# 计算机组成原理课程设计报告

班级：物联网__1301_班　　姓名：_石杰元_____ 学号：_20133717_____

完成时间：_____2016/1/10_____

一、课程设计目的

1．在实验机上设计实现机器指令及对应的微指令〔微程序〕并验证，从而进一步掌握微程序设计控制器的根本方法并了解指令系统与硬件结构的对应关系；

2．通过控制器的微程序设计，综合理解计算机组成原理课程的核心知识并进一步建立整机系统的概念；

3．培养综合实践及独立分析、解决问题的能力。

二、课程设计的任务

针对 COP2000 实验仪，从详细了解该模型机的指令/微指令系统入手，以实现乘法和除法运算功能为应用目标，在 COP2000 的集成开发环境下，设计全新的指令系统并编写对应的微程序；之后编写实现乘法和除法的程序进行设计的验证。

三、 课程设计使用的设备〔环境〕

1．硬件

- COP2000 实验仪
- PC 机

2．软件

- COP2000 仿真软件

四、课程设计的具体内容〔步骤〕

1．详细了解并掌握 COP 2000 模型机的微程序控制器原理，通过综合实验来实现

**该模型机指令系统的特点：**

COP2000 模型机包括了一个标准 CPU 所具备所有部件，这些部件包括：运算器 ALU、累加器 A、工作存放器 W、左移门 L、直通门 D、右移门 R、存放器组 R0-R3、程序计数器 PC、地址存放器 MAR、堆栈存放器 ST、中断向量存放器 IA、输入端口 IN、输出端口存放器 OUT、程序存储器 EM、指令存放器 IR、微程序计数器 uPC、微程序存储器 uM，以及中断控制电路、跳转控制电路。其中运算器和中断控制电路以及跳转控制电路用 CPLD 来实现，其它电路都是用离散的数字电路组成。微程序控制局部也可以用组合逻辑控制来代替。

模型机为 8 位机，数据总线、地址总线都为 8 位，但其工作原理与 16 位机相同。相比而言 8 位机实验减少了烦琐的连线，但其原理却更容易被学生理解、吸收。

模型机的指令码为 8 位，根据指令类型的不同，可以有 0 到 2 个操作数。指令码的最低两位用来选择 R0-R3 存放器，在微程序控制方式中，用指令码做为微地址来寻址微程序存储器，找到执行该指令的微程序。而在组合逻辑控制方式中，按时序用指令码产生相应的控制位。在本模型机中，一条指令最多分四个状态周期，一个状态周期为一个时钟脉冲，每个状态周期产生不同的控制逻辑，实现模型机的各种功能。模型机有 24 位控制位以控制存放器的输入、输出，选择运算器的运算功能，存储器的读写。

模型机的缺省的指令集分几大类： 算术运算指令、逻辑运算指令、移位指令、数据传输指令、跳转指令、中断返回指令、输入/输出指令。用户可以通过 COP2000 计算机组成原理实验软件或组成原理实验仪来设计自己的指令集。

模型机的寻址方式分五种：

累加器寻址：

操作数为累加器 A，例如“CPL　A”是将累加器 A 值取反，还有些指令是隐含寻址累加器 A，例如“OUT”是将累加器 A 的值输出到输出端口存放器 OUT。

存放器寻址：

参与运算的数据在 R0-R3 的存放器中，例如 “ADD　A，R0”指令是将存放器 R0 的值加上累加器 A 的值，再存入累加器 A 中。

存放器间接寻址：

参与运算的数据在存储器 EM 中，数据的地址在存放器 R0-R3 中，例如 “MOV　A，@R1”指令是将存放器 R1 的值做为地址，把存储器 EM 中该地址的内容送入累加器 A 中。

存储器直接寻址：

参与运算的数据在存储器 EM 中，数据的地址为指令的操作数。例如“AND　A，40H”指令是将存储器 EM 中 40H 单元的数据与累加器 A 的值做逻辑与运算，结果存入累加器 A。

立即数寻址：

参与运算的数据为指令的操作数。例如 “SUB　A，#10H”是从累加器 A 中减去立即数 10H，结果存入累加器 A。

**该模型机微指令系统的特点〔包括其微指令格式的说明等〕：**

① 总体概述

该模型机的微命令是以**直接表示法**进行编码的，其特点是操作控制字段中的**每一位代表一个微命令**。这种方法的优点是简单直观，其输出直接用于控制。缺点是微指令字较长，因而使控制存储器容量较大。

② 微指令格式的说明

模型机有 24 位控制位以控制存放器的输入、输出，选择运算器的运算功能，存储器的读写。微程序控制器由微程序给出 24 位控制信号，而微程序的地址又是由指令码提供的，也就是说 24 位控制信号是由指令码确定的。该模型机的微指令的长度为 24 位，其中微指令中只含有微命令字段，没有微地址字段。**其中微命令字段采用直接按位的表示法，哪位为 0，表示选中该微操作，而微程序的地址那么由指令码指定。**这 24 位操作控制信号的功能如表 2 所示：〔按控制信号从左到右的顺序依次说明〕

表 2　微指令控制信号的功能

| 操作控制信号 | 控 制 信 号 的 说 明 |
|---|---|
| XRD | 外部设备读信号，当给出了外设的地址后，输出此信号，从指定外设读数据。 |
| EMWR | 程序存储器 EM 写信号。 |
| EMRD | 程序存储器 EM 读信号。 |
| PCOE | 将程序计数器 PC 的值送到地址总线 ABUS 上。 |
| EMEN | 将程序存储器 EM 与数据总线 DBUS 接通，由 EMWR 和 EMRD 决定是将 DBUS 数据写到 EM 中，还是从 EM 读出数据送到 DBUS。 |

| | |
|---|---|
| IREN | 将程序存储器 EM 读出的数据打入指令存放器 IR 和微指令计数器 μPC。 |
| EINT | 中断返回时去除中断响应和中断请求标志，便于下次中断。 |
| ELP | PC 打入允许，与指令存放器的 IR3、IR2 位结合，控制程序跳转。 |
| MAREN | 将数据总线 DBUS 上数据打入地址存放器 MAR。 |
| MAROE | 将地址存放器 MAR 的值送到地址总线 ABUS 上。 |
| OUTEN | 将数据总线 DBUS 上数据送到输出端口存放器 OUT 里。 |
| STEN | 将数据总线 DBUS 上数据存入堆栈存放器 ST 中。 |
| RRD | 读存放器组 R0~R3，存放器 R?的选择由指令的最低两位决定。 |
| RWR | 写存放器组 R0~R3，存放器 R?的选择由指令的最低两位决定。 |
| CN | 决定运算器是否带进位移位，CN=1 带进位，CN=0 不带进位。 |
| FEN | 将标志位存入 ALU 内部的标志存放器。 |
| X2 | X2、X1、X0 三位组合来译码选择将数据送到 DBUS 上的存放器。 |
| X1 | |
| X0 | |
| WEN | 将数据总线 DBUS 的值打入工作存放器 W 中。 |
| AEN | 将数据总线 DBUS 的值打入累加器 A 中。 |
| S2 | S2、S1、S0 三位组合决定 ALU 做何种运算。 |
| S1 | |
| S0 | |

COP2000 中有 7 个存放器可以向数据总线输出数据，但在某一特定时刻只能有一个存放器输出数据. 由 X2,X1,X0 决定那一个存放器输出数据。

| X2 X1 X0 | 输出存放器 |
|---|---|
| 0 0 0 | IN_OE 外部输入门 |
| 0 0 1 | IA_OE 中断向量 |
| 0 1 0 | ST_OE 堆栈存放器 |
| 0 1 1 | PC_OE PC 存放器 |
| 1 0 0 | D_OE 直通门 |
| 1 0 1 | R_OE 右移门 |
| 1 1 0 | L_OE 左移门 |
| 1 1 1 | 没有输出 |

COP2000 中的 ALU 由一片 CPLD 实现. 有 8 种运算, 通过 S2, S1, S0 来选择。运算数据由存放器 A 及存放器 W 给出，运算结果输出到直通门 D。

| S2 S1 S0 | 功能 |
|---|---|
| 0 0 0 | A+W 加 |
| 0 0 1 | A-W 减 |
| 0 1 0 | A\|W 或 |
| 0 1 1 | A&W 与 |
| 1 0 0 | A+W+C 带进位加 |
| 1 0 1 | A-W-C 带进位减 |
| 1 1 0 | ~A A 取反 |
| 1 1 1 | A 输出 A |

**2. 计算机中实现乘法和除法的原理**

**〔1〕无符号乘法**

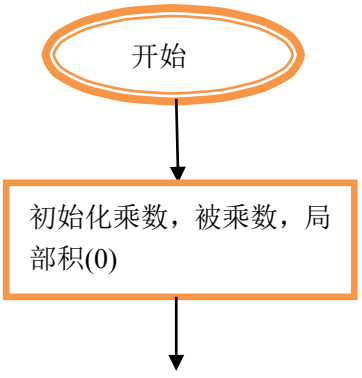
**①实例演示〔4 位乘法具体例子演算的算式〕：**

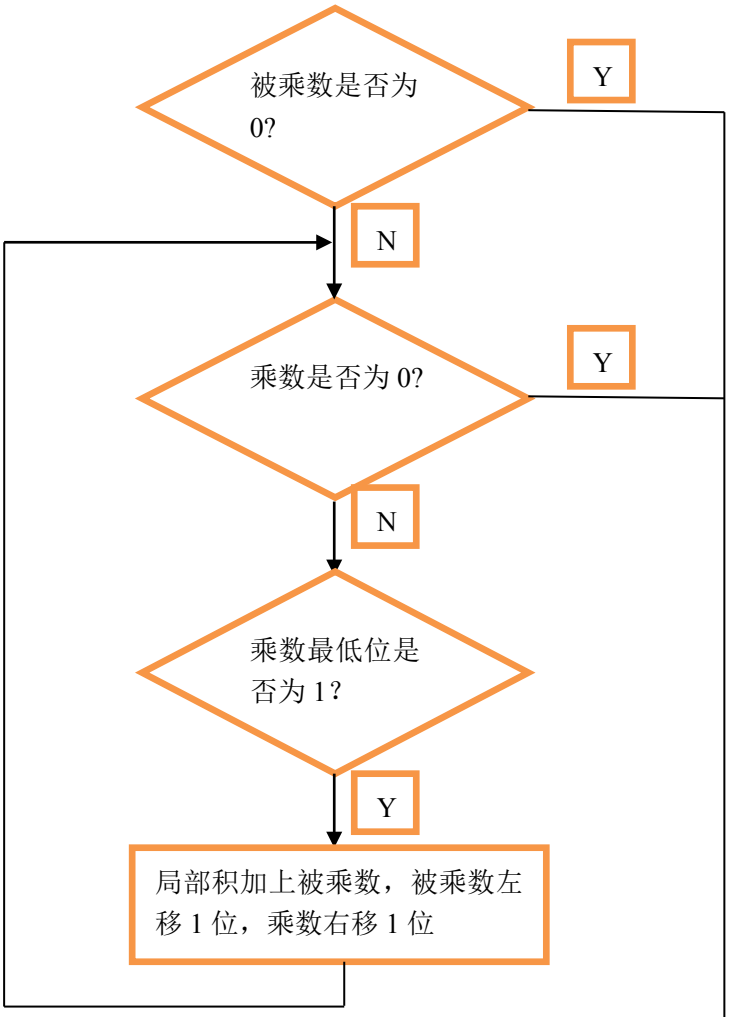
**乘数与被乘数假设为 1100〔12〕与 1000〔8〕，结果应该为 96〔十进制〕。**

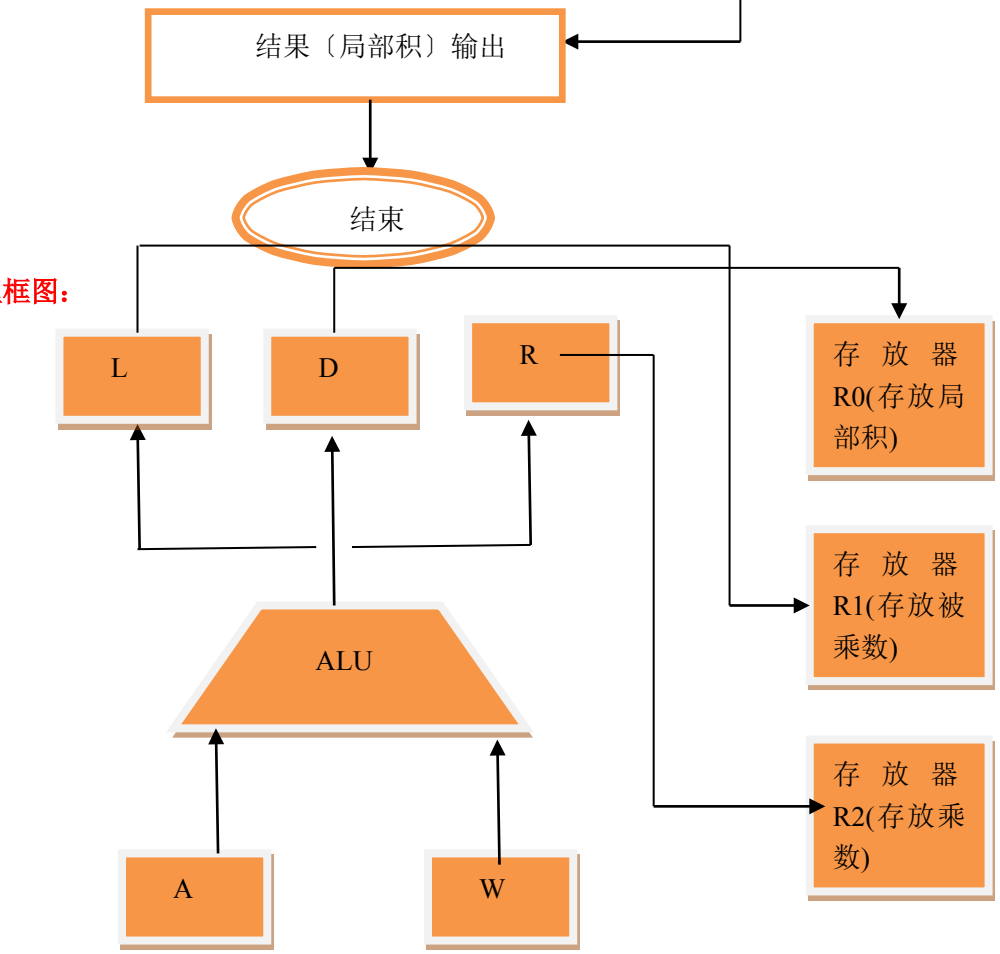
**运算图示为：**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 1 | 0 | 0 | | | | 被乘数 |
| × | | | | 1 | 0 | 0 | 0 | | | | 乘数 |
| | | | | 0 | 0 | 0 | 0 | | | | 初始局部积 |
| + | | | | 0 | 0 | 0 | 0 | | | | 乘数最低位为 0，局部积加 0，被乘数左移一位，乘数右移一位。 |
| | | | | 0 | 0 | 0 | 0 | | | | |
| + | | | 0 | 0 | 0 | 0 | | | | | 情况同上 |
| | | | 0 | 0 | 0 | 0 | 0 | | | | |
| + | | 0 | 0 | 0 | 0 | | | | | | 情况同上 |
| | | 0 | 0 | 0 | 0 | 0 | 0 | | | | |
| + | 1 | 1 | 0 | 0 | | | | | | | 乘数最低位为 1，局部积加被乘数，被乘数左移一位，乘数右移一位 |
| 〔0〕 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | | | | 计算完毕，运算结果为 01100000〔96〕 |

**算法流程图：**




被乘数是否为0?
Y
N
乘数是否为 0?
Y
N
乘数最低位是否为 1?
Y
局部积加上被乘数，被乘数左移 1 位，乘数右移 1 位


结果〔局部积〕输出
结束
L
D
R
存 放 器 R0(存放局部积)
存 放 器 R1(存放被乘数)
存 放 器 R2(存放乘数)
ALU
A
W

**硬件原理框图：**

相关说明： 将 R1 打入 A 中，R0 存放的为局部积，局部积初值为 0，假设乘数最低位为 1，之后被乘数与局部积通过 ALU 加和，结果存于 R0 中。

由于上一步，R1(即被乘数)已在 A 中，所以直接通过对 X2X1X0 的控制可实现 A 的逻辑左移 1 位。

之后将 R2〔乘数〕打入 A 中，通过对 X2X1X0 的控制可实现 A 的逻辑右移 1 位 。

期间有判断乘数、被乘数是否为 0 的操作。

〔2〕无符号除法

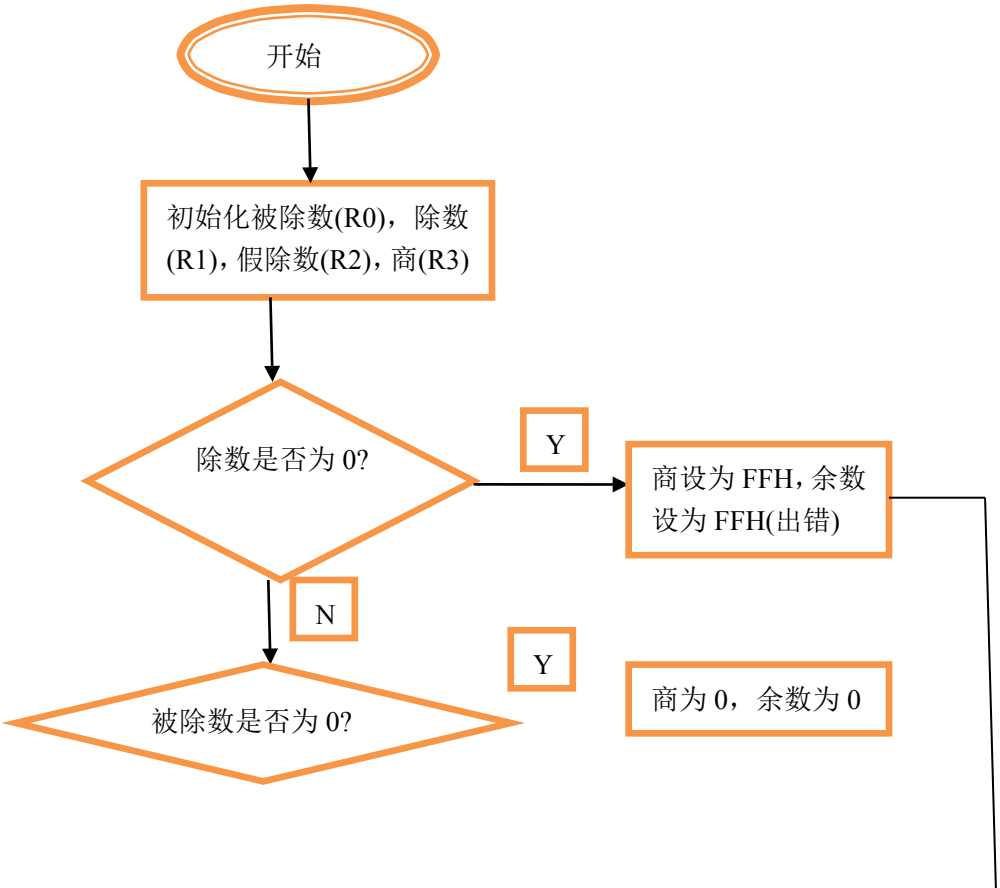
实例演示〔8 位被除数，4 位除数，具体例子演算的算式〕：

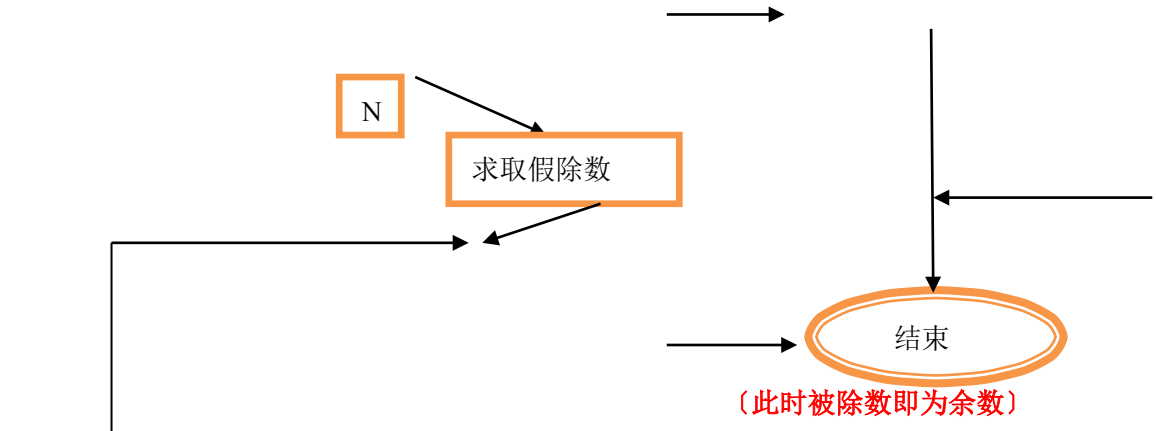
〔86 除以 10〕：

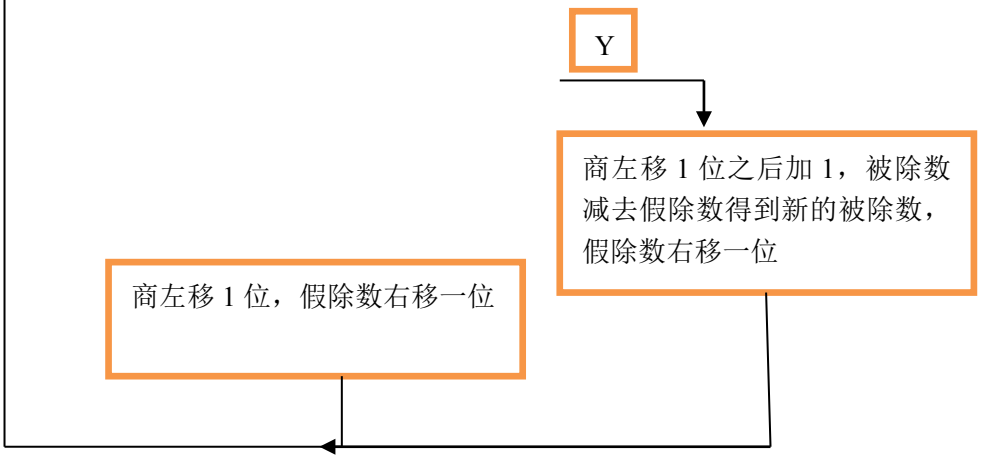
| | | |
|---|---|---|
| 商 | 0 | 初始商 |
| 1 0 1 0 | 0 1 0 1 0 1 1 0 | 除数与被除数 |
| | 1 0 1 0 0 0 0 0 | 由除数初始化假除数，此处将除数左移 4 位即可，其他情况需要另外考虑移位数 |
| | 0 1 0 1 0 1 1 0 | 判断被除数与假除数的关系，小于假除数 |
| 〔0〕 | 0 1 0 1 0 0 0 0 | 假除数右移一位，商左移一位 |
| | 0 0 0 0 0 1 1 0 | 被除数大于假除数，相减产生新的被除数 |
| 〔01〕 | 0 0 1 0 1 0 0 0 | 假除数右移一位，商左移一位并加一 |
| | 0 0 0 0 0 1 1 0 | 被除数小于假除数 |
| 〔010〕 | 0 0 0 1 0 1 0 0 | 假除数右移一位，商左移一位 |
| | 0 0 0 0 0 1 1 0 | 被除数小于假除数 |
| 〔0100〕 | 0 0 0 0 1 0 1 0 | 假除数右移一位，商左移一位 |
| | 0 0 0 0 0 1 1 0 | 被除数小于假除数 |
| 〔01000〕 | 0 0 0 0 0 1 0 1 | 假除数右移一位，商左移一位 |
| | | 此时假除数小于除数，算法结束 |

此时商为 01000 余数为 00000110。

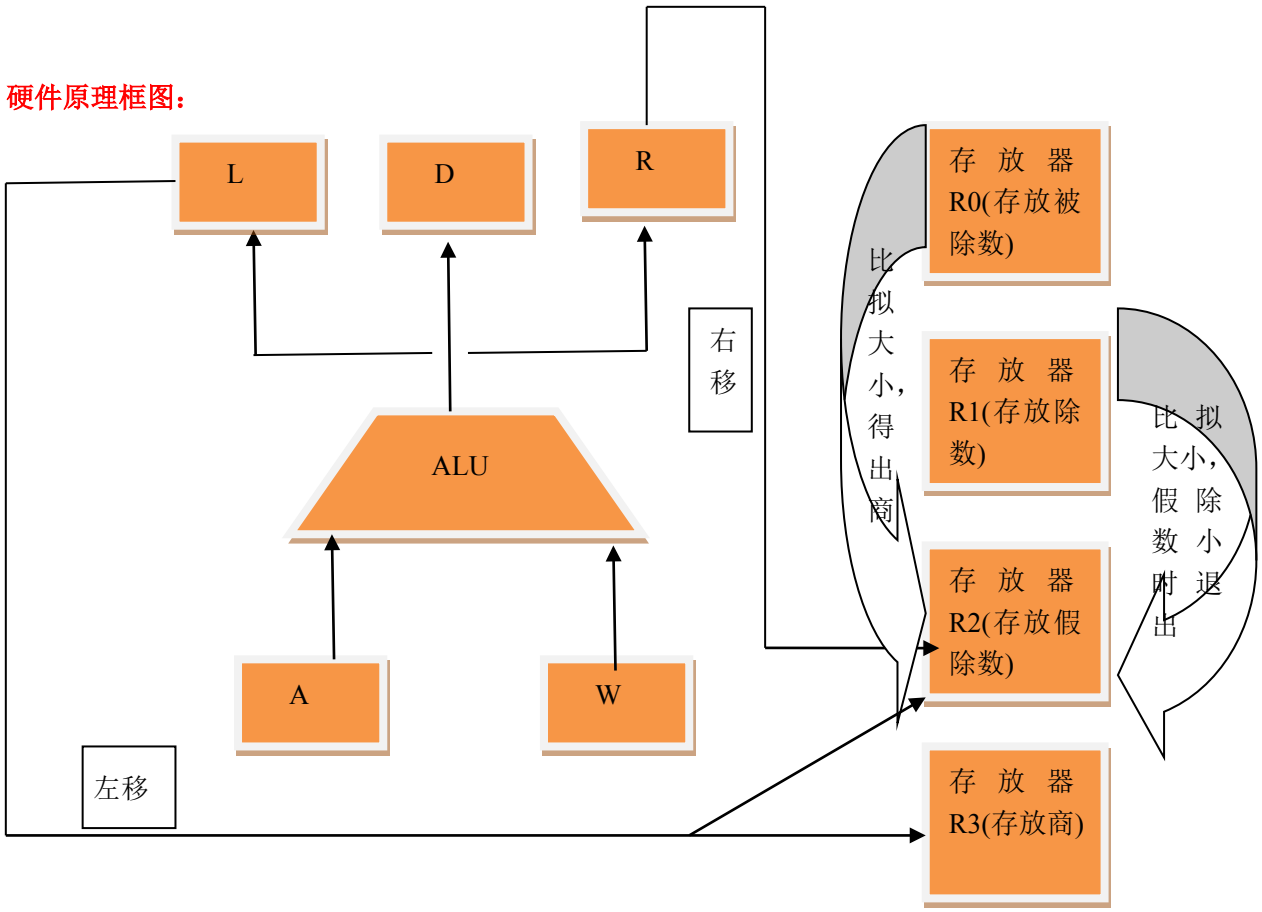
算法流程图：




N
求取假除数
结束
〔此时被除数即为余数〕


Y
商左移 1 位之后加 1，被除数减去假除数得到新的被除数，假除数右移一位
商左移 1 位，假除数右移一位

**硬件原理框图：**


L
D
R
存 放 器 R0(存放被除数)
比拟大小，得出商
右移
存 放 器 R1(存放除数)
比 拟 大小，假 除 数 小 时 退 出
ALU
存 放 器 R2(存放假除数)
A
W
左移
存 放 器 R3(存放商)

**退出时，余数存于 R0 中，商存于 R3 中**

**3．对应于以上算法如何分配使用 COP2000 实验仪中的硬件**

**〔初步分配，设计完成后再将准确的使用情况填写在此处〕**

符号乘法对应于 COP2000 实验仪的硬件具体分配使用情况如下表所示：

表　无符号乘法的硬件分配情况

| 硬件名称 | 实现算法功能描述 |
|---|---|
| 存放器 R0 | 计算时用来存放局部积和最后的积 |
| 存放器 R1 | ① 初始化时，用来存放被乘数；<br>② 在程序执行的过程中，用来存放向左移位后的被乘数。 |
| 存放器 R2 | ① 初始化时，用来存放乘数；<br>② 在程序执行的过程中，用来存放向右移位后的乘数。 |
| 累加器 A | 执行 ADD R?,A〔加法〕、SHL R?〔左移一位〕、SHR R?〔右移一位〕等命令时所必须使用的存放器。 |
| 存放器 W | 执行 ADD R?,A〔加法〕、TEST R?,#II〔测试与〕等双操作数命令时所必须使用的存放器。 |
| 左移门 L | 用来实现相应数据左移一位的运算，并能够控制该运算后的结果是否输出到数据总线。 |
| 直通门 D | 用来控制 ALU 的执行结果〔未经移位〕是否输出到数据总线。 |
| 右移门 R | 用来实现相应数据右移一位的运算，并能够控制该运算后的结果是否输出到数据总线。 |
| 程序计数器 PC | ① 控制程序按顺序正常执行；<br>② 当执行转移指令时，从数据线接收要跳转的地址，使程序能够按需要自动执行。<br>③ 当要从 EM 中读取数据时，由 PC 提供地址。 |
| 存储器 EM | 存储指令和数据。 |
| 微程序计数器 μPC | 向微程序存储器 μM 提供相应微指令的地址。 |
| 微程序存储器 μM | 存储相应指令的微指令，微程序。 |
| 输出存放器 OUT | 可以将运算结果输出到输出存放器 OUT〔本实验未用〕。 |
| 堆栈 ST | 可以用来暂存操作数或者对存放器值进行保护〔本实验未用〕 |

表　无符号除法的硬件分配情况

| 硬件名称 | 实现算法功能描述 |
|---|---|
| 存放器 R0 | 存放被除数，最后余数所在地 |
| 存放器 R1 | 初始化时，用来存放除数； |
| 存放器 R2 | ① 初始化时，用来存放除数；<br>② 存放移位后的假除数；<br>③ 在程序执行的过程中，用来存放向右移位后的假除数。 |

| | |
|---|---|
| 存放器 R3 | 用来存放商； |
| 累加器 A | 执行 SUB R?,A〔减法〕、SHL R?〔左移一位〕、SHR R?〔右移一位〕等命令时所必须使用的存放器。 |
| 存放器 W | 执行 SUB R?,A〔加法〕、TEST R?,#II〔测试与〕等双操作数命令时所必须使用的存放器。 |
| 左移门 L | 用来实现相应数据左移一位的运算，并能够控制该运算后的结果是否输出到数据总线。 |
| 直通门 D | 用来控制 ALU 的执行结果〔未经移位〕是否输出到数据总线。 |
| 右移门 R | 用来实现相应数据右移一位的运算，并能够控制该运算后的结果是否输出到数据总线。 |
| 程序计数器 PC | ① 控制程序按顺序正常执行；<br>② 当执行转移指令时，从数据线接收要跳转的地址，使程序能够按需要自动执行。<br>③ 当要从 EM 中读取数据时，由 PC 提供地址。 |
| 存储器 EM | 存储指令和数据。 |
| 微程序计数器 μPC | 向微程序存储器 μM 提供相应微指令的地址。 |
| 微程序存储器 μM | 存储相应指令的微指令，微程序。 |
| 输出存放器 OUT | 可以将运算结果输出到输出存放器 OUT〔本实验未用〕。 |
| 堆栈 ST | 用来暂存操作数或者对存放器值进行保护〔本实验未用〕 |

**4．在 COP2000 集成开发环境下设计全新的指令/微指令系统**

**设计结果如表所示〔可按需要增删表项〕**

**（1） 新的指令集**

**〔如果针对乘除法设计了两个不同指令集要分别列表〕**

**乘、除法共用一个指令集：**

| 助记符 | 机器码 1 | 机器码 2 | 指令说明 |
|---|---|---|---|
| _FATCH_ | 000000xx 00-03 | | 实验机占用，不可修改 |
| MOV R?,#II | 000001xx 04-07 | II | |
| MOV A,R? | 000010xx 08-0B | | |
| TEST R?,#II | 000011xx 0C-0F | II | |
| ADD R?,A | 000100xx 10-13 | | |
| ADD R?,#II | 000101xx 14-17 | II | |
| JMP MM | 000110xx 18-1B | MM | |
| OK | 000111xx 1C-1F | | |
| JC MM | 001000xx 20-23 | MM | |
| SHL R? | 001001xx 24-27 | | |
| SHR R? | 001010xx 28-2B | | |
| | 001011xx 2C-2F | | 未使用 |
| | 001100xx 30-33 | | 未使用 |
| JZ MM | 001101xx 34-37 | MM | |
| SUB R?,A | 001110xx 38-3B | | |

| | | | |
|---|---|---|---|
| CMP R?,A | 001111xx 3C-3F | | |
| | | | 未使用 |

（2） 新的微指令集

| 助记符 | 状态 | 微地址 | 微程序 | 数据输出 | 数据打入 | 地址输出 | 运算器 | 移位控制 | μPC | PC |
|---|---|---|---|---|---|---|---|---|---|---|
| _FATCH_ | T0 | 00 | CBFFFF | 浮空 | 指令存放器 IR | PC 输出 | A 输出 | | 写入 | +1 |
| MOV R?,#II | T1 | 04 | C7FBFF | 存储器值 EM | 存放器 R? | PC 输出 | A 输出 | | +1 | +1 |
| | T0 | 05 | CBFFFF | 浮空 | 指令存放器 IR | PC 输出 | A 输出 | | 写入 | +1 |
| MOV A,R? | T0 | 08 | CBF7F7 | R? | 存放器 A,指令存放器 IR | PC 输出 | A 输出 | | 写入 | +1 |
| TEST R?,#II | T2 | 0C | C7FFEF | 存储器值 EM | 存放器 W | PC 输出 | A 输出 | | +1 | +1 |
| | T1 | 0D | FFF7F7 | R? | 存放器 A | 浮空 | A 输出 | | +1 | |
| | T0 | 0E | CBFE9B | ALU 直通 | 指令存放器 IR，标志位 C,Z | PC 输出 | 与运算 | | 写入 | +1 |
| ADD R?,A | T2 | 10 | FFF7EF | R? | 存放器 W | 浮空 | A 输出 | | +1 | |
| | T1 | 11 | FFFA98 | ALU 直通 | 存放器 R? 标志位 C,Z | 浮空 | 加运算 | | +1 | |
| | T0 | 12 | CBFFFF | 浮空 | IR | PC 输出 | A 输出 | | 写入 | +1 |
| ADD R?,#II | T3 | 14 | C7FFEF | 存储器值 EM | 存放器 W | PC 输出 | A 输出 | | +1 | +1 |
| | T2 | 15 | FFF7F7 | R? | 存放器 A | 浮空 | A 输出 | | +1 | |
| | T1 | 16 | FFFA98 | ALU 直通 | 存放器 R? 标志位 C,Z | 浮空 | 加运算 | | +1 | |
| | T0 | 17 | CBFFFF | 浮空 | IR | PC 输出 | A 输出 | | 写入 | +1 |
| JMP MM | T1 | 18 | C6FFFF | 存储器值 EM | 存放器 PC | PC 输出 | A 输出 | | +1 | 写入 |
| | T0 | 19 | CBFFFF | 浮空 | IR | PC 输出 | A 输出 | | 写入 | +1 |
| OK | T0 | 1C | CBFFFF | 浮空 | IR | PC 输出 | A 输出 | | 写入 | +1 |