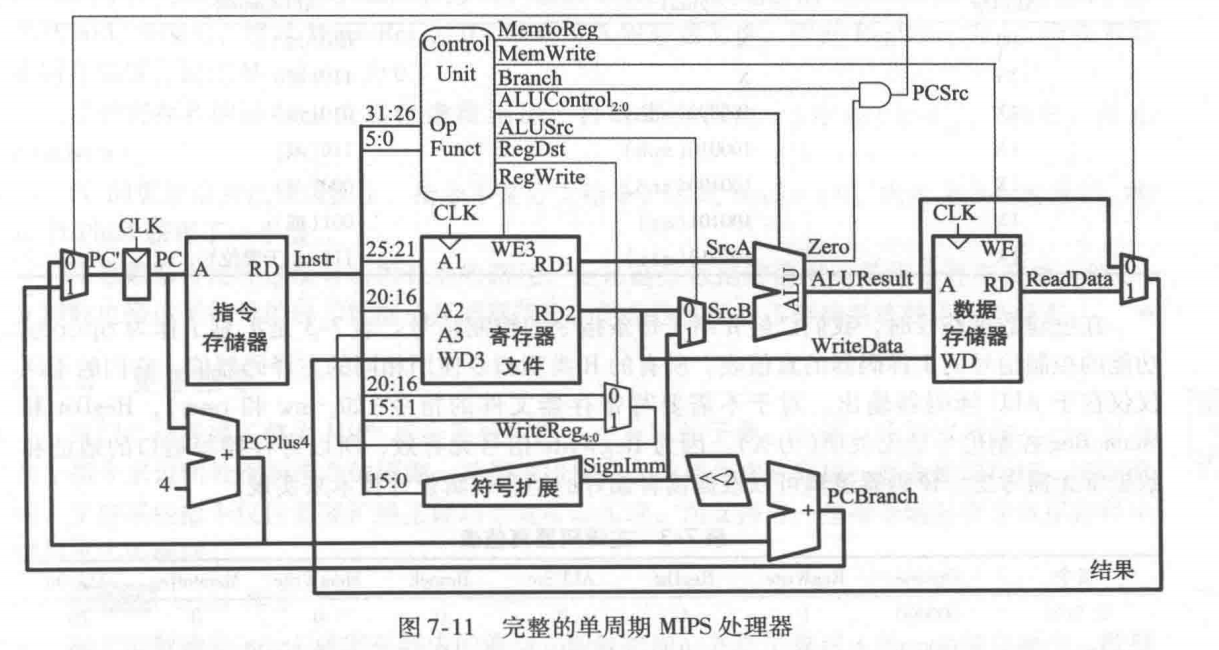
计算机体系结构实验2：32位单周期处理器设计

15307110200 廖文

1. 实验要求
2. 仔细阅读**教材**7.3节(P231-239)内容
3. 参考**教材**7.6节(P270-279)**代码**，完成MIPS单周期处理器设计
4. 用**教材**图7-60(P276)测试代码测试上述设计
5. 增加ori、bne**等**指令，修改上述MIPS单周期处理器：
   1. 修改MIPS处理器原理图(cf图7-14)
   2. 修改主译码器(cf表7-3)、ALU译码器真值表(cf表7-2)

修改Verilog代码

1. 编写MIPS汇编测试代码，测试修改后的MIPS处理器
2. MIPS处理器的设计
3. 教材上给出了单周期MIPS处理器的设计图如下

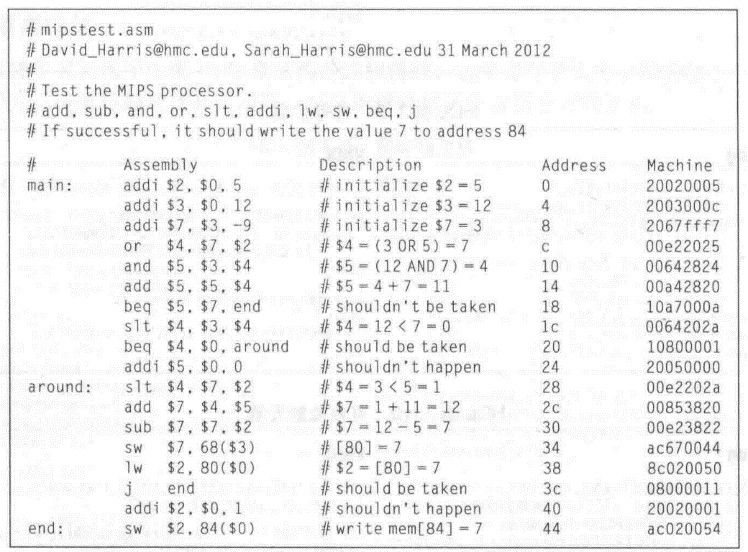


1. 根据教材所给的处理器代码和测试代码测试上述设计。

其中支持的指令包括

R型算术/逻辑指令：and, sub, and, or, slt

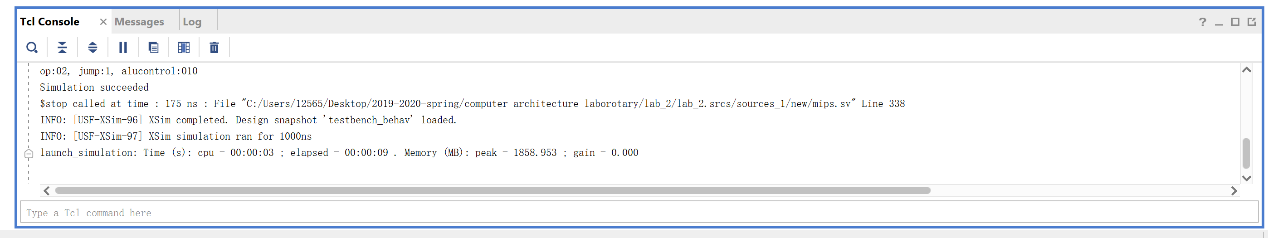
I型算术/逻辑指令：addi

存储器指令：lw, sw

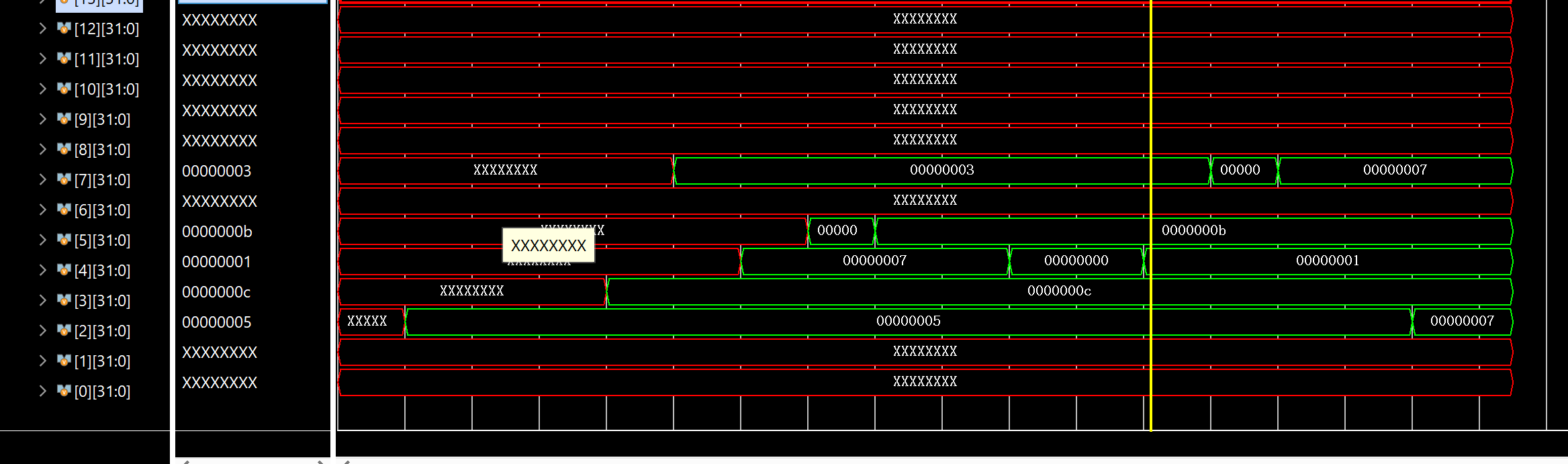
分支指令：beq，j

测试代码如下右图所示。

从命令行的输出可以看到，程序成功执行。



为了更加直观的展示测试程序执行期间处理器的状态，我们在模拟的结果中输出寄存器的值如下。



1. 增加指令

考虑增加ori和bne指令。首先在教材附录1中找到这两条指令的描述。

C:\Users\12565\AppData\Local\TEMP_~1\1593183261(1).png

C:\Users\12565\AppData\Local\TEMP_~1\1593183449(1).png

已经支持的两条相似的指令为

C:\Users\12565\AppData\Local\TEMP_~1\1593183412(1).png

C:\Users\12565\AppData\Local\TEMP_~1\1593183347(1).png

ori指令在addi指令的基础上进行修改即可，不需要增加额外的部件和标志位。

bne指令在beq指令的基础上进行修改，我们增加了一个标志位bne，标识该分支指令在连个寄存器不相同的时候进行跳转。修改后的部件的代码如下。

主译码器的真值表与ALU译码器真值表这里没有画出，对真值表的逻辑的修改见代码中背景为黄色的部分。

//decide the alu control bit given the funct field and aluop field of the instruction

//combinational logic

**module** aludec**(**

**input** **logic** **[**5**:**0**]** funct**,**

**input** **logic** **[**1**:**0**]** aluop**,**

**output** **logic** **[**2**:**0**]** alucontrol**);**

**always\_comb**

**case(**aluop**)**

2'b00**:** alucontrol **<=** 3'b010**;** // add (for lw/sw/addi) end;

2'b01**:** alucontrol **<=** 3'b110**;** // sub (for beq) architecture behave of aludec is

**2'b11: alucontrol <= 3'b001;**

**default:** **case(**funct**)** // R-type instructions begin

6'b100000**:** alucontrol **<=** 3'b010**;** // add process(al1 ) begin

6'b100010**:** alucontrol **<=** 3'b110**;** // sub case aluop is

6'b100100**:** alucontrol **<=** 3'b000**;** // and when "00" => alucontrol <= "010" ; -- add (for lw/sw/addi )

6'b100101**:** alucontrol **<=** 3'b001**;** //or when "01" => alucontrol <= "110"; -- sub (for beq)

6'b101010**:** alucontrol **<=** 3'b111**;** // sit when others => case funct is - R-type instructions

**default:** alucontrol **<=** 3'bxxx**;** // ??? when "100000" => alucontrol <= "010"; - add

**endcase** // when "100010" => alucontrol <= "110"; - sub

**endcase** // when "100100" => alucontrol <= "000"; -- and

**endmodule**

//decide controls according to the opcode of instruction

//combinational logic

**module** maindec**(**

**input** **logic** **[**5**:**0**]** op**,**

**output** **logic** memtoreg**,** memwrite**,**//

**output** **logic** branch**,** **bne,** alusrc**,**

**output** **logic** regdst**,** regwrite**,**

**output** **logic** jump**,**

**output** **logic** **[**1**:**0**]** aluop

**);**

**logic** **[**9**:**0**]** controls**;**

**assign** **{**regwrite**,** regdst**,** alusrc**,** branch**,** bne**,** memwrite**,** memtoreg**,** jump**,** aluop**}** **=** controls**;**

**always\_comb**

**case(**op**)**

**6'b000000: controls <= 10'b1100000010; // RTYPE**

**6'b100011: controls <= 10'b1010001000; // LW**

**6'b101011: controls <= 10'b0010010000; // SW**

**6'b000100: controls <= 10'b0001000001; // BEQ**

**6'b000101: controls <= 10'b0001100001; // BNE**

**6'b001000: controls <= 10'b1010000000; // ADDI**

**6'b001101: controls <= 10'b1010000011; // ORI**

**6'b000010: controls <= 10'b0000000100; // J**

**default: controls <= 10'bxxxxxxxxxx; // illegal op**

**endcase**

**endmodule**

//combinational logic

//parse the instruction and decide the params of the datapath

**module** controller**(**

**input** **logic** **[**5**:**0**]** op**,** funct**,**

**input** **logic** zero**,**

**output** **logic** memtoreg**,** memwrite**,**

**output** **logic** pcsrc**,** alusrc**,**

**output** **logic** regdst**,** regwrite**,**

**output** **logic** jump**,**

**output** **logic** **[**2**:**0**]** alucontrol**);**

**logic** **[**1**:**0**]** aluop**;**

**logic** branch**;**

**logic** bne**;**

maindec md**(**op**,** memtoreg**,** memwrite**,** branch**,** bne**,** alusrc**,** regdst**,** regwrite**,** jump**,** aluop**);**

aludec ad**(**funct**,** aluop**,** alucontrol **);**

**assign** pcsrc **=** branch **&** **(**bne**^**zero**);**

**endmodule**

我们修改后的测试代码如下，其中使用了ori和bne指令。

RAM**[**0**]** **=** 32'h20020004**;**

RAM**[**1**]** **=** 32'h20030008**;**

RAM**[**2**]** **=** 32'h34420001**;**//ori $2 1

RAM**[**3**]** **=** 32'h34630004**;**//ori $3 4

RAM**[**4**]** **=** 32'h2067fff7**;**

RAM**[**5**]** **=** 32'h00e22025**;**

RAM**[**6**]** **=** 32'h00642824**;**

RAM**[**7**]** **=** 32'h00a42820**;**

RAM**[**8**]** **=** 32'h10a7000a**;**

RAM**[**9**]** **=** 32'h0064202a**;**

RAM**[**10**]** **=** 32'h10800001**;**

RAM**[**11**]** **=** 32'h20050000**;**

RAM**[**12**]** **=** 32'h00e2202a**;**

RAM**[**13**]** **=** 32'h00853820**;**

RAM**[**14**]** **=** 32'h00e23822**;**

RAM**[**15**]** **=** 32'hac670044**;**

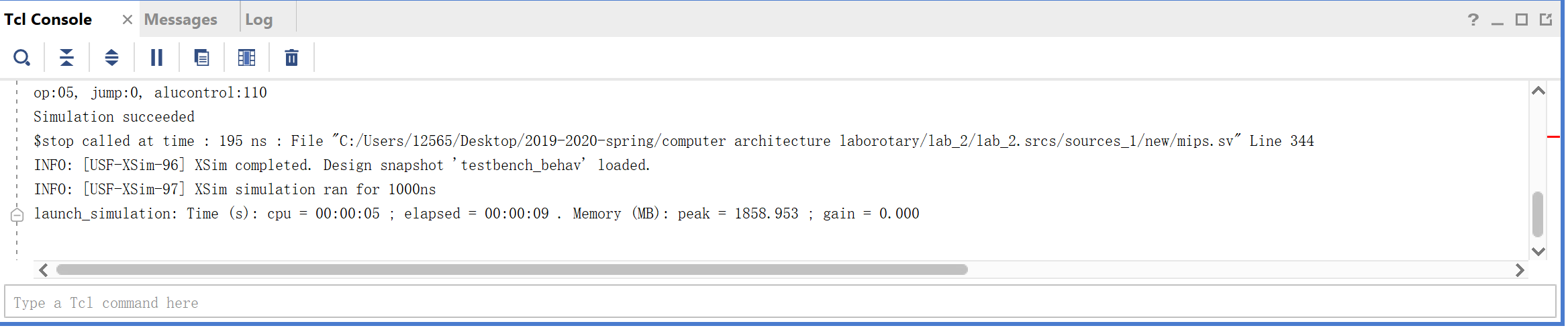
RAM**[**16**]** **=** 32'h8c020050**;**

RAM**[**17**]** **=** 32'h14440001**;**//bne $2, $4, #end

RAM**[**18**]** **=** 32'h20020001**;**

RAM**[**19**]** **=** 32'hac020054**;**

模拟后的命令行输出如下，表明模拟成功。



寄存器的值为

