项目 A 任务书

简单计算机系统的设计与实现

计算机工作原理部分实验

一、项目介绍

二、计算机工作原理

三、简单计算机系统的指令集及编码

四、简单计算机系统的硬件实现

五、简单计算机各基本模块简介

六、程序编写及举例

七、项目要求

八、项目的设计和调试

实验 1 简单计算机系统的基本构成和指令集理解

实验 2 简单计算机系统的数据通路和控制器设计

实验 3 简单计算机系统的设计和调试

一、项目简介

项目目的：

本项目是“微机原理与应用”课程中关于计算机工作原理部分的实验项目，涉及第 1～5 周讲课的内容，目的是通过设计一个简单计算机系统，深入了解计算机系统各部分的基本组成和工作原理，为后续计算机接口部分的学习打下良好基础。该项目可在暑期小学期“电子技术课程设计”的“数字计算器的设计与实现”项目上进行扩展。前提是学生应具备运用 EDA 进行数字电路系统设计与调试的能力。

项目平台：

与小学期“电子技术课程设计”的“数字计算器的设计与实验”相同：PC 机一台、 FPGA 实验板一块（EP2C5Q208C8N+矩阵键盘+数码管），如图 1 所示。

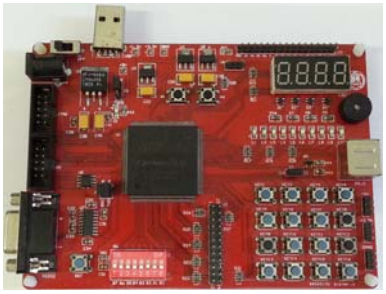

图 1 FPGA 实验板

项目任务：

1、 根据一种精简的 MIPS 指令集，利用 EDA 设计工具，在 FPGA 实验板上完成一个简单计算机系统的设计；

2、 编写程序，实现计算器功能，先用汇编格式指令编写，最后再翻译成机器码。

具体操作步骤描述如下：

从键盘输入算术式，由所设计的计算机系统完成相应运算，并将运算结果显示在数码管和发光二极管上。输入的运算符可以是加、减、与、或、比较等，输入的数字大小在 0~999 之间。

演示举例：完成 48 减 836 的运算

具体是：从键盘输入 5、7 两个字符键，数码管上显示 48； 接着输入减号运算符“-”，此时点亮某个发光二极管；然后输入 8、3、5 三个字符键，数码管上显示 835， 最后按下等号键“=”，运算结果“-787”中的 787 显示在数码管上，负号“-”用发光二极管 L1 显示。

要求从键盘输入的运算符加、减、与、或、比较等运算符分别用实验板上的发光二极管 L6~L10 显示；按下的等号“=”不显示，但发出一声蜂鸣声，提示运算结束。

项目重点：通过对一种简单计算机系统软、硬件的设计和调试，理解计算机系统的组成和工作原理，掌握基于 FPGA 的复杂数字系统的逻辑设计和调试。

二、计算机工作原理

1、通用计算机模型

如果任务是完成一种运算，实际实现中不一定采用计算机原理，用普通的数字电路也可以。例如，下面这个运算：

out = in1 + in2 * in3 + in4 * in5 * in6;

通过使用加法器和乘法器可以搭建如下实现图 2 所示电路：

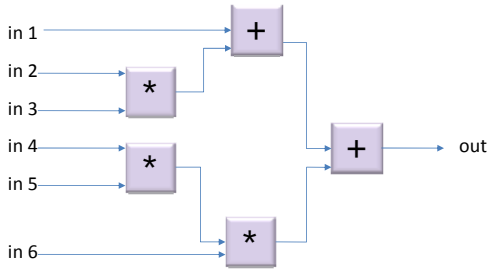

图 2 数字电路实现运算电路 1

电路中总共用了 3 个乘法器和 2 个加法器。如果又要做另外一个运算：

out = in1 + in2 + in3 + in4 * in5 + in6;

上面搭建的电路就没有用了，必须要重新搭建另外一个电路。如图 3 所示。

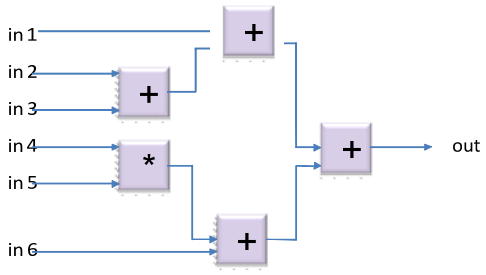

图 3 数字电路实现电路 2

用这种硬件电路方式来实现运算，缺点是每一种新的运算，都要搭建一种新的电路。有没有一种通用的计算设备，一套硬件就能实现所有的功能呢？

有，计算机的产生就是为了解决这个问题。

一个复杂的运算，都是由一些简单的运算组合而成的，一个最简单的运算可以用图 4 所示的模型表示：

图 4 简化运算模型

例如，c=a+b，输入数据是 a 和 b，输出数据是 c，运算符是加法。

更多的运算就需要更多的输入数据和输出数据。首先要有一个存储器将数据存储起来；然后将一些常用的基本运算，如加、减、乘、与、或和非等组合在一起，构成一个部件，叫算术逻辑单元(ALU: Arithmetic Logic Unit)；其次，还需要一个控制器，将存储器中的数据送到 ALU 中去做运算，并将结果存回到存储器中。每一个简单的运算或操作（如到存储器存取数据）都对应一条指令。数据放置到何处，做什么样的运算，都是由指令来告诉控制器的。多条指令构成的指令序列就组成了完成特定功能的程序。

一种通用计算模型如图 5 所示。

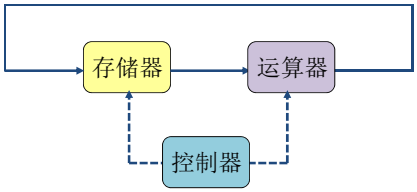

图 5 通用计算模型

任何一台计算机的硬件都要完成相同的基本功能：输入数据、输出数据、处理数据和存储数据。计算机通常由五个典型部件组成：输入设备、输出设备、存储器、运算器和控制器，其中最后两个部件通常合称为处理器 CPU（Center Process Unit）。一台完整的计算机逻辑结构(冯·诺依曼结构)如图 6 所示。

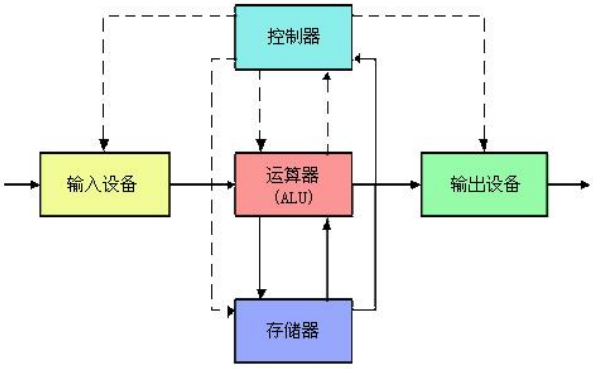

图 6 冯·诺依曼结构

1945 年 6 月，冯·诺依曼在“关于 EDVAC 的报告草案”中，描述了计算机的逻辑结构，提出了“存储程序”的思想。“存储程序(stored-program)”的意思就是将程序存储到计算机内部，计算机自动执行。现在的计算机都是冯·诺依曼机。

与专用数字电路不同，除非给这台硬件配上具体的程序(软件) ，否则它什么也做不了。

程序由指令序列组成，决定了计算机要完成的功能。

2、计算机程序

要控制计算机(硬件)，就需要用计算机能理解的语言发出命令。

数字计算机其实就是一个复杂的数字电路系统，它能理解的最简单的信号是“开”和“关”，即“1”和“0”。目前计算机采用的数制通常为二进制，我们只能采用由“1”和“0”构成的指令去控制计算机硬件。指令的集合就构成了计算机程序。计算机程序是由计算机语言实现的。计算机所能理解的语言(机器语言)就是由“1”和“0”构成的数字组合。

我们常用的 C 或 Java 等计算机语言属于高级语言，由高级语言编写的大型应用程序，可能由数百万行代码构成，并依赖实现复杂功能的软件库。然而计算机硬件只能执行极其简单的低级指令。从复杂的应用程序到简单的指令需要经过几个软件层次，才能将复杂的高层次操作逐步解释或翻译成简单的计算机指令。

图 7 显示了这些软件的层次结构，

外层是应用软件，内层是硬件，

系统软件位于二者之间。

操作系统和编译器是两种重要的系统软件。

图 7 计算机硬件、软件层次示意图

操作系统是用户程序和硬件之间的接口，为用户提供各种服务和监控功能，其最重要的作用是：

- 处理基本的输入和输出操作
- 分配存储器(Storage 和 memory)
- 为多个应用程序提供共享计算机资源的服务

常用的操作系统有：Linux，MacOS 和 Windows。

前面提到，计算机的指令由计算机能够理解并执行的数字“1”和“0”构成的，也可以说指令是由数字表示的。例如，如下二进制数字

0000 1100 0001 1000 0001 1000 0010 0001

告诉计算机将两个数相加。

最初，程序员用二进制数字对计算机进行编程，这是一项非常乏味的工作。所以很快又发明了符合人类思维方式的助记符。然后用手工的方式把助记符翻译成二进制，其过程显然还是很繁琐。随后设计人员开发了一种称为汇编程序的软件，可以利用计算机将助记符形式的指令自动翻译成对应二进制表示的指令，即对应的机器码。

例如，指令(助记符)

add A, B

汇编程序将该符号翻译成

0000 1100 0001 1000 0001 1000 0010 0001

这条指令告诉计算机做两个数的减法运算。这种由助记符号表示的语言称为汇编语言。机器所能理解的二进制语言则称为机器语言。

比起机器语言，汇编语言已经有了很大的进步。但是汇编语言要求程序员写出计算机执行的每条指令，要求程序员像计算机一样思考。在描述相对复杂的算法时，汇编语言还是不方便。

相对于汇编语言，用 C 或 Java 等高级语言实现一些复杂算法要方便得多。但用这些高级语言编制好完成特定功能的程序后，还需要**编译器**完成另外一项重要功能：把用 C、Java 等高级语言编写的程序翻译成计算机硬件能执行的指令。考虑到高级语言的复杂性和硬件执行指令的简单性，这个翻译过程是相当复杂的。当前，编译器可以将功能强大的高级语言翻译成机器语言，这大大提高了软件的生产率。下页图 8 表示了这些程序和编程语言之间的关系。

三、简单计算机系统的指令集及编码

1、什么是指令集

要计算机服从指挥，就必须用计算机能理解的语言。这种语言中的基本单词称为指令，一台计算机的全部指令称为该计算机的指令集。图 9 示意性地表示了指令集在计算机中的重要性。

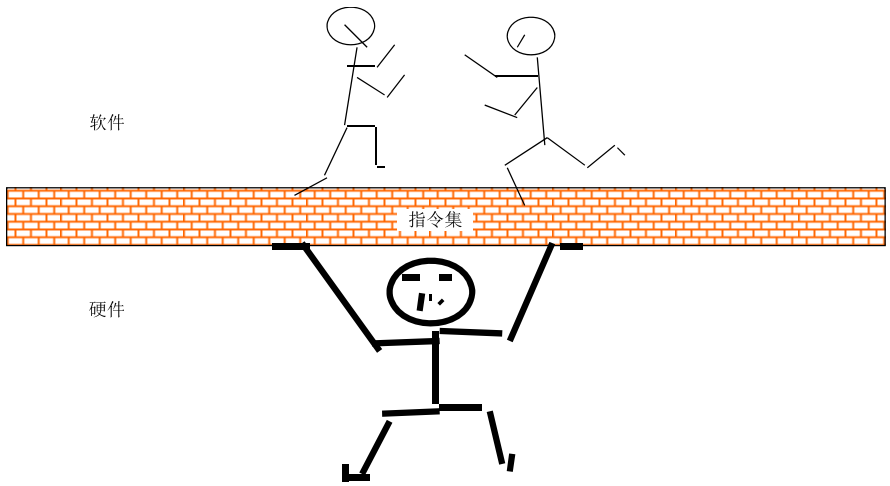

图 9 指令集与计算机软、硬件的关系示意图

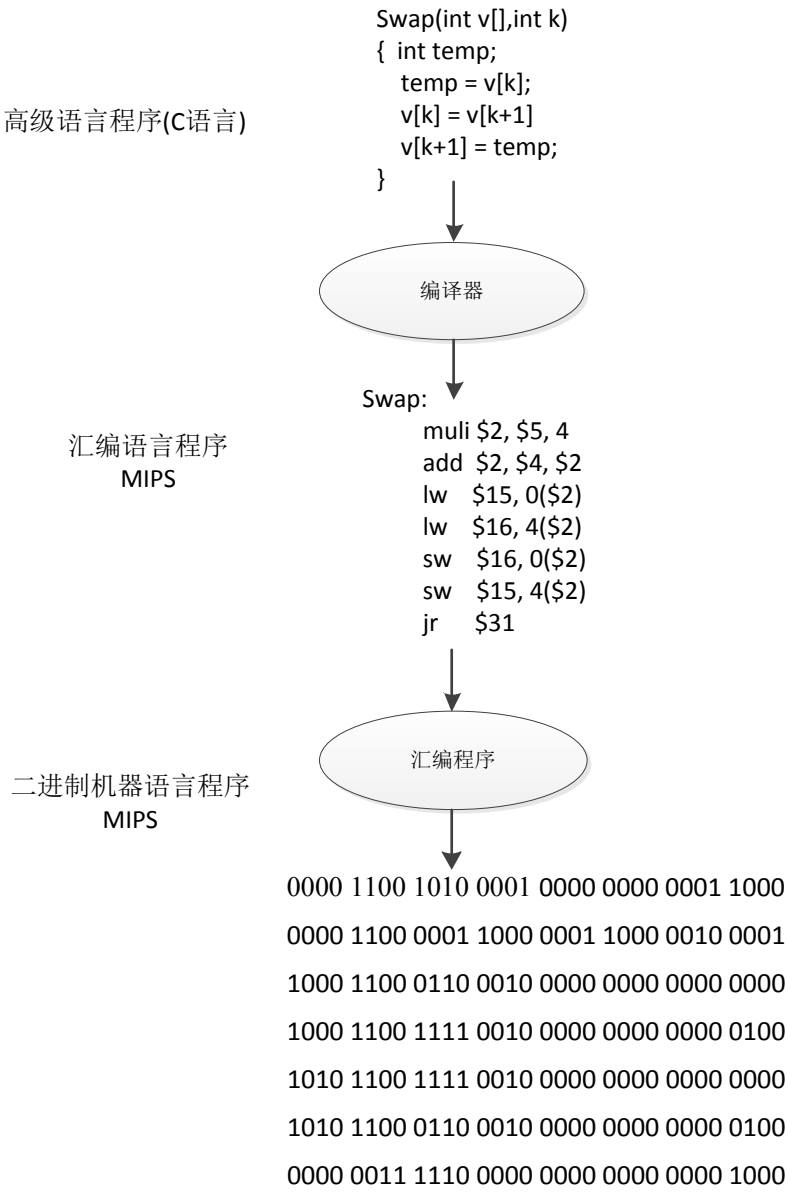

图 8 由 C 程序经编译和汇编得到二进制机器语言的过程

指令集是什么？

例如，要完成 c=a + b 运算。称 a 和 b 为操作数，+为操作码。完成运算的单元是 ALU，操作码所代表的操作是 ALU 中的一个运算。操作数存储在存储器中(称为数据存储器，常用随机存储器 RAM，易失，掉电后数据丢失)。由于从存储器中访问数据慢，因此在离 ALU 很近的地方放置了一些寄存器(Register)，这样中间计算结果就可以存储在寄存器中，不用每次都经过存储器。此外，完成一项运算一般需要多条指令，这些指令序列就组成了完成这个功能的程序。根据存储程序的概念，这段程序也是存储在存储器中的。存储程序的存储器一般称为程序存储器(常用只读存储器 ROM，非易失性，掉电后数据可以保存)。完成各种操作指令的列表就构成了指令集。

计算机的运算模型见图 10。

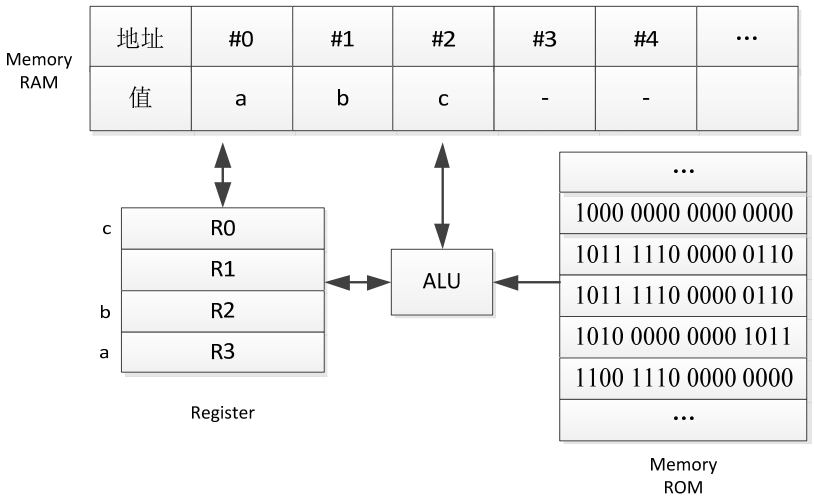

图 10 计算机运算模型

c=a + b 加法运算，可以分解为下面几个步骤完成：

```
ANDI R1, R1, 0      ; 将寄存器 R1 清 0, R1 = R1 AND 0
LW R3, R1, 0        ; 从 Memory[R1+0]（即 RAM 中的地址 0）处取 A，放在 R3 中
LW R2, R1, 1        ; 从 Memory[R1+1]（即 RAM 中的地址 1）处取 B，放在 R2 中
ADD R0, R3, R2      ; 把 R3 和 R2 相加，结构存放到 R0 中
SW R0, R1, 2        ; 把 R0 中的值存放在 Memory[R1+2] （即 RAM 中的地址 2）中
```

在上面汇编语言程序中，每行语句就是一条指令，ANDI、LW、ADD、SW 为操作码，后面紧跟操作数，”;”后面是注释。

从编程者的角度来看，指令集包括一套指令列表和一些寄存器，编程者在此基础上就可以编写程序。

由于计算机只能理解“1”和“0”表示的二进制数字，不认识 LW、ADD 等操作码，所以，这些操作码要被汇编为计算机能认识的格式，即用“1”和“0”表示的二进制格式的机器码。

2、简单计算机系统的指令集及其编码

指令的构成主要包括两部分：操作码和操作数。操作码给出要执行什么样的操作，不同的操作具有不同的操作码，可以根据操作码判断指令的操作类型；操作数给出参与操作的数（称立即数）或者数的来源（来自寄存器，还是 RAM，给出相应的寄存器号或 RAM 单元地址）。执行的操作除上面提到的加、减、与、或、比较等算数逻辑运算外，还有存储单元的读操作、写操作，以及修改指令计数器 PC 的操作，对所有这些操作都要进行编码。

表 A-1 给出的是要求设计的简单计算机系统的指令集，有 15 条指令，每条指令用 4 位二进制位即可区分。表 A-1 中包含了 15 条指令的名称及其对应的操作码和汇编语言格式，第 2 列的 4 位二进制数就表示对相应指令的编码。表 A-1 中，Rd、Rs 和 Rt 都是寄存器，imm 为立即数。如 AND　Rd, Rs, Rt，表示将寄存器 Rs 和 Rt 中的值进行“与”运算，结果送到 Rd 寄存器中。PC 表示程序指针（PC 是一个寄存器，指向将要被执行的指令）。

用同样的办法，可以**对寄存器进行编码**。假如**该计算机有 4 个 8 bit 的寄存器，分别称为 R0、R1、R2、R3， 对应的二进制编号分别为 00、01、10、11。**

这时候就可以定义各条指令在计算机中的格式了。

从表 A-1 可以看出，除了最后一条无条件跳转指令外，其他 14 条指令都有 3 个操作数。这 14 条指令中，有 7 条指令操作数均来自寄存器，另外 7 条操作数中有立即数。我们把指令中操作数均来自寄存器的 7 条指令称为寄存器类指令(R 指令)；操作数中含有立即数的 7 条指令称为立即数指令(I 指令)；无条件跳转指令 JMP 称为 J 指令。

每条指令的编码由 16 位二进制数表示，最高 4 位是操作码。每条指令完成相应操作后，程序指针 PC 也将发生变化，指向下一条指令地址。

表 A-1 简单计算机指令集

操作名称	操作码	汇编语言格式指令	执行操作
与	0000	AND Rd, Rs, Rt	Rd←Rs and Rt; PC ← PC + 1
或	0001	OR Rd, Rs, Rt	Rd←Rs or Rt; PC ← PC +1
不带进位加	0010	ADD Rd, Rs, Rt	Rd←Rs+ Rt; PC ← PC + 1
不带借位减	0011	SUB Rd, Rs, Rt	Rd←Rs– Rt; PC ← PC + 1
无符号数比较	0100	SLT Rd,Rs,Rt	If Rs<Rt, Rd=1 else Rd=0; PC ← PC + 1
带借位减	0101	SUBC Rd, Rs, Rt	Rd←Rs– Rt-(1-C); PC ← PC + 1
带进位加	0110	ADDC Rd, Rs, Rt	Rd←Rs+ Rt+C; PC ← PC + 1
无条件跳转	0111	JMP imm	PC ← imm
立即数与	1000	ANDI Rt, Rs, imm	Rt←Rs and imm; PC ← PC +1
立即数或	1001	ORI Rt, Rs, imm	Rt←Rs or imm; PC ←PC +1
立即数加	1010	ADDI Rt, Rs, imm	Rt←Rs+ imm; PC ← PC +1
读存储器	1011	LW Rt, Rs, imm	Rt←MEM[Rs+imm]; PC ←PC +1
写存储器	1100	SW Rt, Rs, imm	MEM[Rs+imm] ←Rt；PC ← PC +1
相等时跳转	1101	BEQ Rs, Rt, imm	If Rt=Rs, PC←PC+imm+1 else PC←PC+1
不等时跳转	1110	BNE Rs, Rt, imm	If Rt!=Rs, PC←PC+imm+1 else PC←PC+1

其中的 C 是标志寄存器中存放的进位/借位值。PC ← PC +1 表示该条指令执行后，PC 的值变为 PC+1（指向下一条指令）。

下面是对简单计算机系统 15 条指令的具体介绍。

1）R 型指令

该类型指令中有 3 个操作数，均来自寄存器，指令中给出各寄存器号。包括 AND、 OR、 ADD、SUB、 ADDC、SUBC、SLT，共 7 条指令。R 型指令的 16 位二进制编码结构见表 A-2，R 型指令表见表 A-3。

表 A-2 R 型指令二进制编码结构

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op				Rs		Rt		Rd		---					

指令中用到的符号说明如下：

Op：指令操作码；

Rs：第 1 个源操作数的寄存器号；

Rt：第 2 个源操作数的寄存器号；

Rd：目的操作数的寄存器号；

---：表示任意值，编码时通常取 0

表 A-3　R 型指令表

操作名称	操作码	汇编语言格式指令	执行操作
与	0000	AND　Rd, Rs, Rt	Rd←Rs and Rt;　PC ← PC + 1
或	0001	OR　Rd, Rs, Rt	Rd←Rs or Rt;　PC ← PC +1
加	0010	ADD　Rd, Rs, Rt	Rd←Rs+ Rt;　PC ← PC + 1
减	0011	SUB　Rd, Rs, Rt	Rd←Rs– Rt;　PC ← PC + 1
带进位加	0110	ADDC　Rd, Rs, Rt	Rd←Rs+ Rt+C;　PC ← PC + 1
带借位减	0101	SUBC　Rd, Rs, Rt	Rd←Rs– Rt-(1-C);　PC ← PC + 1
无符号数比较	0100	SLT　Rd,Rs, Rt	If　Rs<Rt, Rd=1 else Rd=0; PC ← PC + 1

例1.　将寄存器 R1 的内容“或”R2 的内容，结果存放到 R3 中，指令编码是什么？

根据题意，得到指令的汇编语言格式为 OR　R3, R1, R2

根据寄存器编码，R0～R3 对应的二进制编号分别为 00、01、10、11，将“或”操作的操作码 0001 、3 个操作数的编号 Rs=01，Rt=10，Rd=11 分别按表 A-2 顺序填入 16 位二进制中，见表 A-4，即可得到例 1 中指令的编码是 0001 0110 1100 0000B，或以十六进制形式表示为 0x16C0。

表 A-4　例 1 指令编码

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op				**Rs**		**Rt**		**Rd**		-----					
0	0	0	1	0	1	1	0	1	1	0	0	0	0	0	0

例2.　分析编码为 0x3180 的指令执行何种的操作？

将 0x3180 按二进制展开，如表 A-5，根据表 A-2 各位的含义，利用表 A-3 指令格式，可以得到汇编语言格式指令为：SUB　R2, R0, R1

功能是将 R0 减去 R1，结果存放在 R2 中。

表 A-5　例 2 指令分析

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
减的操作码				Rs: R0		Rt: R1		Rd: R2		-----					

2）I 型指令

指操作数中含有一个 8 位二进制数(称为立即数 immediate)的指令，共包括 ANDI、ORI、ADDI、LW、SW、BEQ、BNE 等 7 条指令，I 型指令 16 位二进制编码结构见表 A-6，I 型指令表见表 A-7。

表 A-6　I 型指令二进制编码(机器码)结构

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op				Rs		Rt		Imm							

指令中用到的符号说明如下：

op：指令操作码；

Rs：第 1 个操作数的寄存器号；

Rt：第 2 个操作数的寄存器号；

imm：立即数；

表 A-7　I 型指令表

操作名称	操作码	汇编语言格式指令	执行操作
立即数与	1000	ANDI　Rt, Rs, imm	Rt←Rs and imm;　　PC ← PC +1
立即数或	1001	ORI　　Rt, Rs, imm	Rt←Rs or imm;　　PC ←PC +1
立即数加	1010	ADDI　Rt, Rs, imm	Rt←Rs+ imm;　　PC ← PC +1
读存储器	1011	LW　　Rt, Rs, imm	Rt←MEM[Rs+imm]; PC ←- PC +1
写存储器	1100	SW　　Rt, Rs, imm	MEM[Rs+imm] ←Rt;　PC ← PC +1
相等时跳转	1101	BEQ　Rs, Rt, imm	If Rt=Rs, PC←PC+imm+1 else PC←PC+1
不等时跳转	1110	BNE　Rs, Rt, imm	If Rt!=Rs, PC←PC+imm+1 else PC←PC+1

MEM[RS+imm]表示一个存储单元的内容，存储单元的地址是寄存器 Rs 的内容与立即数 imm 相加的值。

例 3　当 ANDI 中立即数为 0 时，可以实现对寄存器清零。

如对 R1 清零的指令为 ANDI　R1, R1, 0　指令的编码为 0x8500

例 4　当 ADDI 中寄存器值 Rs 为 0 时，可以将立即数赋给寄存器 Rt。

```
如 ANDI   R0, R0,0        ;R0=0
   ADDI   R0, R0, 7D      ;将 R0+0x7D 的结果赋给 R0，  结果 R0=0x7D
```

例 5　指令 LW 和 SW 分别完成对 RAM 存储单元的读操作、写操作

```
如 ANDI   R3, R3,0        ;R3=0
   LW     R2,R3,7         ;将地址是(R3+7)的存储单元的内容赋给 R2
                          ;此时，R3=0，  所以 R3+7=7，  所以是存储单元 07
   SW     R1, R3, 8       ;将 R1 的内容赋给地址是(R3+8)的存储单元
                          ;此时，R3=0，  所以 R3+7=7，  所以是存储单元 08
```

例 6　由于程序存储器的低四位地址（0x00～0x03）被映射为双向 IO，

因此可以采用 LW 和 SW 指令实现对 IO 口的读、写操作。

```
如 ANDI   R3, R3,0        ;R3=0
   LW     R2,R3,1         ;将地址是(R3+1)的存储单元的内容赋给 R2
                          ;此时，R3=0，R3+1=1，将 I/O 寄存器 1 内容赋给 R2
   SW     R1, R3, 0       ;将 R1 的内容赋给地址是(R3+0)的存储单元，
                          ;此时，R3=0，R3+0=0，将 R1 内容赋给 I/O 寄存器 0
```

例 7　指令 BEQ 和 BNE 是条件转移指令，当指令中的条件满足时，用立即数 imm 修改指令指针 PC 的值；条件不满足时，PC 值加 1.

```
   AND    R0,R0,0         ;使 R0=0
   SLT    R3, R2,R1       ;比较 R2 和 R1 的大小，R2<R1，  R3=1, 否则 R3=0
   BEQ    R3,R0, 0x40     ;R3=0，即 R2 >=R1，PC=PC+0x40+1=PC+0x41; 否则 PC=PC+1
```

上面指令段对 R2 和 R1 的内容进行比较，根据比较的结果修改 PC，从而使 PC 指向不同的 ROM 存储单元地址，取不同的指令执行，这是一种分支结构的程序。

3） J 型指令

只有 1 条，为无条件跳转指令 JMP。

J 型指令二进制编码结构如表 A-8 所示，J 型指令表见表 A-9 所示。

表 A-8　J 型指令二进制编码(机器码)结构

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Op				----				Imm							

表 A-9　J 型指令表

操作名称	操作码	汇编语言格式指令	执行操作
无条件跳转	0111	JMP　imm	PC ←　imm

例 8　指令 JMP 的功能是直接修改指令计数器 PC 的值，在需要产生分支时使用。

四、简单计算机系统的硬件实现

1、CPU 的微架构

如图 9 所示，指令集在计算机软件、硬件之间起到非常重要的作用。从编程者的角度来看，它包括一套指令列表和一些寄存器，编程者在此基础上就可以编写程序。目前，在 PC 领域，有多种指令集，如 Intel 和 AMD 的 CPU 都是基于 x86 指令集，手机上的程序绝大部分是基于 ARM 指令集。由于指令集不同，手机上的程序无法在 PC 上用。

在计算机组成结构中，指令集层次和 CPU 的物理实现层次之间还有一个 CPU 的微架构(Microarchitecture)层。同样是采用 x86 指令集，Intel 和 AMD 的 CPU 各自使用不同的微架构。计算机系统的微架构的设计与指令集密切相关。**设计过程中，必须熟悉指令集中每条指令的数据通路。**下面简要介绍上面给出的简单计算机系统的 15 条指令的数据通路。

2、简单计算机系统的数据通路

1） R 型指令数据通路

R 型指令包括与（AND）、或（OR）、不带进位加（ADD）、不带借位减（SUB）、带进位加(ADDC)、带借位减(SUBC)和比较（SLT）等，指令执行时的数据通路见图 11。

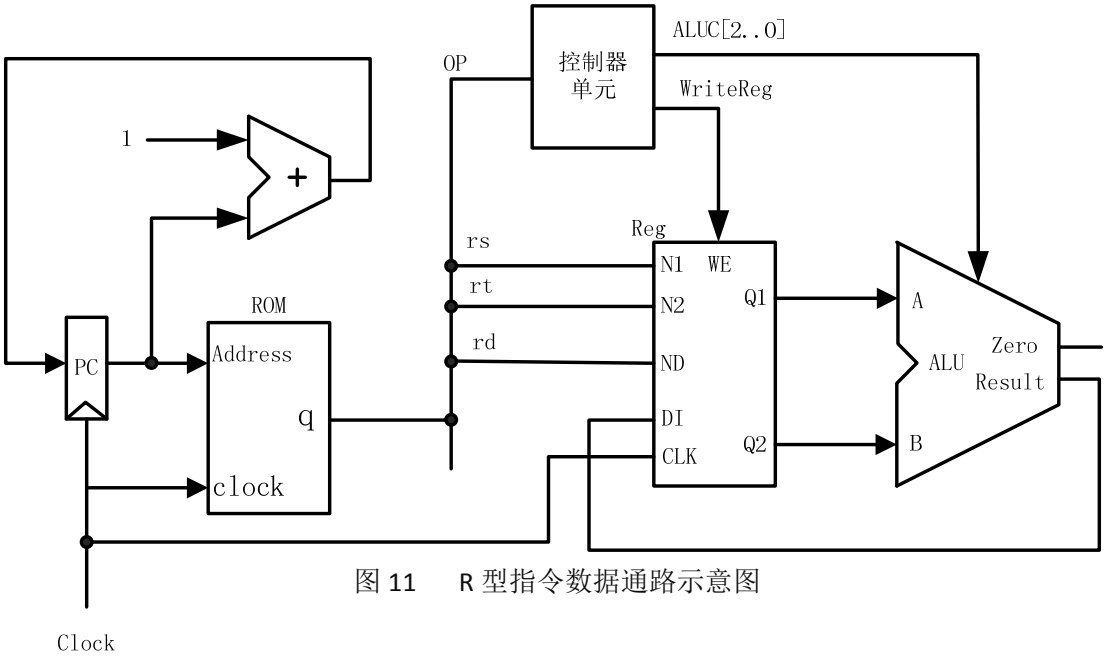

图 11　R 型指令数据通路示意图

图 11 中，PC 为程序指针生成单元，ROM 为程序存储器，Reg 为寄存器组。根据 PC 程序指针内容，从 ROM 中取出相应指令，将指令中相应位分别送至控制单元和寄存器组的相应通道。其中，指令中的高 4 位即操作码 OP 送给控制单元，决定执行何种操作，并分别产生对寄存器组和 ALU 的控制信号 WriteReg 和 ALUOP[2..0]；指令中第 1 和第 2 个操作数 rs，rt 分别送到寄存器组的 N1、N2 端，寄存器组将相应寄存器的内容通过 Q1、Q2 端送给 ALU；ALU 输出的运算结果送回寄存器组的数据输入端 DI，并写入由 rd 所指定的寄存器单元中。

简而言之，R 型指令将 rs、rt 指定的寄存器内容送入 ALU，并根据 OP 控制运算类型，随后将运算结果送至寄存器组数据输入端，写入 rd 寄存器。

2） I 型指令数据通路

I 型指令根据其执行流程不同，又可分为 3 类。第 1 类是寄存器操作指令，包括 ADDI，ANDI，ORI。第 2 类为存储器读写指令，包括 SW，LW。第 3 类为条件跳转指令，包括 BNE，BEQ。

寄存器操作指令

如图 12 所示，I 型指令中 ANDI，ORI，ADDI 将 rs 指定的寄存器内容和 imm 送入 ALU，根据 OP 内容进行相应运算，并将结果送至寄存器组的数据输入端 DI，写入 rt 指定的寄存器中，过程与 R 型指令相似。

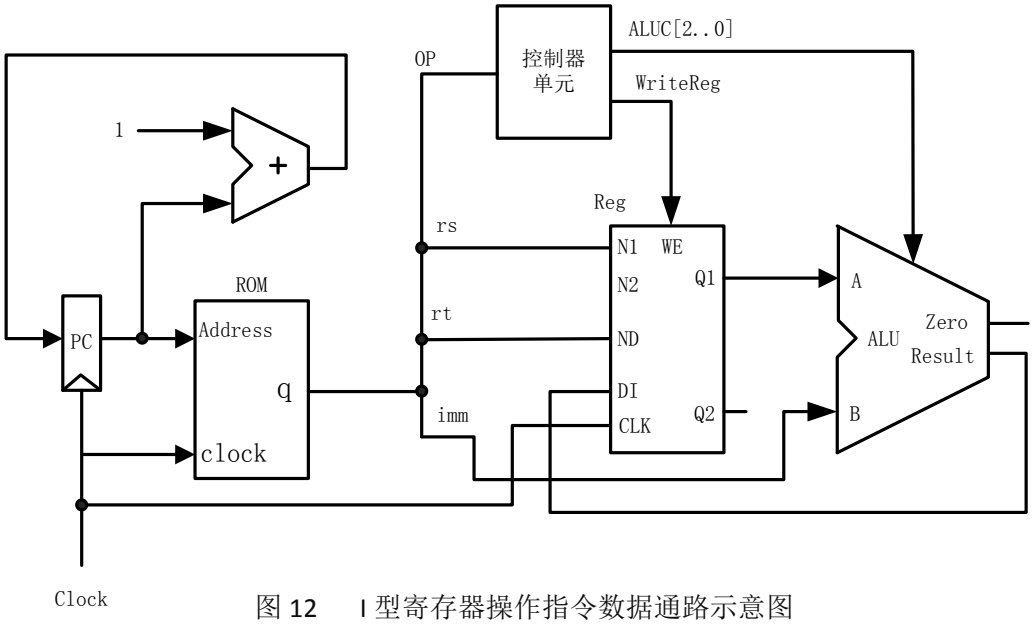

图 12　I 型寄存器操作指令数据通路示意图

存储器操作指令

如图 13 所示，存储器读指令 LW 中的 4 位 OP 送给控制单元，分别产生控制信号 MemTOReg，将 rs 指定的寄存器和 imm 送入 ALU 进行加法运算，将结果作为地址送至 RAM 的 address 端，取得相应 RAM 单元内容后，由 MemROReg 控制数据选择器，送至寄存器组数据输入端 DI，写至 rt 指定的寄存器。

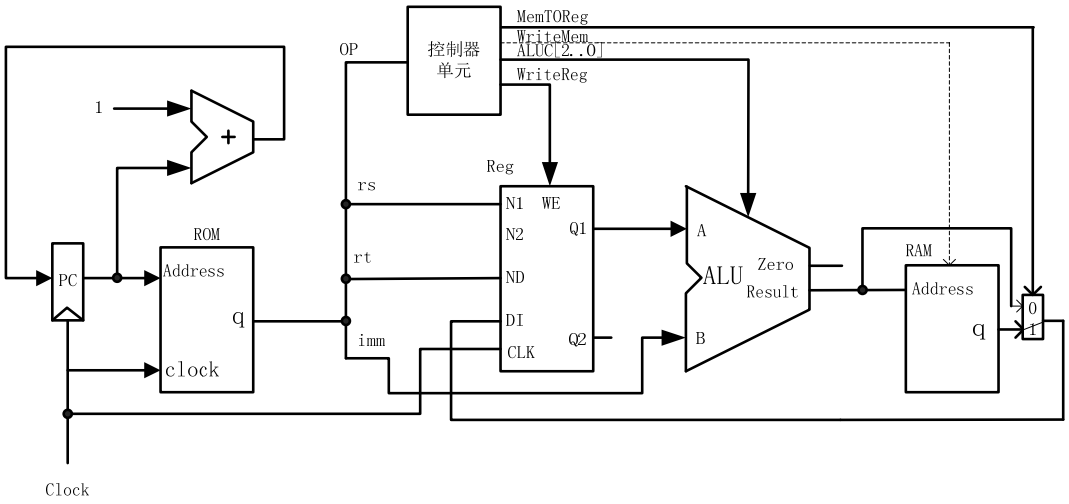

图 13　指令 LW 数据通路示意图

如图 14 所示，将存储器写指令 SW 中的 OP 送给控制单元，产生控制信号 WRITEMEM 和 ALUOP，将 rs 指定的寄存器内容和 imm 送至 ALU，进行加法后得到 RAM 地址，随后将 rt 指定寄存器的内容(从 Reg 的 Q2 端口输出)写入相应 RAM 存储单元中。

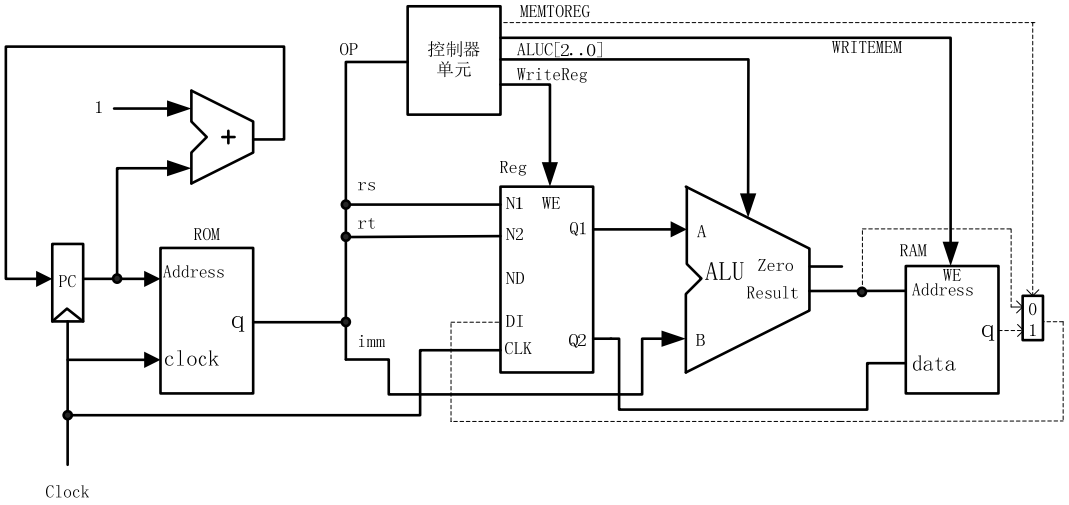

图 14　存储器写指令 SW 数据通路示意图

条件跳转指令

如图 15 所示，BEQ/BNE 指令首先将 rs、rt 指定寄存器的内容送至 ALU 进行运算，控制器模块根据 OP 和 Zero 零标志位送出 Branch 控制信号，控制数据选择器，选择 imm 或者 PC+1 至 PC 寄存器中。

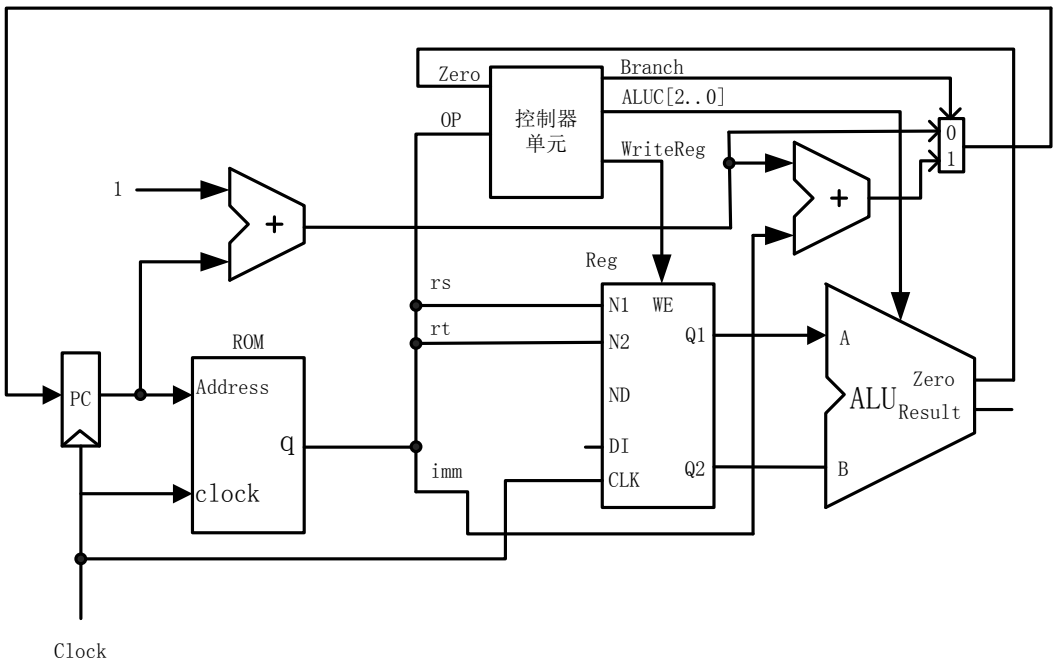

图 15　I 型条件跳转指令数据通路

3)　J 型指令数据通路

J 型指令只有一条 JMP，OP 经控制器送出控制信号 Jump，将指令中的立即数 imm 直接送至 PC 程序指针模块的输入端，在下一个时钟信号到来写入 PC，修改了 PC 的内容。如图 16 所示。

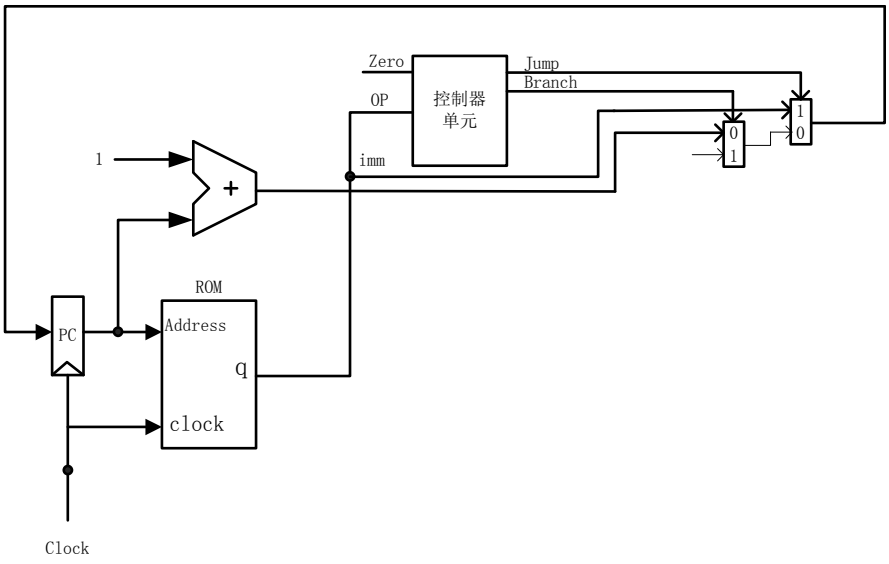

图 16　J 型指令数据通路示意图

3、EDA 设计与简单计算机系统的硬件实现

计算机辅助设计(Computer-aided Design, CAD)在各行业都有广泛应用。计算机本身设计，无论 CPU、存储器电路，还是结构等，也都是在计算机上进行的。

电子设计自动化(Electronic Design Automation, EDA)是电路领域的计算机辅助设计。设计者在 EDA 软件平台上完成基础电路的全套设计，直至将设计交给工厂流片。

硬件描述语言(VHDL, Verilog)主要用于数字电路的建模，建模的层次由高到低分为 4 层：行为级(描述电路的功能，不关心电路的具体结构，只关注算法，主要用于建模和仿真)、RTL 级(也称寄存器传输级，Register Transfer Level)、门级和电路级。

集成电路的设计，主要采用 RTL 级的建模，然后经过逻辑综合(Synthesis)，生成具体的电路。

HDL 代码综合出来的文件称为门级网表(netlist)，它描述了电路的门级结构，门级电路以与门、非门等基础。

集成电路设计中门级网表产生后，即可进行物理设计，输出版图(layout)，下一步就是芯片制造了，这个过程俗称流片(tapeout)。

实验任务主要是采用 Altera 的 EDA 设计工具 Quartus Ⅱ，少数同学可采用 Xilinx 的设计工具 Vivado，根据前面介绍的指令集及其编码，完成一种简单计算机系统(微架构)的设计。

Quartus Ⅱ和 Vivodo 都是一个综合的 EDA 平台，集成有逻辑综合的功能。通过编写 HDL 代码或原理图，综合出的电路系统在 FPGA 芯片上运行。本项目实现的简单计算机系统只需要在 FPGA 板上进行验证。

五、简单计算机系统各基本模块简介

如图 11~16 所示的简单计算机系统的数据通路中，包含的主要模块有寄存器组、ALU、控制单元、ROM、RAM 等，以及一些多路选择开关。其中寄存器组是属于 ALU 的，ROM 和 RAM 同属于存储器，所以图 11~16 中包含了组成计算机的 5 个典型模块中的 3 个：控制器、ALU 和存储器。本实验任务所要求设计的简单计算机系统还需包含输入和输出 2 个模块，对应到图 1 所示 FPGA 电路板上则为键盘模块和数码管显示模块。计算机系统从输入设备（如键盘）接收参加运算的操作数及加、减等运算符，并通过输入设备接口将这些操作数和运算符存放在寄存器或存储系统中，然后经过 ALU 进行运算，并将运算的结果通过输出设备（如数码管）显示出来。上述所有操作，包括操作数等的读入，ALU 运算，输出结果等都是通过执行存放在存储系统中的程序实现的。

从前面指令集的介绍可以看出，ALU 进行运算时，数据来源有 3 处：指令(立即数)、寄存器和存储器(RAM)。CPU 与 RAM 的数据交换有 2 条指令 LW 和 SW。本次简单计算机系统的设计中，对输入模块和输出模块(I/O)的操作等效为对 RAM 的操作，其中 RAM 存储单元 0x00~0x07 号单元被映射到 0x00~0x07 号 I/O 端口寄存器上。通过数码管及其接口电路可直接将存放在 0x00 和 0x01 号 I/O 端口寄存器的 16 位二进制内容显示出来；0x02~0x06 号 I/O 端口寄存器与 4*4 矩阵键盘相连， 键盘输入的第 1 个操作数、运算符和第 2 个操作数对应的数值和运算符编码存放在 0x02~0x06 号 I/O 端口寄存器中。

本项目各部分的功能要求如下：

1. **输入设备**、**输出设备**、以及**输入设备接口模块**、**输出设备接口模块**与暑假小学期“电子技术课程设计”的“计算器的设计与实现”中的相同，可参看“附录 C 暑期计算器已完成模块.pdf”相关部分。该模块的具体实现参考附件。
2. **ALU 运算单元模块**

   ALU 运算单元模块包括两个部分：8 位算术逻辑运算器 ALU 模块和标志寄存器模块， 模块封装图分别见图 17 和图 18，8 位算术逻辑运算器 ALU 模块可完成两个 8 位二进制操作数的“不带进位加”、“不带借位减”、“带进位加”、“带借位减” 、“与”、“或”、“比较”等 7 种运算；标志寄存器模块用于存放算术、逻辑运算过程和结果的一些标志信息，如进/借位标志 Carry、零标志 Zero 等。该模块的具体实现参考附件。

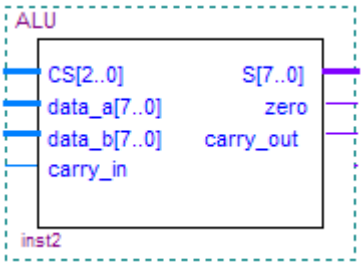

图 17　8 位算术逻辑运算器 ALU 模块的封装图

图 18　标志寄存器模块的封装图

3. 寄存器组模块

CPU 在运算过程中，需要将一些运算数据保存在寄存器中，通常需要的寄存器不止一个，而是以寄存器组的形式出现。寄存器组模块的封装结构图见 19， 该寄存器组内部包含 4 个 8 位寄存器，分别称为 R0、R1、R2、R3， 对应的二进制编号分别为 00B、01B、10B、11B。该模块的具体实现参考附件。

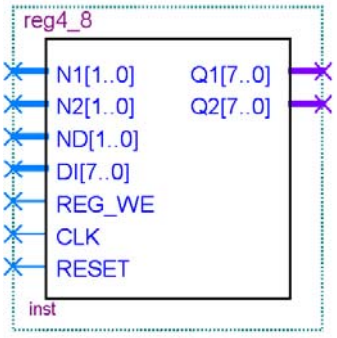

图 19　寄存器组模块的封装结构图

4. ROM 程序存储系统模块

ROM 程序存储系统容量为 256×16，即有 256 个存储单元，地址编号顺序为 0x00~0xFF，每个单元可存放 16 位二进制信息。ROM 主要用来存放用户编写的运行程序。ROM 模块封装结构图见图 20。该模块的具体实现参考附件。

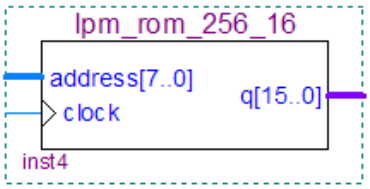

图 20　ROM 程序存储系统模块封装结构图

5. RAM 数据存储系统模块

RAM 数据存储系统容量为 256×8，即有 256 个存储单元，地址编号顺序为 0x00~0xFF， 每个单元可存放 8 位二进制信息。RAM 主要用来存放用户程序的数据。 RAM 模块封装结构图见图 21。该模块的具体实现参考附件。

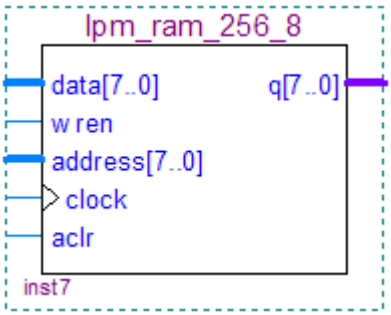

图 21　RAM 数据存储系统模块封装结构图

6. I/O 端口及其映射模块

I/O 端口及其映射模块内部有 8 个双向的 I/O 端口寄存器，编号为 0x00~0x07，其中 0x00、0x01 号 I/O 端口可以与输出设备（数码管）的接口模块相连，控制数码管显示的内容；另 5 个 I/O 端口寄存器可以和输入设备（键盘）的接口模块相连，存放键盘输入的 2 个操作数和 1 个运算符。I/O 端口及其映射模块将对 RAM 存储单元 0x00~0x07 的读/写映射成对 I/O 端口寄存器 0x00~0x07 的读/写， 即所有对 RAM 存储单元 0x00~0x07 的读/写实际是对 I/O 端口寄存器 0x00~0x07 的读/写。 I/O 端口及其映射模块封装图见图 22。该模块的具体实现参考附件。

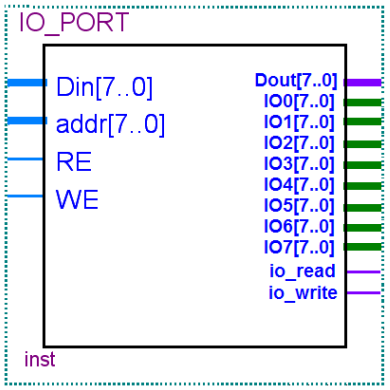

图 22　I/O 端口及其映射模块封装结构图

7. 控制单元模块

控制单元模块包括两个部分：控制器模块和程序指针 PC 模块，模块封装图分别见图 23 和图 24，控制器提供其他模块中用到的各种控制信号，程序指针 PC 模块提供将要执行的指令在 ROM 中的地址。该模块需要根据前面介绍的指令集和数据通路进行设计，设计中可以参考实验任务 1 中相关内容。

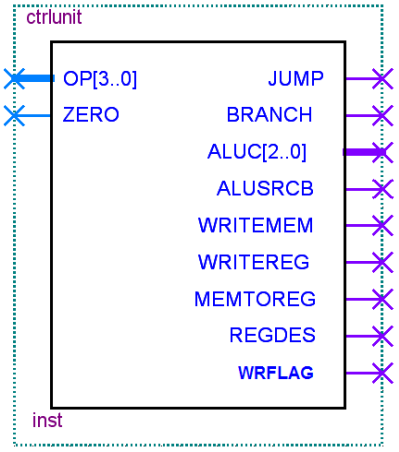

图 23　控制器模块

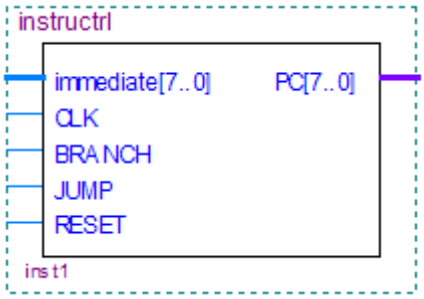

图 24　程序计数器 PC 模块

实验项目中，对简单计算机各基本模块，仅要求完成控制器和 PC 指针模块的设计，其他基本模块会提供一种供参考设计。但需要熟悉所有基本模块的实现，并进行仿真验证。

六、程序编写及举例

完成简单计算机系统的硬件设计后，需要根据指令集编写汇编程序，并将该程序所对应的机器码写到 ROM 中，启动并运行计算机系统。

编写程序时，先分析要实现的功能，参照指令系统中的指令进行基本动作的分解，然后先用汇编语言格式的指令写出这些动作序列（即程序），检查无误后，按照指令的编码规则，再将汇编程序转换成二进制形式的指令（书写时也可以用十六进制表示），然后将其写入 ROM 中，编译后下载至 FPGA，所设计的简单计算机系统通过执行 ROM 中的程序完成相应的功能。

由于设计的指令条数只有 15 条，所以编写程序时，看似简单的一个动作，可能要分成几条指令才能完成。利用双向 I/O 端口，可以和键盘接口模块、数码管接口模块进行交互。

例 1　编写程序完成 8 位减法操作程序（如 125-45， 即 0x7D-0x2D）， 并将结果存放在地址为 0x08 的 RAM 单元中。假设程序存放在从 0x00 单元开始的 ROM 中。

例 1 参考程序如下：

汇编语言格式指令	二进制 机器码指令	十六进制 机器码指令	功能
ANDI　R0,R0,0	1000 0000 0000 0000	0x8000	将 R0 清零
ADDI　R0,R0,0x7d	1010 0000 0111 1101	0xA07D	将 0x7D 赋值给 R0
ANDI　R1, R1,00	1000 0101 0000 0000	0x8500	将 R1 清零
ADDI　R1, R1,0x2d	1010 0101 0010 1101	0xA52D	将 0x2D 赋值给 R1
SUB　R2,R0, R1	0011 0001 1000 0000	0x3180	R2=R0-R1
ANDI　R3,R3,0	1000 1111 0000 0000	0x8F00	将 R3 清零
SW　R2,R3,8	1100 1110 0000 1000	0xCE08	把 R2 数据输出 RAM 单元 0x08
JMP　0	0111 0000 0000 0000	0x7000	跳到程序开始处 0x00,重新运行

假设 FPGA 芯片复位后， 程序计数器 PC 的初始值为 0x00， 则在 ROM 的 0x00 单元处顺序写入的程序代码：0x8000， 0xA07D， 0x8500， 0xA52D， 0x3180， 0x8F00， 0xCD08，0x7000。编译仿真后，验证运算结果，RAM 的 0x08 存储单元存放的值应为 0x50， 即 80。

例 2　编程完成将键盘输入的两个数相减，结果通过数码管显示出来。其中输入的两个数的范围在 0~999 之间。假设程序存放在从 0x00 单元开始的 ROM 中。

演示操作方法：从键盘输入被减数（如 4、5），接着输入运算符减号键“-”，然后再输入减数（如 3、6、9），最后按下等号键“=”，两数相减的结果显示在数码管上（如-324, 负号通过发光二极管 D1 显示）。

假设键盘输入的第 1 个操作数、第 2 个操作数、运算符编码（**假设减运算符“-”存放在键盘接口中的编码为 0x0b**）经键盘接口模块分别顺序存放在 0x02～0x06 号 I/O 端口寄存器中，I/O 端口映射关系见表 A-10。FPGA 芯片复位后，程序计数器 PC 的初始值为 0x00。

表 A-10　I/O 端口映射地址表

I/O 端口地址	对应接口模块内的 I/O 寄存器
0x00	数码管接口 datainL[7..0]
0x01	数码管接口 datainH[7..0]
0x02	矩阵键盘接口 srcL[7..0]
0x03	矩阵键盘接口 srcH[7..0]
0x04	矩阵键盘接口 dstL[7..0]
0x05	矩阵键盘接口 dstH[7..0]
0x06	矩阵键盘接口 ALUOP[7..0]

例 2 参考程序如下：

汇编语言格式指令	二进制 机器码指令	十六进制 机器码指令	功能
ANDI R0,R0,0	1000 0000 0000 0000	0x8000	将 R0 清零
ADDI R0,R0,0x0b	1010 0000 0000 1011	0xA00B	将减运算符编号 0x0b 赋值给 R0
ANDI R3,R3,0	1000 1111 0000 0000	0x8F00	将 R3 清零
LW R2,R3,6	1011 1110 0000 0110	0xBE06	将输入的运算符放在 R2 中
BEQ R2,R0,1	1101 1000 0000 0001	0xD801	比较运算符是否减号,是减号跳到下 2 条处开始运行(即 JMP 的下一条)
JMP 0	0111 0000 0000 0000	0x7000	不是减号跳到程序开始处 0x00,重新运行
ANDI R3,R3,0	1000 1111 0000 0000	0x8F00	将 R3 清零
LW R0,R3,2	1011 1100 0000 0010	0xBC02	将被减数低 8 位放在 R0 中
LW R1,R3,4	1011 1101 0000 0100	0xBD04	将减数低 8 位放在 R1 中
SUB R2,R0, R1	0011 0001 1000 0000	0x3180	R2=R0-R1（低 8 位减）
ANDI R3,R3,0	1000 1111 0000 0000	0x8F00	将 R3 清零
SW R2,R3,0	1100 1110 0000 0000	0xCE00	把 R2 数据输出 I/O 端口 0x00 中,通过数码管显示
ANDI R3,R3,0	1000 1111 0000 0000	0x8F00	将 R3 清零
LW R0,R3,3	1011 1100 0000 0011	0xBC03	将被减数高 8 位放在 R0 中
LW R1,R3,5	1011 1101 0000 0101	0xBD05	将减数高 8 位放在 R1 中
SUBC R2,R0, R1	0101 0001 1000 0000	0x5180	R2=R0-R1(高 8 位带借位减)
ANDI R3,R3,0	1000 1111 0000 0000	0x8F00	将 R3 清零
SW R2,R3,1	1100 1110 0000 0001	0xCE01	把 R2 数据输出 I/O 端口 0x01 中,通过数码管显示
JMP　0	0111 0000 0000 0000	0x7000	跳到程序开始处 0x00,重新运行

由于 FPGA 芯片复位后， 程序计数器 PC 的初始值为 0x00， 则在 ROM 的 0x00 单元处顺序写入的程序代码：0x8000，0xA00B，0x8F00，0xBE02，0xD801，0x7000，0x8F00，0xBC02，0xBD04，0x3180，0x8F00，0xCE00，0x8F00，0xBC03，0xBD05，0x5180，0x8F00，0xCE01，0x7000。编译仿真后，如果从键盘输入“45 – 369 = ”， 数码管上将显示 324，发光二极管 D1 灯亮。

七、项目要求

暑期小学期的“数字计算器的设计与实现”项目已经完成了计算机系统的基本组成中输入和输出接口模块的设计。本学期将根据实验指导书学习和设计 ALU、寄存器组、ROM、RAM 、I/O 端口及其映射、控制单元等模块，**掌握“简单计算机系统”中介绍的指令系统及其编码规则，并能用汇编语言编写程序，并将程序转换成对应的二进制编码(机器码)；在理解了各模块之间的数据通路后，完成对简单计算机系统的综合和调试。**

八、项目设计和调试

课内安排 4 次实验课，分别在第 2～6 周，每次 2 学时，具体内容参见后面的实验 1~3 的安排。按实验盒编号，借给每个同学一套 FPGA 实验系统，同学们需要充分利用课内外时间完成各自的系统设计和调试。实验课地点是西主楼三区 217。第 8 周前统一归还实验盒。请爱护实验设备，勿遗失。

第 1 次　实验 1　简单计算机系统的基本构成和指令集理解
第 2 次　实验 2　简单计算机系统的数据通路和控制器设计
第 3~4 次 实验 3　简单计算机系统的设计和实现