**实验六 CPU综合设计**

**一、实验目的**

1. 掌握复杂系统设计方法。
2. 深刻理解计算机系统硬件原理。

**二、实验内容**

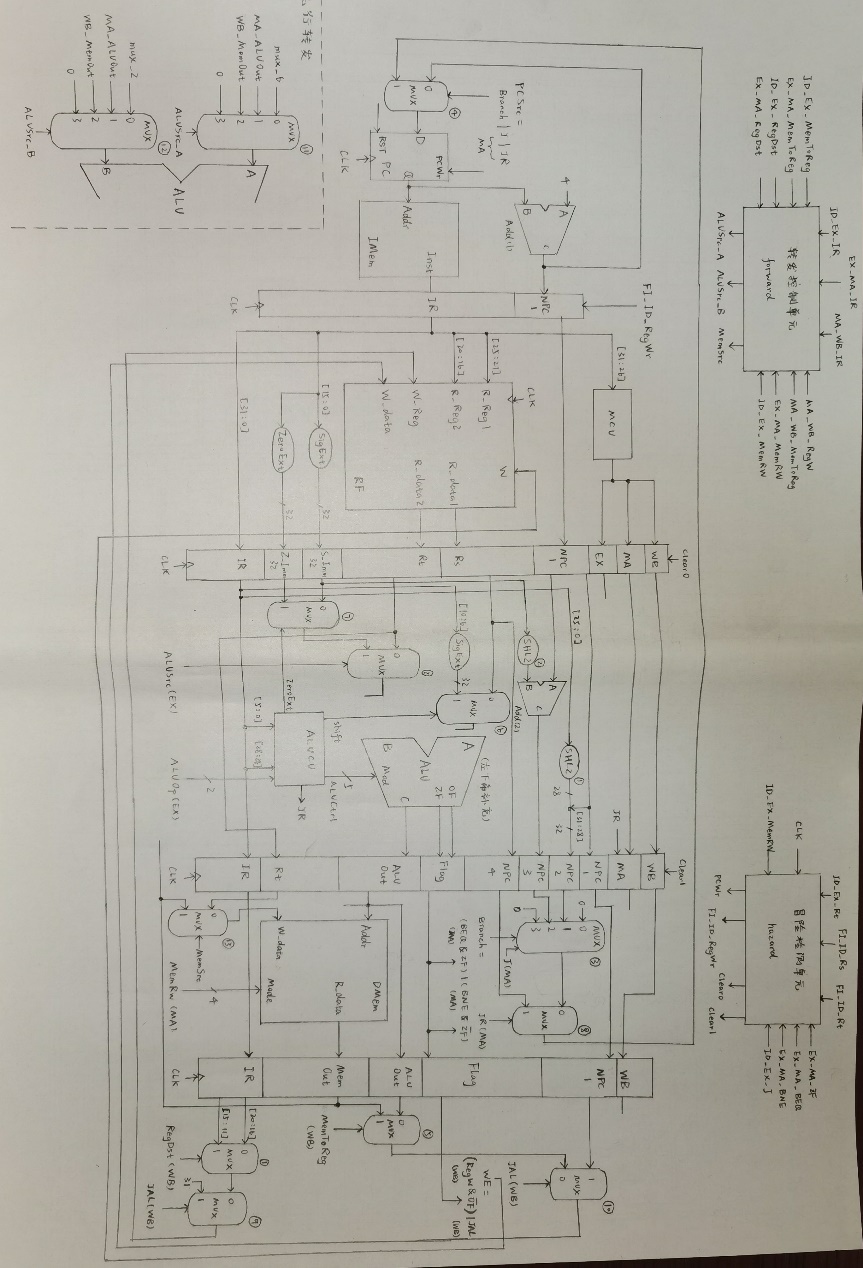
1. 设计一个基于MIPS指令集的CPU，支持以下指令：{addu, subu, ori, lw, sw, beq, lui, nop}（及格）
2. CPU需要包含寄存器组、RAM模块、ALU模块、指令译码模块。
3. 该CPU能运行基本的汇编指令（编写测试程序完成所有指令测试，要求与MARS模拟器运行结果一致）。
4. 在1基础上，扩展指令集，实现MIPS-Lite指令。（A-~A，编写测试程序完成所有指令测试，要求与MARS模拟器运行结果一致）
5. 在4基础上，实现5级流水线CPU。（A+，编写测试程序完成所有指令测试，要求与MARS模拟器运行结果一致）
6. 如发现代码为网上下载代码，成绩一律按不及格处理。

**三、实验要求**

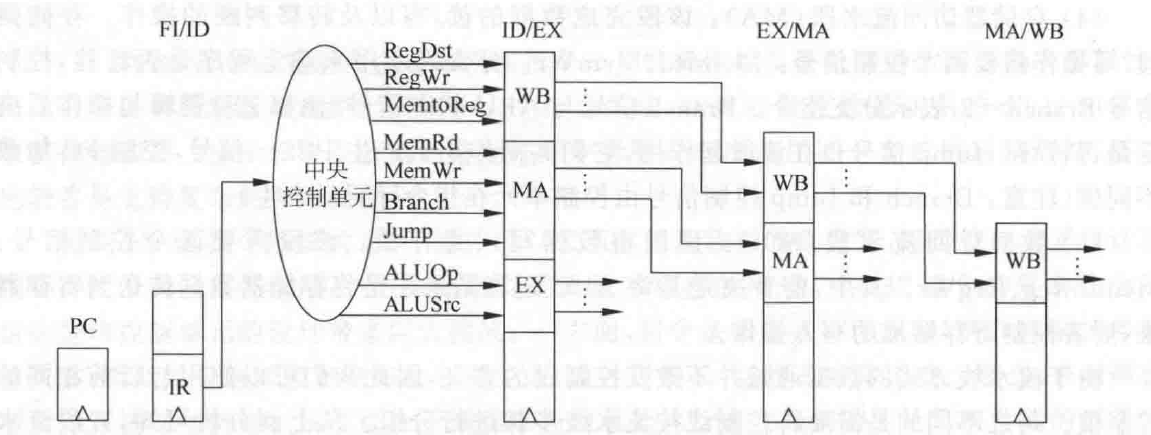
1. 编写相应测试程序，完成所有指令测试。

**四、实验代码及结果**

1、五级流水CPU数据通路



2、流水线中控制信号的存储与转发



3、流水线冒险

（1）结构冒险

通过**部件冗余**的方法解决，以避免硬件资源冲突。

如：分别设置指令存储器与数据存储器、为指令地址的计算单独设置加法器。

（2）数据冒险

通过**先行转发**的方法解决，使用内部数据缓存向后续指令直接提供所需要的数据，而不必等待该数据到达程序员可见的寄存器或者存储器才去使用它。

如：对于两条R型指令，若当前指令的目的寄存器是后续指令的源寄存器，则将ALU输出先行转发至ALU输入；对load指令与store指令间的数据冒险，可以将数据存储器输出先行转发至数据存储器输入；对于load指令与其它指令（非store指令）间的数据冒险，可以将数据存储器输出先行转发至ALU输入。需要注意的是，对于load-use的情况，还需要引入一次流水线阻塞才能消除数据冒险。

（3）控制冒险

对于分支指令，可以采用**静态预测**的方法，即预测分支转移条件不成立，流水线按指令顺序读取下一条指令，当预测失败时，清除分支指令后面的3条指令；

对于无条件转移指令，可以采用**流水线阻塞**的方法，使流水线停顿直至转移地址被计算出来。

4、Verilog代码

（1）多路选择器

|  |
| --- |
| **module** MUX21  **#(parameter** WIDTH **=** 32**)**  **(input** **wire** Sel**,**  **input** **wire** **[**WIDTH**-**1**:**0**]** in1**,** in2**,**  **output** **wire** **[**WIDTH**-**1**:**0**]** out**);**  **assign** out **=** **(**Sel **==** 0**)?** in1**:** in2**;**  **endmodule**  **module** MUX41  **#(parameter** WIDTH **=** 32**)**  **(input** **wire** **[**1**:**0**]** Sel**,**  **input** **wire** **[**WIDTH**-**1**:**0**]** in1**,** in2**,** in3**,** in4**,**  **output** **wire** **[**WIDTH**-**1**:**0**]** out**);**  **assign** out **=** **(**Sel **==** 2'b00**)?** in1**:**  **(**Sel **==** 2'b01**)?** in2**:**  **(**Sel **==** 2'b10**)?** in3**:** in4**;**  **endmodule** |

（2）符号扩展

|  |
| --- |
| **module** SigExt  **#(parameter** INPUT\_WIDTH **=** 16**,**  **parameter** OUTPUT\_WIDTH **=** 32**)**  **(input** **wire** **signed** **[**INPUT\_WIDTH**-**1**:**0**]** in**,**  **output** **wire** **signed** **[**OUTPUT\_WIDTH**-**1**:**0**]** out**);**  **assign** out **=** **{{(**OUTPUT\_WIDTH **-** INPUT\_WIDTH**){**in**[**INPUT\_WIDTH**-**1**]}},** in**};**  **endmodule** |

（3）无符号扩展

|  |
| --- |
| **module** ZeroExt  **#(parameter** INPUT\_WIDTH **=** 16**,**  **parameter** OUTPUT\_WIDTH **=** 32**)**  **(input** **wire** **signed** **[**INPUT\_WIDTH**-**1**:**0**]** in**,**  **output** **wire** **signed** **[**OUTPUT\_WIDTH**-**1**:**0**]** out**);**  **assign** out **=** **{{(**OUTPUT\_WIDTH **-** INPUT\_WIDTH**){**1'b0**}},** in**};**  **endmodule** |

（4）移位器

|  |
| --- |
| **module** SHL2\_1**(**  **input** **wire** **[**25**:**0**]** in**,**  **output** **wire** **[**27**:**0**]** out  **);**    **assign** out **=** **{**in**,** 2'b00**};**  **endmodule**  **module** SHL2\_2**(**  **input** **wire** **[**31**:**0**]** in**,**  **output** **wire** **[**31**:**0**]** out  **);**    **assign** out **=** **{**in**[**29**:**0**],** 2'b00**};**  **endmodule** |

（5）32位加法器

|  |
| --- |
| **module** halfadder**(**S**,** C**,** A**,** B**);**  **input** **wire** A**,** B**;**  **output** **wire** S**,** C**;**  **xor** **(**S**,** A**,** B**);**  **and** **(**C**,** A**,** B**);**  **endmodule**  **module** fulladder**(**S**,** Co**,** A**,** B**,** Ci**);**  **input** **wire** A**,** B**,** Ci**;**  **output** **wire** S**,** Co**;**  **wire** S1**,** D1**,** D2**;**  halfadder HA1**(**S1**,** D1**,** A**,** B**);**  halfadder HA2**(**S**,** D2**,** S1**,** Ci**);**  **or** **(**Co**,** D2**,** D1**);**  **endmodule**  **module** addr\_4bit**(**S**,** C3**,** A**,** B**,** Ci**);**  **input** **wire** **[**3**:**0**]** A**,** B**;**  **input** **wire** Ci**;**  **output** **wire** **[**3**:**0**]** S**;**  **output** **wire** C3**;**  **wire** C0**,** C1**,** C2**;**    fulladder FA0**(**S**[**0**],** C0**,** A**[**0**],** B**[**0**],** Ci**),**  FA1**(**S**[**1**],** C1**,** A**[**1**],** B**[**1**],** C0**),**  FA2**(**S**[**2**],** C2**,** A**[**2**],** B**[**2**],** C1**),**  FA3**(**S**[**3**],** C3**,** A**[**3**],** B**[**3**],** C2**);**  **endmodule**  **module** Add**(**S**,** C**,** A**,** B**);**  **input** **wire** **[**31**:**0**]** A**,** B**;**  **output** **wire** **[**31**:**0**]** S**;**  **output** **wire** C**;**  **wire** C3**,** C7**,** C11**,** C15**,** C19**,** C23**,** C27**;**    addr\_4bit FA4\_0**(**S**[**3**:**0**],** C3**,** A**[**3**:**0**],** B**[**3**:**0**],** 1'b0**),**  FA4\_1**(**S**[**7**:**4**],** C7**,** A**[**7**:**4**],** B**[**7**:**4**],** C3**),**  FA4\_2**(**S**[**11**:**8**],** C11**,** A**[**11**:**8**],** B**[**11**:**8**],** C7**),**  FA4\_3**(**S**[**15**:**12**],** C15**,** A**[**15**:**12**],** B**[**15**:**12**],** C11**),**  FA4\_4**(**S**[**19**:**16**],** C19**,** A**[**19**:**16**],** B**[**19**:**16**],** C15**),**  FA4\_5**(**S**[**23**:**20**],** C23**,** A**[**23**:**20**],** B**[**23**:**20**],** C19**),**  FA4\_6**(**S**[**27**:**24**],** C27**,** A**[**27**:**24**],** B**[**27**:**24**],** C23**),**  FA4\_7**(**S**[**31**:**28**],** C**,** A**[**31**:**28**],** B**[**31**:**28**],** C27**);**  **endmodule** |

（6）程序计数器

|  |
| --- |
| **module** PC**(**  **input** **wire** RST**,** CLK**,** PCWr**,**  **input** **wire** **[**31**:**0**]** D**,**  **output** **reg** **[**31**:**0**]** Q  **);**  **always** **@(posedge** CLK**)** **begin**  **if(**RST**)** Q **<=** 0**;**  **else** **if(**PCWr**)** Q **<=** D**;**  **end**  **endmodule** |

（7）指令存储器

|  |
| --- |
| **module** IMem**(**  **input** **wire** **[**31**:**0**]** Addr**,**  **output** **wire** **[**31**:**0**]** Inst  **);**  **parameter** IMEM\_SIZE **=** 64**;**  **reg** **[**31**:**0**]** RAM**[**IMEM\_SIZE**-**1**:**0**];**    **initial** $readmemh**(**"D:/Desktop/instruction.txt"**,** RAM**);**  **assign** Inst **=** RAM**[**Addr**/**4**];**  **endmodule** |

（8）寄存器组

|  |
| --- |
| **module** RF**(**  **input** **wire** CLK**,** W**,**  **input** **wire** **[**31**:**0**]** W\_data**,**  **input** **wire** **[**4**:**0**]** R\_Reg1**,** R\_Reg2**,** W\_Reg**,**  **output** **wire** **[**31**:**0**]** R\_data1**,** R\_data2  **);**    **reg** **[**31**:**0**]** rf**[**31**:**0**];**  **integer** i**;**  **initial** **for(**i **=** 0**;** i **<** 32**;** i **=** i **+** 1**)** rf**[**i**]** **=** 0**;**    **always** **@(posedge** CLK**)** **begin**  **if(**W **&&** **(**W\_Reg **!=** 0**))** rf**[**W\_Reg**]** **<=** W\_data**;**  **end**    **assign** R\_data1 **=** **(**R\_Reg1 **==** 0**)?** 0**:**  **(**R\_Reg1 **===** W\_Reg**)?** W\_data**:** rf**[**R\_Reg1**];**  **assign** R\_data2 **=** **(**R\_Reg2 **==** 0**)?** 0**:**  **(**R\_Reg2 **===** W\_Reg**)?** W\_data**:** rf**[**R\_Reg2**];**  **endmodule** |

（9）指令译码器

|  |
| --- |
| **module** MCU**(**  **input** **wire** **[**5**:**0**]** Op**,**  **output** **wire** RegDst**,** RegW**,** ALUSrc**,** MemToReg**,** BEQ**,** BNE**,** J**,** JAL**,**  **output** **wire** **[**3**:**0**]** MemRW**,**  **output** **wire** **[**1**:**0**]** ALUOp  **);**  **reg** **[**13**:**0**]** Controls**;**  **assign** **{**RegDst**,** RegW**,** ALUSrc**,** MemToReg**,** MemRW**,** BEQ**,** BNE**,** J**,** JAL**,** ALUOp**}** **=** Controls**;**  **always** **@(\*)** **begin**  **casex(**Op**)**  6'b000000**:** Controls **<=** 14'b1100\_0xxx\_0000\_10**;** //R-R  6'b001xxx**:** Controls **<=** 14'b0110\_0xxx\_0000\_11**;** //R-I    6'b100000**:** Controls **<=** 14'b0111\_1011\_0000\_00**;** //LB  6'b100001**:** Controls **<=** 14'b0111\_1101\_0000\_00**;** //LH  6'b100011**:** Controls **<=** 14'b0111\_1111\_0000\_00**;** //LW  6'b100100**:** Controls **<=** 14'b0111\_1100\_0000\_00**;** //LBU  6'b100101**:** Controls **<=** 14'b0111\_1110\_0000\_00**;** //LHU    6'b101000**:** Controls **<=** 14'bx01x\_1000\_0000\_00**;** //SB  6'b101001**:** Controls **<=** 14'bx01x\_1001\_0000\_00**;** //SH  6'b101011**:** Controls **<=** 14'bx01x\_1010\_0000\_00**;** //SW    6'b000100**:** Controls **<=** 14'bx00x\_0xxx\_1000\_01**;** //BEQ  6'b000101**:** Controls **<=** 14'bx00x\_0xxx\_0100\_01**;** //BNE    6'b000010**:** Controls **<=** 14'bx0xx\_0xxx\_0010\_xx**;** //J  6'b000011**:** Controls **<=** 14'bx0xx\_0xxx\_0011\_xx**;** //JAL    **default:** Controls **<=** 14'bxxxx\_xxxx\_xxxx\_xx**;**  **endcase**  **end**  **endmodule** |

（10）ALU控制器

|  |
| --- |
| **module** ALU\_CU**(**  **input** **wire** **[**1**:**0**]** ALUOp**,**  **input** **wire** **[**5**:**0**]** Funct**,**  **input** **wire** **[**2**:**0**]** ImmOp**,**  **output** **reg** shift**,** ZeroExt**,** JR**,**  **output** **reg** **[**4**:**0**]** ALUControl  **);**  **always** **@(\*)** **begin**  shift **<=** **(**ALUOp **==** 2'b10 **&&** Funct**[**5**:**2**]** **==** 4'b0000**)?** 1**:** 0**;**  ZeroExt **<=** **(**ALUOp **==** 2'b11 **&&** **(**ImmOp **==** 3'b100 **||** ImmOp **==** 3'b101 **||** ImmOp **==** 3'b110**))?** 1**:** 0**;**  JR **<=** **(**ALUOp **==** 2'b10 **&&** Funct **==** 6'b001000**)?** 1**:** 0**;**  **case(**ALUOp**)**  2'b00**:** ALUControl **<=** 5'b00001**;** //ADDU  2'b01**:** ALUControl **<=** 5'b00011**;** //SUBU  2'b10**:** **case(**Funct**)**  6'b100000**:** ALUControl **<=** 5'b00000**;** //ADD  6'b100001**:** ALUControl **<=** 5'b00001**;** //ADDU  6'b100010**:** ALUControl **<=** 5'b00010**;** //SUB  6'b100011**:** ALUControl **<=** 5'b00011**;** //SUBU  6'b011000**:** ALUControl **<=** 5'b00100**;** //MULT  6'b011001**:** ALUControl **<=** 5'b00101**;** //MULTU  6'b011010**:** ALUControl **<=** 5'b00110**;** //DIV  6'b011011**:** ALUControl **<=** 5'b00111**;** //DIVU  6'b101010**:** ALUControl **<=** 5'b01000**;** //SLT  6'b101011**:** ALUControl **<=** 5'b01001**;** //SLTU  6'b000000**:** ALUControl **<=** 5'b01010**;** //SLL  6'b000010**:** ALUControl **<=** 5'b01011**;** //SRL  6'b000011**:** ALUControl **<=** 5'b01100**;** //SRA  6'b000100**:** ALUControl **<=** 5'b01010**;** //SLLV  6'b000110**:** ALUControl **<=** 5'b01011**;** //SRLV  6'b000111**:** ALUControl **<=** 5'b01100**;** //SRAV  6'b100100**:** ALUControl **<=** 5'b01101**;** //AND  6'b100101**:** ALUControl **<=** 5'b01110**;** //OR  6'b100110**:** ALUControl **<=** 5'b01111**;** //XOR  6'b100111**:** ALUControl **<=** 5'b10000**;** //NOR  6'b010000**:** ALUControl **<=** 5'b10010**;** //MFHI  6'b010010**:** ALUControl **<=** 5'b10011**;** //MFLO  **default:** ALUControl **<=** 5'bxxxxx**;**  **endcase**  2'b11**:** **case(**ImmOp**)**  3'b000**:** ALUControl **<=** 5'b00000**;** //ADDI  3'b001**:** ALUControl **<=** 5'b00001**;** //ADDIU  3'b100**:** ALUControl **<=** 5'b01101**;** //ANDI  3'b101**:** ALUControl **<=** 5'b01110**;** //ORI  3'b110**:** ALUControl **<=** 5'b01111**;** //XORI  3'b111**:** ALUControl **<=** 5'b10001**;** //LUI  3'b010**:** ALUControl **<=** 5'b01000**;** //SLTI  3'b011**:** ALUControl **<=** 5'b01001**;** //SLTIU  **default:** ALUControl **<=** 5'bxxxxx**;**  **endcase**  **endcase**  **end**  **endmodule** |

（11）ALU运算器

|  |
| --- |
| **module** ALU**(**  **input** **wire** **[**4**:**0**]** OP**,**  **input** **wire** **[**31**:**0**]** A**,** B**,**  **output** **reg** **[**31**:**0**]** F**,**  **output** **reg** ZF**,** OF**,** Error\_DIV0  **);**  **reg** **[**31**:**0**]** HI**,** LO**;**  **reg** **[**32**:**0**]** temp**;**    **wire** **[**31**:**0**]** Sum**;**  **wire** CF**;**  Add adder**(**Sum**,** CF**,** A**,** B**);**    **initial** **{**HI**,** LO**}** **=** 64'h0000\_0000\_0000\_0000**;**  **always** **@(\*)** **begin**  OF **=** 0**;**  Error\_DIV0 **=** 0**;**  **#**1  **case(**OP**)**  5'b00000**:** **{**F**,** OF**}** **=** **(**A**[**31**]** **^** B**[**31**]** **^** Sum**[**31**]** **^** CF**)?** **{**32'hzzzz\_zzzz**,** 1'b1**}:** **{**Sum**,** 1'b0**};**  5'b00001**:** F **=** Sum**;**  5'b00010**:** **begin**  temp **=** **{**A**[**31**],** A**}** **-** **{**B**[**31**],** B**};**  **{**F**,** OF**}** **=** **(**temp**[**32**]** **^** temp**[**31**])?** **{**32'hzzzz\_zzzz**,** 1'b1**}:** **{**temp**[**31**:**0**],** 1'b0**};**  **end**  5'b00011**:** F **=** A **-** B**;**  5'b00100**:** **{**HI**,** LO**}** **=** **(**$signed**(**A**))** **\*** **(**$signed**(**B**));**  5'b00101**:** **{**HI**,** LO**}** **=** A **\*** B**;**  5'b00110**:** **begin**  **if(**B **==** 0**)** Error\_DIV0 **=** 1**;**  **else** **begin**  HI **=** **(**$signed**(**A**))** **%** **(**$signed**(**B**));**  LO **=** **(**$signed**(**A**))** **/** **(**$signed**(**B**));**  **end**  **end**  5'b00111**:** **begin**  **if(**B **==** 0**)** Error\_DIV0 **=** 1**;**  **else** **begin**  HI **=** A **%** B**;**  LO **=** A **/** B**;**  **end**  **end**  5'b01000**:** F **=** **((**$signed**(**A**))** **<** **(**$signed**(**B**)))?** 1**:** 0**;**  5'b01001**:** F **=** **(**A **<** B**)?** 1**:** 0**;**  5'b01010**:** F **=** B **<<** A**[**4**:**0**];**  5'b01011**:** F **=** B **>>** A**[**4**:**0**];**  5'b01100**:** F **=** **(**$signed**(**B**))** **>>>** A**[**4**:**0**];**  5'b01101**:** F **=** A **&** B**;**  5'b01110**:** F **=** A **|** B**;**  5'b01111**:** F **=** A **^** B**;**  5'b10000**:** F **=** **~(**A **|** B**);**  5'b10001**:** F **=** **{**B**[**15**:**0**],** 16'h0000**};**  5'b10010**:** F **=** HI**;**  5'b10011**:** F **=** LO**;**  **default:** F **=** 32'hzzzz\_zzzz**;**  **endcase**  ZF **=** F **==** 0**;**  **end**  **endmodule** |

（12）数据存储器

|  |
| --- |
| **module** DMem**(**  **input** **wire** **[**3**:**0**]**Mode**,**  **input** **wire** **[**31**:**0**]** Addr**,** W\_data**,**  **output** **reg** **[**31**:**0**]** R\_data**,**  **output** **reg** Error\_Addr  **);**  **parameter** DMEM\_SIZE **=** 256**;**  **reg** **[**7**:**0**]** RAM**[**DMEM\_SIZE**-**1**:**0**];**    **initial** $readmemb**(**"D:/Desktop/data.txt"**,** RAM**);**  **always** **@(\*)** **begin**  Error\_Addr **=** 0**;**  **case(**Mode**)**  4'b1000**:** RAM**[**Addr**+**0**]** **<=** W\_data**[**7**:**0**];** //SB  4'b1001**:** **if(**Addr**[**0**]** **==** 1'b0**)** **begin** //SH  RAM**[**Addr**+**0**]** **<=** W\_data**[**7**:**0**];**  RAM**[**Addr**+**1**]** **<=** W\_data**[**15**:**8**];**  **end**  **else** Error\_Addr **=** 1**;**  4'b1010**:** **if(**Addr**[**1**:**0**]** **==** 2'b00**)** **begin** //SW  RAM**[**Addr**+**0**]** **<=** W\_data**[**7**:**0**];**  RAM**[**Addr**+**1**]** **<=** W\_data**[**15**:**8**];**  RAM**[**Addr**+**2**]** **<=** W\_data**[**23**:**16**];**  RAM**[**Addr**+**3**]** **<=** W\_data**[**31**:**24**];**  **end**  **else** Error\_Addr **=** 1**;**  4'b1011**:** R\_data **=** **{{(**24**){**RAM**[**Addr**+**0**][**7**]}},** RAM**[**Addr**+**0**]};** //LB  4'b1100**:** R\_data **=** **{**24'h000000**,** RAM**[**Addr**+**0**]};** //LBU  4'b1101**:** **if(**Addr**[**0**]** **==** 1'b0**)** //LH  R\_data **=** **{{(**16**){**RAM**[**Addr**+**1**][**7**]}},** RAM**[**Addr**+**1**],** RAM**[**Addr**+**0**]};**  **else** Error\_Addr **=** 1**;**  4'b1110**:** **if(**Addr**[**0**]** **==** 1'b0**)** //LHU  R\_data **=** **{**16'h0000**,** RAM**[**Addr**+**1**],** RAM**[**Addr**+**0**]};**  **else** Error\_Addr **=** 1**;**  4'b1111**:** **if(**Addr**[**1**:**0**]** **==** 2'b00**)** //LW  R\_data **=** **{**RAM**[**Addr**+**3**],** RAM**[**Addr**+**2**],** RAM**[**Addr**+**1**],** RAM**[**Addr**+**0**]};**  **else** Error\_Addr **=** 1**;**  **default:** R\_data **=** 32'hzzzz\_zzzz**;**  **endcase**  **end**  **endmodule** |

（13）FI/ID级间寄存器

|  |
| --- |
| **module** FI\_ID**(**  **input** **wire** CLK**,** FI\_ID\_RegWr**,**  **input** **wire** **[**31**:**0**]** FI\_NPC1**,** FI\_IR**,**  **output** **reg** **[**31**:**0**]** NPC1**,** IR  **);**  **always** **@(posedge** CLK**)** **begin**  **if(**FI\_ID\_RegWr**)** **begin**  NPC1 **<=** FI\_NPC1**;**  IR **<=** FI\_IR**;**  **end**  **end**  **endmodule** |

（14）ID/EX级间寄存器

|  |
| --- |
| **module** ID\_EX**(**  **input** **wire** CLK**,** Clear**,**  **input** **wire** ID\_ALUSrc**,** ID\_BEQ**,** ID\_BNE**,** ID\_J**,** ID\_MemToReg**,** ID\_JAL**,** ID\_RegW**,** ID\_RegDst**,**  **input** **wire** **[**1**:**0**]** ID\_ALUOp**,**  **input** **wire** **[**3**:**0**]** ID\_MemRW**,**  **input** **wire** **[**31**:**0**]** ID\_NPC1**,** ID\_Rs**,** ID\_Rt**,** ID\_S\_Imm32**,** ID\_Z\_Imm32**,** ID\_IR**,**  **output** **reg** ALUSrc**,** BEQ**,** BNE**,** J**,** MemToReg**,** JAL**,** RegW**,** RegDst**,**  **output** **reg** **[**1**:**0**]** ALUOp**,**  **output** **reg** **[**3**:**0**]** MemRW**,**  **output** **reg** **[**31**:**0**]** NPC1**,** Rs**,** Rt**,** S\_Imm32**,** Z\_Imm32**,** IR  **);**  **always** **@(posedge** CLK**)** **begin**  ALUSrc **<=** Clear**?** 1'bx**:** ID\_ALUSrc**;**  BEQ **<=** Clear**?** 1'b0**:** ID\_BEQ**;**  BNE **<=** Clear**?** 1'b0**:** ID\_BNE**;**  J **<=** Clear**?** 1'b0**:** ID\_J**;**  MemToReg **<=** Clear**?** 1'bx**:** ID\_MemToReg**;**  JAL **<=** Clear**?** 1'b0**:** ID\_JAL**;**  RegW **<=** Clear**?** 1'b0**:** ID\_RegW**;**  RegDst **<=** Clear**?** 1'bx**:** ID\_RegDst**;**  ALUOp **<=** Clear**?** 2'bxx**:** ID\_ALUOp**;**  MemRW **<=** Clear**?** 4'b0xxx**:** ID\_MemRW**;**    NPC1 **<=** ID\_NPC1**;**  Rs **<=** ID\_Rs**;**  Rt **<=** ID\_Rt**;**  S\_Imm32 **<=** ID\_S\_Imm32**;**  Z\_Imm32 **<=** ID\_Z\_Imm32**;**  IR **<=** ID\_IR**;**  **end**  **endmodule** |

（15）EX/MA级间寄存器

|  |
| --- |
| **module** EX\_MA**(**  **input** **wire** CLK**,** Clear**,**  **input** **wire** EX\_BEQ**,** EX\_BNE**,** EX\_J**,** EX\_MemToReg**,** EX\_JAL**,** EX\_RegW**,** EX\_RegDst**,** EX\_JR**,** EX\_ZF**,** EX\_OF**,**  **input** **wire** **[**3**:**0**]** EX\_MemRW**,**  **input** **wire** **[**31**:**0**]** EX\_NPC1**,** EX\_NPC2**,** EX\_NPC3**,** EX\_NPC4**,** EX\_ALUOut**,** EX\_Rt**,** EX\_IR**,**  **output** **reg** BEQ**,** BNE**,** J**,** MemToReg**,** JAL**,** RegW**,** RegDst**,** JR**,** ZF**,** OF**,**  **output** **reg** **[**3**:**0**]** MemRW**,**  **output** **reg** **[**31**:**0**]** NPC1**,** NPC2**,** NPC3**,** NPC4**,** ALUOut**,** Rt**,** IR  **);**  **always** **@(posedge** CLK**)** **begin**  BEQ **<=** Clear**?** 1'b0**:** EX\_BEQ**;**  BNE **<=** Clear**?** 1'b0**:** EX\_BNE**;**  J **<=** Clear**?** 1'b0**:** EX\_J**;**  MemToReg **<=** Clear**?** 1'bx**:** EX\_MemToReg**;**  JAL **<=** Clear**?** 1'b0**:** EX\_JAL**;**  RegW **<=** Clear**?** 1'b0**:** EX\_RegW**;**  RegDst **<=** Clear**?** 1'bx**:** EX\_RegDst**;**  JR **<=** Clear**?** 1'b0**:** EX\_JR**;**  MemRW **<=** Clear**?** 4'b0xxx**:** EX\_MemRW**;**    ZF **<=** EX\_ZF**;**  OF **<=** EX\_OF**;**  NPC1 **<=** EX\_NPC1**;**  NPC2 **<=** EX\_NPC2**;**  NPC3 **<=** EX\_NPC3**;**  NPC4 **<=** EX\_NPC4**;**  ALUOut **<=** EX\_ALUOut**;**  Rt **<=** EX\_Rt**;**  IR **<=** EX\_IR**;**  **end**  **endmodule** |

（16）MA/WB级间寄存器

|  |
| --- |
| **module** MA\_WB**(**  **input** **wire** CLK**,**  **input** **wire** MA\_MemToReg**,** MA\_JAL**,** MA\_RegW**,** MA\_RegDst**,** MA\_OF**,**  **input** **wire** **[**31**:**0**]** MA\_NPC1**,** MA\_ALUOut**,** MA\_MemOut**,** MA\_IR**,**  **output** **reg** MemToReg**,** JAL**,** RegW**,** RegDst**,** OF**,**  **output** **reg** **[**31**:**0**]** NPC1**,** ALUOut**,** MemOut**,** IR  **);**  **always** **@(posedge** CLK**)** **begin**  MemToReg **<=** MA\_MemToReg**;**  JAL **<=** MA\_JAL**;**  RegW **<=** MA\_RegW**;**  RegDst **<=** MA\_RegDst**;**  OF **<=** MA\_OF**;**  NPC1 **<=** MA\_NPC1**;**  ALUOut **<=** MA\_ALUOut**;**  MemOut **<=** MA\_MemOut**;**  IR **<=** MA\_IR**;**  **end**  **endmodule** |

（17）数据冒险转发控制单元

|  |
| --- |
| **module** forward**(**  **input** **wire** **[**31**:**0**]** ID\_EX\_IR**,** EX\_MA\_IR**,** MA\_WB\_IR**,**  **input** **wire** ID\_EX\_MemToReg**,** EX\_MA\_MemToReg**,** ID\_EX\_RegDst**,** EX\_MA\_RegDst**,** MA\_WB\_RegW**,** MA\_WB\_MemToReg**,**  **input** **wire** **[**3**:**0**]** EX\_MA\_MemRW**,** ID\_EX\_MemRW**,**  **output** **reg** **[**1**:**0**]** ALUSrc\_A**,** ALUSrc\_B**,**  **output** **reg** MemSrc  **);**  **initial** **begin**  ALUSrc\_A **<=** 2'b00**;**  ALUSrc\_B **<=** 2'b00**;**  MemSrc **<=** 1'b0**;**  **end**  **wire** EX\_MA\_MemWr**;**  **assign** EX\_MA\_MemWr **=** **(**EX\_MA\_MemRW **==** 4'b1000**)** **|** **(**EX\_MA\_MemRW **==** 4'b1001**)** **|** **(**EX\_MA\_MemRW **==** 4'b1010**);**    **wire** ID\_EX\_MemWr**;**  **assign** ID\_EX\_MemWr **=** **(**ID\_EX\_MemRW **==** 4'b1000**)** **|** **(**ID\_EX\_MemRW **==** 4'b1001**)** **|** **(**ID\_EX\_MemRW **==** 4'b1010**);**  **always** **@(\*)** **begin**  //转发ALU输出至ALU输入  **if((**ID\_EX\_RegDst **&** EX\_MA\_RegDst **&** **(**ID\_EX\_IR**[**25**:**21**]** **==** EX\_MA\_IR**[**15**:**11**])** **&** **(**ID\_EX\_IR**[**25**:**21**]** **!=** 5'b00000**))** **==** 1'b1**)**  ALUSrc\_A**[**0**]** **=** 1'b1**;**  **else** **if((**ID\_EX\_RegDst **&** **(~**EX\_MA\_RegDst**)** **&** **(~**EX\_MA\_MemToReg**)** **&** **(**ID\_EX\_IR**[**25**:**21**]** **==** EX\_MA\_IR**[**15**:**11**])** **&** **(**ID\_EX\_IR**[**25**:**21**]** **!=** 5'b00000**))** **==** 1'b1**)**  ALUSrc\_A**[**0**]** **=** 1'b1**;**  **else** **if(((~**ID\_EX\_RegDst**)** **&** **(~**ID\_EX\_MemToReg**)** **&** EX\_MA\_RegDst **&** **(**ID\_EX\_IR**[**25**:**21**]** **==** EX\_MA\_IR**[**20**:**16**])** **&** **(**ID\_EX\_IR**[**25**:**21**]** **!=** 5'b00000**))** **==** 1'b1**)**  ALUSrc\_A**[**0**]** **=** 1'b1**;**  **else** **if(((~**ID\_EX\_RegDst**)** **&** **(~**ID\_EX\_MemToReg**)** **&** **(~**EX\_MA\_RegDst**)** **&** **(~**EX\_MA\_MemToReg**)** **&** **(**ID\_EX\_IR**[**25**:**21**]** **==** EX\_MA\_IR**[**20**:**16**])** **&** **(**ID\_EX\_IR**[**25**:**21**]** **!=** 5'b00000**))** **==** 1'b1**)**  ALUSrc\_A**[**0**]** **=** 1'b1**;**  **else** ALUSrc\_A**[**0**]** **=** 1'b0**;**  **if((**ID\_EX\_RegDst **&** EX\_MA\_RegDst **&** **(**ID\_EX\_IR**[**20**:**16**]** **==** EX\_MA\_IR**[**15**:**11**])** **&** **(**ID\_EX\_IR**[**20**:**16**]** **!=** 5'b00000**))** **==** 1'b1**)**  ALUSrc\_B**[**0**]** **=** 1'b1**;**  **else** **if(((~**ID\_EX\_RegDst**)** **&** **(~**ID\_EX\_MemToReg**)** **&** EX\_MA\_RegDst **&** **(**ID\_EX\_IR**[**20**:**16**]** **==** EX\_MA\_IR**[**20**:**16**])** **&** **(**ID\_EX\_IR**[**20**:**16**]** **!=** 5'b00000**))** **==** 1'b1**)**  ALUSrc\_B**[**0**]** **=** 1'b1**;**  **else** ALUSrc\_B**[**0**]** **=** 1'b0**;**  //转发数据存储器输出至数据存储器输入  **if((**MA\_WB\_RegW **&** EX\_MA\_MemWr **&** **(**MA\_WB\_IR**[**20**:**16**]** **==** EX\_MA\_IR**[**20**:**16**]))** **==** 1'b1**)**  MemSrc **=** 1'b1**;**  **else** MemSrc **=** 1'b0**;**    //转发数据存储器输出至ALU输入  **if((**MA\_WB\_MemToReg **&** **(~**ID\_EX\_MemWr**)** **&** **(**MA\_WB\_IR**[**20**:**16**]** **==** ID\_EX\_IR**[**25**:**21**]))** **==** 1'b1**)**  ALUSrc\_A**[**1**]** **=** 1'b1**;**  **else** ALUSrc\_A**[**1**]** **=** 1'b0**;**  **if((**MA\_WB\_MemToReg **&** **(~**ID\_EX\_MemWr**)** **&** **(**MA\_WB\_IR**[**20**:**16**]** **==** ID\_EX\_IR**[**20**:**16**]))** **==** 1'b1**)**  ALUSrc\_B**[**1**]** **=** 1'b1**;**  **else** ALUSrc\_B**[**1**]** **=** 1'b0**;**  **end**  **endmodule** |

（18）控制冒险检测处理单元

|  |
| --- |
| **module** hazard**(**  **input** **wire** CLK**,**  **input** **wire** **[**4**:**0**]** ID\_EX\_Rt**,** FI\_ID\_Rs**,** FI\_ID\_Rt**,**  **input** **wire** **[**3**:**0**]** ID\_EX\_MemRW**,**  **input** **wire** EX\_MA\_ZF**,** EX\_MA\_BEQ**,** EX\_MA\_BNE**,** ID\_EX\_J**,**  **output** **reg** PCWr**,** FI\_ID\_RegWr**,** Clear0**,** Clear1  **);**  **initial** **begin**  PCWr **<=** 1**;**  FI\_ID\_RegWr **<=** 1**;**  Clear0 **<=** 0**;**  Clear1 **<=** 0**;**  **end**  **wire** ID\_EX\_MemRd**;**  **assign** ID\_EX\_MemRd **=** **(**ID\_EX\_MemRW **==** 4'b1011**)** **|** **(**ID\_EX\_MemRW **==** 4'b1100**)** **|** **(**ID\_EX\_MemRW **==** 4'b1101**)** **|** **(**ID\_EX\_MemRW **==** 4'b1110**)** **|** **(**ID\_EX\_MemRW **==** 4'b1111**);**  **integer** i**,** k**;**  **always** **@(\*)** **begin**  PCWr **=** 1**;**  FI\_ID\_RegWr **=** 1**;**  Clear0 **=** 0**;**  Clear1 **=** 0**;**    //EX阶段检测load-use,流水线停顿1个时钟周期  **if(**ID\_EX\_MemRd **&** **((**ID\_EX\_Rt **==** FI\_ID\_Rs**)** **|** **(**ID\_EX\_Rt **==** FI\_ID\_Rt**)))** **begin**  Clear0 **<=** 1**;**  PCWr **<=** 0**;**  FI\_ID\_RegWr **<=** 0**;**  **end**  //EX阶段检测无条件转移指令,清除后面的无效指令  **if(**ID\_EX\_J**)** **begin**  k **=** 1**;**  i **=** 5**;**  **end**  **if(**k **==** 1**)** **begin**  **if(**i **==** 0**)** k **=** 0**;**  **else** **begin**  //置Clear0为“1”连续3个时钟周期  Clear0 **=** 1**;**  **if(**CLK**)** i **=** i **-** 1**;**  **end**  **end**  //MA阶段检测分支指令,采用静态预测方式,预测失败则清除后面3条指令  **if((**EX\_MA\_BEQ **&** EX\_MA\_ZF**)** **|** **(**EX\_MA\_BNE **&** **(~**EX\_MA\_ZF**)))** **begin**  Clear0 **<=** 1**;**  Clear1 **<=** 1**;**  FI\_ID\_RegWr **<=** 0**;**  **end**  **end**  **endmodule** |

（19）数据通路

|  |
| --- |
| **module** DataPath**(**  **input** **wire** CLK**,** RST  **);**  **wire** **[**31**:**0**]** InstAddr**,** Inst**;**  **wire** **[**31**:**0**]** add\_sum1**,** add\_sum2**;**  **wire** **[**31**:**0**]** mux\_2**,** mux\_3**,** mux\_4**,** mux\_5**,** mux\_6**,** mux\_7**,** mux\_8**,** mux\_10**,** mux\_11**,** mux\_12**,** mux\_13**;**  **wire** **[**4**:**0**]** mux\_1**,** mux\_9**;**  **wire** **[**31**:**0**]** ID\_NPC1**,** ID\_IR**;**  **wire** **[**31**:**0**]** SigExt1**,** SigExt2**,** ZeroExt1**;**  **wire** **[**31**:**0**]** R\_data1**,** R\_data2**;**  **wire** RegDst**,** J**,** JAL**,** RegW**,** BEQ**,** BNE**,** MemToReg**,** ALUSrc**;**  **wire** **[**3**:**0**]** MemRW**;**  **wire** **[**1**:**0**]** ALUOp**;**  **wire** EX\_RegDst**,** EX\_J**,** EX\_JAL**,** EX\_RegW**,** EX\_BEQ**,** EX\_BNE**,** EX\_MemToReg**,** EX\_ALUSrc**;**  **wire** **[**3**:**0**]** EX\_MemRW**;**  **wire** **[**1**:**0**]** EX\_ALUOp**;**  **wire** **[**31**:**0**]** EX\_NPC1**,** EX\_Rs**,** EX\_Rt**,** EX\_S\_Imm32**,** EX\_Z\_Imm32**,** EX\_IR**;**  **wire** shift**,** ZSExt**,** JR**,** ZF**,** OF**,** Error\_DIV0**;**  **wire** **[**4**:**0**]** ALUCtrl**;**  **wire** **[**31**:**0**]** ALUResult**;**  **wire** **[**27**:**0**]** shl1**;**  **wire** **[**31**:**0**]** shl2**;**  **wire** **[**31**:**0**]** JumpAddr**;**  **wire** MA\_BEQ**,** MA\_BNE**,** MA\_J**,** MA\_MemToReg**,** MA\_JAL**,** MA\_RegW**,** MA\_RegDst**,** MA\_JR**,** MA\_ZF**,** MA\_OF**;**  **wire** **[**3**:**0**]** MA\_MemRW**;**  **wire** **[**31**:**0**]** MA\_NPC1**,** MA\_NPC2**,** MA\_NPC3**,** MA\_NPC4**,** MA\_ALUOut**,** MA\_Rt**,** MA\_IR**;**  **wire** **[**31**:**0**]** R\_data**;**  **wire** Error\_Addr**;**  **wire** WB\_MemToReg**,** WB\_JAL**,** WB\_RegW**,** WB\_RegDst**,** WB\_OF**;**  **wire** **[**31**:**0**]** WB\_NPC1**,** WB\_ALUOut**,** WB\_MemOut**,** WB\_IR**;**  **wire** dummy1**,** dummy2**;**  **wire** PCSrc**,** Branch**,** WE**;**  **wire** **[**1**:**0**]** ALUSrc\_A**,** ALUSrc\_B**;**  **wire** MemSrc**,** Clear0**,** Clear1**,** PCWr**,** FI\_ID\_RegWr**;**  **reg** PCSrc1**;**    **initial** PCSrc1 **=** 1'b0**;**    **assign** JumpAddr **=** **{**EX\_NPC1**[**31**:**28**],** shl1**};**  **assign** Branch **=** **(**MA\_BEQ **&** MA\_ZF**)** **|** **(**MA\_BNE **&** **(~**MA\_ZF**));**  **assign** WE **=** **(**WB\_RegW **&** **(~**WB\_OF**))** **|** WB\_JAL**;**  **assign** PCSrc **=** PCSrc1**;**    **integer** i**;**  **always** **@(\*)** **begin**  **if(~**RST**)** **begin**  **if(**i **==** 0**)** **begin**  //检测分支与转移指令  **if(**Branch **==** 1'b1**)** PCSrc1 **=** 1'b1**;**  **else** **if(**MA\_J **==** 1'b1**)** PCSrc1 **=** 1'b1**;**  **else** **if(**MA\_JR **==** 1'b1**)** PCSrc1 **=** 1'b1**;**  **else** PCSrc1 **=** 1'b0**;**  **end**  **else** **begin**  //在复位后MA阶段暂无指令时负责从指令存储器中顺序取指  PCSrc1 **=** 1'b0**;**  **if(**CLK**)** i **=** i **-** 1**;**  **end**  **end**  **else** **begin**  //复位  PCSrc1 **=** 0**;**  i **=** 3**;**  **end**  **end**    //取指  PC pc**(.**PCWr**(**PCWr**),** **.**D**(**mux\_4**),** **.**CLK**(**CLK**),** **.**RST**(**RST**),** **.**Q**(**InstAddr**));**  IMem imem**(.**Addr**(**InstAddr**),** **.**Inst**(**Inst**));**  Add add1**(.**A**(**4**),** **.**B**(**InstAddr**),** **.**S**(**add\_sum1**),** **.**C**(**dummy1**));**  FI\_ID fi\_id**(.**FI\_ID\_RegWr**(**FI\_ID\_RegWr**),** **.**CLK**(**CLK**),** **.**FI\_NPC1**(**add\_sum1**),** **.**FI\_IR**(**Inst**),** **.**NPC1**(**ID\_NPC1**),** **.**IR**(**ID\_IR**));**    //译码  SigExt **#(**16**,**32**)** SigExt16\_32**(.**in**(**ID\_IR**[**15**:**0**]),** **.**out**(**SigExt1**));**  ZeroExt **#(**16**,**32**)** ZeroExt16\_32**(.**in**(**ID\_IR**[**15**:**0**]),** **.**out**(**ZeroExt1**));**  RF rf**(.**CLK**(**CLK**),** **.**W**(**WE**),** **.**W\_data**(**mux\_10**),** **.**R\_Reg1**(**ID\_IR**[**25**:**21**]),** **.**R\_Reg2**(**ID\_IR**[**20**:**16**]),** **.**W\_Reg**(**mux\_9**),** **.**R\_data1**(**R\_data1**),** **.**R\_data2**(**R\_data2**));**  MCU mcu**(.**Op**(**ID\_IR**[**31**:**26**]),** **.**RegDst**(**RegDst**),** **.**J**(**J**),** **.**JAL**(**JAL**),** **.**RegW**(**RegW**),** **.**BEQ**(**BEQ**),** **.**BNE**(**BNE**),** **.**MemToReg**(**MemToReg**),** **.**MemRW**(**MemRW**),** **.**ALUSrc**(**ALUSrc**),** **.**ALUOp**(**ALUOp**));**  ID\_EX id\_ex**(.**Clear**(**Clear0**),** **.**CLK**(**CLK**),** **.**ID\_ALUSrc**(**ALUSrc**),** **.**ID\_BEQ**(**BEQ**),** **.**ID\_BNE**(**BNE**),** **.**ID\_J**(**J**),** **.**ID\_MemToReg**(**MemToReg**),** **.**ID\_JAL**(**JAL**),** **.**ID\_RegW**(**RegW**),** **.**ID\_RegDst**(**RegDst**),** **.**ID\_ALUOp**(**ALUOp**),** **.**ID\_MemRW**(**MemRW**),** **.**ID\_NPC1**(**ID\_NPC1**),** **.**ID\_Rs**(**R\_data1**),** **.**ID\_Rt**(**R\_data2**),** **.**ID\_S\_Imm32**(**SigExt1**),** **.**ID\_Z\_Imm32**(**ZeroExt1**),** **.**ID\_IR**(**ID\_IR**),** **.**ALUSrc**(**EX\_ALUSrc**),** **.**BEQ**(**EX\_BEQ**),** **.**BNE**(**EX\_BNE**),** **.**J**(**EX\_J**),** **.**MemToReg**(**EX\_MemToReg**),** **.**JAL**(**EX\_JAL**),** **.**RegW**(**EX\_RegW**),** **.**RegDst**(**EX\_RegDst**),** **.**ALUOp**(**EX\_ALUOp**),** **.**MemRW**(**EX\_MemRW**),** **.**NPC1**(**EX\_NPC1**),** **.**Rs**(**EX\_Rs**),** **.**Rt**(**EX\_Rt**),** **.**S\_Imm32**(**EX\_S\_Imm32**),** **.**Z\_Imm32**(**EX\_Z\_Imm32**),** **.**IR**(**EX\_IR**));**  //执行  SigExt **#(**5**,**32**)** SigExt5\_32**(.**in**(**EX\_IR**[**10**:**6**]),** **.**out**(**SigExt2**));**  MUX21 **#(**32**)** mux2**(.**Sel**(**EX\_ALUSrc**),** **.**in1**(**EX\_Rt**),** **.**in2**(**mux\_7**),** **.**out**(**mux\_2**));**  MUX21 **#(**32**)** mux6**(.**Sel**(**shift**),** **.**in1**(**EX\_Rs**),** **.**in2**(**SigExt2**),** **.**out**(**mux\_6**));**  MUX21 **#(**32**)** mux7**(.**Sel**(**ZSExt**),** **.**in1**(**EX\_S\_Imm32**),** **.**in2**(**EX\_Z\_Imm32**),** **.**out**(**mux\_7**));**  ALU\_CU alu\_cu**(.**Funct**(**EX\_IR**[**5**:**0**]),** **.**ImmOp**(**EX\_IR**[**28**:**26**]),** **.**ALUOp**(**EX\_ALUOp**),** **.**ALUControl**(**ALUCtrl**),** **.**shift**(**shift**),** **.**ZeroExt**(**ZSExt**),** **.**JR**(**JR**));**  ALU alu**(.**OP**(**ALUCtrl**),** **.**A**(**mux\_11**),** **.**B**(**mux\_12**),** **.**F**(**ALUResult**),** **.**ZF**(**ZF**),** **.**OF**(**OF**),** **.**Error\_DIV0**(**Error\_DIV0**));**  SHL2\_1 shl2\_1**(.**in**(**EX\_IR**[**25**:**0**]),** **.**out**(**shl1**));**  SHL2\_2 shl2\_2**(.**in**(**EX\_S\_Imm32**),** **.**out**(**shl2**));**  Add add2**(.**A**(**EX\_NPC1**),** **.**B**(**shl2**),** **.**S**(**add\_sum2**),** **.**C**(**dummy2**));**  EX\_MA ex\_ma**(.**Clear**(**Clear1**),** **.**CLK**(**CLK**),** **.**EX\_BEQ**(**EX\_BEQ**),** **.**EX\_BNE**(**EX\_BNE**),** **.**EX\_J**(**EX\_J**),** **.**EX\_MemToReg**(**EX\_MemToReg**),** **.**EX\_JAL**(**EX\_JAL**),** **.**EX\_RegW**(**EX\_RegW**),** **.**EX\_RegDst**(**EX\_RegDst**),** **.**EX\_JR**(**JR**),** **.**EX\_ZF**(**ZF**),** **.**EX\_OF**(**OF**),** **.**EX\_MemRW**(**EX\_MemRW**),** **.**EX\_NPC1**(**EX\_NPC1**),** **.**EX\_NPC2**(**JumpAddr**),** **.**EX\_NPC3**(**add\_sum2**),** **.**EX\_NPC4**(**EX\_Rs**),** **.**EX\_ALUOut**(**ALUResult**),** **.**EX\_Rt**(**EX\_Rt**),** **.**EX\_IR**(**EX\_IR**),** **.**BEQ**(**MA\_BEQ**),** **.**BNE**(**MA\_BNE**),** **.**J**(**MA\_J**),** **.**MemToReg**(**MA\_MemToReg**),** **.**JAL**(**MA\_JAL**),** **.**RegW**(**MA\_RegW**),** **.**RegDst**(**MA\_RegDst**),** **.**JR**(**MA\_JR**),** **.**ZF**(**MA\_ZF**),** **.**OF**(**MA\_OF**),** **.**MemRW**(**MA\_MemRW**),** **.**NPC1**(**MA\_NPC1**),** **.**NPC2**(**MA\_NPC2**),** **.**NPC3**(**MA\_NPC3**),** **.**NPC4**(**MA\_NPC4**),** **.**ALUOut**(**MA\_ALUOut**),** **.**Rt**(**MA\_Rt**),** **.**IR**(**MA\_IR**));**    //访存  MUX41 **#(**32**)** mux3**(.**Sel**({**Branch**,** MA\_J**}),** **.**in1**(**0**),** **.**in2**(**MA\_NPC2**),** **.**in3**(**MA\_NPC3**),** **.**in4**(**0**),** **.**out**(**mux\_3**));**  MUX21 **#(**32**)** mux4**(.**Sel**(**PCSrc**),** **.**in1**(**add\_sum1**),** **.**in2**(**mux\_8**),** **.**out**(**mux\_4**));**  MUX21 **#(**32**)** mux8**(.**Sel**(**MA\_JR**),** **.**in1**(**mux\_3**),** **.**in2**(**MA\_NPC4**),** **.**out**(**mux\_8**));**  DMem dmem**(.**Addr**(**MA\_ALUOut**),** **.**R\_data**(**R\_data**),** **.**Mode**(**MA\_MemRW**),** **.**W\_data**(**mux\_13**),** **.**Error\_Addr**(**Error\_Addr**));**  MA\_WB ma\_wb**(.**CLK**(**CLK**),** **.**MA\_MemToReg**(**MA\_MemToReg**),** **.**MA\_JAL**(**MA\_JAL**),** **.**MA\_RegW**(**MA\_RegW**),** **.**MA\_RegDst**(**MA\_RegDst**),** **.**MA\_OF**(**MA\_OF**),** **.**MA\_NPC1**(**MA\_NPC1**),** **.**MA\_ALUOut**(**MA\_ALUOut**),** **.**MA\_MemOut**(**R\_data**),** **.**MA\_IR**(**MA\_IR**),** **.**MemToReg**(**WB\_MemToReg**),** **.**JAL**(**WB\_JAL**),** **.**RegW**(**WB\_RegW**),** **.**RegDst**(**WB\_RegDst**),** **.**OF**(**WB\_OF**),** **.**NPC1**(**WB\_NPC1**),** **.**ALUOut**(**WB\_ALUOut**),** **.**MemOut**(**WB\_MemOut**),** **.**IR**(**WB\_IR**));**    //写回  MUX21 **#(**5**)** mux1**(.**Sel**(**WB\_RegDst**),** **.**in1**(**WB\_IR**[**20**:**16**]),** **.**in2**(**WB\_IR**[**15**:**11**]),** **.**out**(**mux\_1**));**  MUX21 **#(**32**)** mux5**(.**Sel**(**WB\_MemToReg**),** **.**in1**(**WB\_ALUOut**),** **.**in2**(**WB\_MemOut**),** **.**out**(**mux\_5**));**  MUX21 **#(**5**)** mux9**(.**Sel**(**WB\_JAL**),** **.**in1**(**mux\_1**),** **.**in2**(**5'b11111**),** **.**out**(**mux\_9**));**  MUX21 **#(**32**)** mux10**(.**Sel**(**WB\_JAL**),** **.**in1**(**mux\_5**),** **.**in2**(**WB\_NPC1**),** **.**out**(**mux\_10**));**    //流水线冒险处理  forward Forward**(.**ID\_EX\_IR**(**EX\_IR**),** **.**EX\_MA\_IR**(**MA\_IR**),** **.**MA\_WB\_IR**(**WB\_IR**),** **.**ID\_EX\_MemToReg**(**EX\_MemToReg**),** **.**EX\_MA\_MemToReg**(**MA\_MemToReg**),** **.**MA\_WB\_MemToReg**(**WB\_MemToReg**),** **.**ID\_EX\_RegDst**(**EX\_RegDst**),** **.**EX\_MA\_RegDst**(**MA\_RegDst**),** **.**MA\_WB\_RegW**(**WB\_RegW**),** **.**EX\_MA\_MemRW**(**MA\_MemRW**),** **.**ID\_EX\_MemRW**(**EX\_MemRW**),** **.**MemSrc**(**MemSrc**),** **.**ALUSrc\_A**(**ALUSrc\_A**),** **.**ALUSrc\_B**(**ALUSrc\_B**));**  MUX41 **#(**32**)** mux11**(.**Sel**(**ALUSrc\_A**),** **.**in1**(**mux\_6**),** **.**in2**(**MA\_ALUOut**),** **.**in3**(**WB\_MemOut**),** **.**in4**(**0**),** **.**out**(**mux\_11**));**  MUX41 **#(**32**)** mux12**(.**Sel**(**ALUSrc\_B**),** **.**in1**(**mux\_2**),** **.**in2**(**MA\_ALUOut**),** **.**in3**(**WB\_MemOut**),** **.**in4**(**0**),** **.**out**(**mux\_12**));**  MUX21 **#(**32**)** mux13**(.**Sel**(**MemSrc**),** **.**in1**(**MA\_Rt**),** **.**in2**(**WB\_MemOut**),** **.**out**(**mux\_13**));**  hazard Hazard**(.**CLK**(**CLK**),** **.**ID\_EX\_Rt**(**EX\_IR**[**20**:**16**]),** **.**FI\_ID\_Rs**(**ID\_IR**[**25**:**21**]),** **.**FI\_ID\_Rt**(**ID\_IR**[**20**:**16**]),** **.**ID\_EX\_MemRW**(**EX\_MemRW**),** **.**EX\_MA\_ZF**(**MA\_ZF**),** **.**EX\_MA\_BEQ**(**MA\_BEQ**),** **.**EX\_MA\_BNE**(**MA\_BNE**),** **.**ID\_EX\_J**(**EX\_J**),** **.**PCWr**(**PCWr**),** **.**FI\_ID\_RegWr**(**FI\_ID\_RegWr**),** **.**Clear0**(**Clear0**),** **.**Clear1**(**Clear1**));**  **endmodule** |

（20）仿真模块

|  |
| --- |
| **module** sim\_cpu**();**  **reg** CLK**,** RST**;**    **initial** **begin**  CLK **=** 0**;**  RST **=** 1**;**  **fork**  **forever** **#**1 CLK **=** **~**CLK**;**  **#**2 RST **=** 0**;**  **join**  **end**    DataPath DP**(**CLK**,** RST**);**  **endmodule** |

5、指令集测试仿真波形

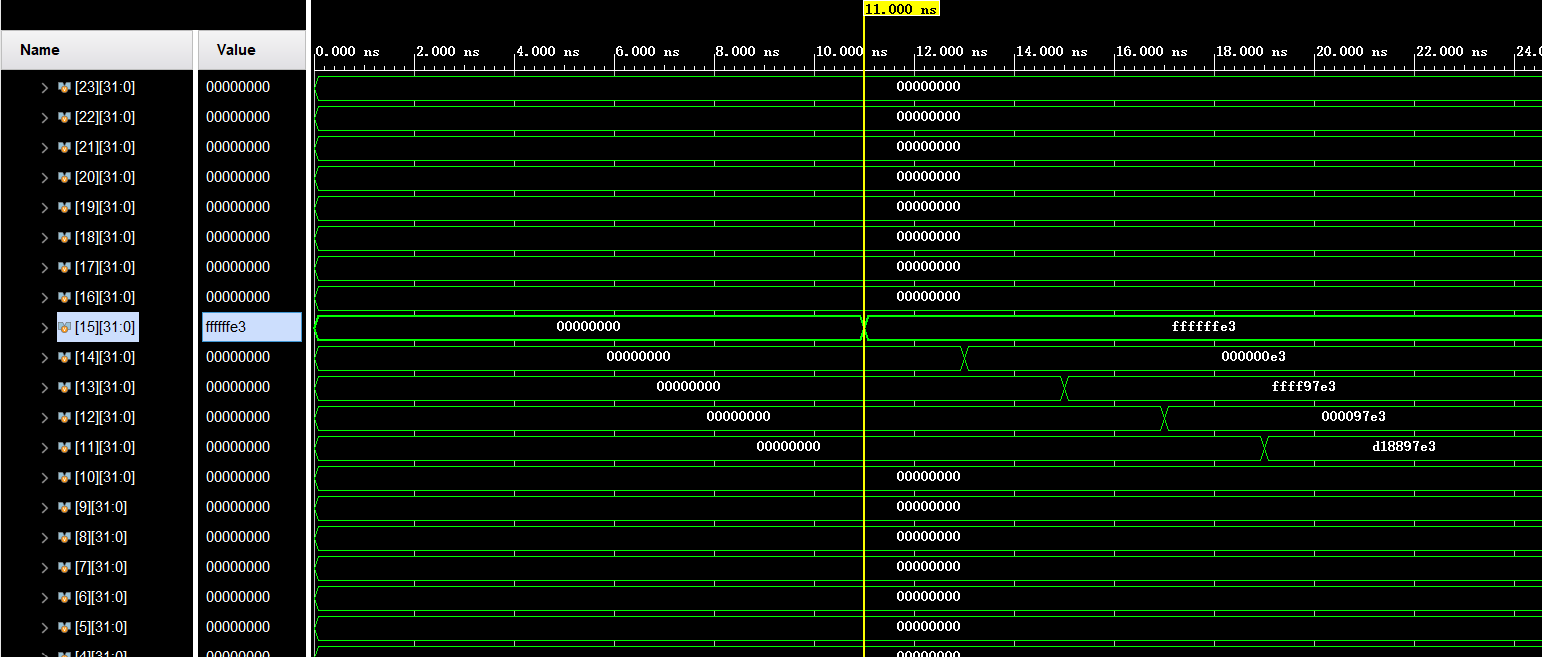
810f00fc //lb $t7 252($t0)

910e00fc //lbu $t6 252($t0)

850d00fc //lh $t5 252($t0)

950c00fc //lhu $t4 252($t0)

8d0b00fc //lw $t3 252($t0)

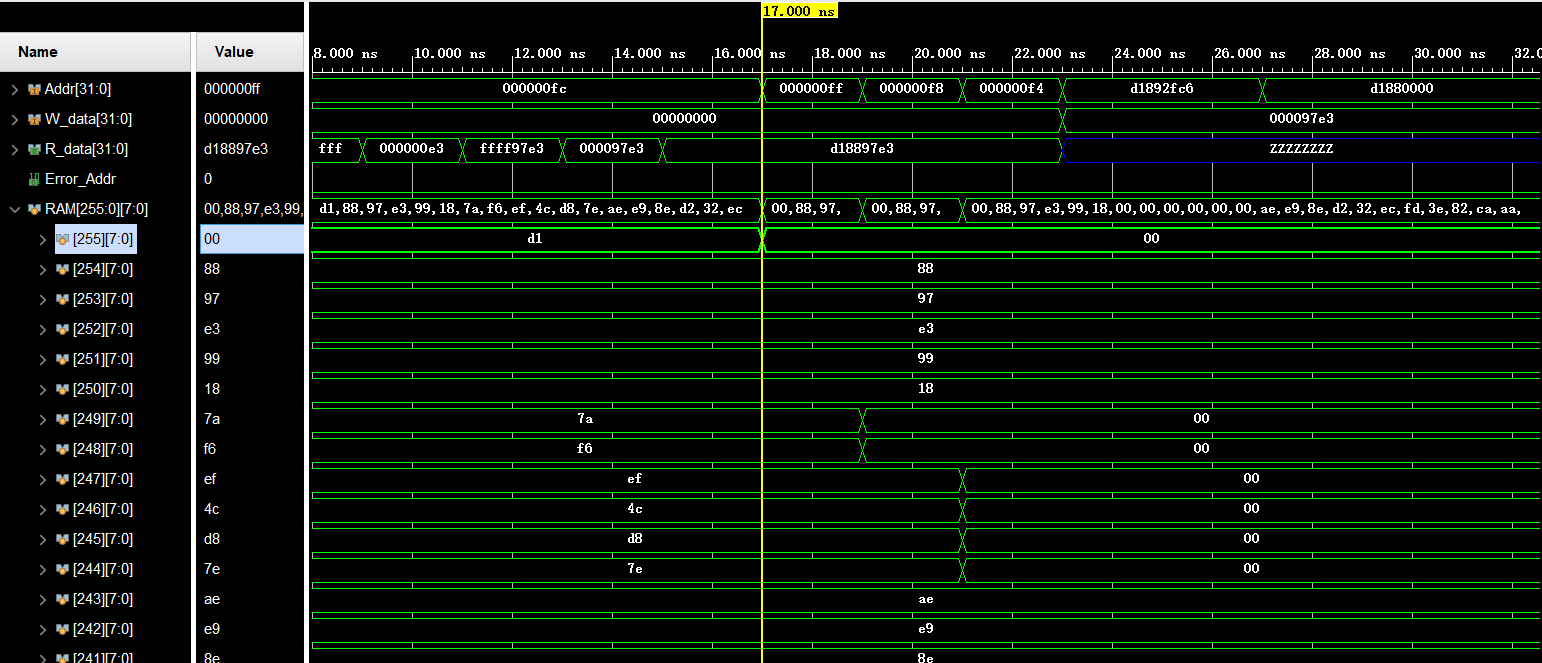


从数据存储器的地址252（十进制）中读取数据0xd18897e3，lb读取结果为0xffffffe3，lbu读取结果为0x000000e3，lh读取结果为0xffff97e3，lhu读取结果为0x000097e3，lw读取结果为0xd18897e3。

a10000ff //sb $zero 255($t0)

a50000f8 //sh $zero 248($t0)

ad0000f4 //sw $zero 244($t0)



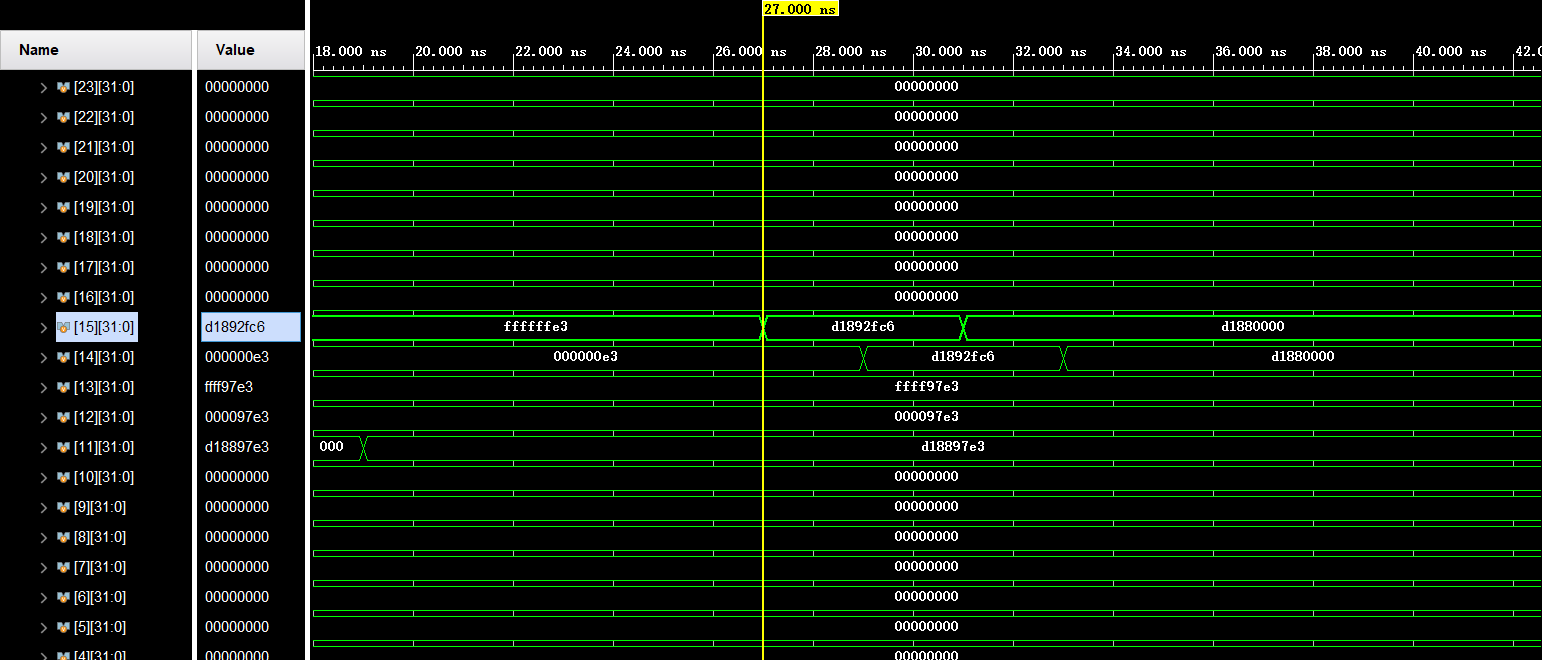
将寄存器zero中的值0存入数据存储器的相应地址，sb在地址255（十进制）存入0x00，sh在地址248（十进制）存入0x0000，sw在地址244（十进制）存入0x00000000。

016c7820 //add $t7 $t3 $t4

016c7021 //addu $t6 $t3 $t4

016c7822 //sub $t7 $t3 $t4

016c7023 //subu $t6 $t3 $t4



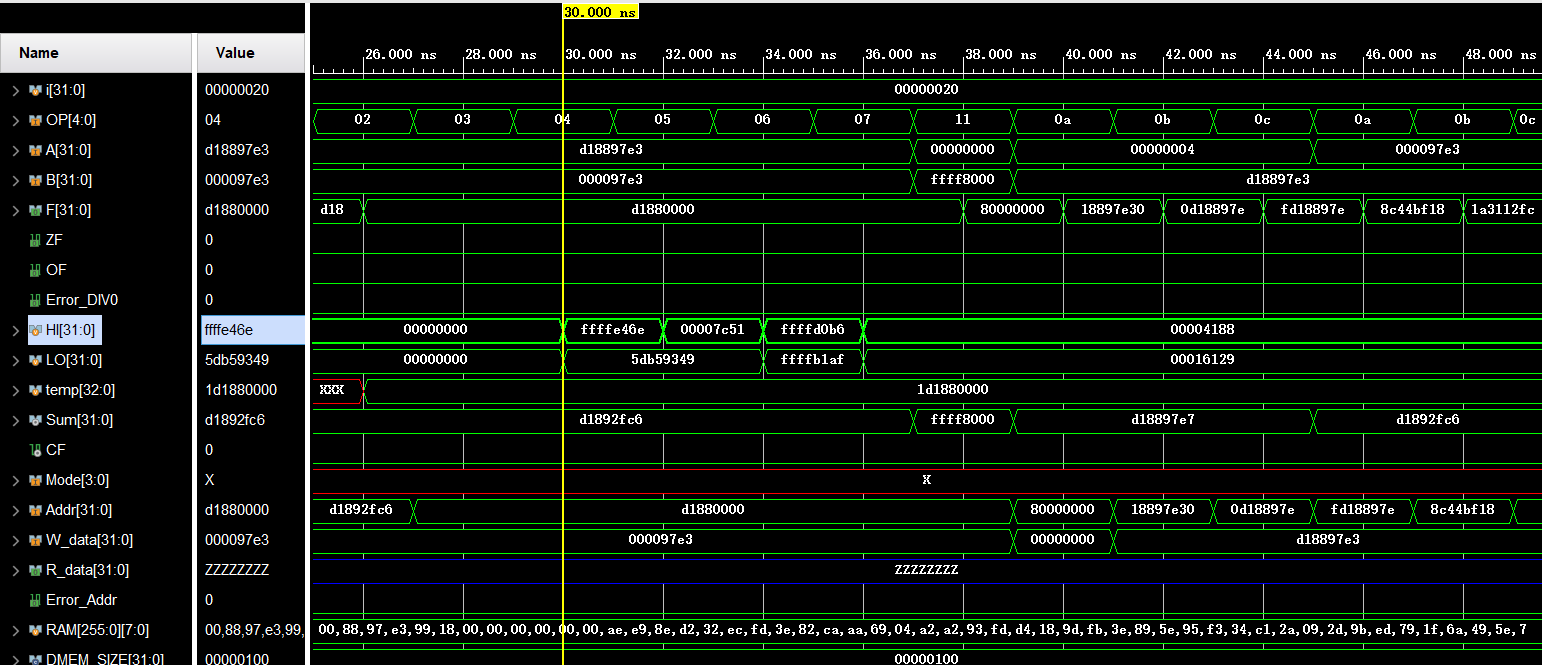
将寄存器t3（11号）中的值0xd18897e3与寄存器t4（12号）中的值0x000097e3进行加减运算。加法运算结果为0xd1892fc6，减法运算结果为0xd1880000。由于加减均未溢出，故add与addu结果相同，sub与subu结果相同。

016c0018 //mult $t3 $t4

016c0019 //multu $t3 $t4

016c001a //div $t3 $t4

016c001b //divu $t3 $t4



将寄存器t3（11号）中的值0xd18897e3与寄存器t4（12号）中的值0x000097e3进行乘除运算。mult结果为0xffffe46e\_5db59349，multu结果为0x00007c51\_5db59349，乘法结果的高32位存入HI寄存器，低32位存入LO寄存器。div商为0xffffb1af，存入LO寄存器，余数为0xffffd0b6，存入HI寄存器；divu商为0x00016129，存入LO寄存器，余数为0x00004188，存入HI寄存器。

3c0a8000 //lui $t2 0x8000

000b7900 //sll $t7 $t3 4

000b7902 //srl $t7 $t3 4

000b7903 //sra $t7 $t3 4

018b7004 //sllv $t6 $t3 $t4

018b7006 //srlv $t6 $t3 $t4

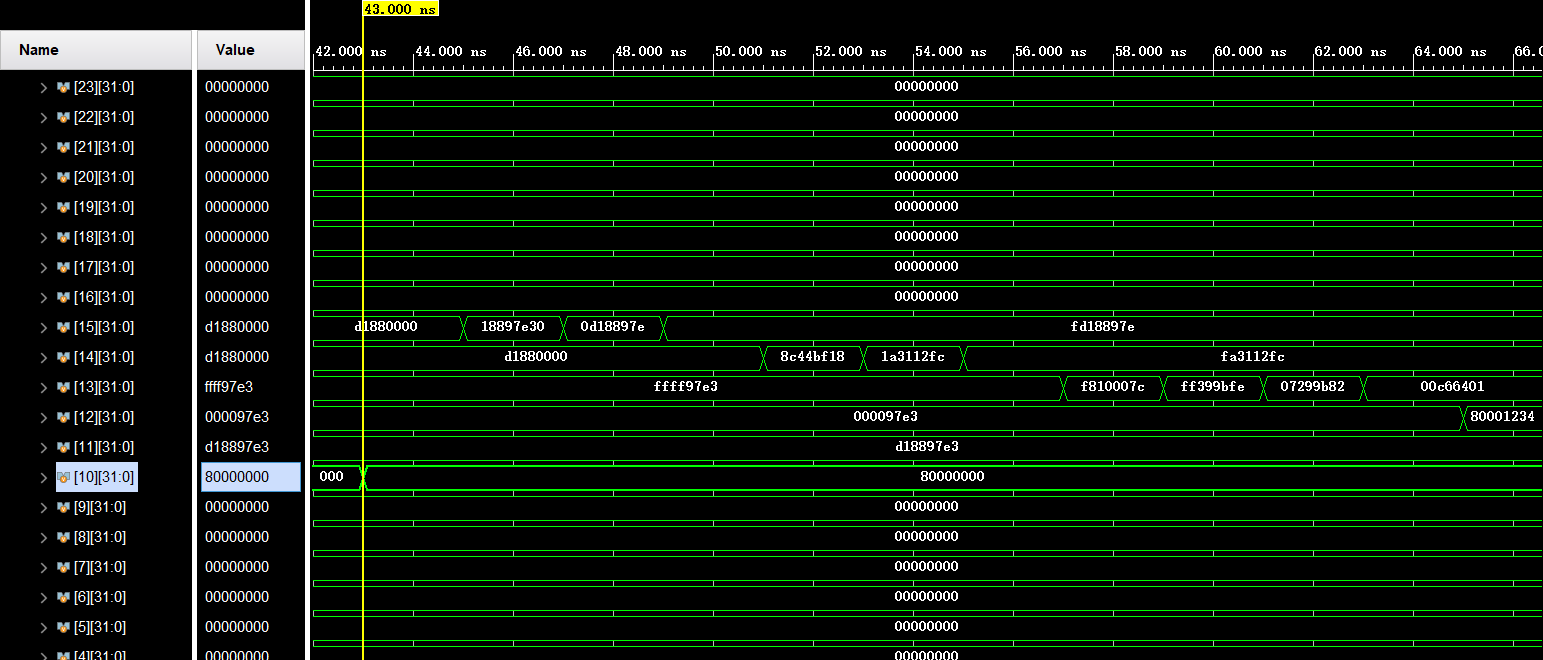
018b7007 //srav $t6 $t3 $t4

01cf6824 //and $t5 $t6 $t7

01cf6825 //or $t5 $t6 $t7

01cf6826 //xor $t5 $t6 $t7

01cf6827 //nor $t5 $t6 $t7



lui将寄存器t2（10号）置为0x80000000。后续6条指令都是对寄存器t3（11号）中的值0xd18897e3进行移位操作，sll逻辑左移4位结果为0x18897e30，srl逻辑右移4位结果为0x0d18897e，sra算术右移4位结果为0xfd18897e。寄存器t4（12号）中的值为0x000097e3，其最低5位的值为3（十进制）。sllv逻辑左移3位结果为0x8c44bf18，srlv逻辑右移3位结果为0x1a3112fc，srav算术右移3位结果为0xfa3112fc。最后4条指令都是对寄存器t6（14号）中的值0xfa3112fc与寄存器t7（15号）中的值0xfd18897e进行逻辑运算。and的结果为0xf810007c，or的结果为0xff399bfe，xor的结果为0x07299b82，nor的结果为0x00c66401。

**注意，srav指令的目的寄存器与and指令的源寄存器都是t6，两条指令间存在数据冒险。**

214c1234 //addi $t4 $t2 0x1234

254b1234 //addiu $t3 $t2 0x1234

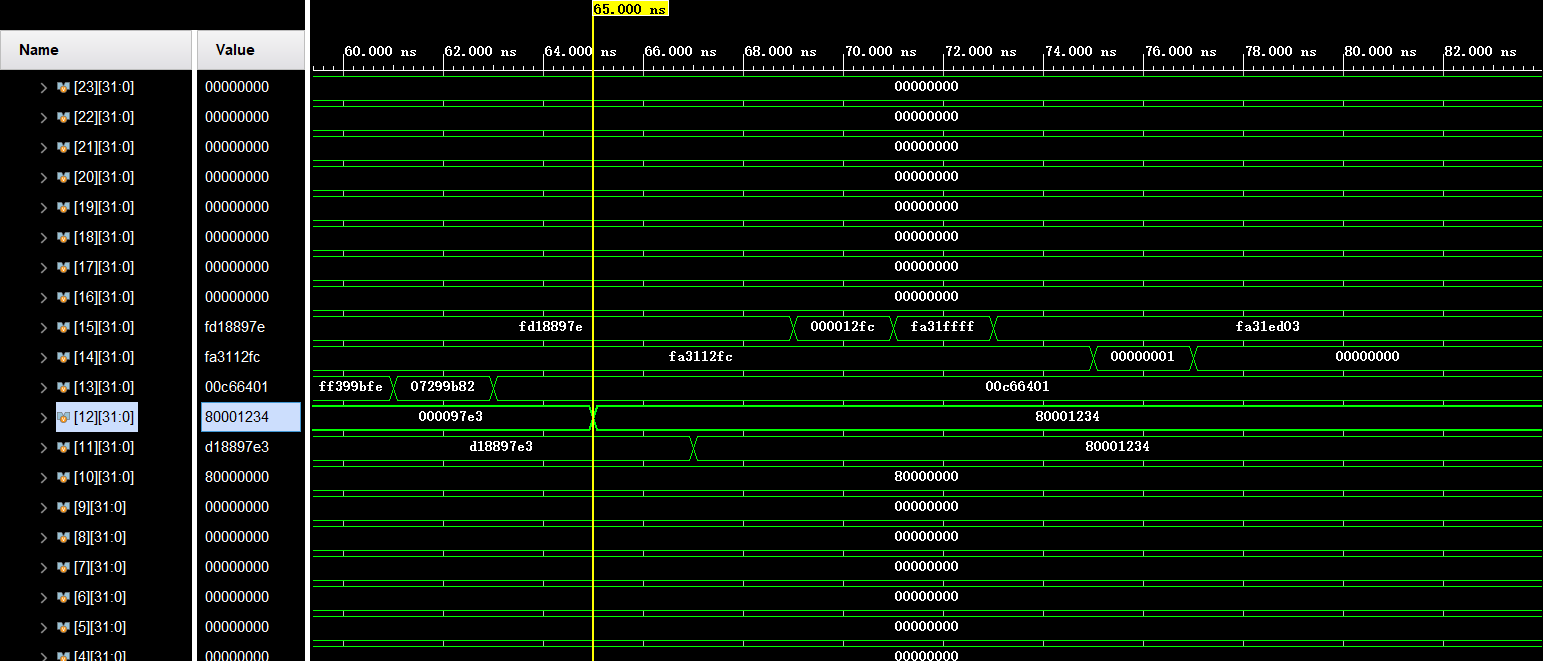
31cfffff //andi $t7 $t6 0xffff

35cfffff //ori $t7 $t6 0xffff

39cfffff //xori $t7 $t6 0xffff

294e0001 //slti $t6 $t2 0x0001

2d4e0001 //sltiu $t6 $t2 0x0001



寄存器t2（10号）的值为0x80000000，与立即数0x1234相加结果为0x80001234，由于结果未溢出，故addi与addiu的结果相同。寄存器t6（14号）的值为0xfa3112fc，与立即数0xffff进行逻辑运算后，andi结果为0x000012fc，ori结果为0xfa31ffff，xori结果为0xfa31ed03。最后两条指令比较0x80000000与立即数0x0001，小于立即数置1。由于slti是有符号比较，0x80000000的最高位1视为符号位，故0x80000000<0<0x0001，结果为1；而sltiu是无符号比较，故0x80000000>0x0001，结果为0。

152a0001 //bne $t1 $t2 loop1

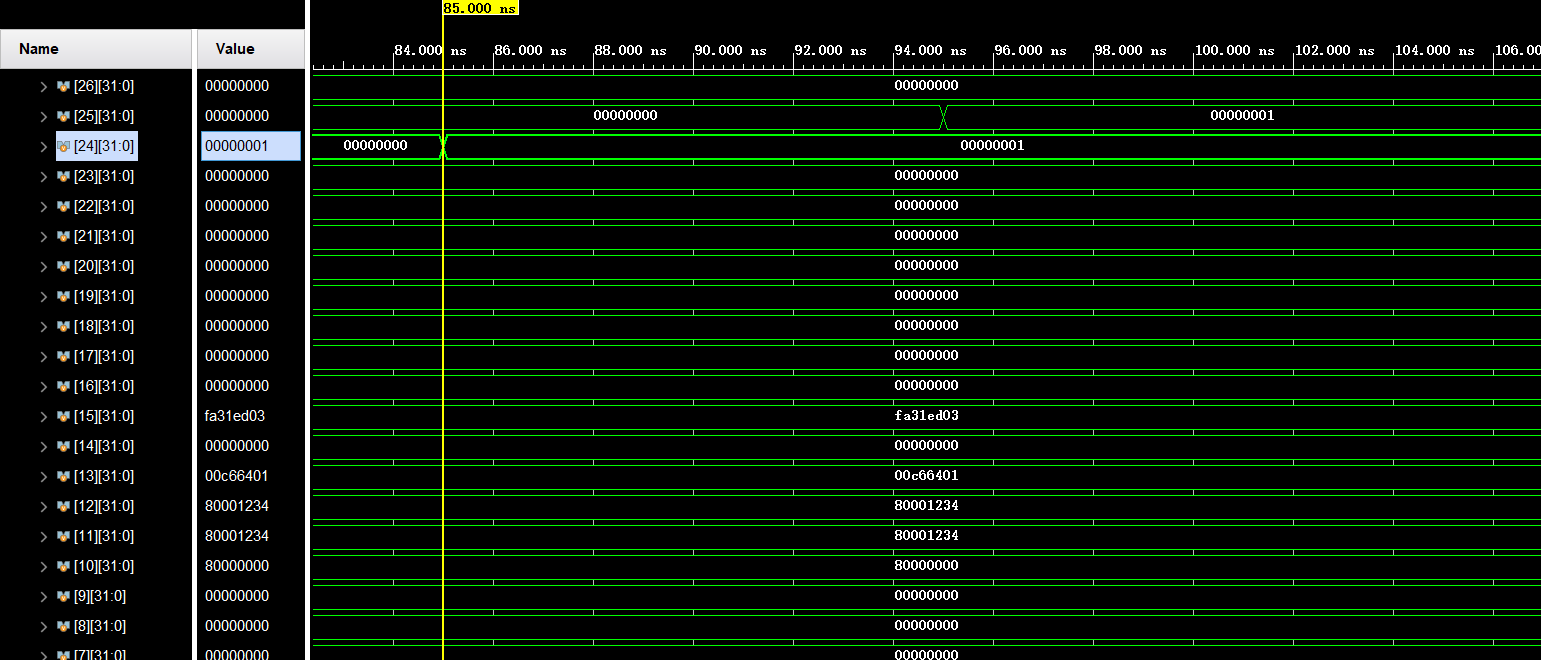
3718ffff //ori $t8 $t8 0xffff

20180001 //loop1: addi $t8 $zero 1

12110001 //beq $s0 $s1 loop2

3718ffff //ori $t8 $t8 0xffff

20190001 //loop2: addi $t9 $zero 1



寄存器t1（9号）值为0x00000000，寄存器t2（10号）值为0x8000000，两者不相等，跳过下一条指令ori，执行loop1的addi指令，寄存器t8（24号）结果为1。寄存器s0（16号）值为0x00000000，寄存器s1（17号）值为0x0000000，两者相等，跳过下一条指令ori，执行loop2的addi指令，寄存器t8（24号）结果不变，寄存器t9（25号）结果为1。

0800002a //j jump1

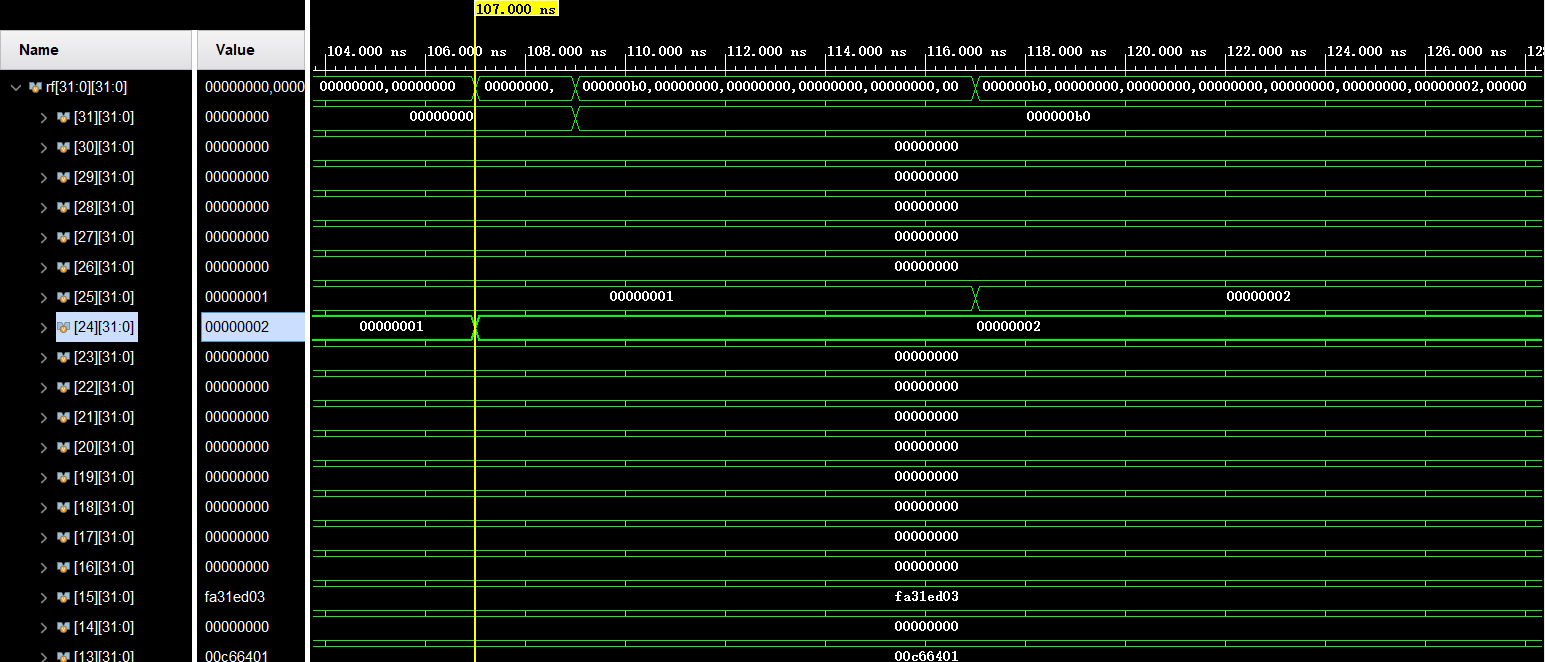
33180000 //andi $t8 $t8 0x0000

20180002 //jump1: addi $t8 $zero 2

0c00002d //jal jump2

33180000 //andi $t8 $t8 0x0000

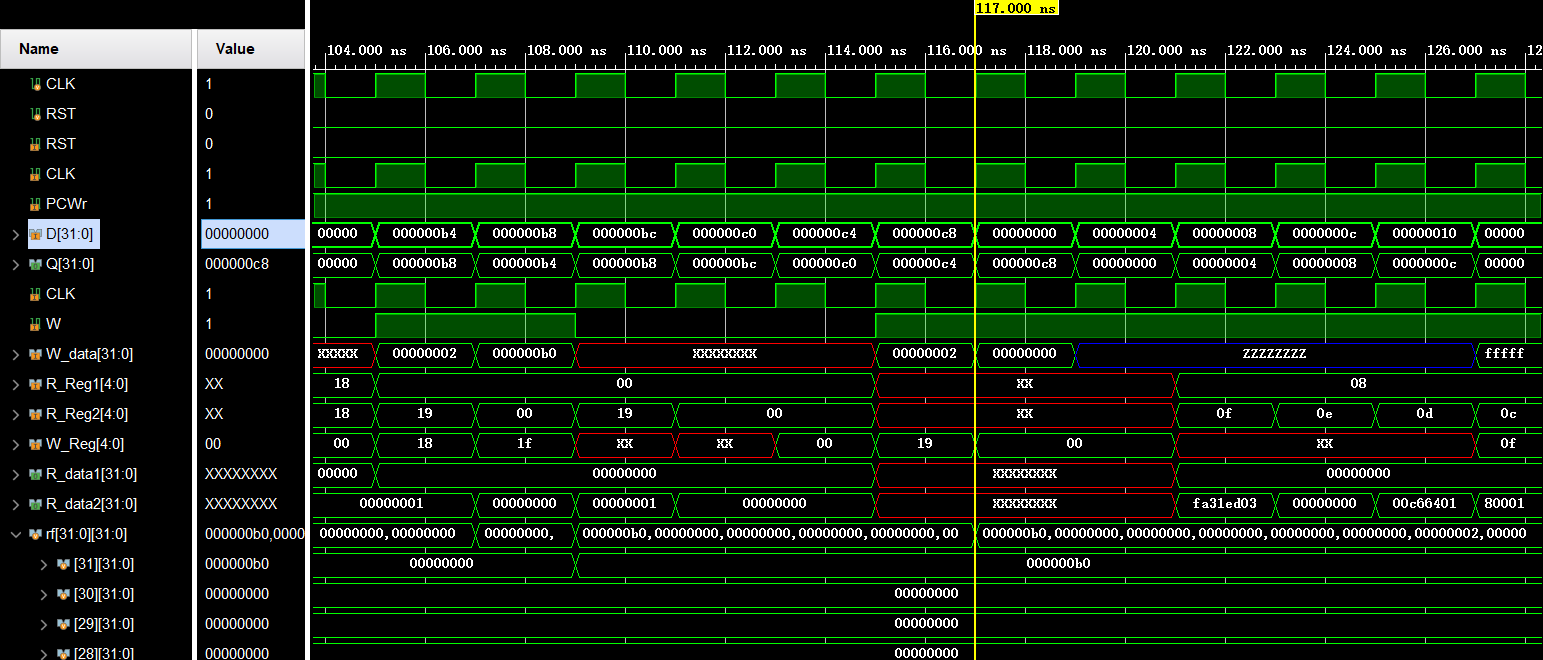
20190002 //jump2: addi $t9 $zero 2



j指令跳过下一条指令andi，执行jump1的addi指令，寄存器t8（24号）结果为2。jal指令跳过下一条指令andi，执行jump2的addi指令，寄存器t8（24号）结果不变，寄存器t9（25号）结果为2，同时，jal指令还将原本要执行的下一条指令地址0x000000b0赋给寄存器ra（31号）。

00000000 //nop

00000008 //jr $zero



nop指令相当于将寄存器zero中的值0逻辑左移0位后再写回到寄存器zero，实际上什么都没做。jr指令跳转到寄存器zero中的地址继续执行，从程序计数器PC的值中可以看出程序确实又从零地址开始执行。

**五、调试和心得体会**

在调试过程中遇到的主要问题是一个信号值并不是非1即0的，还可能存在x态与z态，因此在使用assign语句时就会出现意想不到的结果。此外，在比较两个信号值时，有时需要使用“===”来严格匹配，对于case语句中出现x态的情况则应使用“casex”来执行匹配。

通过本次实验，我一方面对于Verilog语言掌握得更加熟练，另一方面也对计算机系统硬件原理有了更深刻的认识，也了解了程序在CPU上执行的具体过程，以及流水CPU的实现原理和冒险的解决方法。