组合逻辑控制器

前言：计算机各个部件所需要的微操作控制信号均由控制器的“操作控制信号形成部件”（操作控制器）根据指令的要求产生。而“操作控制信号形成部件”的实现有两种方式：一种是采用组合逻辑来实现，即“组合逻辑控制器”；另一种是采用存储逻辑来实现，即“微程序控制器”。

1. 组合逻辑控制器原理

组合逻辑控制器（硬布线控制器）的基本原理是根据指令的要求、当前的时序及外部和内部的状态情况，按时间顺序发送一系列微操作控制信号。它由复杂的组合逻辑门电路和一些触发器构成，是早期计算机控制器的一种设计方法，一旦确定，即不可更改和扩充，这一点非常不利于控制器的设计和调试。但随着RISC技术和VLSI技术的飞速发展，硬布线控制技术又和微程序控制技术共同应用于CPU设计中。

1. 组合逻辑控制器的组成

程序计数器PC、指令寄存器IR、指令译码器、脉冲源和启停控制逻辑与微程序控制器相同，不同的是微程序控制器中的控制存储器在组合逻辑控制器中变成了时序控制信号形成部件，是用组合逻辑的门电路实现的。

1. 程序计数器PC

用来存放当前正在执行的指令地址或者下一条指令的地址。当执行一条指令时，首先需要根据PC中存放的指令地址，将指令由内存取到指令寄存器中，此过程称为“取指令”。与此同时，PC中的地址或自动加1或由转移指针给出下一条指令的地址或在控制器控制下由运算器形成转移地址，转移地址送往程序计数器PC作为下一条指令的地址。此后经过分析指令，执行指令。完成第一条指令的执行，而后根据PC取出第二条指令的地址，如此循环，执行每一条指令。当计算机重启或复位时，程序计数器通常恢复到零。程序计数器工作过程

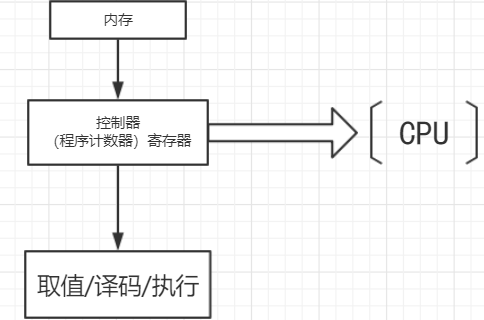
如图-1所示：

图-1

1. 指令寄存器

指令寄存器(IR, Instruction Register)，是临时放置从内存里面取得的程序指令的寄存器，用于存放当前从主存储器读出的正在执行的一条指令。

当执行一条指令时，先把它从内存取到数据寄存器（DR，Data Register）中，然后再传送至指令寄存器。指令划分为操作码和地址码字段，由二进制数字组成。为了执行任何给定的指令，必须对操作码进行测试，以便识别所要求的操作。指令寄存器的相应位送入指令译码器，根据译码结果产生相应的控制信号，从而完成指令规定的运算、传送数据等操作。

取指过程：

1.现行指令地址送至存储器地址寄存器，记作PC→MAR

2.向主存发送读命令，启动主存读操作，记作1→R

3.将MAR（通过地址总线）所指的主存单元中的内容（指令）经数据总线读至MDR内，记作M（MAR）→MDR

4.将MDR的内容送至IR，记作MDR→IR

5.指令的操作码送至CU译码，记作OP（IR）→CU

6.形成下一条指令的地址，记作（PC）+1→PC

为了提高计算机的并行操作能力和运行速度，目前大多数计算机都将指令寄存器扩充为指令队列，或称指令栈，允许预取多条指令。

1. 指令译码器

指令译码器是控制器的主要部件之一。指令由操作码和地址码组成，操作码表示要执行的操作性质，即执行什么操作；地址码是操作码执行时的操作对象的地址，计算机在执行一条指令时，必须首先由指令译码器来分析操作码，以确定执行什么样的操作。将指令寄存器中的操作码字段输入指令译码器，操作码经过译码后的信号与操作控制信号形成部件一起产生该指令所需要的、有一定时序关系的操作控制信号序列。

译码器是组合逻辑电路（硬布线电路）的一个重要部件，分为变量译码和显示译码两类。

变量译码：一般是一种较少输入变为较多输出的部件，一般分为2n译码和8421BCD码译码两类。

显示译码：主要解决二进制数显示成对应的十或十六进制数的转换功能，一般可分为驱动LED和驱动LCD两类。

1. 操作控制信号形成部件

在硬布线控制器中，操作控制信号形成部件是采用组合逻辑电路实现的。该部件的功能是根据指令操作码的译码信号以及时序信号，产生取出指令和执行这条指令所需的各种操作控制信号，以便正确建立数据通路，完成取出指令和执行指令的控制。

1. 数据寄存器

数据寄存器是用来暂时存放计算过程中所用到的操作数、结果和信息。

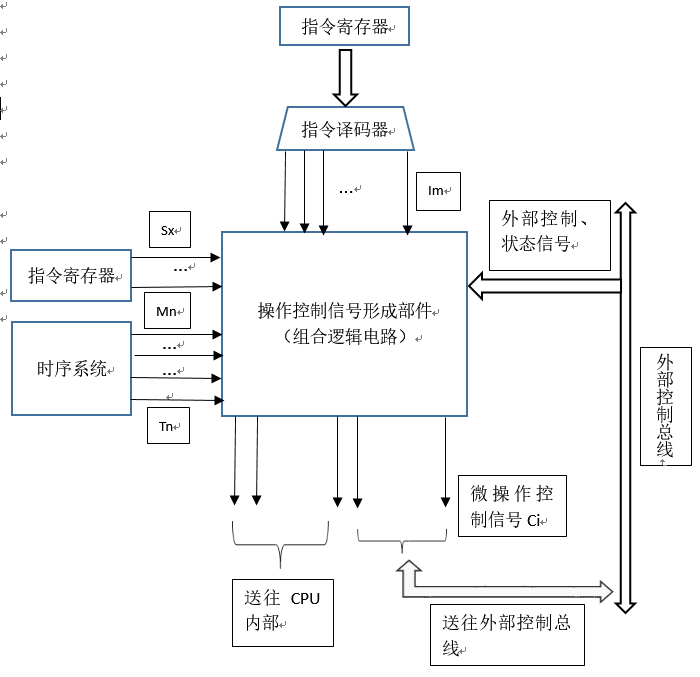
AX, BX, CX, DX可以称为数据寄存器，这4个16位数据寄存器可分为高8位和低8位，因此既可以作为4个16位的数据寄存器使用，也可以作为8个8位的数据集群器使用。在编程时可以存放源操作数、目的操作数或运算结果，以减少存储器访问次数。

1. 状态寄存器

状态寄存器又名条件码寄存器，用来存放两类信息：一类是体现当前指令执行结果的各种状态信息（条件码），如有无进位（CF位）、有无溢出（OF位）、结果正负（SF位）、结果是否为零（ZF位）、奇偶标志位（P位）等；另一类是存放控制信息（PSW：程序状态字寄存器），如允许中断（IF位）、跟踪标志（TF位）等。

1. 时序系统

产生指令系统的全部指令的各种机器周期信号和节拍信号，其中机器周期信号序列取决于当前执行的指令，即与指令译码器的输出有关，而时钟周期则由时序电路产生固定的若干T周期即可。

硬布线控制器的内部组成如图-2所示：  
图-2 硬布线控制器的组成

从图-2中可以看出，硬布线控制器的核心部件是由组合逻辑电路构成的操作控制信号形成部件，又称为操作控制器。操作控制器有4种信号输入：

* 1. 经指令译码器译码产生的指令信息Im，用于指出当前是哪一条指令的指令周期。
  2. 时序系统产生的机器周期信号Mn和节拍信号Tn，用于指出当前处于哪一个机器周期和哪一个节拍。
  3. 状态寄存器的状态信号Sx，用于指出运算器的结果状态及机器内部的其他状态，以决定某些操作信号是否发送。例如在条件转移指令中，根据条件决定是否进行转移。
  4. 外部控制信号Ej，用于指出和传递CPU外部各部件的状态和控制信号，例如存储器和外设的“准备好”信号和中断请求信号等。

操作控制器的功能就是根据这4种信号，输出全机所需要的、带有时间顺序特性的微操作控制信号Ci序列，一部分送到CPU外部构成系统总线的控制总线，例如存储器和外设的读/写访问控制信号；另一部分送到CPU内部以供使用，例如运算器的功能控制信号和各寄存器的控制信号。因此，从逻辑函数的角度看，输出微操作控制信号Ci是4种输入信号的函数：

Ci = fi(Im,Mn,Tn,Sx,Ej)

所以，设计硬布线控制器的过程，也就是求出每个微操作控制信号Ci的逻辑函数fi的过程。

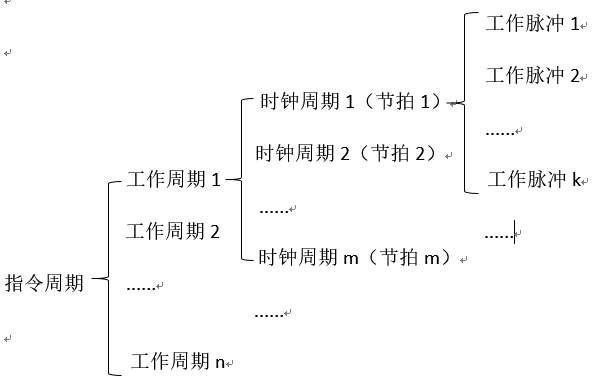
1. 组合逻辑控制器的设计步骤和控制方法

3.1从整个CPU的设计方法的角度来讨论组合逻辑控制器（硬布线控制器）的设计步骤。

1. 确定指令系统，包括指令系统中每条指令的格式、功能和寻址方式，分配操作码。
2. 围绕指令系统的实现，确定CPU的内部结构，包括运算器的功能和组成，控制器的类型、结构组成及各部件的连接方式和数据通路，同时也需要确定时序系统的构成。
3. 在以上基础上，分析每条指令的执行过程，按机器周期顺序，写出所必须发送的微操作控制信号序列。
4. 综合每个微操作控制信号的逻辑函数，并化简和优化。
5. 用逻辑电路实现。

3.2

组合逻辑控制方式的控制器逻辑一般包括了安排时序、拟定指令流程和微命令序列、形成控制逻辑等。

组合逻辑控制器的时序划分一般分为三级时序系统，如图-3所示：  
图-3 三级时序系统

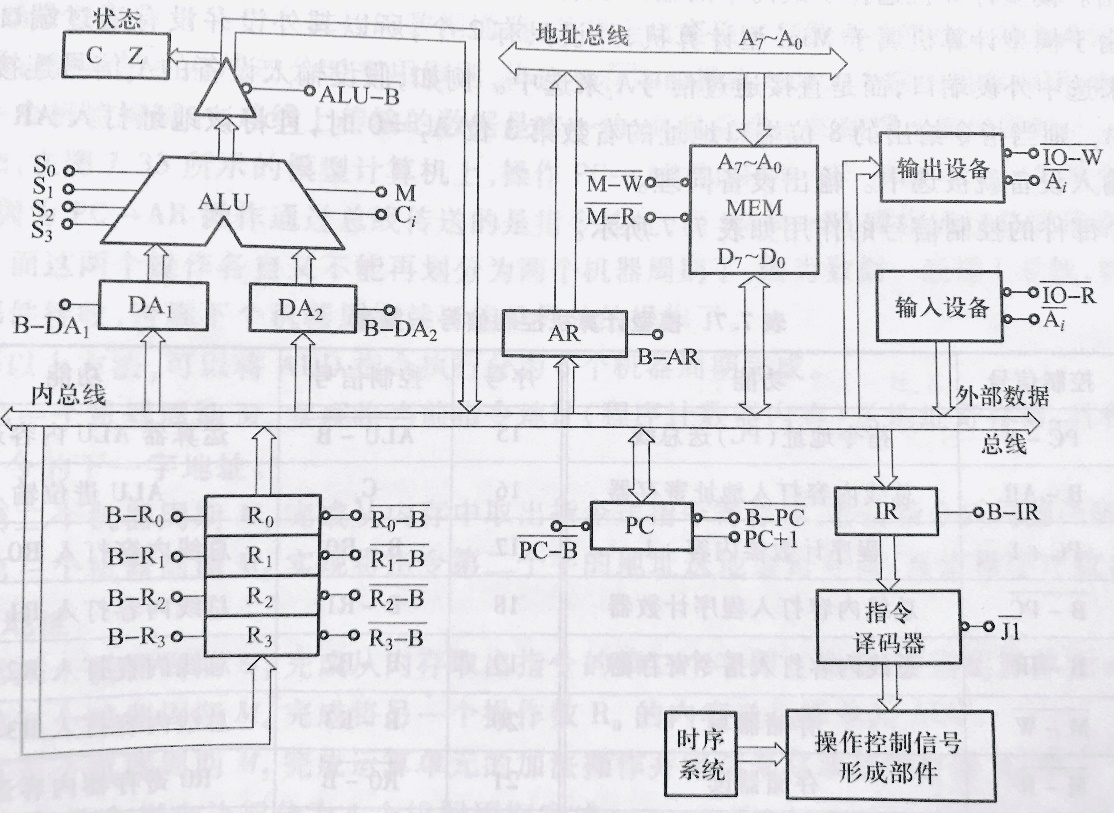
我们以图-4所示的模拟机框图和两条指令（ADD和JMP指令）的执行为例来介绍组合逻辑控制器的具体的控制方法。

图-4 模型机的系统结构框图

1. 指令系统

该模型机CPU要实现一个包含两条指令的指令系统，指令格式及操作码如表-1所示。

表-1 ADD指令和JMP指令的格式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 助记符 | 格式 | 操作码OP | 指令机器码 | 指令功能 |
| ADD R0, 06H | |  |  |  | | --- | --- | --- | | OP | ×× | DR | | 立即数 | | | | 0101 | |  |  | | --- | --- | | 0101 | 0000 | | 0000 0110 | | | (R0) + 06H -> R0 |
| JMP 04H | |  |  | | --- | --- | | OP | ×××× | | 转移地址 | | | | 1000 | |  |  | | --- | --- | | 1000 | 0000 | | 0000 0100 | | | 04H -> PC |

从表中可以看出，ADD指令和JMP指令均为二字长指令。ADD指令的第一个字为操作码和寄存器地址，第二个字是立即数。也就是说，ADD指令的一个操作数为立即数寻址，另一个操作数即目的操作数为寄存器寻址（暂定为R0）。JMP指令的第一个字是操作码，第二个字是转移的直接地址。

通过图-4我们可以看到，该模型计算机字长为8位，CPU内部属于单总线结构，运算器和控制器的各部件通过各种方式挂在内总线上。并且内总线和外部系统总线的数据总线合二为一，所以CPU外的其他部件，如存储器、输入设备、输出设备等也全部挂在同一条总线上。从图中我们可以看出控制器是由程序计数器PC、指令寄存器IR、地址寄存器AR、指令译码器、操作控制信号形成部件、时序系统构成。其中，PC可以直接与内总线交换数据，还可以自动增1；AR从内部总线上输入数据，并直接输出到外部地址总线。

图-4中各个部件的控制信号的作用如表-2所示：

表-2 模型机控制信号一览表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 控制信号 | 功能 | 序号 | 控制信号 | 功能 |
| 1 |  | 指令地址PC送总线 | 15 |  | 运算器ALU内容送总线 |
| 2 | B - AR | 总线内容打入地址寄存器 | 16 | Ci | ALU进位输入 |
| 3 | PC + 1 | 程序计数器内容+1 | 17 | B – R0 | 总线内容打入R0寄存器 |
| 4 |  | 总线内容打入程序计数器 | 18 | B – R1 | 总线内容打入R1寄存器 |
| 5 | B - IR | 总线内容打入指令寄存器 | 19 | B – R2 | 总线内容打入R2寄存器 |
| 6 |  | 存储器写 | 20 | B – R3 | 总线内容打入R3寄存器 |
| 7 |  | 存储器读 | 21 |  | R0寄存器内容送总线 |
| 8 | S3 | S3~S0选择ALU 16种运算之一 | 22 |  | R1寄存器内容送总线 |
| 9 | S2 | 同上 | 23 |  | R2寄存器内容送总线 |
| 10 | S1 | 同上 | 24 |  | R3寄存器内容送总线 |
| 11 | S0 | 同上 | 25 |  | 写I/O端口（输出） |
| 12 | M | M=1，ALU做逻辑运算  M=0，ALU做算术运算 | 26 |  | 读I/O端口（输入） |
| 13 | B – DA1 | 总线内容打入暂存器DA1 | 27 | ‾Ai | 端口地址线（选中IO） |
| 14 | B – DA2 | 总线内容打入暂存器DA2 | 28 | ‾J1 | 指令译码器工作 |

1. 指令执行过程

在设计控制器时，需要把执行过程中的各操作划分为一个个有序的机器周期（或一条条微指令），从而构成一个指令周期。那么，应该如何来设计控制器的控制方法呢？首先，必须在CPU的系统结构基础上，依据指令功能设计出指令执行的数据通路，并规定出实现这些数据通路所必须发送的微操作控制信号；然后，按照总线的分时享用原则，将这些有序的操作划分为不同的机器周期，也就是要保证在一个机器周期中，总线上传输的数据是唯一的，并且总线上的数据不能停留到下一个机器周期。例如在图-4所示模型机上，操作PC -> AR和RAM -> IR就必须划分为两个机器周期完成，因为PC -> AR操作通过总线传送的是指令地址，而RAM -> IR操作通过总线传送的是指令代码。而这两个操作各自又不能再划分为两个机器周期了，因为数据一经送上总线，就必须要被目的部件接收。

根据以上方法，可以将ADD指令执行分为6个机器周期完成。

①第一个机器周期M0实现将当前指令地址（程序计数器内容）送地址寄存器，且程序计数器指向指令的下一字地址。

②第二个机器周期M1完成从内存中取出指令送指令寄存器，并由指令译码器译码。

③第三个机器周期M2实现将指令第二字的地址送地址寄存器，且使程序计数器指向下一条指令地址。

④第四个机器周期M3完成从内存取出指令的第二字即立即数，并送运算单元ALU。

⑤第五个机器周期M4完成将另一个操作数R0的内容送运算单元ALU。

⑥第六个机器周期M5完成运算单元的加法操作并将结果存放在寄存器R0中。

同理，将JMP指令的执行分为4个机器周期完成。

①第一个和第二个机器周期完成与ADD指令一样的操作，也就是取指令的操作。

②第三个机器周期M2实现将指令的第二个字地址送地址寄存器，且使程序计数器指向下一条指令地址。

③第四个机器周期M3完成从内存取出指令的第二个字即转移地址送PC，实现转移。

接下来，我们按照下列方式给出ADD指令和JMP指令的执行过程。其中，冒号“：”左边为条件，右边为要执行的操作。

ADD指令：

ADD·M0 ：PC -> AR, PC + 1 -> PC；（取指令地址）

ADD·M1 ：RAM -> IR, 指令译码；（取指令并译码）

ADD·M2 ：PC -> AR, PC + 1 -> PC; （取指令的第二个字的地址）

ADD·M3 ：RAM ->DA1；（取数据到ALU的暂存器）

ADD·M4 ：R0 -> DA2;（送寄存器数据到ALU的暂存器）

ADD·M5 ：DA1 + DA2 –> R0; （计算并保存结果）

JMP指令：  
 JMP·M0 ：PC -> AR, PC + 1 -> PC；（取指令地址）

JMP·M1 ：RAM -> IR, 指令译码；（取指令并译码）

JMP·M2 ：PC -> AR, PC + 1 -> PC; （取指令的第二个字的地址）

JMP·M3 ：RAM -> PC; （取转移地址并执行转移）

对照图-4和表-2，可以写出对应的每个机器周期所必须发送的微操作控制信号序列。同上，冒号“：”左边为条件，右边为要执行的操作。

* 1. 取指令操作。

M0：, B – AR, PC + 1;

M1: , B – IR, ‾J1

* 1. ADD指令。

ADD·M2 ：, B – AR, PC + 1;

ADD·M3 ：, B – DA1；

ADD·M4 ：, B – DA2 ;

ADD·M5 ：ALUS3S2S1S0MCi(F = A + B), , B – R0;

此处选择S3S2S1S0MCi的值为控制ALU做算术加的运算，假设ALU是由两片74LS181串联而成的，则S3 S2 S1 S0 M Ci的值应为100101,即可以控制ALU 做 A + B的运算。

* 1. JMP指令

JMP·M2 ：, B – AR, PC + 1;

JMP·M3 ：, ;

1. 综合微操作控制信号的逻辑函数

接下来，需要对所有的微操作控制信号进行综合，即对于某一个微操作控制信号，将上述列表中凡是在冒号“：”右边出现该信号的，把其左边的条件（与项）作为一个或项，全部进行或运算，即得到该微操作控制信号的逻辑函数。

B – AR = M0 + ADD·M2 + JMP·M2

=

PC + 1 = M0 + ADD·M2 + JMP·M2

=

B – IR = M1

=

B – DA1 = ADD·M3

=

B – DA2 = ADD·M4

S3 = S0 = Ci = ADD·M5

S2 = S1 = M =

=

B – R0 = ADD·M5

=

在综合每个控制信号的逻辑函数时，需要注意以下几点。

①假设某个微操作控制信号必须在某个机器周期内的Tn时刻有效，则该信号表达式还要和Tn时钟周期信号相与。例如，假设信号必须在T2节拍发送，而B – IR信号必须在T3节拍发送，则他们的逻辑表达式就应该如下：  
 =

B – IR = M1·T3

②当指令系统被扩充时，按照上述方法写出新指令的机器周期序列和微操作控制信号序列，然后重新更新相关微操作控制信号的逻辑函数，即可能添加新的或项。

1. 对于已经确定的指令系统，要对各信号的逻辑函数进行化简、优化。这种化简不仅要从逻辑代数的角度来进行，还要从指令系统的整体逻辑关系上来考虑。例如，假设确定指令系统中只有ADD和JMP两个指令，则可以理解为ADD + JMP = 1, 那么以下4个信号就可以优化为如下表达式：  
    =

B – AR = M0 + M2

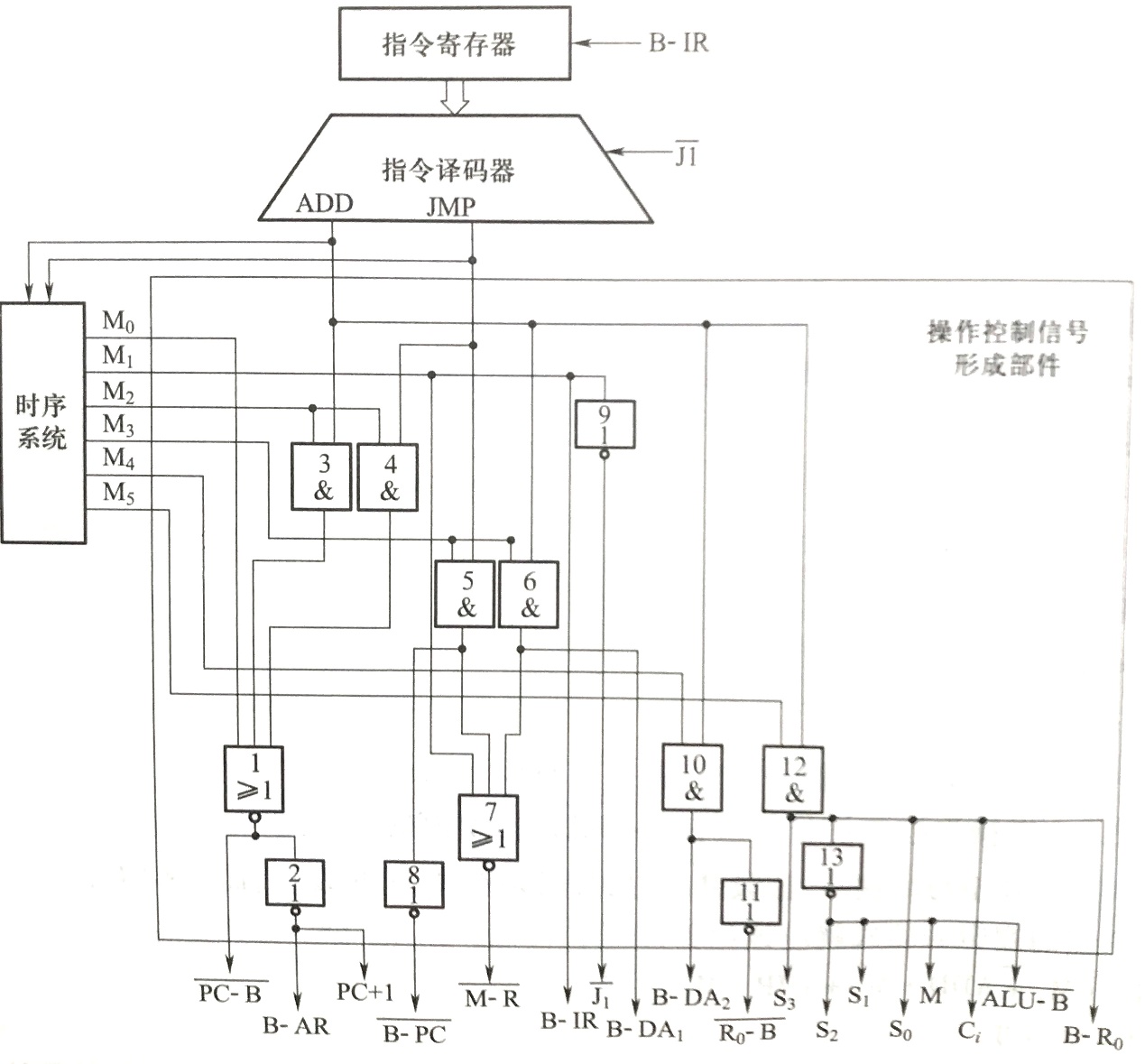
PC + 1 = M0 + M2

=

1. 逻辑实现

采用上述方法对逻辑函数进行优化和化简后，使得逻辑电路最简，用硬件电路实现即可。为实现上述两条指令的指令系统，按照上述步骤设计CPU操作控制信号形成部件，其电路图如图-5所示：

图-5 模型机硬布线控制器电路框图



控制过程：

1. 首先，开机上电后，硬件产生Reset信号，该信号使得PC置初值，即为第一条指令在内存中的地址；同时，时序电路开始工作，机器周期计数器被清零，即产生的第一个机器周期信号为M0
2. M0信号送入操作控制信号形成部件后，由图中的逻辑可知，驱动1号或非门输出0，则信号 = 0，B – AR = 1, PC + 1 = 1, 这些信号发送到相应的部件后，也就执行了送指令地址的操作：PC -> AR, PC + 1 -> PC。
3. M0机器周期结束后，进入M1机器周期，此时M1 = 1，同上，它驱动7号和9号门输出信号 = 0, B – IR = 1, = 0, 也即执行了取指令（读内存）到IR并译码的工作。
4. 指令译码器根据 IR中取到的指令操作码（此处为高4位），译码使得相应的指令信号线输出为有效（高电平），假设此时取到的是JMP指令，则JMP信号有效，送入时序系统后，时序电路按照JMP指令所需的机器周期序列，将会顺序产生M2, M3, M0
5. 由于JMP = 1，M2 = 1，表明此时进入JMP指令的M2机器周期，这两个信号驱动4号和1号、2号门，同样产生信号= 0, B – AR = 1, PC + 1 = 1，执行送指令（第二字）地址的操作：PC -> AR, PC + 1 -> PC
6. M2机器周期结束，进入M3机器周期，而JMP仍旧有效，则将驱动5号和7号、8号门，产生信号 = 0, = 0, 也就是从内存中读出指令第二字（转移地址）送入PC，实现转移。
7. 到这里之后，JMP指令周期已结束，按照JMP指令的机器周期序列，下一个机器周期是M0，即又进入取指令周期，回到②，如此循环往复，直至电源断电。