### 计算机组成实验指导书

### 实验八 基于FPGA-CPU汇编语言设计

### 一、实验目的

1利用FPGA-CPU的指令系统，编写汇编语言测试文件。

2通过指令的运行来测试CPU设计是否正确。

### 二、实验原理

整个CSR-FLY硬件平台主要有PC监控系统、外部程序存储器、FPGA-CPU及其下载相关电路，以及控制电路组成。其中PC监控系统主要是由监控通讯软件及相关通信接口等构成，提供一整套的运行和调试功能。图1是CSR-FLY结构示意图。



图1 CSR-FLY结构示意图

控制电路对存储器的读写操作由PC监控软件通过串口通信来控制。控制电路对FPGA-CPU的监控主要包括两部分：对CPU运行时的调试控制功能，以及对FPGA-CPU内部寄存器、地址总线、数据总线等内部数据的实时监测功能，示意图见图2。



图2 控制电路对FPGA-CPU监控示意图

1. 指令系统

为了指令的规整性和便于译码，主要采用定长操作码方案，操作码长度只有8位和16位两种，绝大部分指令为8位操作码。寻址方式包括了寄存器寻址、立即数寻址、直接地址和相对寻址。

根据指令的执行步骤不同，可以把全部指令分为A、B两组。其中A组指令完成的是通用寄存器之间的数据运算或传送，或其他一些特殊操作，在取指之后可一步完成，包括：ADD，SUB，AND，CMP，XOR，TEST，OR，MVRR，DEC，INC，SHL，SHR，ADC，SBB，JR，JRC，JRNC，JRZ，JRNZ，JRS，JRNS，CLC，STC；B组指令完成的是一次内存读写操作，在取指之后可两步完成，包括：JMPA，LDRR，STRR，MVRD。在指令编码时以指令操作码的最高位来区分 A、B组指令，0为A组指令，1为B组指令。见表1。

表1 指令格式表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令格式 | 汇编语句 | CZVS | 指令类型 | 功能说明 |
| 00000000DRSR | ADD DR,SR | \*\*\*\* | A组指令 | DR←DR+SR |
| 00000001DRSR | SUB DR,SR | \*\*\*\* | A组指令 | DR←DR-SR |
| 00000010DRSR | AND DR,SR | \*\*\*\* | A组指令 | DR←DR and SR |
| 00000011DRSR | CMP DR,SR | \*\*\*\* | A组指令 | DR-SR |
| 00000100DRSR | XOR DR,SR | \*\*\*\* | A组指令 | DR←DR xor SR |
| 00000101DRSR | TEST DR,SR | \*\*\*\* | A组指令 | DR and SR |
| 00000110DRSR | OR DR,SR | \*\*\*\* | A组指令 | DR←DR or SR |
| 00000111DRSR | MVRR | …. | A组指令 | DR←SR |
| 00001000DR0000 | DEC DR | \*\*\*\* | A组指令 | DR←DR-1 |
| 00001001DR0000 | INC DR | \*\*\*\* | A组指令 | DR←DR+1 |
| 00001010DR0000 | SHL DR | \*… | A组指令 | DR,C←DR\*2 |
| 00001011DR0000 | SHR DR | \*… | A组指令 | DR,C←DR/2 |
| 00001100DRSR | ADC DR,SR | \*\*\*\* | A组指令 | DR←DR+SR+C |
| 00001101DRSR | SBB DR,SR | \*\*\*\* | A组指令 | DR←DR-SR-C |
| 01000000OFFSET | JR ADR | …. | A组指令 | 无条件跳转 |
| 01000100OFFSET | JRC ADR | …. | A组指令 | C=1时跳转 |
| 01000101OFFSET | JRNC ADR | …. | A组指令 | C=0时跳转 |
| 01000110OFFSET | JRZ ADR | …. | A组指令 | Z=1时跳转 |
| 01000111OFFSET | JRNZ ADR | …. | A组指令 | Z=0时跳转 |
| 01000001OFFSET | JRS ADR | …. | A组指令 | S=1时跳转 |
| 01000011OFFSET | JRNS ADR | …. | A组指令 | S=0时跳转 |
| 011110000000000 | CLC | 0… | A组指令 | C=0 |
| 011110100000000 | STC | 1… | A组指令 | C=1 |
| 1000000000000000ADR | JMPA ADR | …. | B组指令 | 无条件跳转到ADR |
| 10000010DRSR | LDRR DR,[SR] | …. | B组指令 | DR←[SR] |
| 10000011DRSR | STRR [DR],SR | …. | B组指令 | [DR]←SR |
| 10000001DR0000 | MVRD DR,DATA | …. | B组指令 | DR←DATA |

由于没有中断操作，本机指令的执行步骤可概括如下：

读取指令：地址寄存器←指令地址，修改PC内容使其指向下一条将要执行的指令；

读内存，指令寄存器←读出的内容。

分析指令

执行指令：通用寄存器之间的运算或传送，可1步完成；

读写内存，通常要两步完成。

在控制器方面选用了组合逻辑控制器方案。使用节拍来标记每条指令的执行步骤。由指令而定，不同的执行步骤只有5个，故使用3位节拍就足够了。图3是控制器节拍图。



图3 控制器节拍图

2. CPU结构图

由于没有流水和Cache，CPU的结构还是比较简单的。主要的部件就是算术单元ALU、控制逻辑、通用寄存器组、指令寄存器IR、地址寄存器AR、程序计数器PC、标志寄存器、节拍发生器以及一些数据选择器和译码电路。

运算器方面，由一个ALU部件完成全部算数、逻辑、移位操作，输出设置了4个标志位并保存在标志寄存器内。与之相关的还有寄存器组和数据选择器。16个16位的通用寄存器既是参与运算的数据的来源，又是运算结果的暂存地，而数据选择器为ALU正确选择了两个输入数据。

控制器方面，使用了组合逻辑控制器。 CPU总体结构设计图如图4所示。

图4 CPU总体结构设计图

3. CPU各组成部分说明

（1）ALU：组合逻辑部件，对两个16位的输入及进位输入Cin进行4种运算，由3位控制信号控制，见表2。

表2 ALU的操作

|  |  |
| --- | --- |
| I2I1I0 | 功能 |
| 000 | A+B+Cin |
| 001 | A-B-Cin |
| 010 | A与B |
| 011 | A或B |
| 100 | A异或B |
| 101 | B 左移 |
| 110 | B右移 |

（2）数据选择器BUS\_MUX：组合逻辑部件。由于设计中只有ALU部件有加法器的功能，而对程序计数器的自增并没有再另设一个加法器，故ALU的输入组合情况较多。在对全部指令的执行步骤进行了分析后，共有6种情况，故使用了3位选择信号。输入包括：源寄存器数据，目标寄存器数据，带符号位扩展的偏移地址，PC，以及从内存读取的立即数、跳转地址等数据。在3位选择信号的控制下选择ALU的A、B端输入信号，见表3。

表3 数据选择器BUS\_MUX

|  |  |  |
| --- | --- | --- |
| I5 I4 I3 | ALU\_A | ALU\_B |
| 000 | SR | DR |
| 001 | SR | 0 |
| 010 | 0 | DR |
| 011 | OFFSET | PC |
| 100 | 0 | PC |
| 101 | 0 | DATA |

（3）程序计数器PC：时序逻辑部件，16位寄存器。在控制信号pc\_en的控制下可接收ALU的运算结果（高电平接收，否则保持不变）。输出送往地址寄存器，读取指令内容或者送往数据选择器进行通过ALU自增运算。

（4）地址寄存器AR和指令寄存器IR：16位寄存器，地址寄存器用于存放要读、写的内存单元的地址，输出送往地址总线，输入可能为PC内容，也可能为ALU的输出（读、写内存指令）；指令寄存器存放当前执行指令的内容，它的输入来自从内存读取的指令和数据，输出送往控制逻辑。二者共用两位控制信号REC，见表4。

表4 AR和IR的控制信号

|  |  |
| --- | --- |
| REC | 操作 |
| 00 | 无操作 |
| 01 | AR接收PC |
| 11 | AR接收ALU输出 |
| 10 | IR接收 |

（5）寄存器组和寄存器的选择：本实验中的寄存器都为16位，带有清零端和使能端，实际上在写VHDL程序时，通用寄存器以及AR、IR、PC使用的都是同一个模板。按照设计，通用寄存器共有16个，由指令的低8位的全部或其中的高4位或低4位来从寄存器组中选择源寄存器和目的寄存器。寄存器选择部件reg\_mux选择寄存器。

（6）节拍发生器：时序逻辑器件。用多位触发器的输出信号的不同组合状态，来标识每条指令的执行步骤。其功能相当于一个状态机。如前文已述，节拍发生器共有3位触发器。

（7）控制逻辑：组合逻辑控制方式。它根据IR提供的指令内容和由节拍发生器提供指令的执行节拍及其他一些条件信号，如标志寄存器C、V、Z、S，产生计算机各部件当前时刻要用到的控制信号。

（8）REG\_OUT：选择电路。使用选择信号reg\_sel（4位）和sel（2位）对ir，pc，reg\_in，offset，alu\_a，alu\_b，alu\_out，reg\_testa和16位0进行选择，产生一个16位的reg\_data。

4乘法器

首先我们看一下十进制数的乘法。为了方便起见，我们假定十进制数的各位要么为1要么为0，例如1000×1001：

被乘数 1 0 0 010

乘数 × 1 0 0 110

1 0 0 0

0 0 0 0

0 0 0 0

1 0 0 0

积 1 0 0 1 0 0 010

从上面的步骤我们可以看到，

（1）从右到左用乘数的每一位乘以被乘数，每一次乘得的中间结果比上一次的结果往左移一位。

（2）积的位数比被乘数和乘数的位数要多的多。事实上，如果我们忽略符号位，n位的被乘数和m位的乘数相乘的结果的位数有（n+m）位。因此，乘法必须象加法那样处理溢出问题，如果两个32位的数相乘，积也只有32位的时候，就会出现溢出。

在上面的例子中，我们把十进制数的各位限制为0或1。因此，每一步的乘法相当简单：

（1）如果乘数位是1，则简单的复制被乘数到合适的位置（1×被乘数）；

（2）如果乘数位是0，则在合适的位置置0

因为二进制数的各位是0或1，所以与上面情况类似。

既然已经知道了乘法的基本规律，下一步就是设计高度优化的乘法器硬件。为了让大家更明了乘法器的原理，我们一一列举乘法器的三个版本的改进。我们先假定被乘数和乘数都是正数。

初始的设计模拟我们刚才提到的乘法流程，硬件结构如图5所示。假定乘数在32位乘数寄存器里，64位的积寄存器初始化为0，显然每一步需要把被乘数左移一位。左移32次之后，被乘数的32位会被移到左边，因此我们需要64位的被乘数寄存器，初始状态为低32位是被乘数，高32位是0。这个寄存器每一步左移一位，和中间结果对齐，进行相加，相加的结果存在被乘数寄存器里。同时，乘数寄存器右移以决定是×1还是×0。



图5乘法器

图6 说明了进行每1位乘的3个基本的操作步骤。第1步乘数的最低位决定被乘数是否被加到积寄存器中，第2步的左移操作相当于中间结果左移，第3步的右移操作给出乘数的下一位以决定相应的操作。上述3个步骤重复32次就可以得到乘积的最后结果。



图6 乘法器的流程图

举例说明上述步骤。为了节省空间，我们假定被乘数和乘数都是4位的，210×310，也就是00102×00112。表5说明了每一步每一个寄存器内容的变化。乘数栏中下划线说明了下一步的操作。

表5 乘法器举例

| 重复 | 步骤 | 乘数 | 被乘数 | 积 |
| --- | --- | --- | --- | --- |
| 0 | 初始值 | 0011 | 0000 0010 | 0000 0000 |
| 1 | 1：1→积＝积＋被乘数 | 0011 | 0000 0010 | 0000 0010 |
| 2：被乘数左移1位 | 0011 | 0000 0100 | 0000 0010 |
| 3：乘数右移1位 | 0001 | 0000 0100 | 0000 0010 |
| 2 | 1：1→积＝积＋被乘数 | 0001 | 0000 0100 | 0000 0110 |
| 2：被乘数左移1位 | 0001 | 0000 1000 | 0000 0110 |
| 3：乘数右移1位 | 0000 | 0000 1000 | 0000 0110 |
| 3 | 1：0→无操作 | 0000 | 0000 1000 | 0000 0110 |
| 2：被乘数左移1位 | 0000 | 0001 0000 | 0000 0110 |
| 3：乘数右移1位 | 0000 | 0001 0000 | 0000 0110 |
| 4 | 1：0→无操作 | 0000 | 0001 0000 | 0000 0110 |
| 2：被乘数左移1位 | 0000 | 0010 0000 | 0000 0110 |
| 3：乘数右移1位 | 0000 | 0010 0000 | 0000 0110 |

#### 三、实验步骤

（1） 将子板上的JTAG端口和PC机的USB口用ALTERA下载电缆USB Blaster连接。

（2）将USB转串口通讯线连接到PC机的USB口，将RS232通讯延长线和实验箱背面的串口插座连接，打开实验箱电源。

（3）打开模块双击cpu0.qpf。 执行Tools→Programmer命令，将cpu0.sof下载到FPGA中。注意在执行Programmer命令中，应在program/configure下的方框中打勾，然后下载。

CPU中各信号与灯的对应关系如下：

地址线对应灯RA15～RA0。

C、Z、V、S对应灯FC、FZ、FV、FS，显示ALU的状态标志。

数据线对应RD15～RD0。

寄存器内容对应FR15～FR0。

1．Test1（czCode1.TXT）：8位×8位乘法运算

第1个测试程序，初始值R0=25，R1=6。相乘结果（150）存放在R2中。源代码见czCode1.txt。

（1）在QuartusⅡ下将cpu0.sof下载到FPGA中。

（2）打开通讯软件，执行File →Rule Open命令，打开规则文件czRule.txt。

（3）在通讯软件中，执行File →Code Open命令，打开汇编文件czCode1.txt。

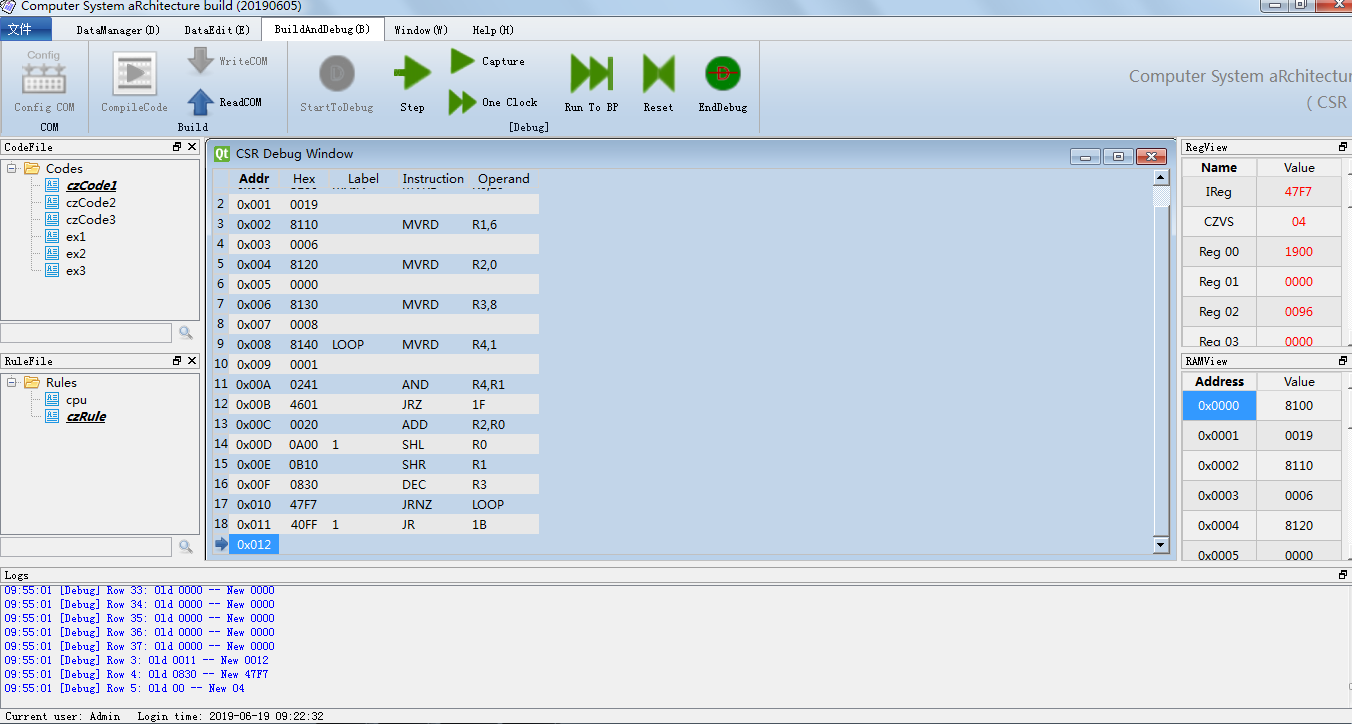
（4）在通讯软件中，执行BuildAndDebug→Compile Code命令，将czCode1.txt编译得到目标二进制文件，显示在机器代码窗口中。

（5）将实验台模式开关置为00000，在通讯软件中，执行BuildAndDebug→Config COM命令，选择端口，设置端口参数。

（6）在通讯软件中，执行BuildAndDebug→WriteCOM命令，将目标二进制代码写入实验台上存储器中。为了检查是否正确写入，执行BuildAndDebug→ReadCOM命令，将存储器中的代码读回，（从0开始，本例中写入18个words，所以从0到17），以做比较。

（6）将实验台模式开关置为01000，为单片机控制FPGA-CPU运行模式。在通讯软件中，利用Debug菜单中的命令调试程序。

（7）最后结果如图4-5所示。

图3.6 测试结果

注意：CPU下载一次后不必再下载，除非重启实验台。但是每次执行新程序前要按实验台上的CPU复位按钮，使FPGA-CPU复位，然后执行StartToDebug→Reset命令，使通讯软件处于开始调试状态。

2．Test2(czCode2.TXT)：内存读写测试

第2个测试程序：将0～32存到0x0100～0x011F，将1～16取出，存到0x0200～0x020F，源代码见EX2.txt。

按照第1个测试程序的实验步骤，对程序进行调试，不过要注意：

（1）不需要对cpu0.sof重新下载。

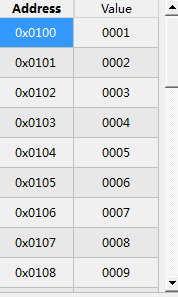
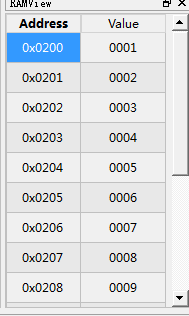
（2）不需要重新打开规则文件czRule.txt。

（3）将实验台模式开关置为00000，将测试文件czCode2.txt编译并下载到实验台上的存储器中。

（4）将实验台模式开关置为01000，首先按实验台上的CPU复位按钮，使FPGA-CPU复位，然后执行通讯软件中的StartToDebug→Reset命令，使通讯软件处于开始调试状态。

（5）设置断点在程序的末尾，双击地址即可，执行通讯软件中的StartToDebug→Run To BP命令，程序运行到断点。

（6）将实验台模式开关置为00000，执行BuildAndDebug→ReadCOM命令，将存储器地址0x100～0x11F、0x0200～0x020F，中的数据读出并比较。

计算机系统结构实验报告

2018级 软件工程 班 姓名 荆薇 学号 201710513088

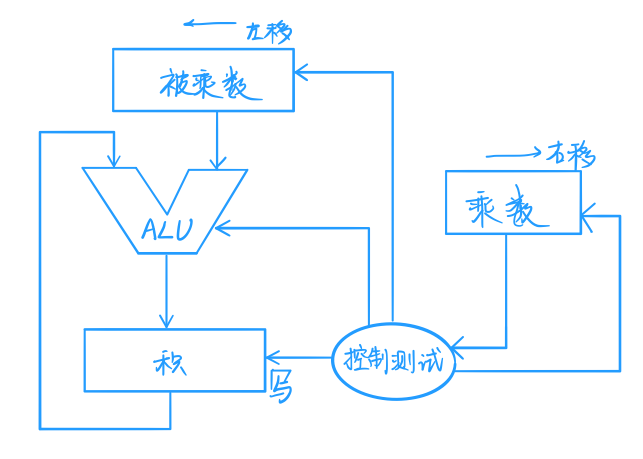
一、实验目的

1、利用FPGA-CPU的指令系统，编写汇编语言测试文件。

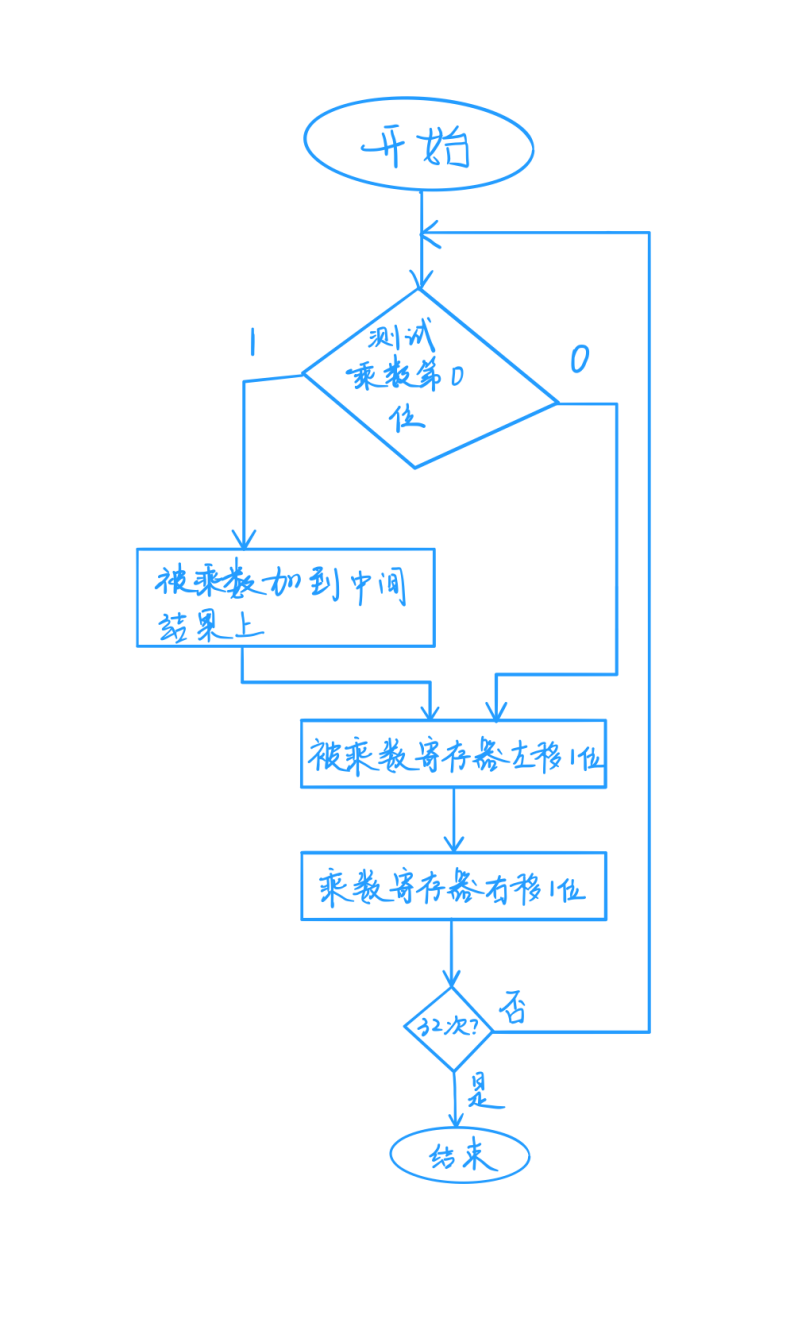
2、通过指令的运行来测试CPU设计是否正确。

二、8位×8位乘法运算程序设计

1、硬件结构设计



2、8位×8位乘法运算程序流程图



3、8位×8位乘法运算程序汇编语言代码

MAIN: MVRD R0,25 ; 被乘数

MVRD R1,6 ; 乘数

MVRD R2,0 ; 结果

MVRD R3,8 ; 循环次数

LOOP: MVRD R4,1

AND R4,R1

JRZ 1F ; r4=0时跳转往前第一个1标号处

ADD R2,R0 ; 否则加到部分积

1: SHL R0 ; 逻辑左移一位

SHR R1 ; 逻辑右移一位

DEC R3 ; r3减1

JRNZ LOOP

1: JR 1B ; 跳转到往后第一个1标号处

4、8位×8位乘法运算程序机器语言代码

0x000 8100

0x001 0019

0x002 8110

0x003 0006

0x004 8120

0x005 0000

0x006 8130

0x007 0008

0x008 8140

0x009 0001

0x00A 0241

0x00B 4601

0x00C 0020

0x00D 0A00

0x00E 0B10

0x00F 0830

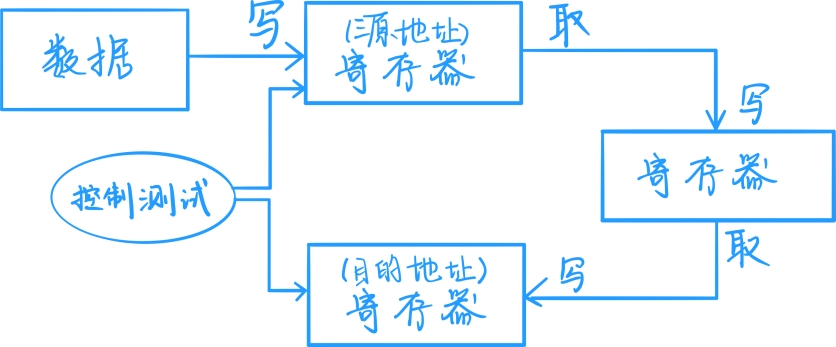
0x010 47F7

0x011 40FF

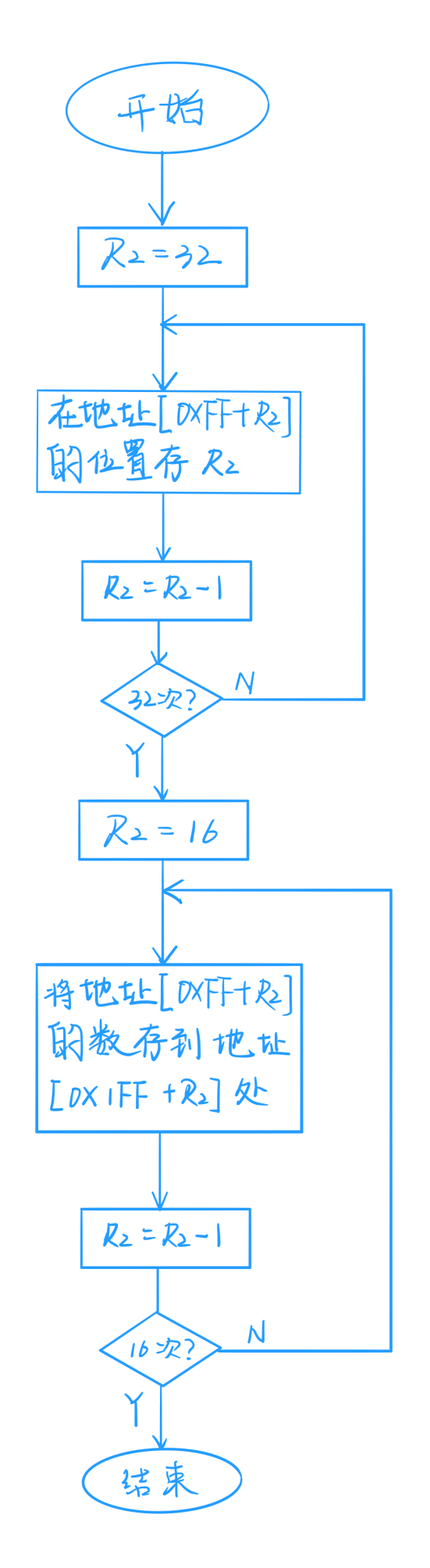
0x012

二、内存读写测试程序设计

1、硬件结构设计



2、内存读写测试程序流程图



3、内存读写测试程序汇编语言代码

MAIN: MVRD R0,0xFF ;源地址

MVRD R1,0x1FF ;目的地址

MVRD R2,32

1: MVRR R3,R0

ADD R3,R2

STRR R3,R2 ;r2数据,r3地址

DEC R2

JRNZ 1B

MVRD R2,16

1: MVRR R3,R0

ADD R3,R2

LDRR R4,R3 ;r4数据,r3地址

MVRR R3,R1

ADD R3,R2

STRR R3,R4 ;r4数据,r3地址

DEC R2

JRNZ 1B

MVRR R5,R5

MVRR R5,R5

1: JR 1B

1. 内存读写测试程序机器语言代码

0x000 8100

0x001 00FF

0x002 8110

0x003 01FF

0x004 8120

0x005 0020

0x006 0730

0x007 0032

0x008 8332

0x009 0820

0x00A 47FB

0x00B 8120

0x00C 0010

0x00D 0730

0x00E 0032

0x00F 8243

0x010 0731

0x011 0032

0x012 8334

0x013 0820

0x014 47F8

0x015 0755

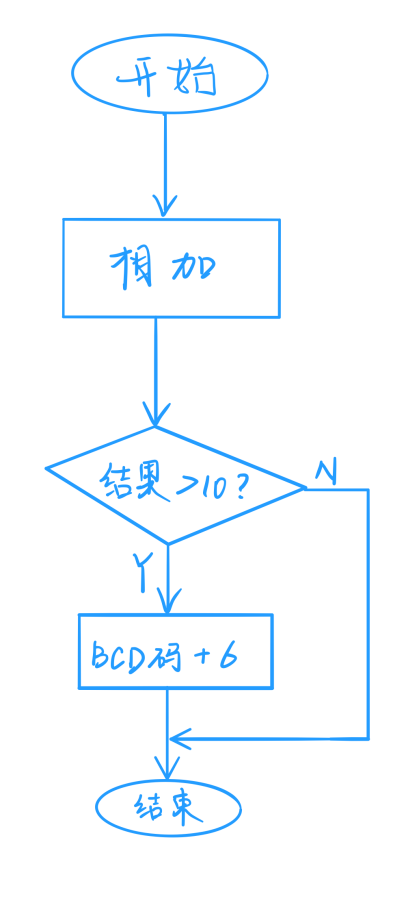
0x016 0755

0x017 40FF

0x018

三、用BCD码完成1位十进制数的加法3+9=12测试程序设计

1、用BCD码完成1位十进制数的加法测试程序流程图



3、用BCD码完成1位十进制数的加法测试程序汇编语言代码

MAIN: MVRD R0,3 ; 3

MVRD R1,9 ; 9

MVRD R2,6 ; BCD码加6

MVRD R4,10 ; 和10比较

ADD R0,R1 ; R0<-R0+R1

CMP R0,R4 ; R0-R4

JRS 1F ; S=1,R0<R4

ADD R0,R2 ; R0<-R0+6

1: JR 1B

4、用BCD码完成1位十进制数的加法测试程序机器语言代码

0x000 8100

0x001 0003

0x002 8110

0x003 0009

0x004 8120

0x005 0006

0x006 8140

0x007 000A

0x008 0001

0x009 0304

0x00A 4101

0x00B 0002

0x00C 40FF

0x00D