计算机组成原理-P7 MIPS 微体系——异常与中断实验报告

一、CP0 设计

端口定义表

信号名	方向	描述
clk	I	时钟信号
reset	I	重置信号
CP0_i_Addr	I	读/写功能所写寄存器地址
CP0_i_WData[31:0]	I	写入数据
CP0_i_PCAtM[31:0]	I	M 级指令 PC
CP0_i_ExcCode[6:2]	I	M 级指令异常代码（0 为无异常）
CP0_i_HWInt[5:0]	I	5 个硬件中断接入
CP0_i_isInsertedNop	I	当前 M 级指令是否为插入的气泡
CP0_i_isBranch	I	当前 M 级指令是否为跳转/分支等带有延迟槽的指令
CP0_i_WEnable	I	CP0 寄存器写入使能
CP0_i_EXLClr	I	清零 EXL 位
CP0_o_IntReq	O	中断/异常请求
CP0_o_EPC[31:0]	O	当前 EPC 的值
CP0_o_RData[31:0]	O	mfc0 指令用，输出 Addr 处寄存器数据

CP0 协处理器处理异常和中断信号，位于 M 级。利用 isInsertedNop 信号和 isBranch 信号及内部一个不可见寄存器（存储上一条非气泡指令是否为分支指令）来判定 BD 位以及设置合适的 EPC。

异常时，各流水寄存器同步复位，EXL 位当场设为 1，屏蔽各种中断。Cause 寄存器写入 BD 位，ExcCode 以及 HWInt 对应的 IP7-2 值并不再改变。中断异常结束执行 eret 指令时，eret 指令达到 D 级时插入一个地址为此时 EPC 的气泡，并改变 PC 值，EXL 在 M 级置 0。

二、桥与 IO 设计

SystemBridge 端口定义表

信号名	方向	描述
BRG_i_Addr[31:0]	I	CPU 输入的要写的地址
BRG_i_ByteEnable[3:0]	I	字节使能
BRG_i_WData_shifted[31:0]	I	偏移后对应到指定位的数据
BRG_i_WEnable[31:0]	I	CPU 输入的使能
BRG_o_RData[31:0]	O	向 CPU 输出设备读入的数据
BRG_i_DEV0_RData[31:0]	I	Dev0 所读数据
BRG_i_DEV1_RData[31:0]	I	Dev1 所读数据
BRG_o_Dev_Addr[31:0]	O	设备工作地址
BRG_o_Dev0_WEnable	O	Dev0 写使能
BRG_o_Dev1_WEnable	O	Dev1 写使能
BRG_o_Dev_WData[31:0]	O	向设备写的数据

向设备写数据时，CPU 给出 BE 和偏移好的数据（e.g. 向字中的第三个字节写一个字节，就是左移 16 位后的数据），Bridge 处理后传给 Device。Bridge 根据地址分析命中哪个设备。读数据时只需地址即可。Bridge 根据命中设备选择输出给 CPU 的数据，结合 CPU 输入的 WEnable 给出各设备的写使能。

三、测试程序

首先根据给出的弱测代码初步测试中断异常功能。

其次测试对异常的检测，例如检测读/写数异常是否能正常被检测：

```
.ktext 0x4180
kstart:
li $k0, 0xe01c
mfc0 $k0, $13
mfc0 $k1, $14
addiu $k1, $k1, 4
mtc0 $k1, $14
eret
j kstart

.text
li $2, 0x5010
li $3, 0x1013
sw $2, 0($3)
lui $12, 0x1111
lw $4, 0($3)
lui $13, 0x2222
lw $5, -1($3)
lui $14, 0x3333
sw $2, 0($2)
lui $15, 0x4444
lw $3, 0($2)
lui $16, 0x5555
lw $3, -1($2)
lui $16, 0x5666

li $3, 0x0002
sw $2, 0($3)
lw $2, 0($3)
li $3, 0x0001
sw $2, 0($3)
lui $17, 0x6666
lw $2, 0($3)
lui $18, 0x7777

li $3, 0x7f10
lw $4, ($3)
lw $5, 4($3)
lw $6, 8($3)
```

```
lw $7, 7($3)
lui $19, 0x8888
lh $8, 7($3)
lui $20, 0x9999
lh $9, 6($3)
lui $21, 0xaaaa
lb $4, 6($3)
lui $22, 0xbbbb
sb $2, 0($3)
lui $23, 0xcccc
sh $2, 2($3)
lui $24, 0xdddd
sw $2, 8($3)
lui $25, 0xeeee
li $3, 0x7fffffff
lw $2, 0x7ffd($3)
sw $2, 0x7ffd($3)
lui $11, 0xffff
li $4, 0x80000000
lw $2, -32768($3)
sw $2, -32768($3)
lui $11, 0xfbef
```

最后，构造一些特殊的中断/异常序列，特别关注在插入的气泡中加入的中断，观察 EPC 等相关行为。

五、思考题

1. 我们计组课程一本参考书目标题中有“硬件/软件接口”接口字样，那么到底什么是“硬件/软件接口”？

指令集架构（Instruction Set Architecture， ISA）就是所谓的软件/硬件接口。软件只需要按照指令集架构进行编写，就可以调动相应硬件完成功能，不必了解具体硬件架构。硬件要能完成指令集架构的功能即可，具体细节可以自行协定。且按照指令集架构编写的软件，在实现了指令集架构的硬件上均可顺利执行。在这个层面上，指令集架构可以算是一种抽象。

2. 在我们设计的流水线中，DM 处于 CPU 内部，请你考虑现代计算机中它的位置应该在何处。

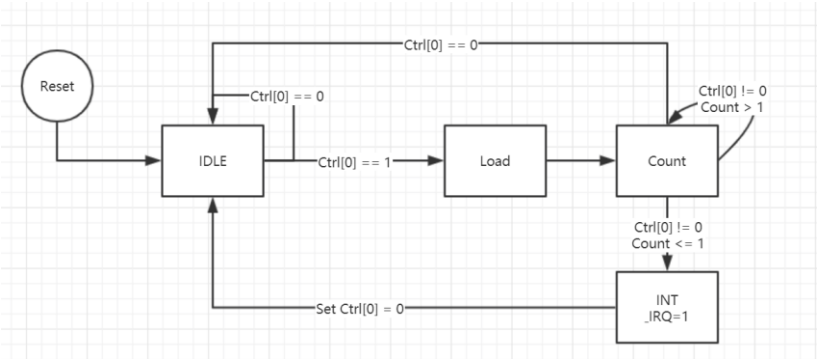
在 CPU 之外，由总线通过系统桥与 CPU 连接。

3. BE 部件对所有的外设都是必要的吗？

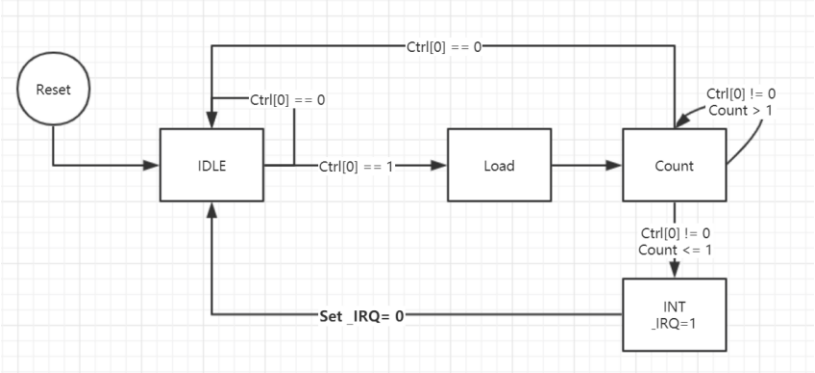
不是，部分外设只能按字存取。

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，并分别针对每一种模式绘制状态转移图

模式 0 下，计时器计时结束后停止计时，中断信号持续有效。



模式 1 下，计时器计时结束后重新开始计时，中断信号仅有效 1 周期。



5. 请开发一个主程序以及定时器的 exception handler。整个系统完成如下功能：

1）定时器在主程序中被初始化为模式 0；

2）定时器倒计数至 0 产生中断；

3）handler 设置使能 Enable 为 1 从而再次启动定时器的计数器。2 及 3 被无限重复。

4）主程序在初始化时将定时器初始化为模式 0，设定初值寄存器的初值为某个值，如 100 或 1000。（注意，主程序可能需要涉及对 CP0.SR 的编程，推荐阅读过后文后再进行。）

```
.ktext 0x4180
li $k0, 0x7f00
lw $k1, 0($k0)
ori $k1, $k1, 0x0001
sw $k1, 0($k0)
eret

.text
li $t0, 0x7f00
li $t1, 50
sw $t1, 4($t0)
mfc0 $t0, $12
ori $t0, $t0, 0xfc01
mtc0 $t0, $12
li $t0, 0x7f00
li $t1, 0x0009
sw $t1, 0($t0)

wait_loop:
beq $0, $0, wait_loop
```

```
addiu $s0, $s0, 1
```

6. 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？

鼠标键盘按键被按下时产生一个中断，CPU 检测相应内存空间对应的位置得知输入信号，进行相应处理然后返回。