，memop是LOAD，rd直接向后传递，MEMout为memory[address]；是STORE，将reg[rs2]写入memory[address]

5、进入WB，根据不同情况，将MEMout或者EXout写入reg[rd]。

3. 运行测试程序，给出动态执行的指令数。（共5个定点程序，每个2分）

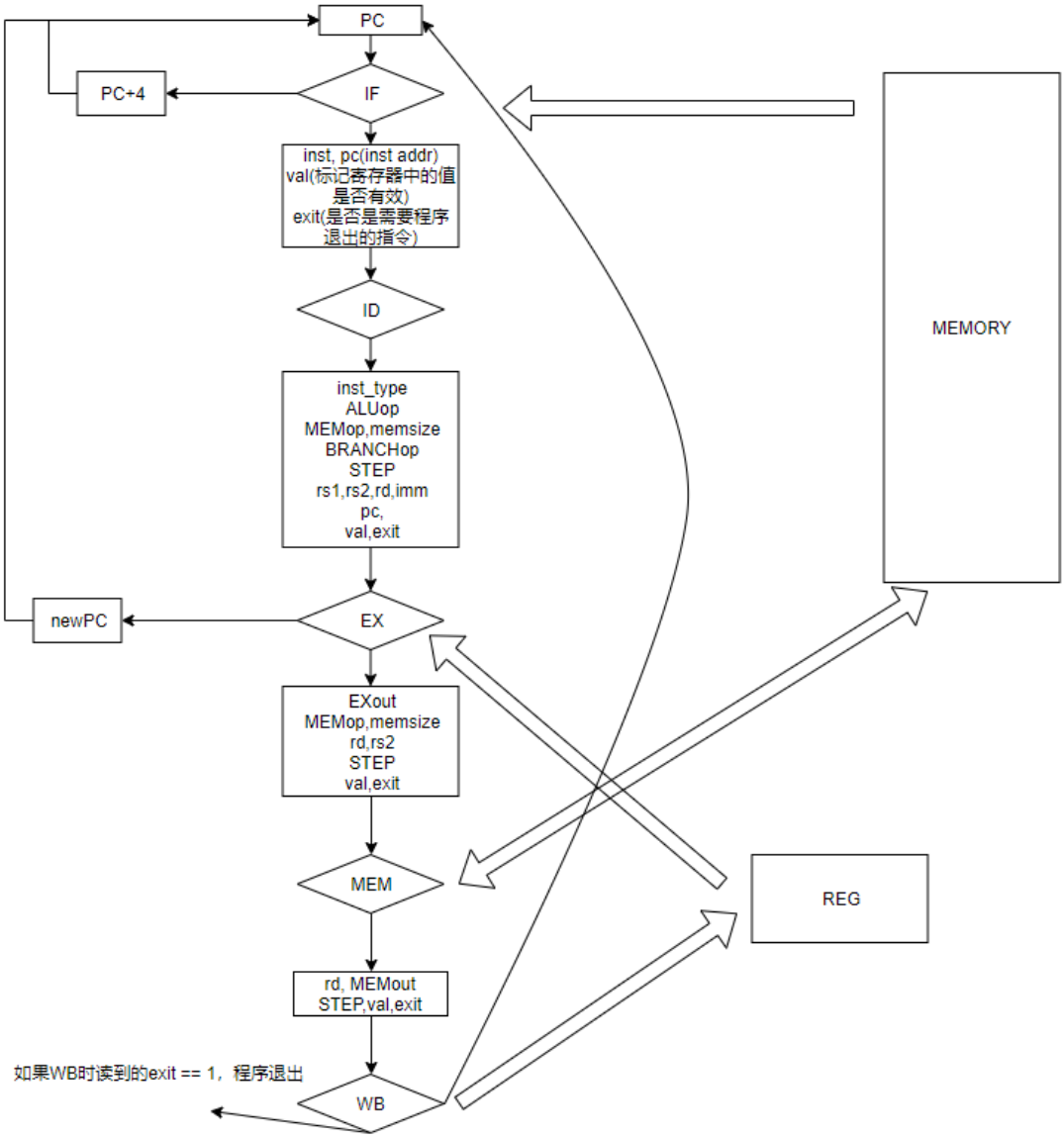
4. 运行测试程序，给出多周期处理器的执行周期数，并计算平均CPI。（共5个定点程序，每个4分）

（这一部分运行的是发布的lab文件中的可执行文件，因为发现目前的编译器重新编译代码后，指令数不一样，应该是由于编译器版本不同，所以以lab发布的可执行文件为准。后面的测试也是如此。）

	test1	test2	test3	test4	test5	test6	test7	test8	test9	test10
insts	28	28	117	162	118	68	132	60	33	23
cycles	122	122	494	689	693	483	559	251	182	140
CPI	4.357	4.357	4.222	4.253	5.872	7.102	4.234	4.183	5.515	6.086

Part II：RISC-V 流水线模拟器（50分）

1. 基于实现的RISC-V ISA，给定流水线处理器的阶段划分，并简单介绍各阶段的工作。（5分）
   流水线阶段划分为5个阶段：
   IF取指令：取得指令，PC=PC+4
   ID译码：根据指令得到控制信号
   EX执行：主要功能，得出计算结果、更改PC
   MEM访存：读写内存
   WB写回：将计算结果或者根据计算结果得到得内存数据写入reg[rd]
2. 给出流水线处理器的数据通路图和控制信号产生逻辑。不限形式，手绘也可。（5分）



信号产生逻辑与多周期中的大部分相同，不同的有：

STEP（标记这个指令需要在哪些步骤中“真的”执行，而不是什么都不做）需要在寄存器中传递下去。

新加入了val，用于表示寄存器中的值是否有效（用于启动和bubble后的处理），启动时，第一个周期只有IF需要工作，bubble后可能会有几个部分不需要工作。

新加入了exit用于执行退出，由于最后一条指令会被提前读入，因此只有在结束的指令运行到WB()阶段时，才能真正的退出。

3. 请简要描述该流水线中会产生的各种冒险，每类均需举例说明。（10分）

数据冒险：一条指令的对寄存器修改在WB阶段才会写入寄存器，因此如果存在“写-读”的情况，就会有数据冒险。

```
    add x1, x2, x3  IF ID EX MEM WB
    add x2, x1, x3     IF ID ()  () EX MEM WB
    add x5, x4, x3        IF ()  () ID EX  MEM WB
```

第一条指令对x1的修改需要在WB后才会真正写入寄存器，因此后续需要读x1的指令都需要将执行阶段延后到WB完成后。

控制冒险：跳转指令和分支指令都需要在EX阶段后才知道下一条指令的真正地址，而之前的预测都是PC+4，如果预测错误，需要将已经预测执行的指令删除。

```
                              |(此时发现预测错误)
    beq rs1, rs2, off IF ID EX MEM WB
    add x1, x2, x3       IF ID-----------(X)
    add x4, x5, x6          IF-----------(X)
    add x7, x8, x9             IF ID EX MEM WB
```

jump指令与之类似

分析数据冒险+控制冒险的情况会不会出现：

jalr需要读rs1，如果rs1存在数据冒险，那么jalr的EX阶段会推迟，那么就不存在控制冒险。

```
    add x1, x2, x3  IF ID EX MEM WB
    jalr x2, x1,imm    IF ID ()  () EX MEM WB
    add x5, x4, x3        IF ()  () ID -----------(x)
    add x6, x7, x8                   IF -----------(x)
```

jal和jalr都需要写rd,如果后续指令需要读rd,那也是在EX阶段，但是后续的两条指令都不可能到EX阶段，所以不存在这种情况。

分析是否有memory的数据冒险，对数据memory的读写只有在MEM阶段，这一阶段不可能同时有两个出现，因此不会有memory的数据冒险。

4. 运行测试程序，给出流水线处理器的执行周期数，并计算平均CPI。（共5个测试程序，每个4分）

这里的周期数采用的算法时，每个时期，IF，ID，EX，MEM，WB都会工作，取工作的最长周期作为这一阶段的执行周期。

5. 请对该流水线处理器中因不同类型的冒险而发生的停顿进行统计，并打印数据和分析。（共5个测试程序，每个2分）

这里统计了数据冒险和结构冒险。

控制冒险统计的是控制冒险发生的次数。

数据冒险由于可能bubble两个周期，也可能bubble一个周期，因此统计的是bubble的周期数，比如：

```
    add x1, x2, x3  IF ID EX MEM WB
    add x2, x1, x3     IF ID ()  () EX MEM WB
    add x5, x4, x3        IF ()  () ID EX  MEM WB
```

这种情况下记为2次数据冒险。

	test1	test2	test3	test4	test5	test6	test7	test8	test9	test10
insts	28	28	117	162	118	68	132	60	33	23
cycles	62	62	243	338	251	151	278	146	66	50
CPI	2.214	2.214	2.076	2.086	2.127	2.220	2.106	2.433	2	2.173
数据	31	31	111	161	113	63	131	83	25	23
控制	0	0	6	6	6	6	6	0	2	0

Part III：其他加分

如有任何加分项目，请在此说明，并给出运行结果。

在多周期和流水线中都完成了系统调用的实现，可以实现输出字符、字符串、整数，输入整数，time函数，退出函数。

图中中计算的ack(3,3)，500次循环，用时5s。