实验报告

智能 1601　　陈一嘉　　201608010125

一、实验内容

设计一个执行 RISC-V 的基本整数指令集 RV32I 的单周期 CPU

二、实验要求

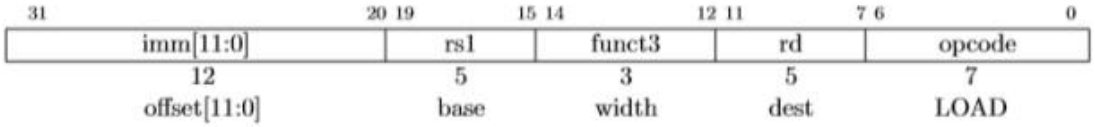
1、硬件设计采用 VHDL 或 Verilog 语言，软件设计采用 C/C++或 SystemC 语言，其它语言例如 Chisel、MyHDL 等也可选。

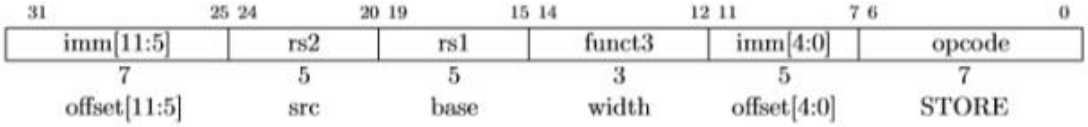
2、实验报告采用 markdown 语言，或者直接上传 PDF 文档

3、实验最终提交所有代码和文档

三、CPU 指令集

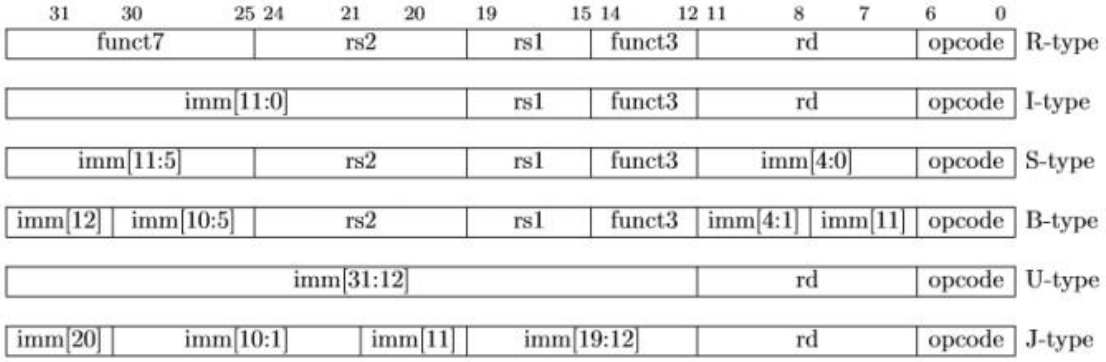
Instruction	Constraints	Code Points	Purpose
LUI	*rd*=x0	2^{20}	*Reserved for future standard use*
AUIPC	*rd*=x0	2^{20}	
ADDI	*rd*=x0, and either *rs1*≠x0 or *imm*≠0	$2^{17}-1$	
ANDI	*rd*=x0	2^{17}	
ORI	*rd*=x0	2^{17}	
XORI	*rd*=x0	2^{17}	
ADDIW	*rd*=x0	2^{17}	
ADD	*rd*=x0	2^{10}	
SUB	*rd*=x0	2^{10}	
AND	*rd*=x0	2^{10}	
OR	*rd*=x0	2^{10}	
XOR	*rd*=x0	2^{10}	
SLL	*rd*=x0	2^{10}	
SRL	*rd*=x0	2^{10}	
SRA	*rd*=x0	2^{10}	
ADDW	*rd*=x0	2^{10}	
SUBW	*rd*=x0	2^{10}	
SLLW	*rd*=x0	2^{10}	
SRLW	*rd*=x0	2^{10}	
SRAW	*rd*=x0	2^{10}	
FENCE	*pred*=0 or *succ*=0	$2^{5}-1$	
SLTI	*rd*=x0	2^{17}	*Reserved for custom use*
SLTIU	*rd*=x0	2^{17}	
SLLI	*rd*=x0	2^{11}	
SRLI	*rd*=x0	2^{11}	
SRAI	*rd*=x0	2^{11}	
SLLIW	*rd*=x0	2^{10}	
SRLIW	*rd*=x0	2^{10}	
SRAIW	*rd*=x0	2^{10}	
SLT	*rd*=x0	2^{10}	





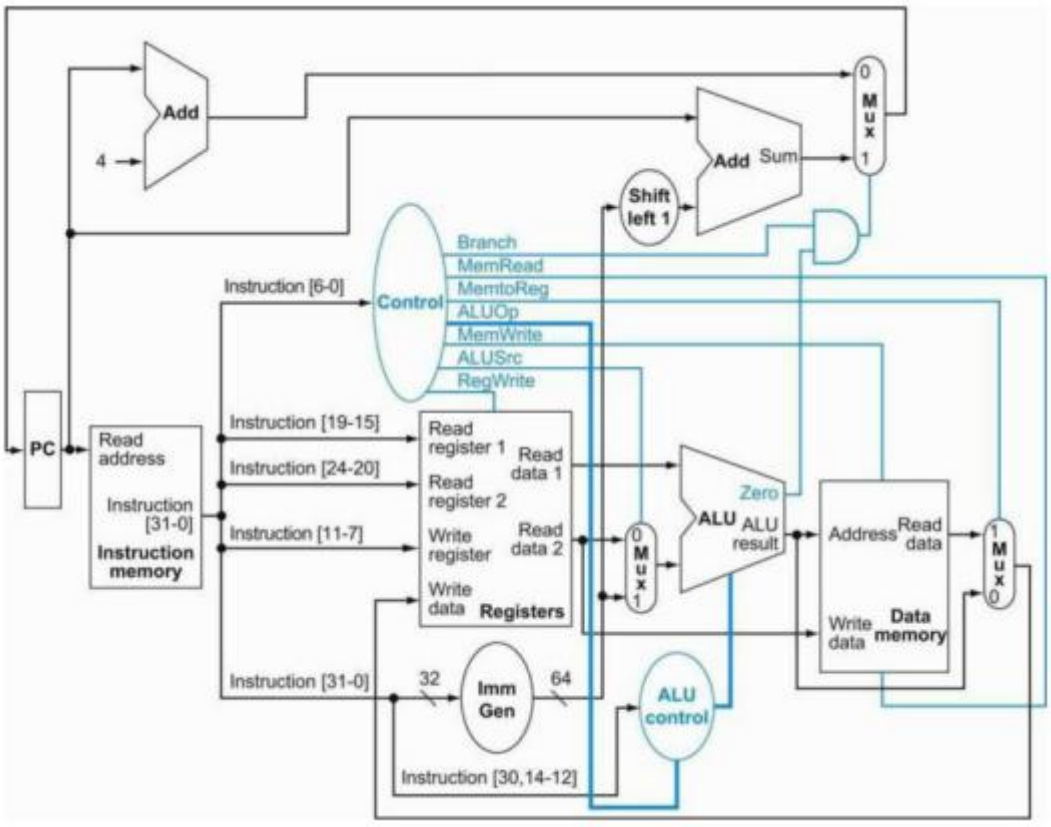
Register operand				
Instruction	rd	rs1	read CSR?	write CSR?
CSRRW	x0	-	no	yes
CSRRW	!x0	-	yes	yes
CSRRS/C	-	x0	yes	no
CSRRS/C	-	!x0	yes	yes
Immediate operand				
Instruction	rd	uimm	read CSR?	write CSR?
CSRRWI	x0	-	no	yes
CSRRWI	!x0	-	yes	yes
CSRRS/CI	-	0	yes	no
CSRRS/CI	-	!0	yes	yes

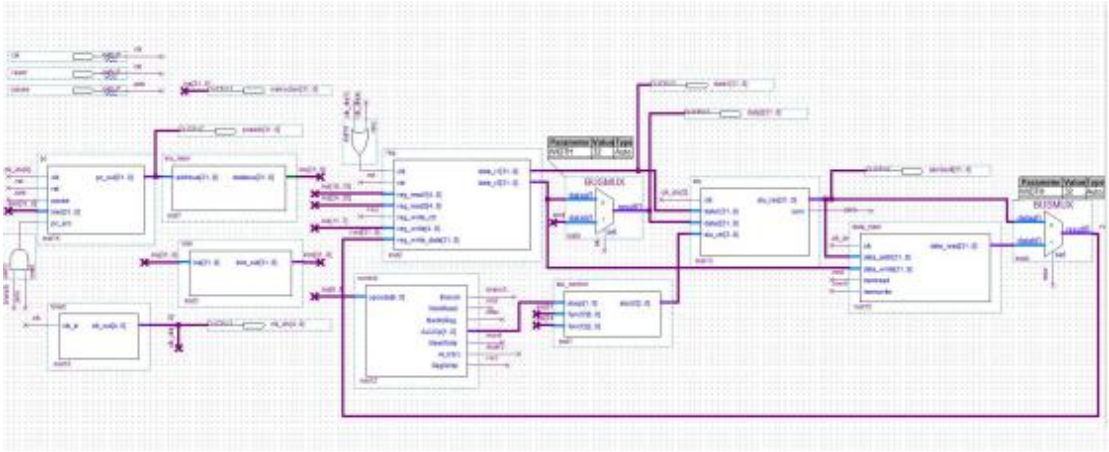
指令类型：



四、实验过程

参考手册给出的 datapath，整体框架如下：



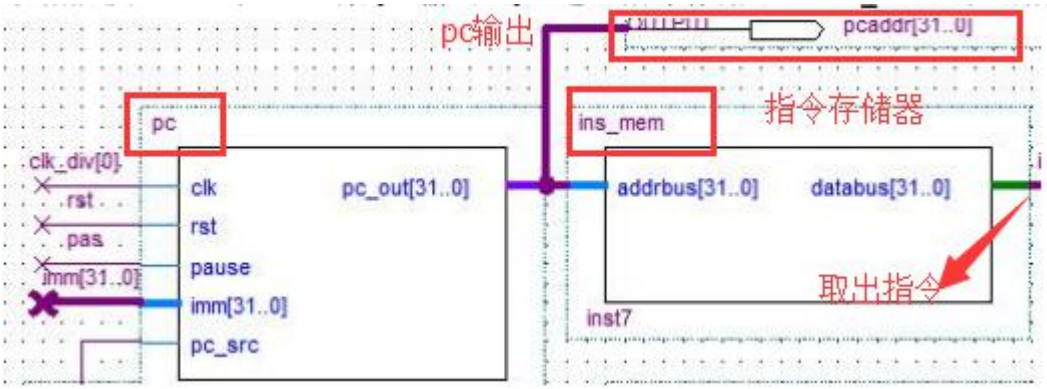


手册中说了流水线 CPU 有 5 个步骤：**取指令-译码-执行-访存-写回，**是将 clk 分为了 5 分时来循环执行这 5 个步骤的：

```
entity time5 is
port(
        clk_in:in std_logic;
        clk_out:out std_logic_vector(4 downto 0)
        );
end time5;

architecture bhv of time5 is
signal temp:std_logic_vector(4 downto 0):="10000";
begin
        t0:process(clk_in)
        begin
                if(rising_edge(clk_in)) then
                        if temp="10000" then
                                temp<="00001";
                        else
                                temp <= temp(3 downto 0) & '0';
                        end if;
                end if;
        end process;
        clk_out <= temp;
end ;
```

（1）**取指 clk1**：产生一条 pc 输出，pc 进入指令存储器（ins_mem）取出指令。

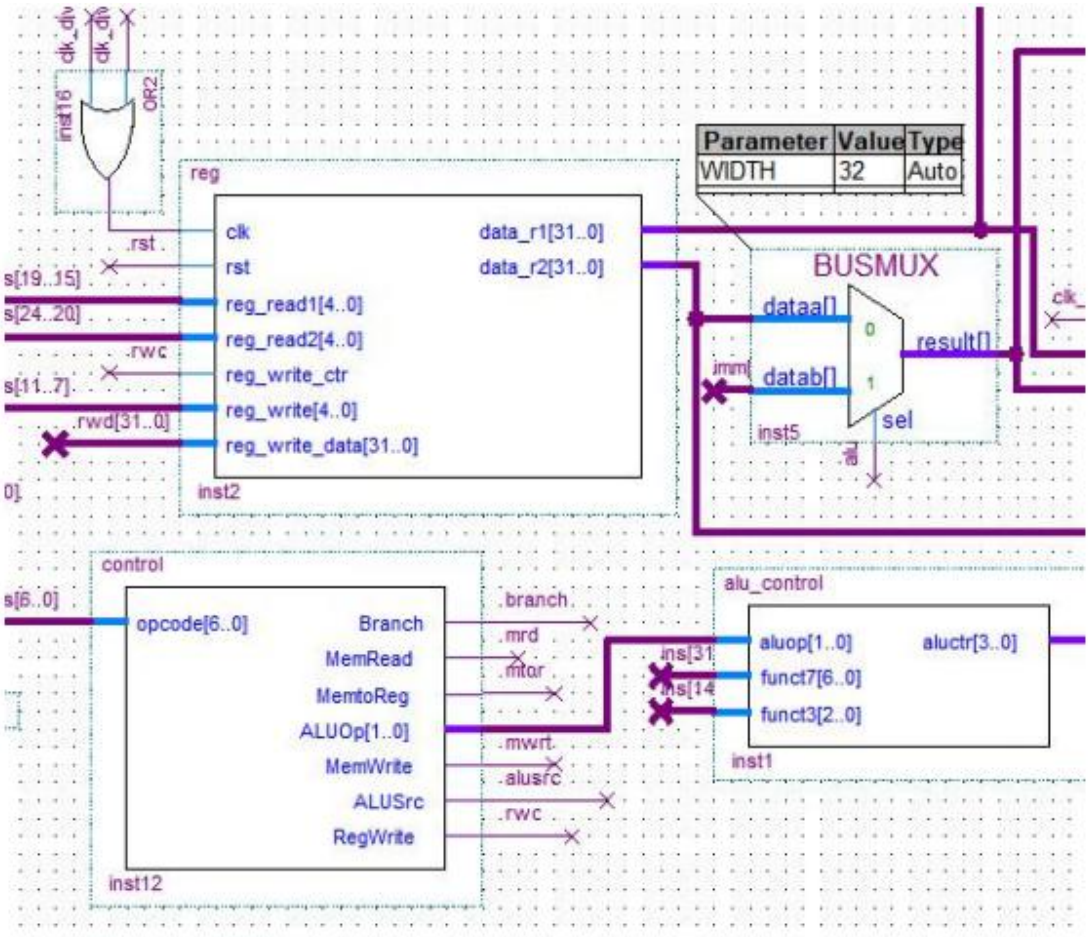


（2）译码 clk2:

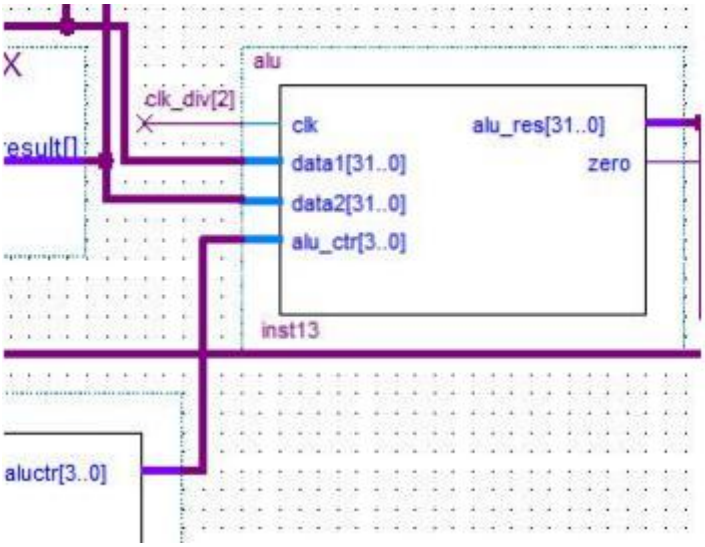
指令取出来后，指令进入 2 个部分：

① imm（立即数产生器）根据指令类型生成相应立即数。

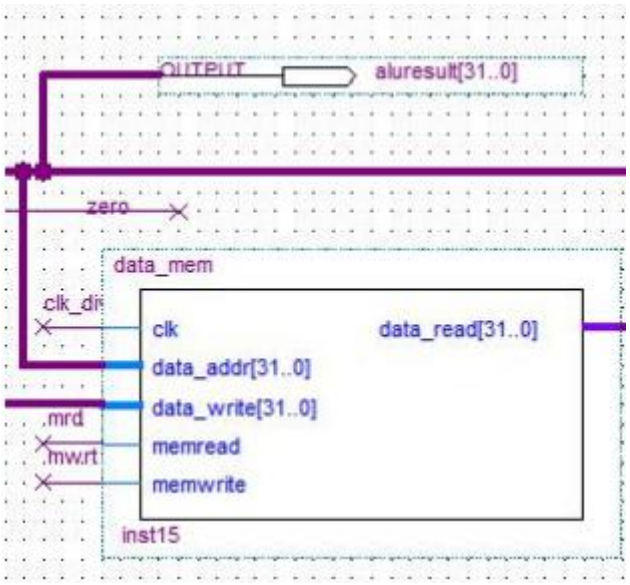
② 同时指令还进入 control（控制器）产生控制信号。



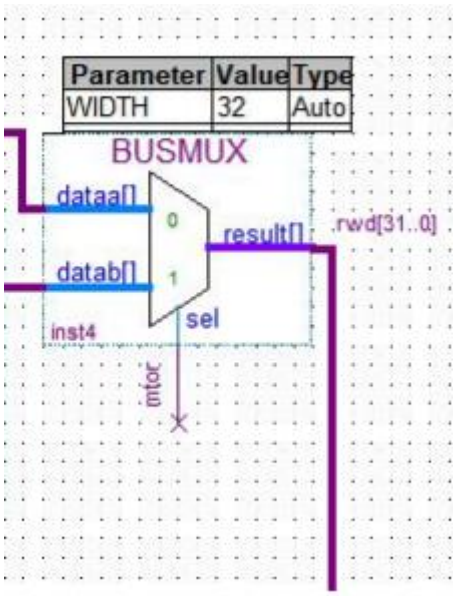
（3）**执行 clk3：** 上升沿产生 alu 结果



（4）访存：



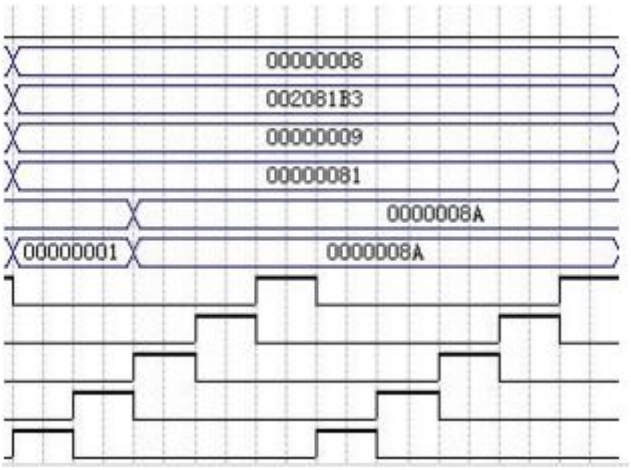
（5）写回：



五、测试样例

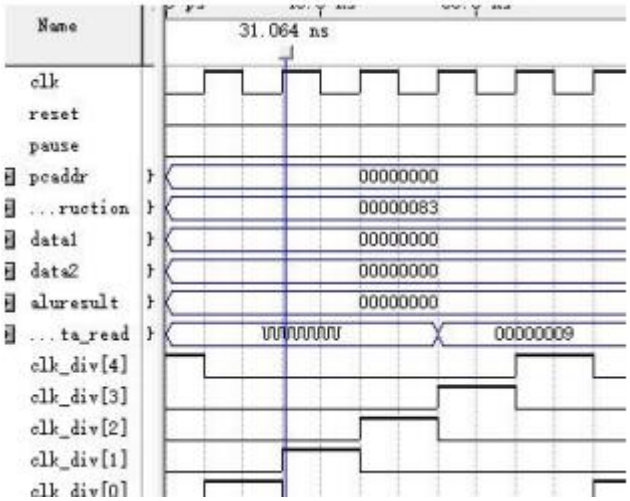
- ADD 指令

0000000 00010 00001 000 00011 0110011　　rs1 与 rs2 相加，存到寄存器 x3



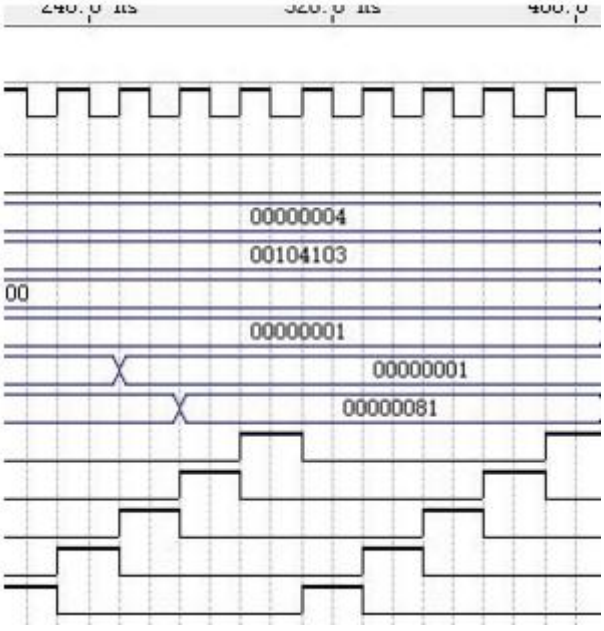
- LB 指令

000000000000 00000 000 00001 0000011



- LBU

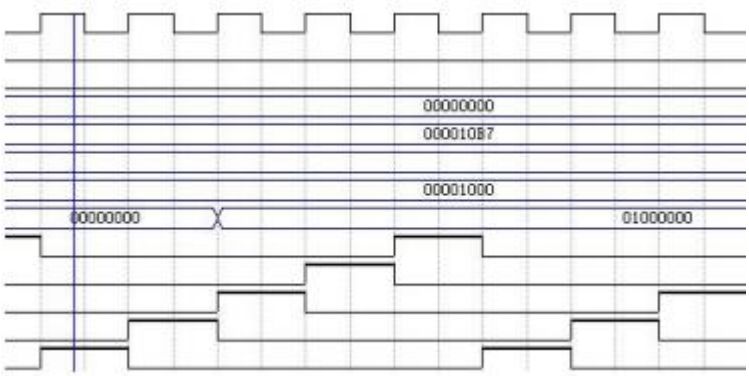
000000000001 00000 100 00010 0000011



- LUI x1，0x1

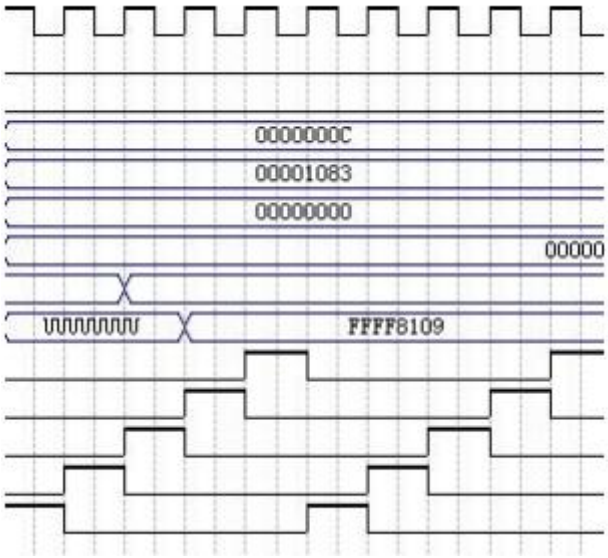
00000000 00000000 0001 00001 0110111

r1=00000000 0000000 00010000 00000000



- LH 16bit signed

000000000000 00000 001 00001 0000011



六、心得体会

感觉挺有难度的，一开始对于指令非常不熟悉，看手册也像看天书一样，然后就慢慢读，慢慢积累，逐渐好了一些。然后回想大二设计 CPU 时的过程，几个模块大致分了一下，着手每个模块每个模块的写。然后过程中贯穿“一切以指令出发”。也遇到了一些问题，有找同学帮忙解决，也有自己思考。最后总算勉强完成，但是没有对每条指令都测试。希望有时间能在后面对整个过程再完善一下。