# P7：MIPS微体系——异常与中断

## 一、设计说明

1. MIPS处理器应该支持MIPS-C4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. 本次MIPS微系统需要支持的异常：

表1 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 |

## 二、微系统设计文档

### 1. CP0设计

#### (1) **模块接口定义**

表1 CP0**模块接口定义**

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| cp0.v | module cp0(  input reset,  input clk,  input [4:0] A1,  input [4:0] A2,  input [31:0] DIn,  input [31:0] PC,  input [6:2] ExcCode,  input [5:0] HWInt,  input We,  input EXLSet,  input EXLClr,  input delayslot,  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  output IntReq,  output [31:0] EPC,  output [31:0] DOut  ); |

#### (2) **接口说明**

表2 CP0接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 同步复位信号  1：复位  0：无效 |
| 3 | A1 | I | 读CP0寄存器编号，执行MFC0指令时产生 |
| 4 | A2 | I | 写CP0寄存器编号，执行MTC0指令时产生 |
| 5 | DIn | I | 执行MTC0指令时写入 |
| 6 | PC | I | EPC所用PC值 |
| 7 | ExcCode | I | 内部异常信号 |
| 8 | HWInt | I | 外部中断信号 |
| 9 | We | I | CP0写使能，执行MTC0指令时产生 |
| 10 | EXLSet | I | 用于置位SR的EXL(EXL为1)，流水线在M阶段产生 |
| 11 | EXLClr | I | 用于清除SR的EXL(EXL为0)，执行ERET指令时产生 |
| 12 | delayslot | I | 分支跳转指令延迟槽标记位 |
| 13 | DOut | O | 执行MFC0指令时读出 |

#### (3)功能定义

表3 CP0功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 执行MFC0指令 | reset有效时，PC被设置为0x00003000 |
| 2 | 执行MTC0指令 | 时钟上升沿来临且pc\_en有效时，PC值更新为nextPC |
| 3 | 执行ERET指令 | 返回正常程序 |
| 4 | 进入异常处理程序 | 产生IntReq信号 |

### 2. 桥与IO设计

#### (1) **模块接口定义**

表4**模块接口定义**

|  |  |
| --- | --- |
| 文件 | 模块接口定义 |
| bridge.v | module bridge(  input [31:2] PrAddr,  input [31:0] PrWD,  input PrWe,  input [31:0] DEV0\_RD,  input [31:0] DEV1\_RD,  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  output [31:2] DEV\_Addr,  output [31:0] DEV\_WD,  output DEV0\_WE,  output DEV1\_WE,  output [31:0] PrRD  ); |

#### (2) **接口说明**

表5接口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号 | 方向 | 描述 |
| 1 | PrAddr | I | 30位地址值，为ALU\_Out在M级的结果 |
| 2 | PrWD | I | 写入DEV的数据 |
| 3 | PrWe | I | 写使能信号 |
| 4 | DEV0\_RD | I | DEV0数据 |
| 5 | DEV1\_RD | I | DEV1数据 |
| 6 | DEV\_Addr | O | 写入地址 |
| 7 | DEV\_WD | O | 写入DEV的数据 |
| 8 | DEV0\_WE | O | 写DEV0使能信号 |
| 9 | DEV1\_WE | O | 写DEV1使能信号 |
| 10 | PrRD | O | 读取的DEV信号 |

#### (3)功能定义

表6功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 连通IO | 沟通CPU与外设 |

1. PC
2. 端口说明

表3-PC端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 复位信号 |
| 3 | Jumpsign | I | 是否进行非PC+4跳转 |
| 4 | JumpAddr[31:0] | I | 非PC+4时NPC值 |
| 5 | freeze | I | 是否暂停 |
| 6 | PC[31:0] | O | 当前指令所在IM地址 |

1. 功能定义

表4-PC功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 存储指令的地址 | 输出当前指令所在IM地址，并计算下一周期指令所在IM地址。 |

1. GRF
2. 端口说明

表7-GRF端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | clk | I | 时钟信号 |
| 2 | reset | I | 异步复位信号，将32个寄存器中全部清零  1：清零  0：无效 |
| 3 | WE | I | 写使能信号  1：可向GRF中写入数据  0：不能向GRF中写入数据 |
| 4 | A1[4:0] | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出到RD1 |
| 5 | A2[4:0] | I | 5位地址输入信号，指定32个寄存器中的一个，将其中存储的数据读出到RD2 |
| 6 | A3[4:0] | I | 5位地址输入信号，指定32个寄存器中的一个，作为RD的写入地址 |
| 7 | WD[31:0] | I | 32位写入数据 |
| 8 | RD1[31:0] | O | 输出A1指定的寄存器的32位数据 |
| 9 | RD2[31:0] | O | 输出A2指定的寄存器的32位数据 |

1. 功能定义

表8-GRF功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 异步复位 | reset为1时，将所有寄存器清零 |
| 2 | 读数据 | 将A1和A2地址对应的寄存器的值分别通过RD1和RD2读出 |
| 3 | 写数据 | 当WE为1且时钟上升沿来临时，将WD写入到A3对应的寄存器内部 |

1. ALU
2. 端口说明

表9-ALU端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | A[31:0] | I | 参与运算的第一个数 |
| 2 | B[31:0] | I | 参与运算的第二个数 |
| 3 | AluOp[2:0] | I | 000：无符号加  001：无符号减  010：与  011：或 |
| 4 | res[31:0] | O | A与B做运算后的结果 |

1. 功能定义

表10-ALU功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能 | 描述 |
| 1 | 加运算 | res = A + B |
| 2 | 减运算 | res = A - B |
| 3 | 与运算 | res = A & B |
| 4 | 或运算 | res = A | B |

1. Controller
2. 端口说明

表15-Controller端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | instr[31:0] | I | instr[31:26],6位控制信号 |
| 2 | PC | I | 指令所在IM地址 |
| 3 | ReadGF1[31:0] | I | 读的寄存器1数据 |
| 4 | ReadGF2[31:0] | I | 读的寄存器2数据 |
| 5 | ReadDM[31:0] | I | 读的数据存储器数据 |
| 6 | ALUresult[31:0] | I | Alu计算结果 |
| 7 | Jumpsign | O | 进行非PC+4跳转 |
| 8 | [31:0] JumpAddr | O | 跳转到的数 |
| 9 | WEgf | O | 寄存器堆写使能 |
| 10 | [4:0] WriteGFadd | O | 写的寄存器地址 |
| 11 | [31:0] WriteGFdata | O | 写的寄存器数据 |
| 12 | [4:0] readGFadd1 | O | 读的寄存器1地址 |
| 13 | [4:0] readGFadd2 | O | 读的寄存器2地址 |
| 14 | WEdm | O | 数据存储器写使能 |
| 15 | [31:0] WriteDMadd | O | 写的数据存储器地址 |
| 16 | [31:0] WriteDMdata | O | 写的数据存储器数据 |
| 17 | [31:0] readDMadd | O | 读的数据存储器地址 |
| 18 | [3:0] ALUopcode | O | Alu运算符 |
| 19 | [31:0] ALUinput1 | O | Alu读入数据1 |
| 20 | [31:0] ALUinput2 | O | Alu读入数据2 |

1. Gap
2. 端口说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号名 | 方向 | 描述 |
| 1 | [31:0] I\_Instr\_out | I | 指令 |
| 2 | [31:0] I\_PC\_out | I | 指令所在地址 |
| 3 | [31:0] D\_Instr\_in | O | 指令 |
| 4 | [31:0] D\_PC\_in | O | 指令所在地址 |
| 5 | [31:0] D\_Instr\_out | I | 指令 |
| 6 | [31:0] D\_PC\_out | I | 指令所在地址 |
| 7 | [31:0] D\_Num1\_out | I | ALU输入数字1 |
| 8 | [31:0] D\_Num2\_out | I | ALU输入数字2 |
| 9 | [31:0] D\_Storedata\_out | I | DM存入数据 |
| 10 | [3:0] D\_ALUopcode\_out | I | ALU运算符 |
| 11 | D\_WEGF\_out | I | 寄存器写使能 |
| 12 | D\_WEDM\_out | I | DM写使能 |
| 13 | [31:0] E\_Instr\_in | O | 指令 |
| 14 | [31:0] E\_PC\_in | O | 指令所在地址 |
| 15 | [31:0] E\_Num1\_in | O | ALU输入数字1 |
| 16 | [31:0] E\_Num2\_in | O | ALU输入数字2 |
| 17 | [31:0] E\_Storedata\_in | O | DM存入数据 |
| 18 | [3:0] E\_ALUopcode\_in | O | ALU运算符 |
| 19 | E\_WEGF\_in | O | 寄存器写使能 |
| 20 | E\_WEDM\_in | O | DM写使能 |
| 21 | [31:0] E\_Instr\_out | I | 指令 |
| 22 | [31:0] E\_PC\_out | I | 指令所在地址 |
| 23 | [31:0] E\_ALUresult\_out | I | 运算结果 |
| 24 | [31:0] E\_Storedata\_out | I | DM存入数据 |
| 25 | E\_WEGF\_out | I | 寄存器写使能 |
| 26 | E\_WEDM\_out | I | DM写使能 |
| 27 | [31:0] M\_Instr\_in | O | 指令 |
| 28 | [31:0] M\_PC\_in | O | 指令所在地址 |
| 29 | [31:0] M\_ALUresult\_in | O | 运算结果 |
| 30 | [31:0] M\_Storedata\_in | O | DM存入数据 |
| 31 | M\_WEGF\_in | O | 寄存器写使能 |
| 32 | M\_WEDM\_in | O | DM写使能 |
| 33 | [31:0] M\_Instr\_out | I | 指令 |
| 34 | [31:0] M\_PC\_out | I | 指令所在地址 |
| 35 | [31:0] M\_ALUresult\_out | I | 运算结果 |
| 36 | [31:0] M\_DMdata\_out | I | DM存入数据 |
| 37 | M\_WEGF\_out | I | 寄存器写使能 |
| 38 | [31:0] W\_Instr\_in | O | 指令 |
| 39 | [31:0] W\_PC\_in | O | 指令所在地址 |
| 40 | [31:0] W\_ALUresult\_in | O | 运算结果 |
| 41 | [31:0] W\_DMdata\_in | O | DM存入数据 |
| 42 | W\_WEGF\_in | O | 寄存器写使能 |

#### (1) **CPU测试代码**

.data

.space 64

arr1: .space 64

arr2: .space 64

.space 64

.text

N0: srl $t0, $t0, 28

addi $t0, $t0, 64

andi $t0, $t0, 0xfffffffe

lhu $zero, 14($t0)

N1: ori $zero, $t1, 37505

N2: ori $t1, $t1, 1

divu $zero, $t1

N3: srl $t2, $t2, 10

N4: sllv $zero, $t1, $zero

N5: sra $t0, $t0, 28

lbu $t1, 78($t0)

N6: srl $t3, $t3, 28

addi $t3, $t3, 64

andi $t3, $t3, 0xfffffffc

sw $t0, 8($t3)

N7: sra $zero, $zero, 28

lb $t3, 68($zero)

N8: srl $t3, $t3, 28

addi $t3, $t3, 64

andi $t3, $t3, 0xfffffffc

sw $t2, 8($t3)

N9: srl $t2, $t2, 28

addi $t2, $t2, 64

sb $t0, 14($t2)

N10: bgtz $t0, N16

N11: sra $t0, $t0, 28

andi $t0, $t0, 0xfffffffc

lw $t2, 76($t0)

N12: j N156

N13: sub $t2, $zero, $t3

N14: sra $t1, $t2, 17

N15: mult $t3, $t0

N16: and $zero, $zero, $t0

N17: sra $zero, $zero, 28

andi $zero, $zero, 0xfffffffe

lh $t3, 72($zero)

N18: xori $zero, $t2, 41539

N19: ori $t2, $t2, 1

divu $zero, $t2

N20: srl $t3, $t3, 28

addi $t3, $t3, 64

lb $t3, 15($t3)

N21: ori $t2, $t2, 1

divu $zero, $t2

N22: slt $t0, $t2, $t3

N23: sra $t2, $t2, 28

lbu $zero, 67($t2)

N24: mtlo $t0

N25: srl $zero, $zero, 28

addi $zero, $zero, 64

lb $t3, 0($zero)

N26: lui $zero, 17273

N27: srl $t0, $t0, 28

addi $t0, $t0, 64

lbu $t2, 3($t0)

N28: add $t3, $t1, $t1

N29: addu $t1, $t1, $t3

N30: or $t2, $zero, $t0

N31: srl $t3, $t3, 28

addi $t3, $t3, 64

andi $t3, $t3, 0xfffffffc

lw $zero, 8($t3)

N32: srl $zero, $zero, 28

addi $zero, $zero, 64

andi $zero, $zero, 0xfffffffc

lw $t0, 4($zero)

N33: beq $t3, $t1, N252

N34: srl $t3, $t3, 2

N35: addu $zero, $t1, $zero

N36: multu $t1, $zero

N37: srl $zero, $zero, 28

addi $zero, $zero, 64

andi $zero, $zero, 0xfffffffe

sh $t3, 8($zero)

N38: add $t0, $t1, $t3

N39: srl $t2, $t2, 28

addi $t2, $t2, 64

andi $t2, $t2, 0xfffffffe

sh $zero, 0($t2)

N40: addi $zero, $t0, -12622

N41: srl $t1, $t1, 28

addi $t1, $t1, 64

lbu $zero, 2($t1)

N42: bgtz $t3, N222

N43: srl $t0, $t0, 28

addi $t0, $t0, 64

andi $t0, $t0, 0xfffffffe

sh $t0, 0($t0)

N44: srav $t2, $t0, $t1

N45: sub $t1, $t1, $t2

N46: bltz $t3, N218

N47: mtlo $t0

N48: jal N261

N49: sra $t0, $t0, 28

andi $t0, $t0, 0xfffffffe

sh $t2, 78($t0)

N50: srl $t2, $t2, 28

addi $t2, $t2, 64

andi $t2, $t2, 0xfffffffe

lh $t3, 0($t2)

N51: sra $t3, $t3, 28

lb $t2, 70($t3)

N52: srav $t3, $t1, $t2

N53: mthi $zero

N54: sltu $t3, $t0, $t2

N55: j N86

N56: sllv $t1, $zero, $zero

N57: and $t3, $t3, $zero

N58: or $t1, $zero, $t1

N59: slt $t1, $t3, $t1

N60: sra $t3, $t3, 28

lb $zero, 75($t3)

N61: srl $t3, $t3, 28

addi $t3, $t3, 64

sb $zero, 3($t3)

N62: slt $zero, $t2, $t3

N63: jal N270

N64: srav $t0, $t1, $zero

N65: addiu $t3, $t2, 62512

N66: srav $zero, $t3, $t3

N67: addiu $t2, $t2, 5344

N68: srl $t3, $t3, 28

addi $t3, $t3, 64

lbu $zero, 9($t3)

N69: blez $zero, N80

N70: mtlo $t0

N71: srl $t0, $t0, 28

addi $t0, $t0, 64

andi $t0, $t0, 0xfffffffe

lh $t1, 6($t0)

N72: srl $t3, $t3, 28

addi $t3, $t3, 64

andi $t3, $t3, 0xfffffffe

lhu $t0, 6($t3)

N73: sra $t1, $t1, 28

sb $t0, 79($t1)

N74: srl $zero, $zero, 28

addi $zero, $zero, 64

lbu $zero, 12($zero)

N75: srl $t3, $t3, 28

addi $t3, $t3, 64

lbu $t2, 6($t3)

N76: bgez $t3, N118

N77: mfhi $zero

N78: sra $zero, $zero, 28

andi $zero, $zero, 0xfffffffe

lh $t1, 78($zero)

N79: sltiu $t0, $t1, -23716

N80: la $ra, N160

jalr $t2, $ra

N81: mthi $zero

N82: ori $t0, $t0, 1

div $zero, $t0

N83: srl $t2, $t2, 28

addi $t2, $t2, 64

andi $t2, $t2, 0xfffffffe

sh $zero, 0($t2)

N84: sra $t3, $t3, 28

andi $t3, $t3, 0xfffffffc

lw $t3, 68($t3)

N85: sra $t3, $t3, 28

sb $t3, 75($t3)

N86: j N192

N87: sub $t1, $zero, $t3

N88: slt $t3, $t3, $t3

N89: srl $t1, $t1, 28

addi $t1, $t1, 64

andi $t1, $t1, 0xfffffffe

lhu $t0, 6($t1)

N90: mfhi $zero

N91: la $t3, N143

jr $t3

N92: mthi $t0

N93: xor $t0, $t2, $t0

N94: sltu $t0, $t0, $zero

N95: la $ra, N184

jalr $t1, $ra

N96: srl $t0, $t0, 0

N97: bltz $t1, N211

N98: xori $t0, $t1, 14807

N99: add $t1, $zero, $t0

N100: srl $t3, $t3, 28

addi $t3, $t3, 64

andi $t3, $t3, 0xfffffffe

sh $zero, 2($t3)

N101: sra $t1, $t1, 28

andi $t1, $t1, 0xfffffffe

lhu $t3, 66($t1)

N102: la $t0, N170

jr $t0

N103: mflo $t1

N104: slti $t2, $t0, 5790

N105: ori $t3, $t3, 36145

N106: srl $zero, $zero, 28

addi $zero, $zero, 64

andi $zero, $zero, 0xfffffffc

lw $t3, 12($zero)

N107: blez $zero, N208

N108: srlv $zero, $t2, $t2

N109: sltiu $t0, $t2, 27776

N110: srav $t0, $t0, $zero

N111: andi $t3, $t3, 49925

N112: bne $t1, $t2, N297

N113: sll $t3, $zero, 20

N114: add $t0, $t2, $t0

N115: ori $t3, $t3, 1

div $t1, $t3

N116: mult $t2, $t0

N117: srl $t3, $t3, 28

addi $t3, $t3, 64

andi $t3, $t3, 0xfffffffe

lh $t3, 6($t3)

N118: addi $t2, $t2, -26557

N119: mflo $t0

N120: multu $t3, $t3

N121: srl $t0, $t0, 28

addi $t0, $t0, 64

sb $t1, 9($t0)

N122: addu $zero, $zero, $t3

N123: srl $t2, $t2, 28

addi $t2, $t2, 64

andi $t2, $t2, 0xfffffffc

sw $t3, 0($t2)

N124: addu $zero, $zero, $t1

N125: la $ra, N191

jr $ra

N126: sra $t1, $t1, 28

andi $t1, $t1, 0xfffffffe

sh $t1, 68($t1)

N127: sra $t0, $t0, 28

andi $t0, $t0, 0xfffffffc

lw $t0, 64($t0)

N128: nop

N129: blez $t2, N153

N130: or $t3, $t0, $t1

N131: sra $t3, $t3, 28

andi $t3, $t3, 0xfffffffc

lw $t3, 76($t3)

N132: bgez $t1, N170

N133: addiu $t3, $t0, 35280

N134: sub $t3, $t2, $t1

N135: sra $t3, $t3, 28

sb $zero, 65($t3)

N136: beq $t2, $t2, N203

N137: andi $t3, $zero, 55586

N138: sra $zero, $zero, 28

sb $t0, 65($zero)

N139: srl $zero, $zero, 28

addi $zero, $zero, 64

andi $zero, $zero, 0xfffffffe

lhu $t0, 6($zero)

N140: srl $zero, $zero, 28

addi $zero, $zero, 64

sb $zero, 1($zero)

N141: mtlo $zero

N142: lui $t2, 44183

N143: bgez $t0, N220

N144: mtlo $t1

N145: sub $t3, $zero, $zero

N146: srl $t1, $t1, 28

addi $t1, $t1, 64

andi $t1, $t1, 0xfffffffe

sh $t0, 2($t1)

N147: or $t1, $t1, $t3

N148: j N297

N149: lui $t1, 12458

N150: slti $t3, $t0, -20234

N151: sra $t0, $t0, 28

sb $zero, 67($t0)

N152: blez $zero, N188

N153: mult $zero, $zero

N154: srl $t0, $t0, 28

addi $t0, $t0, 64

lb $t1, 13($t0)

N155: srl $t1, $t1, 28

addi $t1, $t1, 64

lb $t2, 7($t1)

N156: srl $t2, $t2, 28

addi $t2, $t2, 64

andi $t2, $t2, 0xfffffffc

sw $t3, 8($t2)

N157: multu $t3, $t2

N158: srl $t0, $t0, 28

addi $t0, $t0, 64

andi $t0, $t0, 0xfffffffe

lh $zero, 8($t0)

N159: sub $t2, $zero, $zero

N160: andi $t3, $t1, 28872

N161: ori $zero, $zero, 62473

N162: mflo $zero

N163: srl $zero, $zero, 28

addi $zero, $zero, 64

andi $zero, $zero, 0xfffffffc

sw $zero, 12($zero)

N164: ori $t1, $t1, 35611

N165: sra $t0, $t0, 28

andi $t0, $t0, 0xfffffffc

sw $t3, 64($t0)

N166: sra $zero, $zero, 28

andi $zero, $zero, 0xfffffffe

sh $t0, 66($zero)

N167: srl $zero, $zero, 28

addi $zero, $zero, 64

andi $zero, $zero, 0xfffffffc

sw $t1, 0($zero)

N168: srl $t0, $t0, 28

addi $t0, $t0, 64

andi $t0, $t0, 0xfffffffe

sh $t0, 8($t0)

N169: j N219

N170: addiu $t1, $t0, 39577

N171: mtlo $t1

N172: sub $t1, $t1, $t1

N173: mflo $t0

N174: and $t3, $t3, $zero

N175: nop

N176: addi $zero, $t2, -19050

N177: srl $t3, $t3, 28

addi $t3, $t3, 64

lb $t1, 12($t3)

N178: la $t0, N209

jr $t0

N179: srlv $zero, $t2, $t1

N180: and $t1, $t3, $t3

N181: srl $t0, $t0, 28

addi $t0, $t0, 64

sb $t2, 7($t0)

N182: mflo $t1

N183: srl $t3, $t3, 28

addi $t3, $t3, 64

sb $t1, 3($t3)

N184: sra $t3, $zero, 19

N185: srl $t0, $t0, 28

addi $t0, $t0, 64

lb $t2, 11($t0)

N186: addi $t1, $t1, -15088

N187: andi $t1, $t2, 3119

N188: mfhi $t1

N189: mfhi $t3

N190: xori $t0, $zero, 57913

N191: sra $t0, $t0, 28

andi $t0, $t0, 0xfffffffc

lw $t1, 72($t0)

N192: sra $zero, $zero, 28

andi $zero, $zero, 0xfffffffc

sw $t1, 72($zero)

N193: srl $t1, $t2, 4

N194: nop

N195: subu $t2, $t1, $t0

N196: nor $t0, $t0, $t2

N197: srl $t3, $t3, 28

addi $t3, $t3, 64

andi $t3, $t3, 0xfffffffc

lw $zero, 0($t3)

N198: xor $t2, $t0, $t1

N199: srav $t0, $t2, $t2

N200: srl $t0, $t0, 28

addi $t0, $t0, 64

lb $t3, 11($t0)

N201: srl $zero, $zero, 28

addi $zero, $zero, 64

andi $zero, $zero, 0xfffffffe

lhu $t2, 10($zero)

N202: la $ra, N291

jr $ra

N203: srl $t1, $t1, 28

addi $t1, $t1, 64

andi $t1, $t1, 0xfffffffc

lw $t0, 8($t1)

N204: srlv $zero, $zero, $zero

N205: blez $t3, N207

N206: sra $zero, $zero, 28

lbu $t2, 70($zero)

N207: ori $t0, $zero, 37979

N208: andi $t2, $zero, 26014

N209: srl $zero, $zero, 28

addi $zero, $zero, 64

andi $zero, $zero, 0xfffffffc

sw $t1, 0($zero)

N210: sltiu $t0, $t3, 26027

N211: xor $t2, $zero, $zero

N212: bne $t1, $t2, N228

N213: srl $t1, $t1, 28

addi $t1, $t1, 64

andi $t1, $t1, 0xfffffffe

lhu $zero, 6($t1)

N214: multu $zero, $t2

N215: add $t3, $t3, $t2

N216: mtlo $t2

N217: sra $t3, $t3, 28

andi $t3, $t3, 0xfffffffc

sw $t3, 64($t3)

N218: slti $zero, $zero, 28363

N219: srl $t3, $t3, 28

addi $t3, $t3, 64

lb $t2, 13($t3)

N220: blez $t3, N223

N221: add $zero, $zero, $t1

N222: ori $t0, $t1, 36736

N223: srl $zero, $zero, 28

addi $zero, $zero, 64

andi $zero, $zero, 0xfffffffe

lhu $t3, 2($zero)

N224: addu $t3, $t2, $zero

N225: sltu $t0, $t3, $zero

N226: bgtz $t0, N246

N227: mthi $zero

N228: la $ra, N286

jr $ra

N229: ori $t2, $t2, 1

divu $t2, $t2

N230: sra $t0, $t0, 28

lb $t3, 78($t0)

N231: andi $t2, $t2, 52942

N232: nor $zero, $t1, $t1

N233: srl $t1, $t1, 28

addi $t1, $t1, 64

sb $zero, 5($t1)

N234: slti $t0, $t1, 22549

N235: j N264

N236: sra $zero, $zero, 28

andi $zero, $zero, 0xfffffffe

lhu $zero, 70($zero)

N237: srl $t0, $t0, 28

addi $t0, $t0, 64

andi $t0, $t0, 0xfffffffc

sw $t1, 4($t0)

N238: ori $t3, $t3, 1

div $t3, $t3

N239: bltz $t2, N242

N240: nor $zero, $t0, $t1

N241: bltz $t1, N294

N242: xori $t0, $t1, 37330

N243: and $t2, $t1, $zero

N244: sllv $t2, $t1, $t2

N245: xori $t2, $t3, 49069

N246: xori $zero, $t2, 27235

N247: mflo $t3

N248: srl $t3, $t3, 20

N249: sra $t3, $t3, 28

andi $t3, $t3, 0xfffffffc

sw $t3, 64($t3)

N250: srl $t2, $t2, 28

addi $t2, $t2, 64

andi $t2, $t2, 0xfffffffc

sw $t0, 0($t2)

N251: sra $zero, $zero, 28

sb $t1, 64($zero)

N252: multu $t3, $t0

N253: j N283

N254: slt $t3, $t1, $t1

N255: slti $t3, $zero, -16323

N256: slti $zero, $t3, -10831

N257: ori $t1, $t1, 59911

N258: ori $t0, $t0, 37111

N259: mult $zero, $t2

N260: mtlo $t0

N261: srl $t1, $t1, 28

addi $t1, $t1, 64

andi $t1, $t1, 0xfffffffe

lh $t1, 0($t1)

N262: slt $t2, $t2, $t1

N263: sra $t0, $t0, 28

andi $t0, $t0, 0xfffffffe

lh $t2, 64($t0)

N264: sll $zero, $t1, 14

N265: la $ra, N270

jr $ra

N266: and $t0, $zero, $t3

N267: mthi $t3

N268: or $t1, $zero, $zero

N269: la $ra, N274

jalr $t3, $ra

N270: mfhi $t3

N271: sra $zero, $zero, 28

andi $zero, $zero, 0xfffffffc

sw $t3, 76($zero)

N272: jal N300

N273: mtlo $t1

N274: srl $t1, $t1, 28

addi $t1, $t1, 64

lb $t3, 0($t1)

N275: subu $zero, $t3, $t1

N276: sra $zero, $t3, 26

N277: addiu $zero, $t0, 2101

N278: sllv $t0, $t2, $t0

N279: xor $t3, $t2, $t3

N280: srl $t0, $t0, 28

addi $t0, $t0, 64

andi $t0, $t0, 0xfffffffe

sh $t1, 14($t0)

N281: sra $zero, $zero, 28

sb $t1, 78($zero)

N282: sll $zero, $t1, 23

N283: srl $t2, $t2, 28

addi $t2, $t2, 64

andi $t2, $t2, 0xfffffffc

lw $t0, 0($t2)

N284: or $t2, $zero, $t1

N285: ori $t3, $t3, 1

divu $t2, $t3

N286: xori $t2, $t0, 34731

N287: xor $t2, $t2, $t0

N288: sltu $t3, $zero, $t1

N289: xor $t1, $t3, $t0

N290: ori $t1, $t1, 1

div $t0, $t1

N291: subu $t2, $t1, $zero

N292: bltz $t3, N294

N293: sra $t0, $t0, 28

sb $zero, 64($t0)

N294: jal N296

N295: sra $t0, $zero, 7

N296: sltiu $t2, $zero, 21893

N297: la $ra, EXIT

jr $ra

N298: nor $t0, $t0, $t3

N299: slt $t2, $t0, $t1

N300: nop

EXIT:

beq $zero, $zero, EXIT

nop

(2) 异常测试软件

.ktext 0x4180

mfc0 $k0, $14

addiu $k0, $k0, 4

mtc0 $k0, $14

lui $k0, 12

lui $k1, 2131

div $k1, $k0

eret

mflo $k0

lui $20, 0x3456

lui $21, 0x8654

lui $22, 0x9654

.text

li $5, 0x0000ff11

mtc0  $5, $12

li $2, 0x7fffffff

li $3, 0x7fffffff

add $2, $3, $2

j hhh

lui $17, 0x2333

ori $17, 0x1212

hhh:

ori $18, 0x1111

(3) 中断测试软件

`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date:   14:31:29 11/16/2016

// Design Name:   mips

// Module Name:   D:/ISE/P4/mips\_txt.v

// Project Name:  P4

// Target Device:

// Tool versions:

// Description:

//

// Verilog Test Fixture created by ISE for module: mips

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

////////////////////////////////////////////////////////////////////////////////

module mips\_txt;

    // Inputs

    reg clk;

    reg reset;

    reg interrupt;

    // Outputs

    wire [31:0] addr;

    // Instantiate the Unit Under Test (UUT)

    mips uut (

        .clk(clk),

        .reset(reset),

        .addr(addr),

        .interrupt(interrupt)

    );

    parameter delay\_pc = 32'h00004198;

    integer delay\_count;

    integer interrupt\_counter;

    integer needInterrupt;

    initial begin

        delay\_count = 0;

        interrupt = 0;

        needInterrupt = 0;

        interrupt\_counter = 0;

        // Initialize Inputs

        clk = 0;

        reset = 1;

        #20 reset = 0;

        // Wait 100 ns for global reset to finish

        // Add stimulus here

    end

   always #2 clk = ~clk;

   always @(negedge clk) begin

      if (reset) begin

        interrupt\_counter = 0;

        needInterrupt = 0;

        interrupt = 0;

      end else begin

        if (interrupt) begin

            if (interrupt\_counter == 0) begin

                interrupt = 0;

            end else begin

                interrupt\_counter = interrupt\_counter - 1;

            end

        end else if (needInterrupt) begin

            needInterrupt = 0;

            interrupt = 1;

            interrupt\_counter = 5;

        end else begin

            case (addr)

                delay\_pc:

                    begin

                        if (delay\_count == 0) begin

                            delay\_count = 1;

                            interrupt = 1;

                            interrupt\_counter = 5;

                        end

                    end

            endcase

        end

      end

   end

endmodule

.ktext 0x4180

mfc0 $k0, $14

addiu $k0, $k0, 4

mtc0 $k0, $14

lui $k0, 12

lui $k1, 2131

div $k1, $k0

eret

mflo $k0

lui $20, 0x3456

lui $21, 0x8654

lui $22, 0x9654

.text

li $5, 0x0000ff11

mtc0  $5, $12

li $2, 0x7fffffff

li $3, 0x5ff

add $2, $3, $2

lui $16, 0x1837

lui $17, 0x2333

#### (4) **IO测试软件**

.ktext 0x4180

\_entry:

    mfc0    $1, $13

    ori $k0, $0, 0x1000

    sw  $sp, -4($k0)

    addiu   $k0, $k0, -256

    move    $sp, $k0

    j   \_save\_context

    nop

\_quick\_handle:

    mfc0    $1, $13

    mfc0    $k0, $13

    andi    $k0, $k0, 0x00ff

    srl $k0, $k0, 2

    ori $k1, $0, 0x0004

    beq $k0, $k1, adel\_handler\_quick

    nop

    ori $k1, $0, 0x000a

    beq $k0, $k1, ri\_handler\_quick

    nop

    j \_entry

    nop

adel\_handler\_quick:

    mfc0    $t8,$14

    andi    $t9,$t8,3

    bne     $t9,$0,adel\_type\_1

    nop

    addi    $t9,$t8,-0x3000

    lui     $s7,0xffff

    ori $s7,$s7,0xe000

    and     $t9,$t9,$s7

    bne     $t9,$0,adel\_type\_2

    nop

    j       \_entry

    nop

adel\_type\_1:

    ori       $10,$0,0x3230

    mtc0      $10,$14

    eret

adel\_type\_2:

    ori       $10,$0,0x3240

    mtc0      $10,$14

    eret

ri\_handler\_quick:

    ori       $10,$0,0x3220

    mtc0      $10,$14

    eret

\_main\_handler:

    mfc0    $k0, $13

    andi    $k0, $k0, 0x00ff

    srl $k0, $k0, 2

    ori $k1, $0, 0x0000

    beq $k0, $k1, int\_handler

    nop

    ori $k1, $0, 0x0004

    beq $k0, $k1, adel\_handler

    nop

    ori $k1, $0, 0x0005

    beq $k0, $k1, ades\_handler

    nop

    ori $k1, $0, 0x000a

    beq $k0, $k1, ri\_handler

    nop

    ori $k1, $0, 0x000c

    beq $k0, $k1, ov\_handler

    nop

int\_handler:

    sw  $ra, 0($sp)

    addiu   $sp, $sp, -16

    mfc0    $v0, $12

    sw  $v0, 0($sp)

    mfc0    $v0, $13

    sw  $v0, 4($sp)

    # check INT[3]

    lw  $v0, 0($sp)

    lw  $v1, 4($sp)

    and $v0, $v1, $v0

    andi    $v0, $v0, 0x800

    bne $v0, $0, timer1\_handler

    nop

    # check INT[2]

    lw  $v0, 0($sp)

    lw  $v1, 4($sp)

    and $v0, $v1, $v0

    andi    $v0, $v0, 0x400

    bne $v0, $0, timer0\_handler

    nop

timer0\_handler:

    # first we load the global variable cnt0:

    # ++cnt0, then save to global variable cnt0

    li  $fp, 0x8

    lw  $t0, 0($fp)         # get cnt0

    addi    $s6, $0 , 5

    beq $t0, $s6, skip0

    nop

    addiu   $t0, $t0, 1         # add cnt0

skip0:  sw  $t0, 0($fp)         # update cnt0

    jal restart\_timer

    nop

    # mask INT[2]

    mfc0    $t0, $12

    andi    $t0, $t0, 0x03ff

    ori     $t0, $t0, 0x800

    mtc0    $t0, $12

    j   \_restore\_context

    nop

timer1\_handler:

    # first we load the global variable cnt1:

    # ++cnt1, then save to global variable cnt1

    li  $fp, 0xc

    lw  $t0, 0($fp)         # get cnt1

    addi    $s6, $0 , 5

    beq $t0, $s6, skip1

    nop

    addiu   $t0, $t0, 1         # add cnt1

skip1:  sw  $t0, 0($fp)         # update cnt1

    jal restart\_timer

    nop

    # mask INT[3]

    mfc0    $t0, $12

    andi    $t0, $t0, 0x03ff

    ori     $t0, $t0, 0x400

    mtc0    $t0, $12

    j   \_restore\_context

    nop

restart\_timer:

    # swap two PRESET

    li  $t0, 0x0

    lw  $t0, 0($t0)

    lw  $t5, 4($t0)

    li  $t2, 0x4

    lw  $t2, 0($t2)

    lw  $t6, 4($t2)

    # restart Timer 0

    li  $t1, 0x0

    lw  $t1, 0($t1)

    lw  $t0, 0($t1)     # $t0 is the CTRL Reg of Timer 0

    sw  $0, 0($t1)      # disable Timer 0

    li  $t2, 0x8

    lw  $t2, 0($t2)

    addi    $s6, $0 , 5

    beq $t2, $s6, f0        # check Timer0 pause times

    nop

    sw  $t6, 4($t1)     # refill the count number

    addiu   $t0, $0, 9      # set Timer0.CTRL

    sw  $t0, 0($t1)     # Timer 0 restart count

    f0:

    # restart Timer 1

    li  $t1, 0x4

    lw  $t1, 0($t1)

    lw  $t0, 0($t1)     # $t0 is the CTRL Reg of Timer 1

    sw  $0, 0($t1)      # disable Timer 1

    li  $t2, 0xc

    lw  $t2, 0($t2)

    addi    $s6, $0 , 5

    beq $t2, $s6, f1        # check Timer1 pause times

    nop

    sw  $t5, 4($t1)     # refill the count number

    addiu   $t0, $0, 9      # set Timer1.CTRL

    sw  $t0, 0($t1)     # Timer 0 restart count

    f1:

    jr  $ra

    nop

adel\_handler:

    mfc0    $t0, $14

    mfc0    $k0, $13

    lui $t2, 0x8000

    and $t3, $k0, $t2

    addi    $t0, $t0, 4

    bne $t3, $t2, adel\_nxt

    nop

    addi    $t0, $t0, 4

    adel\_nxt:

    mtc0    $t0, $14

    j   \_restore\_context

    nop

ades\_handler:

    mfc0    $t0, $14

    mfc0    $k0, $13

    lui $t2, 0x8000

    and $t3, $k0, $t2

    addi    $t0, $t0, 4

    bne $t3, $t2, ades\_nxt

    nop

    addi    $t0, $t0, 4

    ades\_nxt:

    mtc0    $t0, $14

    j   \_restore\_context

    nop

ri\_handler:

    mfc0    $t0, $14

    mfc0    $k0, $13

    lui $t2, 0x8000

    and $t3, $k0, $t2

    addi    $t0, $t0, 4

    bne $t3, $t2, ri\_nxt

    nop

    addi    $t0, $t0, 4

    ri\_nxt:

    mtc0    $t0, $14

    j   \_restore\_context

    nop

ov\_handler:

    mfc0    $t0, $14

    mfc0    $k0, $13

    lui $t2, 0x8000

    and $t3, $k0, $t2

    addi    $t0, $t0, 4

    bne $t3, $t2, ov\_nxt

    nop

    addi    $t0, $t0, 4

    ov\_nxt:

    mtc0    $t0, $14

    j   \_restore\_context

    nop

\_restore:

    eret

\_save\_context:

        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      $28, 112($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)

    j   \_main\_handler

    nop

\_restore\_context:

    li  $sp, 0x1000

    addi    $sp, $sp, -256

        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      $28, 112($sp)

        lw      $30, 120($sp)

        lw      $31, 124($sp)

    lw  $k0, 128($sp)

    mthi    $k0

    lw  $k0, 132($sp)

    mtlo    $k0

        lw      $29, 116($sp)

    ori     $1,$0,1

        j   \_restore

    nop

.data

.globl TC0\_BASE TC1\_BASE cnt0 cnt1 cnt0\_double cnt1\_double

TC0\_BASE: .word 0x7f00

TC1\_BASE: .word 0x7f10

cnt0: .word 0

cnt1: .word 0

cnt0\_double: .word 0

cnt1\_double: .word 0

.text

    ori $28, $0, 0x0000

    ori $29, $0, 0x0f00

    mtc0    $0, $12

    j   nxt1

    lw  $0, 1($0)           # 测试延迟槽内 lw 地址不对齐异常

nxt1:

    j   nxt2

    sw  $0, 1($0)           # 测试延迟槽内 sw 地址不对齐异常

nxt2:

    lui $8, 0x7fff

    ori $8, $8, 0xffff

    j   dead\_loop

    addi    $10, $8, 1      # 测试延迟槽内 addi 溢出异常

dead\_loop:

    j   dead\_loop

    nop

## 三、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／软件接口（简称为＂HSI＂）是一个术语，用来描述SoC外围设备的配置和功能，以及它们如何与CPU交互。

可让程序员在不直接操纵硬件的前提下，就编写出可以让硬件实现想要达到的目的的正确的程序，包括指令，寄存器，访存和IO。

### 2)BE 部件对所有的外设都是必要的吗？

不是，不是所有外设需要按照字节访问。

### 3)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

见计时器设计文档

### 4)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。）

### 

### 5)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？

键盘与鼠标在按键被按下或者抬起时都会发生中断，CPU读取对应键盘端口的寄存器来得知发生了什么事件，全键无冲可识别多个寄存器的值。鼠标的移动依赖参数“回报率”，回报率即CPU获取鼠标相对坐标的频率。