实验： PC 寄存器模块与数据存储器模块

实验要求

设计并验证 PC（Program Counter)寄存器与数据存储器(Data Memory)模块。

实验目的

1. 熟悉对时序逻辑电路的设计与调试，体会时序逻辑电路设计与软件设计的区别。
2. 熟悉 CPU 的组成部分。

实验指导

CPU 的运行流程以周期(Cycle)划分。在单周期 CPU 中， CPU 在每个时钟周期执行一条指令。由于组合逻辑电路的传播需要时间，所以在周期内，我们等待电路中的组合逻辑传播(即等待存取和运算），而在每个时钟沿时，所有存取/运算类的组合逻辑信号已经全部传播到位，此时我们更新每个模块中寄存器的值，进入下一个周期的执行。在本课程中，我们更为关心时序逻辑的设计。

PC（Program Counter)寄存器的作用是，在 CPU 运行时，指明当前正在执行的指令的地址，以便读取指令以及计算偏移。考虑最简单的情况， PC 拥有一个初始值，并在每个时钟周期自增 4（指令定长为 32 位)；而特殊情况下， PC 会受分支指令/跳转指令的影响，接受一个外部传入的值，以使得 CPU 的执行跳转到其它位置。

（**注：** PC 寄存器与通用寄存器(General Registers)不同，它拥有独立的电路，并在 CPU 运行时通过与通用寄存器不同的方式来访问和修改。）

在电路开始工作时，其内部的状态(寄存器的值)是不确定的，所以我们需要在电路开始运行时通过一个重置(Reset)信号来将变量赋值为一个确定的值(不要为此使用 initial 语句， initial 无法被编译为实际电路，应当仅在测试文件中使用)。在这里，我们规定，在 CPU 开始运行时， Reset 信号会保持一个周期，请在时钟沿时判断 Reset 信号是否被设置，如果被设置则将 PC 寄存器的值赋值为初始值 0x00003000。

以下是 PC 模块接口的设计参考(鼓励自己设计，不必严格遵循)：

module ProgramCounter

方向	信号名	数组	解释
input	reset		重置(Reset)信号，表示将状态恢复为初始状态。
input	clock		时钟信号，请在时钟沿时(如，上升沿时)对寄存器变量赋值。
input	jumpEnabled		是否开启跳转，如果该信号被设置，则 jumpInput 生效。
input	jumpInput	[31:0]	跳转输入。当开启跳转时，下个周期 PC 的值变为该值。
output	pcValue	[31:0]	输出 PC 寄存器的当前值。

为了简化设计，本课程使用哈佛架构(常用于 MCU 中)，即指令存储器与数据存储器分离(相对地，一般计算机使用冯·诺依曼架构，即指令与数据存储在统一存储器中)。本次实验要求大家实现数据存储器(Data Memory)。

在这里，我们使用一个寄存器数组来实现简单的存储器。读取是一个纯粹的组合逻辑操作，而写入是一个时序逻辑操作。与 PC 寄存器相同，在每个时钟周期结束时，为存储器执行写入操作。与 PC 寄存器相同，我们使用一个重置(Reset)信号来初始化数据存储器的值。

（**注：** 实践中，考虑到电路的成本和体积往往不这么实现存储器，而是使用 RAM。访问 RAM 更加复杂，并且其延迟将远远大于电路内的组合逻辑延迟，所以实践中访问内存往往是 CPU 执行过程中最耗时的操作。）

以下是数据存储器模块接口的设计参考(鼓励自己设计，不必严格遵循)：

module DataMemory

方向	信号名	数组	解释
input	reset		重置(Reset)信号，表示将状态恢复为初始状态(全 0)。
input	clock		时钟信号，请在时钟沿时(如，上升沿时)对寄存器变量赋值。
input	address	[31:0]	输入要读取或写入的目标地址。
input	writeEnabled		是否开启写入，如果该信号被设置，则 writeInput 生效。
input	writeInput	[31:0]	要写入的值。当写入开启时，在时钟沿时向目标地址写入值。
output	readResult	[31:0]	输出从目标地址读取到的值。

为了简化，我们在这里规定数据存储器总是以 32 位(4 字节)对齐读写，所以目标地址的最低两位总是 0 。推荐将数据存储器实现为 32 位 × 1024 (总共 4 KiB)的 reg (即 reg [31:0] data [1023:0];)。并使用目标地址的 [31:2] 来访问。

请在编写上述两个模块后，自行设计测试文件进行测试。在测试 PC 寄存器时，请测试正常自增和跳转两个功能。在测试存储器时， 可以诸周期运行并展开存储器模块的值寄存器观察结果(Vivado)， 或通过将存储器数组中一些成员的值加入波形图来观察结果(其它开发工具)。