



第五讲

微程序设计技术（一）

—— 微指令控制字段的编译方法





三、微程序设计技术

一、学习内容

1. 构建一个模型计算机机，通过模型机的所具有的

部件和控制信号，设计出具体的微指令格式

；

2 . 微指令的三种控制方法；

3 . 微程序设计技术；

学习重点：

1. 微指令三种控制方法

2. 掌握微程序设计方法





三、微程序设计技术

1. 进行微程序设计时，应考虑以下因素：

- 有利于缩短微指令字长
- 有利于减少控制存储器的容量
- 有利于微程序的执行速度
- 有利于对微指令的修改
- 有利于微程序设计的灵活性





三、微程序设计技术

1

微指令的编译法

2

微指令下址字段设计方法

3

微指令格式的类型

4

微程序控存和动态微程序设计

5

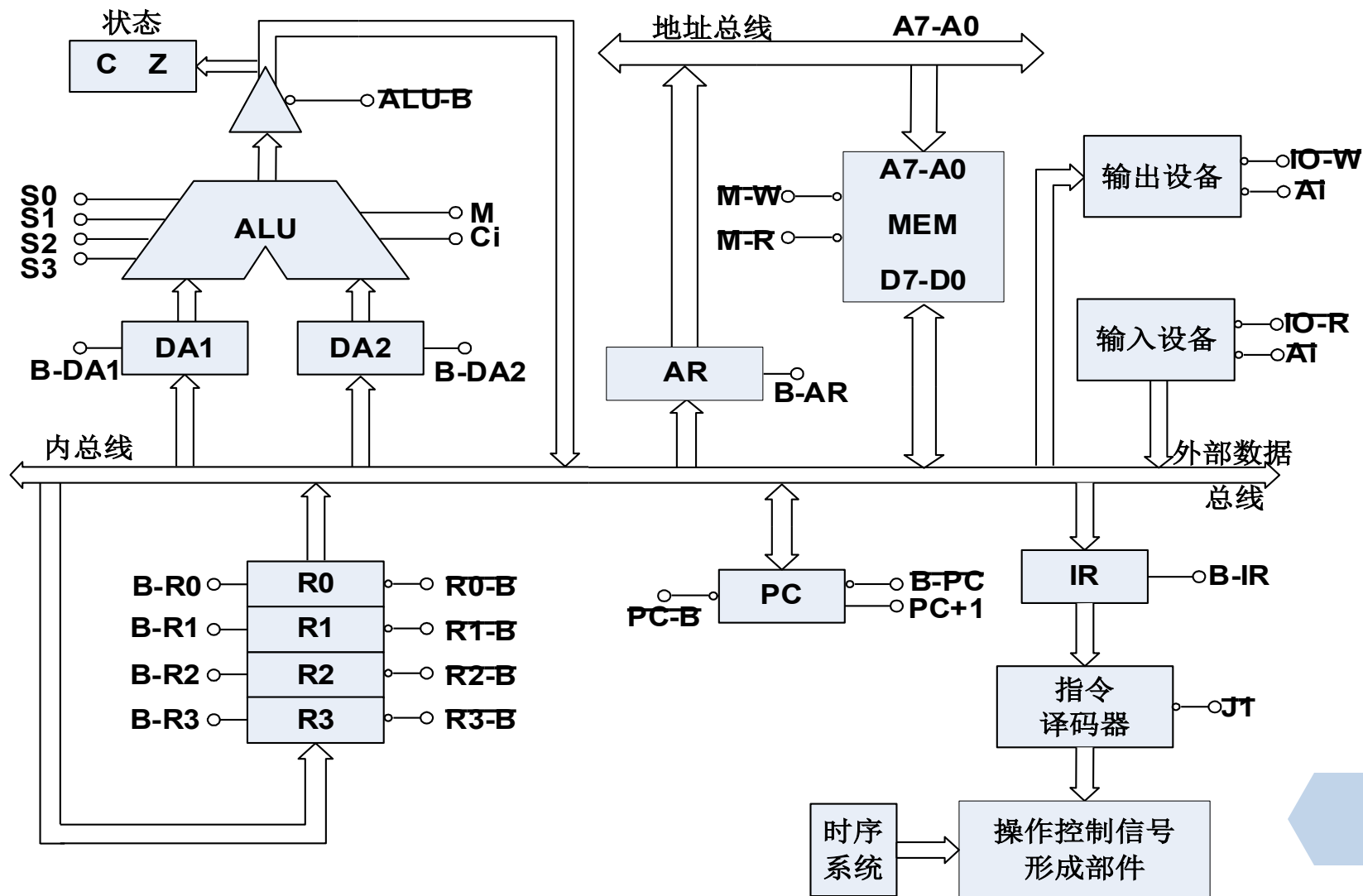
毫微程序设计






三、微程序设计技术

1. 模型机的结构





模型机特点；

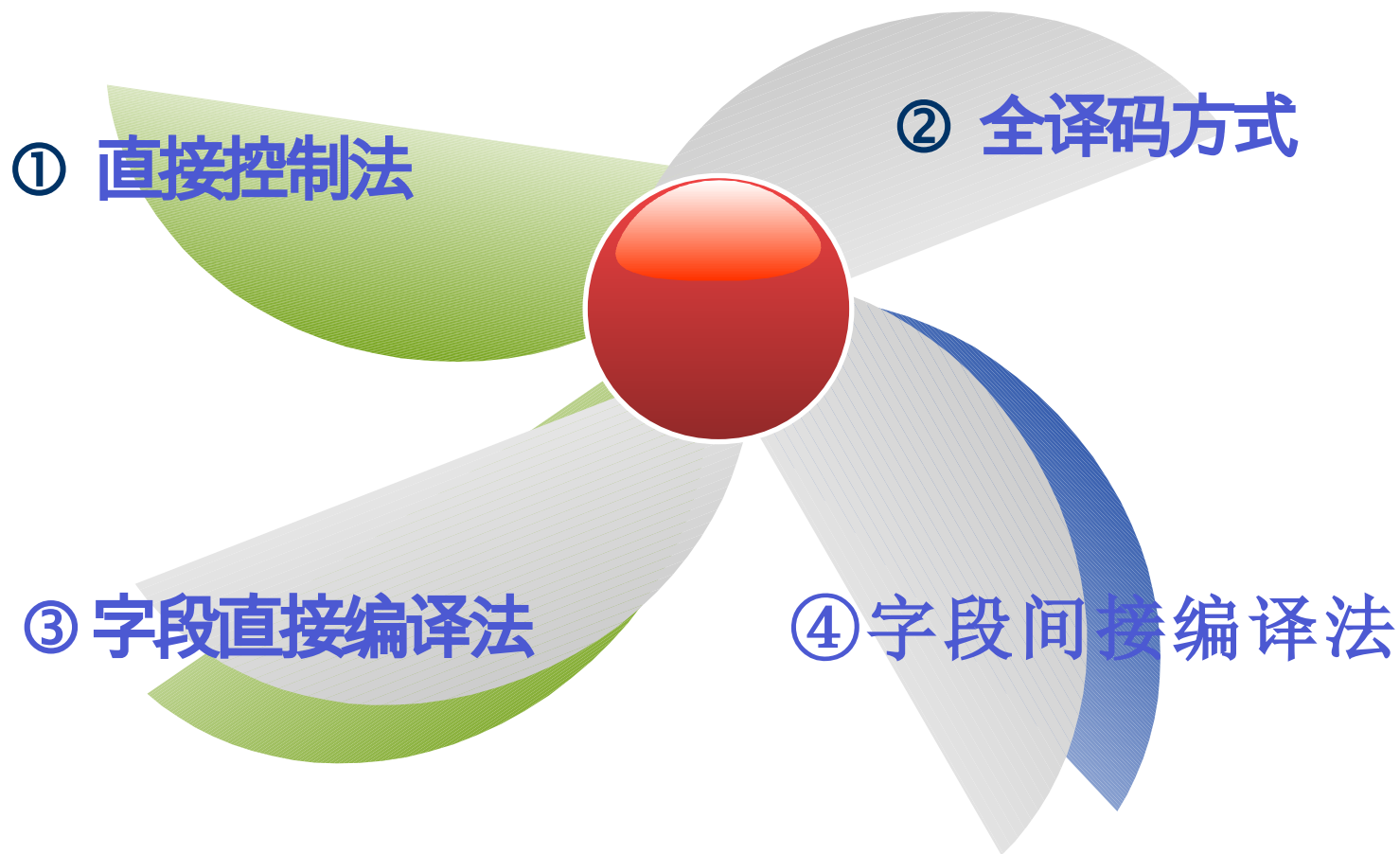
- 1 . 模型机以总线结构构建；
 - 2 . 总线宽度为 8 位二进制；
 - 3 . 数据和指令都通过总线传输；
 - 4 . 总线以分时复用的形式传输数据和指令，每一时刻只能有一个 8 位的二进制数据或指令进行传输（即：每一时刻只能有一个部件输出数据）；
 - 5 . 每个部件有相应的控制信号（微命令）控制其数据输入或输出；
 - 6 . ALU 部件可以实现 8bit 的加 / 减，+ 1，带进位加 / 减，逻辑运算和移位操作等各种运算（CPU 采用 74LS181，移位寄存器采用 74LS299）；
 - 7 . 设指令译码控制信号为 J1# ~ J5 #（低电平有效）
 - 8 . 中断控制
- 

2. 模型机控制信号（微命令表）

| 序号↕ | 控制信号↕ | 功能↕ | 序号↕ | 控制信号↕ | 功能↕ |
|-----|------------------|---|-------------------------------------|-------------------|----------------|
| 1↕ | PC-B#↕ | 指令地址（程序计数器）送总线↕ | 15↕ | ALU-B#↕ | 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↕ | 程序计数器内容加一↕ | 17↕ | B-R0↕ | 总线内容打入 R0 寄存器↕ |
| 4↕ | B-PC↕ | 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↕ | M-W#↕ | 存储器写↕ | 20↕ | B-R3 (B-SP)↕ | 总线内容打入 R3 寄存器↕ |
| 7↕ | M-R#↕ | 存储器读↕ | 21↕ | R0-B#↕ | R0 寄存器内容送总线↕ |
| 8↕ | S ₇ ↕ | S ₇ ~ S ₀ 选择 ALU16 种运算之一↕ | 22↕ | R1-B#↕ | R1 寄存器内容送总线↕ |
| 9↕ | S ₆ ↕ | 同上↕ | 23↕ | R2-B#↕ | R2 寄存器内容送总线↕ |
| 10↕ | S ₅ ↕ | 同上↕ | 24↕ | R3-B# (SP-B#)↕ | R4 寄存器内容送总线↕ |
| 11↕ | S ₄ ↕ | 同上↕ | <div>25</div> <div>J1# 指令译码控制</div> | | |
| 12↕ | M↕ | M 为“1”选择 ALU 做逻辑运算， M 为“0”选择 ALU 做算术运算↕ | | | |
| 13↕ | B-DA1↕ | 总线内容打入暂存器 DA1↕ | | | |
| 14↕ | B-DA2↕ | 总线内容打入暂存器 DA2↕ | | | |



1、微指令的编译法





① 直接控制法

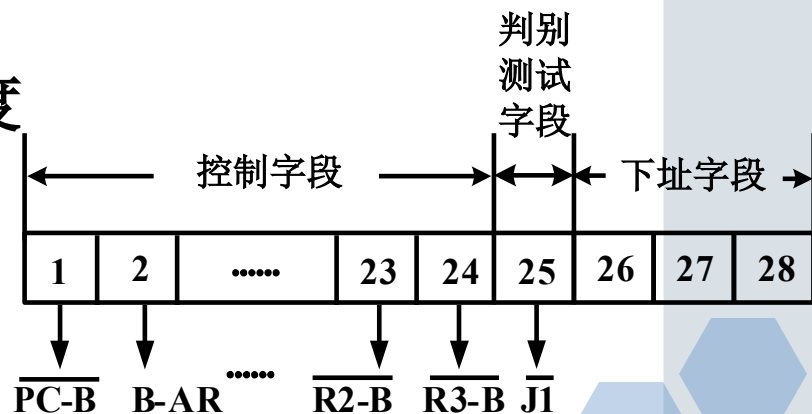
I. 在微指令的控制字段中，每一位代表一个微命令（控制信号），这就是**直接控制法**。

a) 在设计微指令时，如果要发出某个微命令则将控制字段中**对应位置**有效，这样就可以打开或关闭某个控制门

b) 如果是编码控制则置**相应编码值**。

II. **优点**：无需译码，执行速度快；微程序较短。

III. **缺点**：微指令的控制字段太长





② 全译码方式

- ❖ 控制字段的编码方法：将所有的控制信号进行编码，作为控制字段。在执行微指令时，译码产生各个微命令。
 - ✓ 每条微指令只能发送 1 ~ 2 个微命令。
- ❖ 优点：微指令字长很短。
- ❖ 缺点：并行操作能力弱，微程序很长，执行速度慢
- ❖ 一般用于垂直微指令格式。





③ 字段直接编译法

I. 控制字段的编码方法：将控制字段分成

若干个微命令，每个微命令通过编码 / 译码对

可以提高信息位的利用率，缩短微指令字

有利于实现并行操作，加快指令的执行速度

II. 优点：并行操作能力较强，字长扩展方便

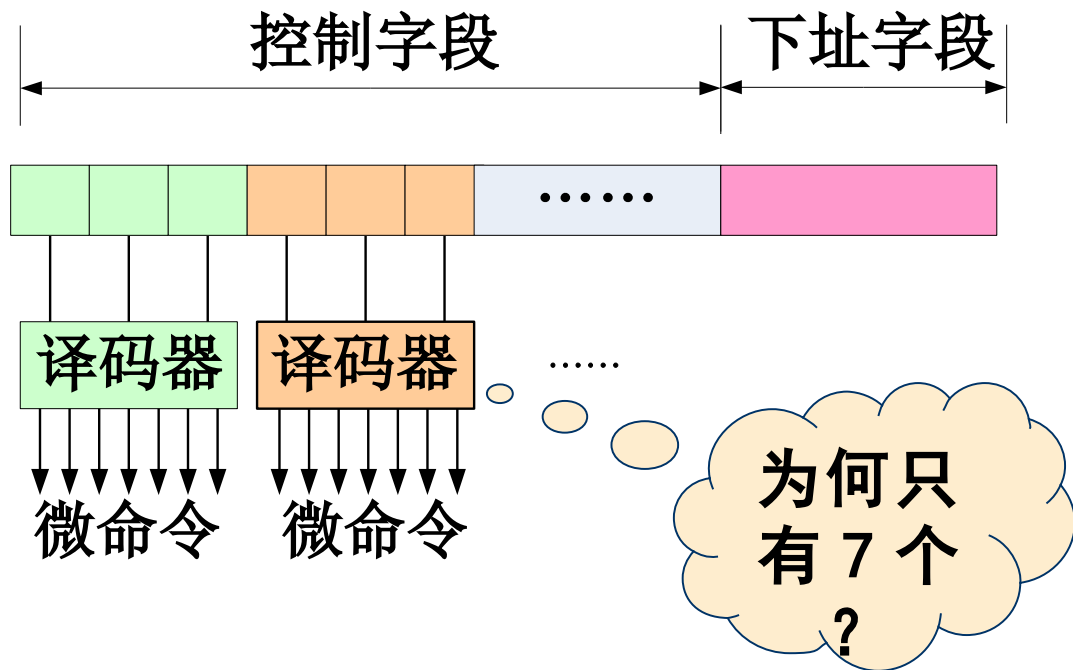
III. 字段直接编译法的基本分段原则是：

- ✓ 相斥性微命令分在同一字段内，相容性微命令分在不同字段内。





③ 字段直接编译法



- ✓ **相斥性微命令**：指在同一个微周期中**不可能**同时出现的微命令。
- ✓ **相容性微命令**：指在同一个微周期中可以同时出现的微命令





用字段直接编译法，重新设计微指令格

- 24 个控制信号中，将总线数据送目的部件的 **7 个控制信号** 组成 1 个字段（BTO）编码和译码，将部件数据送总线的 **4 个控制信号** 组成 1 个字段（OTB）编码和译码
- 11 个控制信号：从直接控制的 11 位缩短到了字段直接译码的 6 位。
- 去掉了 B-R1、B-R2、R3-B#
- 微指令字长从 28 位缩短

| | | | |
|----------------------|----------------------|----------|-------|
| $M_{16} \sim M_{14}$ | $M_{13} \sim M_{11}$ | M_{10} | M_9 |
| BTO | OTB | PC+1 | S_3 |

| 编码 + 译码 | BTO | OTB |
|---------|-------|--------|
| 000 | | |
| 001 | B-DA1 | ALU-B# |
| 010 | B-DA2 | PC-B# |
| 011 | B-IR | R0-B# |
| 100 | B-AR | M-R# |
| 101 | B-R0 | |
| 110 | M-W# | |
| 111 | B-PC# | |



字段直接编译法设计的微程序

| 微地址 | 微指令发出的微操作信号 | 判别测试字段 (J1#) | 下址字段 |
|-----|--|----------------|------|
| 000 | M0 : PC-B#,B-AR,PC+1 | 1 | 001 |
| 001 | M1 : M-R# ,B-IR,J1# | 0 | xxx |
| 010 | JMP•M2 : PC-B#,B-AR,PC+1 | 1 | 011 |
| 011 | JMP•M3 : M-R# , B-PC#,PC+1 | 1 | 000 |
| 100 | ADD•M3 : M-R# , B-DA1 | 1 | 101 |
| 101 | ADD•M4 : R0-B#,B-DA2 | 1 | 111 |
| 110 | ADD•M2 : PC-B#,B-AR,PC+1 | 1 | 100 |
| 111 | ADD•M5 : $S_3S_2S_1S_0MC_i=100101$, ALU-B#,B-R0 | 1 | 000 |



字段直接编译法设计的微程序

| 微地址 | $M_{16} \sim M_{14}$ | $M_{13} \sim M_{11}$ | M_{10} | M_9 | M_8 | M_7 | M_6 | M_5 | M_4 | M_3 | $M_2 \sim M_0$ |
|-----|----------------------|----------------------|----------|-------|-------|-------|-------|-------|-------|-------|----------------|
| | BTO | OTB | PC+1 | S_3 | S_2 | S_1 | S_0 | M | C_i | J1# | 下址 |
| 00H | 100 | 010 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 001 |
| 01H | 011 | 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | *** |
| 02H | 100 | 010 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 011 |
| 03H | 111 | 100 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 000 |
| 04H | 001 | 100 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 101 |
| 05H | 010 | 011 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 111 |
| 06H | 100 | 010 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 100 |
| 07H | 101 | 001 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 000 |