实习报告

东北林业大学

计算机科学与技术专业

一、课程设计目的

1、掌握计算机主要逻辑功能部件单元电路的工作原理，并学会虚拟仿真各功能部件。

2、掌握计算机控制器的工作原理，并学会虚拟仿真控制器的时序发生器、微操作发生器，及学会控制对应的数据流。

3、设计一个简单的模型机，并运行对应的指令系统，在此基础之上理解计算机各功能模块的工作原理。

4、利用相关的模拟仿真软件实现相关的设计。

二、虚拟环境

操作系统：Windows

虚拟模拟软件：logisim

三、课程设计相关原理

1. 补码表示法

补码是计算机系统中表示有符号整数的一种方法。它的主要优点是可以简化加减法运算的实现。

1. 如果补码的符号位为“0”，表示是一个正数，其原码就是补码。
2. 如果补码的符号位为“1”，表示是一个负数。

补码的计算方法如下：

1. 正数的补码与其原码相同。
2. 负数的补码是其绝对值的原码逐位取反，然后加1。

已知一个数的补码，求原码的操作其实就是对该补码再求补码。

1. 补码时序乘法

补码乘法规则如下：

乘数的最低位增加一辅助位yn+1 = 0，下标n是从0开始，而不是从1开始。判断yn-i yn-i+1的值，决定是“+X”或者“-X",或仅右移一位，得部分积。重复第二步，直到最高位参见操作（y1-y0）\* X，但不做位移，结果得[X\*Y]补

|  |  |  |
| --- | --- | --- |
| yn-i | yn-i+1 |  |
| 1 | 0 | +[-X]补 |
| 0 | 1 | +[X]补 |
| 0 | 0 | 直接右移 |
| 1 | 1 | 直接右移 |

1. 补码矩阵乘法

补码表示法用于表示正负整数，其核心思想是利用最高位作为符号位：0表示正数，1表示负数。补码的计算方法是对于正数，补码与其原码相同；对于负数，其补码为将该数绝对值的二进制表示逐位取反（即取反码）后加1。

逐元素相乘按照补码表示进行元素级乘法，考虑符号位和溢出处理。

累加结果对逐元素相乘的结果进行累加，注意溢出和精度问题。

结果矩阵处理将累加结果转换回预定的位宽和格式。

1. 八条模型机

在本实验中，模型机从内存中取出、解释、执行机器指令都由微指令和与之相配合的时序来完成，每条指令都是采用1地址指令，每条指令取址和执行阶段由4个节拍完成

假设我们要在x86 CPU上执行一个简单的加法指令“ADD AX, BX”。

CPU从内存中读取这条机器指令并解码它。

CPU将这条机器指令转换为一个微程序。这个微程序是由一系列微指令组成的，每个微指令都控制着CPU的一个或多个硬件组件。

CPU开始执行微程序。它从控制存储器中读取微指令，并将它们解码为可以在硬件级别上执行的信号或操作。

这些微指令控制着算术逻辑单元（ALU）来完成实际的加法操作。

最后，结果存储在寄存器BX中。

LDA：从指定内存地址加载数据到累加器。

ADD：将内存中指定地址的数据与累加器中的数据相加。

SUB：将内存中指定地址的数据从累加器中的数据中减去。

STA：将累加器中的数据存储到指定内存地址。

CLA：清除累加器中的数据，即将累加器置零。

SHR：将累加器中的数据右移一位。

JMP：跳转到指定的内存地址执行。

STP：停止执行。

指令格式

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| OP（操作码） | | |  | Ad（取地址） | | | | |
| 3位 |  |  | 1位 | | 4位 |  |  |  |

八位模型机对应八条指令

|  |  |  |
| --- | --- | --- |
| OP码汇编码 | 指令助记 | 功能说明 |
| 000 | LDA M | 取值 |
| 001 | ADD M | 相加 |
| 010 | SUB M | 相减 |
| 011 | STA M | 存值 |
| 100 | SHR | 右移 |
| 101 | CLA | 清空 |
| 110 | JMP M | 跳转 |
| 111 | STP | 停机 |

八条指令对应指令的微操作

其中取值周期是八个指令共有的。

FE周期（取指周期）

|  |  |
| --- | --- |
| T0 | PC->MAR 1->R |
| T1 | M[MAR]->MDR |
| T2 | MDR->IR OP(IR)->ID |
| T3 | PC+1->PC |

而在执行周期中，每个执行周期都是完全不同的，分别对应不同的微指令。

EX周期（执行周期）

ADD M

|  |  |
| --- | --- |
| T0 | Ad(IR)-> MAR 1->R |
| T1 | M[MAR]->MDR |
| T2 | MDR->X |
| T3 | ACC+ |

SUB M

|  |  |
| --- | --- |
| T0 | Ad(IR)-> MAR 1->R |
| T1 | M[MAR]->MDR |
| T2 | MDR->X |
| T3 | ACC+ |

LDA M

|  |  |
| --- | --- |
| T0 | Ad(IR)-> MAR 1->R |
| T1 | M[MAR]->MDR |
| T2 | MDR->ACC |
| T3 |  |

STA M

|  |  |
| --- | --- |
| T0 | Ad(IR)-> MAR 1->W |
| T1 | ACC->MDR |
| T2 | MDR-> M[MAR] |
| T3 |  |

JMP M

|  |  |
| --- | --- |
| T0 |  |
| T1 |  |
| T2 |  |
| T3 | Ad(IR)->PC |

CLA

|  |  |
| --- | --- |
| T0 |  |
| T1 |  |
| T2 |  |
| T3 | 0->ACC |

SHR

|  |  |
| --- | --- |
| T0 |  |
| T1 |  |
| T2 |  |
| T3 | ACC/2->ACC |

微操作表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 周期 | 时钟 | 微操作 | ADD | LDA | STA | JMP | CLA | SHR | SUB | STP |
| FE | T0 | PC->MAR | √ | √ | √ | √ | √ | √ | √ | √ |
| 1->R | √ | √ | √ | √ | √ | √ | √ | √ |
| T1 | M[MAR]->MDR | √ | √ | √ | √ | √ | √ | √ | √ |
| T2 | MDR->IR | √ | √ | √ | √ | √ | √ | √ | √ |
| OP(IR)->ID | √ | √ | √ | √ | √ | √ | √ | √ |
| T3 | PC+1->PC | √ | √ | √ | √ | √ | √ | √ | √ |
| EX | T0 | Ad(IR)-> MAR | √ | √ | √ |  |  |  | √ |  |
| 1->R/W | √ | √ | √ |  |  |  | √ |  |
| T1 | M[MAR]->MDR | √ | √ |  |  |  |  | √ |  |
| ACC->MDR |  |  | √ |  |  |  |  |  |
| T2 | ACC+MDR->ACC | √ |  |  |  |  |  | √ |  |
| MDR->ACC |  | √ |  |  |  |  |  |  |
| MDR-> M[MAR] |  |  | √ |  |  |  |  |  |
| T3 | Ad(IR)->PC |  |  |  | √ |  |  |  |  |
| 0->ACC |  |  |  |  | √ |  |  |  |
| ACC/2->ACC |  |  |  |  |  | √ |  |  |

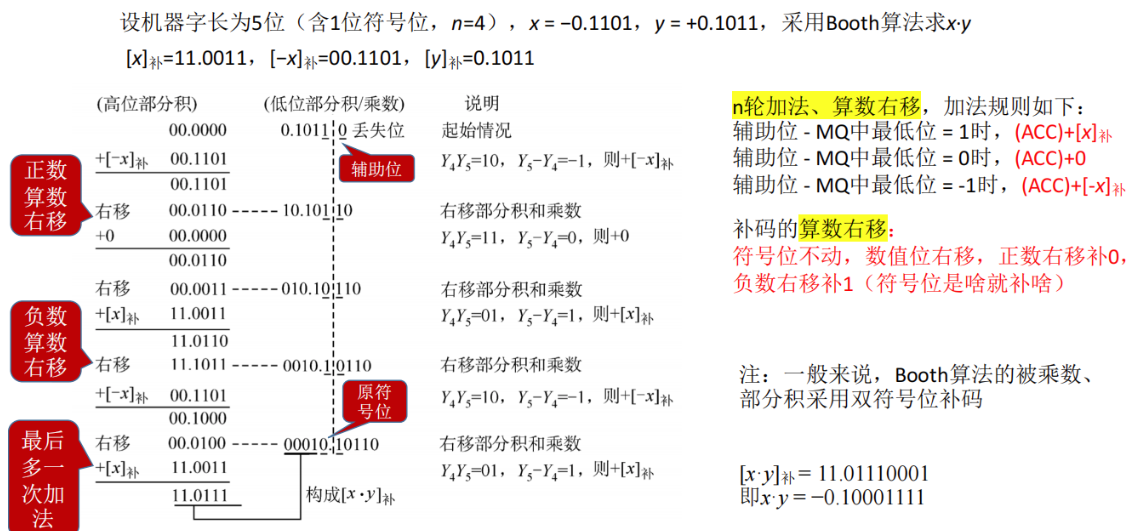
根据微操作表来进行设计电路

四、课程设计过程

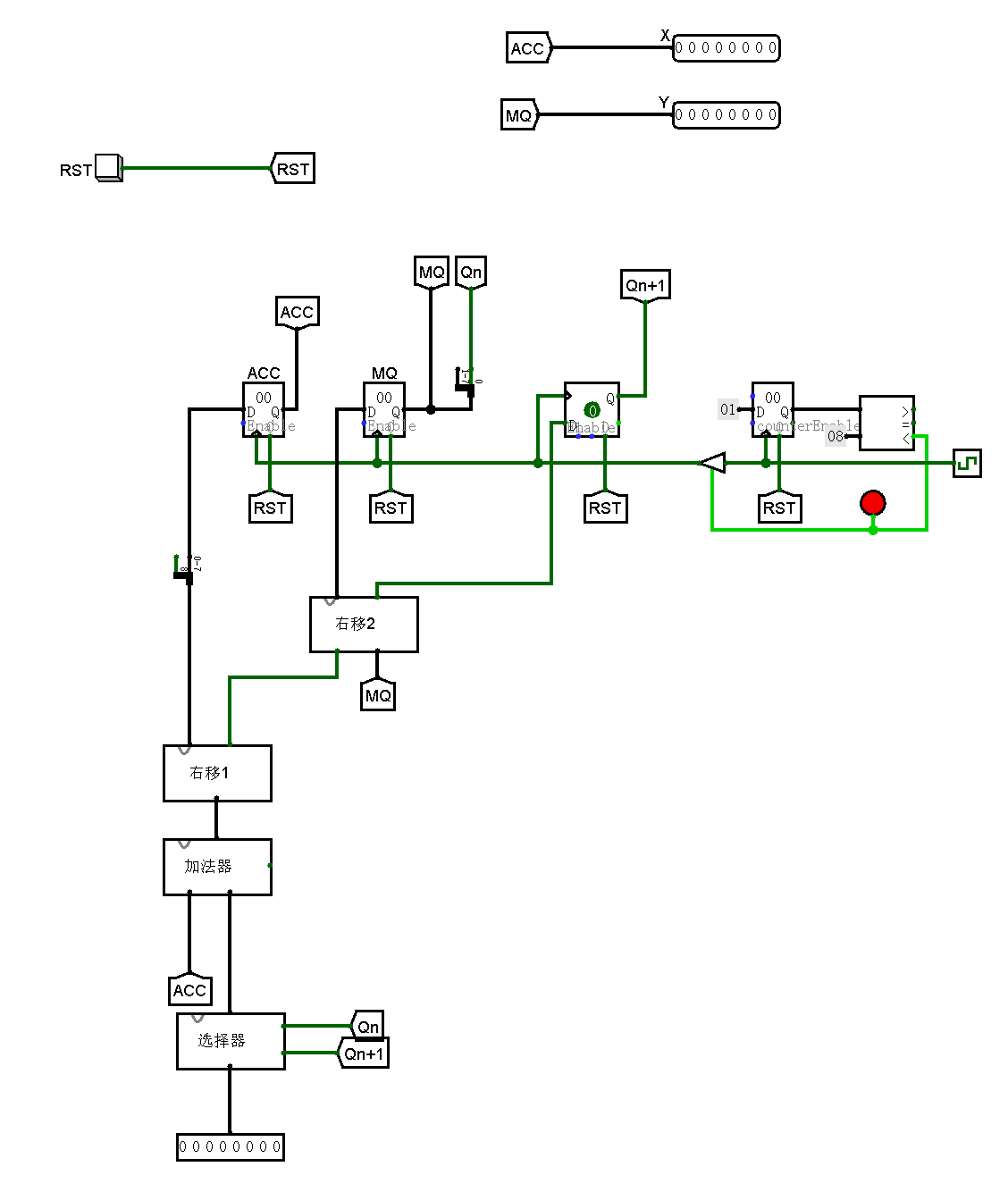
1. 补码时序乘法

进行n轮加法、移位，最后再加一次加法。每次加法可能是加0、加补码、减去补码。每次移位都是补码的算术右移。符号位参与运算

算法原理图：

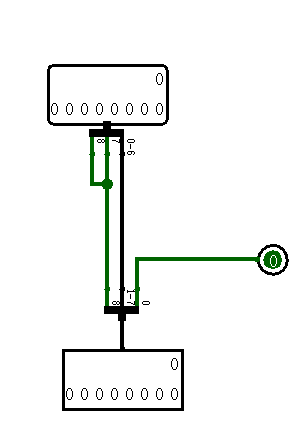


整体图：

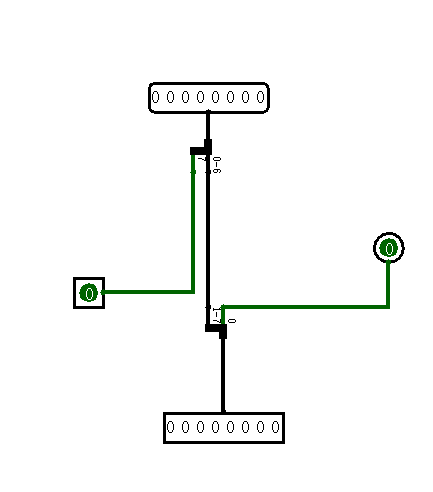


各部分子图：

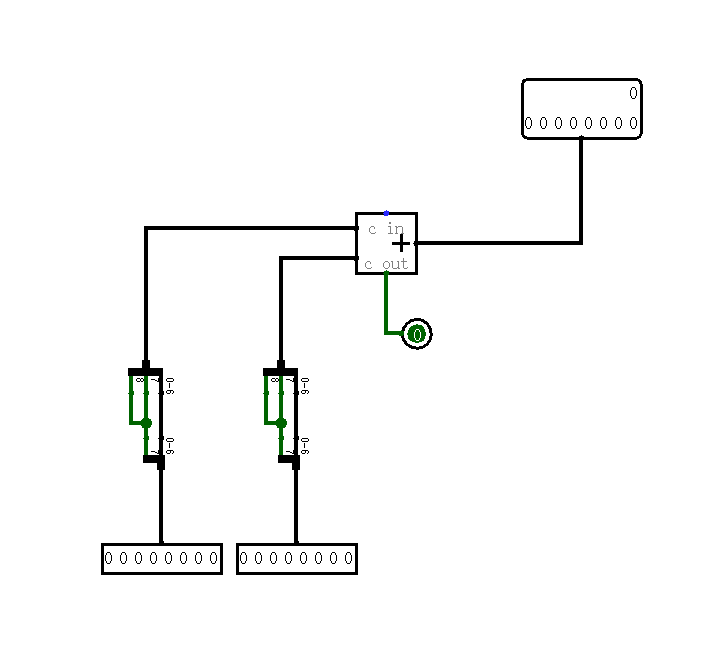
右移1：



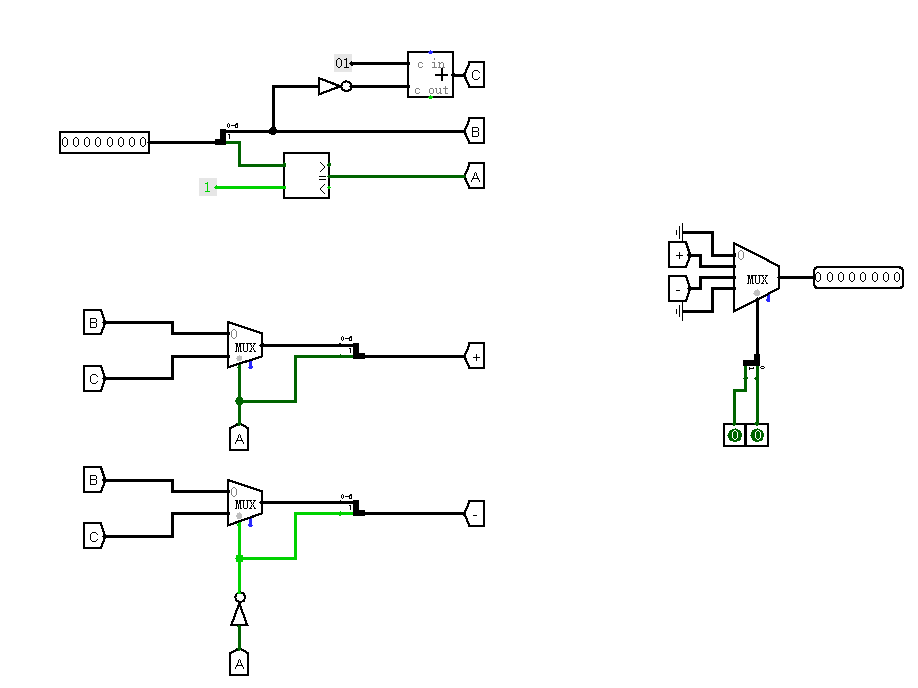
右移2：



加法器：



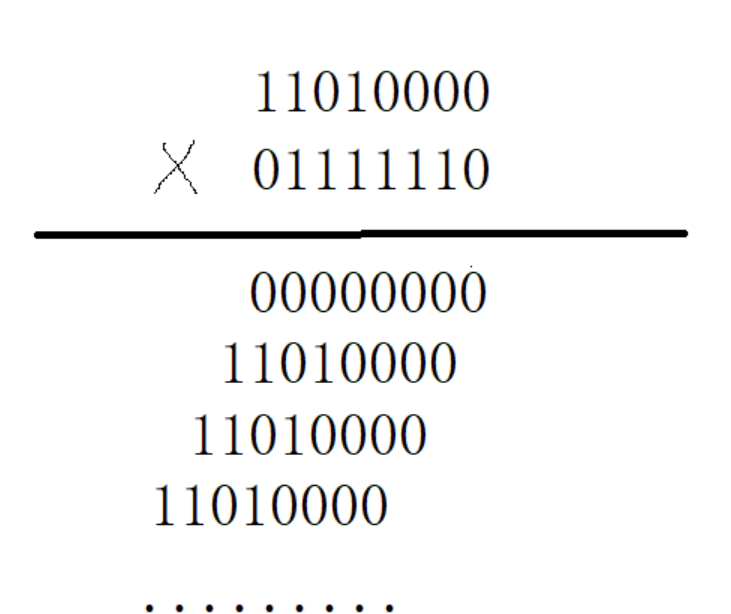
判断端



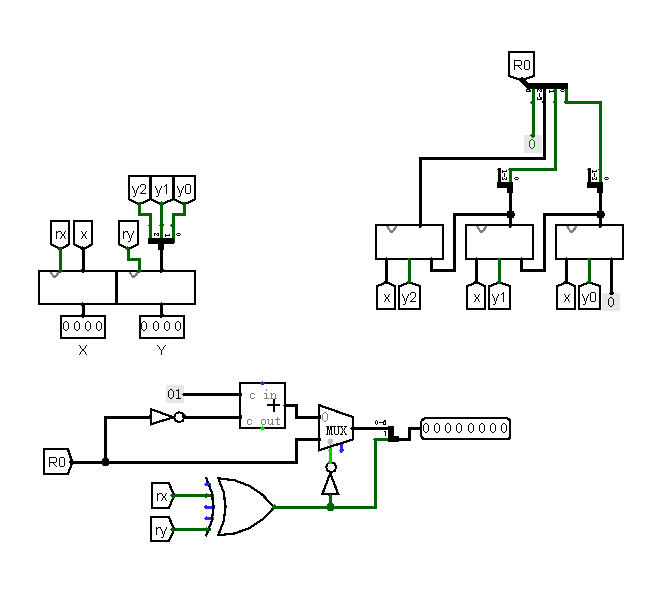
1. 补码矩阵乘法

补码矩阵相当于原码一位一位进行相乘相加，最后在转换为补码。

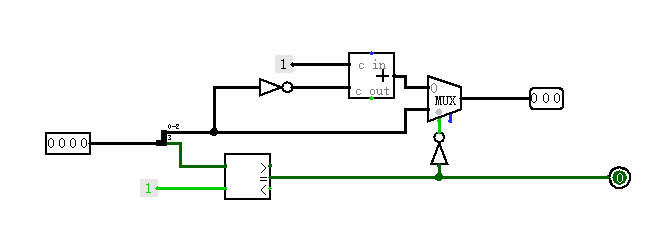
原理图：



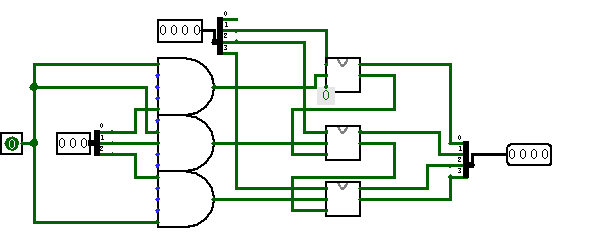
整体图：



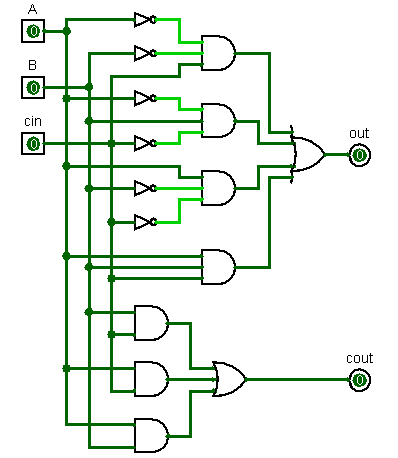
选择器：



加法器：

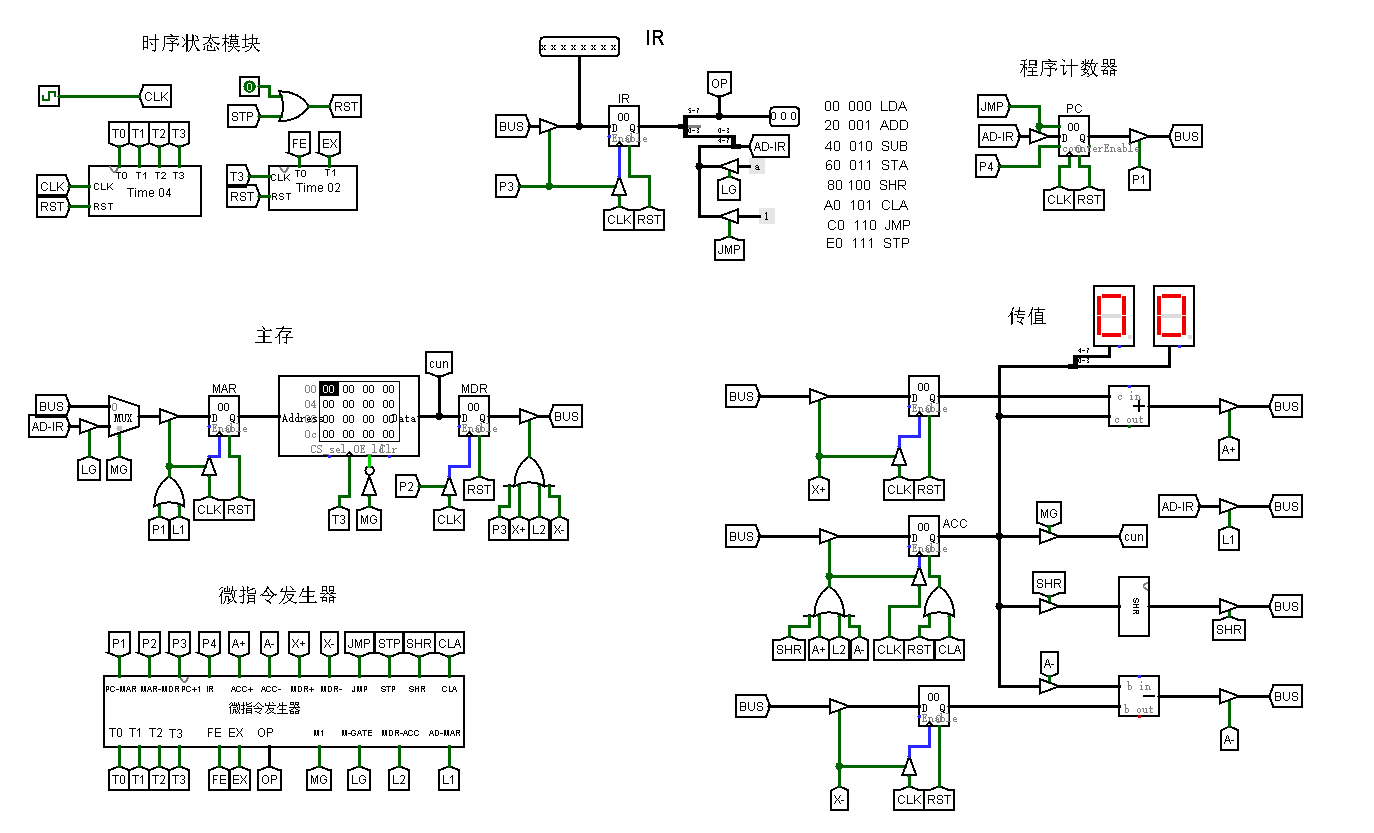


二位全加器：

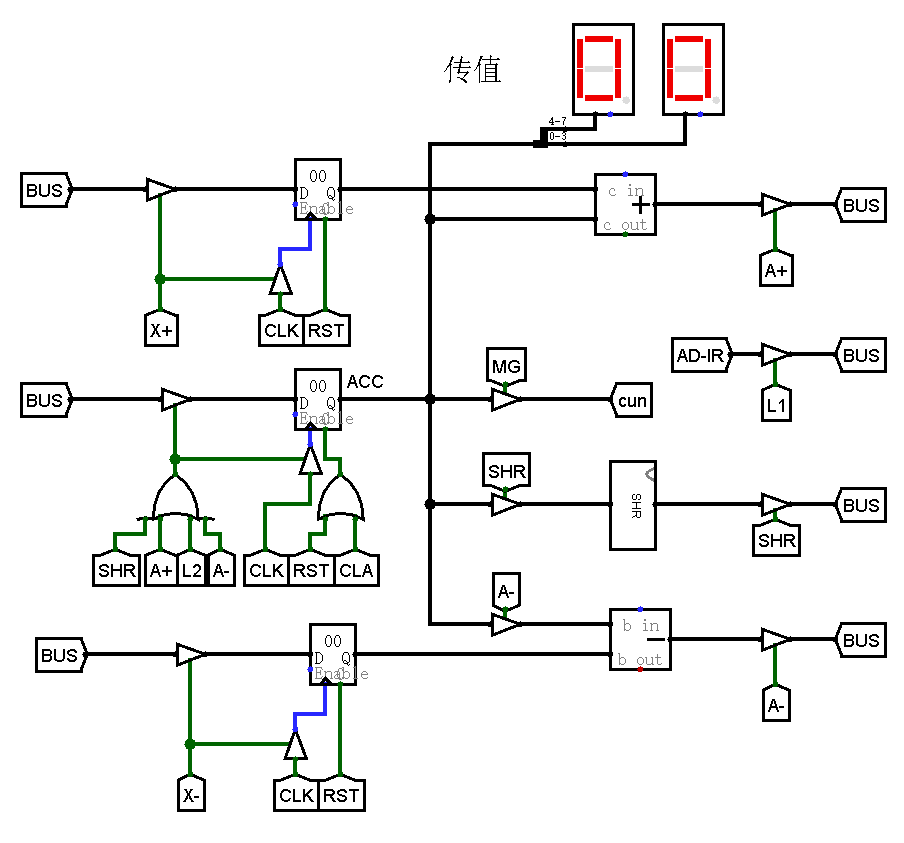
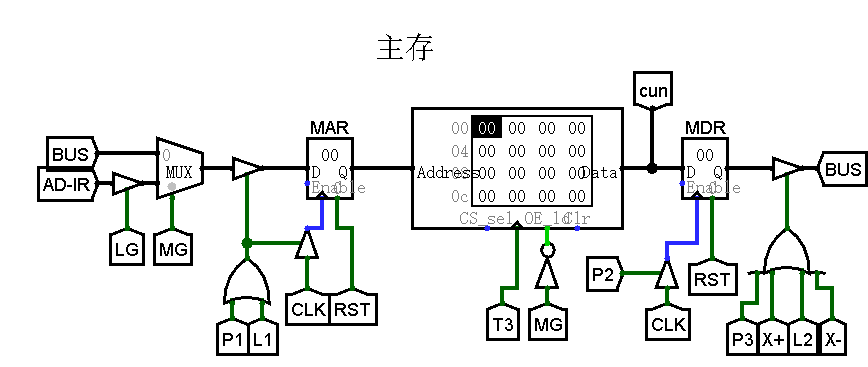
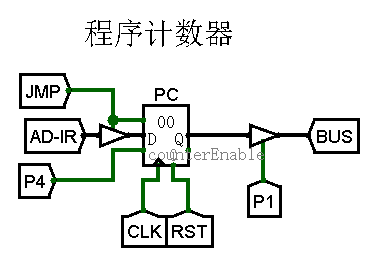
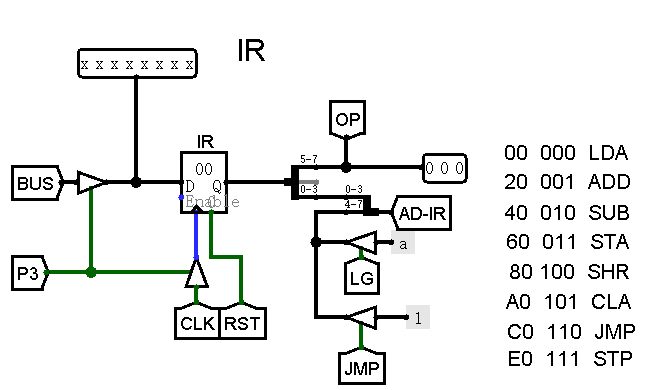
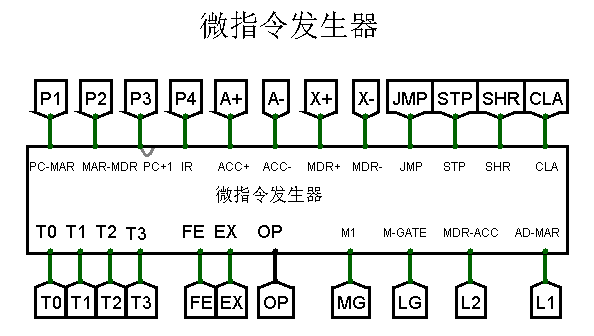
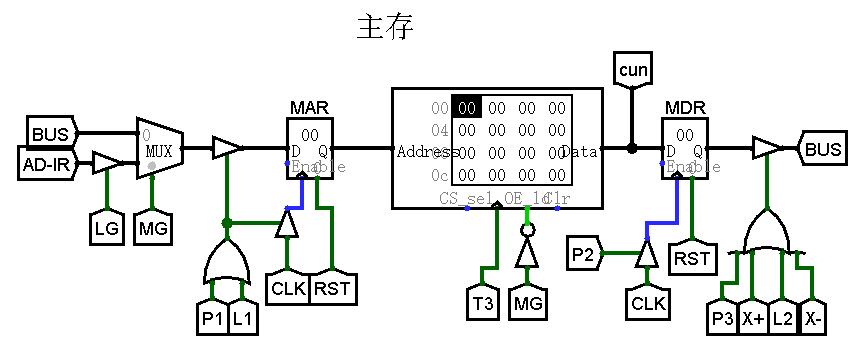
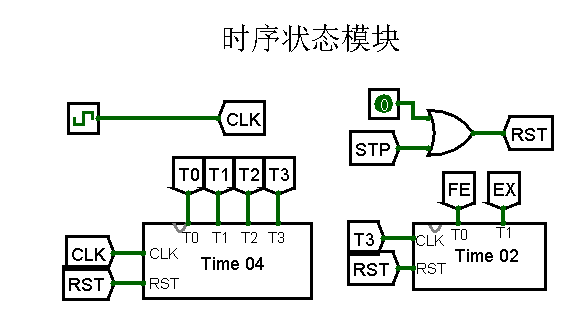


1. 八条模型机

整体图：

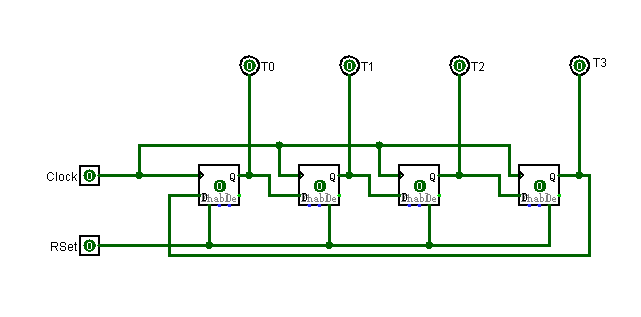


各部分模块：

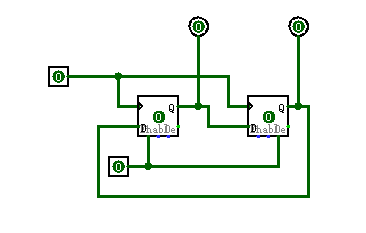


2和4节拍发生器共有8个节拍用于取值和执行周期，前四拍用于取值，后四拍用于执行。

节拍时钟：



周期时钟：



(1) LDA指令

操作：从内存中加载数据到累加器。

从PC中取指，加载到IR中。

从IR中提取操作数（内存地址）。

将该地址的数据加载到累加器（ACC）中。

(2) ADD指令

操作：将指定内存地址的数据与累加器中的数据相加。

从PC中取指，加载到IR中。

从IR中提取操作数（内存地址）。

从该地址读取数据，与ACC中的数据相加，结果存回ACC。

(3) SUB指令

操作：将指定内存地址的数据从累加器中的数据中减去。

从PC中取指，加载到IR中。

从IR中提取操作数（内存地址）。

从该地址读取数据，从ACC中的数据中减去，结果存回ACC。

(4) STA指令

操作：将累加器中的数据存储到指定内存地址。

从PC中取指，加载到IR中。

从IR中提取操作数（内存地址）。

将ACC中的数据存储到该地址。

(5) CLA指令

操作：清除累加器中的数据。

从PC中取指，加载到IR中。

清除ACC中的数据。

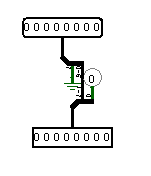
(6) SHR指令

操作：将累加器中的数据右移一位。

从PC中取指，加载到IR中。

右移ACC中的数据。

右移SHR电路：



(7) JMP指令

操作：跳转到指定的内存地址执行。

从PC中取指，加载到IR中。

从IR中提取操作数（内存地址）。

将该地址加载到PC中。

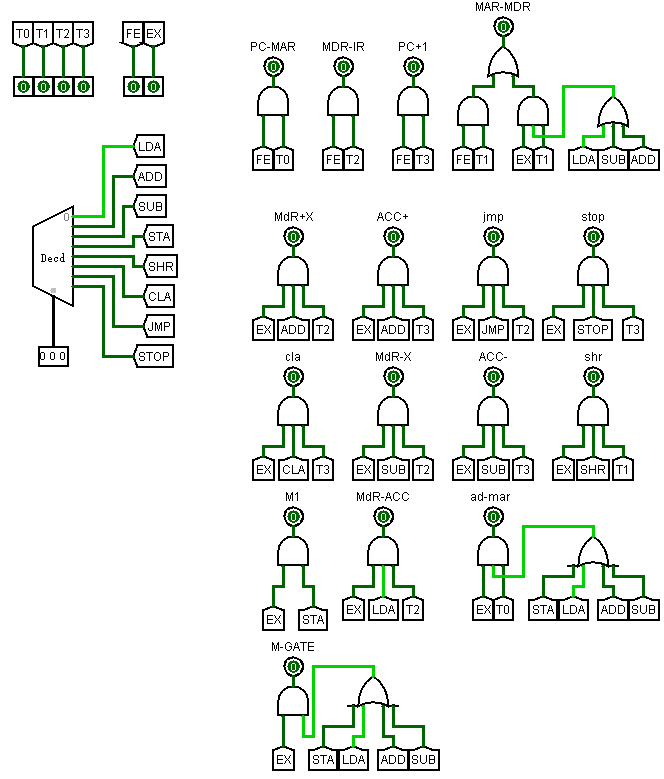
(8) STP指令

操作：停止执行。

从PC中取指，加载到IR中。

停止时钟信号，停止执行。

微指令发生器内部电路图：



五、实习小结

这次实验的目的是通过补码一位乘法、补码矩阵乘法和八条指令模型机的设计与实现，加深对计算机组成原理的理解，掌握计算机基本运算的实现方法，了解指令系统的构建过程。实现了补码一位乘法运算，通过模拟计算机运算的基本操作，理解补码表示法及其在运算中的应用。实现了补码矩阵乘法，模拟了矩阵运算的过程，加深了对多位补码运算的理解。使用Logisim仿真软件设计并实现了包含LDA、ADD、SUB、STA、CLA、SHR、JMP、STP八条指令的模型机，通过模拟简单的指令系统，理解计算机指令的执行过程和控制逻辑。

通过补码进行二进制数的加减运算，解决了正负数统一处理的问题。理解了矩阵运算在计算机中的实现方法，尤其是多位数的运算。学习了如何设计计算机的基本指令系统，并通过仿真软件实现，使抽象的理论知识得以具体化。

在实现补码一位乘法时，初始对补码表示法及其运算规则的理解不够深入，导致运算结果出现偏差。通过查阅资料和反复调试，最终正确实现了补码运算。补码矩阵乘法的实现过程中，涉及多位数的运算，运算过程复杂且容易出错。通过逐步拆解问题，分步实现每一部分的功能，并通过测试逐一排除错误，最终成功实现了矩阵乘法。八条指令模型机的设计涉及控制逻辑的实现，初次设计时对控制信号的理解不够透彻，导致指令执行过程不顺畅。通过仔细分析每条指令的执行过程，重新设计控制信号的生成逻辑，最终实现了完整的指令系统。

通过这次实验，我深刻理解了补码表示法及其在计算机运算中的应用，掌握了矩阵运算的实现方法，了解了计算机指令系统的设计与实现过程。此外，通过动手实践，锻炼了自己的问题分析与解决能力，加深了对计算机组成原理的理解。实验过程中遇到的困难和解决过程，也使我更加意识到理论知识与实际应用的差距，促使我在今后的学习中更加注重理论与实践的结合。

这次实验不仅提升了我的专业技能，还增强了我对计算机科学的兴趣，为今后的学习和研究打下了坚实的基础。

实习日志表

|  |  |
| --- | --- |
| 6月17日 | 指导教师布置课程设计任务，说明课程设计要求 |
| 6月18日 | 地点：丹青913  了解补码表示法、矩阵运算和指令系统的基本原理。通过教材、学术论文和在线资源深入学习，掌握计算机组成原理中的核心概念，尤其是补码运算和指令执行的细节。 |
| 6月19日 | 地点：丹青913  搜索与补码运算、矩阵乘法和计算机指令系统相关的资料，包括技术文档、示例代码和教学视频。收集和整理资料，形成对各个项目的全面理解，确保设计和实现过程中有足够的参考依据。 |
| 6月20日 | 地点：丹青913  分析课程设计的具体要求，明确每个项目的功能和性能需求。细化需求，制定具体的实现目标，为后续设计和开发工作提供指导。 |
| 6月21日 | 地点：丹青913  根据需求分析，设计补码一位乘法、补码矩阵乘法和八条指令模型机的实现方案。绘制流程图和结构图，详细描述每个模块的功能和交互方式，确保设计方案可行且易于实现。 |
| 6月24日 | 地点：丹青913  根据设计方案，使用Logisim仿真软件实现各个项目。 |
| 6月25日 | 地点：丹青913  对每个实现的项目进行全面测试，确保其符合设计要求。编写测试用例，模拟各种输入情况，验证运算结果和指令执行的正确性，发现并修复潜在的问题。 |
| 6月26日 | 地点：丹青913  针对测试过程中发现的问题，进行详细的调试和优化。分析问题根源，设计方案，提高项目的稳定性和性能，确保最终实现的质量。 |
| 6月27日 | 地点：丹青913  总结整个课程设计的过程和收获。包括项目的设计与实现、遇到的问题及解决方法、个人的学习心得和反思，为未来的学习和研究提供参考和借鉴。 |
| 6月28日 | 撰写相关的课设报告，查阅相关的资料 |

|  |
| --- |
| 指导教师意见  综合成绩：  指导教师签字：  年 月 日 |