第一章 概述

80C51系列命名规则：

- 80C51/52
- 89C51/52
- 87C51/52

89C51与89C52的区别？

1.片内通用RAM：128字节/256字节

2. 片内FLASH ROM容量：4k/8k bytes

3. 定时器：89C52有3个定时器，比89C51多T2

第二章 基本结构
1、IO口

4个8位口（P0，P1，P2，P3），
其中P0,P2可作为扩展数据/地址线

- IO口驱动：

灌电流能力强于拉电流能力

P0口作为通用IO口时，必须加上拉电阻

IO口在复位时的状态

Cortex-M的GPIO的**灌电流能力与拉电流能力相同**

第二章 基本结构

2、特殊引脚

RST：高电平有效

/EA：片外程序存储器访问允许

ALE：锁存

/PSEN：取指令

第二章 基本结构
3、特殊指针和寄存器

- PC
- SP
- DPTR
- ACC
- PSW

第二章 基本结构
4、机器周期和指令周期

- 一个机器周期 ＝ 12个振荡脉冲周期
- **指令周期：**执行一条指令所需要的时间称为指令周期。不同指令可包含有一、二、四个机器周期。
- **指令时序图**

第二章 基本结构
5、存储器结构和地址空间

- **存储器的两种基本结构：**

冯.诺伊曼结构，哈佛结构

3个存储器地址空间：

- 片内、片外统一的 64 KB程序存储器地址空间；
- 内部256B（80C52 为384B）数据存储器地址空间；
- 片外64 KB的数据存储器地址空间。

80C51和STM32F407各属于哪种结构？为什么？

第二章 基本结构

5、存储器结构和地址空间

怎样区分程序存贮器和数据存储器

- RAM和ROM
- 读取指令：

MOV　　A,ADDR

MOV　　A,@R0

MOVX

MOVC

第二章 基本结构
5、存储器结构和地址空间

4个工作寄存器组

- 由RS0，RS1确定

RS_1	RS_0	寄存器组	$R_0 \sim R_7$ 地址
0	0	组0	00 ~ 07H
0	1	组1	08 ~ 0FH
1	0	组2	10 ~ 17H
1	1	组3	18 ~ 1FH

第二章 基本结构
6、位寻址区

位寻址区与RAM地址对应关系

- 共256位地址
- 前128地址：20H~2FH
- 后128地址：对应RAM

  >=80H，且后半字节为0或8

第二章 基本结构
7、堆栈 SP

- PUSH
- POP
- 后进先出
- 复位后SP的内容是07H
- LCALL，RET，中断都会对堆栈操作

第三章 指令系统

- **CISC** 指令集，指令长度不一
- **RISC**
- 不需要记住具体的机器码，但要能够分析典型指令的指令长度和执行时间
- 指令由操作码和操作数组成
  推论（两个操作数的指令长度必定是3个字节，执行时间是2个机器周期）

第三章 指令系统

1、7种寻址方式

- 寄存器寻址
- 直接寻址
- 立即寻址
- 变址寻址
- 寄存器间接寻址
- 相对寻址
- 位寻址

第三章 指令系统

1、7种寻址方式

注意点：

- 80C52高128字节RAM寻址必须用寄存器寻址，对特殊功能寄存器（位于高128字节地址）必须用直接寻址。
- 寄存器间接寻址只能用R0和R1。
- CLR　　ADDR是位寻址指令。
- 其他不合法指令，如 MOV　R1,R0

  PUSH　 R0

第四章 程序设计
1、指令和伪指令

- 伪指令的含义

EQU

DB----字节

DW----字

Little endian：一个Word中的低位的Byte放在内存中这个Word区域的低地址处 。Cortex系统一般采用该方式。

Big endian：一个Word中的高位的Byte放在内存中这个Word区域的低地址处 。

80C51系统16位表格存放（DW）采用该方式。

第四章 程序设计
2、要掌握的编程方法

- 运算（多字节加减，移位，BCD码）
- 查表（MOVX）
- 散转（JMP @A+DPTR）

```
PUSH
PUSH
RET
```

第五章 中断

- 中断的概念
- 5个中断源
- 中断的优先级

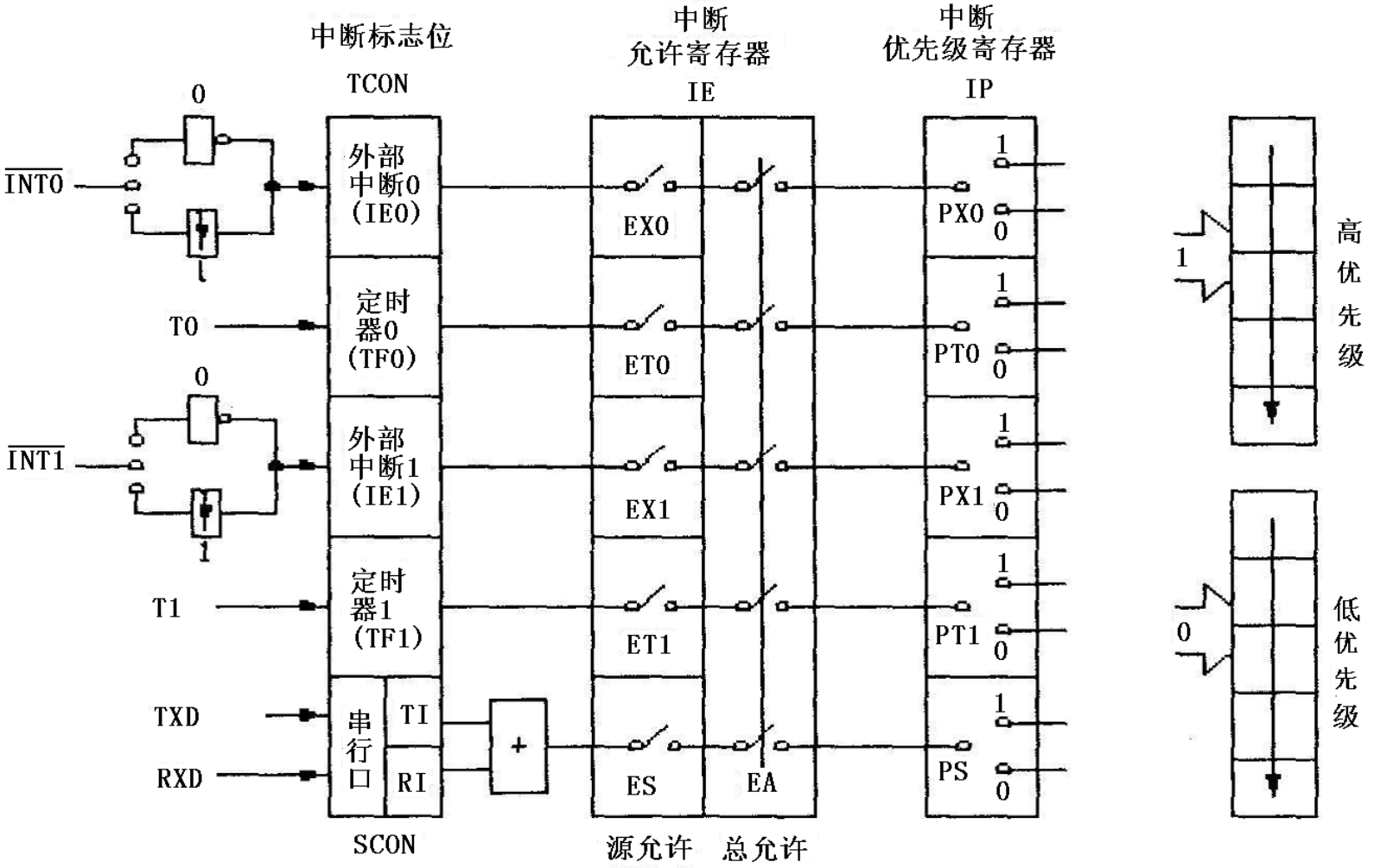
  中断允许寄存器IE

  中断优先级寄存器IP

第五章 中断

1、中断矢量

中断源	中断矢量地址
外部中断0（INT0）	0003H
定时器/计数器0（T0）	000BH
外部中断1（INT1）	0013H
定时器/计数器1（T1）	001BH
串行口（RI、TI）	0023H
定时器/计数器2	002BH



80C51的中断系统结构示意图

第五章 中断
2、中断标志

- 每个中断都有一个对应标志位
- 响应中断时，除串行中断RI，TI需要手动清除外，其余硬件*自动清除*

第五章 中断
3、中断响应时间

- 中断响应时间不确定

最短：3个机器周期（大部分情况下都是3~4个机器周期）

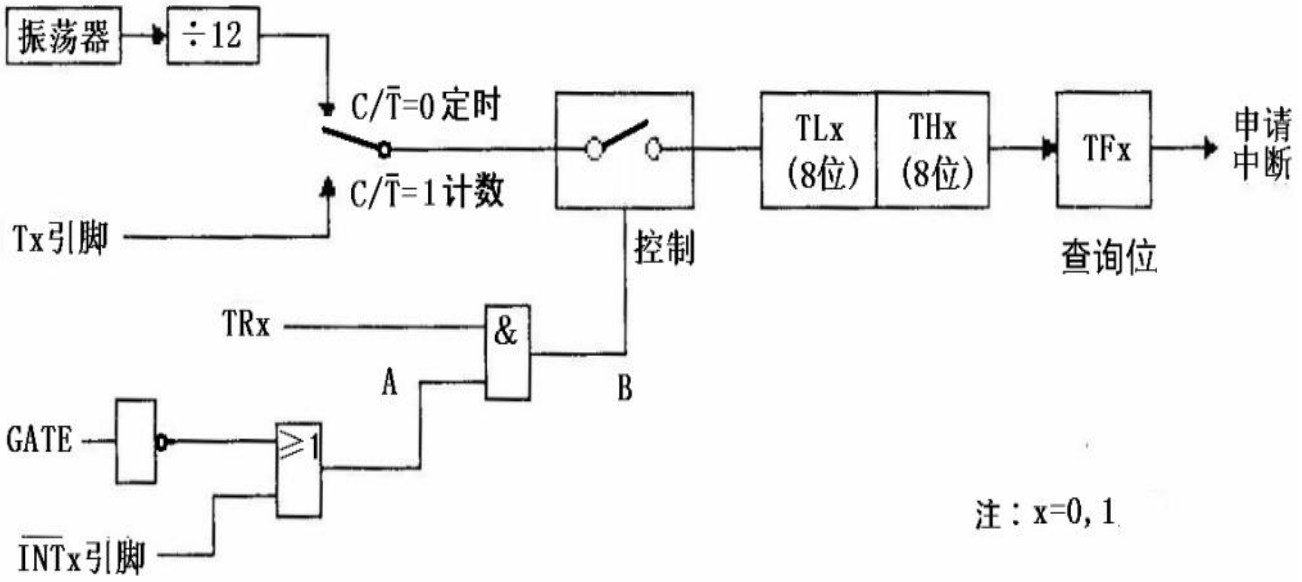
最长：8个机器周期

第五章 中断
4、中断现场保护

```
        ORG     0013H          ;;中断地址
        LJMP    INT_1

INT_1:
        PUSH    PSW
        PUSH    ACC
        SETB    RS0
        CLR     RS1
……
        POP     ACC
        POP     PSW
        RETI
```

第六章 定时/计数器

- 80C51：2个定时器
- 80C52：3个定时器
- **内部结构，定时器与计数器的区别**



第六章 定时器

与定时器 / 计数器T0、T1有关的特殊功能寄存器

- **TMOD**：工作方式控制寄存器

地址89H，不能位寻址

低4位用来定义T0，高4位用来定义T1

位序	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
位符号	GATE	$C/\overline{T}$	M_1	M_0	GATE	$C/\overline{T}$	M_1	M_0

定时/计数器1　　定时/计数器0

第六章 定时器

- TCON: 控制寄存器

控制寄存器TCON是一个逐位定义的8位寄存器，字节地址为88H，位寻址的地址为88H～8FH。其格式如下：

位地址	8F	8E	8D	8C	8B	8A	89	88
位符号	TF_1	TR_1	TF_0	TR_0	IE_1	IT_1	IE_0	IT_0

TR_0、TR_1——定时器运行控制位

0为停止工作，1为启动定时器/计数器

第七章 串行通信

- 串行通信的基本概念：

同步通信和异步通信

单工、双工、半双工

波特率

传输方式：*起始位，停止位，低位在前*

通信双方的波特率允许误差？

第七章 串行通信

口 80C51串行口控制

➢ SCON：串行（状态）控制寄存器

单元地址98H 位地址9FH ~ 98H

位地址	9F	9E	9D	9C	9B	9A	99	98
位符号	SM_0	SM_1	SM_2	REN	TB_8	RB_8	TI	RI

第八章 系统扩展
1、并行扩展

三组总线

地址总线（**AB**）

P2：高8位　　**P0**：低8位（与数据总线复用）

数据总线（**DB**）

P0（与地址总线复用）

控制总线（**CB**）

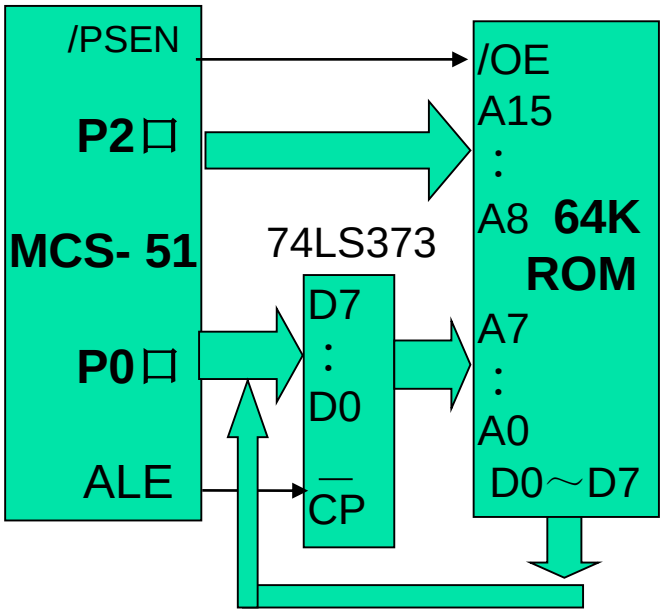
ALE

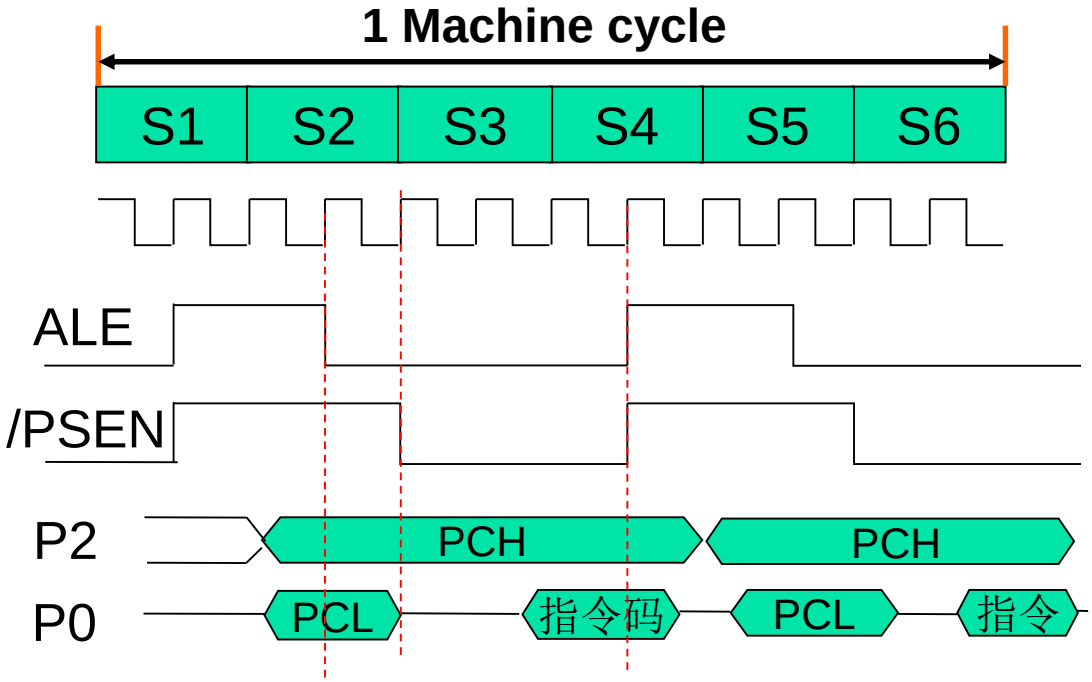
PSEN （**Program Store Enable**）

RD、**WR**

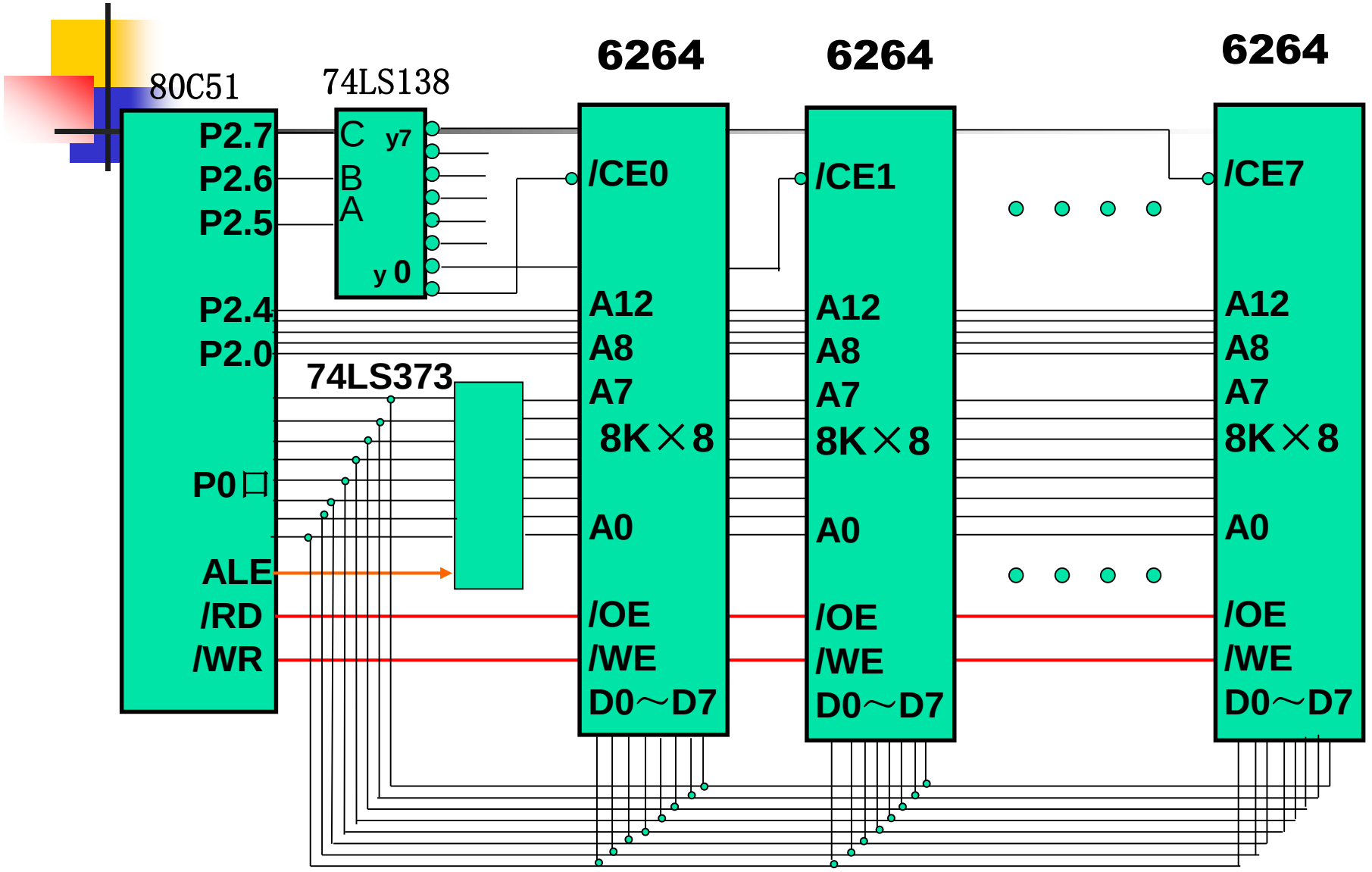
片外存储器取指操作时序

每个机器周期包括6状态（**S1-S6**），每状态包括2节拍





80C51外扩多片 SRAM连接图（地址线全译码）

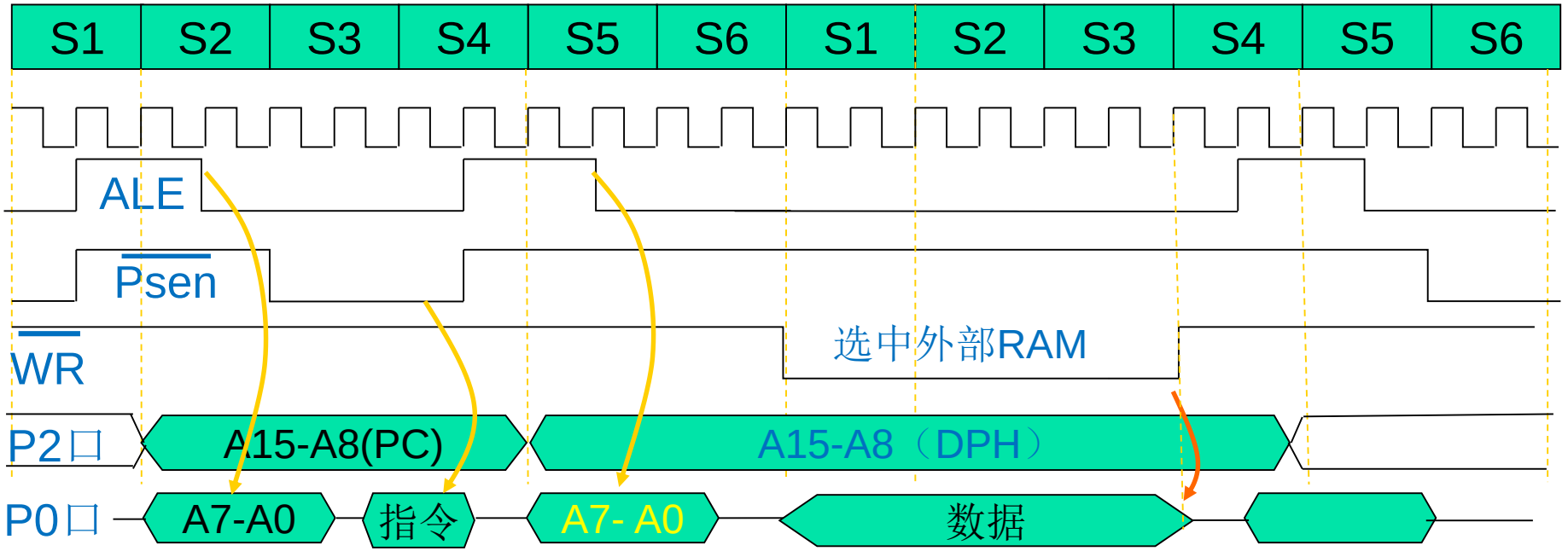


数据存储器扩展

P0输出的4种状态(数据复用):

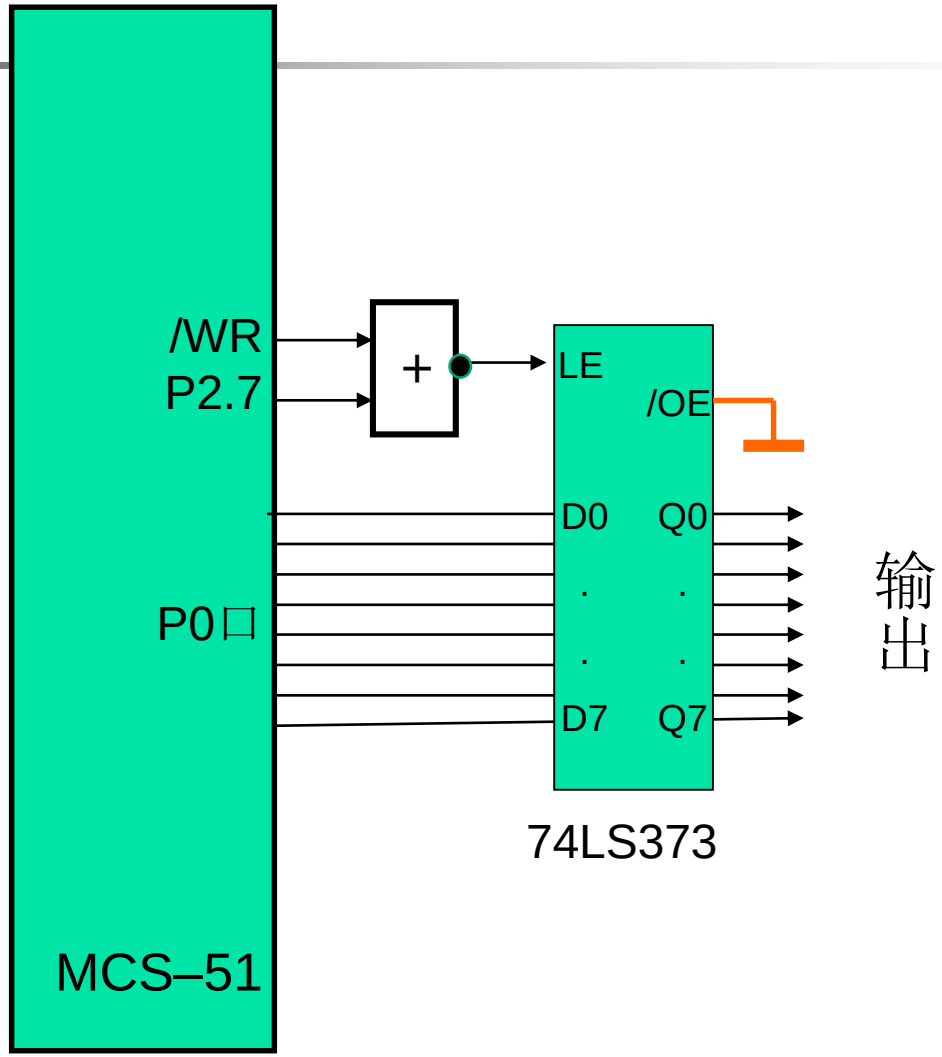
1.ROM的低位地址　　2.ROM的数据

3.RAM的低位地址　　4.RAM的数据



数据写入外部RAM

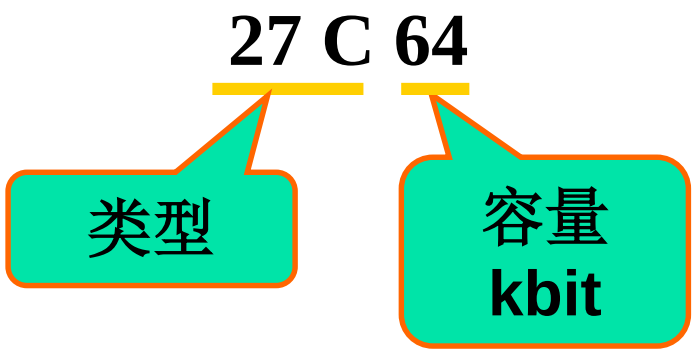
简单输出口的扩展



第八章 系统扩展
2、SRAM和ROM的命名规律

ROM/EPROM/EEPROM—27XX,28XX,29XX系列
RAM—61XX，62XX系列
串行EEPROM—24XX系列

XX：kbit = k/8 kbyte



第八章 系统扩展

3、串行总线扩展

SPI总线原理：三线制　MOSI，MISO，CLK

I^2C总线原理：二线制　　SCL，SDA

1-wire总线原理

采用串行总线的典型芯片（型号要记住）：
24C02/04/08/16,DS1307

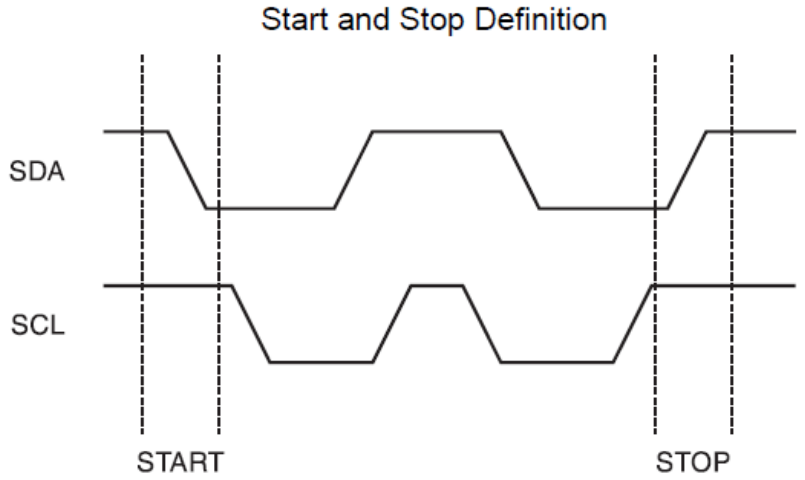
93C46

18B20

第八章 系统扩展

3、串行总线扩展

I²C总线的Start和Stop时序



推论：

在传输数据时，当SCL高电平，SDA不能变化。

因SCL由主机控制，所以：

主机在SCL低电平时输出数据至SDA；

在SCL高电平时读取SDA数据。

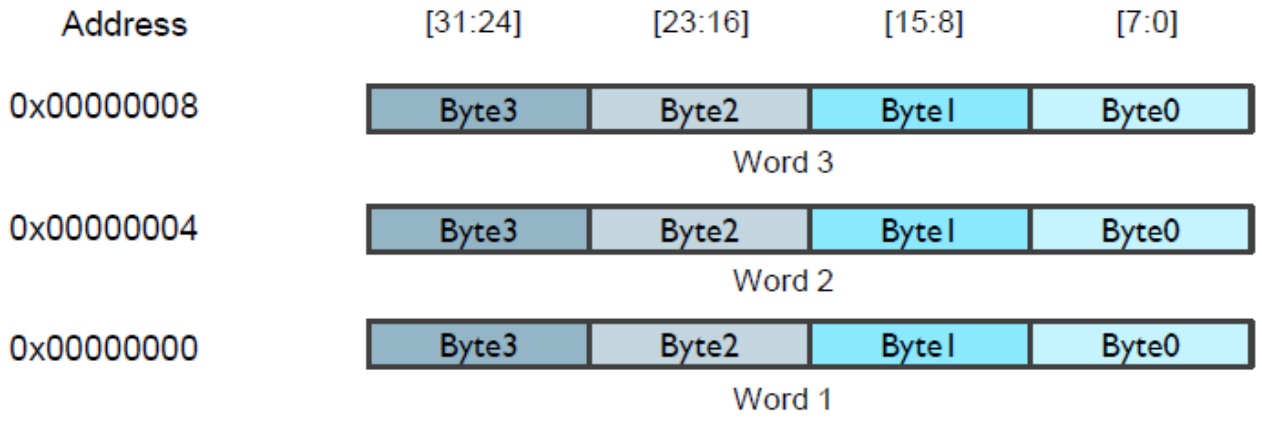
第九章 ARM基础

1、Cortex-M3/M4处理器数据宽度

- ARM Cortex-M为32位RISC处理器
  - 32位寄存器
  - 32位内部数据
  - 32位总线接口
- ARM Cortex-M采用32位寻址，地址空间4GB
- 除了32位数据，还可以处理
  - **字节/Byte** (8 bits)
  - **半字/Halfword** (16 bits= 2 bytes)
  - **字/Word** (32 bits =4 bytes)
  - **双字/Doubleword** (64 bits =8 bytes)

第九章 ARM基础

2、Cortex-M3/M4 数据存贮顺序

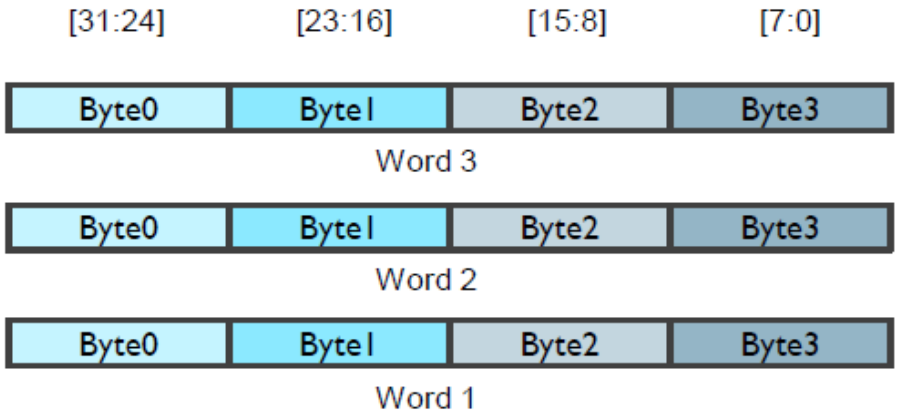


Little endian 32-bit memory

– 小端模式(Little endian): 一个Word中的最低Byte存贮在bit 0 to bit 7

– 大端模式(Big endian): 一个Word中的最低Byte存贮在bit 24 to bit 31

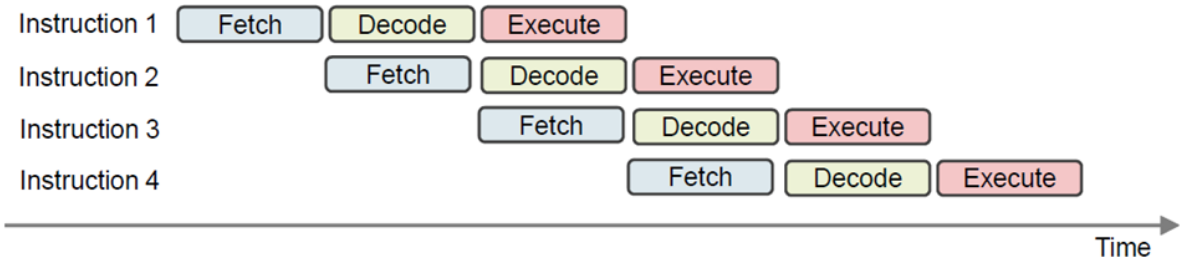
• Cortex-M4 理论上支持 little endian 和 big endian，但一般采用 little endian



Big endian 32-bit memory

第九章 ARM基础
3、Cortex-M3/M4 指令流水线

- Cortex采用支持Thumb2
  - **同时支持**ARM指令（32位指令）和Thumb指令（16位指令）
  - 兼顾了指令密度和指令速度
- 具有3级流水线
  - **取指**
  - **译码**
  - **执行**



–某些指令执行多个周期, 此情况下流水线暂停;
– 跳转到分支后流水线清空;
– 可一次读取两条指令(16-bit 指令)

第九章 ARM基础

4、通用寄存器

R0-R7：低寄存器（一些16位指令只能访问低寄存器）
R8-R12：高寄存器（可用于32位和16位指令）

R13：　栈指针（SP）
用于PUSH和POP操作，物理上实际存在两个栈指针：
主栈指针MSP和进程栈指针PSP

R14：链接寄存器（LR）
用于函数或子程序调用时返回地址的保存。

R15：程序计数器（PC）
可读可写，读操作返回当前指令地址加4；
写PC引起跳转操作。

程序状态寄存器 xPSR（标志位）

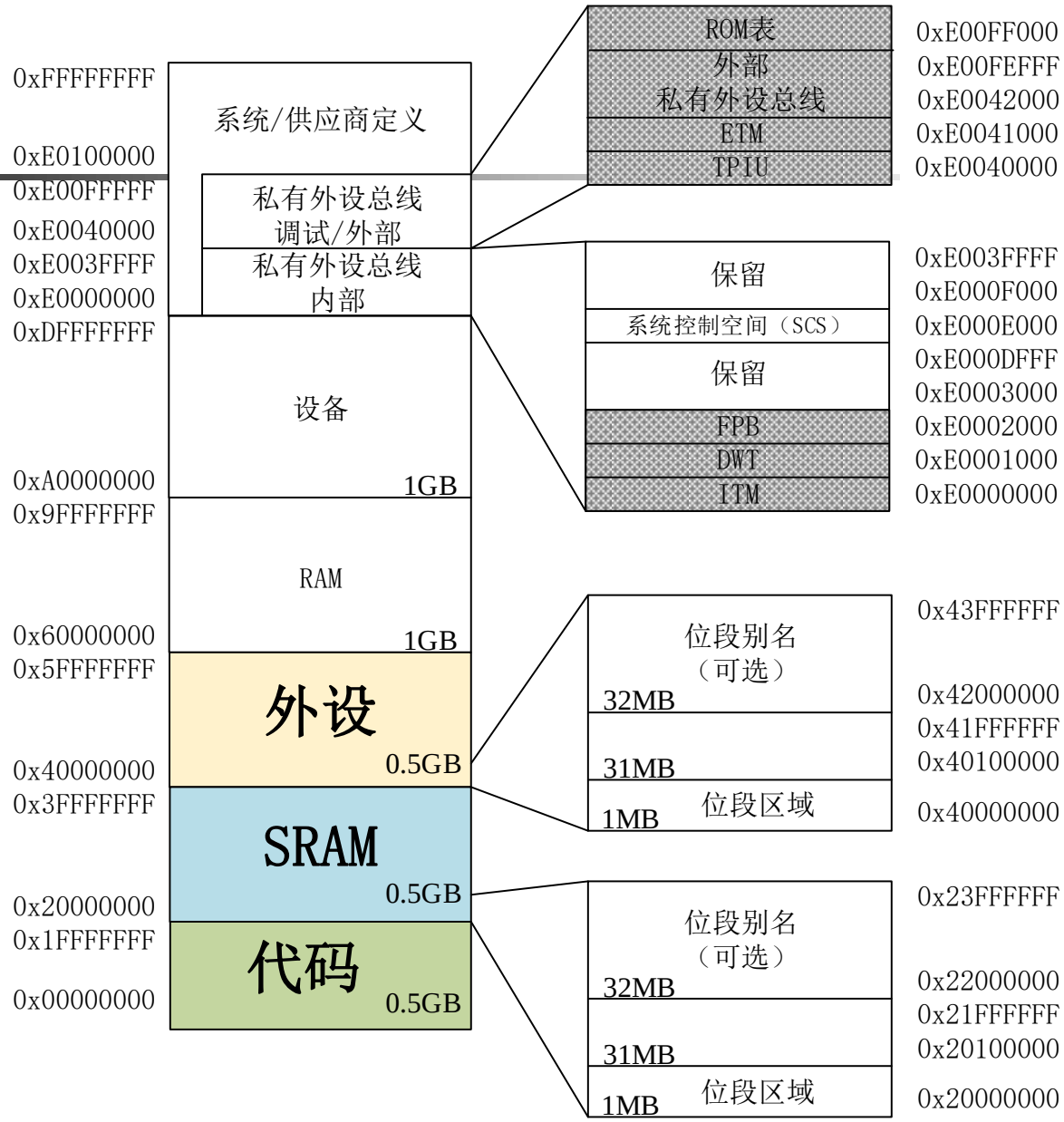
第九章 ARM基础

5、存贮器区

特性：

- 可访问多达4GB的存贮器空间
- 根据典型用法，分为：
  - **程序代码访问区（如CODE区）**
  - **数据访问区　　（如RAM区）**
  - **外设**
  - **处理器的内部控制和调试部件**

存贮器映像（4GB）


0xFFFFFFFF
系统/供应商定义
0xE0100000
0xE00FFFFF
私有外设总线
调试/外部
0xE0040000
0xE003FFFF
私有外设总线
内部
0xE0000000
0xDFFFFFFF
设备
1GB
0xA0000000
0x9FFFFFFF
RAM
1GB
0x60000000
0x5FFFFFFF
外设
0.5GB
0x40000000
0x3FFFFFFF
SRAM
0.5GB
0x20000000
0x1FFFFFFF
代码
0.5GB
0x00000000
ROM表
外部
私有外设总线
ETM
TPIU
0xE00FF000
0xE00FEFFF
0xE0042000
0xE0041000
0xE0040000
保留
系统控制空间（SCS）
保留
FPB
DWT
ITM
0xE003FFFF
0xE000F000
0xE000E000
0xE000DFFF
0xE0003000
0xE0002000
0xE0001000
0xE0000000
位段别名
（可选）
32MB
31MB
1MB
位段区域
0x43FFFFFF
0x42000000
0x41FFFFFF
0x40100000
0x40000000
位段别名
（可选）
32MB
31MB
1MB
位段区域
0x23FFFFFF
0x22000000
0x21FFFFFF
0x20100000
0x20000000

第九章 ARM基础

6、堆栈区

Cortex-M4使用的是“向下生长的满栈”模型。堆栈指针SP指向最后一个被压入堆栈的32位数值。在下一次压栈时，**SP先自减4，再存入新的数值**。

- 堆栈指针寄存器 （R13）
- PUSH 指令和 POP 指令
- 对32位操作，对齐到4字节边界，SP的最低两位总是为0
- 在物理上存在两个栈指针，主栈指针（MSP）和进程栈指针（PSP），PSP指针用于运行操作系统。

第九章 ARM基础

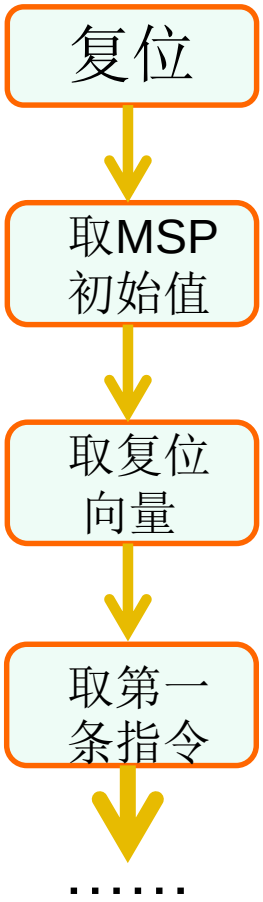
7、嵌套向量中断控制器(NVIC)

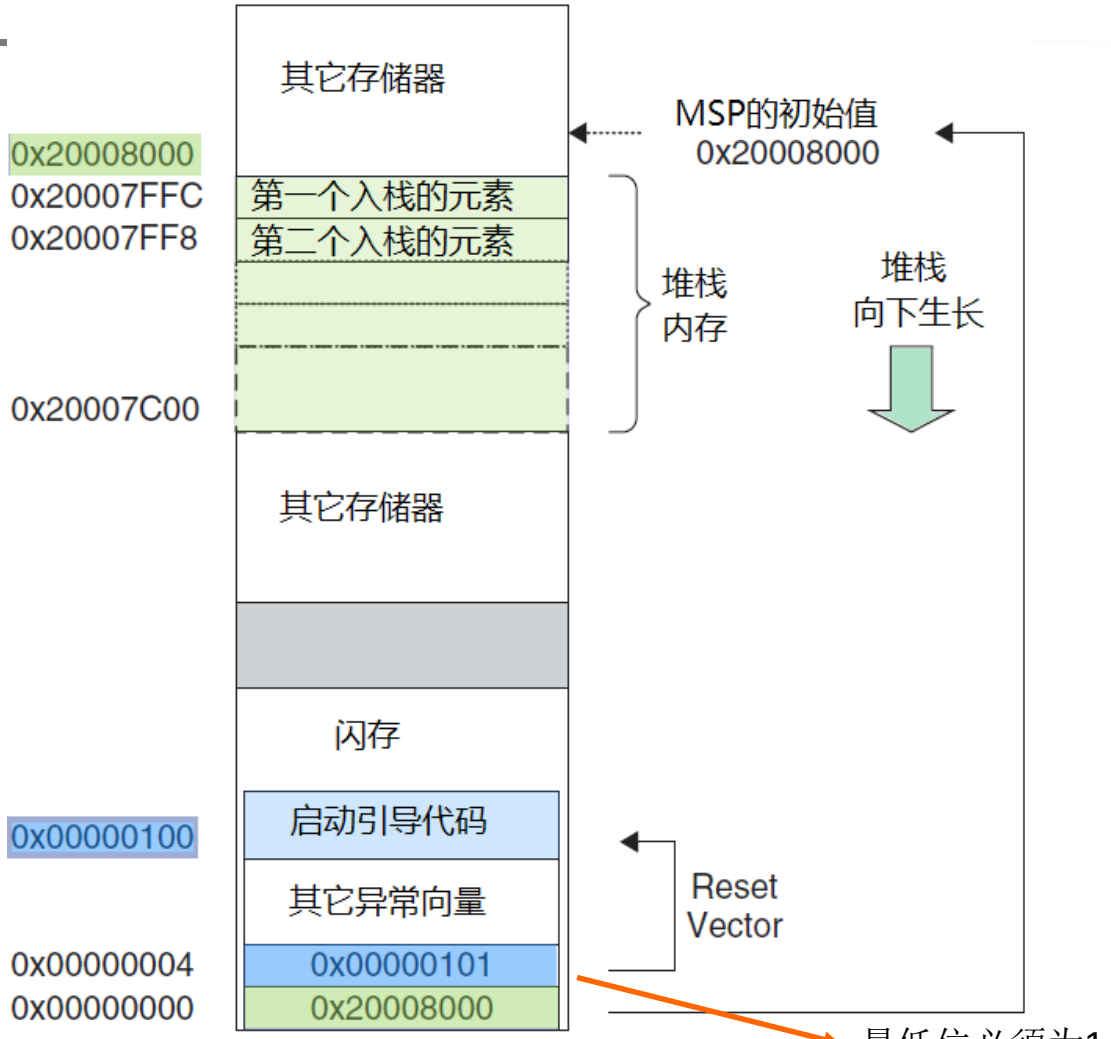
- NVIC处理异常和中断配置、优先级以及中断屏蔽，位于存贮器映像的系统控制空间（SCS）
- 中断等一些异常具有可编程的优先级。
- 中断可嵌套，即“抢占”式，高优先级可打断低优先级中断过程。
- 向量化的异常/中断入口。

第九章 ARM基础

8、复位流程

启动过程





第九章 ARM基础

9、指令集

特点：**32位或16位的**RISC指令集

- 指令丰富，功能强大

要求掌握：

（1）处理器内部数据转移指令：

MOV R1，R2

（2）存贮器访问指令：

LDRB R1，[R2,#2]

STRB R1，[R2,#4]

（3）跳转指令 B，BX

注意带后缀的条件跳转指令

（4）IT，ITE指令

第九章 ARM基础

9、指令集

重点以下存贮器访问指令：

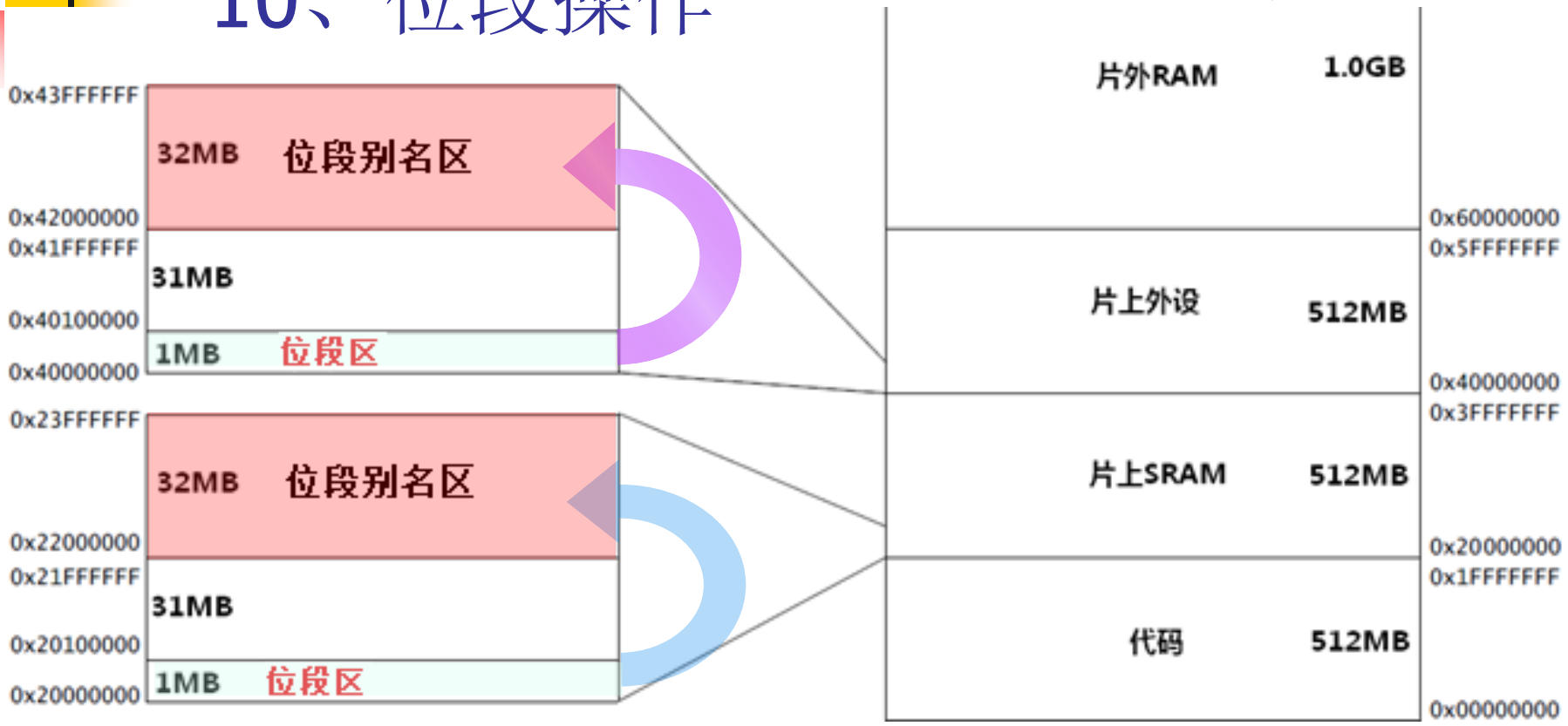
```
LDRB    R4，[R1]          ;;从存贮器地址[R1]读一个字节存入R4
LDR     R0，[R1,#08]!
                          ;;在读取[R1+0x8]到R0后，R1被更新为R1+0x8
                          ;;感叹号(!) 表示指令执行完成后更新存放地址的
                            寄存器

STR     R0, [R1]          ;;将R0的内容写到[R1]地址
STRD    R0,R1,[R2,#08]!   ;;将R0，R1作为双字，写到
                                [R2+0x8]地址并更新R2
LDR     R0, [PC, #0x20]   ;;利用PC偏移加载字数据到R0
```

使用后缀指令：

MOVS　　LDRNE　　ADDEQ

第九章 ARM基础

10、位段操作



位段操作的优点：

- 简化编程
- 位操作是一种“原子操作”，不会被其他指令打断

“不可中断的一个或一系列操作”

第十章 ARM编程

1、结构体变量的地址对应关系

优点：

- 相关的设置参数放置在一个结构体中；
- 对外设操作时，将相关寄存器地址映射到结构体的成员。
- 结构体变量地址对应该结构体的首地址

```
typedef struct
{
  __IO uint32_t MODER;
  __IO uint32_t OTYPER;
  __IO uint32_t OSPEEDR;
  __IO uint32_t PUPDR;
  __IO uint32_t IDR;
```

GPIO 寄存器映射

偏移	寄存器	31	30	29	28
0x00	GPIOA_MODER	MODER15[1:0]		MODER14[1:0]	
	Reset value	1	0	1	0
0x04	GPIOx_OTYPER (where x = A..I/)				
	Reset value				
0x08	GPIOx_OSPEEDER (where x = A..I/ except B)	OSPEEDR15[1:0]		OSPEEDR14[1:0]	
	Reset value	0	0	0	0
0x0C	GPIOA_PUPDR	PUPDR15[1:0]		PUPDR14[1:0]	
	Reset value	0	1	1	0

第十章 ARM编程

2、通用IO口

16位 GPIOA　16位 GPIOB　16位 GPIOC　……　16位 GPIOI

GPIO 端口模式寄存器 (GPIOx_MODER) (x = A..I)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位 2y:2y+1 **MODERy[1:0]**：端口 x 配置位 (Port x configuration bits) (y = 0..15)

这些位通过软件写入，用于配置 I/O 方向模式。

00：输入（复位状态）

01：通用输出模式

10：复用功能模式

11：模拟模式

4种模式

第十章 ARM编程

2、通用IO口

寄存器操作：

```
GPIOA->BSRRH= GPIO_Pin_9;       //对GPIOA.9清零
GPIOA->BSRRL= GPIO_Pin_9;       //对GPIOA.9置1
//GPIO_Pin_9预定义
#define GPIO_Pin_9                ((uint16_t)0x0200)
```

GPIO 端口置位/复位寄存器 (GPIOx_BSRR) (x = A..I)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

位 31:16 **BRy**：端口 x 复位位 y (Port x reset bit y) (y = 0..15)

这些位为只写形式，只能在字、半字或字节模式下访问。读取这些位可返回值 0x0000。

0：不会对相应的 ODRx 位执行任何操作

1：对相应的 ODRx 位进行复位

注意：如果同时对 BSx 和 BRx 置位，则 BSx 的优先级更高。

位 15:0 **BSy**：端口 x 置位位 y (Port x set bit y) (y= 0..15)

这些位为只写形式，只能在字、半字或字节模式下访问。读取这些位可返回值 0x0000。

0：不会对相应的 ODRx 位执行任何操作

1：对相应的 ODRx 位进行置位

第十章 ARM编程

3、中断系统

STM32F407包含10个内核中断和82个可屏蔽中断通道。

中断向量在启动文件startup_stm32f40_41xxx.s中设置

```
__Vectors       DCD     __initial_sp               ; Top of Stack
                DCD     Reset_Handler              ; Reset Handler
                DCD     NMI_Handler                ; NMI Handler
                DCD     HardFault_Handler          ; Hard Fault Handler
                DCD     MemManage_Handler          ; MPU Fault Handler
                DCD     BusFault_Handler           ; Bus Fault Handler
                DCD     UsageFault_Handler         ; Usage Fault Handler
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     SVC_Handler                ; SVCall Handler
                DCD     DebugMon_Handler           ; Debug Monitor Handler
                DCD     0                          ; Reserved
```

第十章 ARM编程

3、中断系统

STM32F407中断/异常的响应过程

- 入栈： 把8个寄存器的值压入栈（未使用浮点功能）
- 取向量：从向量表中找出对应的服务程序入口地址
- 选择堆栈指针MSP/PSP，更新堆栈指针SP；
  更新连接寄存器LR (LR被更新成一种特殊的值，称为“EXC_RETURN”，值为0xFFFFFFFD ， 并且在异常返回时使用。
  更新程序计数器PC

第十章 ARM编程

3、中断系统

NVIC的中断优先级：

优先级包括抢占优先级和响应优先级：

- 共占4bit，哪几个bit表示抢占优先级或响应优先级可设置；
- 编号越小，优先级越高；
- 抢占优先级高的中断可以打断优先级低的中断；
- 抢占优先级相同时，两个中断同时到达，则先处理响应优先级高的中断。

例：
中断3的抢占优先级为2，响应优先级为1；
中断6的抢占优先级为3，响应优先级为0；
中断7的抢占优先级为2，响应优先级为0；
则
中断7和中断3可打断中断6；
中断7和中断3不可互相打断；
中断7和中断3同时触发时响应中断7；

第十章 ARM编程

4、定时器

- 14个定时器

  定时器1和8：高级控制定时器，16位自动重装载；

  定时器2和5：通用定时器，32位自动重装载；

  定时器3和4：通用定时器， 16位自动重装载；

  定时器6和7：基本定时器， 16位自动重装载；

  定时器9-14：通用定时器， 16位自动重装载；

- 定时器相互完全独立，不共享任何资源；
- 功能强大，可用于定时、计数、PWM控制等；

第十章 ARM编程

4、定时器

➢**要求掌握：** PWM模式下的占空比设置

1、预分频设置：

TIM1->PSC=P-1;

2、定时器周期和计数方式设置；

TIM1->ARR= N-1;

3、PWM占空比设置

TIM1->CCR1=PWM;

第十章 ARM基础

5、直接内存访问

DMA（Direct Memory Access）：无需CPU直接控制传输，通过硬件为RAM与I/O设备开辟直接传输数据的通道，提高系统的效率。

作用：高速数据传输，如连续AD输入，DA输出，通常与定时器配合。

怎样通过设置DMA功能，连续高速采样AD口输入波形？

第十章 ARM基础

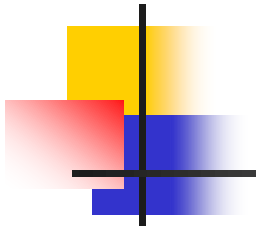
6、串行通信/接口编程

1、设置相应的IO，IO类型选特殊功能（AF）；

2、在IO口复用功能表中找到相应的功能号并设置；

3、初始化串行口参数；

	PB4	NJTRST		TIM3_CH1			SPI1_MISO	SPI3_MISO
	PB5			TIM3_CH2		I2C1_SMBA	SPI1_MOSI	SPI3_MOSI I2S3_SD
	PB6			TIM4_CH1		I2C1_SCL		
	PB7			TIM4_CH2		I2C1_SDA		
Port B	PB8			TIM4_CH3	TIM10_CH1	I2C1_SCL		
	PB9			TIM4_CH4	TIM11_CH1	I2C1_SDA	SPI2_NSS I2S2_WS	



答疑时间：待定