32位MIPS处理器实验需求文档

SHS小组
乔奕 郭嘉丞 陈雨兰

目 录

1 引言

撰写本需求文档的目的在于，明确项目目标和具体需要完成的功能，其中具体讨论了CPU 的模块设计和与操作系统的相关接口，对于操作系统中与CPU 实现相关不大的部分并不涉及，如有需要，可阅读本小组的操作系统移植实验记录。

我们实现的CPU 为一个无缓存的多时钟流水线CPU ，操作系统为清华大学教学操作系统ucore 。

1.1 背景

系统名称：32位MIPS处理器

任务提出者：刘卫东老师 白晓颖老师

开发者：计32 乔奕 计44 郭嘉丞 计35 陈雨兰

1.2 实验目标

1. 使用提供的开发板,在FPGA 上编程实现一个基于标准32 位MIPS 指令集的子集的流水CPU,支持异常、中断、TLB 等。
2. 在该CPU 上运行ucore 操作系统,进入用户态及shell 环境,正常执行shell 命令。
3. 修改ucore, 实现简单的远程文件执行功能,即通过串口从PC 上获取ELF 文件,并在本地执行。
4. 可选实现对VGA、ps/2 keyboard 等其它外设的支持。

1.3 参考资料

《实验指导文档》

《计算机组成与设计硬件/软件接口》

2 功能需求

2.1 ALU

ALU 负责双操作数的算术逻辑运算，由ALUOp 信号控制，完成数据和地址的算术、逻辑、移位运算，输出结果。其中，乘法指令不通过ALU 计算，由乘法器支持，PC + 4 运算由专门的加法器支持。

为支持流水线CPU 的正常运行，部分运算如跳转指令的地址计算不通过ALU 。

2.2 乘法器

乘法器的实现独立于ALU，最后的乘法结果为64 位二进制，写入HI 和LO 寄存器。

乘法运算耗时较长，可能超过一个时钟周期，此时需要阻塞系统。

2.3 寄存器堆

对于32 位MIPS 系统，需要在寄存器堆中实现32 个通用32 位寄存器，在指令解码阶段读取寄存器内容，在写回阶段完成写入。

2.4 CP0

系统控制协处理器CP0 主要提供管理CPU 资源所需的机制，包括MMU、TLB 与异常处理控制。通过调用MFC0，MTC0 指令，CP0 提供了统一的对外接口以完成对寄存器组的访问。

下表为需要实现的CP0 寄存器及其主要功能。

编号	寄存器名称	寄存器功能
0	Index	用于TLBWI 指令访问TLB 入口的索引序号
2	EntryLo0	作为TLBWI 及其他TLB 指令接口，管理偶数页入口
3	EntryLo1	作为TLBWI 及其他TLB 指令接口，管理奇数页入口
9	BadVAddr	捕捉最近一次地址错误或TLB 异常时的虚拟地址
10	Count	每隔一个时时增加1，用作计时器，并可使能控制
11	EntryHi	TLB 异常时，系统将虚拟地址部分写入EntryHi 寄存器中用于TLB 匹配信息
12	Compare	Compare 保持一定值，当Count 值与Compare 相等时，SI_TimerInt 引脚变高电平直到有数值写入Compare，用于定时中断
13	Status	表示处理器的操作模式、中断使能及诊断状态
15	Cause	记录最近一次异常原因，控制软件中断请求和中断处理派分向量
16	EPC	存储异常处理之后程序恢复执行的地址
18	EBase	识别多处理器系统中不同的处理器异常向量的基地址

Status 寄存器 Status 寄存器的区域联合作用，可以创建协理器的工作模式。当以下所有条件都成立时启用中断：

Status[0]:IE = 1

Status[0]:EXL = 0

Status[0]:ERL = 0

此时设置IM（Status[16:9]）位和IE 位可以使能中断。

EXL 与ERL 任一位置1 都可使系统进入Kernel 模式，否则为User 模式。异常处理开始时，将Status[1] 赋值为1，在执行ERET 指令时将Status[1]赋值为0。

Cause 寄存器 Cause 寄存器记录了最近一次异常的原因，也控制软件中断请求以及中断处理派分的向量。Cause[6:2] 表示异常号。

Count 寄存器 每经过一个周期，Count 自增1 ，需内置一个加法器。

2.5 MMU

内存管理单元MMU 通过TLB异常实现。我们需要在CPU上实现一个TLB列表，每次访存需要通过TLB列表把线性地址（即虚拟地址）转化为物理地址。此外，还需要实现TLB重填的功能，包括设计若干CP0寄存器，并实现TLB重填指令tlbwi 。

MIPS 内存地址分配 本实验中，地址0x80000000以上，属于kseg0区域，供操作系统内核使用，这一区域不进行map，因此，在我们实现的MMU 中，应当判断地址是否大于0x80000000，若是，则不进行映射，若否，进入TLB查找阶段。从0x00000000到0x80000000是KUSE域，这块内存会被用于用户进程的虚存分配。

通过TLB异常实现MMU

1. CPU发起访存，使用了32位线性地址。TLB模块抽取线性地址中的前20位，作为VPN，在TLB表中查找。若查找得到，则根据表中的PPN，结合offset得到物理地址，然后直接访存。
2. 若无法找到，将该线性地址传入CP0中的vaddr寄存器，然后触发TLBmiss异常，进入异常处理程序。
3. 操作系统取出vaddr中的值，即访存失败的线性地址。若地址所在页表不存在，则建立新表。之后，把新表所对应的物理地址，存入CP0 寄存器中。之后，操作系统利用汇编语句直接调用tlbwi
4. 此时，CPU根据实现了的tlbwi 指令，利用CP0寄存器中的值，重新填充TLB中的某一项。此处，轮流重填TLB中的项（也可以随机填充）。
5. mips异常机制会重新执行上一条代码，此时能够正常访存。

2.6 异常处理

本实验中，由硬件检测到异常发生，填写相应的CP0 寄存器后，跳转到操作系统的异常处理函数。操作系统根据CP0 寄存器的值判断异常的类型，分发到相应的处理代码。

2.6.1 异常向量

MIPS32 CPU上有两组异常处理向量，根据SR(BEV)位(SR寄存器的bit22)切换：

BEV==1: ROM上的异常处理 这是我们CPU刚刚启动时处于的状态。由于操作系统的启动还没有完成，我们的异常处理向量被放在ROM固件中。由于没有缓存系统，所以没有缓存错误的入口点。我们的CPU也没有支持EIC异常。由于我们的操作系统还没有启动，我们在实现中简单的将所有的ROM异常都直接进行无限循环。

我们将会使用具体的异常向量表如下：

入口地址	类型
0xBFC00200	简单的TLB重填
0xBFC00380	其他所有异常

BEV==0: RAM上的异常处理 当操作系统完成自己的异常向量的装载时，就会将此位置零。并且向C0寄存器堆中的EBase寄存器写入我们的异常向量的基址。实际上的异常向量基址Base为EBase[29:12]&0...0，此时我们会用到的异常向量表如下：

入口地址	类型
BASE+0x000	简单的TLB重填
BASE+0x180	其他所有异常

2.6.2 异常描述

异常与中断列表如下：

异常号	异常名	描述
0	Interrupt	外部中断，异步发生，由硬件引起
1	TLB Modified	内存修改异常，发生在Memory 阶段
2	TLBL	读未在TLB 中映射的内存地址触发的异常
3	TLBS	写未在TLB 中映射的内存地址触发的异常
4	ADEL	读访问一个非选节地址触发的异常
5	ADES	写访问一个非选节地址触发的异常
8	SYSCALL	系统调用
10	RI	执行未定义指令异常
11	Co-Processor Unavailable	调试访问不存在的协处理器异常
23	Watch	Watch 寄存器监控异常

可能用到的中断号如下：

中断号	设备
0	系统计时器
1	键盘
3	通讯端口COM2
4	通讯端口COM1

2.6.3 CP0寄存器设置

EPC 如果Cause(BD)为1，也就是异常发生在延迟槽中，实际发生异常的指令就为EPC+4。但是我们返回执行的位置仍然应该是EPC，否则分枝指令的跳转就会无法执行。因此CPU必须在内部记录指令是否在延迟槽内。如果在延迟槽内的指令发生了异常，应该设置PC=EPC-4。

2.7 串口

2.8 指令集

我们采用的是MIPS32 的标准子集作为指令集，共计48 条指令，每一条指令是一个32位字。由于编译器版本不同，无法保证囊括所有所需指令，可能需要实现这48 条以外的指令。

具体的指令内容详见附录A 。

2.9 数据通路

3 性能需求

实现多周期流水CPU，用旁路处理数据冒险。主频取决于流水线耗时最长的模块。

4 运行环境

主要硬件设备信息如下：

FPGA	Xilinx Spartan6 xc6slx100
RAM	32-bit 字长，4 块，共8MB
Flash	16-bit 字长，共8MB
CPLD	与FPGA 相连，用于I/O
以太网接口	100MB

A MIPS指令集

表中rs，rt，rd 为寄存器编号，immediate 为立即数。