流水线CPU设计说明文档

一、数据通路设计

1、数据通路具体实现

见表格文档

二、模块规格

1、PC

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 同步复位信号 |
| npc[31:0] | I | 32位存储数据 |
| pc[31:0] | O | 输出32位PC存储的数据 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当reset为高电平时PC同步复位为0x0000\_3000 |
| 2 | 写入npc | 时钟上升沿写入npc端口传入的值 |

2、NPC

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| pc[31:0] | I | 当前指令PC值 |
| NPCOp[1:0] | I | 计算NPC功能选择 |
| Imm[25:0] | I | 26位立即数 |
| ra[31:0] | I | 32位寄存器值 |
| npc[31:0] | O | 输出32位NPC的值 |
| pc4[31:0] | O | 输出32位PC+4的值 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 次地址计算 | NPCOp=00，计算输出顺序地址（PC+4）  NPCOp=01，计算输出beq地址  NPCOp=10，计算输出jal地址  NPCOp=11，计算输出jr地址 |

3、IM

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| addr[31:0] | I | 输入32位地址 |
| Instr[31:0] | O | 输出32位指令 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 取出指令 | 根据addr的值从IM中取出指令 |

4、GRF

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| A1[4:0] | I | 第1个读出寄存器的编号 |
| A2[4:0] | I | 第2个读出寄存器的编号 |
| A3[4:0] | I | 写入寄存器的编号 |
| RD1[31:0] | O | A1指向寄存器的值 |
| RD2[31:0] | O | A2指向寄存器的值 |
| WD[31:0] | I | 写入寄存器的值 |
| WE | I | 写入使能 |
| clk | I | 时钟信号 |
| reset | I | 同步复位信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | reset=1时在时钟上升沿所有寄存器复位为0x0000\_0000 |
| 2 | 读出寄存器 | A1和A2对应的32位寄存器值分别通过RD1和RD2输出 |
| 3 | 写入寄存器 | WE=1时，WD的值写入A3所指的寄存器 |

5、ALU

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| A[31:0] | I | 第1个32位操作数 |
| B[31:0] | I | 第2个32位操作数 |
| C[31:0] | O | 32位计算结果 |
| ALUOp[1:0] | I | ALU功能选择 |
| zero | O | A和B相等比较结果 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | ALU数学逻辑计算 | ALUOp=00，C=A+B  ALUOp=01，C=A-B  ALUOp=10，C=A|B  ALUOp=11，保留 |
| 2 | 相等比较 | zero=1，A=B  zero=0，AB |

6、DM

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| addr[31:0] | I | 地址输入 |
| WD[31:0] | I | 写入数据 |
| RD[31:0] | O | 读出数据 |
| WE | I | 写入使能 |
| clk | I | 时钟信号 |
| reset | I | 异步复位信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | RST=1时，在时钟上升沿RAM复位为零 |
| 2 | 读出数据 | 将addr所表示的内存地址中的值读出 |
| 3 | 写入数据 | WE=1时，将WD的值写入addr所表示的内存地址中 |

7、EXT

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| Imm[15:0] | I | 16位立即数输入 |
| out[31:0] | O | 32位扩展结果输出 |
| EXTOp[1:0] | I | 扩展功能信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 扩展立即数 | EXTOp=00，进行零扩展  EXTOp=01，进行符号扩展  EXTOp=10，进行LUI扩展  EXTOp=11，保留 |

8、CMP

（1）端口定义

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| A[31:0] | I | 32位数据输入 |
| B[31:0] | I | 32位数据输入 |
| equal | O | 相等信号输出 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 相等比较 | A==B, equal = 1  A!=B, equal = 0 |

9、multdiv

（1）端口定义

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| A[31:0] | I | 32位数据输入 |
| B[31:0] | I | 32位数据输入 |
| HiLoOp[2:0] | I | 乘除法运算信号 |
| C[31:0] | O | 32位数据输出 |
| busy | O | 模块忙信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | multdiv运算 | `MULT\_OP: 乘法运算  `MULTU\_OP: 无符号乘法运算  `DIV\_OP: 除法运算  `DIVU\_OP: 无符号除法运算 |
| 2 | HI、LO读取 | `MFHI\_OP: HI读取  `MFLO\_OP: LO读取 |
| 3 | HI、LO写入 | `MTHI\_OP: HI写入  `MTLO\_OP: LO写入 |

10、DM\_DECODE

（1）端口定义

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| Instr[31:0] | I | 32位指令输入 |
| byte[1:0] | I | 2位地址数据输入 |
| BE[3:0] | O | 字节写使能输出 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | DM字节使能译码 | sb: byte==00 ? BE=0001  byte==01 ? BE=0010  byte==10 ? BE=0100  byte==11 ? BE=1000  sh: byte==0X ? BE=0011  byte==1X ? BE=1100  sw: BE=1111 |

11、DM\_EXT

（1）端口定义

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| Din[31:0] | I | 32位数据输入 |
| DMEXTOp[2:0] | I | DM输出扩展选择运算 |
| byte[1:0] | I | 字节输入 |
| Dout[31:0] | O | 32位数据输出 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 扩展选择运算 | `LB\_OP, `LBU\_OP,  `LH\_OP, `LHU\_OP,  `LW\_OP |

12、CP0

（1）端口定义

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 同步复位信号 |
| WE | I | 写使能信号 |
| EXLSet | I | SR寄存器EXL位置位 |
| EXLClr | I | SR寄存器EXL位复位 |
| isDB | I | 判断指令是否属于延迟槽 |
| isEret | I | 判断指令是否为eret |
| A1[4:0] | I | 读寄存器地址 |
| A2[4:0] | I | 写寄存器地址 |
| Din[31:0] | I | 32位写入数据 |
| pc[31:0] | I | 当前指令PC的值 |
| ExcCode[4:0] | I | 5位异常码输入 |
| HWInt[7:2] | I | 中断位输入 |
| IntReq | O | 产生中断或异常信号输出 |
| epc[31:0] | O | EPC寄存器数据输出 |
| Dout[31:0] | O | 32位数据输出 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读取寄存器 | Dout端口输出A1所指向的寄存器的值 |
| 2 | 写入寄存器 | 当写使能有效时，A2指向寄存器将被写入Din端口的值 |
| 3 | 异常/中断信号产生 | 当MEM级接收到外部设备的中断信号，MEM级isEret信号有效，或MEM级指令产生异常时，IntReq为1 |
| 4 | EXL的置/复位 | EXLSet=1时，时钟上升沿EXL=1  EXLClr=1时，时钟上升沿EXL=0  二者同时为1时，时钟上升沿EXL=0 |

13、Bridge

（1）端口定义

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| PrAddr[31:0] | I | 32位数据输入 |
| PrWD[31:0] | I | DM输出扩展选择运算 |
| PrBE31:0] | I | 字节输入 |
| PrWE | I | 32位数据输出 |
| IntTimer0 | I | Timer0中断信号输入 |
| IntTimer1 | I | Timer1中断信号输入 |
| IntOuter | I | 外部中断信号输入 |
| DEV0\_RD[31:0] | I | 设备0读取数据输入 |
| DEV1\_RD[31:0] | I | 设备1读取数据输入 |
| PrRD[31:0] | O | 向处理器发送设备读取数据 |
| DEV\_Addr[31:0] | O | 向设备发送访问地址信息 |
| DEV\_WD[31:0] | O | 向设备发送写入数据 |
| HWInt[7:2] | O | 输出中断位信号 |
| DEV0\_WE | O | 设备0写使能信号 |
| DEV1\_WE | O | 设备1写使能信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读取设备数据 | PrAddr发送给各个设备，并接受各个设备数据，选择与PrAddr相符的设备数据，用PrRD发送出去 |
| 2 | 写入设备寄存器 | 当PrWE有效时，根据PrAddr选择的有效使能设备端口，根据PrBE信号将PrWD从端口DEV\_WD发送出去 |
| 3 | 向处理器发送设备中断信息 | 收集外部设备中断信号，并将其对应位置位，从HWInt端口发送 |

14、Controller

（见第二部分：控制器设计）

三、控制器设计

1、端口定义及功能说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| opcode[5:0] | I | Instr[31:26] |
| funct[5:0] | I | Instr[5:0] |
| equal | I | 相等比较信号 |
| greater | I | 大于比较信号 |
| less | I | 小于比较信号 |
| DMWr | O | DM写使能信号 |
| ALUOp[3:0] | O | ALU运算信号 |
| HiLoOp[2:0] | O | MultDiv运算信号 |
| ASel | O | ALU的B端MUX选择信号：  0：寄存器RD2值  1：shamt数据 |
| BSel | O | ALU的B端MUX选择信号：  0：寄存器RD2值  1：EXT扩展数据 |
| EXTOp[1:0] | O | EXT扩展信号 |
| RFWr | O | GRF写使能信号 |
| WDSel[1:0] | O | 寄存器的WD端MUX选择信号：  00：ALU运算结果  01：DM输出数据  10：PC + 4  11：保留 |
| WRSel[1:0] | O | 寄存器的A3端MUX选择信号：  00：Instr[20:16]  01：Instr[15:11]  10：0x1f  11：0x00 |
| NPCOp[1:0] | O | IFU中NPC运算信号 |

2、真值表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Instr | addu | subu | ori | lw | sw | beq | lui | j | jal | jr |
| Op | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 | 000010 | 000011 | 000000 |
| Func | 100001 | 100011 | NA | NA | NA | NA | NA | NA | NA | 001000 |
| DMWr | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| ALUOp | ADDU | SUBU | OR | ADDU | ADDU | XX | ADDU | XX | XX | XX |
| BSel | 0 | 0 | 1 | 1 | 1 | X | 1 | X | X | X |
| EXTOp | XX | XX | 00 | 01 | 01 | XX | 10 | XX | XX | XX |
| RFWr | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| WDSel | 00 | 00 | 00 | 01 | XX | XX | 00 | XX | 10 | XX |
| WRSel | 01 | 01 | 00 | 00 | XX | XX | 00 | XX | 10 | XX |
| NPCOp | 00 | 00 | 00 | 00 | 00 | 0 Zero | 00 | 10 | 10 | 11 |
| Instr | bgez | bgtz | blez | bltz | bne | jalr | slt | slti | sltiu | sltu |
| Op | 000001 | 000111 | 000110 | 000001 | 000101 | 000000 | 000000 | 001010 | 001011 | 000000 |
| Func | NA | NA | NA | NA | NA | 001001 | 101010 | NA | NA | 101011 |
| DMWr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ALUOp | XX | XX | XX | XX | XX | XX | SLT | SLT | SLTU | SLTU |
| BSel | X | X | X | X | X | X | 0 | 1 | 1 | 0 |
| EXTOp | XX | XX | XX | XX | XX | XX | XX | 01 | 01 | XX |
| RFWr | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| WDSel | XX | XX | XX | XX | XX | XX | 00 | 00 | 00 | 00 |
| WRSel | XX | XX | XX | XX | XX | XX | 01 | 00 | 00 | 01 |
| NPCOp | 0 Zero | 0 Zero | 0 Zero | 0 Zero | 0 Zero | 11 | 00 | 00 | 00 | 00 |
| Instr | addi | addiu | andi | xori | sll | sllv | sra | srav | srl | srlv |
| Op | 001000 | 001001 | 001100 | 001110 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 |
| Func | 100001 | 100011 | NA | NA | 000000 | 000100 | 000011 | 000111 | 000010 | 000110 |
| DMWr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ALUOp | ADD | ADDU | AND | XOR | SLL | SLL | SRA | SRA | SRL | SRL |
| BSel | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| EXTOp | 01 | 01 | 00 | 00 | XX | XX | XX | XX | XX | XX |
| RFWr | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| WDSel | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| WRSel | 00 | 00 | 00 | 00 | 01 | 01 | 01 | 01 | 01 | 01 |
| NPCOp | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| Instr | lh | lhu | lb | lbu | sh | sb | and | nor | or | xor |
| Op | 100001 | 100101 | 100000 | 100100 | 101001 | 101000 | 000000 | 000000 | 000000 | 000000 |
| Func | NA | NA | NA | NA | NA | NA | 100100 | 100111 | 100101 | 100110 |
| DMWr | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| ALUOp | ADDU | ADDU | ADDU | ADDU | ADDU | ADDU | AND | NOR | OR | XOR |
| BSel | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| EXTOp | 01 | 01 | 01 | 01 | 01 | 01 | XX | XX | XX | XX |
| RFWr | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| WDSel | 01 | 01 | 01 | 01 | XX | XX | 00 | 00 | 00 | 00 |
| WRSel | 00 | 00 | 00 | 00 | XX | XX | 01 | 01 | 01 | 01 |
| NPCOp | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| Instr | add | sub | mult | multu | div | divu | mfhi | mflo | mthi | mtlo |
| Op | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 | 000000 |
| Func | 100000 | 100010 | 011000 | 011001 | 011010 | 011011 | 010000 | 101010 | 010001 | 010011 |
| DMWr | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ALUOp | ADD | SUB | MULT | MULTU | DIV | DIVU | MFHI | MFLO | MTHI | MTLO |
| BSel | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| EXTOp | XX | XX | XX | XX | XX | XX | XX | XX | XX | XX |
| RFWr | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| WDSel | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| WRSel | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 | 01 |
| NPCOp | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

3、暂停控制设计表

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | |  | ID/EX级 | | | | EX/MEM级 | | | | |
| 指令类型 | Calc\_R | Calc\_I | Load | JAL | Calc\_R | Calc\_I | Load | JAL | busy |
| 目标寄存器 | rd | rt | rt | $31 | rd | rt | rt | $31 |  |
| Tnew | 1 | 1 | 2 | 0 | 0 | 0 | 1 | 0 |  |
| 指令类型 | 源寄存器 | Tuse |  |  |  |  |  |  |  |  |  |  |
| Calc\_R | rs/rt | 1 |  |  |  | Stall |  |  |  |  |  |  |
| Calc\_I | rs | 1 |  |  |  | Stall |  |  |  |  |  |  |
| Load | rs | 1 |  |  |  | Stall |  |  |  |  |  |  |
| Beq | rs/rt | 0 |  | Stall | Stall | Stall |  |  |  | Stall |  |  |
| Store | rs | 1 |  |  |  | Stall |  |  |  |  |  |  |
| rt | 2 |  |  |  |  |  |  |  |  |  |  |
| JR | rs | 0 |  | Stall | Stall | Stall |  |  |  | Stall |  |  |
| MDuse |  |  |  |  |  |  |  |  |  |  |  | stall |

4、转发控制设计表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 流水级 | | 源寄存器 | | 涉及指令 | | | | 转发MUX | | | | | | | 控制信号 | | | | 输入0 | |
| IF/ID | | rs | | branch, jreg | | | | MUX\_ForwardRS\_ID | | | | | | | ForwardRS\_ID | | | | GRF.RD1 | |
| rt | | branch | | | | MUX\_ForwardRT\_ID | | | | | | | ForwardRT\_ID | | | | GRF.RD2 | |
| ID/EX | | rs | | calc\_r, calc\_i,load,store | | | | MUX\_ForwardRS\_EX | | | | | | | ForwardRS\_EX | | | | RegData1\_ID\_EX | |
| rt | | calc\_r | | | | MUX\_ForwardRT\_EX | | | | | | | ForwardRT\_EX | | | | RegData2\_ID\_EX | |
| EX/MEM | | rt | | store | | | | MUX\_ForwardRT\_MEM | | | | | | | ForwardRT\_MEM | | | | DM\_WD\_EX\_MEM | |
|  | | |  | | ID/EX级 | | | | | | | | | | |
| 指令类型 | | Calc\_R | | | | Calc\_I | Load | | | JAL | | |
| 目标寄存器 | | rd | | | | rt | rt | | | $31 | | |
| Tnew | | 1 | | | | 1 | 2 | | | 0 | | |
| 流水级 | 源寄存器 | |  | |  | | | |  |  | | |  | | |
| IF/ID | rs | |  | | / | | | | / | / | | | PC8\_EX | | |
| rt | |  | | / | | | | / | / | | | PC8\_EX | | |
| ID/EX | rs | |  | | / | | | | / | / | | | / | | |
| rt | |  | | / | | | | / | / | | | / | | |
| EX/MEM | rt | |  | | / | | | | / | / | | | / | | |
|  | | | EX/MEM级 | | | | | | | | | | | | | |
| Calc\_R | | | Calc\_I | | | | | Load | | | JAL | | |
| rd | | | rt | | | | | rt | | | $31 | | |
| 0 | | | 0 | | | | | 1 | | | 0 | | |
| 流水级 | 源寄存器 | |  | | |  | | | | |  | | |  | | |
| IF/ID | rs | | ALUout\_MEM | | | ALUout\_MEM | | | | | / | | | PC8\_MEM | | |
| rt | | ALUout\_MEM | | | ALUout\_MEM | | | | | / | | | PC8\_MEM | | |
| ID/EX | rs | | ALUout\_MEM | | | ALUout\_MEM | | | | | / | | | PC8\_MEM | | |
| rt | | ALUout\_MEM | | | ALUout\_MEM | | | | | / | | | PC8\_MEM | | |
| EX/MEM | rt | | / | | | / | | | | | / | | | / | | |
|  | | | MEM/WB级 | | | | | | | | | | | | | | | | |
| Calc\_R | | | | Calc\_I | | | | | Load | | | | | | JAL | |
| rd | | | | rt | | | | | rt | | | | | | $31 | |
| 0 | | | | 0 | | | | | 0 | | | | | | 0 | |
| 流水级 | 源寄存器 | |  | | | |  | | | | |  | | | | | |  | |
| IF/ID | rs | | / | | | | / | | | | | / | | | | | | / | |
| rt | | / | | | | / | | | | | / | | | | | | / | |
| ID/EX | rs | | RegWriteData\_WB | | | | RegWriteData\_WB | | | | | RegWriteData\_WB | | | | | | RegWriteData\_WB | |
| rt | | RegWriteData\_WB | | | | RegWriteData\_WB | | | | | RegWriteData\_WB | | | | | | RegWriteData\_WB | |
| EX/MEM | rt | | RegWriteData\_WB | | | | RegWriteData\_WB | | | | | RegWriteData\_WB | | | | | | RegWriteData\_WB | |

四、测试模块

1、测试A

测试程序：

.text 0x00003000

li $s0, 0x80000010

li $s1, 0x80000003

add $t0, $s0, $s1

addu $t1, $s0, $s1

li $s0, 0x00000010

sub $t0, $s1, $s0

li $s0, 0x7fffffff

addi $t0, $s0, 1

mtlo $0

lw $t0, 3($0)

mult $s0, $s1

mflo $a0

lh $t0, 1($0)

li $s0, 0x00003000

lb $t0, 0($s0)

div $s0, $s1

sw $t0, 1($0)

mflo $a0

sh $t0, 3($0)

li $s0, 0x00007f10

mtlo $s0

sb $t0, 0($s0)

mflo, $t0

sw $t0, 8($s0)

li $t0, 0x00003071

jr $t0

mflo, $t0

lwl $t1, -100($t2)

go:

beq $0, $0, go

nop

.ktext 0x00004180

mfc0 $k0, $13

mfc0 $k1, $14

srl $k0, $k0, 2

andi $k0, $k0, 0x001f

beq $k0, $0, return

nop

addiu $k1, $k1, 4

srl $k1, $k1, 2

sll $k1, $k1, 2

mtc0 $k1, $14

return:

eret

mult $k0, $k1

期望结果：

@00003000: $ 1 <= 80000000

@00003004: $16 <= 80000010

@00003008: $ 1 <= 80000000

@0000300c: $17 <= 80000003

@00004180: $26 <= 00000030

@00004184: $27 <= 00003010

@00004188: $26 <= 0000000c

@0000418c: $26 <= 0000000c

@00004198: $27 <= 00003014

@0000419c: $27 <= 00000c05

@000041a0: $27 <= 00003014

@00003014: $ 9 <= 00000013

@00003018: $16 <= 00000010

@00004180: $26 <= 00000030

@00004184: $27 <= 0000301c

@00004188: $26 <= 0000000c

@0000418c: $26 <= 0000000c

@00004198: $27 <= 00003020

@0000419c: $27 <= 00000c08

@000041a0: $27 <= 00003020

@00003020: $ 1 <= 7fff0000

@00003024: $16 <= 7fffffff

@00004180: $26 <= 00000030

@00004184: $27 <= 00003028

@00004188: $26 <= 0000000c

@0000418c: $26 <= 0000000c

@00004198: $27 <= 0000302c

@0000419c: $27 <= 00000c0b

@000041a0: $27 <= 0000302c

@00004180: $26 <= 00000010

@00004184: $27 <= 00003030

@00004188: $26 <= 00000004

@0000418c: $26 <= 00000004

@00004198: $27 <= 00003034

@0000419c: $27 <= 00000c0d

@000041a0: $27 <= 00003034

@00003038: $ 4 <= fffffffd

@00004180: $26 <= 00000010

@00004184: $27 <= 0000303c

@00004188: $26 <= 00000004

@0000418c: $26 <= 00000004

@00004198: $27 <= 00003040

@0000419c: $27 <= 00000c10

@000041a0: $27 <= 00003040

@00003040: $16 <= 00003000

@00004180: $26 <= 00000010

@00004184: $27 <= 00003044

@00004188: $26 <= 00000004

@0000418c: $26 <= 00000004

@00004198: $27 <= 00003048

@0000419c: $27 <= 00000c12

@000041a0: $27 <= 00003048

@00004180: $26 <= 00000014

@00004184: $27 <= 0000304c

@00004188: $26 <= 00000005

@0000418c: $26 <= 00000005

@00004198: $27 <= 00003050

@0000419c: $27 <= 00000c14

@000041a0: $27 <= 00003050

@00003050: $ 4 <= 00000000

@00004180: $26 <= 00000014

@00004184: $27 <= 00003054

@00004188: $26 <= 00000005

@0000418c: $26 <= 00000005

@00004198: $27 <= 00003058

@0000419c: $27 <= 00000c16

@000041a0: $27 <= 00003058

@00003058: $16 <= 00007f10

@00004180: $26 <= 00000014

@00004184: $27 <= 00003060

@00004188: $26 <= 00000005

@0000418c: $26 <= 00000005

@00004198: $27 <= 00003064

@0000419c: $27 <= 00000c19

@000041a0: $27 <= 00003064

@00003064: $ 8 <= 00007f10

@00004180: $26 <= 00000014

@00004184: $27 <= 00003068

@00004188: $26 <= 00000005

@0000418c: $26 <= 00000005

@00004198: $27 <= 0000306c

@0000419c: $27 <= 00000c1b

@000041a0: $27 <= 0000306c

@0000306c: $ 8 <= 00003071

@00003074: $ 8 <= 00007f10

@00004180: $26 <= 00000010

@00004184: $27 <= 00003070

@00004188: $26 <= 00000004

@0000418c: $26 <= 00000004

@00004198: $27 <= 00003074

@0000419c: $27 <= 00000c1d

@000041a0: $27 <= 00003074

@00003074: $ 8 <= 00007f10

@00004180: $26 <= 00000028

@00004184: $27 <= 00003078

@00004188: $26 <= 0000000a

@0000418c: $26 <= 0000000a

@00004198: $27 <= 0000307c

@0000419c: $27 <= 00000c1f

@000041a0: $27 <= 0000307c

2、测试B

测试程序：//interrupt occurs in 0x3018, 0x301c, 0x3020, 0x3028, 0x302c, 0x3030, 0x3058, and only once.

.text 0x00003000

ori $v0, $0, 0xff11

mtc0 $v0, $12

li $s0, 0x80000010

li $s1, 0x80000003

add $t0, $s0, $s1

lw $t0, 3($0)

sw $t0, 1($0)

sw $t0, 8($s0)

li $t0, 0x00003031

jr $t0

mflo, $t0

lwl $t1, -100($t2)

go:

beq $0, $0, go

nop

.ktext 0x00004180

mfc0 $k0, $13

mfc0 $k1, $14

srl $k0, $k0, 2

andi $k0, $k0, 0x001f

beq $k0, $0, return

nop

addiu $k1, $k1, 4

srl $k1, $k1, 2

sll $k1, $k1, 2

mtc0 $k1, $14

return:

eret

mult $k0, $k1

期望结果：

@00003000: $ 2 <= 0000ff11

@00003008: $ 1 <= 80000000

@0000300c: $16 <= 80000010

@00003010: $ 1 <= 80000000

@00003014: $17 <= 80000003

@00004180: $26 <= 00001000

@00004184: $27 <= 00003018

@00004188: $26 <= 00000400

@0000418c: $26 <= 00000000

@00004180: $26 <= 00000030

@00004184: $27 <= 00003018

@00004188: $26 <= 0000000c

@0000418c: $26 <= 0000000c

@00004198: $27 <= 0000301c

@0000419c: $27 <= 00000c07

@000041a0: $27 <= 0000301c

@00004180: $26 <= 00001000

@00004184: $27 <= 0000301c

@00004188: $26 <= 00000400

@0000418c: $26 <= 00000000

@00004180: $26 <= 00000010

@00004184: $27 <= 0000301c

@00004188: $26 <= 00000004

@0000418c: $26 <= 00000004

@00004198: $27 <= 00003020

@0000419c: $27 <= 00000c08

@000041a0: $27 <= 00003020

@00004180: $26 <= 00001000

@00004184: $27 <= 00003020

@00004188: $26 <= 00000400

@0000418c: $26 <= 00000000

@00004180: $26 <= 00000014

@00004184: $27 <= 00003020

@00004188: $26 <= 00000005

@0000418c: $26 <= 00000005

@00004198: $27 <= 00003024

@0000419c: $27 <= 00000c09

@000041a0: $27 <= 00003024

@00004180: $26 <= 00000014

@00004184: $27 <= 00003024

@00004188: $26 <= 00000005

@0000418c: $26 <= 00000005

@00004198: $27 <= 00003028

@0000419c: $27 <= 00000c0a

@000041a0: $27 <= 00003028

@00004180: $26 <= 00001000

@00004184: $27 <= 00003028

@00004188: $26 <= 00000400

@0000418c: $26 <= 00000000

@00003028: $ 8 <= 00003031

@00004180: $26 <= 80001000

@00004184: $27 <= 0000302c

@00004188: $26 <= 20000400

@0000418c: $26 <= 00000000

@00003030: $ 8 <= 00000000

@00004180: $26 <= 00000010

@00004184: $27 <= 00003030

@00004188: $26 <= 00000004

@0000418c: $26 <= 00000004

@00004198: $27 <= 00003034

@0000419c: $27 <= 00000c0d

@000041a0: $27 <= 00003034

@00004180: $26 <= 00001000

@00004184: $27 <= 00003034

@00004188: $26 <= 00000400

@0000418c: $26 <= 00000000

@00004180: $26 <= 00000028

@00004184: $27 <= 00003034

@00004188: $26 <= 0000000a

@0000418c: $26 <= 0000000a

@00004198: $27 <= 00003038

@0000419c: $27 <= 00000c0e

@000041a0: $27 <= 00003038

3、测试C

测试程序：

.text 0x00003000

li $s0, 0x00007f00

li $t0, 0x00000010

li $t1, 0x00000009

sw $t0, 4($s0)

sw $t1, 0($s0)

li $s0, 0x00007f10

li $t0, 0x00000111

li $t1, 0x0000000b

sw $t0, 4($s0)

sw $t1, 0($s0)

ori $v0, $0, 0xff11

mtc0 $v0, $12

go:

beq $0, $0, go

nop

.ktext 0x00004180

mfc0 $k0, $13

mfc0 $k1, $14

srl $k0, $k0, 2

andi $k0, $k0, 0x001f

beq $k0, $0, return

nop

addiu $k1, $k1, 4

mtc0 $k1, $14

eret

return:

mfc0 $k0, $13

srl $k0, $k0, 10

andi $k0, $k0, 0x003f

li $t0, 1

li $t1, 2

li $t2, 4

or $v0, $t0, $k0

beq $v0, $0, pass1

nop

pass1:

or $v0, $t0, $k0

beq $v0, $0, pass2

nop

li $s0, 0x00007f00

lw $t0, 0($s0)

ori $t0, $t0, 0x0001

sw $t0, 0($s0)

j pass3

pass2:

or $v0, $t0, $k0

beq $v0, $0, pass3

nop

lw $a0, 0($0)

addiu $a0, $a0, 1

sw $a0, 0($0)

j pass3

pass3:

eret

ori $s0, $0, 0xffff

期望结果：

@00003000: $16 <= 00007f00

@00003004: $ 8 <= 00000010

@00003008: $ 9 <= 00000009

@00003014: $16 <= 00007f10

@00003018: $ 8 <= 00000111

@0000301c: $ 9 <= 0000000b

@00003028: $ 2 <= 0000ff11

@00004180: $26 <= 80000400

@00004184: $27 <= 00003030

@00004188: $26 <= 20000100

@0000418c: $26 <= 00000000

@000041a4: $26 <= 80000400

@000041a8: $26 <= 00200001

@000041ac: $26 <= 00000001

@000041b0: $ 8 <= 00000001

@000041b4: $ 9 <= 00000002

@000041b8: $10 <= 00000004

@000041bc: $ 2 <= 00000001

@000041c8: $ 2 <= 00000001

@000041d4: $16 <= 00007f00

@000041d8: $ 8 <= 00000008

@000041dc: $ 8 <= 00000009

@000041e8: $ 2 <= 00000009

@00004180: $26 <= 00000400

@00004184: $27 <= 00003030

@00004188: $26 <= 00000100

@0000418c: $26 <= 00000000

@000041a4: $26 <= 00000400

@000041a8: $26 <= 00000001

@000041ac: $26 <= 00000001

@000041b0: $ 8 <= 00000001

@000041b4: $ 9 <= 00000002

@000041b8: $10 <= 00000004

@000041bc: $ 2 <= 00000001

@000041c8: $ 2 <= 00000001

@000041d4: $16 <= 00007f00

@000041d8: $ 8 <= 00000008

@000041dc: $ 8 <= 00000009

@000041e8: $ 2 <= 00000009

@00004180: $26 <= 00000400

@00004184: $27 <= 00003030

@00004188: $26 <= 00000100

@0000418c: $26 <= 00000000

@000041a4: $26 <= 00000400

@000041a8: $26 <= 00000001

@000041ac: $26 <= 00000001

@000041b0: $ 8 <= 00000001

@000041b4: $ 9 <= 00000002

@000041b8: $10 <= 00000004

@000041bc: $ 2 <= 00000001

@000041c8: $ 2 <= 00000001

@000041d4: $16 <= 00007f00

@000041d8: $ 8 <= 00000008

@000041dc: $ 8 <= 00000009

@000041e8: $ 2 <= 00000009

4、测试D

测试程序：

li $2, 0xffff0000

li $3, 0x0000ffff

mtc0 $3, $14

mfc0 $3, $14

sw $2, 0($0)

lw $3, 0($0)

mtc0 $3, $14

mfc0 $3, $14

addu $3, $3, $3

jal go

mtc0 $31, $14

go:

mfc0 $3, $14

nop

期望结果：

@00003000: $ 1 <= ffff0000

@00003004: $ 2 <= ffff0000

@00003008: $ 3 <= 0000ffff

@00003010: $ 3 <= 0000ffff

@00003014: \*00000000 <= ffff0000

@00003018: $ 3 <= ffff0000

@00003020: $ 3 <= ffff0000

@00003024: $ 3 <= fffe0000

@00003028: $31 <= 00003030

@00003030: $ 3 <= 00003030

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”？

硬件和软件的设计都采用分层的方法构建计算机系统，通过抽象，将下级的实现细节隐藏起来，但又保留有为上级所使用的接口。将硬件和底层软件抽象起来并组织为接口的，就是指令集体系结构，它使得软件设计师不必过多考虑硬件设计的具体细节，只需针对其指令集进行编写，即可完成对硬件层次的操作。

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。

在现代计算机中，DM应和IM一样都位于主板中的内存部分，CPU通过内存管理单元与内存相连接，以此进行访问。

同时，对于性能的考虑，CPU中也存在指令、数据的一、二级缓存，我认为，IM、DM正是它们的简化形式。

3、BE 部件对所有的外设都是必要的吗？

不是，并非所有的外设都要求支持按字节或半字进行访问和存储。在一些I/O交换数据量极大的设备中，只需按字或更大的单元进行访存。

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在计数为零后将中断信号置1，并将计数使能位复位为零，当CPU对其技术使能位写入“1”后，复位中断信号，继续执行相应工作。

模式1在计数为零后将中断信号置1，但不对技术使能位进行操作，在下一周期进入IDLE阶段后复位中断信号，继续执行相应工作。

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。）

.text 0x00003000

ori $t2, $0, 0xff01

mtc0, $t2, $12

ori $t0, $0, 9

ori $t1, $0, 1000

ori $s0, $0, 0x7f00

sw $t1, 4($s0)

sw $t0, 0($s0)

.ktext 0x00004180

ori $k0, $0, 0x7f00

lw $k1, 0($k0)

ori $k1, $k1, 0x0001

sw $k1, 0($k0)

eret

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？

键盘、鼠标是通过中断请求的方式进行IO操作的。当在键盘、鼠标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，把按下的按键编码从I/O数据缓冲区读到寄存器（或者鼠标的操作），最后放入内存中。