流水线 CPU（Verilog 实现）实验报告

一、CPU 设计方案综述

（一）总体设计概述

本 CPU 为 Verilog 实现的流水线 MIPS-CPU，支持的指令集包括{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、MFC0、MTC0、ERET}，同时支持异常与中断。

（二）关键模块定义

1. CP0

表格 1 CP0 模块端口定义

信号名	方向	描述
Clk	I	时钟信号
Reset	I	复位信号
WE	I	CP0 寄存器写使能信号
Addr	I	读写寄存器地址
DIn	I	CP0 寄存器写入数据
BD	I	是否是延迟槽指令
PC	I	受害指令
ExcCode	I	异常类型
HWInt	I	输入中断信号
IntReq	O	中断请求
EPC	O	EXC 寄存器输出至 NPC
DOut	O	CP0 寄存器输出数据
tbReq	O	有外部中断请求，测试约定
eret	I	是否是 eret 指令

2. Bridge

表格 2 Bridge 模块端口定义

信号名	方向	描述
PrAddr	I	判断读写设备的地址
PrWD	I	DM 读出数据
PrWE	I	写使能信号
DEV1RD	I	DEV1 输出
DEV0RD	I	DEV0 输出
PrRD	O	Bridge 输出
DEVAddr	O	DEV 读写地址
DEVWD	O	DEV 写入数据
DEV0WE	O	DEV0 写使能
DEV1WE	O	DEV1 写使能

（三）IO 设计

1. CPU 模块除了 P6 设计的输入输出以外，在 P7 的设计中添加了 PrRD（I），HWInt(I)，PrWE(O)，macro_pc(O)，tbReq(O)新端口。

其中前三个端口在上述的模块定义中已经提到，下面着重说一下 macro_pc 和 tbReq 端口。

Macro_pc:宏观 PC 表示整个 CPU “宏观”运行指令所对应的 PC 地址。所谓“宏观”指令，表示该指令之前的所有指令序列对 CPU 的更新已完成，该指令及其之后的指令序列对 CPU 的更新未完成。由于本人的 CP0 设计在 M 级，故一般来说宏观 PC 即 M 级 PC 值，当异常出现时，也可能为 E,D,F 级的 PC 值。

tbReq:对于外部中断来说，如果不加处理的话就会出现 interrupt 一直为 1 的情况，也就是评测中的 130/1。所以在 mips 程序中我们需要设计操作以相应中断后将其置 0，我们需要给与 tb 一个反馈信号，也就是 addr 为 32’h7f20 和 byteen 为 4’b1111。所以通过 cp0 中的判断之后引出一个 tbReq 信号用以反馈 tb。

二、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”？（Tips：什么是接口？和我们到现在为止所学的有什么联系？）

提到接口，自然就联系到封装。在一个项目中，也许由于这样那样的原因，我们不能或不需要了解所有细节，或许一个人负责这一部分，另一个人负责那一部分，这时每个人只需要对应相应的接口就可以合作完成任务。比如我们所设计的 CPU，对外表现只有几个输入输出，但是内部封装了许多细节的东西，然而在 tb 看来时无所谓的，它只需要对那几个输入输出传入接收数据即可。

2、BE 部件对所有的外设都是必要的吗？

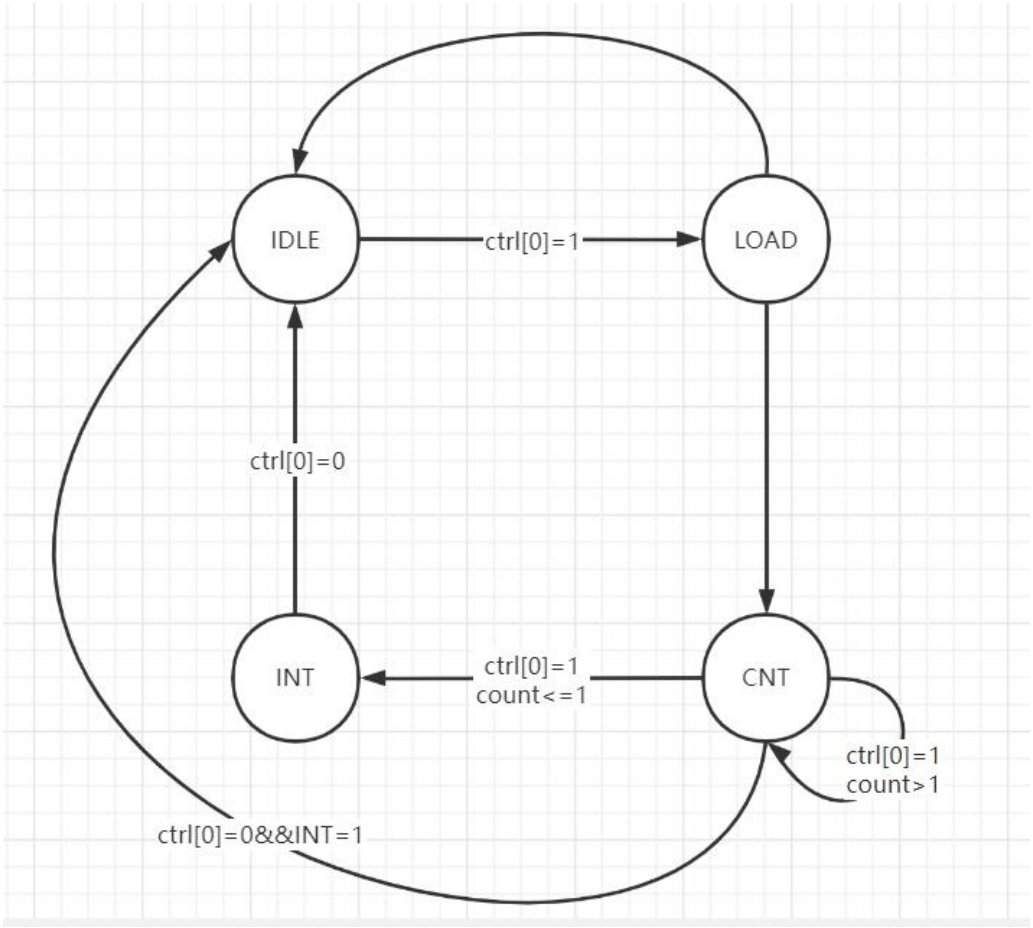
不一定所有的外设都需要通过 BE，可以根据外设的属性功能选择是否经过 BE，如我的设计中 DM 就没有经过 BE。

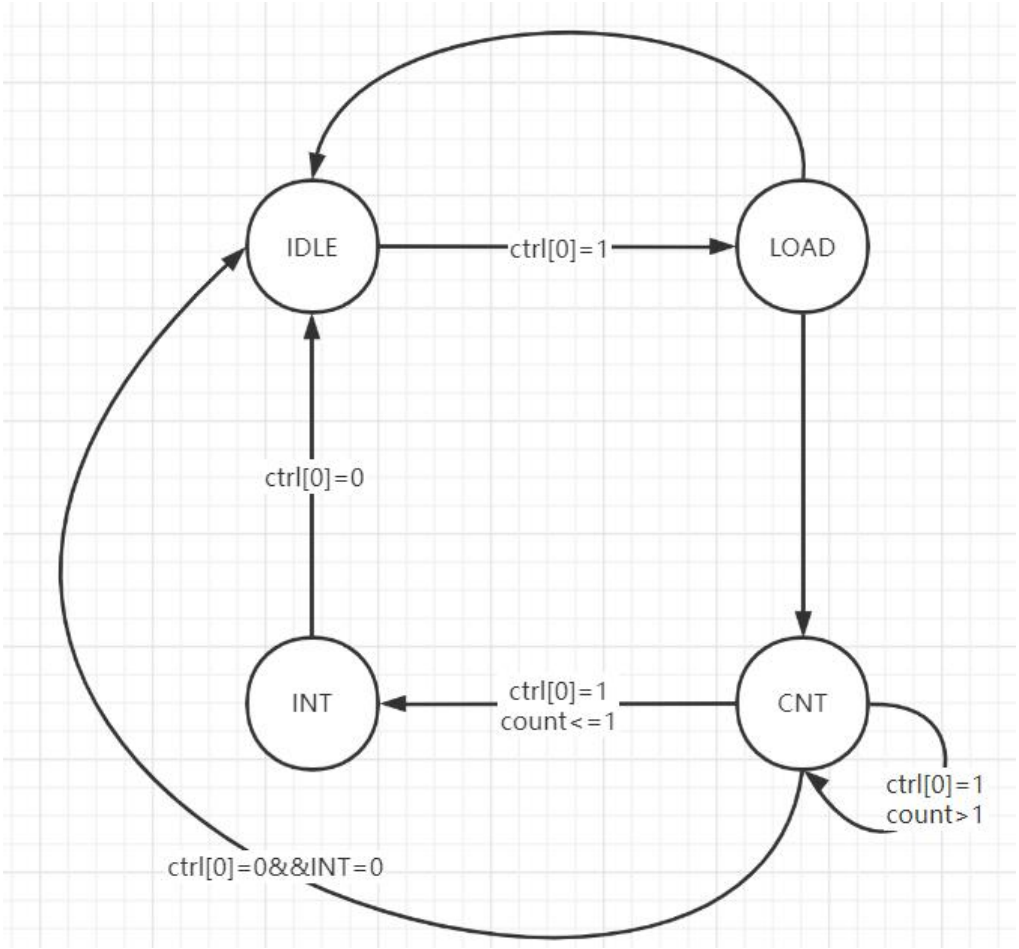
3、请阅读官方提供的定时器源代码，阐述两种中断模式的异同，并分别针对每一种模式绘制状态转移图。

异：一个可以持续输出 interrupt 信号，一个只输出一个周期的 interrupt 信号。

同：都是通过倒计时方式实现中断信号，倒数到 0 时，如果允许中断则输出中断信号。

1：





0：

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：

（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。）

```
.text

ori $t0, $t0, 0xfc01

mtc0 $t0, $12          #SR 赋初值，IM 全为 1
```

```
ori $s0, $0, 0x7f04

ori $s1, $0, 100

sw $s1, 0($s0)



ori $s0, $0, 0x7f00

ori $s1, $0, 9              #1001

sw  $s1, 0($s0)           #将 Ctrl 的 IM 位置 1，说明允许中断，Mode 置 0，
说明是模式 0，Enable 置 1，可以计数



end:

beq $0, $0, end

nop



.ktext 0x4180

ori $s0, $0, 0x7f00

ori $s1, $0, 9              #1001

sw  $s1, 0($s0)           #将 Ctrl 的 IM 位置 1，说明允许中断，Mode 置 0，
说明是模式 0，Enable 置 1，可以计数



ERET:

eret
```

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、鼠标这类的低速设备是通过中断请求的方式进行 IO 操作的。即当键盘上按下一个按键的时候，键盘会发出一个中断信号，中断信号经过中断控制器传到 CPU，然后 CPU 根据不同的中断号执行不同的中断响应程序，然后进行相应的 IO 操作，把按下的按键编码读到寄存器（或者鼠标的操作），最后放入内存中。

键盘→(中断请求信号+键盘对应的输入码<ASCII 码>)→南桥→北桥→CPU 处理器→内存→硬盘

三、测试程序

1. Eret

```
## 测试 eret 后的指令的执行情况 ##
ori $1,1
ori $2,2
ori $3,3
ori $4,4
ori $5,5
ori $6,6
ori $7,7
ori $8,8
ori $9,9
ori $10,10

lw $11,3($0)
addi $12,$12,1

lw $11,3($0)
addi $12,$12,1

lw $11,3($0)
addi $12,$12,1

lw $11,3($0)
addi $12,$12,1
mfc0 $13,$12

lw $11,3($0)
addi $12,$12,1
mfhi $14
mflo $15

lw $11,3($0)
addi $12,$12,1
mfhi $14
mflo $15

lw $11,3($0)
addi $12,$12,1
```

```
lw $11,3($0)

addi $13,$13,13
addi $13,$13,13
test_end:
beq $0,$0,test_end
nop

.ktext 0x4180
mfc0 $28,$14
addi $28,$28,4
mtc0 $28,$14
beq $12,$0,eret0
nop
beq $12,$1,eret1
nop
beq $12,$2,eret2
nop
beq $12,$3,eret3
nop
beq $12,$4,eret4
nop
beq $12,$5,eret5
nop
beq $12,$6,eret6
nop
eret

eret0:
eret
addu $2,$12,$3
sub $3,$2,$5
xori $3,$0,134

eret1:
eret
nop
beq $0,$0,test_end ## M 级 eret 与 D 级跳转 PC 测试
j test_end

eret2:
eret
sw $2,0($0)
lw $4,0($0)
```

```
eret3:
eret
mtc0 $2,$12
mtc0 $3,$12
mtc0 $4,$12

eret4:
eret
mthi $3
mtlo $4

eret5:
eret
mult $4,$5
mult $4,$6

eret6:
eret
nop
j test_end ## M 级 eret 与 D 级跳转 PC 测试

2. Interrupt
## 测试 interrupt interrupt 和延迟槽、暂停等机制的组合 interrupt 和异常的组合 ##

## 测试正常指令 interrupt##
ori $2,$0,64513 ## CP0-SR IE:1 IM:111111
mtc0 $2,$12
ori     $2,$0,132445     ################################################ interrupt
ori $3,$0,2491
addu $2,$3,$2
subu $2,$2,$3  ################################################ interrupt
xori $4,$2,58247
nor $2,$3,$4
andi $5,$2,3122
slt $6,$5,$2
sltiu   $6,$5,23144      ################################################ interrupt
srav $3,$2,$3  ################################################ interrupt
sll $5,$4,4   ################################################ interrupt 连续中断
sb $3,7($0)
```

```
sw $2,0($0)
nop
sw $4,4($0) ############################################ interrupt
lb $2,1($0)
lw $4,8($0)
lhu $3,6($0)
beq    $0,$0,test1       ##############################################
interrupt
addiu $3,$4,1324
test1:
mfc0 $6,$13
bgez $0,test2  ############################################## interrupt
subiu $2,$5,13298
test2:
mfhi $4  ############################################## interrupt
jal test3  ############################################## interrupt
lw $4,0($0)
test3:
ori $2,$0,31745 ## SR IM:011111 IE:1
mtc0      $2,$12      ##      测     试     mtc0     被     中     断
############################################ interrupt
ori $2,$0,64513 ## SR IE:1 IM:111111
mtc0      $2,$12      ##      测     试     mtc0     被     中     断
############################################ interrupt
ori $2,$0,31745 ## SR IM:011111 IE:1
mtc0 $2,$12
ori $2,$0,64513 ## SR IE:1 IM:111111
mtc0      $2,$12      ##      测     试     mtc0     被     中     断
############################################ interrupt
mfc0 $2,$12
mfc0 $2,$13  ############################################ interrupt
mfc0 $2,$14
mfc0 $2,$12
addi $3,$3,1
addu $2,$3,$4  ############################################## interrupt
ori $2,$2,13842
beq $0,$0,test4 ## M 级 interrupt 与 D 级跳转 PC 测试
nop
lw $2,0($0)
test4:
xori $2,$1,40234
subu $2,$4,$1
ori $2,$0,4244 ############################################## interrupt
sw $3,8($0)
```

```
j test5 ## M 级 interrupt 与 D 及跳转 PC 测试
nop
addi $2,$4,1324
test5:
ori $5,$0,5
addu $5,$5,$5
ori $5,$2,13918
nop


## interrupt 与延迟槽 ##
beq $0,$0,test6
nop  ############################################ interrupt & db
test6:
ori $2,$0,0x3108 ## test7
jalr $3,$2
addu $3,$3,$3  ############################################ interrupt
& db
test7:
bne $0,$0,test8 ## 不跳转的延迟槽
addu $2,$2,$2  ############################################ interrupt
& db
test8:
jal test9
subu    $3,$0,$31        ############################################
interrupt & db
test9:
beq $0,$0,test10 ## 跳转到延迟槽
test10:
addu $3,$3,$3 ############################################ interrupt &
db
addu $4,$0,$2
nop


## interrupt 与暂停 ##
ori $2,$0,0x3144 ## test11
sw $2,0($0)
lw $3,0($0)
addu $3,$3,$3 ############################################ interrupt &
stall
lw $4,0($0)
jalr $5,$4 ############################################ interrupt &
stall
```

```
addu $5,$5,$5
test11:
mfc0 $2,$12
xor $3,$0,$2
mfc0 $3,$0
beq     $3,$0,test12     ############################################
interrupt & stall
addu $3,$4,$3
test12:
nop


## interrupt 与异常 ##
li $2,0x316d ## test13 的 PC：0x316c
jr $2          ## PC 字未对齐(非受害指令)
nop
ori $2,$0,20
test13:
ori   $2,$0,2   ##   取   指   异   常   (   PC   字   未   对   齐   )
############################################ interrupt & exc
addi $3,$3,1
teq       $3,$3       ##       未       知       指       令       异       常
############################################ interrupt & exc
addi $3,$3,1
li $2,0xfbc123d
sw $2,1($0) ## sw 字未对齐
addi $3,$3,1
sh        $2,1($0)        ##        sh        字        未        对        齐
############################################ interrupt & exc
addi $3,$3,1
sh $2,0x7f10($0) ## sh 存 Timer
addi $3,$3,1
sb $2,0x7f10($0) ## sb 存 Timer
addi $3,$3,1 ############################################ exc 结束立
刻 interrupt
li $2,0x7fffffff
sw $2,1232($2) ## sw 溢出
addi $3,$3,1
sh $2,1232($2) ## sh 溢出
addi $3,$3,1 ############################################ exc 结束立
刻 interrupt
sb $2,1232($2) ## sb 溢出
addi $3,$3,1
sw $2,0x7f08 ## sw COunt
```

```
addi $3,$3,1
ori $2,$0,0x7f20
sw $2,0($2) ## sw 超出范围
addi $3,$3,1 ############################################## exc 结束立刻 interrupt
ori $2,$0,0x4120
sh $2,0($2) ## sh 超出范围
addi $3,$3,1
ori $2,$0,0x7f1c
sb $2,0($2) ## sb 超出范围
addi $3,$3,1
lw $2,1($0) ## lw 字未对齐
addi $3,$3,1
lh $2,1($0) ## lh 半字未对齐
############################################## interrupt & exc
addi $3,$3,1
lhu $2,1($0) ## lhu 半字未对齐
addi $3,$3,1
lh $2,0x7f00($0) ## lh 取 Timer
addi $3,$3,1
lhu $2,0x7f00($0) ## lhu 取 TImer
addi $3,$3,1 ############################################## exc 结束立刻 interrupt
lb $2,0x7f10($0) ## lb 取 Timer
addi $3,$3,1
lbu $2,0x7f10($0) ## lbu 取 Timer
addi $3,$3,1
li $2,0x7fffffff
lw $2,1232($2) ## lw 溢出
addi $3,$3,1
lh $2,1232($2) ## lh 溢出
addi $3,$3,1
lhu $2,1232($2) ## lw 溢出
addi $3,$3,1
lb $2,1232($2) ## lb 溢出
addi $3,$3,1
lbu $2,1232($2) ## lbu 溢出
addi $3,$3,1
ori $2,$0,0x7f20
lw $2,0($2) ## lw 超出范围
addi $3,$3,1
ori $2,$0,0x3000
lh $2,0($2) ## lh 超出范围
addi $3,$3,1
```

```
ori $2,$0,0x5000
lhu $2,0($2) ## lhu 超出范围
addi $3,$3,1
ori $2,$0,0x7a00
lb $2,0($2) ## lb 超出范围
addi $3,$3,1
ori $2,$0,0x8000
lbu $2,0($2) ## lbu 超出范围
addi $3,$3,1
li $2,0x7fffffff
add $2,$2,$2 ## 加法溢出
addi $3,$3,1
addi $2,$2,1232 ## addi 溢出
addi $3,$3,1
li $4,-0x24201322
sub $2,$4,$2 ## sub 溢出
addi $3,$3,1
db:
li $10,0x7fffffff
beq $0,$0,test14
add          $10,$10,$10          ##          db          异          常
############################################## interrupt db exc
ori $4,$0,4 ## 跳过不执行(是否返回 beq 测试)
test14:
addi $3,$3,1 ############################################## nterrupt
li $10,0x7fffffff
ori $5,$0,0x32e8 ## test15
jalr $6,$5
add          $10,$10,$10          ##          db          异          常
############################################## interrupt db exc
test15:
addi $3,$3,1
mult $5,$10
beq $0,$0,test16
mflo $13 ############################################## interrupt db
stall
test16:
div $5,$2
j test17
mfhi $13 ############################################## interrupt db
stall
test17:
addu $13,$13,$13
test_end:
```

```
beq $0,$0,test_end
nop



.ktext 0x4180
## $30 记录异常次数
## $29 记录中断次数
## $28 调整 EPC
## $27 取 Cause 的 BD 判断
## $26 取 Cause 的 ExcCode
## $25 取 SR
## 规定延迟槽异常为$10 导致的异常（如 add $10,$10,$10 溢出），db_hanlder
将$10 清 0 后返回，$30 加 1
## 规定非延迟槽指令异常，则 exc_handler 将 EPC 对齐后 EPC+4 跳过该指令，
$30 加 1
## 规定中断处理 EPC 不改变 返回原指令，$29 加 1
mfc0 $28,$14
mfc0 $27,$13
mfc0 $26,$13
mfc0 $25,$12
sll $26,$26,25
srl $26,$26,27 ## ExcCode（通过移位消除 IP 和 BD 的影响）
srl $27,$27,31 ## BD
beq $26,$0,interrupt_handler ## ExcCode=0 中断处理
nop
bgtz $27,db_handler ## BD=1
nop
## 延迟槽指令异常同时被外部中断，优先处理外部中断
exc_handler:
andi $28,$28,0xfffffffc ## 字对齐
addi $28,$28,4 ## EPC+4
mtc0 $28,$14
addi $30,$30,1 ## 记录异常次数
eret



db_handler:
or $10,$0,$0 ## $10 清零 确保下次延迟槽不会异常
addi $30,$30,1 ## 记录异常次数
eret
```

```
interrupt_handler:
addi $29,$29,1
eret ##记录中断次数
```

3. 非异常边界

```
 ############ 测试非异常情况下的功能 ###########
 ori $1,24981
 ori $2,0x9123
 sw $1,0($0) ## DM 0
 sw $1,0x1564($0) ## DM middle
 sw $1,0x2ffc($0) ## DM end
 sw $2,0x7f00($0) ## Timer0 CTRL
 sw $2,0x7f04($0) ## Timer0 PRESET
 sw $2,0x7f10($0)  ## Timer1 CTRL
 sw $2,0x7f14($0) ## Timer1 PRESET
 sh $1,0($0) ## DM 0
 sh $1,0x1462($0) ## DM middle
 sh $1,0x2ffe($0) ## DM end
 sb $2,0($0) ## DM 0
 sb $2,0x2331($0) ## DM middle
 sb $2,0x2fff($0) ## DM end
 nop
 lw $1,0($0) ## DM 0
 lw $1,0x1564($0) ## DM middle
 lw $1,0x2ffc($0) ## DM end
 lw $2,0x7f00($0) ## Timer0 CTRL
 lw $2,0x7f04($0) ## Timer0 PRESET
 lw $2,0x7f08($0) ## Timer0 COUNT
 lw $2,0x7f10($0)  ## Timer1 CTRL
 lw $2,0x7f14($0) ## Timer1 PRESET
 lw $2,0x7f18($0) ## Timer1 COUNT
 lh $1,0($0) ## DM 0
 lh $1,0x1462($0) ## DM middle
 lh $1,0x2ffe($0) ## DM end
 lhu $1,0($0) ## DM 0
 lhu $1,0x1462($0) ## DM middle
 lhu $1,0x2ffe($0) ## DM end
 lb $2,0($0) ## DM 0
 lb $2,0x2331($0) ## DM middle
 lb $2,0x2fff($0) ## DM end
 lbu $2,0($0) ## DM 0
 lbu $2,0x2331($0) ## DM middle
 lbu $2,0x2fff($0) ## DM end
 or $1,$0,$0
```

```
 li $2,698742315
 li $3,1448741332
 add $4,$2,$3 ## 结果为最大正数
 li $2,2147483647
 li $3,-2147483647
 add $5,$3,$2 ## 最大正数+对应负数 结果为 0
 li $2,1284291024
 li $3,-1284291024
 add $6,$3,$2 ## 任意正数+对应负数 结果为 0
 li $2,-2147483648
 add $7,$2,$0 ## 结果为最小负数
 li $2,2147450880
 li $3,32767
 addi $8,$2,32767 ## 16 位最大正数 结果为 32 位最大正数
 addi $9,$0,-32768 ## 16 位最小负数
 li $2,2147483647
 li $3,2147483647
 sub $10,$2,$3 ## 最大正数-最大正数
 sub $11,$2,$0 ## 最大正数-0 结果为最大正数
 sub $12,$0,$2 ## 0-最大正数
 li $2,-2147483648
 li $3,-2147483648
 sub $13,$2,$3 ## 最小负数-最小负数
 sub $14,$3,$0 ## 最小负数-0 结果为最小负数
 ################## 测试是否将已知指令误判为 RI ##########
ori $1,0x2411
ori $2,0x948f
sw $1,0($0)
sw $2,0($0)
lb $3,3($0)
lbu $4,3($0)
lh $5,2($0)
lhu $6, 6($0)
lw $7,4($0)
sb $5,7($0)
sh $2,10($0)
sw $3,12($0)
add $8,$1,$0
addu $9,$8,$8
sub $2,$0,$1
subu $7,$6,$5
mult $2,$3
multu $7,$4
div $2,$3
```

```
divu $2,$5
sll $10,$4,13
srl $11,$3,2
sra $12,$6,8
sllv $13,$8,$2
srlv $14,$10,$7
srav $15,$2,$3
and $16,$2,$7
or $17,$16,$9
xor $18,$13,$3
nor $19,$18,$7
addi $20,$0,2348
addiu $21,$20,772
andi $22,$21,2424
ori $23,$18,1324
xori $24,$21,3019
lui $25,10379
slt $22,$21,$23
slti $23,$23,2345
sltiu $24,$1,2487
sltu $21,$3,$2
beq $0,$0,all_test1
nop
all_test1:
bne $0,$1,all_test2
nop
all_test2:
blez $0,all_test3
nop
all_test3:
bgtz $1,all_test4
nop
all_test4:
bltz $6,all_test5
nop
all_test5:
bgez $0,all_test6
nop
all_test6:
j all_test7
nop
all_test7:
jal all_test8
nop
```

```
all_test8:
ori $10,$0,0x3214 ## all_test9
jalr $10
nop
all_test9:
ori $10,$0,0x3220 ## all_test10
jr $10
nop
all_test10:
mfhi $12
mflo $13
mthi $15
mtlo $19
mfc0 $17,$15
mtc0 $0,$12
test_end:
beq $0,$0,test_end
nop
############



.ktext 0x4180
## $30 记录异常次数
## $29 记录中断次数
## $28 调整 EPC
## $27 取 Cause 的 BD 判断
## $26 取 Cause 的 ExcCode
## $25 取 SR
## 规定延迟槽异常为$10 导致的异常（如 add $10,$10,$10 溢出），db_hanlder
将$10 清 0 后返回，$30 加 1
## 规定非延迟槽指令异常，则 exc_handler 将 EPC 对齐后 EPC+4 跳过该指令，
$30 加 1
## 规定中断处理 EPC 不改变 返回原指令，$29 加 1
mfc0 $28,$14
mfc0 $27,$13
mfc0 $26,$13
mfc0 $25,$12
sll $26,$26,25
srl $26,$26,27 ## ExcCode（通过移位消除 IP 和 BD 的影响）
srl $27,$27,31 ## BD
beq $26,$0,interrupt_handler ## ExcCode=0 中断处理
nop
bgtz $27,db_handler ## BD=1
```

```
nop
## 延迟槽指令异常同时被外部中断，优先处理外部中断
exc_handler:
andi $28,$28,0xfffffffc ## 字对齐
addi $28,$28,4 ## EPC+4
mtc0 $28,$14
addi $30,$30,1 ## 记录异常次数
eret



db_handler:
or $10,$0,$0 ## $10 清零 确保下次延迟槽不会异常
addi $30,$30,1 ## 记录异常次数
eret



interrupt_handler:
addi $29,$29,1
eret ##记录中断次数
```

4. 异常测试

```
## 测试各类指令异常
li $2,0x3011 ## test1 的 PC：0x3010
jr $2          ## PC 字未对齐(非受害指令)
nop
ori $2,$0,20
test1:
ori $2,$0,2 ## 取指异常（PC 字未对齐）
addi $3,$3,1
teq $3,$3 ## 未知指令异常
addi $3,$3,1
li $2,0xfbc123d
sw $2,1($0) ## sw 字未对齐
addi $3,$3,1
sh $2,1($0) ## sh 字未对齐
addi $3,$3,1
sh $2,0x7f10($0) ## sh 存 Timer
addi $3,$3,1
sb $2,0x7f10($0) ## sb 存 Timer
addi $3,$3,1
li $2,0x7fffffff
sw $2,1232($2) ## sw 溢出
addi $3,$3,1
```

```
sh $2,1232($2) ## sh 溢出
addi $3,$3,1
sb $2,1232($2) ## sb 溢出
addi $3,$3,1
sw $2,0x7f08 ## sw COunt
addi $3,$3,1
ori $2,$0,0x7f20
sw $2,0($2) ## sw 超出范围
addi $3,$3,1
ori $2,$0,0x4120
sh $2,0($2) ## sh 超出范围
addi $3,$3,1
ori $2,$0,0x7f1c
sb $2,0($2) ## sb 超出范围
addi $3,$3,1
lw $2,1($0) ## lw 字未对齐
addi $3,$3,1
lh $2,1($0) ## lh 半字未对齐
addi $3,$3,1
lhu $2,1($0) ## lhu 半字未对齐
addi $3,$3,1
lh $2,0x7f00($0) ## lh 取 Timer
addi $3,$3,1
lhu $2,0x7f00($0) ## lhu 取 TImer
addi $3,$3,1
lb $2,0x7f10($0) ## lb 取 Timer
addi $3,$3,1
lbu $2,0x7f10($0) ## lbu 取 Timer
addi $3,$3,1
li $2,0x7fffffff
lw $2,1232($2) ## lw 溢出
addi $3,$3,1
lh $2,1232($2) ## lh 溢出
addi $3,$3,1
lhu $2,1232($2) ## lw 溢出
addi $3,$3,1
lb $2,1232($2) ## lb 溢出
addi $3,$3,1
lbu $2,1232($2) ## lbu 溢出
addi $3,$3,1
ori $2,$0,0x7f20
lw $2,0($2) ## lw 超出范围
addi $3,$3,1
ori $2,$0,0x3000
```

```
lh $2,0($2) ## lh 超出范围
addi $3,$3,1
ori $2,$0,0x5000
lhu $2,0($2) ## lhu 超出范围
addi $3,$3,1
ori $2,$0,0x7a00
lb $2,0($2) ## lb 超出范围
addi $3,$3,1
ori $2,$0,0x8000
lbu $2,0($2) ## lbu 超出范围
addi $3,$3,1
li $2,0x7fffffff
add $2,$2,$2 ## 加法溢出
addi $3,$3,1
addi $2,$2,1232 ## addi 溢出
addi $3,$3,1
li $4,-0x24201322
sub $2,$4,$2 ## sub 溢出
addi $3,$3,1
db:
li $10,0x7fffffff
beq $0,$0,test2
add $10,$10,$10 ## db 异常
ori $4,$0,4 ##
test2:
addi $3,$3,1
li $10,0x7fffffff
ori $5,$0,0x318c # test3
jalr $6,$5
add $10,$10,$10 ## db 异常
test3:
addi $3,$3,1
li $10,0x7fffffff
bne $0,$0,test4
addi $10,$10,13924 ## 不跳转 db 异常
nop
test4:
addi $3,$3,1
lw $4,1($0) ## lw 字未对齐
addi $3,$3,1
beq $0,$0,test5 ## M 级异常与 D 级跳转 PC 测试
nop
addu $4,$4,$5
test5:
```

```
teq $1,$2 ## 未知指令
addi $3,$3,1
j test6 ## M 级异常与 D 级跳转 PC 测试
nop
addiu $4,$4,3
xor $4,$2,$4
test6:
addi $4,$2,1002
addi $4,$4,1
test_end:
beq $0,$0,test_end
nop



.ktext 0x4180
## $30 记录异常次数
## $29 记录中断次数
## $28 调整 EPC
## $27 取 Cause 的 BD 判断
## $26 取 Cause 的 ExcCode
## $25 取 SR
## 规定延迟槽异常为$10 导致的异常（如 add $10,$10,$10 溢出），db_hanlder
将$10 清 0 后返回，$30 加 1
## 规定非延迟槽指令异常，则 exc_handler 将 EPC 对齐后 EPC+4 跳过该指令，
$30 加 1
## 规定中断处理 EPC 不改变 返回原指令，$29 加 1
mfc0 $28,$14
mfc0 $27,$13
mfc0 $26,$13
mfc0 $25,$12
sll $26,$26,25
srl $26,$26,27 ## ExcCode（通过移位消除 IP 和 BD 的影响）
srl $27,$27,31 ## BD
beq $26,$0,interrupt_handler ## ExcCode=0 中断处理
nop
bgtz $27,db_handler ## BD=1
nop
## 延迟槽指令异常同时被外部中断，优先处理外部中断
exc_handler:
andi $28,$28,0xfffffffc ## 字对齐
addi $28,$28,4 ## EPC+4
mtc0 $28,$14
addi $30,$30,1 ## 记录异常次数
```

```
eret



db_handler:
or $10,$0,$0 ## $10 清零 确保下次延迟槽不会异常
addi $30,$30,1 ## 记录异常次数
eret


interrupt_handler:
addi $29,$29,1
eret ##记录中断次数
```