**一、 实验目标：**

设计完成一个连续取指令并进行指令译码的电路，从而掌握设计简单数据通路的基本方法。

**二、实验内容**

本实验分成三周（三次）完成：1）首先完成一个译码器；2）接着实现一个寄存器文件；3）最后添加指令存储器和地址部件等将这些部件组合成一个数据通路原型。

**三、实验环境**

硬件：桌面PC

软件：Linux Chisel开发环境

**四、****实验步骤及说明**

本次试验分为三个部分：

1. 设计译码电路，输入位32bit的一个机器字，按照课本MIPS 指令格式，完成add、sub、lw、sw指令译码，其他指令一律译码成nop指令。输入信号名为Instr\_word，对上述四条指令义译码输出信号名为add\_op、sub\_op、lw\_op和sw\_op，其余指令一律译码为nop；

给出Chisel设计代码和仿真测试波形，观察输入Instr\_word为add R1,R2,R3; sub R0,R5,R6，lw R5,100(R2), sw R5,104(R2)、JAL RA,100(R2)时，对应的输出波形

Instr\_decoder

Instr\_word[31:0]

add\_op

sub\_op

lw\_op

sw\_op

nop

1. 设计寄存器文件，共32个32bit寄存器，允许两读一写，且0号寄存器固定读出位0。四个输入信号为RS1、RS2、WB\_data、Reg\_WB，寄存器输出RS1\_out和RS2\_out；寄存器内部保存的初始数值等同于寄存器编号

给出Chisel设计代码和仿真测试波形，观察RS1=5,RS2=8，WB\_data=0x1234,Reg\_WB=1的输出波形和受影响寄存器的值。

Reg\_file

RS1[4:0]

RS2[4:0]

WB\_data[31:0]

Reg\_WB

RS1\_out[31:0]

RS2\_out[31:0]

1. 实现一个32个字的指令存储器，从0地址分别存储4条指令add R1,R2,R3; sub R0,R5,R6，lw R5,100(R2), sw R5,104(R2)。然后组合指令存储器、寄存器文件、译码电路，并结合PC更新电路（PC初值为0）、WB\_data和Reg\_WB信号产生电路，最终让电路能逐条指令取出、译码（不需要完成指令执行）。

给出Chisel设计代码和仿真测试波形，观察四条指令的执行过程波形，记录并解释其含义。

PC

（初值为0）

+4

指令内存

Instr\_decoder

Reg\_file

RS1[4:0]

RS2[4:0]

WB\_data[31:0]

Reg\_WB

RS1\_out[31:0]

RS2\_out[31:0]

add\_op

sub\_op

lw\_op

sw\_op

nop