计算机组成原理复习纲要

徐卫霞　（第九、十章）

杜泽林　（第五、七、八章）

孙吉鹏　（第一、四、六章）

第一章 计算机系统概论

1. 冯诺伊曼计算机的特点　P8

1.计算机由运算器，存储器，控制器，输入设备和输出设备五部分组成

2.指令和数据以同等地位存放于存储器中，并可按地址寻访

3.指令和数据均由二进制数表示

4.指令由操作码和地址码构成，操作码用来表示操作的性质，地址码用来表示操作数在存储器的位置

5.指令在存储器内按顺序存放，通常是按顺序执行，特定条件下可根据运算结果或特定条件改变执行顺序

2. 计算机的工作步骤　P13 （介绍的很全面，串联整本书，适合复习结束看）
3. 计算机的主要技术指标　P17

   机器字长：CPU 一次能处理数据的位数，通常与 CPU 寄存器位数有关。

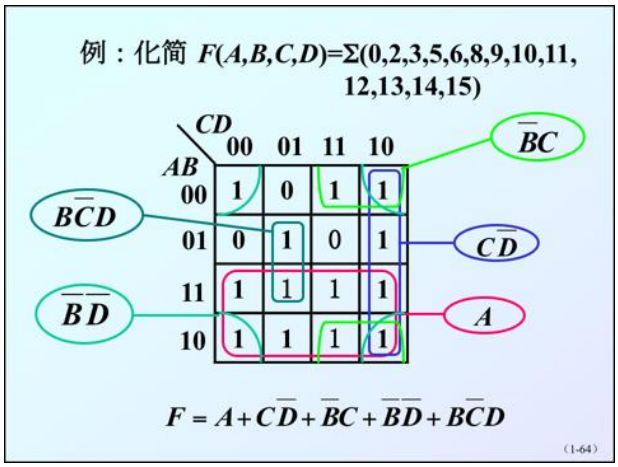
   存储容量：主存容量和辅存容量

   运算速度：与许多因素有关，主频，操作类型，主存速度等

补充章节：数字逻辑

1. 基本逻辑关系
   与（•）逻辑乘法
   或（+）逻辑加法
   非
2. 最小项
   定义： 在一个具有 n 个变量的逻辑函数中，如果一个与项包含了所有 n 个的变量，而且每个变量都是以原变量或反变量的形式作为一个因子仅出现一次，那么这样的与项就称为该逻辑函数的一个最小项。对于 n 个变量的全部最小项共有 2^n 个。
   最小项编号：为了表达方便，人们通常用 mi 表示最小项，其下标 i 为最小项的编号。编号的方法是：最小项中的原变量取 1，反变量取 0，则最小项取值为一组二进制数，其对应的十进制数便为该最小项的编号。
3. 卡诺图化简（详见上课时 数字逻辑 1 课件后半部分 一言难尽提一提重点）
   1. 逻辑相邻：相邻单元输入变量的取值只能有一位不同
   2. 若两个最小项中只有一个变量以原、反状态相区别，则称它们为逻辑相 邻。
      逻辑相邻的项可以合并，消去一个因子。
   3. 相邻单元的个数是 2^N 个，并组成矩形时，可以合并。
   4. 先找面积尽量大的组合进行化简，可以减少更多的因子。
   5. 各最小项可以重复使用。
   6. 所有的 1 都被圈过后，化简结束。
   7. 化简后的逻辑式是各化简项的逻辑和。
   8. 圈的数目越少越简；圈内的最小项越多越简。
   9. 卡诺图中所有的 1 都必须圈到，不能合并的 1 必须单独画圈。

   课件上例题：吉鹏智库不拥有此版权，向原作者致敬！



第四章 存储器

1. 存储器的分类 P68
   RAM？ROM？半导体存储器？主存？
2. 存储器的层次结构 P70
   缓存-主存 主存-辅存

为什么有这种结构？　　P71

3. 主存　P73
   字？字长？字节？字地址？字节地址？
4. 静态 RAM 和动态 RAM 的不同点　P77
   静态为触发器原理，读出后仍保持原状态，不需要再生，电源断电时丢失信息，做实验用的 MM214 就是静态存储器。
   动态为靠电容存储电荷原理，不断电也会丢失，需要刷新，比静态集成度高，能耗低，各类计算机广泛使用。但是存取速度慢于静态的。
5. 动态 RAM 的刷新　P86
   1. 集中刷新　规定的刷新周期内集中一段时间对所有存储单元逐行刷新，死时间连续易发生冲突。
   2. 分散刷新　每行的刷新分散到每个存取周期，但每个存储周期长了，拖慢了系统速度。
   3. 异步刷新　用最大刷新间隔除以单元行数算出每个存储单元行的最大刷新时间间隔，以这个间隔为刷新周期。降低了冲突的概率。
6. 半导体存储器的扩展　P91
   位扩展　增加存储字长，如让 2 个 1K*4 扩展为 1K*8，本质上就是让一个地址同时选中两片芯片，两片芯片分别存储这个数据的前后部分（高低位）。
   字扩展　增加存储器字的数量，如让 2 个 1K*4 扩展为 2K*4，本质上就是把译码前的信号当作地址扩展，译码片选后的信号控制多个芯片的使能端，使一个“地址”只选中一个芯片，因为这时的“地址”位数扩大了（多了片选译码的输入位），实现了存储字的扩展。

   字、位扩展
   结合起来，一个扩展地址通过译码片选出一个输出位，让这一位同时接两个位扩展芯片的使能端，实现字位扩展结合。
   好题：　例 4.1　例 4.2
   片选小技巧：
   1.如果三八译码器的三个输入地址没法满足片选地址的需求，如需要四位地址来区分各芯片，则考虑将第四位地址线和三八译码器的一条输出与起来，这样可以实现多位区分。
   2.害怕高位地址不做三八译码器的输入导致多选怎么办？把所有高位地址对应成三八译码器或访存的使能端就好。
7. Cache　　P109
   出现原因？（作用）主存 CPU 速度不匹配。可能保障？程序访问的局限性。
   命中率？Cache-主存系统平均访问时间？Cache 基本结构原理？P112 图 4.50
   Cache 的读写？
   写直达法
   写时同时写 Cache 和主存，速度慢，但读时快。
   写回法
   写时只写 Cache，加标志位区分是否与主 存对应块相同，只在替换时写回内存，写速度快但读失效时需访存，读时慢。
8. 主存地址，Cache 地址各字段的划分与映射 P117
   直接映射
   主存和 Cache 低 b 位都为块内地址，Cache 的高 c 位为 Cache 块地址，主存的高地址 m

位被分为 t+c，t 为标记位判断对应 Cache 存储块内容有效性。映射公式 i = j mod C，i 为 Cache 块号，j 为主存块号。缺点不够灵活，缓存利用率不高。

全相联映射

标记位由 t 变为 t+c。主存块可以映射到任意 Cache 块，但实现复杂。

组相联映射

两种方法的结合，采取块间直接相联，块内全相联的方法，将 Cache 分为 q+r+b，q 为组位，r 为组内位，b 为块内位，对应主存分为 s+q+b，q 来决定映射到哪几个可能的 Cache 块组，s 为剩余位，用作字块标记，决定映射到组内哪一块。一般又称为 2^r 路组相联。

第五章 输入输出系统

一、输入输出系统的概述

1、输入输出系统的发展概况 P156—5.1,1

2、输入输出系统的组成——I/O 软件+ I/O 硬件

（1）I/O 软件

1.三个功能：将用户编制的程序（或数据）输入主机内、将运算结果输送给用户、实现输入输出系统与主机工作的协调。

2.I/O 指令与通道指令详见 P158~P159

（2）I/O 硬件 P159

3、I/O 设备与主机的联系方式

（1）I/O 设备编址方式——统一编址/不统一编址（单独编址）

1.统一编址就是将 I/O 地址看作是存储器地址的一部分。

2.不统一编址就是指 I/O 地址和储存器地址是分开的，所有对 I/O 设备的访问必须有专用的 I/O 指令。

（2）设备寻址——实质上是译码

（3）传送方式——并行传送/串行传送

（4）联络方式——立即响应方式/异步工作采用应答信号联络/同步工作采用同步时标联络

1.异步工作方式下，I/O 设备与 CPU 各自完成自身的任务，一旦出现联络信号，彼此才准备交换信息。详见 P161 图 5.6 给出的模型

2.同步工作要求 I/O 设备与 CPU 工作速度完全同步

（5）I/O 设备与主机的连接方式——辐射式（初级阶段）/总线式

4、I/O 设备与主机信息传送的控制方式

五种方式：程序查询方式、程序中断方式、直接存储器存取方式 DMA、I/O 通道方式、I/O 处理机方式。主要介绍前三种方式。

（1）程序查询方式——P163 图 5.9

CPU 通过程序不断查询 I/O 设备是否已做好准备，若设备未准备就绪，就继续查询，直到查得设备准备就绪，就将数据从 I/O 接口送至 CPU。这种方式下 CPU 和 I/O 设备处于串行工作状态，在反复的查询过程中 CPU 不能执行原程序，相当于原地踏步，工作效率不高。

（2）程序中断方式——P164 图 5.11

CPU 启动 I/O 设备后，继续执行自身程序，只是当 I/O 设备准备就绪并向 CPU 发出中断请求后才予以响应。这种方式解决了 CPU 原地踏步的问题。这种方式将在之后详细

介绍。

（3）DMA 方式

主存与 I/O 设备之间有一条数据通路，不需要调用中断服务程序就可以直接访问主存。当这种访问与 CPU 访问主存发生冲突时，CPU 总是将总线占有权让给 DMA，此时称为窃取周期。这种方式将在之后详细介绍。

二、I/O 设备——输入设备+输出设备

详见 P166—5.2

三、I/O 接口

1、总线方式的 I/O 接口：数据线+设备选择线+命令线+状态线

2、接口的功能和组成：选址+传送命令+传送数据+反映 I/O 设备工作状态

3、接口类型

（1）数据传送方式：并行接口、串行接口

（2）功能选择的灵活性：可编程接口、不可编程接口

（3）通用性：通用接口、专用接口

（4）数据传送控制方式：程序型接口、DMA 型接口

四、程序查询方式

程序查询方式的核心问题在于每时每刻不断查询 I/O 设备是否准备就绪。

程序查询流程及程序查询方式的接口电路请仔细阅读 P191 图 5.33、图 5.34 和 P192 图 5.35

五、程序中断方式

这里主要介绍 I/O 中断处理，有关中断的其他内容请参考第八章。

1、I/O 中断的产生

CPU 启动 I/O 设备后，继续执行自身程序，等到设备完成工作并向 CPU 发出准备就绪请求后，再转入 I/O 中断服务程序。P194 图 5.36 有助于理解程序中断方式。

2、程序中断方式的接口电路

（1）中断请求触发器 INTR&中断屏蔽触发器 MASK

1.当多个中断源向 CPU 提出请求时，CPU 必须坚持任何瞬间只能接受一个请求的原则。

2.CPU 总是在同一时间，执行完每条指令后发出查询信号。

3．INTR 为“1”时表示提出中断请求，MASK 为“1”时表示封锁其中断源的请求，接口内完成触发器 D 的状态为 1 表示设备就绪。结合 P195 图 5.37 理解 INTR、MASK 与 D 的关系。

（2）排队器 P196 图 5.38

1.此处介绍的排队器是设在各个接口电路中的排队器，又称链式排队器。设在 CPU 内部的排队器请参考第八章 P361 图 8.25

2.下面一排门电路是其核心。当它对应的 INTR 输入为 1 时便迫使其后的所有 INTP 变为低电平，封锁他们的中断请求。

（3）中断向量的形成部件——设备编码器

向量地址≠中断服务程序的入口地址

（4）程序中断方式接口电路的基本组成 P198 图 5.41

3、I/O 中断处理过程

中断请求→中断判优→中断响应→中断服务→中断返回

详见 P198——2.I/O 中断处理过程

4、中断服务程序的流程

（1）保护现场

保护程序断点（中断隐指令）、保护 CPU 内部寄存器数据（中断服务程序）

（2）中断服务

处理中断请求对应的问题

（3）恢复现场

将 CPU 内部寄存器的数据恢复到中断前

（4）中断返回

返回程序断点，继续执行原程序

*CPU 一旦响应了某中断源的请求后，便由硬件线路自动关中断。此后如果不用指令开中断，意味着当前中断服务程序执行过程中不在响应其它中断请求。如果在保护现场后用软件指令开中断，那么 CPU 可以在执行当前中断服务程序时响应优先级更高的中断，从而形成多重中断。请仔细阅读 P201 图 5.43

六、DMA 方式

1、DMA 方式的特点

主存与 I/O 设备之间有一条数据通路，不需要调用中断服务程序就可以直接访问主存。

由于 DMA 接口和 CPU 共享主存，有可能造成冲突，故采用以下三种方法完成 DMA 与主存交换数据

（1）停止 CPU 访问主存

（2）周期挪用

（3）DMA 与 CPU 交替访问

2、DMA 接口的功能和组成

（1）DMA 接口的功能：向 CPU 申请 DMA 传送、处理总线控制权的转交、在 DMA 期间管理系统总线、确定传送的起始地址和数据长度、数据传送完成后发出 DMA 完成信号

（2）DMA 接口的组成

详见 P205 图 5.47

3、DMA 的工作过程

（1）DMA 传送过程

预处理→数据传送→后处理；详见 P206~P208

（2）DMA 接口与系统的连接方式 P209 图 5.49

具有公共请求线的 DMA 请求&独立的 DMA 请求

*与程序中断方式相比 DMA 方式的特点：

1.从数据传送看，程序中断方式靠程序传送，DMA 方式靠硬件传送。

2.从 CPU 响应时间看，程序中断方式是在一条指令执行结束时响应，而 DMA 方式可在指令周期内的任一存取周期结束时响应。

3.程序中断方式有处理异常的能力，DMA 方式没有这种能力。DMA 方式主要用于大批数据的传送，如硬盘存取、图像处理、高速数据采集系统，可提高数据吞吐量。

4.程序中断方式需要中短现行程序，故需保护现场；DMA 方式不中断现行程序，无需保护现场。

5.DMA 的优先级比程序中断优先级高。

第六章　计算机的运算方法

1．进位计数制之间的转换（整数，小数）P293

2．定点数：原反补之间的转换　　P220

3．浮点数：给出给定格式的浮点数代码写出真值　　P229

229 页中间的 N 的阶码是二进制表示法

浮点数定义，尾数，阶码，基数，规格化数（好处？），浮点数表示范围，上 / 下溢出，如何规格化浮点数

好题：例 6.6

移位运算：补码的好处，左移符号位不变，补 0；右移符号位不变，补符号位

4． 二进制补码的加减运算及溢出判断　　P235

例 6.10，6.11

（【-B】补 ）= （【B】补）符号位求反后再求补

一位符号位判溢出：观察两运算数与结果符号位是否相同

两位符号位判溢出：真符号看高位，结果两位符号位不同则溢出。

5． 定点原码一位乘法和定点补码一位乘法的计算过程 P243

乘，加，移

表 6.8　例 6.17

1) 符号位先心算，两数的绝对值进行定点一位乘

2) 几位运算数就运算几次加法和右移

3) 乘数末位值确定被乘数是否与原部分积相加, 之后结果右移一位, 形成新部分积，　高位补零，之后乘数也右移一次，次低位做新的最低位，最高位放部分积的最低位。

4) 每次加法，被乘数仅与原部分积高位相加，低位被移至乘数所空出的高位位置。

5) 最后的结果是 符号位 ．部分积 乘数

补码一位乘　　P250

例 6.19　例 6.20

符号位运算中自然生成，运算步骤

乘数>0

　　运算步骤与原码一位乘相同，乘加移，只不过符号位变为两位，运算时部分积符号位一直补 1，部分积为补码，尾数右移高位始终补 1。

乘数<0

　　按原码一位乘的运算规则运算，最后加上（ [-x]补 ）负的被乘数的

补码校正。

6． 浮点数的加减运算　　P269

例 6.30

1)对阶，小阶向大阶看齐，之后小阶尾数补码右移补 1，可能会丢失精度

2)尾数求和

3)规格化，1/2 <=| S |<1,补码实现

两符号位相同，符号位与数值位第一位相同则左移，| S |<1/2

两符号位不同，说明上下溢出，需要右移尾数使其减小，符号位右移给到数值位高位，直到符号位相同，符号位与第一位数值位不同规格化结束。

4)舍入判断

0 舍 1 入（右移丢谁再加谁）；恒置 1 法（尾数末位横置 1）

5)溢出判断

观察阶符，阶符两位 01/10 则发生上 / 下溢出

7． 原码一位除 P258

例 6.25

不恢复余数法，加减交替法

符号位自行异或判断

[x/y]原 保证 x*<y*
r > 0 2r-y*,减去除数
r < 0 2r+y*,加除数
步骤为 ； 判正负，上商 10，余数和商左移，加减除数
第一次被除数为余数，y*>x*，减去除数后结果为负，第一次一定上商 0，之后有几位就左移，加减，判余数正负，上商几次。
结果为符号位. 商的后 n 位（第一位选择正常一定为 0）。
左移余数末位补 0

第七章 指令系统

一、指令的一般格式：
1、操作码+地址码。
2、操作码的长度可以是固定的也可以会变化的，通常采用拓展操作码技术，使操作码的长度随地址数的减少而增加。P301 图 7.2& P302 例 7.1
3、可按地址数的不同对指令分类。P303
4、指令字长 P304
二、寻址方式
（一）指令寻址
1、顺序寻址：PC+1.
2、跳跃寻址：转移类指令。
（二）数据寻址
1、指令中的寻址特征字段指明寻址方式
2、区分形式地址 A 和有效地址 EA
3、十种寻址方式（详见 P310~P319）
（1）立即寻址—立即数（图 7.8）
（2）（存储器）直接寻址：EA=A；A 的位数限制寻址范围（图 7.9）
（3）隐含寻址—EA 由操作码确定（图 7.10）
（4）间接寻址：EA=（A）；与直接寻址相比，它扩大了寻址范围；访存次数增加（图 7.11）
（5）寄存器（直接）寻址：不需要访问内存（图 7.13）
（6）寄存器间接寻址：操作数地址存放在寄存器中（图 7.14）
（7）基址寻址：EA=A+(BR)；基址寄存器中的数据+形式地址→有效地址（图 7.15）
（8）变址寻址：EA=A+(IX) ；变址寄存器中的数据+形式地址→有效地址（图 7.16）
*注意区分（7）（8）见 P315 最后一段
（9）相对寻址：EA=（PC）+A；位移量 A 通常是补码（图 7.17）
*P318 例 7.2
（10）堆栈寻址—可被视为一种隐含寻址（图 7.18）
三、设计指令
P323~P325 例 7.4/7.5/7.6/7.7
P335 题 7.16
四、RISC 技术
1、RISC 技术的起因 P326

2、RISC 的特点 P330
3、RISC 和 CISC 的比较 P333

第八章 CPU 的结构和功能

一、CPU 的结构——运算器+控制器
（一）CPU 的功能
1、本章重点介绍控制器的功能，运算器的功能在第六章
2、控制器的基本功能：取指令+分析指令+执行指令（具体介绍见 P337）
（二）CPU 的结构
ALU+寄存器+中断系统+CU+系统/内部总线
P338 图 8.1、图 8.2
（三）CPU 的寄存器 P338~P341
1、用户可见（用户可以对其操作）：通用寄存器、数据寄存器、地址寄存器、条件码寄存器
2、控制和状态寄存器（通常用户不能对其操作）
MAR 存储器地址寄存器、MDR 存储器数据寄存器、PC 程序计数器、IR 指令寄存器
*以上四种寄存器在指令执行过程中起重要作用，以取指令为例：PC→MAR→M→MDR→IR
还有 PSW 程序状态字寄存器
二、指令周期——CPU 取出并执行完一条指令的全部时间
取指周期+（间址周期）+执行周期+（中断周期）
注：上面给出的指令周期中间址周期实质上也是在完成取指的工作，是否进行间址要根据指令的寻址特征判断。
三、指令流水
四、中断系统——中断请求+中断判优+中断响应+终端服务+中断返回
（一）引起中断的因素&终端系统需解决的问题 P359CPU 响应中断的条件：中断请求&&没有屏蔽&&程序状态字允许中断
（二）中断请求标记和中断判优逻辑
1、INTR 中断请求触发器，它可以分散在接口中，也可以集中设在 CPU 内组成中断请求标记寄存器。
2、中断判优逻辑：
（1）硬件排队：
1.链式排队器：INTR 分散在各个接口电路中的情况 P196 图 5.38
2．集中在 CPU 内的排队器 P361 图 8.25
*两种排队器都通过电路设计达到优先级高的信号封锁优先级低的信号的目的。
（2）软件排队 P361 图 8.26
（三）终端服务程序入口地址的寻找 P361~P362
1、硬件向量法——中断向量的形成部件实质上是一个编码器，它的输入来自排队器的输出。
2、软件查询法
（四）中断响应 P362~P364
1、CPU 响应中断的条件：中断请求&&没有屏蔽&&程序状态字允许中断
2、响应中断的时间：指令执行结束后，CPU 发出终端查询信号，CPU 获知中断请求。
3、中断隐指令——进入中断周期，CPU 自动完成的一系列操作
（1）保护程序断点
（2）寻找中断服务程序入口地址——向量地址送至 PC，使 CPU 执行下一条无条件转

移指令，转至终端服务程序入口地址

（3）关中断

4、保护现场和恢复现场

保护现场：保护程序断点（由中断隐指令完成）+保护 CPU 内部各寄存器的内容（由中断服务程序完成）

恢复现场：恢复 CPU 内部各寄存器的内容到中断前（由中断服务程序完成）

5、中断屏蔽技术——主要用于多重中断

（1）多重中断（中断嵌套）：新出现的中断可能打断当前执行的中断服务程序，转去执行新的中断服务程序。

（2）实现多重中断的条件

1.提前设置开中断指令（在保护 CPU 内部寄存器之后，防止数据丢失）

2.优先级别高的中断源有权中断优先级别低的中断源

（3）屏蔽技术——MASK 屏蔽触发器+屏蔽字 P366 图 8.33

1.每个 INTR 对应一个 MASK，将所有 MASK 组合在一起构成屏蔽寄存器，屏蔽寄存器中的内容被称为屏蔽字。

2.屏蔽技术可以改变优先级——让中断服务程序加载新屏蔽字

*P368 图 8.35 中处理完 A 之后经由 B、C 逐级进入 D 的期间就是在加载 B 和 C 新的屏蔽字。

*置新屏蔽字后要在服务程序最后恢复原屏蔽字。

*具体过程见 P368 图 8.36，P369 例 8.2

第九章 控制单元的功能

理解指令周期、机器周期、时钟周期、节拍、控制信号的关系，重点：控制单元

一、指令周期：

1. 取指周期：(PC)→MAR;

M(R),M(MAR)→MDR;此处注意 “读信号” 的持续时间

MDR→IR,(PC)+1→PC;

2. 间址周期：完成取操作数有效地址

IR(AD)→MAR;

M(R),M(MAR)→MDR;

MDR→IR(AD); PS：此处指令寄存器的数据地址字段比 MDR 的长度要小，这样写会有错误，此处是为了突出间址。

3. 执行周期：P376-378 P378 图 9.1

4. 中断周期（详情见第八章、第五章中断部分）

相关题目：P393 9.1

二、控制单元：主要功能是发出各种微操作命令（控制信号）序列

了解控制单元：可根据 P379 图 9.2 了解 P379 内容

“ADD @X” 为例理解指令流程各阶段相应的控制信号：（P376—P377）

1. 不采用 CPU 内部总线的方式

2. 采用 CPU 内部总线的方式

PS：两种方式的基本流程相同，只是有 CPU 内部总线的方式更简洁明朗，各种信号大都上总线进行传输。

相关题目：P382 例 9.1，9.2

三、多级时序：
机器周期：所有指令执行的一个基准时间，在各种操作中访问存储器的时间最长，故通常以访问一次存储器的时间定为机器周期（在同步控制条件下成立）
时钟周期（节拍）：完成一组微操作命令的时间，节拍的时间与时钟周期相同
指令周期、机器周期、时钟周期、节拍的关系：指令周期是 CPU 完成一条指令的时间，它包括若干个机器周期，一个机器周期包括若干个时钟周期（节拍）。P387 图 9.9
此部分：P386 图 9.8。
*（了解）控制单元的控制方式：同步控制方式、异步控制方式、联合控制方式、人工控制方式。

第十章 控制单元的设计

控制单元设计分为组合逻辑设计和微程序设计两种方法
一、组合逻辑设计：
取指周期是公共部分，间址周期和执行周期根据寻址方式的不同和指令的不同功能相应的微操作不同，相应的控制信号也不尽相同。
步骤：
1. 列出微操作命令的操作时间表：
具体步骤：各个指令的取指周期步骤相同，故对应的微操作命令也相同。写出各个指令执行流程的各个节拍里的微操作命令，得到操作实践表 P402 表 10.1
写各个指令执行流程的微操作命令时，注意节拍安排：P396 写流程相关例题：P398 例 10.2
2. 得到各个控制信号的最简逻辑表达式：输入信号（如 MARi,PCi 等）需要配合 CP，输出信号则不需要。
   如 MARi：(T0+T3(MOV+STA+ADD))·CP，（解释：MARi 出现在了 T0 节拍中，T3 节拍中的所有命令并没有都出现 MARi，故选择出现了的 MOV,STA,ADD 命令与 T3 进行“与操作”,MARi 还需跟 CP 配合。）
3. 画逻辑电路 P403 图 10.3

二、微程序设计
微命作：同一节拍中执行的一个或一组微操作
微指令：由微命令和下地址字段组成
微程序：一组可以完成特定功能的微指令。例如取指周期有三个阶段，故分为三条微指令。
控制存储器（简称控存）:CM，位于 CU 中，用于存放所有微程序 详细结构见 P404 图 10.4
控存地址寄存器：μMAR
控存数据寄存器：μIR，用于存放从控存读出的微指令，在执行时预取下一条微指令，从而形成流水线，并且“不断流”
执行流程：
取指阶段：（取指周期微程序是公共部分），例如取指阶段的三条微指令存于控存 00000,00001,00010 中，将取指周期微程序首地址（00000）送入μMAR，进入取值阶段，通过μMAR 内容加“1”跟换地址信息，从而依次执行 00001,00010.
执行周期：取指周期执行完 00010 后 IR 的内容（ADD,STA,LDA 等指令的执行周期的微程序在控存中的首地址）送到μMAR 中，开始执行周期，执行周期执行完后跳转到取指周期首地址，再开始新的微程序的执行。
下地址字段：包括“方式”和下一条微指令的地址，此处“方式”可举例：“1”表示顺序执

行，“0”表示跳转

微指令的编码方式：1。直接编码方式：微指令的的微命令字段中，每一位都代表一个微操作命令，P407 图 10.7

2.字段直接编码方式 :互斥微命令放在一个字段中, 减少了微命令字段的位数 P407 图 10.8

3.字段间接编码方式 4.混合编码等

后续微指令的地址的形成方式：

1.直接由微指令的下地址字段给出，又称为断定地址

2.根据机器指令的操作码形成

3.增量计数器法 4.分支转移 5.通过测试网络形成 6.由硬件产生微程序入口地址

微指令格式：1.水平型微指令 2.垂直型微指令

PS：微指令的存储方式：（补充说明，课本上没有）

1. 微命令部分放在微指令的后半部分：可以充分利用存储空间。
2. 微命令部分放在微指令的前半部分：不能可以充分利用存储空间，造成空间浪费
3. MAPROM：MAPROM 用于存放所有微程序的编码（如 ADD 为 000，STA 为 001 等）和它们在控存中的首地址，程序执行时先通过 MAPROM 找到微程序首地址，然后执行。类似于间址寻址，要访问两次存储器，速度比较慢。

[1]