版本历史

文档更新记录	文档名:	Lab08_例外与中断支持（二）
	版本号	V0.1
	创建人:	计算机体系结构研讨课教学组
	创建日期 :	2019-10-22

更新历史

序号	更新日期	更新人	版本号	更新内容
1	2019/10/22	贾凡	-	草稿。
2	2019/10/28	邢金璋	V0.1	初版。

文档信息反馈: xingjinzhang@loongson.cn

1 实验九 例外和中断支持（二）

在学习并尝试本章节前，你需要具有以下环境和能力：

(1) 装有 Vivado 的电脑一台。

(2) 熟悉 Vivado，并能初步使用。

如果对 Vivado 不熟悉，请参考课程讲义中的第一讲内容。

(3) 初步掌握 Verilog 的简单语法。

(4) 熟悉龙芯体系结构实验箱（Artix-7）。

(5) 了解 CPU 流水线结构。

通过本章节的学习，你将获得：

(1) 深入理解例外和中断的工作过程。

(2) 理解软件和硬件中断的区别和各个的用途。

(3) 掌握在流水线 CPU 中添加例外和中断的方法。

本次实验需要参考的文档包括但不限于：

(1) Lab09 任务书（本文档）。

(2) 体系结构研讨课总讲义之第七章“例外和中断的支持”。

(3) 体系结构研讨课总讲义之附录 D“‘体系结构研讨课’MIPS 指令系统规范”。

(4) MIPS 官网文档——MIPS32 手册。

1.1 实验目的

1. 加深对流水线结构的理解。
2. 学会在流水线 CPU 中添加例外和中断的方法。

1.2 实验设备

1. 装有 Xilinx Vivado 的计算机一台。
2. 龙芯体系结构教学实验箱（Artix-7）一套。

1.3 实验任务

本次实验只有一个任务：

1. 在实验八的 CPU 代码基础上，完成：
   a) CPU 增加 BREAK 指令，也就是增加 break 例外支持，。
   b) CPU 增加地址错、整数溢出、保留指令例外支持。
   c) CPU 增加 CP0 寄存器 COUNT、COMPARE、BADVADDR。
   d) CPU 增加时钟中断支持，时钟中断要求固定绑定在硬件中断 5 号上，也就是 CAUSE 对应的 IP7 上。

e) CPU 增加 6 个硬件中断支持，编号为 0~5，对应 CAUSE 的 IP7~IP2。

f) CPU 增加 2 个软件中断支持，对应 CAUSE 的 IP1~IP0。

g) 完成 lab9 功能测试。

h) 推荐在 lab9 的 myCPU 上运行记忆游戏程序，并正确运行。对于运行成功的同学，本次实验会有少许加分。

2. 本次实验要求以组为单位提交实验报告和 RTL 代码，以报告评分和现场检查评分作为最后的实验得分：

(1) 报告评分：描述自己的设计方案，记录调试过程（错误记录应该配截图）。实验报告模板请使用 lab3 的模板。

(2) 现场检查评分：检查包含仿真检查和上板检查。

1.4 实验检查

检查前需提交实验报告和调试好的 RTL 代码。本次实验在 2019 年 11 月 5 日进行检查。

现场检查，分仿真检查和上板检查：

1) 仿真检查：对照波形描述例外处理的通路。

2) 上板检查：查看上板行为。

现场检查要求能正确应对检查者的提问，并根据要求进行正确的操作演示

1.5 实验提交

提交的作品包括纸质档和电子档。

(1) 纸质档提交

提交方式：课上现场提交，每组提交一份。

截止时间：2019 年 11 月 5 日 18:10。

提交内容：纸质档 lab9 实验报告。

(2) 电子档提交

提交方式：打包上传到 Sep 课程网站 lab9 作业下，每组只需要一人提交。

截止时间：2019 年 11 月 5 日 18:10。

提交内容：电子档为一压缩包，文件名是“lab9_箱子号.zip”，目录层次如下（请将其中的“箱子号”替换为本组箱子号）。

|-lab9_箱子号/	目录，lab9 作品。
| |--lab9_箱子号.pdf/	Lab9 实验报告，实验报告模板参考“Lab03 实验报告模板_仅供参考.docx”
| **|--myCPU /**	**目录，myCPU 源码。目录请加一个 readme，简单描述下各文件。**

1.6 实验环境

本次实验的硬件环境沿用 lab7 发布的 UCAS_CDE，软件环境使用新发布的软件环境 func_lab9.zip 和 memory_game_lab9.zip，本次软件加入了对前述的新加入指令的测试内容。

本次实验的步骤是：

1) 准备好 lab8 的实验环境，UCAS_CDE，该环境会也会作为 lab9 的实验环境。

2) 将 lab9 发布的软件程序 func_lab9.zip，解压后，将 func_lab9/拷贝到 UCAS_CDE/soft/目录里，与 func_lab8 同层次。

3) 打开 cpu132_gettrace 工程（UCAS_CDE/cpu132_gettrace/run_vivado/cpu132_gettrace/cpu132_gettrace.xpr）。

4) 对 cpu132_gettrace 工程中的 inst_ram 重新定制，此时选择加载 func_lab9 的 coe（UCAS_CDE/soft/func_lab9/obj/inst_ram.coe）。

5) 运行 cpu132_gettrace 工程的仿真（进入仿真界面后，直接点击 run all 等待仿真运行完成），生成新的参考 trace 文件 golden_trace.txt（UCAS_CDE/cpu132_gettrace/golden_trace.txt）。要等仿真运行完成，golden_trace.txt 才有完整的内容。

6) 打开 myCPU 工程（UCAS_CDE/mycpu_verify/run_vivado/mycpu_prj1/mycpu_prj1.xpr）。

7) 对 myCPU 工程中的 inst_ram 重新定制，此时选择加载 func_lab9 的 coe（UCAS_CDE/soft/func_lab9/obj/inst_ram.coe）。

8) 运行 myCPU 工程的仿真（进入仿真界面后，直接点击 run all），开始 debug。

9) 仿真通过后，进行综合、布局布线和生成 bit 流文件，并进行上板验证。

在 func_lab9 功能测试通过后，推荐运行记忆游戏（memory_game_lab9.zip）的测试，步骤如下（注意记忆游戏测试无法运行仿真）：

1) 准备好 lab9 的实验环境，UCAS_CDE。

2) 将 lab9 发布的记忆游戏程序 memory_game_lab9.zip，解压后，将 memory_game_lab9/拷贝到 UCAS_CDE/soft/目录里，与 func_lab9 同层次。

3) 打开 myCPU 工程（UCAS_CDE/mycpu_verify/run_vivado/mycpu_prj1/mycpu_prj1.xpr）。

4) 对 myCPU 工程中的 inst_ram 重新定制，此时选择加载 memory_game_lab9 的 coe（UCAS_CDE/soft/memory_game_lab9/obj/inst_ram.coe）。

5) 进行综合、布局布线和生成 bit 流文件。

6) 将生成的 bit 流文件下载到开发板上，进行记忆游戏的测试。