CPU 部件实验

实验介绍

本实验将实现一些 CPU 的必要部件，以便于以后 CPU 的设计

实验目标

1. 使用 verilog 实现 extend（作为作业 4 提交）
2. 使用 verilog 实现 PC 寄存器（作为作业 5 提交）
3. 使用 verilog 实现 ram（作为作业 6 提交）

实验原理

接口定义，可以直接复制到文件中

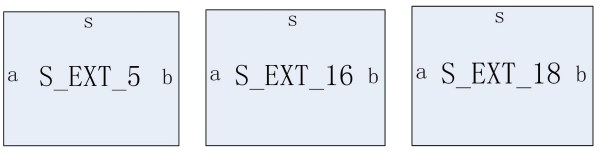
（请务必按照接口定义编写代码，在将来的实验中也是如此，模块名也请按照给出的定义命名）

1. Extend 模块功能及接口定义

扩展模块的功能为，将输入的数据扩展为 32 位数据

```
module ext #(parameter WIDTH = 16)(
        input [WIDTH - 1:0] a,          //输入数据，数据宽度可以根据需要自行定义
        input sext,                     //sext 有效为符号扩展，否则 0 扩展
        output [31:0] b                 //32 位输出数据，
    );
```

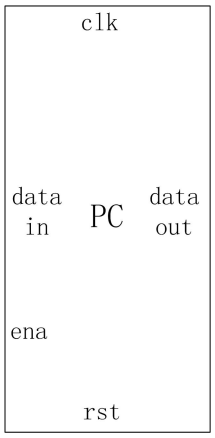
PS：MIPS 指令中有两种扩展需要实现，分别为 16 位扩展和 18 位扩展，使用***#(参数 1,参数 2,…)***方式来编写代码避免重复编写模块，可以查询 verilog 语法



2. pc 寄存器功能机接口定义

```
module pcreg(
        input clk,                  //  1 位输入，寄存器时钟信号，上升沿时为 PC
                                        寄存器赋值
        input rst,                  //  1 位输入，重置信号，高电平时将 PC 寄存器清零
                                    //  注：当 ena 信号无效时，rst 也可以重置寄存器
        input ena,                  //  1 位输入,有效信号高电平时 PC 寄存器工读入
                                        data_in 的值，否则保持原有输出
        input [31 : 0] data_in,     //  31 位输入，输入数据将被存入寄存器内部
        output reg [31 : 0] data_out //  31 位输出，工作时始终输出 PC 寄存器内部存储的
                                        值
    );
```

提示：上升沿触发使用 always @ (posedge clk)



3. ram 模块功能及接口定义

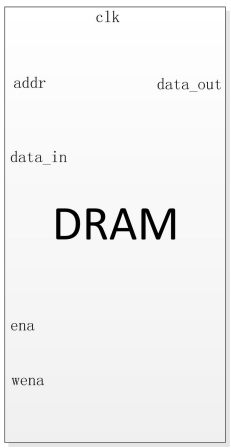
```
module ram (
        input clk,          //  存储器时钟信号，上升沿时向 ram 内部写入数据
        input ram_ena,      //  存储器有效信号，有效时存储器才运行，否则输出 z
        input wena,         //  存储器写有效信号，与 ram_ena 同时有效时才可写存
                                储器
        input [4:0] addr,   //  输入地址，指定数据读写的地址
        input [31:0] data_in, //  存储器写入的数据，在 clk 上升沿时被写入
```

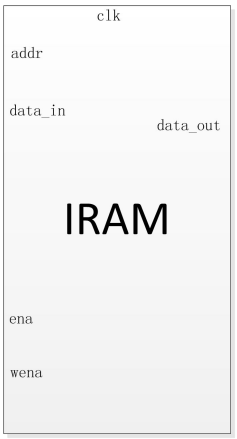
output [31:0] data_out // 存储器读出的数据，ram 工作时持续输出相应地址的数据

);

提示：可以使用 reg 数组来实现，大小至少 1024 bit

$readmemh("文件名", 数组名) 语句可以使用文件初始化 reg 数组





实验步骤

1. 新建 ISE 工程
2. 编写各个模块
3. 用 modelsim 仿真测试各模块