计算机系统设计

课程实验报告

实验名称　相对简单CPU的设计

学生姓名　张宁

学生学号　201608010312

专业班级　智能1601

实验名称：相对简单 CPU 电路设计

实验目标：利用 VHDL 设计相对简单 CPU 的电路并验证。

实验要求

- 采用 VHDL 描述电路及其测试平台
- 采用时序逻辑设计电路
- 采用从 1 累加到 n 的程序进行测试

实验内容

相对简单 CPU 的设计需求

CPU 设计步骤：

1.相对简单 CPU 的设计规格说明：

- 地址总线 16 位，数据总线 8 位
- 有一个 8 位累加寄存器 AC，一个 8 位通用寄存器 R，一个 1 位的零标志
- 有一个 16 位 AR 寄存器，一个 16 位程序计数器 PC，一个 8 位数据寄存器 DR，一个 8 位指令寄存器 IR，一个 8 位临时寄存器 TR
- 有 16 条指令，每条指令 1 个或 3 个字节，其中操作码 8 位。3 字节的指令有 16 位的地址

2.指令集结构：

指令	指令码	操作
NOP	0000 0000	无
LDAC	0000 0001 Γ	AC←M[Γ]
STAC	0000 0010 Γ	M[Γ]←AC
MVAC	0000 0011	R←AC
MOV R	0000 0100	AC←R
JUMP	0000 0101 Γ	GOTO Γ
JMPZ	0000 0110 Γ	IF (Z = 1) THEN GOTO Γ
JPNZ	0000 0111 Γ	IF (Z = 0) THEN GOTO Γ
ADD	0000 1000	AC←AC + R, IF (AC + R = 0) THEN Z←1 ELSE Z←0
SUB	0000 1001	AC←AC - R, IF (AC - R = 0) THEN Z←1 ELSE Z←0
INAC	0000 1010	AC←AC + 1, IF (AC + 1 = 0) THEN Z←1 ELSE Z←0
CLAC	0000 1011	AC←0, Z←1
AND	0000 1100	AC←AC∧R, IF (AC∧R = 0) THEN Z←1 ELSE Z←0
OR	0000 1101	AC←AC∨R, IF (AC∨R = 0) THEN Z←1 ELSE Z←0
XOR	0000 1110	AC←AC⊕R, IF (AC⊕R = 0) THEN Z←1 ELSE Z←0
NOT	0000 1111	AC←AC', IF (AC' = 0) THEN Z←1 ELSE Z←0

3. 一些寄存器

- 16 位的地址寄存器 AR：通过引脚 A[15..0]向存储器提供地址。

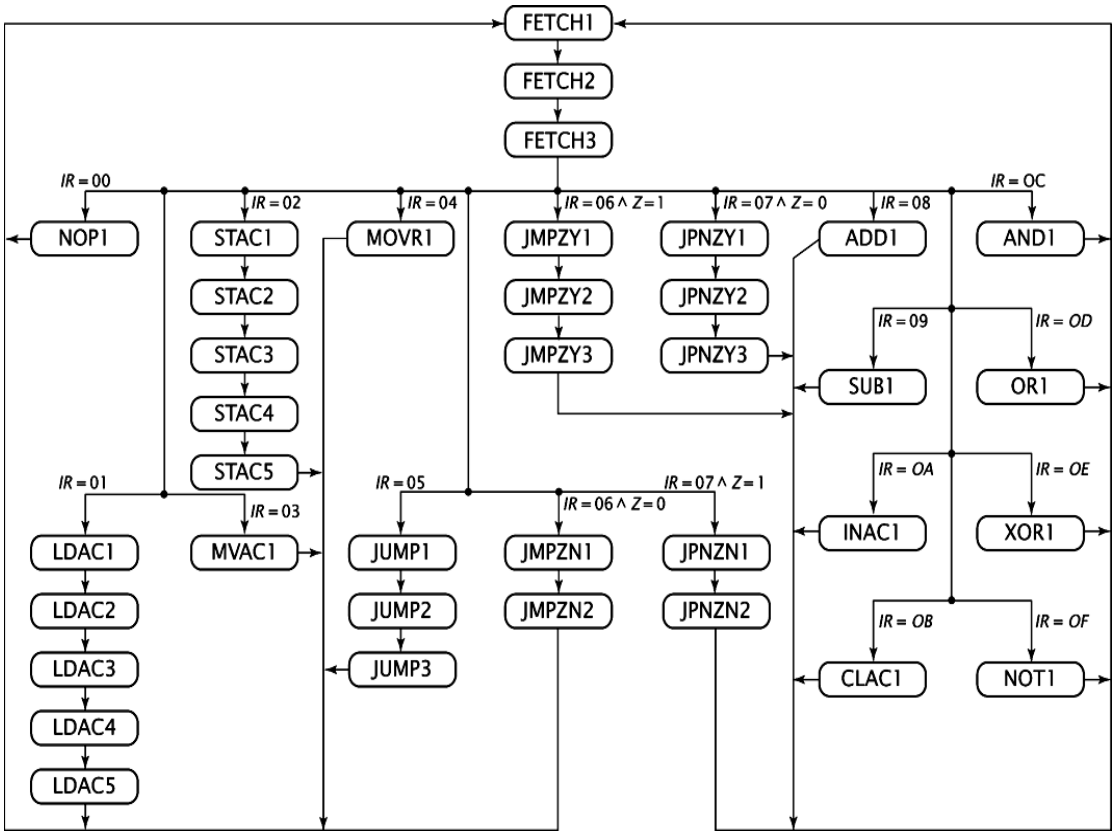
- 16 位的程序计数器 PC：存放的是将要执行的下一条指令的地址，或者指令需要的下一个操作数的地址。
- 8 位的数据寄存器 DR：通过 D[7..0]从存储器中接收指令和数据并且向存储器传送数据。
- 8 位的指令寄存器 IR：存放的是从存储器中取出来的操作码。
- 8 位的临时寄存器 TR：在指令执行过程中，临时存储数据。（程序员不能访问）

相对简单 CPU 设计方案

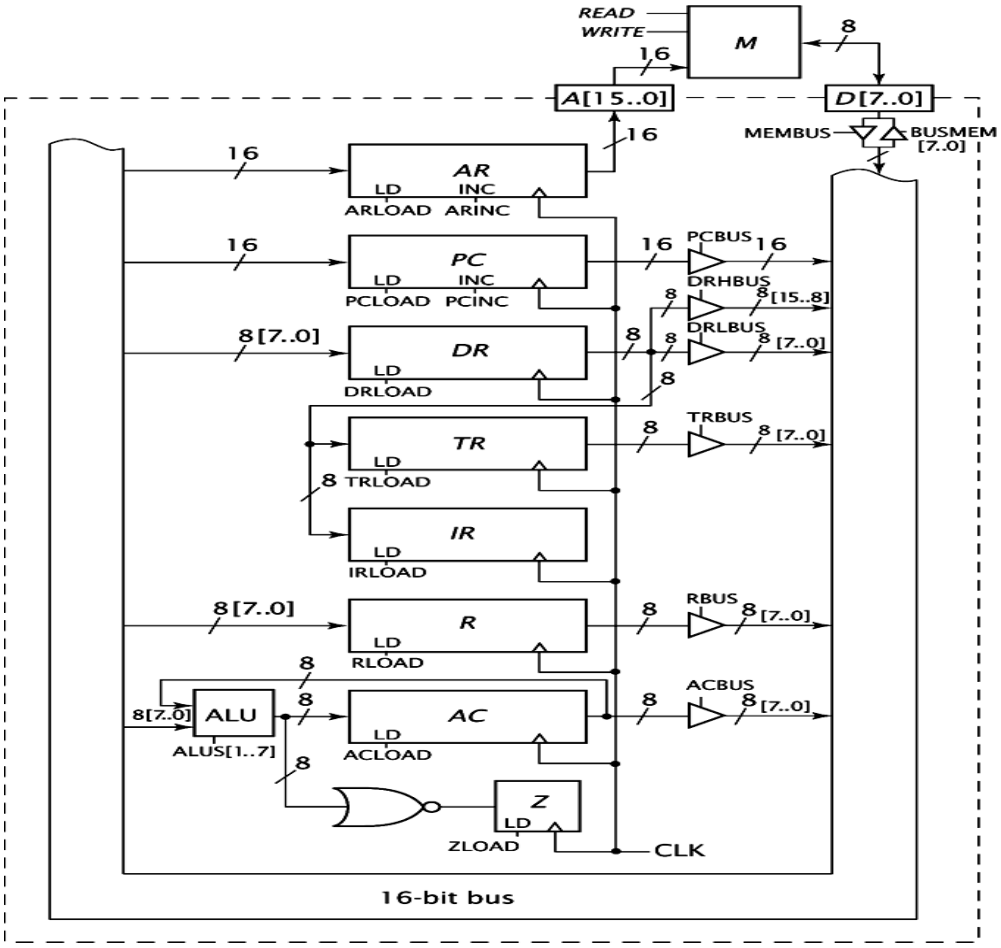
相对简单 CPU 的设计方案请详见课件，主要思路如下：

1、明确 CPU 的功能、目的和基本规格；
2、设计指令集结构；
3、取指令、译码（画出状态图）；
4、执行指令（明确指令的状态）；
5、创建数据通路；
6、设计 ALU 等；
7、设计控制单元；
8、产生 CPU 的状态。

根据指令执行过程得到状态之间的转换如下：



根据每个指令的状态转换，数据之间的传输关系构建数据通路如下：



内存设计：

例如计算 1+2+3+4+5 的值，指令执行过程如下：

指令	1st Loop	2nd Loop	3rd Loop	4th Loop	5th Loop
CLAC	AC＝0				
STAC total	total＝0				
STAC i	i＝0				
LDAC i	AC＝0	AC＝1	AC＝2	AC＝3	AC＝4
INAC	AC＝1	AC＝2	AC＝3	AC＝4	AC＝5
STAC i	i＝1	i＝2	i＝3	i＝4	i＝5
MVAC	R＝1	R＝2	R＝3	R＝4	R＝5
LDAC total	AC＝0	AC＝1	AC＝3	AC＝6	AC＝10
ADD	AC＝1	AC＝3	AC＝6	AC＝10	AC＝15
STAC total	total＝1	total＝3	total＝6	total＝10	total＝15
LDAC n	AC＝5	AC＝5	AC＝5	AC＝5	AC＝5
SUB	AC＝4, Z＝0	AC＝3, Z＝0	AC＝2, Z＝0	AC＝1, Z＝0	AC＝0, Z＝0
JPNZ Loop	JUMP	JUMP	JUMP	JUMP	NO JUMP

所以根据要实现的功能，mem 里面保存的内容如下：

```
signal memdata: memtype(4095 downto 0) := (
0 => RSCLAC,
1 => RSSTAC,--total=0
2 => std_logic_vector(to_unsigned(total_addr, 8)),
3 => X"00",
4 => RSSTAC,--i=0
5 => std_logic_vector(to_unsigned(i_addr, 8)),
6 => X"00",
7 => RSLDAC,  -- loop
8 => std_logic_vector(to_unsigned(i_addr, 8)),
9 => X"00",
10 => RSINAC,
11 => RSSTAC,
12 => std_logic_vector(to_unsigned(i_addr, 8)),
13 => X"00",
14 => RSMVAC,
15 => RSLDAC,
16 => std_logic_vector(to_unsigned(total_addr, 8)),
17 => X"00",
18 => RSADD, --ac=ac+r
19 => RSSTAC,
20 => std_logic_vector(to_unsigned(total_addr, 8)),
21 => X"00",
22 => RSLDAC,
23 => std_logic_vector(to_unsigned(n_addr, 8)),
24 => X"00",
25 => RSSUB,
26 => RSJPNZ,
27 => std_logic_vector(to_unsigned(loop_addr, 8)),


    28 => X"00",
    29 => X"00",  -- total
    30 => X"00",  -- i
    31 => "00000101",  -- n
    others => RSNOP
);
```

代码分为三个文件：

cpu.vhd：

- 根据状态图对指令执行状态译码
  - 其中 LDAC、STAC 需要 5 个状态完成，JUMP 需要 3 个状态完成，其余均一个状态完成。

```
constant ldac1:     std_logic_vector(5 downto 0) := "001110";
constant ldac2:     std_logic_vector(5 downto 0) := "001111";
constant ldac3:     std_logic_vector(5 downto 0) := "010000";
constant ldac4:     std_logic_vector(5 downto 0) := "010001";
constant ldac5:     std_logic_vector(5 downto 0) := "010010";
```

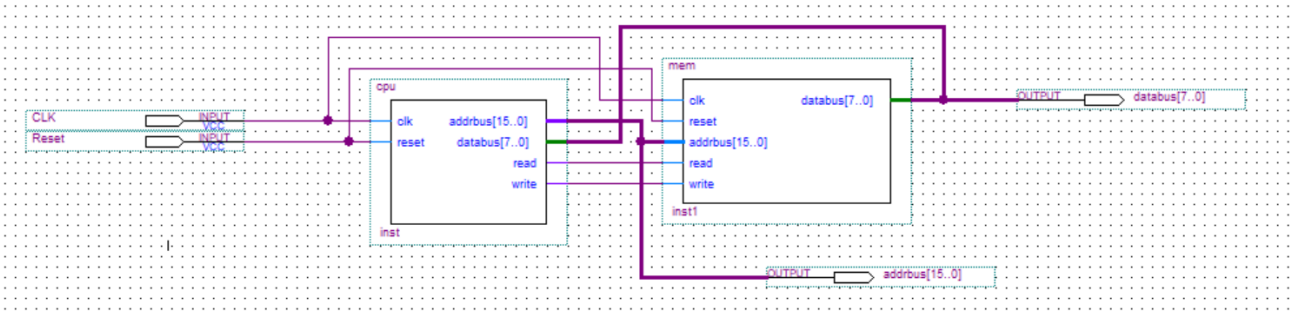
- 时钟上升沿时，根据指令更新寄存器的值
- 产生下一状态的进程（根据当前指令产生下一状态）
- 对每个状态译码（信号如何变化）

mem.vhd:

在 mem.vhd 文件中，验证 1+2+…+n。

comp.bdf:

在 comp 工程中，把 cpu.vhd、rsisa.vhd、mem.vhd 三个文件添加进来，通过画图的方式连接。



测试平台

Windows10　Quartus II 9.0

仿真：Quartus 下进行波形仿真

测试输入

我们采用从 1 累加到 n 的程序作为测试输入：

用相对简单 CPU 编程计算 1＋2＋……＋n，则其高级语言的代码片断如下：

```
total ＝0;
FOR i＝1 TO n DO {total＝total＋i};
```

可以把数值 n 存储在标明为 n 的存储单元中，结果存在标明为 total 的内存单元处，内存单元 i 用于存储求和的次数。确定运算步骤如下：

1：total＝0，i＝0

2：i＝i＋1

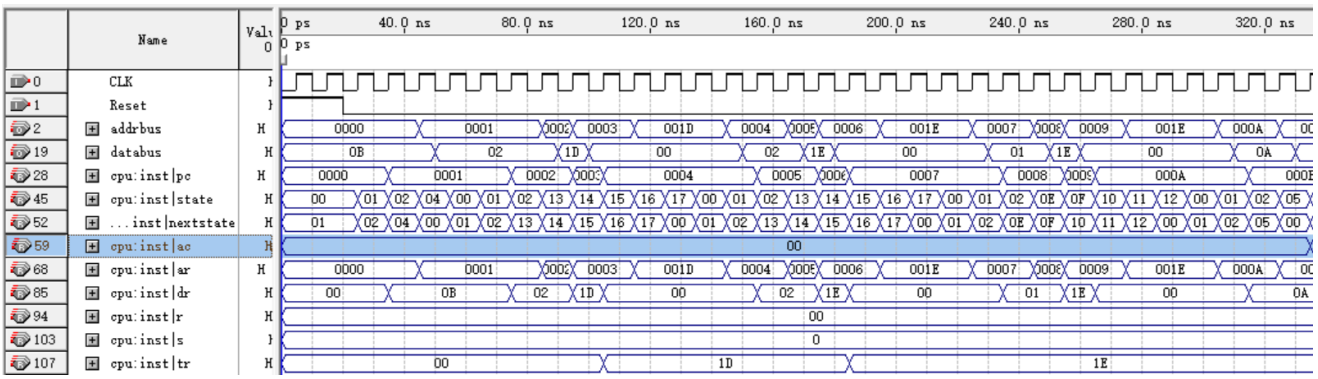
3：total＝total＋i

4：IF i≠n THEN GOTO 2

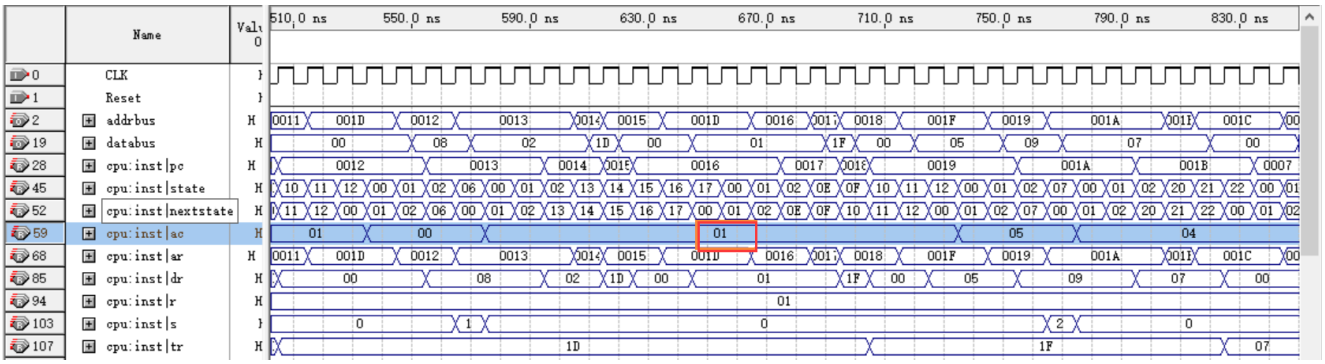
测试记录

相对简单 CPU 运行测试程序的 PC 寄存器、IR 寄存器、AC 累加器等信号波形截图如下：

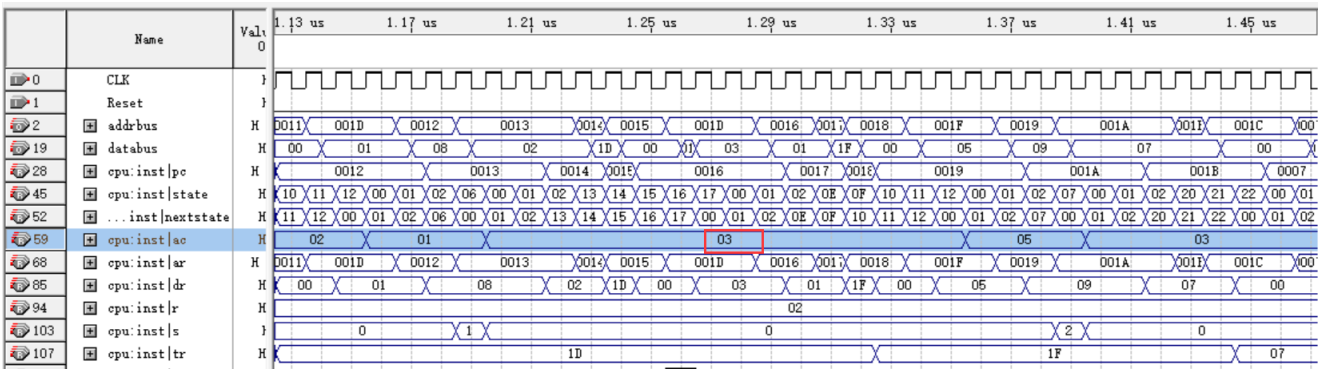
初始状态时各个寄存器的状态如下图所示：



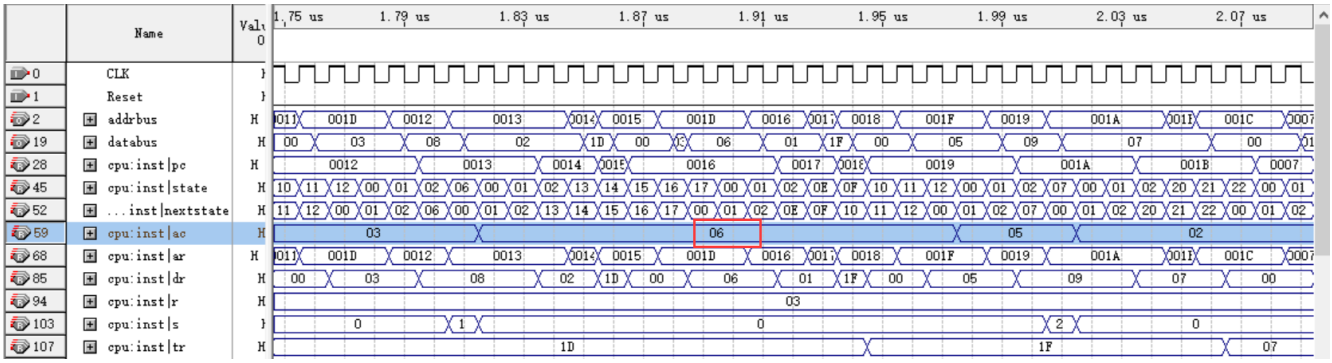
当 total 的值为 1 时各个寄存器的值如下：



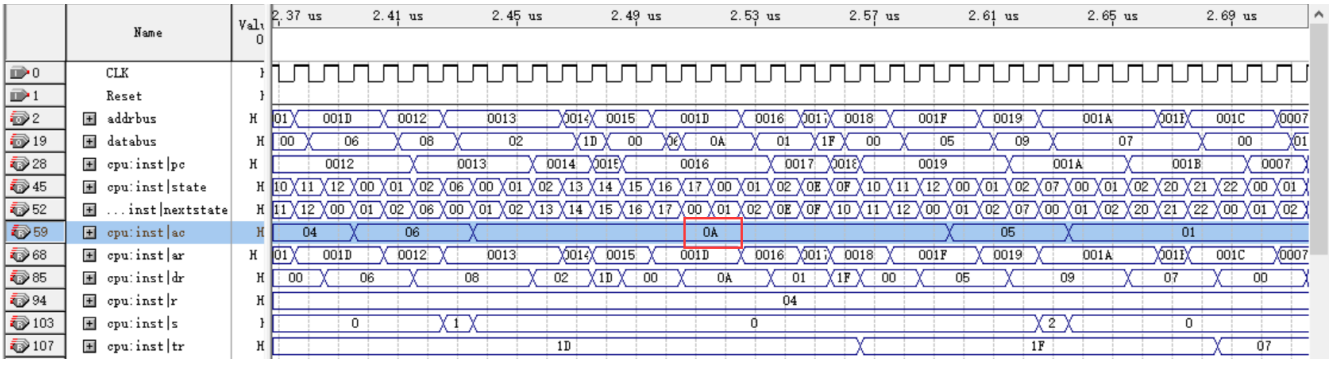
当 total 的值为 3 时各个寄存器的值如下，其中 ac 值现在为 03，前一个 ac 值为 01，是上一次累加后的结果：



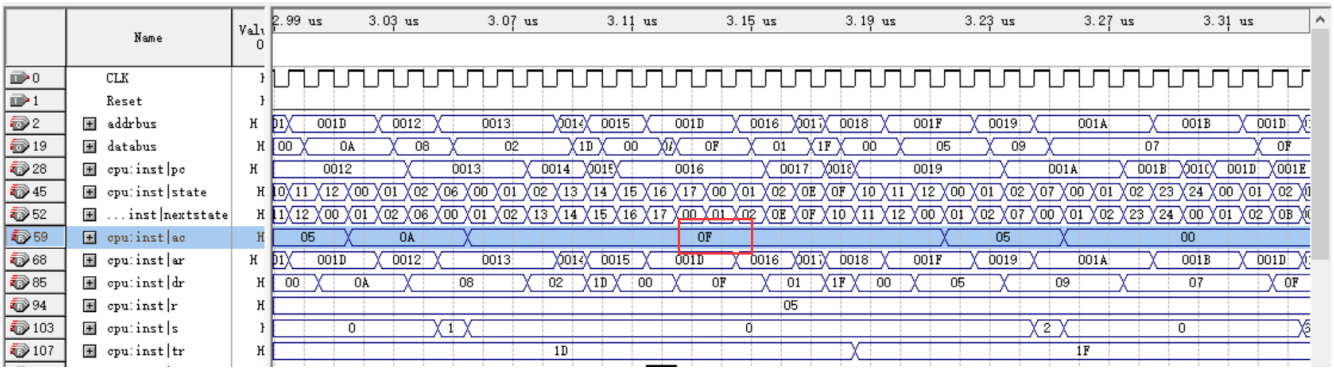
当 total 的值为 6 时各个寄存器的值如下：



当 total 的值为 10 时各个寄存器的值如下：



当 total 的值为 15 时各个寄存器的值如下：



分析和结论

从仿真结果来看，CPU 实现了对测试程序指令的读取，译码和执行，得到的运算结果正确。根据结果可以认为所设计的相对简单 CPU 实现了所要求的功能，完成了实验目标。

通过这次 CPU 设计，让我重新温故了 VHDL 语言的写法以及 quartus 软件的应用，因为大二时做过一个 CPU，所以基本的思路还是有的，通过这门课的学习，让我对 CPU 的设计有了更深的认识。