模块接口

信号名	方向	描述
clk	I	时钟信号
reset	I	同步复位信号
interrupt	I	外部中断信号
macroscopic_pc [31:0]	O	宏观 PC
i_inst_addr [31:0]	O	IM 读取地址（取指 PC）
i_inst_rdata [31:0]	I	IM 读取数据
m_data_addr [31:0]	O	DM 读写地址
m_data_rdata [31:0]	I	DM 读取数据
m_data_wdata [31:0]	O	DM 待写入数据
m_data_byteen [3:0]	O	DM 字节使能信号
m_int_addr [31:0]	O	中断发生器待写入地址
m_int_byteen [3:0]	O	中断发生器字节使能信号
m_inst_addr [31:0]	O	M 级 PC
w_grf_we	O	GRF 写使能信号
w_grf_addr [4:0]	O	GRF 待写入寄存器编号
w_grf_wdata [31:0]	O	GRF 待写入数据
w_inst_addr [31:0]	O	W 级 PC

数据通路

五级流水线

阶段	简称	功能概述
取指阶段（Fetch）	F	从指令存储器中读取指令
译码阶段（Decode）	D	从寄存器文件中读取源操作数并对指令译码以便得到控制信号
执行阶段（Execute）	E	使用 ALU 执行计算
存储阶段（Memory）	M	读或写数据存储器
写回阶段（Writeback）	W	将结果写回到寄存器文件

sw 指令：向内存写入对应的字

地址[1:0]	m_data_byteen [3:0]	用途
XX	1111	`m_data_wdata[31:24]` 写入 byte3 `m_data_wdata[23:16]` 写入 byte2 `m_data_wdata[15:8]` 写入 byte1 `m_data_wdata[7:0]` 写入 byte0

sh 指令：向内存写入对应的半字

地址[1:0]	m_data_byteen [3:0]	用途
0X	0011	`m_data_wdata[15:8]` 写入 byte1 `m_data_wdata[7:0]` 写入 byte0
1X	1100	`m_data_wdata[31:24]` 写入 byte3 `m_data_wdata[23:16]` 写入 byte2

sb 指令：向内存写入对应的字节

地址[1:0]	m_data_byteen [3:0]	用途
00	0001	`m_data_wdata[7:0]` 写入 byte0
01	0010	`m_data_wdata[15:8]` 写入 byte1
10	0100	`m_data_wdata[23:16]` 写入 byte2
11	1000	`m_data_wdata[31:24]` 写入 byte3

T_{use}：表示数据到了 D 级之后还需要多少个周期要使用，每个指令的T_{use}是固定不变的（指令进入IF/ID寄存器后，其后的某个功能部件再经过多少cycle就必须要使用相应的寄存器值）
T_{new}：表示数据还有多长时间产生，会随着数据的流水动态减少（位于ID/EX及其后各流水线的指令，再经过多少个时钟周期，能够产生要写入寄存器的结果）
会产生结果的指令：cal_r类，cal_i类，ld类———可充当供给方，考察其T_{new}

测试方案

文件夹

思考题

1、请查阅相关资料，说明鼠标和键盘的输入信号是如何被 CPU 知晓的?

IO设备的输入输出有好几种方式，键盘、鼠标这类的低速设备是通过中断请求的方式进行IO操作的。即当键盘上按下一个按键的时候，键盘会发出一个中断信号，中断信号经过中断控制器传到CPU，然后CPU根据不同的中断号执行不同的中断响应程序，然后进行相应的IO操作，把按下的按键编码读到寄存器（或者鼠标的操作），最后放入内存中。

2、请思考为什么我们的 CPU 处理中断异常必须是已经指定好的地址？如果你的 CPU 支持用户自定义入口地址，即处理中断异常的程序由用户提供，其还能提供我们所希望的功能吗？如果可以，请说明这样可能会出现什么问题？否则举例说明。（假设用户提供的中断处理程序合法）

中断异常处理需要指定好地址的原因主要是出于系统的稳定性、可靠性和安全性考虑。让用户自定义中断处理程序的入口地址可能会引入一些潜在的问题：

1. 安全性问题： 允许用户自定义中断处理程序可能导致安全漏洞。恶意用户可以利用这一特性来执行恶意代码，破坏系统的稳定性，甚至攻击其他用户的数据。系统设计者通常会限制对关键系统部分的用户自定义，以减少潜在的攻击面。
2. 性能问题： 如果用户提供的中断处理程序效率低下或者包含不当的操作，可能会影响系统整体的性能。系统设计者希望中断处理程序能够尽快地完成执行，以便恢复正常的程序执行。
3. 一致性问题： 操作系统和系统软件通常需要一个固定的接口来与硬件进行通信，以确保一致性和兼容性。如果允许用户提供自定义的中断处理程序入口地址，可能导致接口不稳定，使得软件开发变得更加困难。
4. 可维护性问题： 系统维护和升级变得更加困难，因为系统设计者需要考虑用户提供的中断处理程序可能带来的各种变化。这增加了系统的复杂性，降低了可维护性。
5. 硬件约束： 某些硬件设计可能并不支持在运行时动态地改变中断处理程序的入口地址。硬件设计通常是基于一定的假设和约定的，而用户自定义中断处理程序可能会违反这些约定。

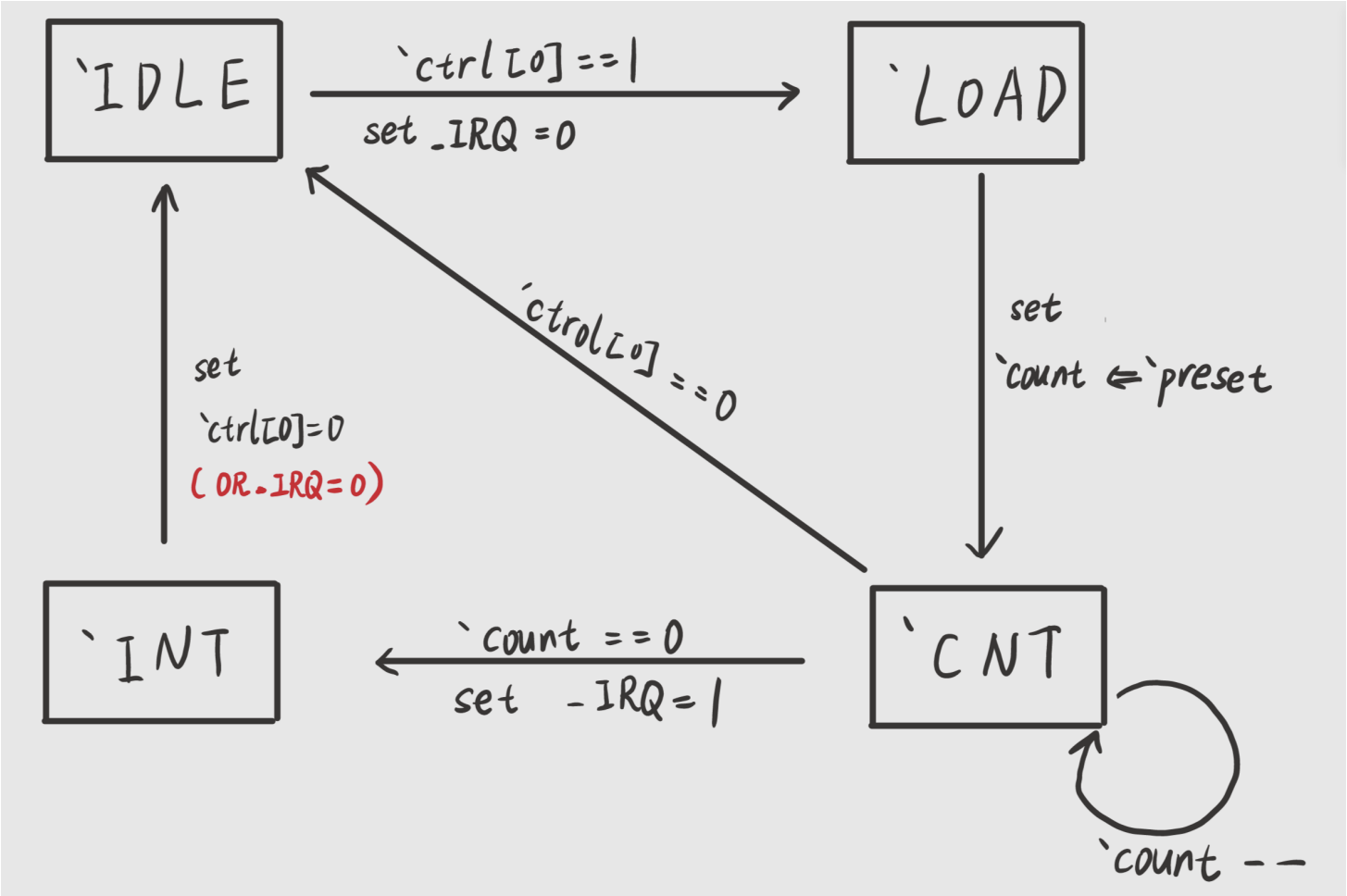
总的来说，限制中断处理程序的入口地址是为了确保系统的安全性、可靠性和一致性。虽然理论上用户提供的中断处理程序可能是合法的，但引入这样的机制会增加系统设计和维护的复杂性，带来潜在的安全和性能问题。因此，通常情况下，中断处理程序的入口地址由系统设计者事先指定，并且用户不被允许在运行时修改。

3、为何与外设通信需要 Bridge?

让CPU不需要关心具体的数据从何而来，只需要知道地址就可以了。一个模块只做一件事情。达到了高聚合的目的。

4、请阅读官方提供的定时器源代码，阐述两种中断模式的异同，并针对每一种模式绘制状态移图。

0：
当计数器倒计数为 0 后，计数器停止计数，此时控制寄存器中的使能 Enable 自动变为 0。当使能 Enable 被设置为 1 后，初值寄存器值再次被加载至计数器，计数器重新启动倒计数。模式 0 通常用于产生定时中断。
1：
当计数器倒计数为 0 后，初值寄存器值被自动加载至计数器，计数器继续倒
计数。模式 1 通常用于产生周期性脉冲。



5、倘若中断信号流入的时候，在检测宏观 PC 的一级如果是一条空泡（你的 CPU 该级所有信息均为空）指令，此时会发生什么问题？在此例基础上请思考：在 P7 中，清空流水线产生的空泡指令应该保留原指令的哪些信息？

会导致宏观PC为0或者为x。
在清空ID/EX流水线寄存器的时候，应该保留PC信息。

6、为什么 jalr 指令为什么不能写成 jalr $31, $31？

如果 jalr $31, $31 的延迟槽内发生异常或需要响应中断。那么 $31 寄存器的值已经被 jalr 改变，但是处理异常结束后，会再次执行 jalr 指令，这就会跳转到不正确的 PC 地址。

7、[P7 选做] 请详细描述你的测试方案及测试数据构造策略。

主要针对中断和异常（TC难捏）