# 第 十 讲

## 微程序控制方式下模型机的设计实例（三）

-- 模型机微程序设计

# 五、模型机微程序设计

1. 模型机微程序设计的步骤
2. 微程序流程图的编写
3. 微地址及下址字段的分配
4. 微指令代码的编写
5. 微程序设计举例

# 1、模型机微程序设计的步骤

**设计的步骤为：**

- (1) 设计实验模型机的结构和数据通路；
- (2) 设计指令的功能、格式（包括指令码）及寻址方式；
- (3) 在以上的基础上，编写微程序流程图；
- (4) 根据指令码和转移方式 J1# ～ J5#，分配微地址及下址字段。
- (5) 根据微指令格式，编写微指令代码

# 2、微程序流程图的编写

1. (1) 机器指令的功能由微程序完成，一条机器指令对应着一段微程序。每条指令的微程序都包含三部分：

① 取指令微程序段

② 根据操作码散转至微程序入口的微指令

③ 该机器指令的独立微程序段

# 2、微程序流程图的编写

1. 每一条指令的前两部分都相同，称作公操作，不同的是第三部分的独立微程序段，取决于该机器指令的寻址方式和功能，用于实现的指令规定的特殊功能。
2. 例如：取指令及散转的公共微程序段为：

① PC→AR， PC＋1

② RAM→IR

③ J1 #散转至微程序入口。

# 2、微程序流程图的编写

1. (2) 每条微指令可以实现：

①总线上的一个数据传送：例如 PC→AR

②进行运算器的一个运算：例如 DA1+DA2→DR

③启动存储器的一个读/写：例如 RAM→DR。

④启动 IO 设备的一个读/写（输入/输出）：例如 DR→LED。

# 2、微程序流程图的编写

1. (3) 对于有二个操作码的指令（格式二），微程序段至少经过两个散转：第 1 次为 J1# 散转，分辨出寻址方式并计算出有效地址，第 2 次为 J2# 散转，分辨出指令并实现其功能。
2. (4) 编写指令的微程序流程图，不仅数据通路要可行，还要考虑微码编写是否可行。

# 2、微程序流程图的编写

- 例如， JMP Addr 指令，功能为将指令第二字的转移地址 Addr 送至 PC，实现转移的微指令为 RAM→PC
- 但其微码的编写却不可行，发送信号 M-R#（微码 FUNC FS= 0100 ）， B-PC# ， PC+1 （微码 FUNC FS=0001 ），冲突；
- 因此，必须用两条微指令实现：RAM→DA1 ； DA1→PC 。

# 2、微程序流程图的编写

- 又例如，取指令及散转的公共微程序段为：

  1. PC→AR， PC＋1
  2. RAM→IR
  3. J1 #散转至微程序入口

- 在电路原理上，J1#信号可以在b）微指令一起发送，但由于J1#信号与M-R#信号被编码在同一字段，因此必须分为两条微指令完成。

1. (5) 对于同一条指令，可能存在不同的微程序流程图，但均能实现指令的功能。

# (1) 通用寄存器之间的传送操作

1. 通过源寄存器内容送总线，而目的寄存器将总线上数据打入来实现。

例如：

1. 指令 MOV DR，SR ：功能为将源寄存器 SR 的内容送目的寄存器 DR 。
   - ✓ 其微程序段为一条微指令：
   - ✓ SR→DR 。

# (2) 存储器访问操作

1. 读访问操作：通过以下两条微指令实现：

   ① 送地址到总线，并打入地址寄存器 AR；

   ② 启动存储器读操作（M-R#），并将读出的数据从总线上接收至目的部件。

2. 例如，指令 MOV DR，[SR]：功能为将源寄存器 SR 所指示的存储器地址单元的内容送目的寄存器 DR，即源操作数是寄存器间接寻址。其微程序段为以下两条：

   - ✓ SR→AR
   - ✓ RAM→DR

# (2) 存储器访问操作

1. 取指令也是一种典型的存储器读访问操作。
2. 写访问操作：通过以下两条微指令实现：

   ① 送地址到总线，并打入地址寄存器 AR；

   ② 送数据到总线，启动存储器写操作（M-W#）。
3. 例如，指令 MOV [DR]，SR：功能为将源寄存器 SR 的内容写至目的寄存器 DR 所指示的存储器地址单元，即目的操作数是寄存器间接寻址。其微程序段为以下两条：

   ① DR→AR

# (3) IO 设备的访问操作

1. IO 设备的读访问操作：通过 2 条微指令实现
   a）送地址到总线，并打入地址寄存器 AR；
   b）启动 IO 的读操作（IO-R#），并将输入的数据从总线上接收至目的部件。
2. IO 设备的写访问操作：通过 2 条微指令实现
   a）送地址到总线，并打入地址寄存器 AR；

# (3) IO 设备的访问操作

1. 例如，指令 IN　DR，【PORTAR】，功能为将端口地址为 PORTAR 的输入设备的数据送目的寄存器 DR。其微程序段为三条微指令：

① PC→AR，　PC+1

② RAM→AR

③ IO→DR

# (4) 运算器的运算操作

1. 运算器的运算操作：通过三条微指令实现

① 送第一个数据到暂存器 DA1（或者 DA2）；

② 送第二个数据到暂存器 DA2（或者 DA1）；

③ 选择 ALU 运算功能并进行运算，结果送（ ALU-B# ）目的部件；

# (4) 运算器的运算操作

1. 例如，指令 ADD　SR，DR，功能为将源寄存器 SR 的内容与目的寄存器 DR 的内容相加，并送 DR。其微程序段为三条微指令：

① SR→DA1

② DR→DA2。

③ DA1+DA2→DR。

# 3、微地址及下址字段的分配

① 在指令系统中的所有指令的微程序流程图编制完成后，首先要分配每条机器指令的微程序入口地址。步骤如下：

I. 确定发送 J1# 信号的那条微指令的下址字段

II. 确定指令的操作码

III. 根据 J1# 转移的规则确定每条指令的微程序入口地址。

# 3、微地址及下址字段的分配

② 微程序入口地址计算：根据指令译码电路可以得出（J1# 译码控制，且下址 =10 H）：

. 当 I7I6 ≠ 11 时： 微程序入口地址 =10H+$I_7I_6I_5I_4$;

. 当 I7I6 = 11 时： 微程序入口地址 =30H+$I_5I_4I_3I_2$;

. J2# 译码控制时，只改变微地址的低二位，所以，形成的位地址为：

$MA_6MA_5MA_4MA_3MA_2I_3I_2$

. 同理，J3# 译码控制时，形成的位地址为：

$MA_6MA_5MA_4MA_3MA_2KaKb$

J4# 译码控制时，形成的位地址为：

$MA_6MA_5MA_4MA_3MA_2FCZF$

J5# 译码控制时，形成的位地址为：

$MA_6MA_5MA_4MA_3MA_2MA_1INT$

# 4、微指令代码的编写

- 根据写好的微程序流程图，参照数据通路，首先排列出每条微指令必须发送的微操作控制信号，然后，对照微指令格式，写出这些微操作控制信号对应的微代码。
- 举例

# 4、微指令代码的编写

| 序号 | 微指令 | 应发送的微操作控制信号 | 微指令编码（$M_{24}$ ~ $M_7$） | | | | |
|---|---|---|---|---|---|---|---|
| | | | BTO | OTB | FUNC | FS | $S_3$ ~ $S_0$ $MC_i$ |
| 1 | PC→AR,PC+1 | PC-B#,B-AR, PC+1 | 110 | 111 | 000 | 1 | 000000 |
| 2 | RAM→IR | M-R# ，B-IR | 011 | 000 | 010 | 0 | 000000 |
| 3 | J1# 散转 | J1# | 000 | 000 | 001 | 1 | 000000 |
| 4 | SR→DR | SR-B# ，B-DR | 100 | 011 | 000 | 0 | 000000 |
| 5 | DR→RAM | DR-B# ，M-W# | 000 | 100 | 001 | 0 | 000000 |
| 6 | DA1+DA2→DR | ALU(F=A 加 B) ，ALU-B# ，B-DR | 100 | 001 | 111 | 1 | 100101 |
| 7 | DA1→LED | ALU(F=A) ，ALU-B# ，IO-W# | 000 | 001 | 011 | 0 | 111110 或 000001 |
| 8 | DA1→PC | ALU(F=A) ，ALU-B#,B-PC# ，PC+1 | 111 | 001 | 000 | 1 | 111110 或 000001 |

# 5、微程序设计举例

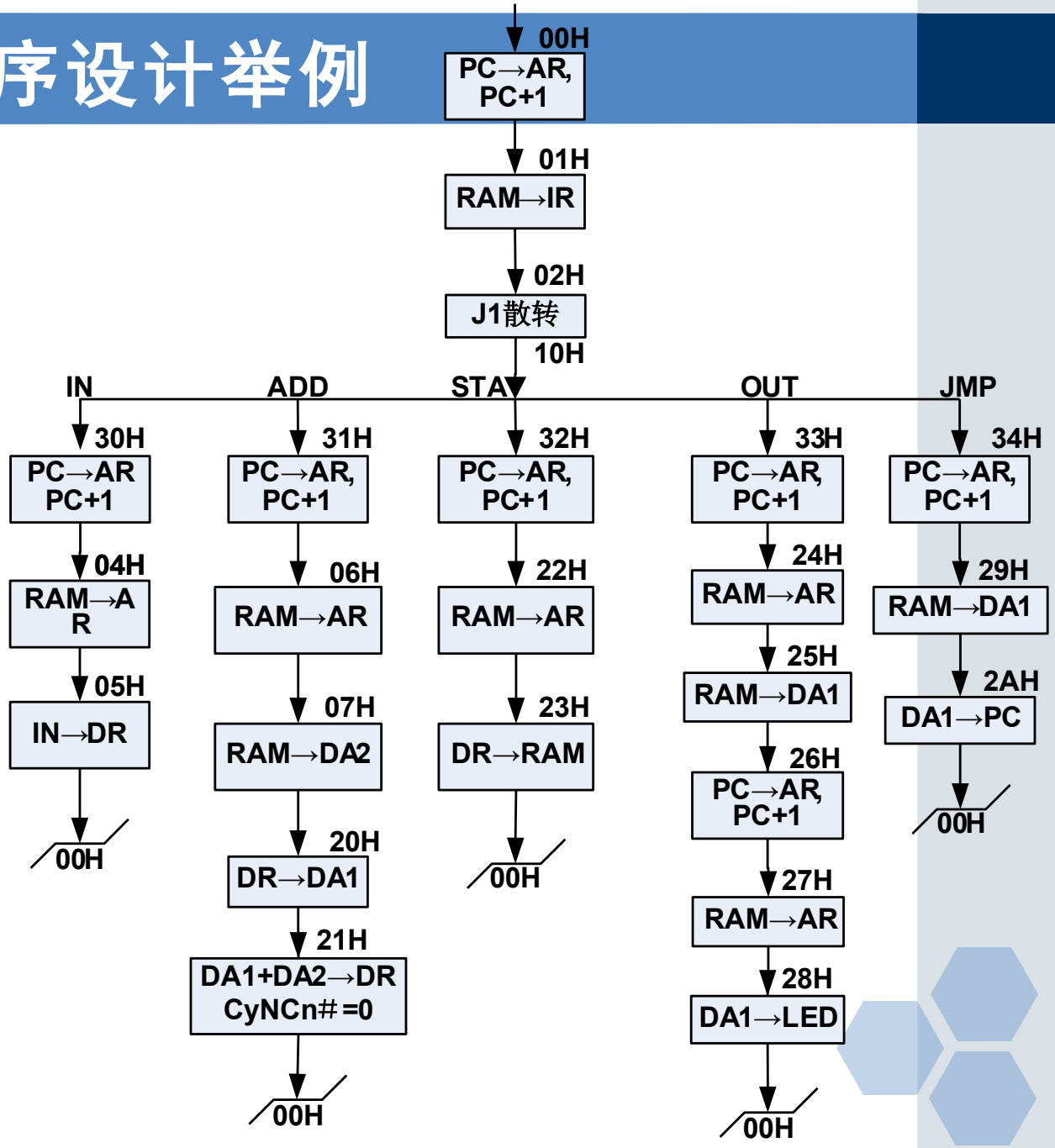
1. 假设模型机指令系统有5条机器指令，格式为：


| OP | 指令 | 功能 |
|---|---|---|
| 0000 | IN DR,[PORTAR] | IO(PORTAR) →DR |
| 0001 | ADD DR,[ADDR] | (DR)+MEM(ADDR) →DR |
| 0010 | STA [ADDR],DR | (DR) → MEM(ADDR) |
| 0011 | OUT [PORTAR], [ADDR] | MEM(ADDR) → IO(PORTAR) |
| 0100 | JMP ADDR | ADDR →PC |

# 5、微程序设计举例

1. 画出微程序流程图

2. 分配微地址



# 5、微程序设计举例

1. 编写每条微指令的代码，装入控存

| 微地址 | 微指令 | 微指令编码（$M_{23}$～$M_0$） | | | | | |
|---|---|---|---|---|---|---|---|
| | | BTO | OTB | FUNC | FS | $S_3$～$S_0$ MCi | 下址 |
| 00H | PC→AR，PC+1 | 110 | 111 | 000 | 1 | 000000 | 0000001 |
| 01H | RAM→IR | 011 | 000 | 010 | 0 | 000000 | 0000010 |
| 02H | J1# 方式散转 | 000 | 000 | 001 | 1 | 000000 | 0010000 |
| 04H | RAM→AR | 110 | 000 | 010 | 0 | 000000 | 0000101 |
| 05H | IN→DR | 100 | 000 | 100 | 0 | 000000 | 0000001 |
| 06H | RAM→AR | 110 | 000 | 010 | 0 | 000000 | 0000111 |
| 07H | RAM→$DA_2$ | 010 | 000 | 010 | 0 | 000000 | 0100000 |
| 20H | DR→$DA_1$ | 001 | 100 | 000 | 0 | 000000 | 0100001 |
| 21H | $DA_1$ +$DA_2$→DR, | 100 | 001 | 111 | 1 | 100101 | 0000001 |
| 22H | RAM→AR | 110 | 000 | 010 | 0 | 000000 | 0100011 |
| 23H | DR→RAM | 000 | 100 | 001 | 0 | 000000 | 0000001 |

# 5、微程序设计举例

1. 编写每条微指令的代码，装入控存

| 微地址 | 微指令 | 微指令编码（$M_{23}$～$M_0$） | | | | | |
|---|---|---|---|---|---|---|---|
| | | BTO | OTB | FUNC | FS | $S_3$～$S_0$ MCi | 下址 |
| 25H | RAM→$DA_1$ | 001 | 000 | 010 | 0 | 000000 | 0100110 |
| 26H | PC→AR， PC+1 | 110 | 111 | 000 | 1 | 000000 | 0100111 |
| 27H | RAM→AR | 110 | 000 | 010 | 0 | 000000 | 0101000 |
| 28H | $DA_1$→OUT | 000 | 001 | 011 | 0 | 000001 | 0000001 |
| 29H | RAM→$DA_1$ | 001 | 000 | 010 | 0 | 000000 | 1000000 |
| 2AH | $DA_1$→PC, PC+1 | 111 | 001 | 000 | 1 | 000001 | 0000001 |
| 30H | PC→AR， PC+1 | 110 | 111 | 000 | 1 | 000000 | 0000100 |
| 31H | PC→AR， PC+1 | 110 | 111 | 000 | 1 | 000000 | 0000110 |
| 32H | PC→AR， PC+1 | 110 | 111 | 000 | 1 | 000000 | 0100010 |
| 33H | PC→AR， PC+1 | 110 | 111 | 000 | 1 | 000000 | 0100100 |
| 34H | PC→AR， PC+1 | 110 | 111 | 000 | 1 | 000000 | 0101001 |

# 5、微程序设计举例

1. 编写测试程序的机器指令码，装入主存

| 地址 | 内容 | 助记符 | 备注 |
|---|---|---|---|
| 00H | 11000000B | ; IN $R_0$ ，[PORTAR] | IN ->$R_0$ |
| 01H | 00000000B | ; 端口地址 00H | |
| 02H | 11000100B | ; ADD $R_0$ ，[10H] | $R_0$+[10H]-> $R_0$ |
| 03H | 00010000B | ; 直接地址 10H | |
| 04H | 11001000B | ; STA $R_0$ ，[10H] | $R_0$->[10H] |
| 05H | 00010000B | ; 直接地址 10H | |
| 06H | 11001100B | ; OUT [10H] ，[PORTAR] | [10H]->LED |
| 07H | 00010000B | ; 直接地址 10H | |
| 08H | 00000000B | ; 端口地址 00H | |
| 09H | 11010000B | ; JMP 00H | 00H→PC |
| 0AH | 00000000B | ; 直接地址 00H | |
| …… | …… | | |

# 六、微程序控制器与硬布线控制器的比

| 比较内容 | 微程序控制器 | 硬布线控制器 |
| --- | --- | --- |
| 工作原理 | 微操作控制信号事先以微程序的形式存放在控存中，执行指令时读出即可 | 微操作控制信号由组合逻辑电路根据当前的指令码、状态和时序，即时产生 |
| 执行速度 | 慢 | 快 |
| 规整性 | 较规整 | 繁琐、不规整 |
| 应用场合 | CISC CPU | RISC CPU |
| 易扩充 | 易扩充修改 | 困难 |

# 本章小结

- 控制器是计算机硬件的核心部件，是根据机器指令产生执行指令时全机所需要的操作控制信号，协调控制计算机各个部件有序工作。掌握重点：
- **控制器的功能**：取指令、分析指令、执行指令
- **控制器的组成**：
  - 专用寄存器： PC 、 IR 、
  - 指令译码器 ID
  - 时序信号产生器（操作控制器）
  - 时序系统

# 本章小结

- **指令的执行过程：**由取指令阶段和执行阶段构成，取指令阶段的操作是公共的；而执行阶段的操作由指令操作码决定。不同的指令和不同的寻址方式，其执行过程是不一样的。

# 本章小结

- **控制器有两种设计方法：**
  - 硬布线控制器：它是将指令执行时的各个机器周期的微操作信号用组合／时序逻辑电路来实现；速度快，但设计复杂繁琐，适合于 RISC 结构。

# 本章小结

- 微程序控制器：一条机器指令由一段微程序解释，控制信号由微指令给出，指令功能的实现是由一些微指令给出的微操作有序组合实现。微程序控制器相对硬布线控制器速度慢，但设计比较规整，易于实现指令系统修改，适合于CISC结构。

通常，我们在复杂指令系统中，采用微程序控制器和硬布线控制器结合使用，对于使用频率高的指令采用硬布线实现，而对于一些复杂而又使用频度低的指令采用微程序控制器，以提高系统的性价比.