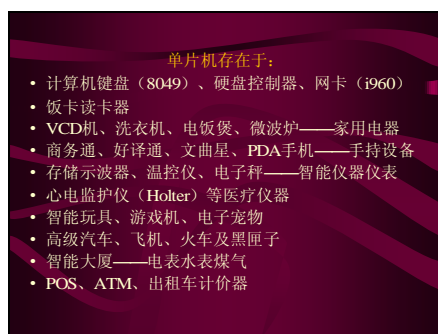


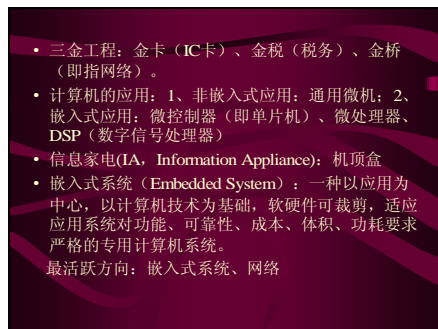
幻灯片 1



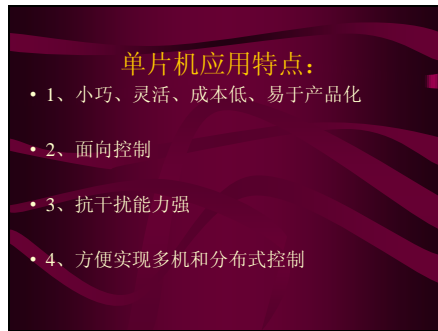
幻灯片 2



幻灯片 3



幻灯片 4



单片机应用特点:

- 1、小巧、灵活、成本低、易于产品化
- 2、面向控制
- 3、抗干扰能力强
- 4、方便实现多机和分布式控制

幻灯片 5



单片机现状: 微机产量80%

- 1、4位机
NEC公司 μ PD75XX系列和NS公司COP400
INTEL 4040
- 2、8位机
INTEL MCS-51系列 (PHILIPS, ATMEL, WINBOND)
MOTOROLA 68HCH
MICROCHIP PIC16XX
ZILOG Z8
NEC公司 μ PD 78XX

幻灯片 6



- 3、16位机
INTEL MCS-96系列
NS HPC16040
NEC 783XX
PHILIPS XA系列
- 4、32位机
INTEL 386EX, StrongARM, xScale
MOTOROLA 68KMX1

幻灯片 7

单片机的编程规范化 网络化发展

- 实时多任务操作系统RTOS (Real-Time Operating System) :
VxWorks, PSOS, QNX, WindowsCE
- 现场总线技术Field Bus:
LonWorks Local Network
CAN Bus Control Area Network

幻灯片 8

第一章单片机基础知识

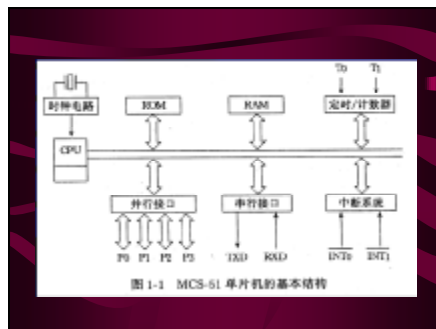
1.1 8051单片机的特点

一、单片机的概念 (*)

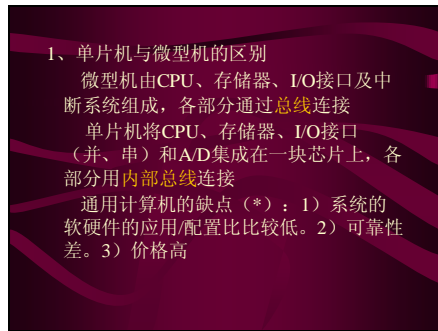
在一块硅片上集成了中央处理器CPU、数据存储器RAM、程序存储器ROM、定时器/计数器和多种I/O接口电路的微型计算机即为单片机 (microcontroller)

因为它是为了实时控制应用而设计制造, 所以又称为微控制器。

幻灯片 9



幻灯片 10

A dark blue slide with white text. The text is organized into a list and paragraphs. The background has a subtle pattern of wavy lines.

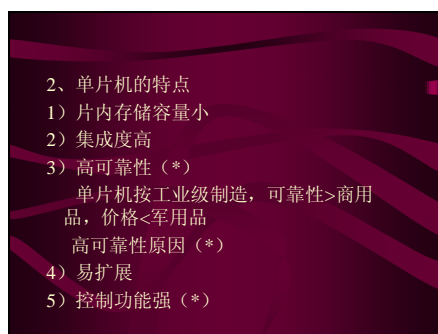
1、单片机与微型机的区别

微型机由CPU、存储器、I/O接口及中断系统组成，各部分通过**总线**连接

单片机将CPU、存储器、I/O接口（并、串）和A/D集成在一块芯片上，各部分用**内部总线**连接

通用计算机的缺点（*）：1）系统的软硬件的应用/配置比较低。2）可靠性差。3）价格高

幻灯片 11

A dark blue slide with white text. The text is organized into a list and paragraphs. The background has a subtle pattern of wavy lines.

2、单片机的特点

- 1) 片内存储容量小
- 2) 集成度高
- 3) 高可靠性（*）

单片机按工业级制造，可靠性>商用品，价格<军用品

高可靠性原因（*）

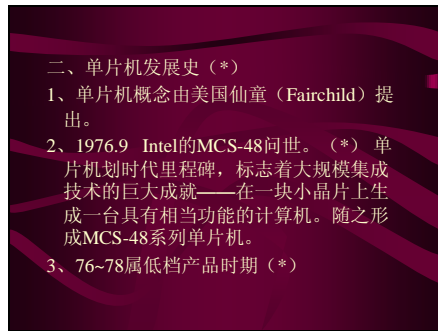
- 4) 易扩展
- 5) 控制功能强（*）

幻灯片 12

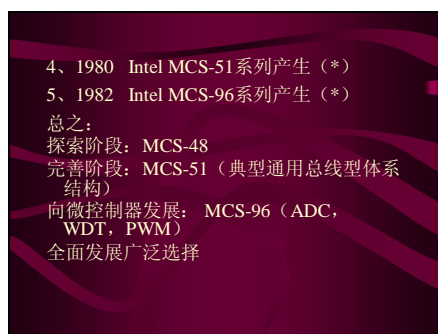
A dark blue slide with white text. The text is organized into a list. The background has a subtle pattern of wavy lines.

- 6) 性能价格比高
- 7) 低功耗
- 8) 保密性好*

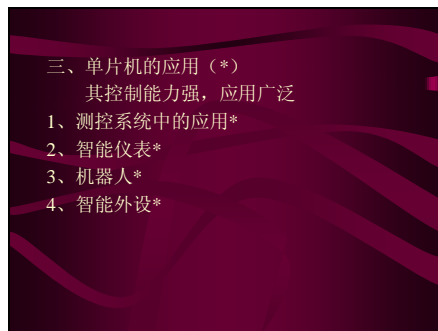
幻灯片 13



幻灯片 14



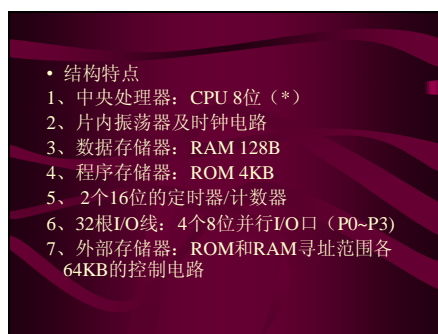
幻灯片 15



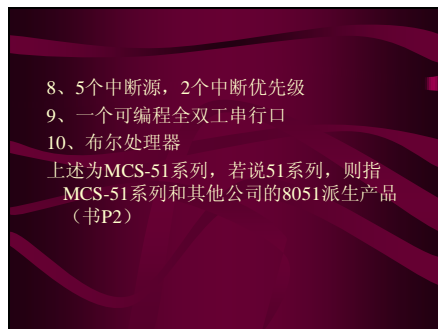
幻灯片 16



幻灯片 17



幻灯片 18



幻灯片 19

总结与综述:
 MCS-51 INTEL 1980年
 单片机标志:
 MCS-48, MCS-51, MCS-96(16位)
 8位机: 8051系列 教学首选

- 8051掩膜
- 8031无ROM, EPROM, FLASH
- 8751EPROM

低功耗基本型:
 • 80C51, 80C31, 87C51

幻灯片 20

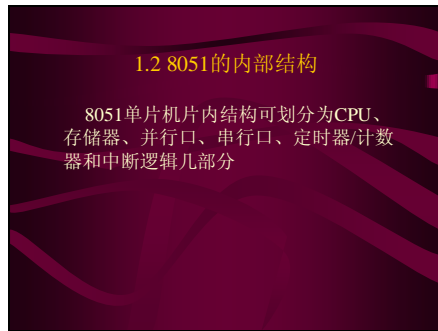
80年代中期专利互让的形式
 51系列衍生产品

- Atmel 89C51, 89C52, 89C2051
- Philips 80C51, 80C552, 87C752
- Dallas 80C390, 80C400
- Infineon C517, C509, 80C537
- ADI ADuC812, ADuC824
- TI MSC1210
- Cygnal C8051F

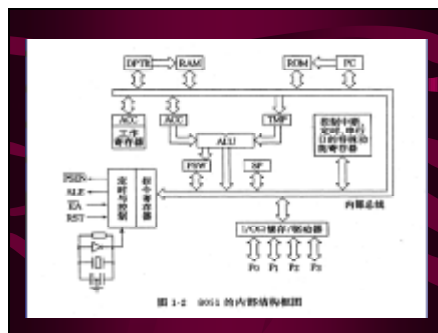
幻灯片 21

	AT89C51	AT89C52
闪存	4KB	8KB
内存	128B	256B
工作频率	24MHz	24MHz
输入/输出线	32	32
定时/计数器	2	3
中断源	5	8
串行口	1	1

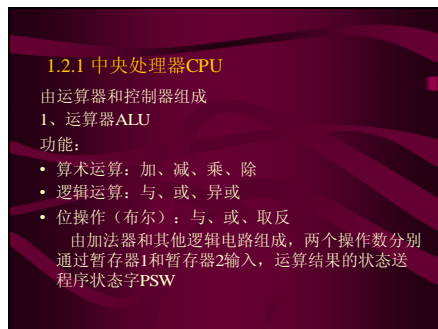
幻灯片 22



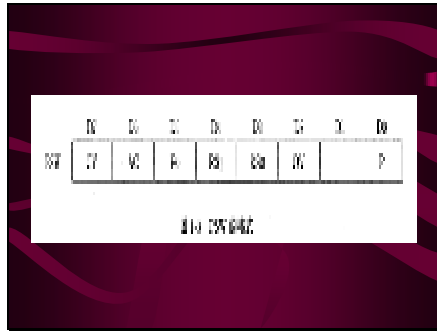
幻灯片 23



幻灯片 24



幻灯片 25



幻灯片 26

2、控制器、时钟电路和基本时序周期

控制逻辑包括：定时和控制逻辑、指令寄存器、译码器、地址指针DPTR、程序计数器PC

单片机工作过程(*)

8051控制器功能：在单片机内部协调各功能部件之间的数据传送、数据运算等操作，并对单片机发出若干控制信息。

幻灯片 27

1) 程序计数器 (PC)

16位，存放即将执行的指令地址。

功能：CPU工作时可根据PC内容去程序存储器中到对应的地址取指令代码。

可顺序递增或被赋予新的值

2) 指令寄存器IR (*)

3) 指令译码器ID (*)

4) 数据指针DPTR (*)

16位地址寄存器

幻灯片 28

5) 定时与控制部件: 产生CPU所需的机器时钟 (*)

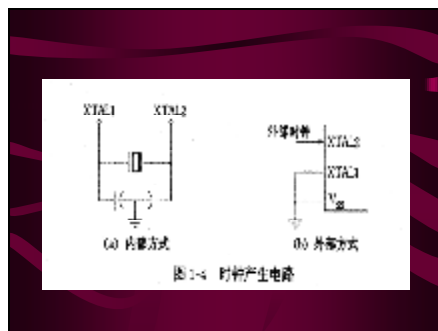
内部高增益放大器引脚XTAL1和XTAL2

时钟产生方式:

内部方式: 石英晶体晶振

外部方式: 外部振荡信号作8051时钟

幻灯片 29



幻灯片 30

6) 时序 (*)

概念: 一条指令译码产生的一系列微操作信号在时间上有严格的先后次序, 这种次序就是计算机的时序。

- a、振荡周期
- b、时钟周期
- c、机器周期
- d、指令周期

幻灯片 31

- 振荡周期: $1/f_{OSC}$
- 时钟周期: $2/f_{OSC}$
- 机器周期: $12/f_{OSC}=T$
- 指令周期: $1\sim 4T$ (*)

幻灯片 32

1.2.2 存储器组织

存储器特点 (*):

- 程序存储器 / 分开哈佛型
- 数据存储器 / 合并普林斯顿型

幻灯片 33

1.2.2.1 程序存储器 (只读) (*)

最大64K, 放程序和始终要保留的常数

1、程序存储器分布

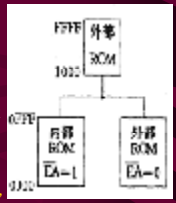
- 1) 内部 (片上): 4K
- 2) 外部 (扩展): 64K
 - 0000H~0FFFH 4K
 - 1000H~FFFFH 60K
- 3) 用PC作为地址指针, 通过16位地址总线

幻灯片 34

4) 8031上无内部程序存储器, 8051有4K ROM, 8751有4K EPROM作为程序存储器

5) 内外部的低4K空间地址重叠, 不能同时使用
 /EA=1: 内部
 /EA=0: 外部

8031无内部程序存储器, 其EA应接地(*) (当PC超过4KB, 自动转1000H~FFFFH (片外))



幻灯片 35

2、程序存储器使用时注意的问题 (*)

- 1) 注意/EA是否接地
- 2) 编程时用户主程序应放在0030H后

例: `ORG 0000H`
`LJMP min`
`ORG 0030H`
`min: NOP`
`NOP`
`LJMP min`

幻灯片 36

1.2.2.2 数据存储器 (可读写) (*)

放程序运行中所需的常数或变量

- 1、外部64K (movx):
0000H~FFFFH
- 2、内部数据存储器
分为物理上独立且性质不同的几个区 (256B, mov)



幻灯片 37

片内部分2块:

- 00~7FH: 128B, RAM区
- 80H~FFH: 128B, 特殊功能寄存器区 (SFR区) 或8032、8052的RAM区

The diagram shows a memory map with address ranges 00, 08, 10, 18, 20, 28, 30, 38, 40, 48, 50, 58, 60, 68, 70, 78, 80, 88, 90, 98, 100, 108, 110, 118, 120, 128, 130, 138, 140, 148, 150, 158, 160, 168, 170, 178, 180, 188, 190, 198, 200, 208, 210, 218, 220, 228, 230, 238, 240, 248, 250, 258, 260, 268, 270, 278, 280, 288, 290, 298, 300, 308, 310, 318, 320, 328, 330, 338, 340, 348, 350, 358, 360, 368, 370, 378, 380, 388, 390, 398, 400, 408, 410, 418, 420, 428, 430, 438, 440, 448, 450, 458, 460, 468, 470, 478, 480, 488, 490, 498, 500, 508, 510, 518, 520, 528, 530, 538, 540, 548, 550, 558, 560, 568, 570, 578, 580, 588, 590, 598, 600, 608, 610, 618, 620, 628, 630, 638, 640, 648, 650, 658, 660, 668, 670, 678, 680, 688, 690, 698, 700, 708, 710, 718, 720, 728, 730, 738, 740, 748, 750, 758, 760, 768, 770, 778, 780, 788, 790, 798, 800, 808, 810, 818, 820, 828, 830, 838, 840, 848, 850, 858, 860, 868, 870, 878, 880, 888, 890, 898, 900, 908, 910, 918, 920, 928, 930, 938, 940, 948, 950, 958, 960, 968, 970, 978, 980, 988, 990, 998, 1000. Internal RAM is shown from 00 to 7FH. SFR is shown from 80H to FFH. External RAM is shown from 00 to 00H.

幻灯片 38

内部RAM (低128B) (*)

- 通用寄存器区: 4组 (R0~R7)
- 可位寻址区: 20H~2FH (16个)
- 用户RAM

幻灯片 39

The diagram shows a memory map with address ranges 00, 08, 10, 18, 20, 28, 30, 38, 40, 48, 50, 58, 60, 68, 70, 78, 80, 88, 90, 98, 100, 108, 110, 118, 120, 128, 130, 138, 140, 148, 150, 158, 160, 168, 170, 178, 180, 188, 190, 198, 200, 208, 210, 218, 220, 228, 230, 238, 240, 248, 250, 258, 260, 268, 270, 278, 280, 288, 290, 298, 300, 308, 310, 318, 320, 328, 330, 338, 340, 348, 350, 358, 360, 368, 370, 378, 380, 388, 390, 398, 400, 408, 410, 418, 420, 428, 430, 438, 440, 448, 450, 458, 460, 468, 470, 478, 480, 488, 490, 498, 500, 508, 510, 518, 520, 528, 530, 538, 540, 548, 550, 558, 560, 568, 570, 578, 580, 588, 590, 598, 600, 608, 610, 618, 620, 628, 630, 638, 640, 648, 650, 658, 660, 668, 670, 678, 680, 688, 690, 698, 700, 708, 710, 718, 720, 728, 730, 738, 740, 748, 750, 758, 760, 768, 770, 778, 780, 788, 790, 798, 800, 808, 810, 818, 820, 828, 830, 838, 840, 848, 850, 858, 860, 868, 870, 878, 880, 888, 890, 898, 900, 908, 910, 918, 920, 928, 930, 938, 940, 948, 950, 958, 960, 968, 970, 978, 980, 988, 990, 998, 1000. Internal RAM is shown from 00 to 7FH. SFR is shown from 80H to FFH. External RAM is shown from 00 to 00H.

幻灯片 40

1.通用寄存器区(*)

4个组

- 0区00H~07H
- 1区08H~0FH
- 2区10H~17H
- 3区18H~1FH

由PSW中的RS1,RS0来决定用哪个工作区
(00,01,10,11)

设置4个工作寄存器区的原因

幻灯片 41

0区		1区		2区		3区	
地址	R	地址	R	地址	R	地址	R
00H	R0	08H	R0	10H	R0	18H	R0
01H	R1	09H	R1	11H	R1	19H	R1
02H	R2	0AH	R2	12H	R2	1AH	R2
03H	R3	0BH	R3	13H	R3	1BH	R3
04H	R4	0CH	R4	14H	R4	1CH	R4
05H	R5	0DH	R5	15H	R5	1DH	R5
06H	R6	0EH	R6	16H	R6	1EH	R6
07H	R7	0FH	R7	17H	R7	1FH	R7

幻灯片 42

例: 若程序分三段, 在1段R0工作在0区
(00H), 在2段R0中的内容要改变, 在
3段程序中要用到1段中的R0的内容(0区
中A*B, 结果送R0, 1区中02H送R0)

幻灯片 43

```

CLR RS0
CLR RS1
MOV A, #02H
MOV B, #03H
MUL AB
MOV R0, A
SETB RS0
MOV R0, #02H
INC R0
MOV A, R0
CLR RS0
MOV B, R0
ADD A, B
END
    
```

幻灯片 44

2. 可位寻址区 (*)

- 位寻址区范围: 20~2FH, 16字节
- 位地址范围: 00~7FH, 128位

幻灯片 45

表 8-3 位寻址区与位地址

位寻址区	Op	Op	Op	Op	Op	Op	Op
20H	21H	22H	23H	24H	25H	26H	27H
28H	29H	2AH	2BH	2CH	2DH	2EH	2FH
30H	31H	32H	33H	34H	35H	36H	37H
38H	39H	3AH	3BH	3CH	3DH	3EH	3FH
40H	41H	42H	43H	44H	45H	46H	47H
48H	49H	4AH	4BH	4CH	4DH	4EH	4FH
50H	51H	52H	53H	54H	55H	56H	57H
58H	59H	5AH	5BH	5CH	5DH	5EH	5FH
60H	61H	62H	63H	64H	65H	66H	67H
68H	69H	6AH	6BH	6CH	6DH	6EH	6FH
70H	71H	72H	73H	74H	75H	76H	77H
78H	79H	7AH	7BH	7CH	7DH	7EH	7FH

幻灯片 46

*: 1、2FH的D3位对应的位地址是多少?
若 (2FH) = 28H, 则2FH.3=?

2、mov C, 00H
mov A, 00H
(20H) = 55H, (00H) = 55H, 则C,
A的值分别为多少?

3、mov C, 20H
mov A, 20H
(24H) = 55H, (20H) = 33H

幻灯片 47

3. 用户RAM (数据缓冲区、堆栈区、数据区)

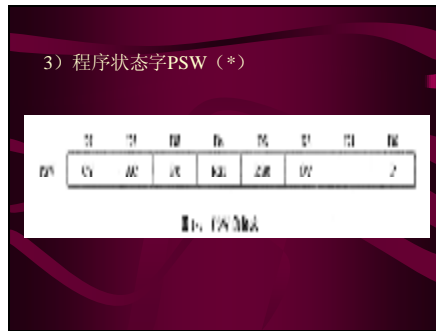
- 30H~7FH
- 堆栈, 向上增长

幻灯片 48

4. 专用寄存器区 (特殊功能寄存器)

- 位于内部RAM的80H~FFH, 只能采用直接寻址方式
- 除PC和4组R0~R7外其他都是SFR,
- 有的寄存器可以进行位操作, 有的不行。可位寻址的SFR其地址可被8整除 (即其字节地址的低位非0即8)

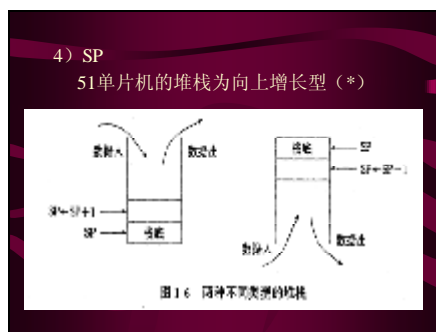
幻灯片 52



幻灯片 53

- CY: 进位标志。算术运算时最高位有进位/借位时硬件自动置1, 软件也可直接对其进行操作。
- AC: 半进位标志。D3->D4进位/借位时置1, 软件不能对其进行操作。
- FO: 状态标记, 用软件置位或清零
- RS1,RS0: 通用寄存器选择位
- OV: 溢出标志。带符号数运算结果超出-128~+127、无符号数乘法结果超过255或除数为0时置1, 否则OV=0
- P: 奇偶标志。每条指令执行完, A中的1的个数为奇数时置1, 偶数时置0。常用于串行通讯的奇偶校验

幻灯片 54



幻灯片 55

- 初始时指向栈底, 初始值=07H, 事实上从08H单元开始放数
- 注: 初始化程序中应对SP修改 (*)

```
MOV SP, #30H
```

幻灯片 56

5) DPTR数据指针 (*)

16位, 也可分成两个8位的寄存器DPH, DPL。

```
MOV DPH, #05H  
MOV A, DPH ; (A) = 05H
```

当DPTR放16位对64KB外部数据存储器寻址时, 可作为间址寄存器用

```
MOV DPTR, #1234H  
MOVX A, @DPTR  
(1234H)=08H, 则 (A) = 08H
```

幻灯片 57

当DPTR放16位对程序存储器访问时, 可作为基址寄存器用

```
MOVC A, @A+DPTR  
A<=((A)+(DPTR))
```

注: 在中断服务程序中, 若要将DPTR中内容压栈时, 应分为高8位 (DPH) 和低8位 (DPL) 分别压栈 (因为栈指针是8位寄存器)

幻灯片 58

与通用微机不同的特点

- 程序存储器和数据存储器严格分开
- 特殊功能寄存器和内部数据存储器统一编址

幻灯片 59

1.2.3 片内并行接口

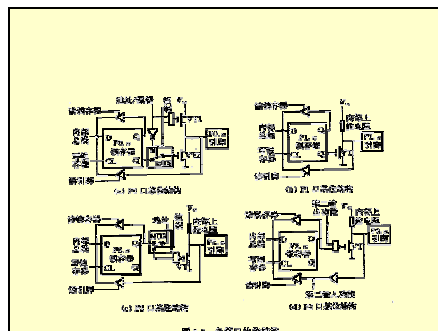
8051有4个8位并行接口, P0、P1、P2、P3, 共32根I/O线。

每个口有4部分: 端口锁存器, 输入缓冲器, 输出驱动器, 端口引脚

它们是准双向口, 每条I/O线均能独立的作用输入或输出

作输出数据时可以锁存, 作输入数据时可以缓冲。

幻灯片 60



幻灯片 61

1、端口功能

- 1) P0口: 输入/输出
 - a. 当利用单片机制作简单的控制系统时, P0口可作为一般的输入/输出
 - b. 当利用单片机制作复杂的控制系统时, P0口可作为数据I/O口和地址的低8位的输出/输入, 分时工作。
- 2) P1口: 功能没有P0口强, 作为一般的数据输入, 输出, 按位可编程的I/O口
- 3) P2口: 简单系统中作为一般的数据I/O口, 复杂系统中, 仅作为地址的高8位输出, 和P0一起组成16位地址总线。
- 4) P3口: 双功能口。第一功能为一般I/O口, 第二功能为特殊功能。

幻灯片 62

2、端口操作 (*)

P0口

介绍图中元器件。图中为P0口的一位结构。模拟开关的位置由来自CPU的控制信号决定。

幻灯片 63

控制信号为低电平: 开关与/Q相连, P0用作一般的I/O口。

控制信号为高电平: 开关打向上方, P0作为地址/数据分时使用。

当P0作为输出/输入使用时, 应外接上拉电阻

Q) P0口某一位结构

幻灯片 64

工作过程 (*):

1) 作为一般输出时: CPU先在控制线上加一低电平, 使MUX开关与锁存器/Q相连, 然后再将输出的数据送到总线上, 最后向锁存器发出一个写脉冲, 锁存器就将数据锁存起来, 并通过MUX, T2将该数输出

幻灯片 65

工作过程 (*):

2) 作为地址输出时: CPU先在控制线上加一高电平, 将与门开锁, 并将MUX开关拨向上方, 然后CPU将地址信号送到地址线上, 通过与门, MUX, T1, T2将地址输出到引脚上

幻灯片 66

工作过程 (*):

3) 作为输入口时: 只是数据输入, 不能作为地址输入。

a、读引脚: CPU利用控制信号读引脚脉冲将三态门缓冲器打开, 此时P0.n信号将送至内部总线。

注: P0口为准双向口, 编程时要读取管脚上的内容时, 首先要向P0口送“FF” (即置1), 然后再进行读操作。

```

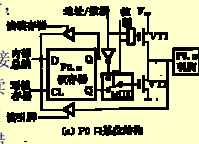
mov A, #0FFH
mov P0, A
mov A, P0 ; 读引脚
    
```

幻灯片 67

工作过程 (*) :

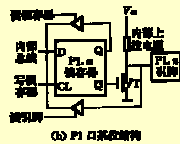
3) 作为输入口时:

b、读锁存器: CPU通过读锁存器使图中上方的三态门打开, 就将Q的值读入内部总线上。不直接读引脚上的数而读锁存器Q端上的数是为了避免可能错读引脚上的电平信号。



幻灯片 68

P1口: 一个准双向口, 作通用I/O口使用, 也有读引脚和读锁存器, 也可用于“读-修改-写”, 输入时, 先写入“FF”



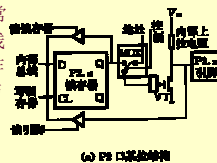
幻灯片 69

P2口 (准双向口)

在不接外部存储器或片外存储器容量小于256B的系统中, 由P0输出低8位地址, P2作I/O口; 反之, P0低8位, P2高8位。

对于8031单片机

来说, P2口通常只作为地址总线口使用, 而不作I/O口线直接与外部设备连接。



幻灯片 73

1.2.4 8051的内部资源 (*)

- 串行口
- 定时器/计数器
- 中断系统

幻灯片 74

1.2.5 8051的芯片引脚 (书P10)

CTDP1: 0	1	40	V _{CC}
CTDP1: 1	2	39	P0.0 AD0
P1: 0	3	38	P0.1 AD1
P1: 1	4	37	P0.2 AD2
P1: 2	5	36	P0.3 AD3
P1: 3	6	35	P0.4 AD4
P1: 4	7	34	P0.5 AD5
P1: 5	8	33	P0.6 AD6
P1: 6	9	32	P0.7 AD7
RST/VPP	10	31	EA/V _{PP}
RXD	11	30	ALE/PROG
TXD	12	29	PSEN
INT0	13	28	P2.7 A15
INT1	14	27	P2.6 A14
T0	15	26	P2.5 A13
T1	16	25	P2.4 A12
WR	17	24	P2.3 A11
RD	18	23	P2.2 A10
XTAL1	19	22	P2.1 A9
V _{SS}	20	21	P2.0 A8

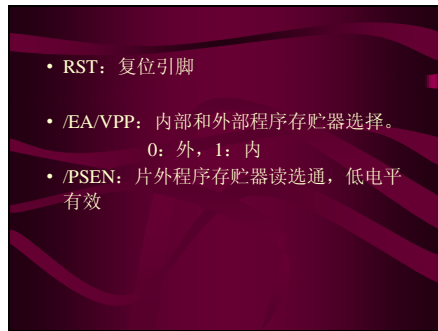
图 1-8 MCS-51引脚图

幻灯片 75

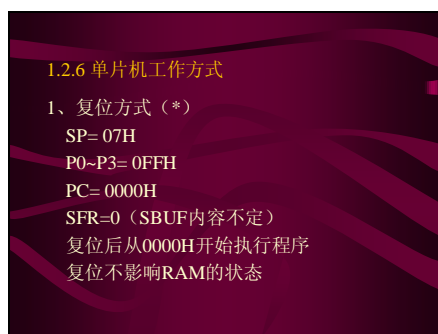
引脚功能

- VCC: 电源5V
- VSS: 接地
- XTAL1和XTAL2: 外接晶振引脚
- P0口: 地址/数据总线
- P1口: 准双向通用I/O口
- P3口: 多用途端口
- ALE/PROG: 地址锁存信号输出端

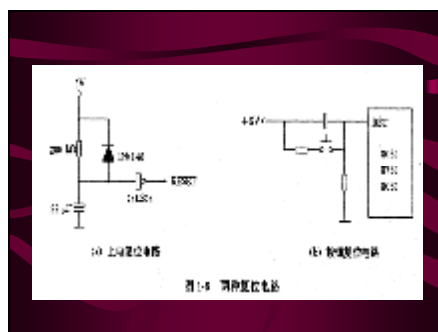
幻灯片 76



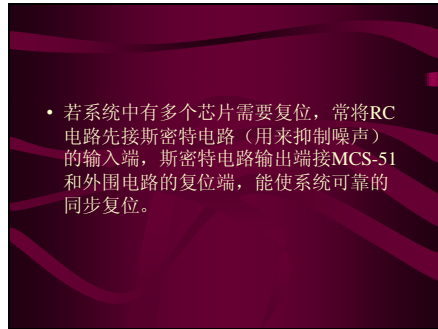
幻灯片 77



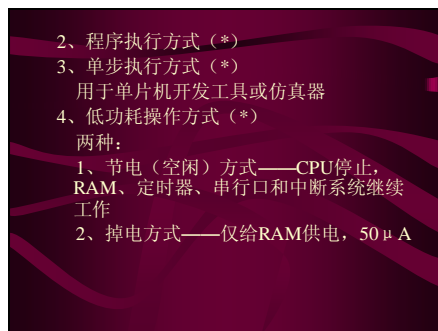
幻灯片 78



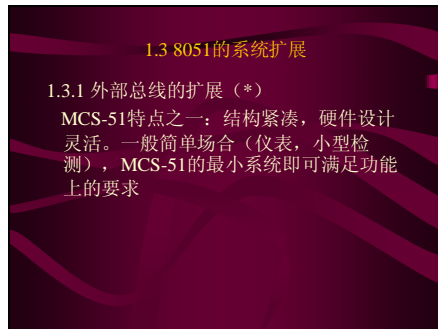
幻灯片 79



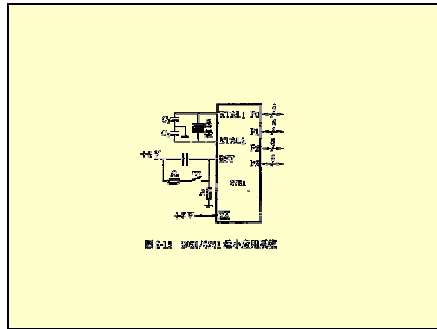
幻灯片 80



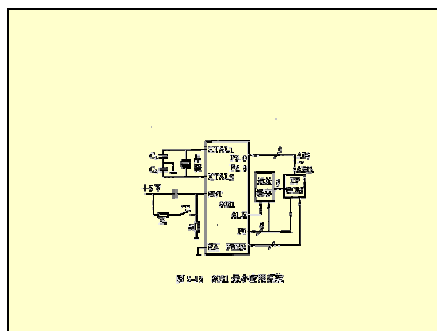
幻灯片 81



幻灯片 82



幻灯片 83



幻灯片 84

对于复杂的应用场合，需较大的存储器容量和较多I/O接口时，最小系统不行，则MCS-51可提供很强的扩展功能，可直接外接标准的存储器电路和I/O接口电路，以构成功能很强，规模较大的系统。

系统扩展的任务：1、把系统所需的外设与单片机连起来，使单片机系统能与外界进行信息交换，即I/O扩展。

2、扩展单片机容量

幻灯片 85

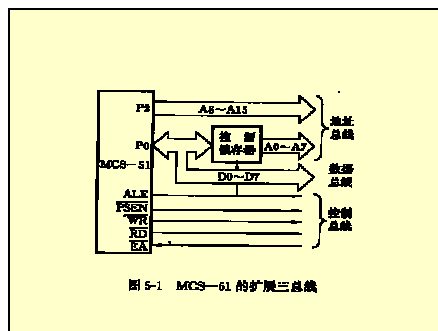
8051 无对外专用的地址总线和数据总线，在进行对外扩展存储器或I/O接口时，需要首先扩展对外总线。（*）

DB: 8位。P0实现，用于传送指令和数据信息（双向口）。

AB: 最大16位。P0: 低8位; P2: 高8位

CB: /RD, /WR, /PSEN, /EA, ALE

幻灯片 86



幻灯片 87

1.3.2 外部程序存储器的扩展 (*)

1、工作过程

1) 一般电路

2) 工作时序

2、EPROM接口设计

1) EPROM简介

a、2716 2K*8 A0-A10; 接P0和P2低3位, I/O-I/O7 (D0-D7); 接P0口 /CE: 片选 /OE: 数据输出选通线

b、2764 8K*8 A0-A12

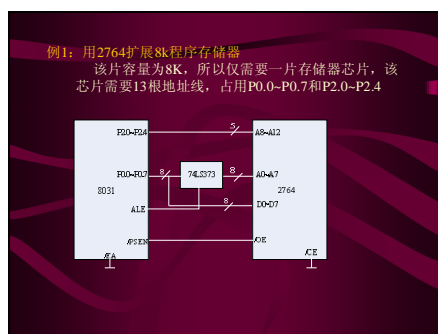
c、27256 32K*8 A0-A14

d、2732 27128

幻灯片 88

2、扩展方法
CPU提供三种信号线
DB: P0口接I/O0~I/O7
AB: P0口接A0~A7, P2口
CB: ALE接锁存器C端, /PSEN接/OE
扩展单片时, 应将存储芯片的/CE接地
扩展多片时: a、各片的EPROM的地址线并联, 利用CPU中地址剩余线通过译码器(或直接)接各个EPROM的/CE, 实现片选
b、各片EPROM数据线并联, 接P0口

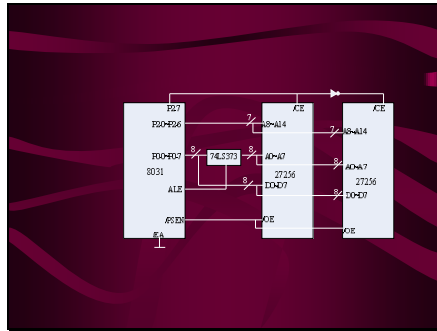
幻灯片 89



幻灯片 90

例2: 利用27256扩展64K程序存储器
每片容量为32K, 所以共需2片。每片的地址线需要15根。占用P0.0~P0.7和P2.0~P2.6
第一片的地址为: 0000H~7FFFH
第二片的地址为: 8000H~FFFFH

幻灯片 91



幻灯片 92

1.3.3 数据存储器的扩展设计

- 1、芯片引脚: /WE, /OE
- 2、工作时序
- 3、数据存储器扩展性能
 - 1) 数据存储器与程序存储器的地址可重叠0000H~FFFFH
 - 2) 数据存储器与程序存储器的数据总线也可并联(控制总线不同)
 - 3) 数据存储器应与其它外围设备(如A/D, D/A)统一编址

幻灯片 93

4、扩展方法

- 1) AB: 程序存储器和数据存储器共用
 - 2) DB: 8条, 和EPROM的8条数据线并联接到P0口
 - 3) CB: ALE接锁存器G端, /WR接/WE, /RD接/OE
- 5、RAM介绍
6116, 6264, 62256

幻灯片 94

5、RAM介绍

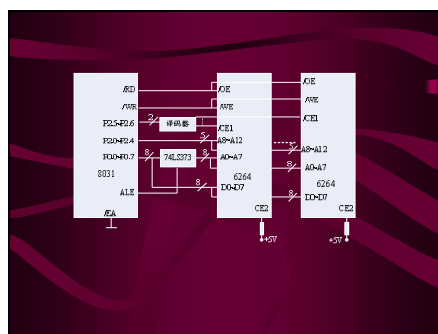
- 1) 6116
容量: 2K*8
引脚信号: A0-A10, /CE, /WE, /OE
I/O0-I/O7 (D0-D7)
- 2) 6264
容量: 8K*8
引脚信号: A0-A12, /CE1, CE2 (通常接高电平), /WE, /OE,
I/O0-I/O7 (D0-D7)
- 3) 62256

幻灯片 95

例: 利用6264设计容量为32K的外部数据存储

- 6264为8K, 所以共需4片。
- 每片的地址线为13根。占用P0.0~P0.7和P2.0~P2.4
- 用P2.5~P2.6作为片选
- 6264的CE2接高电平

幻灯片 96



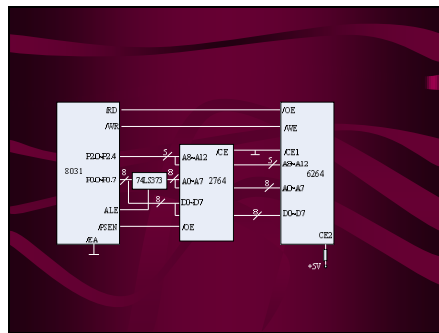
幻灯片 97

例: 利用2764和6264扩展8k程序存储器和8k数据存储器

- 1、6264和2764的地址线和数据线并联
- 2、控制线分开

程序存储器: /PSEN——/OE
 数据存储器: /WR——/WE
 /RD——/OE

幻灯片 98



幻灯片 99

总结

- 1、8051的三总线结构:
 - DB: 8位 P0实现
 - AB: P0与P2实现
 - CB: /RD, /WR, /PSEN, /EA, ALE
- 2、扩展多片数据存储器、程序存储器芯片时, 各片的数据线和地址线均并联, 由片选信号以及控制信号来选择哪一个芯片被选通进行工作。

幻灯片 100

1.4 指令系统

分类(*)：
指令构成：OP D
MCS-51根据其指令编码长短的不同有单、双、三字节指令。

1、单字节指令

1) 8位全表示操作码OP 例：NOP机器码为00H

2) 8位又有操作码，又有寄存器编码 例：
MOV A, Rn ; 11101....

幻灯片 101

2、双字节指令(*)
指令编码占两个字节，第一个为操作码，第二个为操作数
例：MOV A, #85H ; 机器码为：第一字节74H, 第二字节为85H

3、三字节指令(*)
第一字节为操作码，第二字节为第一操作数，第三个字节为第二操作数
例：MOV direct, #DATA
MOV 78H, #80H
第一字节：75H, 第二字节：78H, 第三字节：80H

幻灯片 102

书写格式(*)：

标号：操作码 操作数；注释
标号—第一个字母不能为数字，字母个数不能多于6个。且须用大写英文字母开始。标号可有可无，若有，则代表该指令第一个字节所存放的存储器单元的地址。所以标号又称为符号地址，汇编时把该地址赋值给标号。

操作数—可有0~3个，若操作数以字母开头，则前面应加0。多个操作数之间用“,”隔开。

幻灯片 103

1.4 指令系统

1.4.1 寻址方式 (寻找操作数的方式, 分类)

(*)

1、立即寻址

MOV A, #6FH ; A←6FH, 双字节指令, op为74H, 数为6FH, 两者相跟放在程序存储器中。

MOV DPTR, #1234H ; DPTR←1234H
三字节指令

注: 立即数用#data表示
寻址空间: 程序存储器

幻灯片 104

2、直接寻址 (*)

指令中含有操作数的直接地址, 该地址指出了参与操作的数据所在的字节地址或位地址。

此方式中操作数存储空间有三种:

- 1) 内部数据存储器 的低128个字节单元 (00H-7FH)
MOV A, 4FH; A←(4FH) 二字节
- 2) 位地址空间(*)
MOV C, 00H; C的位地址为D7H
- 3) 专用功能寄存器 (只能用直接寻址方式访问, 专用功能寄存器有专门的地址, 寄存器名只是一个代号, 给其送数, 其实是给一个地址送数)
MOV A, P0 MOV IE, #85H
寻址空间: 内部RAM低128字节, SFR

幻灯片 105

3、寄存器寻址

指令指出某一个寄存器中的内容为操作数, 指令的操作码中包含了参加操作的寄存器的编号

MOV A, R0; A←(R0)
ADD A, R0; A←(A)+(R0)
INC R0

寻址空间: R0~R7, A, B, C, DPTR, AB (乘法), 其R0~R7由操作码低三位的8种组合表示, A, B, C, DPTR则隐含在操作码中

幻灯片 106

4、寄存器间接寻址 (*)

1) 访问片内RAM或片外的低256字节空间时, 可用R0, R1作为间址寄存器。此类指令为单字节指令, 操作码的最低位表示是采用R0还是R1

MOV A, @R1; A<=((R1))
 (R1)=40H, A<=(40H)
 PUSH 30H; (SP)<=(30H)

MOV 60H, #3BH MOV R0, #60H MOV A, @R0
 则(A)=3BH

MOVX A, @R0; A<=((R0)) 外部数据存储器
 External——MOV P2, #10H P2 放高8位地址

幻灯片 107

2) 访问片外RAM时, 可用DPTR作间址寄存器, 它为16位, 可以访问片外整个64K的地址空间。

MOVX A, @DPTR; A<=((DPTR))

寻址空间: 内部RAM(@R0, @R1, @SP);
 外部RAM(@R0, @R1, @DPTR)

3) 执行push.pop时, 也可采用sp为间址寄存器

幻灯片 108

5、基址加变址寻址(*)

基址寄存器: DPTR, PC 变址寄存器: A

MOVC A, @A+DPTR
 ; A<=((DPTR)+(A)) 单字节

MOVC A, @A+PC
 ; PC=(PC)+1
 ; A<=((PC)+(A))

变址寻址方式只适用于8051的程序存储器, 用于读取数据表

幻灯片 109

6、相对寻址 (*)

PC内容为基地址, 指令的第二字节为偏移量, 两者相加, 和为跳转指令的转移目的地址

SJMP rel ;PC<=PC+2
PC<=PC+rel

SJMP 08H ; 双字节指令
(PC)+rel=>PC

幻灯片 110

7、位寻址

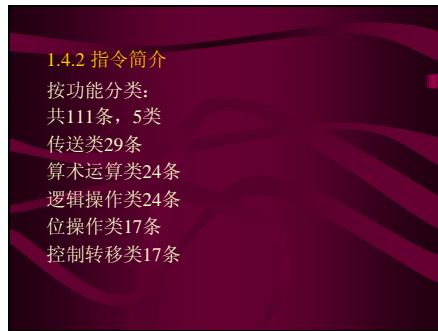
SETB bit ;(bit)<=1

寻址空间: 内部RAM可位寻址区;
SFR 可位寻址位

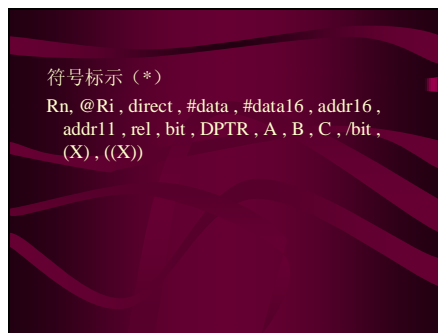
幻灯片 111

方式	利用的变量	寻址空间
立即寻址		程序存储器
直接寻址		片内RAM低128个字节, 专用寄存器, 片内RAM的位寻址空间
寄存器寻址	R0-R7,A,B, CY,DPTR	
寄存器间接寻址	@R0,@R1,SP	片内RAM低128个字节
	@R0,@R1, @DPTR	外部RAM
基址加变址	@A+PC,	程序存储器
	@A+DPTR	
相对寻址	PC+偏移量	程序存储器, 跳转范围256B
位寻址		片内RAM位寻址区和可位寻址的专用功能寄存器

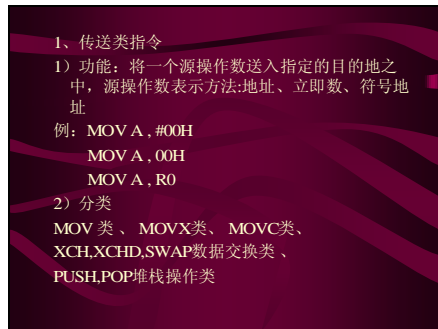
幻灯片 112



幻灯片 113



幻灯片 114



幻灯片 115

3) MOVX类指令

a、外部数据存储器地址由DPTR指向
MOVX A, @DPTR ; A<=((DPTR))
MOVX @DPTR, A ; ((DPTR))<=A

例: MOV DPTR, #2007H
MOVX A, @DPTR
INC DPTR
MOVX @DPTR, A
RET

幻灯片 116

• 例: 把片外数据存储器2040H单元中的数取出, 传送到3000H单元中

```
MOV DPTR, #2040H
MOVX A, @DPTR
MOV DPTR, #3000H
MOVX @DPTR, A
```

幻灯片 117

b、外部数据存储器地址由Ri和P2口指向

Ri: R0, R1指向地址低8位(*)

P2: 指地址高8位

MOVX A, @Ri ; A<=((Ri)+(P2))

MOVX @Ri, A ; ((Ri)+(P2))<=A

例: MOV P2, #20H
MOV R0, #07H
MOVX A, @R0
INC R0
MOVX @R0, A

幻灯片 118

总结 (*): 1、A与片外数据存储器的地址
传送通过P0和P2进行, 数据传送由P0传输
2、访问外部数据存储器只能用间址方式
3、只能和A进行传送
4、访问片外I/O口使用这四条指令, 因为
其统一编址

幻灯片 119

4) MOVC指令(*)
常用于查表, 数据表格可放在程序存储器
器中
a、MOVC A, @A+PC
;PC<=PC+1
;A<=((A)+(PC))
例: 1000H: MOVC A, @A+PC
(A)=30H
则(A)=?

幻灯片 120

例: 设MOVC指令放在3000H处, 试用
MOVC A, @A+PC将3041H中的内容送
入A中(*)
ORG 2FFEh
MOV A, #40H
MOVC A, @A+PC
RET

幻灯片 121

```
b(*), MOVC A, @A+DPTR
;A<=((A)+(DPTR))
例: MOV DPTR, #3041H
MOV A, #00H
MOVC A, @A+DPTR
RET
```

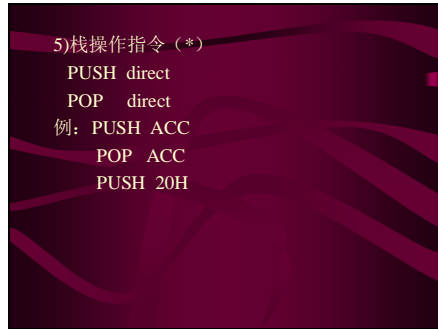
幻灯片 122

```
例: 累加器中的内容为一位BCD码, 用查表法获得相应的ASCII码
INC A
MOVC A, @A+PC
RET
TAB:DB 30H
DB 31H
DB 32H
DB 33H
.....
DB 39H
```

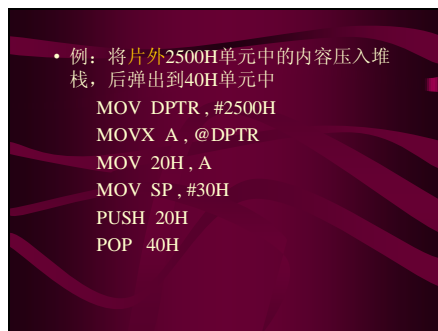
幻灯片 123

- MOVC只能用基址+变址方式, 且目的操作数只能为A

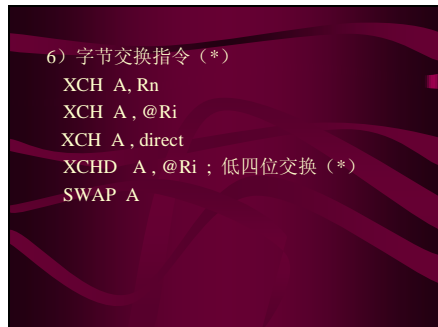
幻灯片 124



幻灯片 125



幻灯片 126



幻灯片 127

例:将20H单元的内容与A中内容互换, 而
后将A的高4位存入Ri指示的RAM单元中
的低4位, A的低4位存入该单元的高4位

```
XCH A, 20H
SWAP A
MOV @Ri, A
```

幻灯片 128

2、算术运算类(*)
其运算结果会影响CY, Ac, OV
但INC和DEC不影响

a、不带进位加法指令ADD

```
ADD A, Rn; ADD A, @Ri;
ADD A, direct; ADD A, #data
```

例:(A)=C3H, (R0)=20H, (20H)=AAH

```
ADD A, @R0
```

则Ac=0, CY=1, OV=1, (A)=6DH

幻灯片 129

b (*)、带进位加法ADDC

例: A=AEH, (20H)=81H CY=1

```
ADDC A, 20H
```

则: CY=1, Ac=1, OV=1, (A)=30H

多用于多字节数的加法运算

例: 有两个无符号16位数分别存于30H和
32H开始的单元中, 设

```
(30H)=AFH, (31H)=0AH, (32H)=90H,
(33H)=2FH, 高字节在高地址单元, 低字  
节在低地址单元, 算两数之和并存入32  
开始的单元中
```

幻灯片 130

```
CLR C
MOV R0, #32H
MOV A, 30H
ADD A, @R0
MOV @R0, A
MOV A, 31H
INC R0
ADDC A, @R0
MOV @R0, A
```

幻灯片 131

3、增量指令 INC

```
INC A INC R2 INC 45H INC @R0
INC DPTR
```

注: 1) 若原来为0FFH, 则加1后溢出为00H, 不影响任何标志

2) 对于 INC direct指令, 若direct是I/O端口 (P0~P3) 时, 执行的是读-修改-写操作 (*)

3) INC DPTR是唯一的一条16位加1指令, 执行过程中若低8位有进位可直接向高8位进位而不用通过CY传送。

幻灯片 132

4 (*)、十进制调整指令 DA

对A参与的BCD码加法所获得的8位结果进行十进制调整

```
DA A
```

注: 若A0~3>9或Ac=1, 则A<=(A)+#06H
若A4~7>9或CY=1, 则A<=(A)+#60H
若上述条件均成立, 则A<=(A)+#66H

例 (*): 十进制数68+53=121

注: DA A使用时一般跟在ADD和ADDC指令之后, 用来对加法 and 进行修正

幻灯片 133

例: 设计一个BCD加法程序, 设被加数放在32H, 31H, 30H中, 加数放在42H, 41H, 40H中, 和放在32H, 31H, 30H中

```
CLR C
MOV R0, #30H
MOV R1, #40H
MOV A, @R0
ADD A, @R1
DA A
MOV @R0, A
INC R0
INC R1
```

幻灯片 134

```
MOV A, @R0
ADDC A, @R1
DA A
MOV @R0, A
INC R1
INC R0
MOV A, @R0
ADDC A, @R1
DA A
MOV @R0, A
END
```

幻灯片 135

5、SUBB带借位减法

(Substret with Borrow)

```
SUBB A, Rn
```

没有不带借位标志的减法指令

```
SUBB A, 20H
```

; $A \leftarrow (A) - (20H) - CY$

其他用法与ADDC类似

幻灯片 136

6、减1指令
DEC
DEC A DEC R5 DEC 3EH DEC @R1
其中, DEC direct 与 INC用法类似
注: 不影响标志

幻灯片 137

7、乘法指令
MUL AB
实现两个8位无符号整数的乘法操作。结果在累加器A（低字节）和B寄存器（高字节）中
注: 若积>255, 则OV=1, 否则清零, 而CY总是为0
乘法指令是指令系统中执行时间最长的2条指令之一, 需要4个机器周期

幻灯片 138

8、除法
DIV AB
A/B, 把A中的8位无符号整数除以B中的8位无符号整数, 结果商在ACC中, 余数(不是小数部分)在B中
注: CY=0, OV=0, 只有当除数为0时, A和B中的内容为不确定值, 此时OV=1, 除法溢出
除法指令是指令系统中执行时间最长的2条指令之一, 需要4个机器周期

幻灯片 139

例: 把A中的二进制数转换成BCD码(三位), 要求百位放在50H, 十位, 个位放在51H中

```
MOV B, #100
DIV AB
MOV 50H, A
MOV A, B
MOV B, #10
DIV AB
SWAP A
ADD A, B
MOV 51H, A
RET
```

幻灯片 140

3、逻辑操作类

1) 累加器清0

```
CLR A
```

若对寄存器进行操作, CLR指令只能对寄存器A进行操作, 且不影响CY, AC, OV等标志

2) 累加器内容按位取反

```
CPL A
```

若对寄存器进行操作, CPL指令只能对寄存器A进行操作, 且不影响CY, AC, OV等标志

幻灯片 141

3)、左循环移位

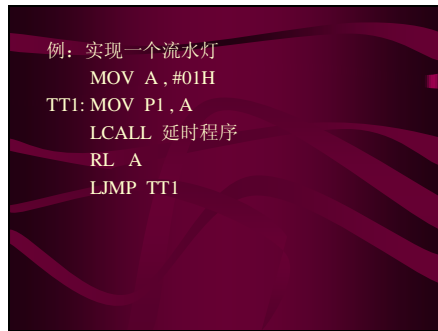
a、A内容循环左移(*) RL A

不影响标志

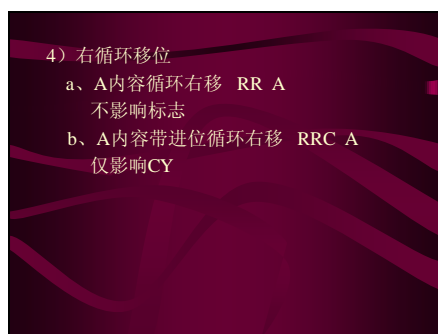
b、A内容带进位循环左移(*) RLC A

只影响CY

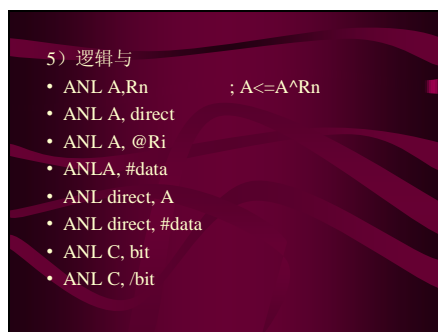
幻灯片 142



幻灯片 143



幻灯片 144



幻灯片 145

该指令若以A作为目标寄存器, 则影响奇偶标志位P, 若直接地址是P0~P3时, 可进行读-修改-写的操作(数据为锁存器内容), 可利用其对单元内容清零

例: 对内部20H单元的低4位清零, 高4位保持不变, 设(20H)=87H

```
ANL 20H, #0F0H
```

则(20H)=80H

幻灯片 146

6) 逻辑或 ORL

该指令若以A作为目标寄存器, 则影响奇偶标志位P, 若直接地址direct是P0~P3时, 可进行读-修改-写的操作(数据为锁存器内容), 可利用其对单元内容指定位置1

例: 已知TMOD各位均为0, 欲将其D7, D5, D2, D0位置1

```
ORL TMOD, #0A5H
```

幻灯片 147

例: 根据A中位4~0的状态, 用“与”, “或”控制P1口的位4~0状态, P1口的高三位不变

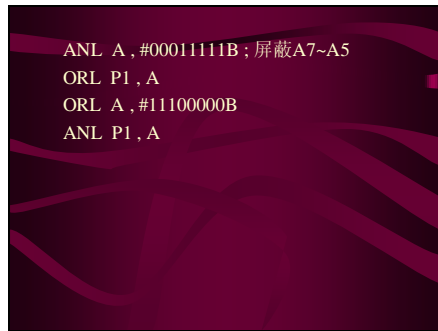
```
ANL A, #00011111B;屏蔽A7~A5
```

```
ANL P1, #11100000B;清P1口的低5位
```

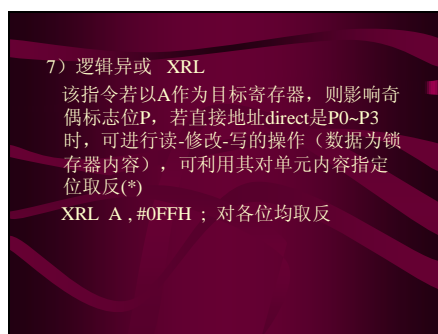
```
ORL P1, A;按 A4~A0设置P1.4~P1.0
```

如此, 位口线先输出0状态, 再按A4~A0置位, 可能发生一个机器周期短暂的“闪烁”, 所以改用下面的程序(*)

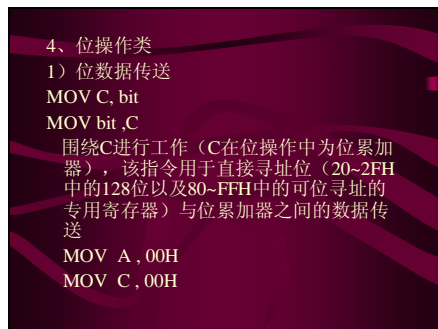
幻灯片 148



幻灯片 149



幻灯片 150



幻灯片 151

若直接寻址位为P0~P3口中的某一位, 指令执行时, 先读入端口的全部内容(8位), 而后通过C修改过后, 再把8位内容传送到端口锁存器, 因此, 它也是一种读-修改-写指令。

幻灯片 152

例: 把P1.0的状态传送到P1.6
MOV C, P1.0
MOV P1.6, C
读入P1口的8位数, 修改P1.6位, 而后把8位数再写到P1口的锁存器
例: 设片内数据存储器中(20H)=7FH,
则 MOV C, 07H执行后, (C)=0

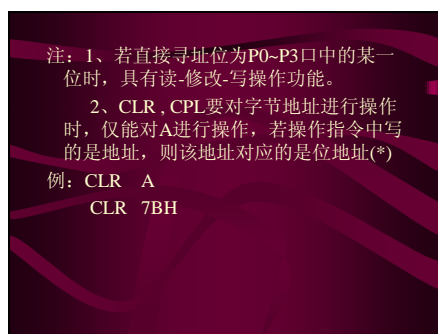
幻灯片 153

• 位地址的表示
00H, 20H.1, RS1, PSW.4

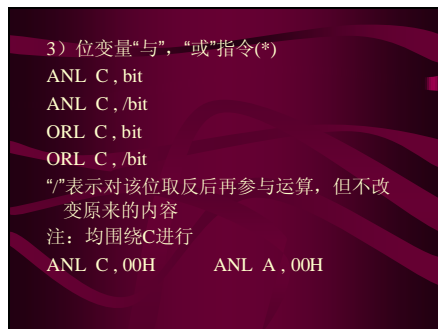
幻灯片 154



幻灯片 155



幻灯片 156



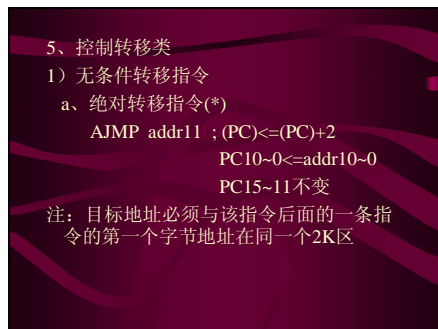
幻灯片 157



幻灯片 158



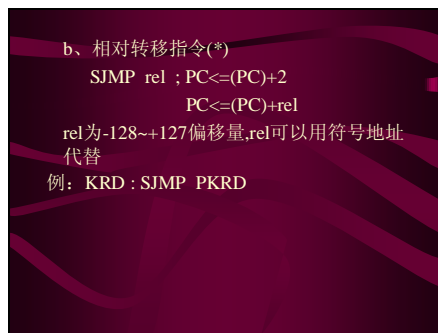
幻灯片 159



幻灯片 160



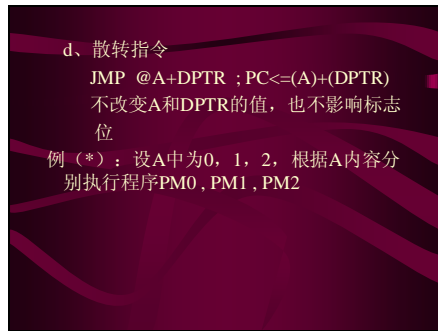
幻灯片 161



幻灯片 162



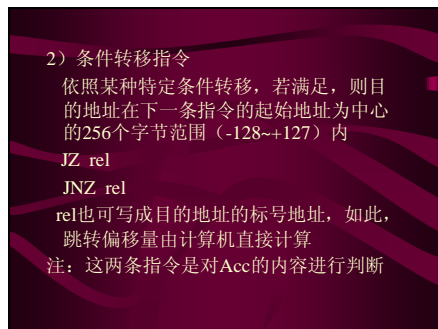
幻灯片 163



幻灯片 164



幻灯片 165



幻灯片 166

```
3) 比较不相等转移指令
CJNE A, direct, rel
CJNE A, #data, rel
CJNE Rn, #data, rel
CJNE Ri, #data, rel
比较两数大小, 若不相等则转移。
PC=(PC)+3 PC=(PC)+rel
若数1<数2, 则CY=1, 否则CY=0
该指令不影响任何一个操作数的内容
```

幻灯片 167

```
例: T2: CJNE A, #FFH, T1
操作: A=#FFH PC<=(PC)+3 CY=0 顺序
      执行
      A<#FFH CY=1 PC<=(PC)+3
           PC<=(PC)+rel
      A>#FFH CY=0 PC<=(PC)+3
           PC<=(PC)+rel
```

幻灯片 168

```
例: T2:CJNE A, #FFH, T1
      LJMP T3
      T1:JC T4
      LJMP T5
例: 比较内部RAM中30H和40H中的两个
无符号数的大小, 将大数存入50H, 小数
存入51H单元中, 若两数相等, 则使片内
RAM的7F位置1
```

幻灯片 169

```
MOV A, 30H
CJNE A, 40H, Q1
SETB 7FH
RET
Q1: JC Q2
MOV 50H, A
MOV 51H, 40H
RET
Q2: MOV 50H, 40H
MOV 51H, A
RET
```

幻灯片 170

4) 减1不为0转移指令

```
DJNZ Rn, rel
DJNZ direct, rel
```

源操作数减1再送给源操作数, 若不为0, 则跳转

例: 从P1.0输出15个方波 (*)

```
MOV R2, #30
PULSE: CPL P1.0
DJNZ R2, PULSE
```

该方波的周期为6个机器周期

幻灯片 171

5) 调用及返回指令

a. 绝对调用 (*)

```
ACALL addr11 ; PC<=(PC)+2
              SP<=(SP)+1
              (SP)<=(PC)L
              SP<=(SP)+1
              (SP)<=(PC)H
              PC<=addr10-0
```

注: 子程序地址必须与ACALL后一条指令的第一个字节在同一个2K区域的存储器区内。不影响标志位

幻灯片 172

b、长调用指令 (*)

```
LCALL addr16 ; PC<=(PC)+3
              SP<=(SP)+1
              (SP)<=(PC)L
              SP<=(SP)+1
              (SP)<=(PC)H
              PC<=addr16
```

注: 可调用存放在存储器中64K字节范围内任何地方的子程序

幻灯片 173

e、子程序返回指令

```
RET
```

弹出栈顶相邻两单元内容送入PC, SP内容减2

d、中断返回指令 (*)

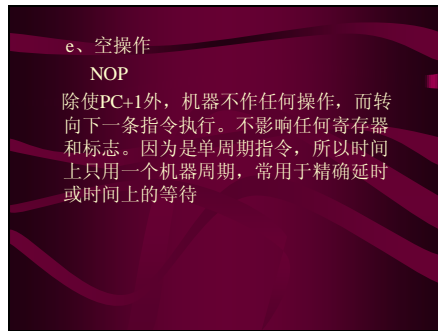
```
RETI
```

弹出栈顶相邻两单元内容送入PC, SP内容减2, 释放中断逻辑使之能接受同级或低级的另一个中断请求, 若在执行RETI指令时, 有一个同级或低级的另一个中断请求已经被挂起, 则CPU要在至少执行了中断返回指令之后的下一条指令才能去响应被挂起的中断

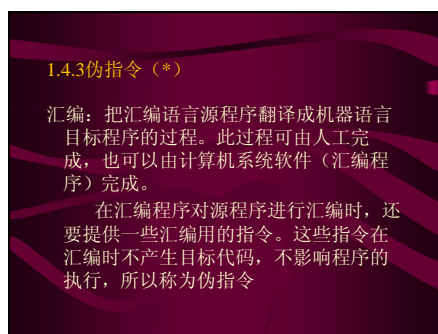
幻灯片 174

```
MOV R1, #10H
MOV R0, #30H
MOV A, @R0
      ——进入中断, 且有中断挂起
XCH A, @R1
此特性可以用来实现单步操作
```

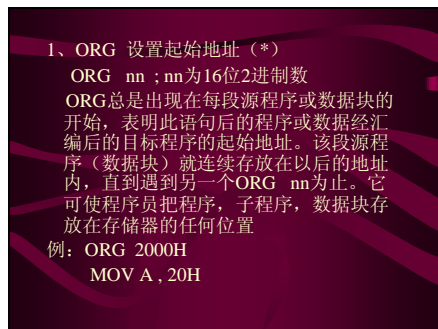
幻灯片 175



幻灯片 176



幻灯片 177



幻灯片 178

注: 一般要求ORG定义空间地址由小到大, 且不能重叠(*)

```
ORG 3000H      ORG 2000H
MOV A, 20H     MOV A, 20H
ORG 2700H      ORG 2001H
MOV A, 21H     MOV A, 21H
END            END
```

注: 若在源程序开始不放ORG指令, 则汇编将从0单元开始编排目标程序

幻灯片 179

2、DB 定义字节 (*)

标号: DB 字节常数或字符或表达式
把字节常数或字节串存入内存连续单元中

例: ORG 1000H
 SEG1: DB 53H, 74H, 78H, '1', '2'
 SEG2: DB 23H, 'DAY'
 END

注: DB后的若为数值, 则其取值范围为00-FFH, 即不能超过一个字节。若为字符串, 其长度应限制在80个字符内 (由汇编程序决定)

幻灯片 180

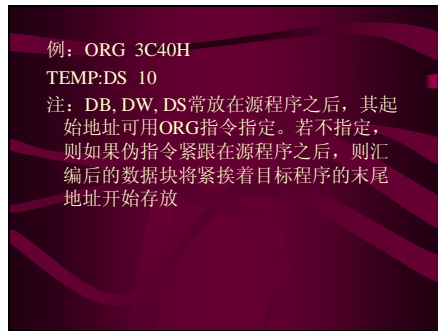
3、DW 定义一个字

标号: DW 字或字符串
汇编后, 低位地址存低位字节, 高位地址存高位字节

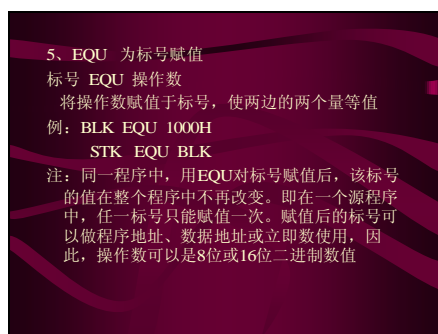
4、DS 预留存储区 (*)

标号: DS 项
由标号指定单元开始, 定义一个存储区, 以备源程序使用, 存储区内预留的单元数由“项”的值决定

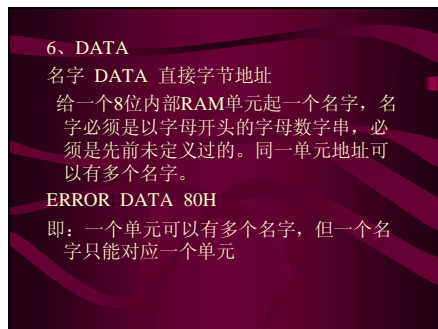
幻灯片 181



幻灯片 182



幻灯片 183



幻灯片 184

7、XDATA 直接字节地址
名字 XDATA 直接字节地址
给一个8位外部RAM单元起一个名字，名字必须是以字母开头的字母数字串，必须是先前未定义过的。同一单元地址可以有多个名字。
IO_PORT XDATA 0CF04H
即：一个单元可以有多个名字，但一个名字只能对应一个单元

幻灯片 185

8、BIT 位地址
名字 BIT 位地址
给一个可位寻址的位单元起一个名字，名字必须是以字母开头的字母数字串，必须是先前未定义过的。同一单元地址可以有多个名字。
SW1 BIT 30H
即：一个单元可以有多个名字，但一个名字只能对应一个单元

幻灯片 186

9、END 源程序结束
标号: END 地址
其中，标号和地址可有可无，该指令用来指示汇编语言源程序段已经结束
注：1、一个源程序中只允许出现一个END语句
2、END必须放在整个程序（包括伪指令）的最后，是源程序模块的最后一个语句。若END语句出现在中间，则汇编程序将不汇编END后面的语句

幻灯片 187

```
例:      ORG 8400H
          MOV A, R2
          MOV DPTR, #TBJ3
          MOVC A, @A+DPTR
          JMP @A+DPTR
TBJ3:    DW PRG0
          DB PRG1
          DB PRG2
PRG0     EQU 8450H
PRG1     EQU 80H
PRG2     EQU B0H
          END
```

幻灯片 188

例: 晶振频率12MHz, 试编一个3毫秒延时子程序

幻灯片 189

```
解:
DELAY:  MOV R7, #6      ;1us
        D1: MOV R6, #248 ;1us
           DJNZ R6, $    ;2us*248=496us
           DJNZ R7, D1   ;(2us+496us+1us)*6=2.994ms
           RET           ;子程序总延时= 2us+2994us
```

幻灯片 190

例: 编制一个循环闪烁的程序。有八个发光二极管, 每次其中某个灯闪烁点亮10次后, 转移到下一个灯闪烁10次, 循环不止。本程序的硬件连接如图所示。当P1.0输出高电平时, LED灯亮, 否则不亮

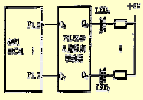


图 10 1 LED 闪烁电路

幻灯片 191

例: 编制一个循环闪烁的程序。有八个发光二极管, 每次其中某个灯闪烁点亮10次后, 转移到下一个灯闪烁10次, 循环不止。本程序的硬件连接如图所示。当P1.0输出高电平时, LED灯亮, 否则不亮

其程序如下:

```

MOV A, #01H ; 灯亮初值
SHIFT: LCALL FLASH ; 调闪亮10次子程序
RR A ; 右移一位
SIMP SHIFT ; 循环
FLASH: MOV R2, #0AH ; 闪烁10次计数
FLASH1: MOV P1, A ; 点亮
LCALL DELAY ; 延时
MOV P1, #00H ; 熄灭
LCALL DELAY ; 延时
DJNZ R2, FLASH1; 循环10次
RET
    
```

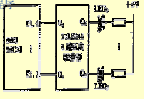


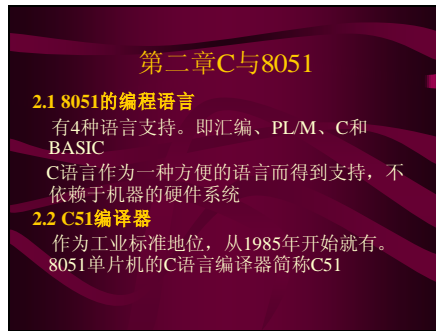
图 11 1 LED 闪烁电路

幻灯片 192

- 书P32例
- 作业

P34 7, 8
P36 34, 36

幻灯片 193

A slide with a dark purple background and a wavy pattern. The text is in yellow and white.

第二章C与8051

2.1 8051的编程语言
有4种语言支持。即汇编、PL/M、C和BASIC
C语言作为一种方便的语言而得到支持，不依赖于机器的硬件系统

2.2 C51编译器
作为工业标准地位，从1985年开始就有。
8051单片机的C语言编译器简称C51

幻灯片 194

A slide with a dark purple background and a wavy pattern. The text is in white.

KEIL和IAR领先

KEIL以它的紧凑代码和使用方便领先;
以它性能完善和资料完善领先;
FRANKLIN V4.0 ARCHIMEDES V4.0

幻灯片 195

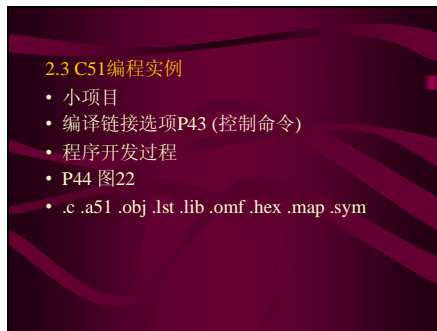
A slide with a dark purple background and a wavy pattern. The text is in white.

- 单片机的C语言应用程序设计（修订版）
北京航空航天大学出版社
- Intel Microcontroller Data Sheet.1984
Schltz,Thomas W. C and
8051:Programming
for multitasking .Prentice Hall
- 嵌入式C编程技术
- 单片机与嵌入式系统应用2001(1~6)

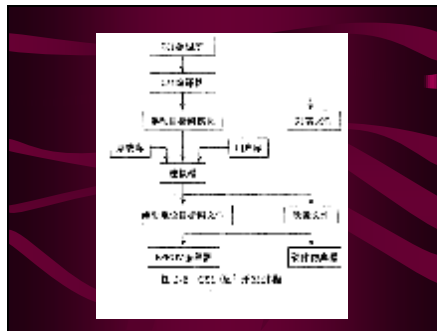
幻灯片 196



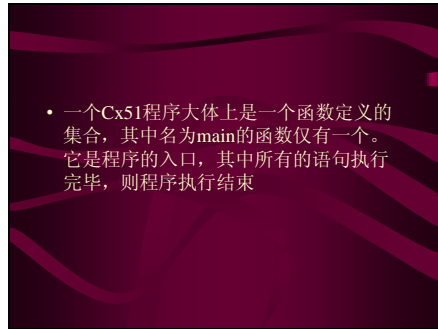
幻灯片 197



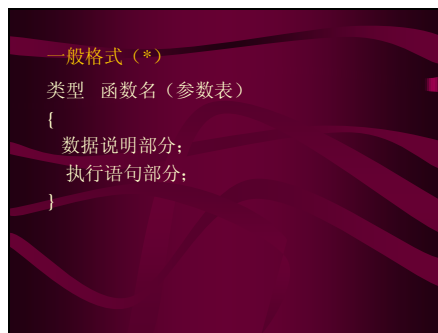
幻灯片 198



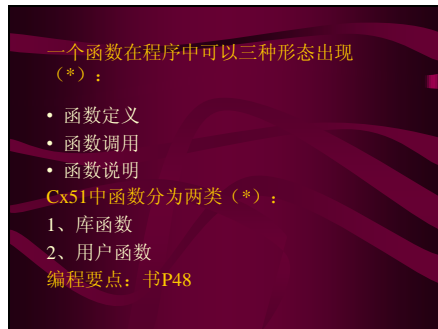
幻灯片 199



幻灯片 200



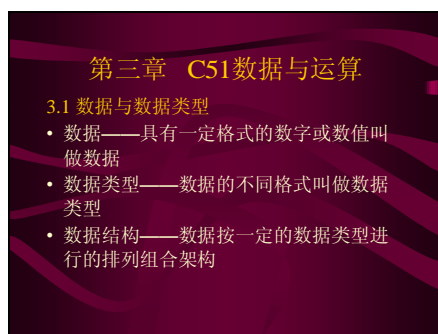
幻灯片 201



幻灯片 202



幻灯片 203



幻灯片 204



幻灯片 208

3.3 C51数据的存贮类型与8051存贮器结构

因为keil c面向单片机及其硬件控制系统, 所以它定义的任何数据类型必须以一定的存储类型的方式定位在8051的某一存储区中

8051系列机在物理上有四个存储空间:

- 1、片内程序存储器空间
- 2、片外程序存储器空间
- 3、片内数据存储器空间
- 4、片外数据存储器空间

幻灯片 209

8051片内数据存储器可划分为两类:

- 00H-7FH为片内低128字节RAM区
- 80H-0FFH为特殊功能寄存器区

其中, 低字节RAM区又可以划分为3个区域:

- 1、通用寄存器区00H-1FH

每个寄存器可以用寄存器名寻址也可直接用字节地址寻址

- 2、可位寻址区

可以按字节寻址操作也可按位地址操作

- 3、用户RAM区

幻灯片 210

- 程序存储器与数据存储器严格分开, 特殊功能寄存器与片内数据存储器统一编址
- 片内数据存储器是存放临时性传递变量或使用频率较高的变量的。访问片内数据存储器速度较快, 经常使用的变量置于片内数据存储器, 而将不常用的置于片外数据存储器中。寻址方式可使用直接和间接寻址

幻灯片 211

存储类型说明:

Keil C51通过将常量、变量定义成不同的存储类型的方法,将它们定位在不同的存储区中。

- data 直接寻址内部数据存储区,访问速度快(128B),00-7FH
- bdata 可位寻址内部数据存储区,允许位与字节混合访问(16B)
- idata 间接寻址内部数据存储区,可访问片内全部RAM地址空间(256B),00-FFH
- pdata 分页寻址外部数据存储区(256B),由MOVX @Ri访问(高位P2)
- xdata 外部数据存储区(64KB),由MOVX @DPTR访问
- code 代码存储区(64KB)程序由MOVC @A+DPTR访问

幻灯片 212

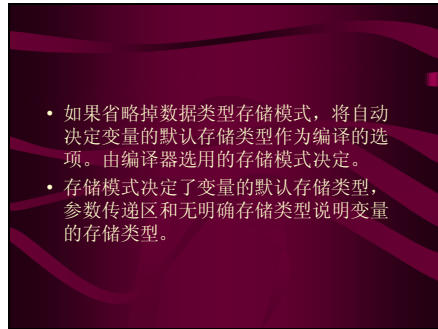
- 当使用code存储类型定义数据时,C51编译器会将其定义在代码空间或FLASH
- 访问片内数据存储区data、bdata、idata比访问片外数存相对要快一些,因此,可将经常使用的变量置于片内数存

幻灯片 213

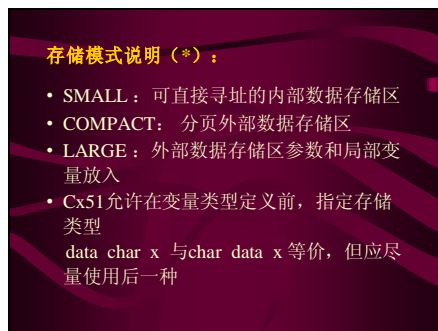
例:

```
bit flag ;布尔值
code uchar table[ ]={1,2,3,"help",0xff};
idata uint temp;
data char var; /char datavar;等价尽量用后一种
static unsigned long xdata array[100]; 静态变量
extern float idata x,y,z;模块化编程
uint pdata dimension;
uchar xdata vector [10][4][4];
char bdata flags;
bit flag_0=flags^0;
sbitP1_1=P1^1;
```

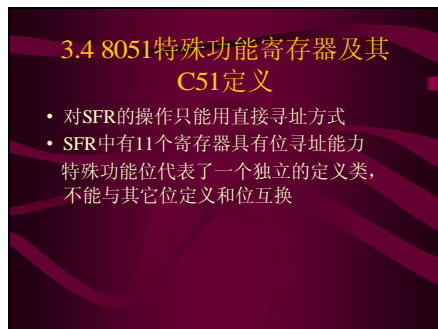
幻灯片 214



幻灯片 215



幻灯片 216



幻灯片 217

图 3-3 8051 特殊功能寄存器一览表

寄存器	地址	功能
+ ACC	00H	累加器
+ B	01H	寄存器 B
+ PSW	02H	程序状态字
+ SP	03H	堆栈指针
+ DPTR	04H	数据指针(高 8 位)
+ DPTR	05H	数据指针(低 8 位)
+ IE	0AH	中断允许寄存器
+ IP	08H	中断优先级寄存器
+ P0	80H	端口 0
+ P1	90H	端口 1
+ P2	A0H	端口 2
+ P3	B0H	端口 3
+ PCON	87H	电源控制寄存器
+ SCON	98H	串行口控制寄存器
+ SBUF	99H	串行口缓冲寄存器
+ TCON	8BH	定时器控制寄存器
+ TMOD	8CH	定时器模式寄存器
+ TL0	82H	定时器 0 低 8 位
+ TH0	83H	定时器 0 高 8 位
+ TL1	86H	定时器 1 低 8 位
+ TH1	87H	定时器 1 高 8 位

幻灯片 218

说明:

- 1、bit: 8051 单片机有位操作指令完整布尔处理器
- 2、char: 即字节; int: 即字;分为有符号和无符号数。无论何时应尽可能地使用无符号字符变量。因为它能被 8051 所接受。如果使用有符号和无符号两种数据类型, 那么就得使用两种格式类型的库函数, 占用的存储空间成倍增长。需要进行额外的操作来测试代码的符号位, 这无疑会降低代码效率

幻灯片 219

- 3、变量可以组合为结构和联合, 也可定义多维数组, 可通过指针访问变量
- 4、sbit, sfr 简化对 8051 的 SFR 的访问 (*)

幻灯片 220

- 复习SFR (*)
- sfr SFR的预定义标示符=绝对地址
- sfr16 SFR的预定义标示符=绝对地址

注: 1、可以与变量标示符一样, 用预定义标示符(即SFR名)去存取SFR。
2、定义时标示符必须使用SFR名, 且必须把原来分配好的绝对地址赋给预定义标示符

例: sfr Acc=0xE0;
sfr P0=0x80;

幻灯片 221

由于8051系列中不同的单片机的寄存器数量与类型不同, 所以可采用头文件。
头文件reg51.h中有所有8051的SFR及可位寻址的位的定义, 只要

```
#include <reg51.h>
```

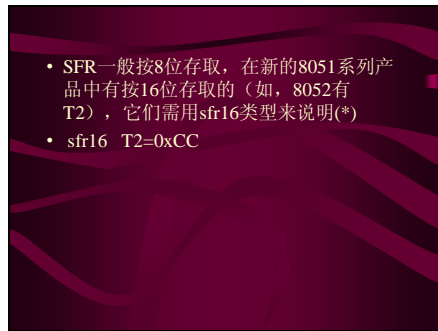
幻灯片 222

程序的开头都加上以下三行

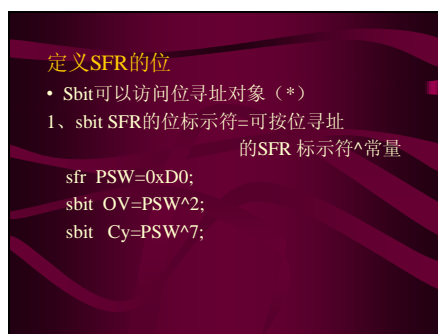
```
#include<reg51.h>
#define uchar unsigned char
#define uint unsigned int
```

- 头文件reg51.h中有所有8051的SFR及可位寻址位的定义

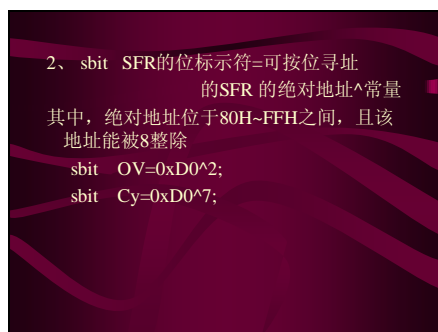
幻灯片 223



幻灯片 224



幻灯片 225



幻灯片 226

3、sbit SFR的位标示符=可按位寻址的SFR的绝对位地址
该绝对位地址位于80H~FFH之间
sbit OV=0xD2;
sbit Cy=0xD7;
特殊功能位代表了一个独立的定义类,不能与其他位定义和位域互换

幻灯片 227

3.5 8051并行接口及其Cx51定义

一、复习并行口

二、(*)编程时,8051片内I/O口与片外扩展I/O口可统一在头文件中定义,也可在程序的开始位置定义,方法如下:

1、片内I/O口(用sfr定义)

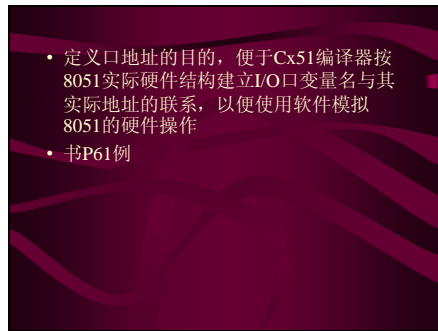
```
sfr P0=0x80  
sfr P1=0x90
```

幻灯片 228

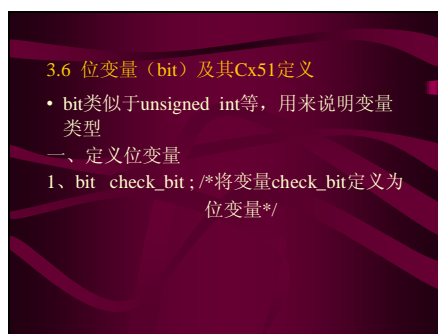
2、片外扩展I/O口,根据其硬件译码地址,将其视为片外数据存储器中的一个单元,用define定义(*)。

```
#include<absacc.h>  
#define PORTA XBYTE[0xffc0]  
{  
i=PORTA;  
PORTA=i;  
}
```

幻灯片 229



幻灯片 230



幻灯片 231



幻灯片 232

```
2、函数可包含类型为bit的参数，也可将其
   作为返回值
   bit func(bit b0, bit b1)
   {
       .....
       return(b1)
   }
```

幻灯片 233

```
3、可位寻址对象
   指可以字节或位寻址的对象。该对象应
   位于8051片内可位寻址RAM中。允许数
   据类型为idata的对象放入8051片内可
   寻址RAM区中。定义可分为两步：
   1) 先定义变量的数据类型和存储类型(*)
       bdata int ibase;
       bdata char bary[4];
```

幻灯片 234

```
2)然后可使用sbit定义可独立寻址访问的对象位(*)
   sbit mybit0=ibase^0;
   sbit mybit15=ibase^15;
   sbit Ary07=bary[0]^7;
   sbit Ary37=bary[3]^7;
   对象ibase和bary也可以字节寻址
   Ary37=0; /*位寻址*/
   bary[3]='a'; /*字节寻址*/
   注：1、Sbit定义要求基址对象的存储类型为
       bdata，否则只有绝对的特殊位（SFR中的位定
       义）定义是合法的。
       2、位置（^操作符）后的最大值依赖于指
       定的基类型（*）
```

幻灯片 235

例: 判浮点数的符号位是否为正

```

union float long
{
    float bdata f;
    long bdata l;
}fl;
sbit float_sign=fl.P31;
if (!float_sign) /*符号位为正*/
{.....}
else
{.....}

```

	7	0
高地址	S	E
	E	M
低地址		M
		M

幻灯片 236

3.7 Cx51运算符, 表达式及其规则

3.7.1 Cx51算术运算符及其表达式

- 1、Cx51最基本的五种算术运算符 (*)
+, -, *, /, %
- 2、优先级
先乘除, 后加减, 括号最优先。从左至右
- 3、类型转换

幻灯片 237

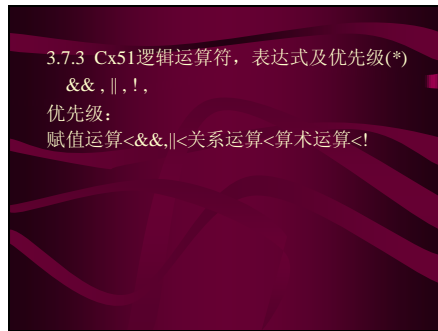
3.7.2 Cx51关系运算符, 表达式及优先级

<, >, <=, >= 优先级相同 (高)
==, != 优先级相同 (低)

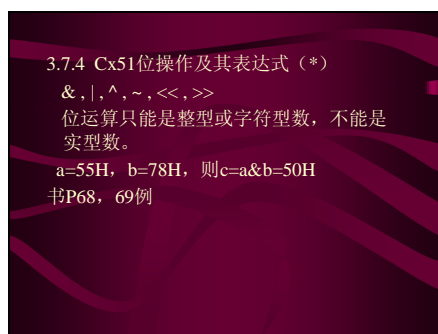
赋值运算符 < 关系运算符 < 算术运算符

注: 1、关系运算中关系成立, 结果为“1”, 不成立, 结果为“0”。(*)
2、两个指针也可参与比较 (*)

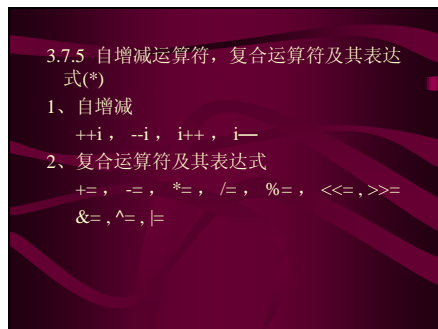
幻灯片 238



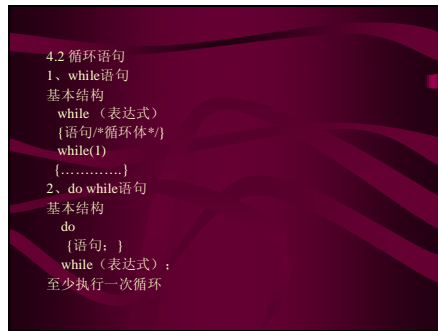
幻灯片 239



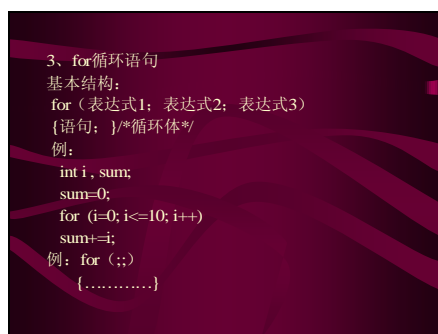
幻灯片 240



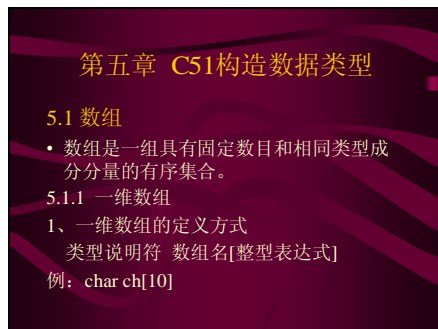
幻灯片 244



幻灯片 245



幻灯片 246



幻灯片 247

2、数组的初始化

若在定义说明数组的同时, 给数组赋初值

1) 在定义数组时对数组的全部元素赋初值

```
int idata a[6]={0, 1, 2, 3, 4, 5}
```

2) 只对数组的部分元素初始化

```
int idata a[10]={0, 1, 2, 3, 4, 5}
```

3) 定义时若不对数组全部元素赋初值, 则全部元素缺省赋值为0

```
int idata a[10] /*每个值都为0*/
```

书P88, 89例

幻灯片 248

5.1.2 二维数组 (*)

1、定义

类型说明符 数组名[常量表达式][常量表达式]

```
int a[3][5]
```

2、二维数组初始化

1) 对数组的全部元素赋初值

```
int a[2][2]={{1, 2}, {3, 4}}
```

```
int a[2][2]={1, 2, 3, 4}
```

2) 对数组中部分元素赋初值

```
int a[2][2]={{1}, {3}}
```

解释P92例

幻灯片 249

5.2 指针

- 指针是C语言的一个重要概念, 也是重要特色之一。

幻灯片 250

- 5.2.1 指针的基本概念

变量的指针就是变量的地址。
指向变量的指针变量: 若有一个变量专门来存放另一个变量的地址, 则该变量称为指向变量的指针变量。

1、指针变量的定义

类型标示符 *指针变量名

例: `int *ap; /*定义ap为指针变量, 指向整型变量*/`

幻灯片 251

2、指针变量的引用 (*)

```
int a;
int *ap;
ap=&a; /*ap指向变量a,&为取地址符*/
指针运算符"*", *ap和a等价
*ap表示ap所指向的变量
x=*ap; /**ap所指向的变量赋值给x*/
*ap=0xff; /*a=0xff*/
```

幻灯片 252

- 5.2.2 数组指针和指向数组的指针变量

数组的指针就是数组的起始地址。指向数组的指针变量用来存放一个数组的起始地址。

幻灯片 253

5.2.3关于KEIL C51的指针类型

支持“基于存储器的指针”和“一般”指针两种类型。

- 基于存储器的指针
在编译时一般被“行内”编码，无须库调用。即指针所指的对象所在的存储空间由定义时c源代码中的存储器类型决定。该指针只需1~2字节
- 一般指针
一般指针包括3个字节。2字节偏移（即指针的地址）和1字节存储器类型。为了表示这种指针必须用长整数来定义存储类型。

幻灯片 254

指针定义 (*)

static	data	unsigned char	data	*标示符
auto	idata	char	idata	
register	pdata	unsigned int	pdata	
extern	xdata	int	xdata	
	code	unsigned long	code	
	bdata	long	bdata	
		float		
		struct		
		union		

幻灯片 255

- 第二列指出指针变量存放在何处，缺省时决定于编译用存储模式
- 第四列指明指针所指的对象存放在何处，缺省时为三字节的通用指针。

幻灯片 256

1、基于存储器的指针
由C源代码中的存储类型决定。用这种指针可
高效访问对象且只需1至2字节

- 1个字节idata*, data *.pdata*
- 2个字节code *.xdata*
- 例: char xdata *px

其中, xdata为指针指向的对象变量定位存
储空间, 指针长度为2个字节, 所指的对象是
个字符型, 指针自身在默认存储器区(决定于
编译模式)

幻灯片 257

- char idata *px; 所指的对象变量在idata
中, 指针为1个字节
- data char xdata *px; 除指明指针所指对象
在xdata中, 还指明指
针自身存放在data中
- char xdata *data px; 与上式等价
与早期C51版本兼容

幻灯片 258

- 例 (*)
struct time
{ char hour;
char min;
char sec;
struct time xdata *pxtime}
struct time idata *px;
px->pxtime->hour=12;

幻灯片 259

2、一般指针
即定义时未指明所指对象的存储空间
共3字节: 1个存放存储器类型, 2个存放地址偏移量

低	存储器类型	指针所指变量的地址空间
	偏移量高位	
高	偏移量低位	

幻灯片 260

• 存储器类型编码 (*)

存储器类型	idata/data/bdata	xdata	pdata	code
值	0x00	0x01	0xFE	0xFF

幻灯片 261

5.3 结构

- 把多个不同类型的变量结合在一起形成的一个组合型变量, 称为结构变量, 简称结构

5.3.1 结构的定义

1、定义结构类型

```
struct 结构名
{
    结构成员说明 /*类型标示符 成员名*/
};
```

幻灯片 262

```
struct person
{
    char name;
    int age;
};
```

struct person为程序员自己定义的结构类型, 它和系统定义的标准类型一样可以用来定义变量的类型

幻灯片 263

2、定义结构类型变量

1) 先定义结构的类型, 再定义类型为该结构的变量

```
struct person
{
    char name;
    int age;
};
struct person data1, data2;
```

幻灯片 264

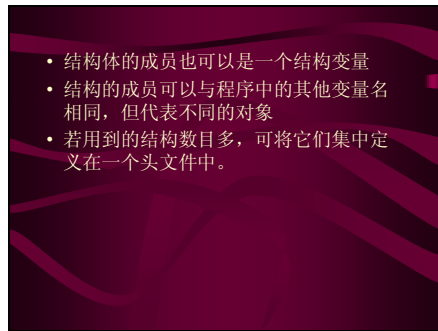
2) 定义结构类型的同时, 定义该结构的变量

```
struct person
{
    char name;
    int age;
}data1, data2;
```

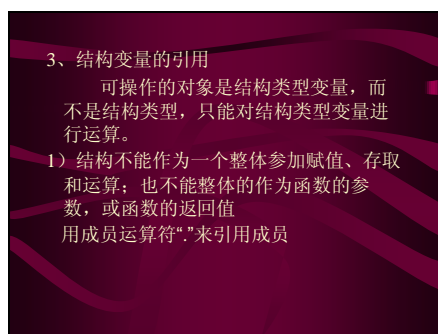
3) 直接定义结构类型变量

```
struct
{
    char name;
    int age;
}data1, data2;
```

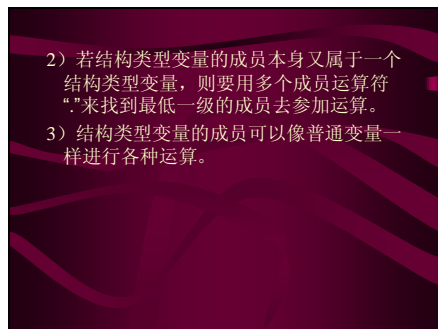
幻灯片 265



幻灯片 266



幻灯片 267



幻灯片 268

```
struct person
{
    char name;
    int age;
}
struct fac
{
    int machinenum;
    struct person worker;
}data;
引用时: data.worker.age
```

幻灯片 269

5.3.2 结构数组

定义: 数组中的每个元素都具有相同结构类型的结构变量。

```
struct person
{
    char name;
    int age;
};
struct person worker[10];
```

幻灯片 270

```
struct person
{
    char name;
    int age;
}worker[10];
或:
struct
{
    char name;
    int age;
}worker[10];
书P107 例
```

幻灯片 271

- 接口芯片用到一些固定外部数据存储器地址

```
#include <absacc.h>
#define PORT XBYTE[0xffc0];
#define COM XBYTE[0xdfff];
#define DAT XBYTE[0xdffe];
#define XBYTE((char *)0x20000L)
```

幻灯片 272

5.4 共用体 (联合)

- 定义格式
union 共用体类型标示符
{
 类型说明符 变量名;
};
包含多个不同数据类型的元素。
与结构的区别: 其包含的各个成员只能分时共享
同一存储空间。

幻灯片 273

```
• 定义变量
union press
{
  int i;
  char ch;
};
union press cnut;
引用: cnut.i cnut.ch
```

幻灯片 274

```
或:  
union press  
{  
    int i;  
    char ch;  
}cnut;  
或:  
union  
{  
    int i;  
    char ch;  
}cnut;
```

幻灯片 275

5.5 枚举

- 枚举数据类型是一个有名字的某些**整数型常量**的集合。这些整数型常量是该类型变量可取的**所有的合法值**。枚举定义应当列出该类型变量的可取值。
- 定义格式
enum 枚举名 {枚举值列表} 变量列表;
或:
enum 枚举名 {枚举值列表};
enum 枚举名 变量列表;

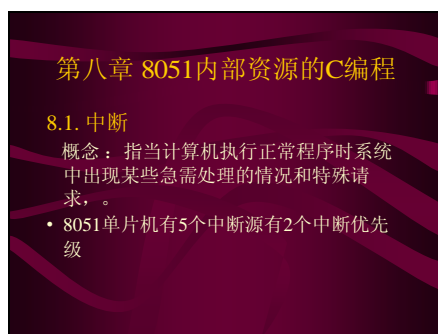
幻灯片 276

- enum day {1, 2, 3, 4, 5, 6, 7} d1, d2;
- 或
- enum day {1, 2, 3, 4, 5, 6, 7};
enum day d1, d2;

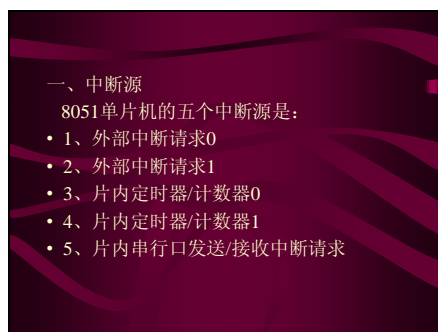
幻灯片 277



幻灯片 278



幻灯片 279



幻灯片 280

- 外中断方式电平触发IT=0
- 边沿触发IT=1
- 中断触发控制位IT0,IT1

Interrupt Type control

- CPU在每个机器周期的S5P2检测引脚,若是电平型,检测到低电位低电平>T;若是边沿型,前一次为,高后一次为低;定时器计数器是加1计数的溢出,是指由全1进入全0;串行口指发送或接收一帧串行数据

幻灯片 281

记录中断、设置中断请求触发器标志位、中断标志:

- IE0: Interrupt Edge flag 硬件置位
- IE1: 硬件清除
- TF0: Timer Overflow硬件置位
- TF1: 硬件清除
- TI: Transmit Interrupt 硬件置位
- RI: Receive Interrupt 必须软件清除,相应标志在SFR中

幻灯片 282

TCON(Timer/counter Control register) 88H

D7 D6 D5 D4 D3 D2 D1 D0

--	--	--	--	--	--	--	--

幻灯片 283

