流水线CPU（Verilog）实验报告

一、数据通路设计

表格 1 IFU

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | I/O | 位数 | 功能 |
| reset | I | 1 | 复位信号 |
| clk | I | 1 | 时钟信号 |
| PCen | I | 1 | 使能端 |
| Branch | I | 1 | B跳转指令条件是否满足 |
| Jump | I | 2 | 判断跳转指令的类型 |
| IntReq | I | 1 | 中断信号 |
| Eret | I | 1 | 返回信号 |
| EPC | I | 32 | 返回地址 |
| PC\_beq | I | 32 | 如果进行beq跳转的位置 |
| PC\_j | I | 32 | 如果进行j跳转的位置 |
| PC\_jr | I | 32 | 如果进行jr跳转的位置 |
| Instr | O | 32 | 输出指令码 |
| Pc4 | O | 32 | 输出pc+4 |
| Pc8 | O | 32 | 输出pc+8 |

表格 2 Decode

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | I/O | 位数 | 功能 |
| reset | I | 1 | 复位信号 |
| clk | I | 1 | 时钟信号 |
| Instr | I | 32 | 流入D级的信号 |
| PC4 | I | 32 | PC+4 |
| PC8 | I | 32 | PC+8 |
| A3 | I | 5 | WB时写入寄存器的地址 |
| ForwardRSD | I | 2 | 判断写入rs寄存器的值 |
| ForwardRTD | I | 2 | 判断写入rt寄存器的值 |
| AO | I | 31 | EXMEM级寄存器中AO的值 |
| WD | I | 31 | WB级中回写入寄存器的值 | I | 32 | WB时写入寄存器的数据 |
| PC\_8 | I | 31 | J型指令PC的返回值 |
| WE | I | 1 | 寄存器写使能 |
| D1 | O | 32 | 读出数据 |
| D2 | O | 32 | 读出数据 |
| EXT | O | 32 | 扩展后的立即数 |
| Branch | O | 1 | B跳转条件是否满足 |
| Jump | O | 2 | 输出跳转类型 |
| Eret | O | 1 | D级是否为eret |
| PC\_beq | O | 32 | Beq类型应该跳转的位置 |
| PC\_J | O | 32 | J类型应该跳转的位置 |
| PC\_Jr | O | 32 | Jr类型的应该跳转的位置 |
| RegW | O | 1 | 写使能 |

表格 2 Excution

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | I/O | 位数 | 功能 |
| reset | I | 1 | 复位信号 |
| clk | I | 1 | 时钟信号 |
| Instr | I | 32 | E级运行的指令 |
| D1 | I | 32 | 运算数1 |
| D2 | I | 32 | 运算数2 |
| EXT | I | 32 | 扩展后的立即数 |
| ForwardRSE | I | 2 | Rs寄存器旁路转发类型 |
| ForwardRTE | I | 2 | Rt寄存器旁路转发类型 |
| AOM | I | 32 | 数据冲突转发（由M级） |
| WD | I | 32 | 数据冲突转发（由W级） |
| PC\_8 | I | 32 | 数据冲突转发（由M级） |
| RegW\_EX | I | 1 | Decode级判断的寄存器写使能 |
| RegW | O | 1 | Excution级判断后的写使能 |
| busy | O | 1 | 进行乘除法运算的标志 |
| ALUOut | O | 32 | 输出运算结果 |
| DMData | O | 32 | 应该写入DM的值 |

表格 3 DM

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | I/O | 位数 | 功能 |
| reset | I | 1 | 复位信号 |
| clk | I | 1 | 时钟信号 |
| IR\_D | I | 32 | D级指令 |
| IR\_E | I | 32 | E级指令 |
| Instr | I | 32 | M级指令 |
| IR\_W | I | 32 | W级指令 |
| PC\_M | I | 32 | M级PC |
| HWInt | I | 6 | 硬件中断 |
| Add | I | 32 | 读入数据对应的Mem地址 |
| PrRD | I | 32 | 外部设备的输入数据 |
| ForwardRTM | I | 1 | 数据转发判断 |
| ForwardB | I | 32 | A2寄存器的值 |
| WD | I | 32 | W级的转发值 |
| MemW | O | 1 | DM写使能 |
| IntReq | O | 1 | 中断信号 |
| EPC | O | 32 | 返回PC值 |
| Dataout | O | 32 | 输出数据 |
| Datain | O | 32 | 要写入MEM的值 |

表格 4 Write

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | I/O | 位数 | 功能 |
| Instr | I | 32 | W级指令 |
| AO | I | 32 | 写入寄存器的数据（ALU运算结果） |
| DR | I | 32 | 写入寄存器的数据（DM取出的数据） |
| PC8 | I | 32 | 写入寄存器的数据（PC+8） |
| A3 | O | 5 | W级指令寄存器写入地址 |
| RegWD | O | 32 | W级指令寄存器写入数据 |

表格 5 CP0

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | I/O | 位数 | 功能 |
| reset | I | 1 | 复位信号 |
| clk | I | 1 | 时钟信号 |
| IR\_D | I | 32 | D级指令 |
| IR\_E | I | 32 | E级指令 |
| IR\_M | I | 32 | M级指令 |
| IR\_W | I | 32 | W级指令 |
| A1 | I | 5 | GRF的寄存器号 |
| A2 | I | 5 | CP0的寄存器号 |
| DIn | I | 32 | 写入CP0寄存器的值 |
| PC | I | 32 | M级PC |
| HWInt | I | 6 | 外部设备的中断信号 |
| IntReq | O | 1 | CPU主程序中断信号 |
| EPC | O | 32 | 中断程序返回地址 |
| DOut | O | 32 | CP0读出数据 |

表格 6 timer

|  |  |  |  |
| --- | --- | --- | --- |
| 端口 | I/O | 位数 | 功能 |
| reset | I | 1 | 复位信号 |
| clk | I | 1 | 时钟信号 |
| WE | I | 1 | 写使能 |
| ADD\_I | I | 32 | 写入数据地址 |
| DAT\_I | I | 32 | 写入数据 |
| DAT\_O | O | 32 | 读出数据地址 |
| IntReq | O | 1 | CPU主程序中断信号 |
| EPC | O | 32 | 中断程序返回地址 |
| DOut | O | 32 | CP0读出数据 |
| IRQ | O | 1 | Timer的中断信号 |

二、控制器设计

表格 5 控制信号说明

|  |  |
| --- | --- |
| 控制信号 | 作用 |
| RegDst | 选择写入的寄存器：00-Rt 01-Rd 10-31 |
| ALUSA | 选择ALUSA：0-RS 1-sll类指令s |
| ALUSB | 选择ALUB：0-RT 1-扩展的立即数 |
| MemtoReg | 选择写入寄存器的数据：00-ALUout 01-MemD 10-PC |
| RegWrite | 寄存器写使能 |
| MemWrite | DM写使能 |
| Extend | 扩展类型：0-零扩展 01-符号扩展 |
| Jump | 跳转类型：00-+4 01-beq、jal 10-j 11-jr |
| ALUop | ALU运算类型：00-add 01-sub 10-or 11-lui |
| ALUXop | 乘除单元运算类型：000-mult 001-multu 010-div 011-divu 100-mthi 101-mtlo 110-mfhi 111-mflo |
| Bop | B跳转类型：01-beq 10-bne 11-bgezal |
| LOp | Ld指令类型：000-lw 001-lb 010-lbu 011-lh 100-lhu |
| Cal\_r | R型计算类指令 |
| Cal\_i | 立即数计算类指令 |
| B | Beq型指令 |
| Ld | 从DM中Load类型指令 |
| St | 向DM中存数据类型指令 |
| Md | 乘除运算类型指令 |
| Mf | 读HI、LO |
| Mt | 写入HI、LO |
| J | J指令 |
| Jal | Jal指令 |
| Jalr | Jalr指令 |
| Jr | Jr指令 |

三、测试程序

|  |  |  |
| --- | --- | --- |
| 代码 | 地址 | 期望值 |
| ori $8,1 | 3000 | $ 8 <= 00000001 |
| ori $9,4 | 3004 | $ 9 <= 00000004 |
| lui $10,0x1234 | 3008 | $10 <= 12340000 |
| lui $11,0x5678 | 300c | $11 <= 56780000 |
| addu $12,$8,$9 | 3010 | $12 <= 00000005 |
| subu $13,$12,$11 | 3014 | $13 <= a9880005 |
| beq $13,$0,label1 | 3018 | 不跳转 |
| lui $10,1 | 301c | $10 <= 00010000 |
| lui $10,2 | 3020 | $10 <= 00020000 |
| label1: |  |  |
| addu $0,$11,$12 | 3024 | $ 0 <= 56780005 |
| j label2 | 3028 | 跳转至3034 |
| sw $10,0($9) | 302c | \*00000004 <= 00020000  延迟槽 |
| sw $11,0($9) | 3030 |  |
| label2: |  |  |
| lw $14,0($9) | 3034 | $14 <= 00020000 |
| jal label3 | 3038 | $31 <= 00003040 |
| addu $15,$14,$14 | 303c | $15 <= 00040000  延迟槽 |
| addu $15,$15,$15 | 3040 |  |
| label3: |  |  |
| ori $10,1 | 3044 | $10 <= 00020001 |
| ori $16,0x3030 | 3048 | $16 <= 00003030 |
| sll $11,$10,1 | 304c | $11 <= 00040002 |
| srl $12,$10,1 | 3050 | $12 <= 00010000 |
| add $13,$11,$12 | 3054 | $13 <= 00050002 |
| sub $14,$11,$12 | 3058 | $14 <= 00030002 |
| addi $15,$14,0x1234 | 305c | $15 <= 00031236 |
| or $15,$15,$14 | 3060 | $15 <= 00031236 |
| and $17,$15,$14 | 3064 | $17 <= 00030002 |
| xor $8,$15,$14 | 3068 | $8 <= 00001234 |
| addiu $15,$14,-1 | 306c | $15 <= 00030001 |
| subu $17,$15,$16 | 3070 | $17 <= 0002cfd1 |
| jr $16 | 3074 | 跳转至3030 |
| Nop | 3078 | 延迟槽 |

四、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”？

软件的代码转化成为机器码后，被输入到CPU里，CPU连接的外接设备的寄存器，也通过代码改变外接设备的寄存器值，从而对其产生控制。而外接寄存器可以控制外接硬件的工作。达到软件和硬件的协调。

1. DM在我们现在的流水线中处于CPU内部，请你考虑实际上它的位置应该在何处。

DM是主存，应该放在CPU外部，由bridge进行DM和CPU内部cache的数据沟通。

1. BE部件对所有的外设都是必要的吗？

不必要，只支持读取字的外设不需要BE。

1. 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重置初值寄存器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。）

.text

li $t1,1 //t1=1

li $t2,5 //t2=5

li $s1,0x00000c01 //s1=110000000001

mtc0 $s1,$12 //sr允许timer0和timer1的中断

sw $t2,0x7f04 //timer0的preset置5

sw $t1,0x7f00 //timer0 允许计数，不允许中断

li $t1,9 //t1=1001

sw $t1,0x7f00 //timer0以模式0开始计数，允许中断

mtc0 $t1,$13 //等待倒计时

mtc0 $t2,$14 //倒计时

mfc0 $t3,$13 //倒计时

mfc0 $t4,$14 //倒计时

add $t1,$t1,$t1 //倒计时

add $t1,$t1,$t2 //中断

add $t1,$t2,$t2

add $t1,$t1,$t3

add $t1,$t1,$t4

add $t1,$t2,$t4

add $t1,$t2,$t3

.ktext 0x00004180

mfc0 $16,$14 //保存epc到16号寄存器

addiu $16,$16,0

li $t1,9 //t1=9

sw $t1,0x7f00 //timer0允许计数，允许中断

mtc0 $16,$14 //epc的值重新回到EPC寄存器里

eret

1. 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？

当鼠标和键盘输入信号时，会给CPU一个中断信号，若CPU没有屏蔽外部中断信号，则产生中断，CPU对鼠标和键盘输入的信息进行处理，CPU读取数据后返回，继续后面的代码。

1. 你该如何判断你的CP0实现是正确的？请简述你的测试方法。

我把CP0看作是一个在M级的寄存器堆，并且单设立一个cp0写使能的信号。转发暂停时冲突的考虑和Load指令、Store指令是相同的。mfc0可以完全归入Load类，mtc0则同Sw一样考虑了rt的转发暂停。并根据两条指令所需要和产生寄存器值的阶段，制造冲突进行测试。

1. 试解释“异常嵌套”。

异常嵌套是指当CPU发生中断，执行异常处理的时候，再次遇到中断的情况，每次中断的时候把数据存入堆栈。

1. **你的CP0的设计有何独特与精妙之处？你对MIPS架构有了何种认识？**

**我把cp0放在了M级，把mtc0和mfc0在操作时看成lw、sw指令。产生中断时，M级及之前级的指令不执行，EPC根据情况保存前面或者后面一条指令的周期。返回时，eret在D级被感知后，会进行PC跳转，没有延时槽所以自身产生一个nop，之后eret走到M级的时候，对CP0中SR寄存器进行允许中断的恢复，进而可以保证中断程序在eret之前不会对下一次中断有所影响。CP0是站在更高的位置上调控整个CPU与外界的配合。**

命令行：java -jar mars4\_5.jar a db mc CompactDataAtZero dump 0x00004180-0x00004ffc HexText E:\exc.txt E:\eh.asm