Project1实验报告

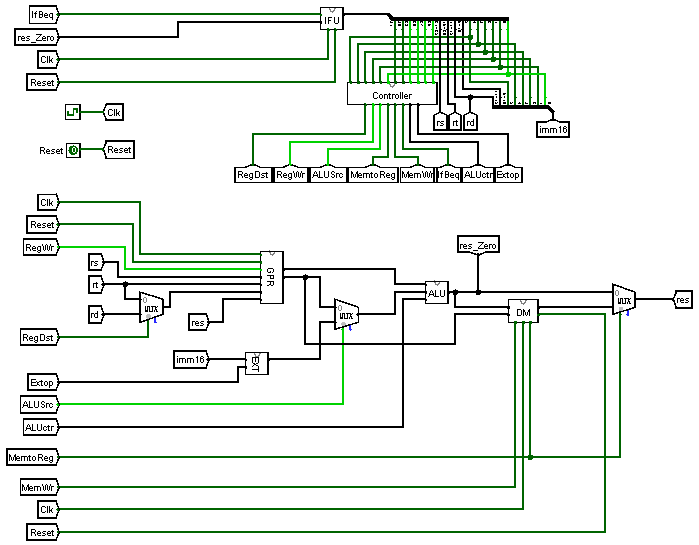
——用logisim完成单周期处理器开发

一、实验内容及要求

1.设计的处理器应支持指令集MIPS—Lite2：addu，subu，ori，lw，sw，beq，lui。其中addu和subu可以不支持实现溢出

2.处理器为单周期设计

二、具体设计及实现

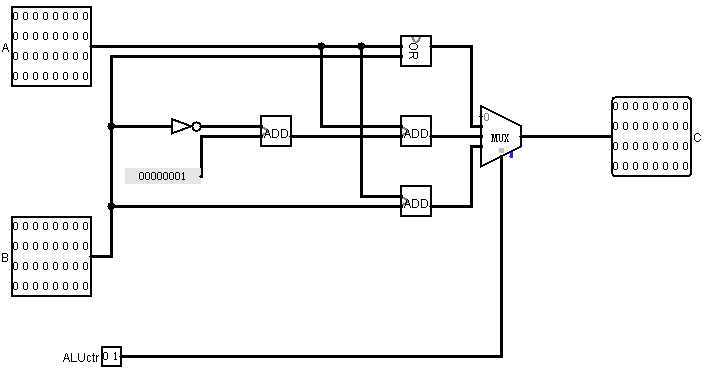
（一）总体结构

1.顶层设计视图包括如下部件：ALU（算术逻辑单元）、EXT（扩展单元）、GPR（寄存器堆）、DM（数据存储器）、IFU（取指令单元）、Controller（控制器）以及多路选择器。

2.顶层设计视图的有效信号包括：时钟信号Clk和清零信号Reset。

（二）各部件设计

1.算术逻辑单元ALU

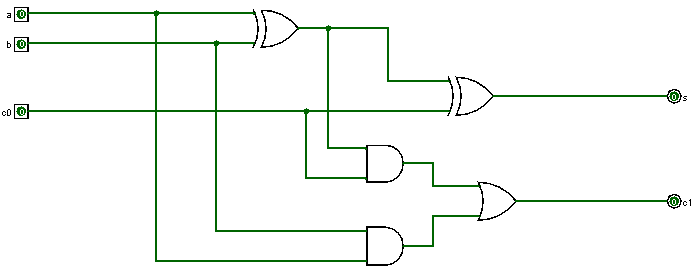


（1）ALU的输入为两个32位数据，输出为一个32位结果，可以实现addu，subu及ori操作。其中控制信号ALUctr控制具体执行的操作，ALUctr = 01，执行ori操作，ALUctr = 10，执行subu操作，ALUctr = 11，执行addu操作。

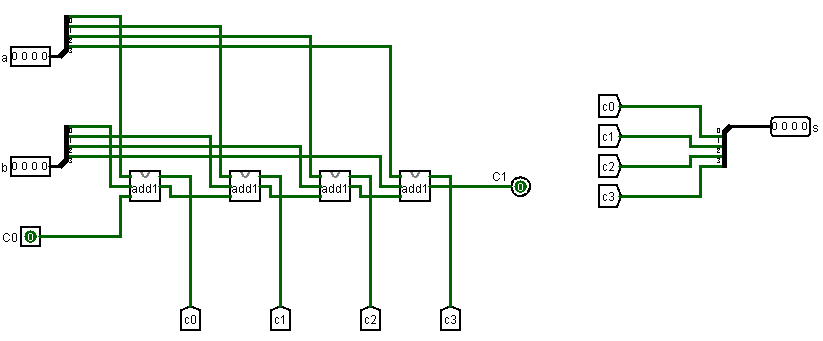
（2）三种运算功能的实现

a）加法实现——32位加法器ADD

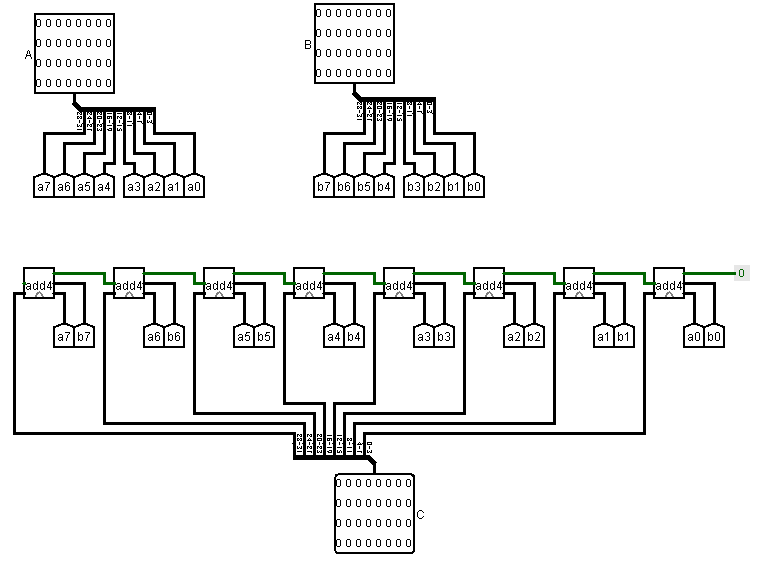
首先，构造一个1位全加器。



利用4个1位全加器构造一个4位串行全加器。



利用8个4位全加器构造出一个32位加法器。

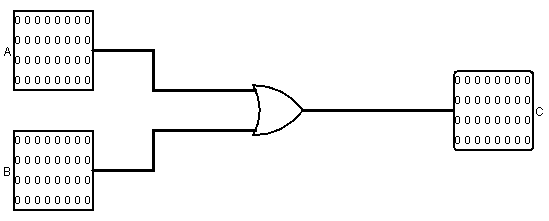


b）减法实现

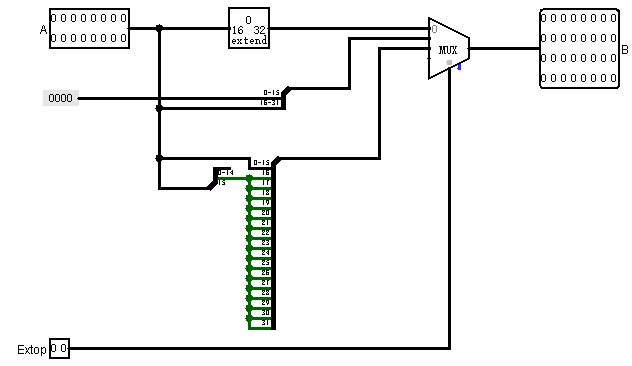
减法实现时利用补码的相关知识，将一个输入数据按位取反末位加一，再与另一输入数据通过32位加法器相加即可实现。

c）或运算实现——32位或运算器

构造一个32位或运算器OR来实现两个32位数据的或运算。



2.扩展单元EXT



（1）扩展单元输入为16位数据，输出为按不同方式扩展后得到32位数据。其中控制信号Extop控制数据的扩展方式，Extop = 00，高位补0，Extop = 01，低位补0，Extop = 10，高位按照符号位扩充。

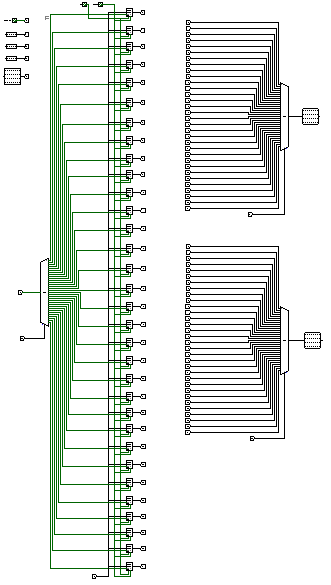
（2）各种扩充方式的实现

a）高位补0：采用Bit Extender组件，高位补0，将16位扩充成32位。

b）低位补0：采用Splitter组件，将16位输入数据送入Splitter的高16位，将低16位置常量0x0000，即实现低位补0功能。

c）高位按符号位扩充：采用Splitter组件，将16位输入数据送入Splitter低16位，同时将该16位输入数据的最高位送入Splitter的全部高16位中，即实现高位按照符号位扩充的功能。

3.寄存器堆GPR



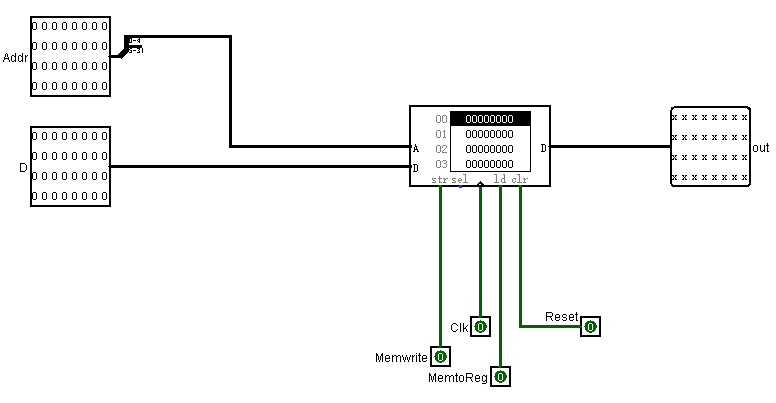
（1）寄存器堆的输入有一个32位写入数据，一个写入数据寄存器号，两个读出数据寄存器号，输出为两个32位输出数据。其中控制信号Regwriter控制当前执行的操作，Regwriter = 1，将数据写入寄存器，Regwriter = 0，读出两个指定寄存器中的数据。三个输入的寄存器号控制数据写入的寄存器以及读出数据的寄存器。

（2）写入数据和读出数据的实现

a）写入数据：Regwriter = 1时，通过多路选择器及写入寄存器号将输入数据写入被选中的寄存器。

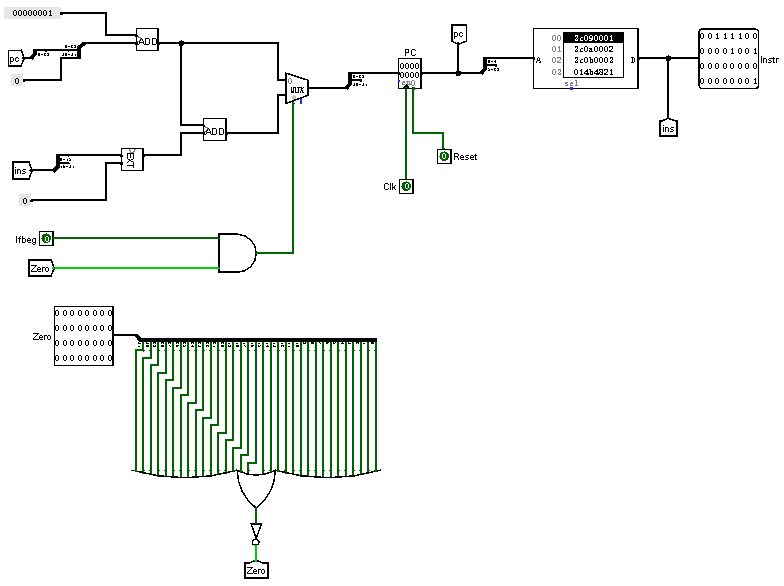
b）读出数据：Regwriter = 0时，通过两个多路选择器，将被选中的两个寄存器中的数据输出。

4.数据存储器DM



数据存储器主要由一个RAM构成，其输入为一个写入地址和一个32位写入数据，输出为一个32位读出数据。其中Memwrite控制数据写入内存，Memwrite为1时，数据写入内存。MemtoReg控制数据从内存中取出，MemtoReg = 1时，从内存中取出数据。

5.取指令单元IFU



（1）IFU 内部包括PC寄存器和IM指令存储器。输出为长32位的一条指令。

（2）PC相关的实现：

a）PC是一个32位的指令计数器，在顺序执行指令时，每一次PC = PC + 4，在执行跳转beq指令时，每一次PC = PC + 4 + imm \* 4。由于每一条指令为4字节，于是可以设置30位PC’，将最低两位固定。

b）将PC’设为30位之后，每一次顺序执行只需PC’ = PC’ + 1，通过一个加法器实现该操作。

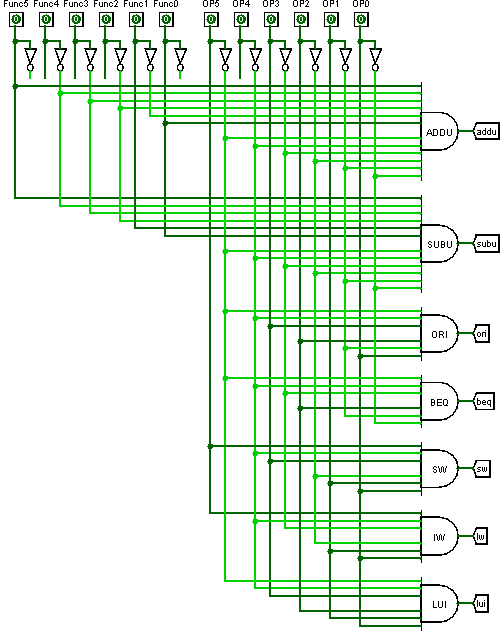
c）将PC’设为30位之后，每一次跳转指令发生时执行PC’ = PC’ + 1 + imm，通过扩展单元将从指令中取出的立即数扩展至32位再与PC’ + 1后的结果通过加法器相加即实现该操作。

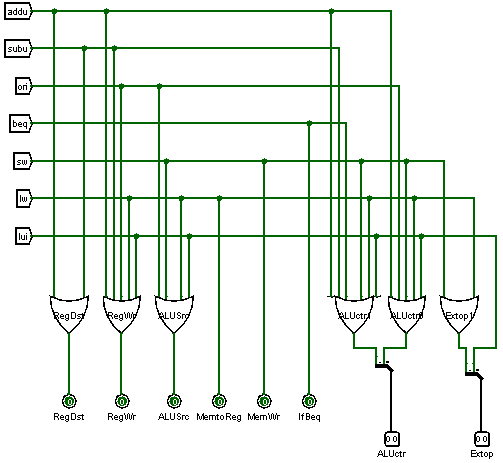
d）通过一个控制信号来选择是顺序执行还是执行beq指令。Ifbeq信号代表此条指令是否为beq指令，Ifbeq = 1时，该指令为beq指令，Ifbeq为0时，该指令应顺序执行。Zero代表beq指令中两个寄存器的差是否为0，Zero为1时，差为0，Zero为1时，差不为0。当Ifbeq = 1且Zero = 1时，执行beq指令，于是用与门连接Ifbeq和Zero信号，并将结果作为多路选择器的控制信号。

（3）指令存储器IM的实现：

指令存储器通过一个ROM实现，PC寄存器的输出为取出指令的地址，ROM的输出为当前取出的指令。

6.控制器Controller





（1）数据通路每个功能部件的每个控制信号的真值表如下

|  |  |  |
| --- | --- | --- |
| Func | 100001 | 100011 |
| OP | 000000 | | 001101 | 000100 | 101011 | 100011 | 001111 |
|  | addu | subu | ori | beq | sw | lw | Lui |
| RegDst | 1 | 1 | 0 | x | x | 0 | 0 |
| RegWr | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| ALUSrc | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| MemtoReg | 0 | 0 | 0 | x | x | 1 | 0 |
| MemWr | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| IfBeq | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| ALUctr | 11 | 10 | 01 | 10 | 11 | 11 | 11 |
| Extop | x | x | 00 | x | 10 | 10 | 01 |

（2）控制信号实现：

首先通过操作码功能得到各个指令标识，再通过指令标识形成最终的控制信号。

三、程序测试

程序测试所用指令如下：

lui $t1, 0001

lui $t2, 0002

lui $t3, 0003

addu $t1, $t2, $t3

subu $t1, $t3, $t2

ori $t1, $t2, 0x1000

sw $t2, 0x1000($t4)

lw $t4, 0x1000($t5)

beq $t1, $t2, +2

addu $t5, $t4, $t3

subu $t6, $t5, $t1

beq $t2, $t4, +2

lui $t6, 0x0004

lui $t7, 0x0005

lui $t8, 0x0006

其所对应机器码如下：

3c090001

3c0a0002

3c0b0003

014b4821

016a4823

35491000

ad8a1000

8dac1000

112a0002

018b6821

01a97023

114c0002

3c0e0004

3c0f0005

3c180006

将该指令加载进IFU对其进行测试得到如下结果：

$t1（9） 00021000

$t2（10） 00020000

$t3（11） 00030000

$t4（12） 00020000

$t5（13） 00050000

$t6（14） 0002f000

$t7（15） 00000000

$t8（24） 00060000

内存单元00001000（在本实验中DM的00号单元） 00020000

该测试结果与预期结果相同。