标题：ex3.4实验报告

日期：2023年9月3日

作者：chms

练习3.4要求：

1. 启动代码(汇编)增加初始化中断相关寄存器和 timer 寄存器的内容，timer 设置为几千个时钟周期产生一次中断的频率。增加中断入口跳转。

2. 主程序(c 语言)反复做数学计算，例如算 1+..+100，算某个大整数等。

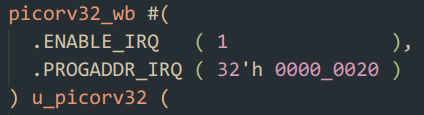
3. 编写 ISR（中断处理子程序/汇编），功能包括保护现场、设置 timer 寄存器、全局变量ticktack 自增、回复现场并从中断返回。

4. 调试并确认反复的时钟中断 ISR 调用不会影响主程序的计算工作。

模块名称picorv32\_wrapper

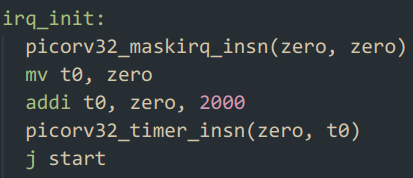
1. 配置picorv32核

设置ENABLE\_IRQ参数，并将中断处理程序地址设置为0x00000020，即32地址处。



1. 启动汇编

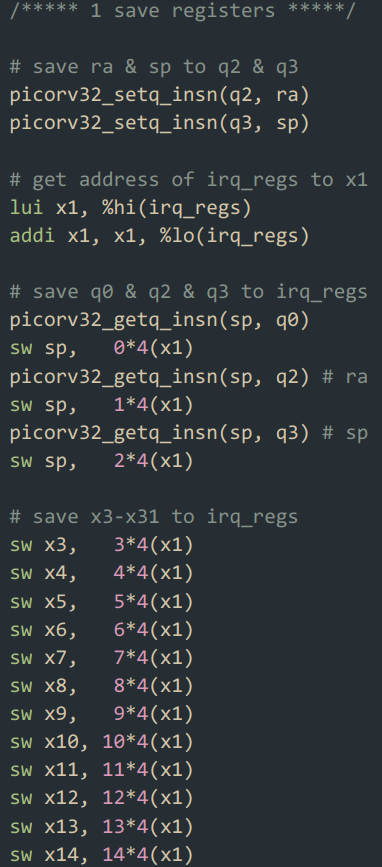
初始化q1（irq flag），设置irq timer为2000，即2000个周期产生一次timer中断。随后跳转到start。



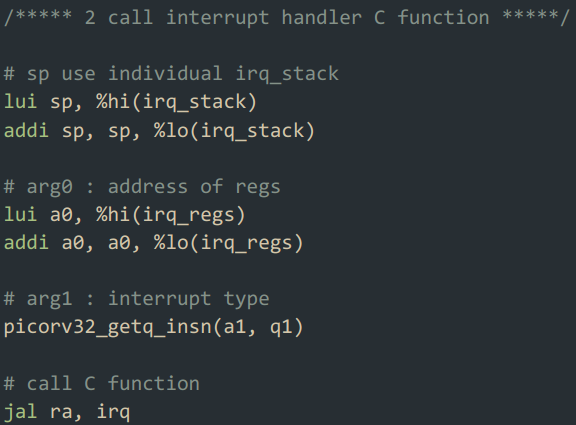
编写中断处理程序irq\_handler。使其地址对齐为32(0x00000020)，便于调用。



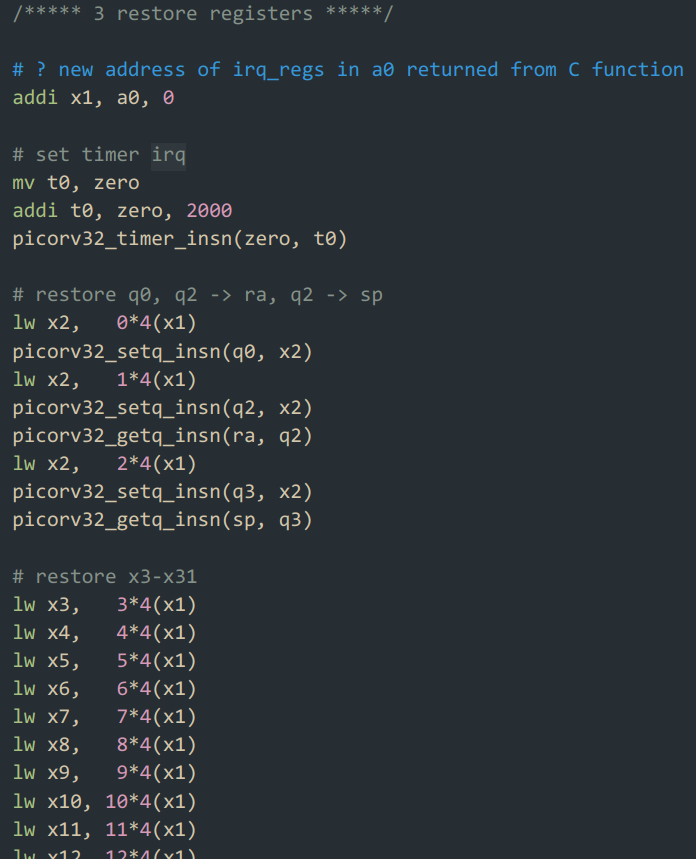
保存现场寄存器。



调用C ISR。



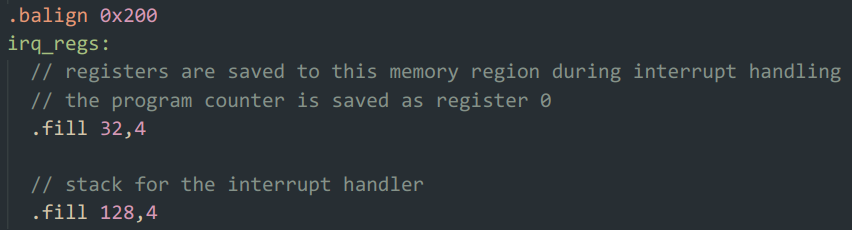
恢复现场。



退出中断。



irq\_regs：用于存储保存的现场寄存器的区域。

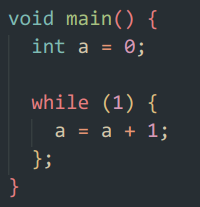


独立的irq堆栈。



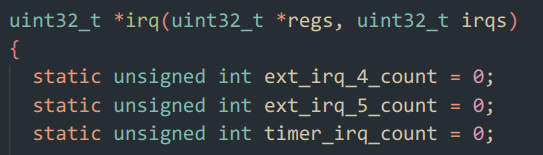
1. 主程序

简单递增计算。



1. ISR

使用了picorv32的例程。



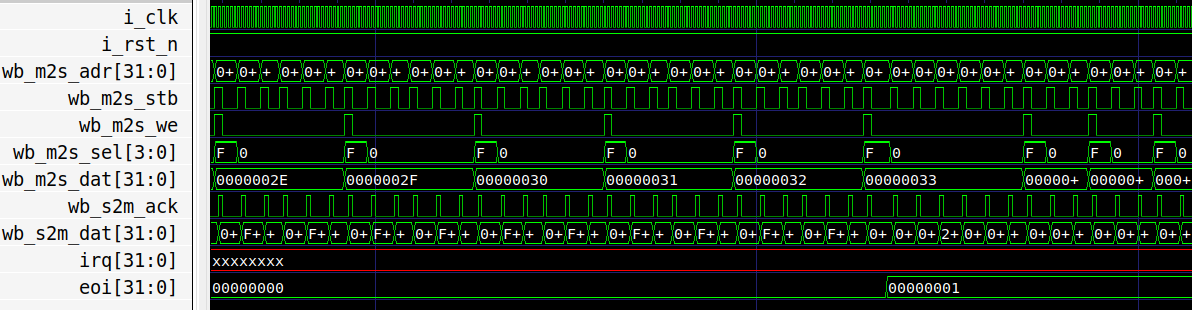
测试：

Timescale：采用1ns/10ps

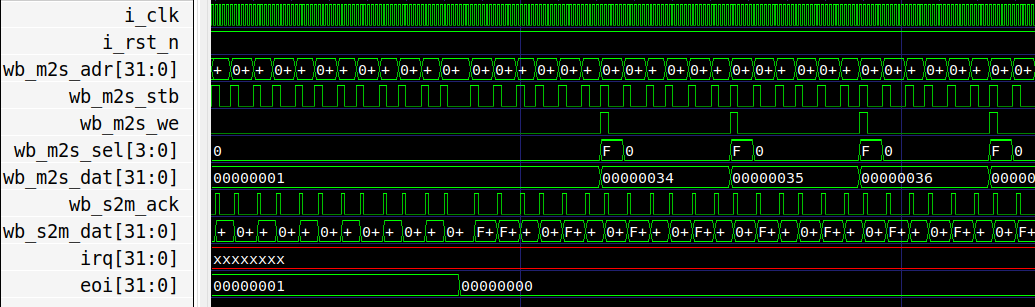
Define：无

Testbench采用10MHz时钟，激励在第一个时钟上升沿后给出。

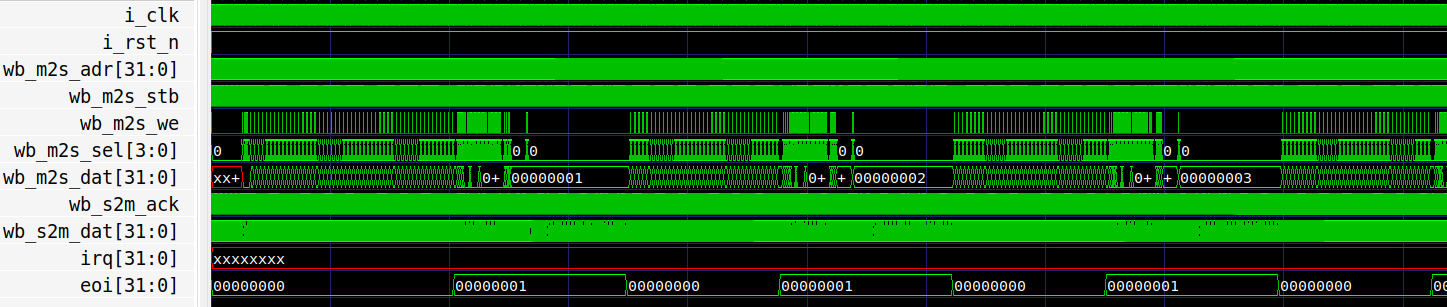
1. 计数到0x33时，产生中断，eoi拉高，表示进入中断处理程序。



1. 退出中断处理程序后，eoi拉低，主程序继续进行计数。同时，timer中断计数增加1。



1. 多次进入中断，timer中断计数递增。



1. 可以看出，中断并不影响主程序计数。

