INTEL 8080A/8085A 微处理器及其接口的应用

—王诚志—

无锡江宁机械厂职工大学 印

目　录

第一章 微处理器
MicroProcessor (μp)

第一节 8080 微处理器芯片

一、概 述：

微处理机：即控制机与运算机集成在一块或数块大规模集成电路上的中央处理机 cpu。

制造工艺：目前采用 pMOS、NMOS、CMOS，及双极型〔包括 I^2L〕等技术。

pMOS：使用较早，生产工艺成熟，成品率高，器件工作速度较忙。

NMOS：目前应用较多，速度快，集成度高。适合大规模集成电路生产。

CMOS：结合 pMOS、NMOS 的工艺特点，速度介于两者之间，集成低于 NMOS。

特点：功耗低，抗噪声能力强，体积小，重量轻，适用于航天设备 技术装置。

双极型技术：主要采用低功耗的肖特基 TTL 速度最快。

执行一条指令：双极型 —— 70 ~ 100 ns

MOS型 —— > 1 μs

主要缺点：功耗大，集成度低，主要用来制造位片器件，还不能在一个芯片上制造完整的微处理器。

μp 的字长：4位，八位，12位，16位，目前八位机应用最广泛。

μp 结构：有四种，单片微处理器；单片微型计算机；双片微计算机，位片式结构（bit chip）。

1. 单片微处理器：将中央处理器的主要功能（不是全部功能）集成在一个芯片上，通常将时钟电路，石英晶体外配，而新型的Intel 8085等也将时钟电路组合在up中，而晶体外设。
2. 单片微型计算机：把up、时钟电路，存贮口I/O接口集成在一个单片上，价格低廉，功能较少，宜用于简单场合。
3. 双片微型计算机：由两个片子组成。一片是up，它与单片up功能相似，並包含时钟，具有一些直接输入输出能力。另一片集成了存贮器及I/O接口。
4. 位片式器件：还能称作up，它主要包含运算器件，没有控制部份。

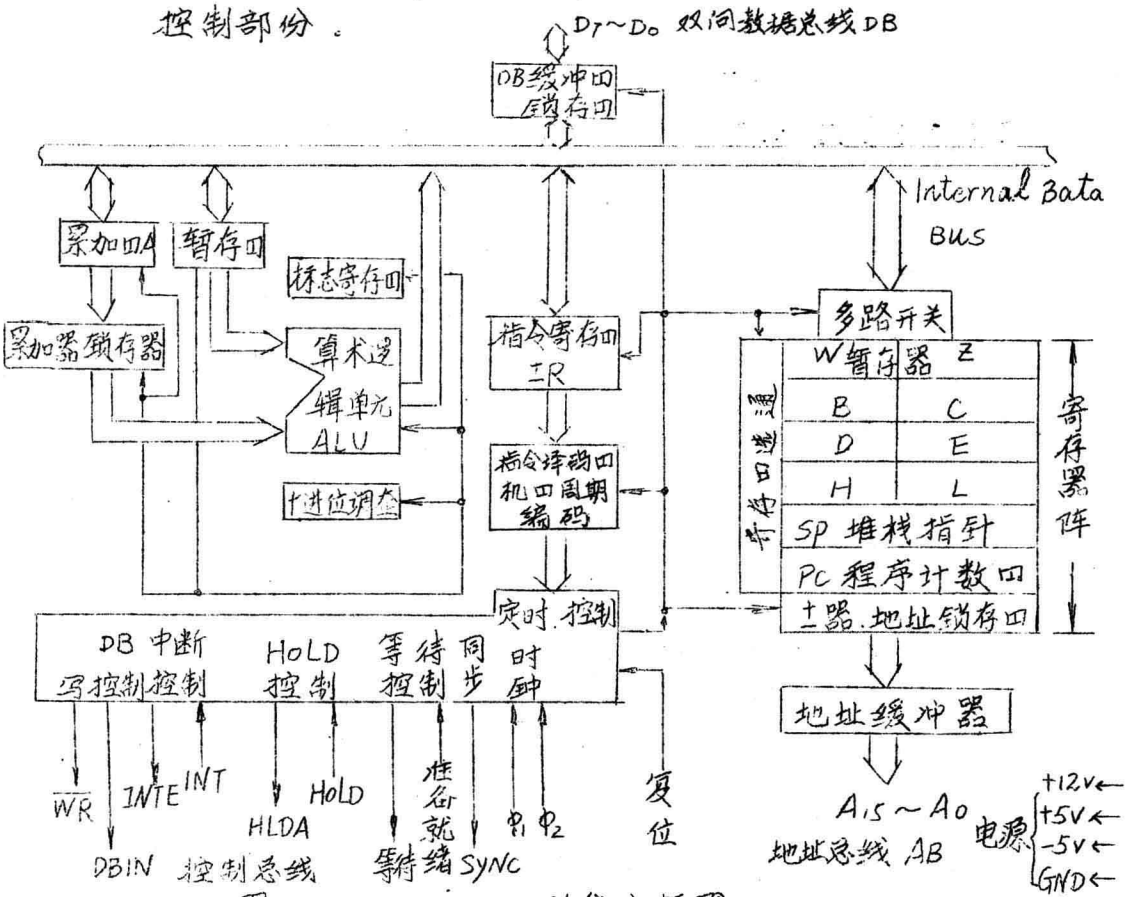


图1-1 8080 CPU功能方框图

二、微处理器的内部逻辑

典型 up 包括四部分：

1. 运标逻辑单元ALU .

2、寄存器阵列和地址逻辑 .

3. 指令寄存器和控制部分 .

4、双向了态数据总线缓冲器 .

Intel Corp . 1971年12月发表 8080 up . 是最早的八位 up 采用 pmos 工艺 . 指令执行时间 20 us

1973 年第一代 8Bit up . 采用NMOS 工艺 . 提高了集成度性能改进 . 指令执行时间缩短2 us

1975年发表8080A为改进型 。速度 、集成度都提高了。

8080内部主要包括以下部分：如图 1—1 所示 。

1. 运标器 . Arithmetic unit

2. 控制器 . Controler

3. 六个八位通用寄存器 。

4. 一个16位程序计标器 .

5. 一个16位堆栈指示器 .

6. 五位状态码寄存器 .

7. 八位数据总线 .

8. 16位地址总线 .

9. 控制母线

分别介绍如下：

1. 标术逻辑单元（ALU）

功能：执行标术 . 逻辑运标和移位操作

包含部分 . An 8-Bit accumulator (A)

An 8-Bit temporary accumulator(ACT)

A 5-bit flag register: Zero.Carry.Sign.

parity. auxiliary carry.

An 8-bit temporary Register (TMP)

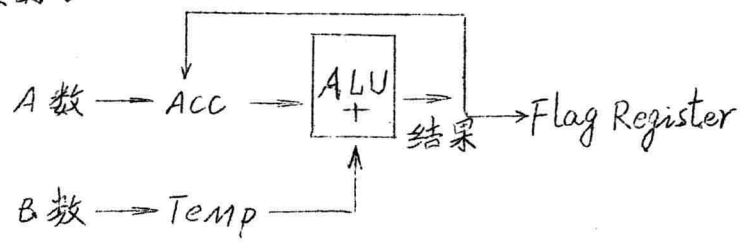
运称叫内部主要包含八位并行二进制全加器：参加运称的加数被加数分别放在暂存器与累加器锁存器，运算结果通过内部数据母线送入累加器，（也可存运算的中间结果）

累加器： 存放运算器处理中的一个操作数，并能存放运算结果或中间结果。

暂存器： 暂时存放从存贮器或寄存器经内部总线送来的数据，是不可编程的即不能用编程方法来改变它的内容。

累加器锁存器： 用来锁存即将由运算器进行处理的累加器中的数据，也是不可编程的。

两数相加过程：



运算器只作二进制加法，若采用 = - + 调正指令：DAA，微处理器就能作十进制数运算。在执行 DAA 指令时，ACC 的内容和 Auxiliary Corry fl:P flop作十进制调正操作，所以AC 内容需要测试。

2. 程序计数器 PC

PC 是一个16位的寄存器，专用来存放将要执行的下一条指令的地址。

指令：指示机器作何种类型操作的命令。

通常指令在存贮器内按顺序存放，每个存贮单元都编了号码，这个号码称为地址码，Address Code

up执行指令步骤：首先将pc内容经地址母线送Memory.

按地址在对应单元内取回八位二进制码 → Instruction Register. 此组二进制数规定了 up 要进行的操作，所以称作操作码 opcode。操作码经译码后按规定内容执行。此过程通称为取指令过程。

指令通常在存贮器内按次序逐条执行。每当程序计数器发出它的内容就自动加 1，保证程序指令自动按顺序执行。

PC 为 16 位，故可寻址 65，536 个单元。

3. 指令寄存器和指令译码器：

取 operation code → Instr Reg 经 Instru decoder 后把二进制数的操作码变成脉冲电位控制信号，并与时钟信号相结合，在适当时刻使相应电路产生合适的动作。

4. 通用寄存器：

6 个 8 位通用寄存器，B、C；D、E；H、L。

是 up 的内部工作单元，作用相当于存贮器的存贮单元。有了它们在运算中间就减少了访问存贮器的次数，提高了运算速度。可成对使用，组成三个 16 位寄存器，记作 B、D、H。

多路转换口：Multiplexer

Data bus ⇌ Multiplexer ⇌ 被寻址的寄存器或通用寄存器。

通用寄存器不仅存放数据，也可存放地址码。当进行存数或取数操作时，可将该地址码经地址锁存器，地址缓冲器 → Addr bus，按地址完成存取操作。

W·Z 称为暂存寄存器，

暂时存放内部 Data bus 上的数据，它是不可编程的

W、Z 也可临时存放地址。如 SHLD addr 指令，addr → W.Z，然后执行 HL → Memery，完成直接存 HL 操作。

5. 堆栈指示器 SP

在存贮器中指定一部分单元作堆栈用，该单元称为堆栈存贮器。

SP 堆栈指示器是 up 内的一个16位寄存器，它存放堆栈顶的地址。

栈顶：8080 中指示为地址 m，该单元存放数C。写为：(SP) = m

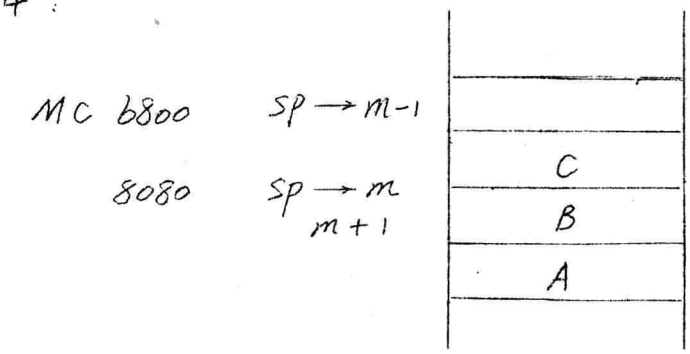
MC 6800 中指示为 m − 1，该单元未存放数

栈顶指针指示数C上方的一个单元，(SP) = m−1

堆栈与一般存贮器的区别：

堆栈是按"先进右出"、"右进先出"的规则存取数据的。即遵守 FILO 原则（FiRstin lastout）。

在 8080 中：



入栈次序：先作 SP − 1 即 m − 1，然后按 SP 地址将数入栈。

栈 入栈 ← A ← B ← C

退栈次序：栈 A → B → C ⋯⋯ 退线。按 SP 内容即 m 地址取相应数退栈，然后 SP + 1 → SP 栈顶指针指向 m + 1。

堆栈用于中断服务时，保护 up 各寄存器的现有内容，以防再次使用寄存器时被破坏。

6. 状态码寄存器；Condition code Register or Flag Register 简称 CCR. PSW。亦称程序状态字 PSW；program state Word。

功能：存放标志，逻辑操作后运算器的各种状态。

在 8080中CCR 也是一个8 bit 寄存器，仅使用5个状态码，其余3位是固定值，分析如下：

b_7							b_0
S	Z	0	Ac	0	P	1	Cy

(1) 进位位（C），相当于运称器的第九位。两数加、减在第八位有进位或借位时，在第九位C被置1，否则 C=0，移位操作时，C就被移入相应位的数。

左移操作。 C ← b_7 ← b_6 - - - - - ← b_0

(2) 辅助进位位 Ac

当两数相加、减时，第四位向第5位有进位，借位时，AC 置=1，否则 AC=0，或说 b_3 对 b_4 的进位，借位状态。在 DAA=-+ 进制调正时，测试AC 位后再进行操作。

(3) 符号位 S

指示运称器中 最高位 b_7 的情况 $b_7=1$ S=1 负数(补码)

$b_7=0$ S=0 正数

(4) 另位 Z

操作后运算器8位全为另，则 Z=1 指示另的状态

运算器结果中有一位不为另， Z=0

检查运算结果用

(5) 奇偶位 P

用于奇偶校验。即8位二 进制数中1的个数为奇数还是偶数。

奇校验：要求一组二进制数中1的个数为奇数 P=0

若为偶数 P=1。

例：00001011；P=0

8080采用奇校验 00101011；P=1

偶校验：要求一组二进制数中1的个数为偶数 P=0

若为奇数 P=1。

例：10101001； P=0

00101001； P=1

7. 双向数据总线缓冲口/锁存口

它由①输入缓冲口 2) 数据锁存口 3) 输出驱动口三部分组成. ①输入缓冲口：收息从外部输到芯片后，经缓冲口电路送到cpu的内部数据总线。

②数据锁存口：把内部数据总线的收息接收并保存起来，内部总线的数据收息一经锁存后，内部数据总线即可以与数据缓冲口/锁存口脱开，向外部送出的数据收息可以从锁存口中取得。

③. 输出缓冲驱动口：将内部数据总线或数据锁存口来的收息经功率放大后，送出片外。

三. 微处理器的内部操作

1. 指令格式

指令一般由操作码与地址码组成

op code 操作码：规定了up要执行的操作（有时隐含了操作数据）。

Operand 地址码：指明了操作的数据或操作数放在何处。

三字节指令

操作码
地址（高位）
地址（低位）

MC6800

操作码
地址（低位）
地址（高位）

8080
Z-80

二字节指令

操作码
地址码

单字节指令

操作码

图 1·2 指令存放格式

D ⟵ S

目的寄存口 ⟵ 源寄存口

例：MOV Y_1, Y_2 $Y_1 \longleftarrow Y_2$

二进制数码：

01DDDSSS

D, S的三位二进制数表示内部寄存回序号

Register	code
B	000
C	001
D	010
E	011
H	100
L	101
M	110
A	111

Register pairs	Registers	code
B	B, C	00
D	D, E	01
H	H, L	10
SP	SP	11

上述二字节及三字节指令的地址范围为：

8位地址码：寻址范围 0～255

16位地址码 寻址范围 0～65,535

2. 8080 的状态及机回周期

8080 的频率 —— 2MC，双相时钟 ϕ_1, ϕ_2 定时

① 机回状态：相当于一个时钟周期。up 在此期间完成最微小的基本操作。

8080 的机回状态：480 ns～500 ns (8085为320 ns)

②. 机回周期：对存贮回或 I/O 口进行一次存取操作，所需要的时间相当于一个机回周期，每个机回周期要求3～5个状态。

③. 指令周期：完成一条指令所需要的时间。

指令周期包括：Fetch 和 execution 周期。

每一个机回周期发送一组地址码寻址。取回指令的一个字节。

一般指令周期的机回周期数与指令的字节数相同，不另外需要附加周期，而另外一些指令则需要增加机回周期。

取指令所需的时间与指令长短有关。

执行指令的时间与指令类型有关。

8080 一条指令需 1～5 个机口周期，共 4～18 个状态。

每个状态 500 ns　　每条指令需 2～9 μs

3．取周期

执期指令：首先从存贮口内取操作码 → Instruction Register → decode 以决定下一步的操作。

取指令操作码的过程总是在指令的第一个机口周期 M_1 称为取周期参图 1.3

M_1 内各个状态如下：

1．T_1 状态：　Pc 输出

Pc 的地址码经 Address Bus → Menory，经存贮口地址译码口译码。选中相应的存贮口单元，经几百 ns 后，被选单元内容 → 存贮口数据输出端，並准备送上 data bus

2．T_2 状态：　Pc + 1 → Pc 以便进行下一机口周期或下一指令的操作。

3．T_3 状态：　opcode → instr Register

存贮口已将指令操作码放在数据母线上 → Instr Reg 並由 Instr decoder 译码，决定下面的操作。

取周期共用了 M_1 的三个状态

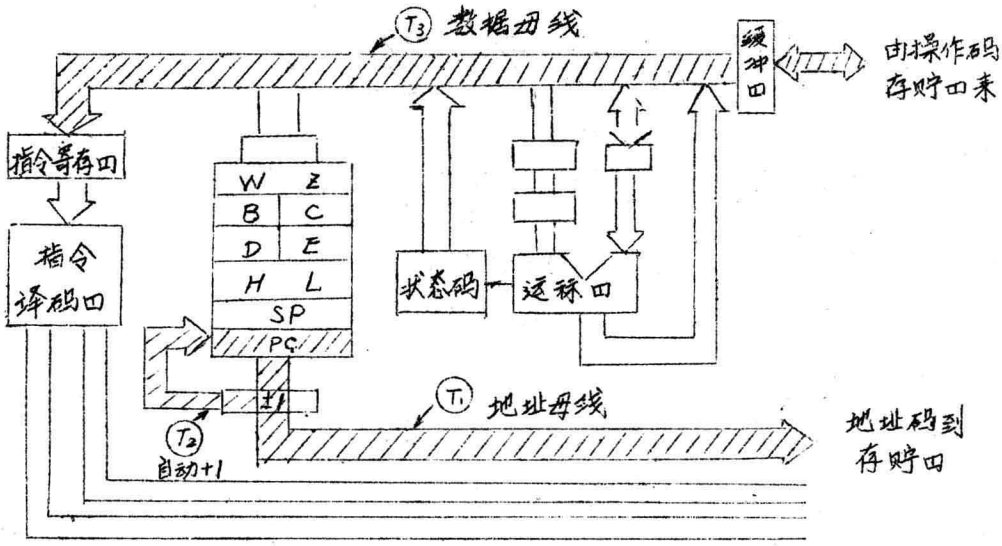


图 1·3 取指令操作码

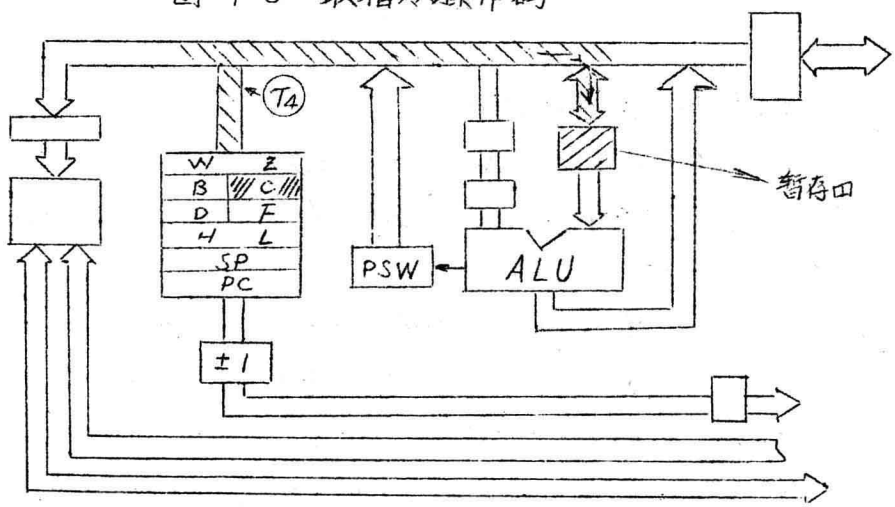


图 1·4 寄存口C —— 暂存口

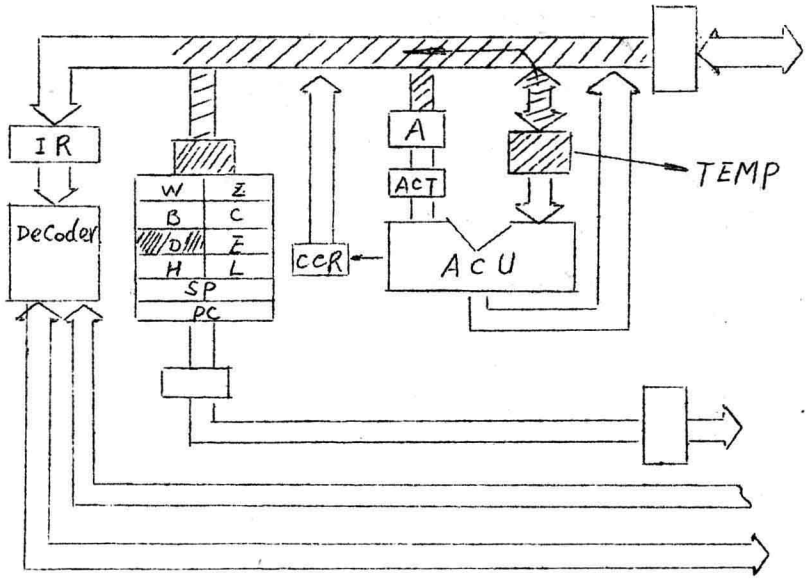


图 1.5 （暂存口）→ D

4. 8080典型指令的执行：

1.）传数指令 MOV D, C ; (D)←(C)

经过三个状态的取周期后，opcod→instruction Register

接着的操作为：

M_1：T_4：把寄存口C的内容→Temporary如图1.4所示

T_5：Temp→寄存口D，图1.5所示

本指令共用5个状态 约0.5 μs

C→Temp→D 为何用两个状态，不能直接C→D？

因通用寄存口按地址寻址，一次只能寻址一个寄存口，地址母线不能同时放两个地址码，所以中间借助暂存口Temp传送。

2）寄存口加法指令 add r: A←(A)+(r)

M_1 的 T_1～T_3 同前所述。此时为 add B; A←(A)+(B)

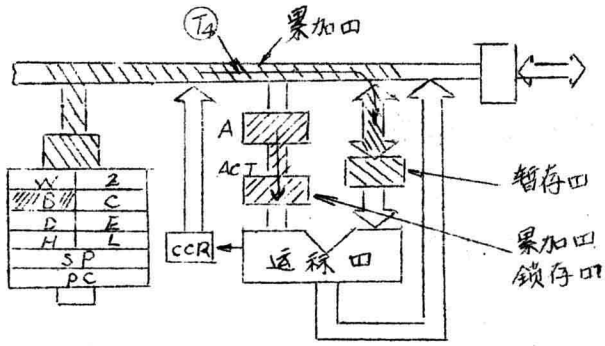
T_4 允许同时两次传输，（图1.6a）互不干扰，加快速度。

① (B) → Temp 通过内部 Data bus

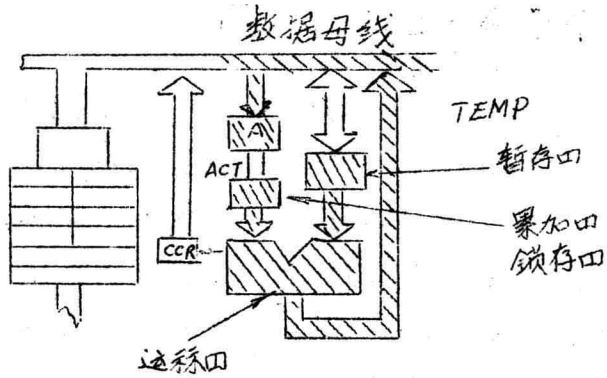
② 累加器 → 累加器锁存器 通过与 DB 无关的短的内部总线。M_1 的 T_5 与 M_2 的 T_1 状态都不进行该指令的操作。(而采用重迭操作法) 在 M_2 的 T_2 (累加器锁存器) } → DB → Accumulator
(Temp)

(图 1.6b)

这里重迭操作法: 节省时间 见图 1.7

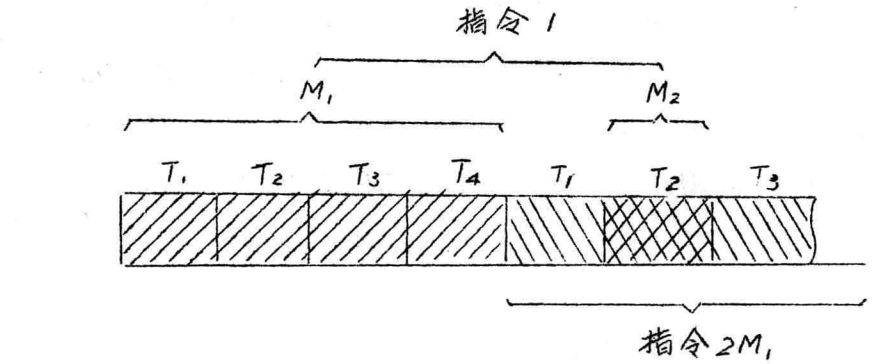


(a) 同时两次传输



(b) 加法操作

图 1.6 ADD 指令操作



指令1所用的状态

指令2所用的状态

指令1指令2共用的状态

图 1.7 指令的重迭操作

指令1的M_2周期与指令2的M_1周期在T_2重迭。这样第一条加法指令只用M_1的4个状态，共2μs。若再用T_5，M_1为2.5μs。

① 重迭运行不会产生乱操作？

指令2的M_1是取周期，仅用PC、PC+1，Addr Bus
指令1的M_2加法操作，仅用ALU及Data Bus
} 两者互不干扰。

② 能否在M_2的T_1进行加法操作？不能，因8080在每个机器周期的T_1状态，还要经Data Bus向外发出周期状态信息，以规定机回周期的类型，若此时作加法，会发生母线冲突。

③ 取数指令 LDA，a_1a_2

A ← (a_1a_2) 三字节指令

指令执行过程如下：图 1.8

M_1：T_1 — T_4 指令取周期