P7 异常与中断

一. 整体结构

1. 处理器应支持 MIPS-lite4 指令集。

MIPS-C4 = {LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、SUB、SUBU、MULT、MULTU、DIV、DIVU、SLL、SRL、SRA、SLLV、SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO、ERET、MFC0、MTC0}

2. 处理器为流水线设计。

二. 模块规格

1.CP0.v

模块接口

文件	模块接口定义
CP0.v	module CP0(input clk, input reset, input temp_M, input [31:0] IR_M, input [31:0] IR_W, input [4:0] A1, // mfc0 input [4:0] A2, // mtc0 input [31:0] Din, input [31:0] PC, input [6:2] ExcCode, input [7:2] HWInt, //六个中断设备 input WE, input EXLSet, input EXLClr, input interrupt,

	output IntReg, output [31:0] EPC, output [31:0] Dout //Cp0 寄存器输出数据);

功能定义

序号	功能名称	功能描述
1	CP0	CP0

2. CPU.v

模块接口

文件	模块接口定义
CPU.v	module cpu(input clk, input reset, input interrupt, input [31:0] PrRD, input [7:2] HWInt, output PrWE, output [31:0] PrAddr,//PrAddr output [31:0] PrWD, //PrWD output [31:0] addr);

功能定义

序号	功能名称	功能描述
1	CPU	CPU

3. Bridge.v

模块接口

文件	模块接口定义
Bridge.v	module bridge(input [31:0] PrAddr, input [31:0] PrWD, input [31:0] DEV0_RD, input [31:0] DEV1_RD, input PrWE, output DEV0_WE, //写使能

	output DEV1_WE, //写使能 output [31:0] PrRD, // ReadData output [31:0] DEV_Addr, //Dev 地址 output [31:0] DEV_WD //写入数据);

功能定义

序号	功能名称	功能描述
1	Bridge	Bridge

4.DEV0.v

模块接口

文件	模块接口定义
DEV0.v	module DEV0(input clk, input reset, input [31:0] Addr, input WE, input [31:0] DataIn, output [31:0] DataOut, output IRQ);

功能定义

序号	功能名称	功能描述
1	DEV0	计时器

5.DEV1.v

模块接口

文件	模块接口定义
DEV1.v	module DEV1(input clk, input reset, input [31:0] Addr, input WE, input [31:0] DataIn, output [31:0] DataOut, output IRQ

	);

功能定义

序号	功能名称	功能描述
1	DEV1	计时器

ExcCode 用来传递每一级的异常编码

本project需要支持的异常：

ExcCode	助记符	描述
0	Int	中断
4	AdEL	取数或取指时地址错误
5	AdES	存数时地址错误
10	RI	不认识的（或者非法的）指令码
12	Ov	自陷形式的整数算术指令（例如add）导致的溢出

四．测试程序

(1)异常中断测试

1.取数异常

```
ktext 0x4180
  mfc0   $k0, $14
  addu   $k0, $k0, 4
  mtc0   $k0, $14
  eret
.text
  ori $28, $0, 0x0000
  ori $29, $0, 0x0000
  lui $8, 0x7000
```

```
  lui $9, 0xf000
  lw $9,3($0)
  sub $10, $8,$9
  or  $10, $8, $9
  ori $28, $0, 0x0000
  ori $29, $0, 0x0000
  lui $8, 0x7000
  lui $9, 0xf000
  lh $9,3($0)
  sub $10, $8,$9
  or  $10, $8, $9
  ori $28, $0, 0x0000
  ori $29, $0, 0x0000
  ori $8, 0x7fffffff
  lui $9, 0x1
  add $10, $8,$9
  lw $a0,0x1000($8)
  or  $10, $8, $9
```

2.存数异常

(1).text

```
  ori $8, 0x7f00
  lui $9, 0xf000
  sw $9,3($0)
  sub $10, $8,$9
```

```
  or  $10, $8, $9
  ori $8, 0x7f00
  lui $9, 0xf000
  sh  $9,1($0)
  sub $10, $8,$9
  or  $10, $8, $9
```

3.非法指令

```
  sh  $9,1($0)
  bgezla $t2,next
  nop
  next:
  sub $10, $8,$9
  or  $10, $8, $9
```

4.算术溢出

```
.ktext 0x4180
  mfc0   $k0, $14
  sub $8,$8,$8
  mtc0   $k0, $14
  eret
.text
  lui $8, 0x7fff
  lui $9, 0x7fff
  add $10, $8,$9
  or  $10, $8, $9
```

```
  lui $8, 0x7fff
  addi $10, $8,0x7fff0000
  or  $10, $8, $9
  lui $8, 0x7fff
  ori $8, $8, 0xffff
  addi   $10, $8, 1
  lui $8, 0x7000
  lui $9, 0xf000
  sub $10, $8,$9
  or  $10, $8, $9
```

5.延迟槽中的异常

（1）.ktext 0x4180

```
  mfc0 $1,$13
  sub $9,$9,$9
  eret
.text
  ori $8, 0x7fffffff
  ori $9, 0x1000
  j next
  add $10, $8,$9
  lw $a0,0x1000($8)
  eee:
```

```
or  $10, $8, $9
ori$8, 0x7fffffff
ori $9, 0x1000
j eee
add $10, $8,$0
lw $a0,0x1000($8)
eee:
or  $10, $8, $9
j end
sw $1,1($0)
end:
ori $a0,$0,0
```

6.中断

```
.text
ori $7,$0,0xfc01
mtc0 $7,$12
li $5,0x98765432
li $6,0xfedcba98
div $5,$6
mflo $6
label:
beq $0,$0,label
nop
```

```
.ktext 0x4180
_entry:
  mfc0   $k0, $14
  mfc0   $k1, $13
  ori $k0, $0, 0x1000
  sw  $sp, -4($k0)

  addiu  $k0, $k0, -256
  move   $sp, $k0

  beq $0,$0,_save_context
  nop

_main_handler:
  mfc0   $k0, $13
  ori    $k1, $0, 0x007c
  and $k0, $k1, $k0
  beq    $0, $k0, _restore_context
  nop
  mfc0   $k0, $14
  addu   $k0, $k0, 4
  mtc0   $k0, $14
  beq $0,$0,    _restore_context
  nop
```

```
_restore:
  eret

_save_context:
  sw    $1, 4($sp)
      sw    $2, 8($sp)
      sw    $3, 12($sp)
      sw    $4, 16($sp)
      sw    $5, 20($sp)
      sw    $6, 24($sp)
      sw    $7, 28($sp)
      sw    $8, 32($sp)
      sw    $9, 36($sp)
      sw    $10, 40($sp)
      sw    $11, 44($sp)
      sw    $12, 48($sp)
      sw    $13, 52($sp)
      sw    $14, 56($sp)
      sw    $15, 60($sp)
      sw    $16, 64($sp)
      sw    $17, 68($sp)
      sw    $18, 72($sp)
      sw    $19, 76($sp)
```

```
    sw     $20, 80($sp)
    sw     $21, 84($sp)
    sw     $22, 88($sp)
    sw     $23, 92($sp)
    sw     $24, 96($sp)
    sw     $25, 100($sp)
    sw     $26, 104($sp)
    sw     $27, 108($sp)
    sw     $28, 112($sp)
    sw     $29, 116($sp)
    sw     $30, 120($sp)
    sw     $31, 124($sp)
  mfhi   $k0
  sw $k0, 128($sp)
  mflo   $k0
  sw $k0, 132($sp)
  beq $0,$0,    _main_handler
  nop



_restore_context:
  lw $1, 4($sp)
    lw     $2, 8($sp)
```

```
lw     $3, 12($sp)
lw     $4, 16($sp)
lw     $5, 20($sp)
lw     $6, 24($sp)
lw     $7, 28($sp)
lw     $8, 32($sp)
lw     $9, 36($sp)
lw     $10, 40($sp)
lw     $11, 44($sp)
lw     $12, 48($sp)
lw     $13, 52($sp)
lw     $14, 56($sp)
lw     $15, 60($sp)
lw     $16, 64($sp)
lw     $17, 68($sp)
lw     $18, 72($sp)
lw     $19, 76($sp)
lw     $20, 80($sp)
lw     $21, 84($sp)
lw     $22, 88($sp)
lw     $23, 92($sp)
lw     $24, 96($sp)
lw     $25, 100($sp)
lw     $26, 104($sp)
```

```
    lw     $27, 108($sp)
    lw     $28, 112($sp)
    lw     $29, 116($sp)
    lw     $30, 120($sp)
    lw     $31, 124($sp)
lw  $k0, 128($sp)
mthi   $k0
lw  $k0, 132($sp)
mtlo   $k0
    beq $0,$0,    _restore
nop
```

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”？

   硬件接口

   CPU 与外设(被控对象)在硬件上连接构成一个有机整体

   如日常生活中鼠标键盘。

   软件

   目标：控制设备工作方式，传递信息。

   如控制窗口驱动程序。

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。

主存里，CPU 通过 cache 与主存交换数据或直接交换数据。

3. BE 部件对所有的外设都是必要的吗？

不是，本次加的 timer 只能用 lw,sw,用不到 BE 了。

4.详见设计文档。

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：

定时器在主程序中被初始化为模式 0；定时器倒计数至 0 产生中断；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。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


  ori $t0,$0,0x00007f00
  sw $a0,0($t0)
  eret
.text
  ori $t1,0x00007f00
  ori $a0,0x0001
  ori $a3,0xfc01
  ori $a1,2
  sw $a1,4($t1)
  sw $a0,0($t1)
```

```
mtc0 $a3,$12

ori $29,0x100
```

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？

设备实际上包括两部分接口控制器和设备主体，设备主体不直接与主机连接，而是通过接口控制器与主机连接。鼠标和键盘的输入信号相当于中断，当键盘、鼠标有信息时，产生一个中断然后中断例程会从端口读入数据到寄存器。CPU 接收到中断请求之后进入中断处理程序获得鼠标键盘的信息。