



微机原理原理部分期末总结



第二章

□ 了解单片机内部各组成部分

如CPU包含几个部分、程序存储器、数据存储器等

□ 内部存储器结构以及地址空间 （重点掌握）

存储空间分配、堆栈

□ 掌握一些重要的SFR：PC/DPTR，PSW等作用、内容

□ 掌握单片机引脚的种类、功能、特点

I/O口及特点，控制信号线（ALE/PSEN/RD/WR、EA）

□ 掌握单片机内部的时序单位

□ 掌握单片机工作方式



举例

- 1) **PSW**中的**P**和**F0**分别是什么什么标志位？
- 2) 低位地址锁存需要的信号？
- 3) 12M晶振，1个节拍、状态、**ALE**脉冲周期以及机器周期的时间长短？
- 4) **80C51**存储器有何特点？物理上有几个存储空间？逻辑上有几个？
- 5) **80C51**读片外程序存储器、数据存储器的控制信号是否一样？是什么？
- 6) 作为I/O时，哪个I/O口需要上拉电阻？
- 7) 堆栈设在哪里？**SP**存放的内容是？复位时多少
- 8) 查表法编程中，表数据存放在？
- 9) 低功耗方式意义？有几种？



第三章 指令系统

- 熟练掌握寻址方式：7种
- 掌握指令的执行过程
- 熟练掌握各类指令的用法及其格式

数据传送

算术运算

逻辑运算

控制转移

布尔（位）操作指令

寻址方式小结

序号	寻址方式	使用的变量	寻址空间
1	寄存器寻址	R0~R7、A、B、DPTR	
2	直接寻址		片内RAM低128B 专用寄存器
3	寄存器间接寻址	@R0、@R1、SP	片内RAM
		@R0、@R1、@DPTR	片外RAM
4	立即寻址		程序存储器
5	位寻址		片内RAM的位寻址区 可以位寻址的专用寄存器位
6	变址寻址	@A+DPTR、@A+PC	程序存储器
7	相对寻址	PC+偏移量	程序存储器



举例：常见的错误指令

MOV A, @R2

MOV R0, R1

PUSH R0

指令的执行：

LCALL/ACALL, LJMP/AJMP



第四章 汇编语言程序设计

□ 掌握4种典型的程序结构（共5种）：

顺序，分支，循环，子程序,(中断程序)

□ 掌握几种典型的程序算法：


- **数据排序（冒泡法）、极值查找、搜索编程**
- **多字节无符号数加法/减法**
- **查表**
- **能够按照要求编写、读懂完整的程序，包括伪指令**



□ 程序设计的方法、步骤、特点：

- 分析问题：确定需要做什么？
- 求解问题：确定如何做？如算法，流程
- 实现：用相应的指令组来实现
- 调试：

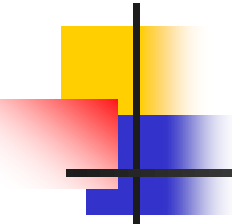
对复杂程序，通常需要流程图来规范及具体化程序设计



例：把内部RAM中起始地址为DATA的数据串传送到外部RAM以BUFFER为首地址的区域，直到发现“\$”字符的ASCII码为止，数据串的最大长度在内存20H中。

```
        MOV    R0,#DATA                ; 数据区首地址
        MOV    DPTR,#BUFFER            ; 数据区长度指针
LOOP:    MOV    A,@R0
        CJNE   A,#24H , LOOP2          ; 判是否为 “$”符(24H)
        SJMP   LOOP1                   ; 是 “$”符，则结束
LOOP2:   MOV    A,@R0                   ; 不是 “$”符，则传送
        MOVX   @DPTR,A
        INC    R0
        INC    DPTR
        DJNZ   20H,LOOP                 ; 数据串未查完，继续
LOOP1:   RET
DATA:    ...                           ; 数据串
```

注：本题中循环控制条件有二个，一个是条件循环控制，以找到ASCII码“\$”符为循环结束条件，这是主要的结构；第二个是计数循环结构，万一找不到ASCII码“\$”符，则由数据串的最大长度作为计数循环控制。



```

    ORG    0000H
    SJMP   MAIN;

    ORG    0100H;

MAIN:  MOV    DPTR, #1000H;
       MOV    R1, #20H;

CMP:   MOV    20H, #" $" ; $的ASCII码24H
       MOV    21H, #01

LP:    MOVX   A, @DPTR
       CJNE   A, 20H, LP1
       SJMP   LPEND

LP1:   INC    21H
       INC    DPTR
       DJNZ   R1, LP
       MOV    21H, #00H

LPEND: SJMP $
       END

```

这个程序是作用？

如果1000H里面开始放的数据为：

01H, 02H, 03H, 09H, 10H
 11H, 12H, 13H, 19H, 20H
 21H, 22H, 23H, 29H, 30H

21H=？



第五章 中断系统原理及应用

□ 掌握80C51的5个中断源及其中断矢量

中断优先级，内部中断源、外部中断源、清标志位（自动、需要软件）

□ 掌握中断相关的初始化、SFR设定

□ 理解中断响应的过程

□ 掌握编写中断服务程序的技能



第六章 定时/计数器原理及应用

- 掌握80C51的2个定时器T0, T1的4种工作方式
- 掌握计算计数初值, 初始化等内容
- 掌握计数方式、定时方式的应用编程
- 掌握看门狗T3的作用 (238~239)
- 掌握定时器的中断、查询方式编程



第一步：定时常数计算

第二步：TMOD的设定（即控制字）

第三步：编写程序（查询或中断法）

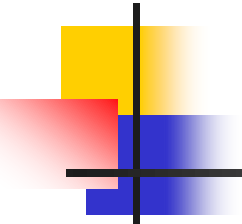
TCON 102, 109

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

位序	B ₇	B ₆	B ₅	B ₄	B ₃	B ₂	B ₁	B ₀
位符号	GATE	C/T	M ₁	M ₀	GATE	C/T	M ₁	M ₀

定时/计数器1

定时/计数器0



1、用定时器/计数器T0以定时的方法在P3.1引脚上输出周期位400us，占空比为9:1的矩形脉冲，以方式2实现。加上必要的伪指令，并对源程序加注释

9:1的波形，周期400us， 360us 40us

第一步：定时常数计算

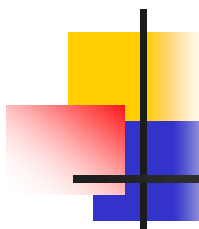
$f_{osc} = 6\text{MHz} \rightarrow T_c = 2\mu\text{s},$

方式2计数器长度 $L = 8$

定时时间 $t = 40\mu\text{s}$

计算出定时常数： $TC = 256 - 40/2 = 236 = 0\text{ECH},$

如果定时360us， $TC = 256 - 360/2 = 76 = 4\text{CH}$



ORG 0000H

AJMP MAIN

ORG 000BH;

AJMP INQP

MAIN: MOV TH0, #0ECH
MOV TL0, #0ECH;
MOV TMOD, #02H;
MOV R0, #01H;
SETB ET0;
SETB EA;
SETB TR0;
CLR P3.1;
AJMP \$

ORG 0200H

INQP: DJNZ R0, BACK;

CPL P3.1

MOV C, P3.1

JC ONE

MOV R0, #01H;

AJMP BACK

ONE: MOV R0, #09H;

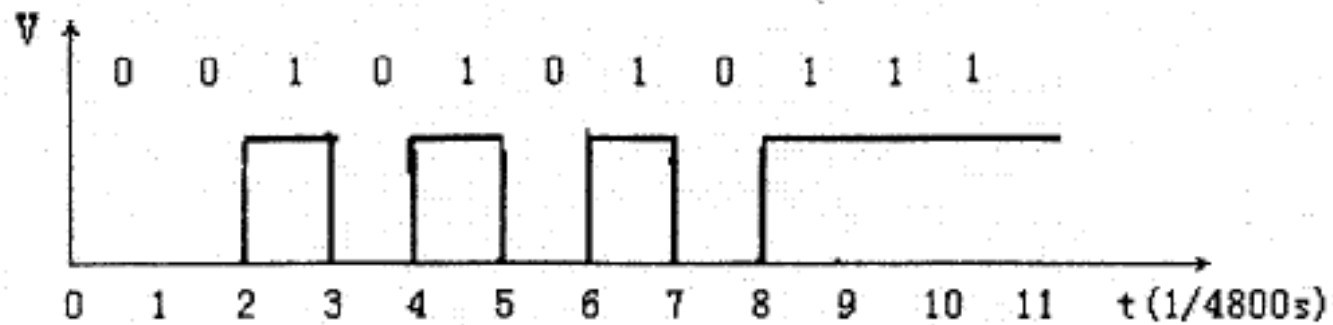
BACK: RETI

中断的方式



第七章 串行口原理及应用

- 掌握80C51串行口的4种工作方式、原理
- 掌握串行口初始化相关的SFR设置
- 掌握波特率的计算、定时器的使用
- 掌握串行口的中断、查询方式编程
- 掌握串行口发送以及接收程序的编写
- 了解多机通信方式的工作原理，不要求编程



方式3，波特率？数据？采用何种校验方式？



第X章 ARM-Cortex介绍

□ 基础知识

总线宽度、指令集类别、数据存储格式

RISC（精简指令集计算机）/CISC（复杂指令集计算机）

□ 处理器构架

寄存器PC（R15）、存储器映像（4GB空间的划分，便于程序移植）

堆栈（向下生长，4个字节，MSP/PSP）

□ 异常和中断：向量表

□ 指令集：Thumb2 RISC指令集，两种指令：32位或16位

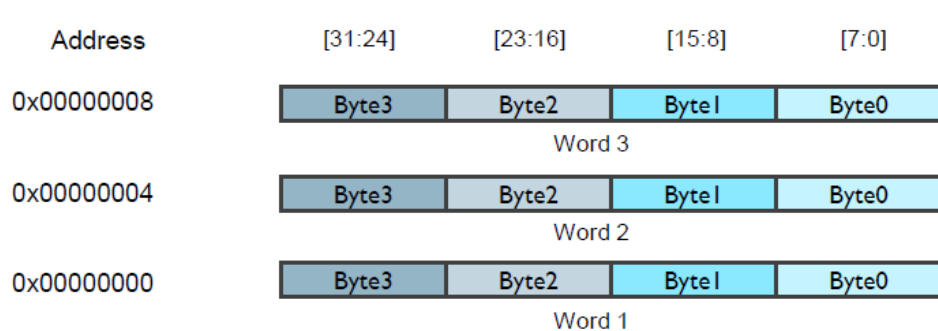
□ 存储器系统

□ 基础知识

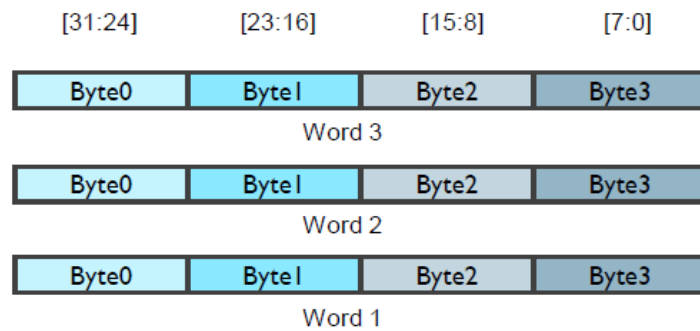
总线宽度、指令集类别、数据存储格式

RISC（精简指令集计算机）/CISC（复杂指令集计算机）

- ARM Cortex-M为32位RISC处理器
- 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：
Byte(8 bits), Halfword(16 bits), word(32 bits), doubleword (32 bits)
- 输出存储大端、小端模式



Little endian 32-bit memory



Big endian 32-bit memory

□ 处理器构架

寄存器PC (R15)、存储器映像 (4GB空间的划分, 便于程序移植)
堆栈 (向下生长, 4个字节, MSP/PSP)

寄存器

r0
r1
r2
r3
r4
r5
r6
r7
r8
r9
r10
r11
r12
r13
r14
r15

通用寄存器
通用寄存器
通用寄存器
通用寄存器
通用寄存器
通用寄存器
通用寄存器
通用寄存器
通用寄存器
通用寄存器
通用寄存器
通用寄存器
通用寄存器
栈指针 (SP)
链接寄存器 (LR)
程序计数器 (PC)

通用寄存器组

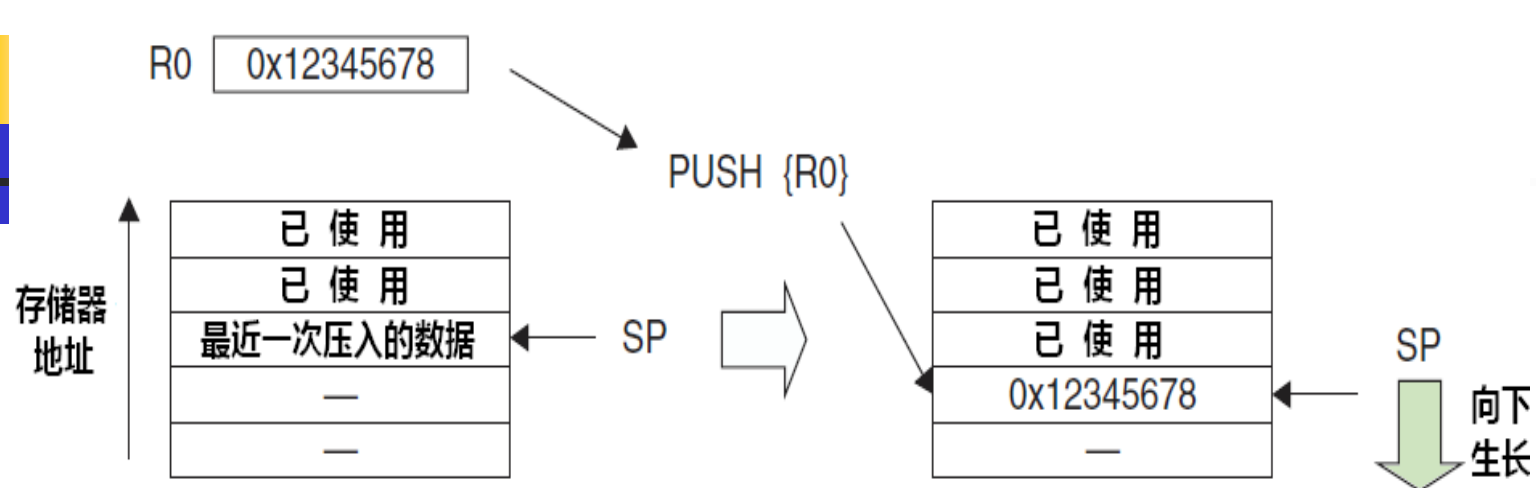
MSP

主栈指针

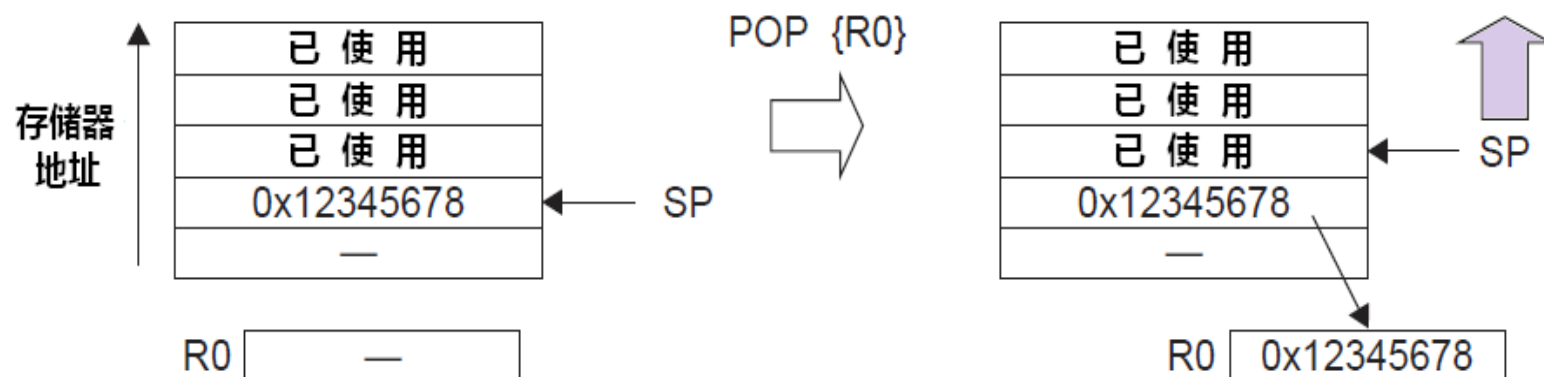
PSP

进程栈指针

ARM的堆栈区



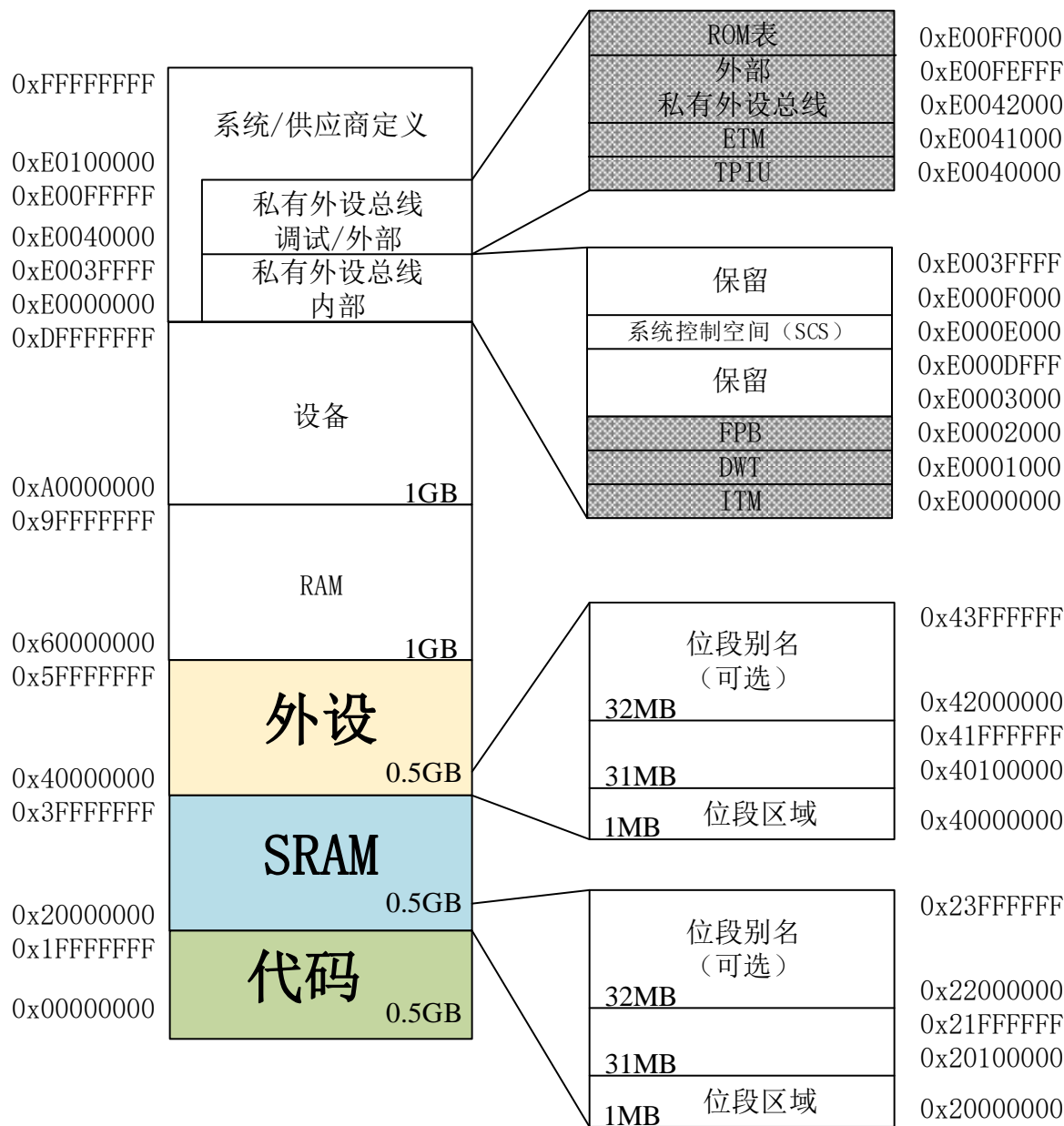
Cortex堆栈的PUSH操作, $SP=SP-4$



Cortex堆栈的POP操作, $SP=SP+4$

存储器映像 (4GB)

一般用于存放数据
但也可以存放程序
一般用于存放程序



□异常和中断:

向量表，优先级

- 异常是会改变程序流的事件，当其产生时，处理器暂停当前执行的任务，转而执行异常处理程序。
- 在ARM架构中，中断是异常的一种。
- 当Cortex内核响应了一个发生的异常后，对应的异常服务例程(ESR)就会执行。
- 为了决定ESR的入口地址，Cortex使用了“向量表查表机制”
- 向量表其实是一个32位整数数组，每个元素对应一种异常，该元素的值则是该ESR的入口地址。
- 向量表在地址空间中的位置是可以设置的，通过NVIC中的一个重定位寄存器来指出向量表的地址。复位后，该寄存器的值为0。因此，在地址0处必须包含一张向量表，用于初始时的异常分配。

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

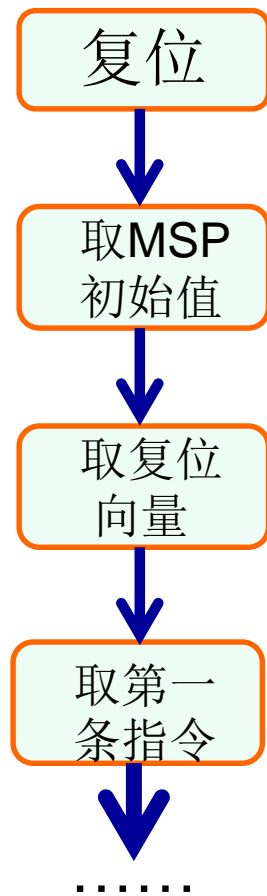
中断7和中断3可打断中断6;

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

复位流程

启动过程

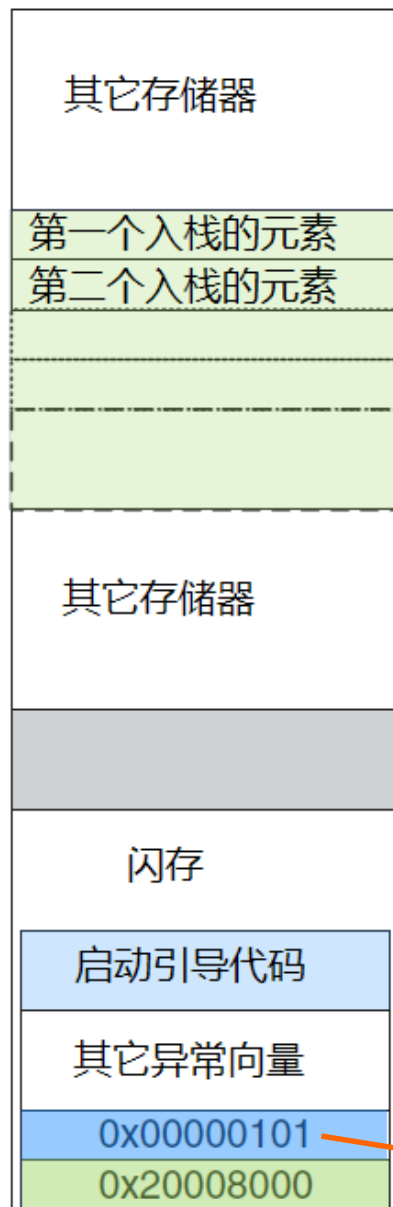


0x20008000
0x20007FFC
0x20007FF8

0x20007C00

0x00000100

0x00000004
0x00000000



MSP的初始值
0x20008000

堆栈
向下生长

Reset
Vector

最低位必须为1,
Thumb状态执行



□指令集:

Thumb2 RISC指令集，两种指令：32位或16位

M3/M4: Thumb-2指令集（ISA: Instruction Set Architecture）

- 32位或16位的RISC指令集
- 指令丰富，功能强大

80C51：单字节、双字节、三字节

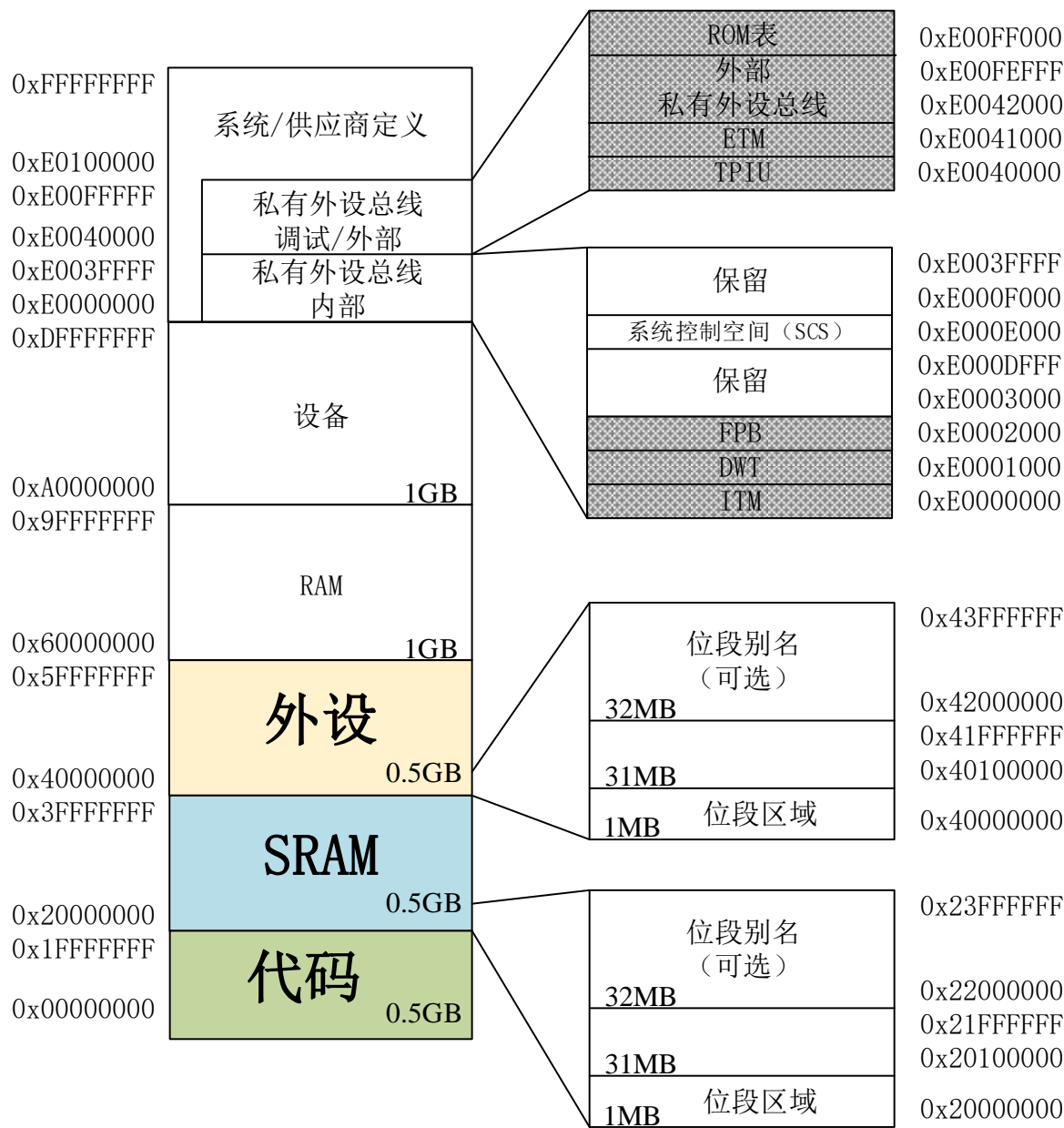
□ 存储器系统

32位/4G，存储器映像、多总线、大端/小端

- 32位总线，4GB存储器空间
- 指令和数据空间统一编址（存储器映像）
- 哈佛结构，指令和数据可以同时访问
- 基于AMBA（高级微控制器总线结构）的总线接口设计：有多个总线
- 同时支持大端和小端的存储器系统

存储器映像 (4GB)

一般用于存放数据
但也可以存放程序
一般用于存放程序



ARM处理器连接存储器和外设

