第5章 STC单片机CPU子系统

何宾

2018.03

本章主要内容

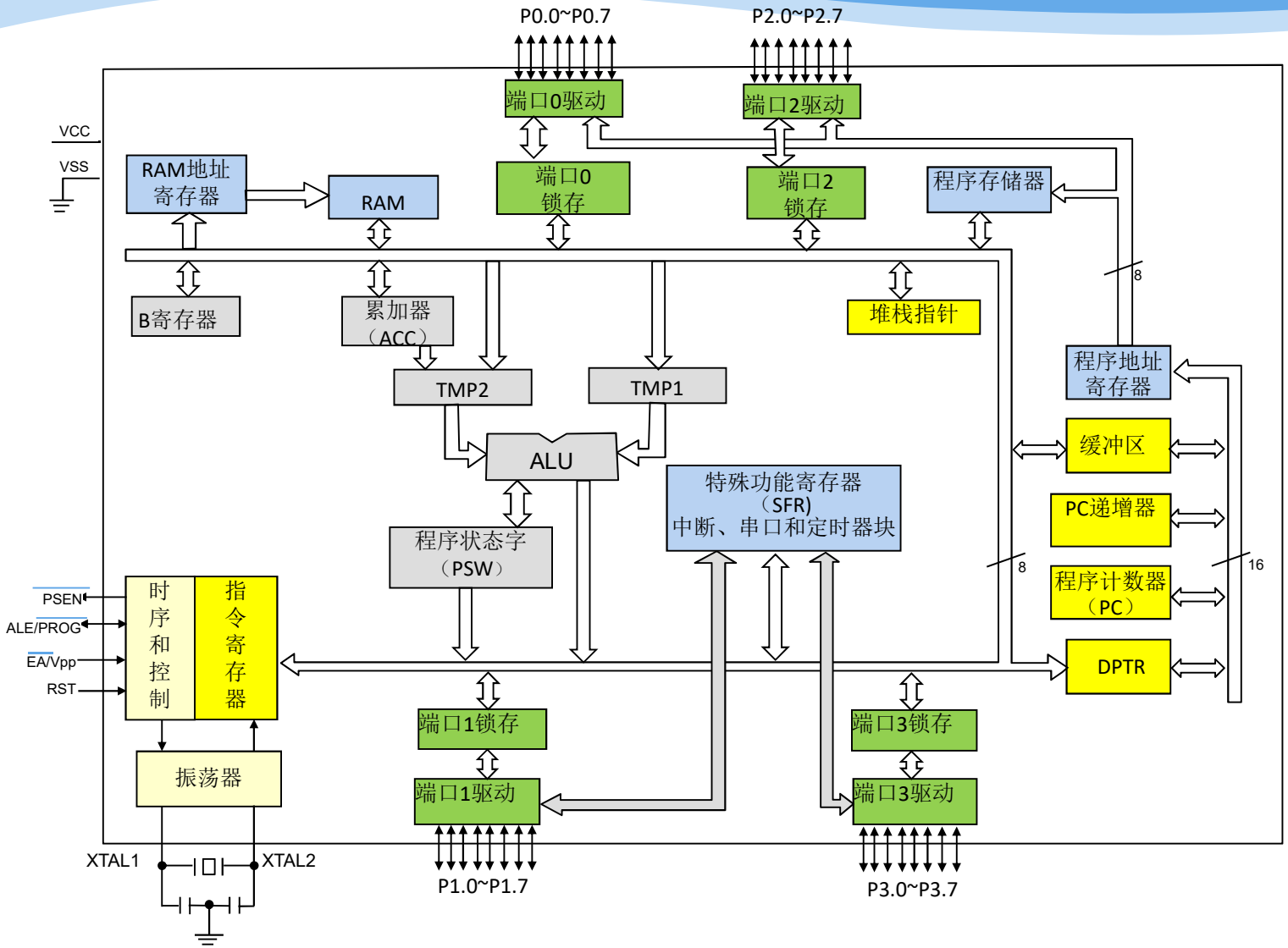
- STC单片机CPU内核功能单元
- STC单片机存储器结构和地址空间
- STC单片机中断系统原理及功能

STC单片机CPU内核功能单元

8051单片机自诞生的那天开始，到现在已经持续了30多年。

- 在这期间，人们对其性能不断的进行改进，使得其整体性能提高了10倍以上。
- 目前，以8051 CPU内核为核心的单片机仍然发挥着其巨大的生命力。
- 虽然8051 CPU的内核比较简单，但是以其为核心的单片机系统却包含了构成计算机系统的全部要素.

STC单片机CPU内核功能单元



STC单片机CPU内核功能单元

STC内的8051 CPU核是高性能、运行速度经过优化的8位中央处理单元（Central Processing Unit，CPU）。8051 CPU外围主要包括：

- 内部数据RAM；
- 外部数据空间；
- 特殊功能寄存器；
- CPU时钟分频器。

STC单片机CPU内核功能单元

STC 8051 CPU的特性主要包括：

- 采用流水线RISC结构，执行速度比工业标准8051快十几倍
- 与工业标准8051指令集100%兼容；
- 大多数指令使用1个或2个时钟周期执行；
- 256个字节的内部数据RAM；
- 使用双DPTR扩展标准8051结构；
- 提供了片外扩展的64KB外部数据存储器；
- 提供了多达21个中断源；
- 新特殊功能寄存器使能：快速访问STC单片机I/O端口，以及控制CPU时钟频率；

STC单片机CPU内核功能单元

任何一个中央处理单元CPU都包含有控制器和运算器两大基本模块。下面将通过STC单片机分析8051 CPU子系统的功能。

STC单片机CPU内核功能单元
--控制器

控制器是CPU中最重要的功能部件之一。其作用是控制CPU内的各个组成部件协调的工作，保证CPU的正常运行。

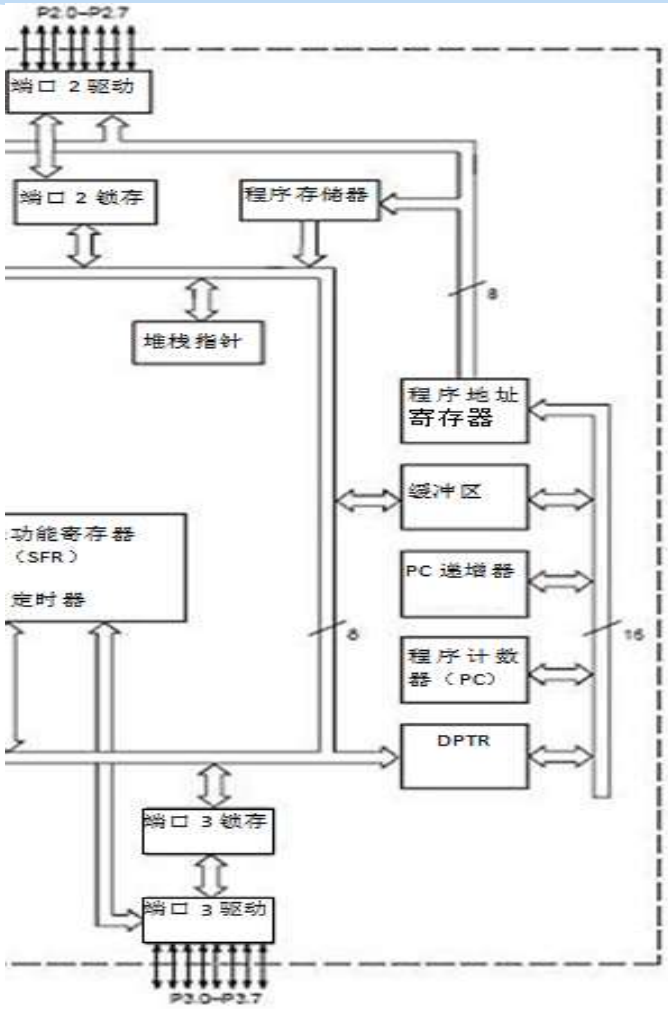
控制器
--程序计数器

单片机最重要的特点之一就是采用了存储程序的体系结构，即需要执行的代码保存在一个称之为程序存储器的单元中。

- 通过程序计数器（Program Counter，PC）从程序存储器中源源不断地取出所要执行的代码。因此，程序计数器PC是CPU中最基本的控制部分。
- PC的特点就是总是指向下一条所要执行的指令的地址空间。
- 程序计数器、PC递增计数器、缓冲区、程序地址寄存器都挂在其结构右侧的一条总线上。程序地址存储器的输出连接到程序存储器上，而程序存储器连接到内部总线上。

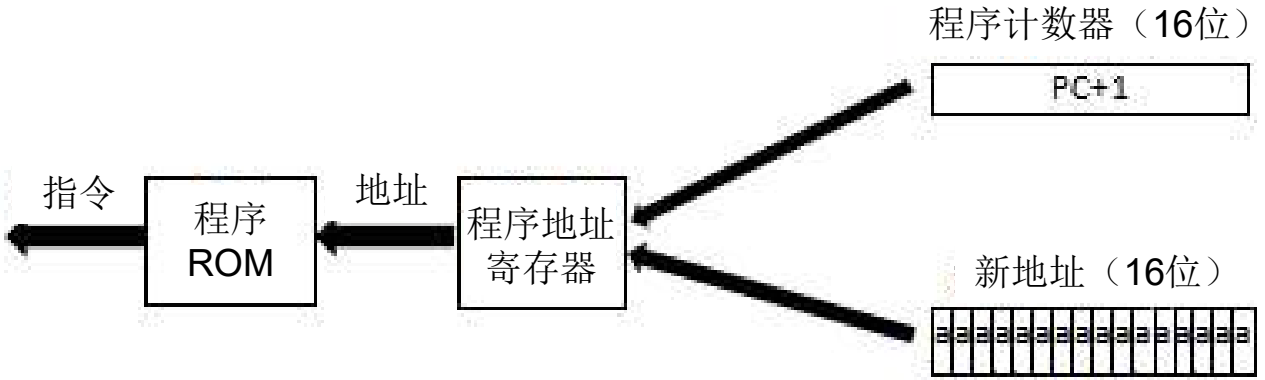
控制器
--程序计数器

- 前面已经提到在程序存储器中，保存的是程序的机器代码，即机器指令。
- 从图中可以知道，程序地址寄存器的输出用于给程序存储器提供地址，而程序存储器的输出用于提供机器指令的内容。



控制器
--程序计数器

- 程序计数器其实质就是实现递增功能的计数器而已，只不过是因为计数器的计数值作为程序存储器的地址而已。
- 在上一图中，程序计数器的宽度为16位。也就是说，地址深度为2^{16}，地址的范围为0~65535，即64K。因此，程序存储器的深度最大为64K。
- 程序计数器并不能总是让程序地址寄存器递增。这是因为，执行机器指令可以分成顺序执行和非顺序执行，如下图。



控制器
--程序计数器

- 顺序执行
  - 是指按机器指令的前后顺序，顺序执行执行指令，即：把PC+1后的值送给程序地址寄存器，作为程序ROM的地址。
  - 然后，程序ROM送出指令。这就是所说的，程序计数器总是指向下一条要执行的指令。
- 非顺序执行
  - 在编写的软件代码中，经常出现条件判断语句、跳转语句、程序调用语句和中断调用等。
  - 因此，当执行程序代码的过程中遇到这些指令时，程序的执行顺序并不是按照PC+1->PC来执行程序，而是将这些语句所指向的新指令所在的新目标地址赋给程序地址寄存器。

STC单片机CPU内核功能单元
--指令通道

指令通道包含取指单元、译码单元、执行指令单元。本质上，取值、译码和执行指令实质上就是一个有限自动状态机。也就是经常所说的，微指令控制器。

- 取指单元
  - 根据PC所指向的存放指令的程序存储器地址，取出指令。
  - 在8051单片机中，程序存储器的宽度为8位。但是，8051的机器指令有8位、16位或24位，即：1个字节、2个字节或3个字节。
  - 所以，对于不同的指令来说，取值令所需要的时钟周期并不相同。因此，可能需要几个时钟周期才能完成取指操作。

STC单片机CPU内核功能单元
--指令通道

- **译码单元**
  - 根据取出指令的操作码部分，对指令进行翻译。
  - 这个翻译过程，就是将机器指令转换成一系列的逻辑控制序列，这些控制序列将直接控制CPU内的运算单元。
- **执行指令单元**
  - 当完成译码过程后，根据逻辑控制序列（微指令）所产生的逻辑行为，控制运算器单元，完成指令需要的实现的操作行为。

STC单片机CPU内核功能单元
--指令通道

比如，假设此时取出一条机器指令（汇编助记符表示）：

ADD　A，Rn

其机器指令的格式为：

0010，1rrr　　// rrr表示寄存器的编号

该指令完成寄存器Rn和累加器A数据相加的操作。

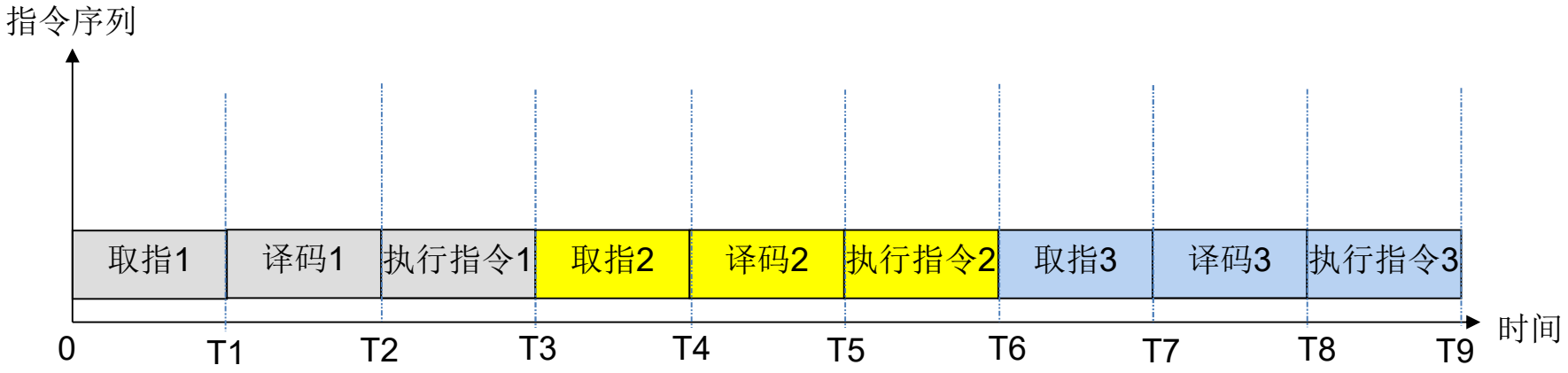
STC单片机CPU内核功能单元
--指令通道

- 其译码和执行指令的过程应该包含：
  - 从寄存器Rn中取出数据送入ALU的一个输入端口TMP1，表示为：
    （Rn）->（TMP1）
  - 从累加器ACC中取出数据送入ALU另一个输入端口TMP2，表示为：
    （ACC）->（TMP2）
  - 将TMP1和TMP2的数据送到ALU进行相加，产生结果，表示为：
    （TMP2）+（TMP1）->（ALU）
  - 将ALU产生的结果，通过内部总线送入到ACC累加器中。
    （ALU）->（总线）->（ACC）

STC单片机CPU内核功能单元
--指令通道

流水线技术

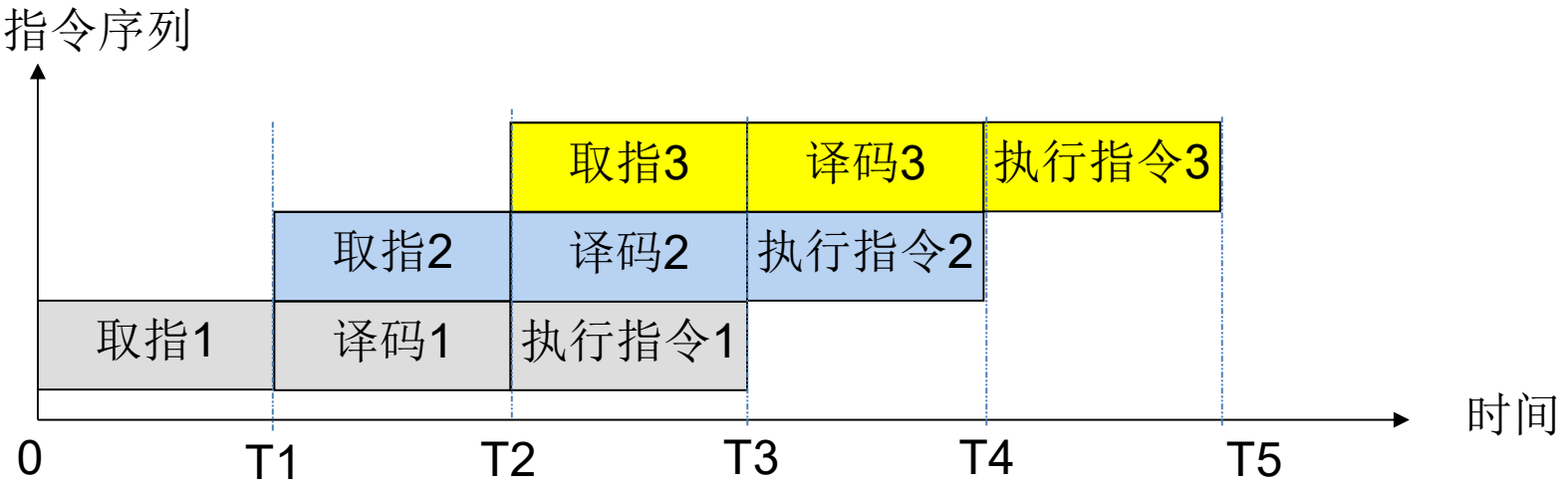
传统8051单片机的指令通道采用的是串行结构，如下图所示。为了分析问题方便，下面假设每条指令的取值周期、译码周期和执行指令的周期都是一样的。当采用串行执行结构时，对于3条指令来说，需要T9个时间周期才能执行完成。



3条指令执行的串行执行结构原理

STC单片机CPU内核功能单元
--指令通道

与传统的8051 CPU取指通道相比，STC指令通道采用了改进后的（二级/三级）流水线结构，如图所示。从图中可以看出，当采用三级流水线结构以后，只需要T5个周期就能执行完3条指令。基于前面的假设条件，指令通道的吞吐量提高了1倍。当执行指令的数量增加时，流水线结构优势将更加明显。



3条指令执行的流水线结构原理

STC单片机CPU内核功能单元
--双数据指针

数据指针（Dual Data Pointer，DPTR）是一个16位的专用寄存器。

- 由DPL（低8位）和DPH（高8位）组成，其地址为82H（DPL，低字节）和83H（DPH，高字节）。
- DPTR是8051中唯一可以直接进行16位操作的寄存器。
- 此外，也可以按照字节分别对DPH和DPL进行操作。

STC单片机CPU内核功能单元
--双数据指针

如果STC单片机没有外部数据总线，则该单片机只存在一个16位的数据指针。

- **如果单片机有外部数据总线，则该单片机设计了两个16位的数据指针DPTR0和DPTR1，这两个数据指针公用一个地址空间。**
- **通过软件设置特殊功能寄存器（Special Function Register，SFR）中P_SW1（地址为A2H）的第0位选择数据指针。**

Mnemonic	Address	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR1 P_SW1	A2H	Auxiliary Register 1	S1_S1	S1_S0	CCP_S1	CCP_S0	SPI_S1	SPI_S0	0	DPS	0000,0000

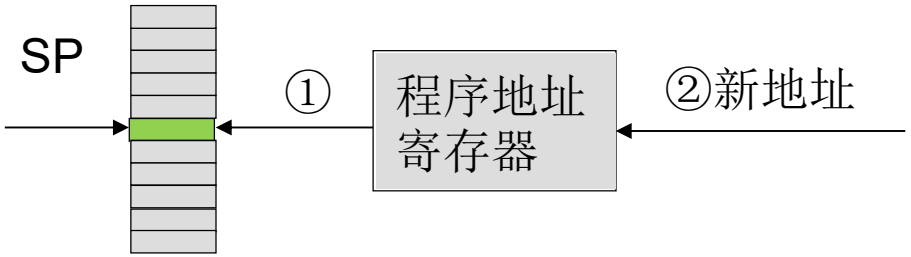
STC单片机CPU内核功能单元
--双数据指针

- 数据指针选择（Data Pointer Select，DPS）来选择所使用的DPTR。
  - DPS=0，选择DPTR0（0x83:0x82），其中：0x83为16位DPTR0的高寄存器DPH0；0x82为16位DPTR0的低寄存器DPL0；
  - DPS=1，选择DPTR1（0x83:0x82），其中：0x83为16位DPTR1的高寄存器DPH1；0x82为16位DPTR1的低寄存器DPL1；

STC单片机CPU内核功能单元
--堆栈及指针

在单片机中，有一个称为堆栈的特殊存储空间。

- 其作用主要用于保存现场。
  - 典型的，当在执行程序的过程中遇到跳转指令时，就需要将当前PC+1指向的下一跳指令的地址保存起来，等待执行完跳转指令的时候，再将所保存的下一条指令的地址恢复到程序地址寄存器中。



STC单片机CPU内核功能单元
--堆栈及指针

在STC单片机中，用于控制指向存储空间位置的是一个堆栈指针（Stack Pointer，SP）

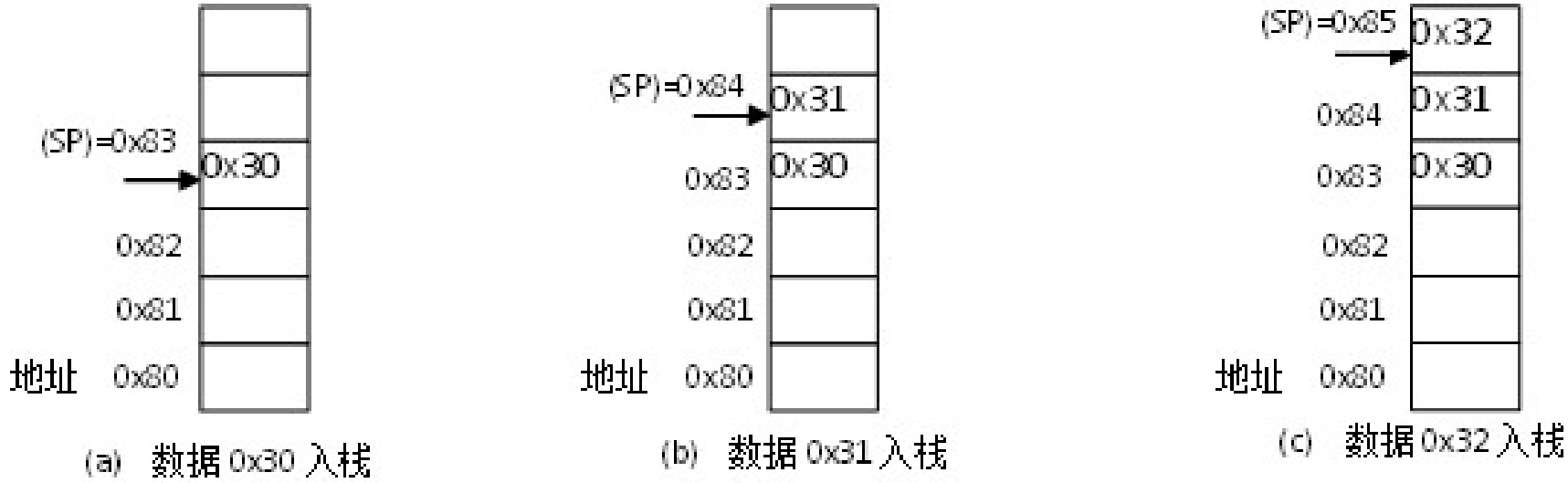
- 它实际上就是一个8位的专用寄存器
- 该寄存器的内容就是栈顶的地址，也就是用于表示当前栈顶在内部RAM块中的位置。

以三个数据0x30、0x31和0x32入栈和出栈为例。假设在对堆栈进行操作前，当前SP的内容为0x82，也就是SP指向堆栈存储空间地址为0x82的位置。

STC单片机CPU内核功能单元
--堆栈及指针

入栈操作

- 从该过程可以看出，随着数据的入栈操作，（SP）递增，SP总是指向最新保存的数据的存储器的位置，也就是通常所说的，SP总是指向栈顶的位置

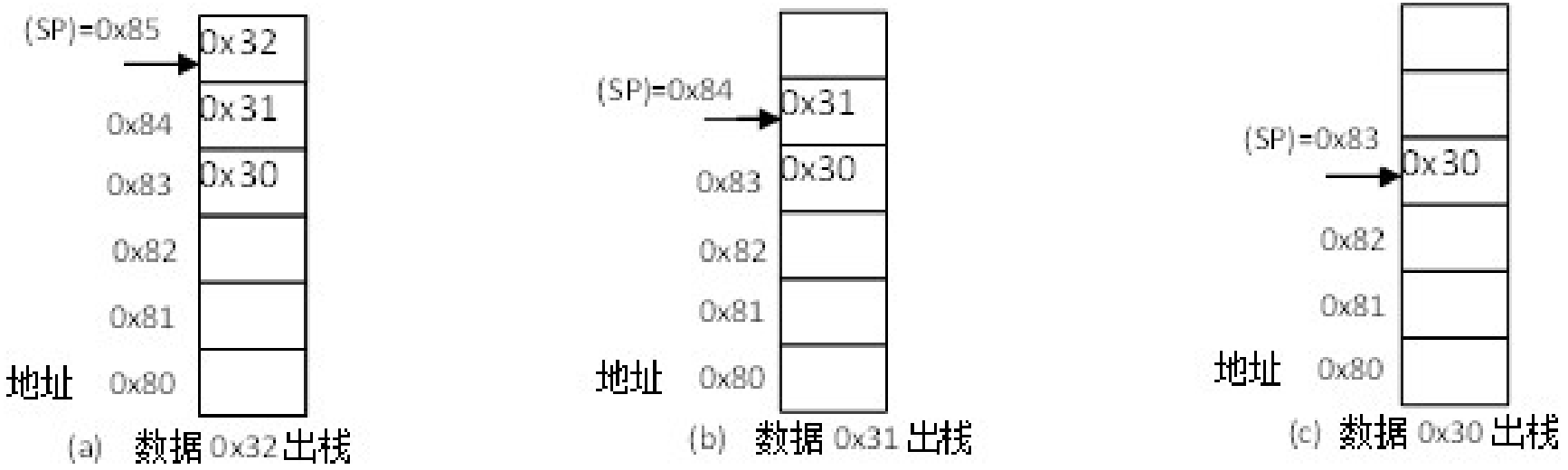


数据入栈操作

STC单片机CPU内核功能单元
--堆栈及指针

出栈操作时

- 从该过程可以看出，随着数据的入栈操作，（SP）递增，SP总是指向最新保存的数据的存储器的位置，也就是通常所说的，SP总是指向栈顶的位置。



数据出栈操作

STC单片机CPU内核功能单元
--运算器

运算器用于执行丰富的数据操作功能。STC的8051 CPU内的运算器包括：

- 8位算术逻辑单元
- 累加器
- B寄存器
- 程序状态字

运算器
--8位算术逻辑单元

STC的8051 CPU内的运算器中，最核心的部件就是算术逻辑单元（Arithmetic and Logic Unit，ALU）。8051 CPU的ALU位宽为8位，它可以实现的功能包括：

- 算术运算
  - 典型地，实现8位的加、减、乘和除运算；
- 其它运算
  - 典型地，实现递增、递减、BCD十进制调整和比较运算；
- 逻辑运算
  - 典型地，实现逻辑与（AND）、逻辑或（OR）、逻辑异或（XOR），逻辑取反（NOT）和旋转/移位操作；
- 按位运算
  - 典型地，置位、复位、取补、如果没有设置则跳转操作、如果设置则跳转并且清除操作和移入/移出进位标志寄存器；

运算器
--累加器

累加器（Accumulator，ACC），也简写为A

- 用于大多数指令结果的累加器。
- 为累加器所分配的地址空间位于特殊功能寄存器（Special Function Register，SFR）0xE0的位置。

运算器
--B寄存器

在乘法和除法操作中，B寄存器有特殊用途；而在其它情况，它用于通用寄存器。

- 乘法操作
  - 参与乘法运算的一个操作数保存在B寄存器中，另一个保存在A寄存器中。
  - 在乘法运算后，所得乘积的高8位保存在B寄存器中，而乘积的低8位保存在A寄存器中。
- 除法操作
  - 参与除法运算的被除数保存在A寄存器中，除数保存在B寄存器中
  - 在除法运算后，所得的商保存在A寄存器中，而余数保存在B寄存器中。

运算器
--程序状态字

在程序状态字（Program Status Word，PSW）寄存器中，保存一些具有特殊含义的比特位，这些位反映当前8051 CPU内的工作状态。

- 该寄存器位于SFR空间0xD0地址。下表给出PSW寄存器的内容。

比特位	7	6	5	4	3	2	1	0
名字	CY	AC	F0	RS1	RS0	OV	RSV	P

运算器
--程序状态字

- CY
  - 进位标志。
  - 算术和位指令操作影响该位。
  - 在进行加法运算时，如果最高位有进位；或者在进行减法运算时，如果最高位有借位，则将CY设置为1；否则设置为0。
- AC
  - 辅助进位标志。
  - ADD，ADDC，SUBB指令影响该位。
  - 在进行加法运算时，如果第3位向第4位有进位；或者在进行减法运算时，如果第3位向第4位有借位，则将AC设置为1；否则设置为0。
  - 设置辅助进位标志的目的是为了便于BCD码加法、减法运算的调整。

运算器
--程序状态字

- **F0**
  - 通用标志0。
- **RS1和RS0**
  - 寄存器组选择位。用于选择不同的寄存器组，其含义如下表所示。

RS1	RS0	当前使用的工作寄存器组（ (R0~R7)
0	0	0
0	1	1
1	0	2
1	1	3

运算器
--程序状态字

- OV
  - 溢出标志。
  - ADD、ADDC、SUBB、MUL和DIV指令影响该位状态。
- RSV
  - 保留位。
- P
  - 奇偶标志位。
  - 在每条指令执行后，设置或者清除该比特位，该位用于表示累加器ACC中1的个数。如果ACC中有奇数个1时，则将P设置为1；否则，如果ACC中有偶数个1（包括0个1的情况）时，则将P设置为0。

STC单片机CPU内核功能单元
--特殊功能寄存器

特殊功能寄存器（Special Function Register，SFR），SFR是具有特殊功能的RAM区域，它是一系列控制寄存器和状态寄存器的集合。

- 这些寄存器用于对STC单片机内的各个功能模块进行管理、控制和监视。

注：（1）STC15系列单片机的SFR和高128字节的RAM共用相同的地址范围，都使用80H~FFH的区域。因此，需要使用直接寻址方式访问SFR。

（2）当寄存器地址能够被8整除时才可以进行位操作，其他区域不可以进行位操作。

STC单片机CPU内核功能单元

--特殊功能寄存器

地址	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	地址
0xF8	P7	CH	CCAP0H	CCAP1H	CCAP2H				0xFF
0xF0	B	PWMCFG	PCA_PWM0	PCA_PWM1	PCA_PWM0	PWMCR	PWMIF	PWMFDCR	0xF7
0xE8	P6	CL	CCAP0L	CCAP1L	CCAP2L				0xEF
0xE0	ACC	P7M1	P7M0				CMPCR1	CMPCR2	0xE7
0xD8	CCON	CMOD	CCAPM0	CCAPM1	CCAPM2				0xDF
0xD0	PSW	T4T3M	T4H RL_TH4	T4L RL_TL4	T3H RL_TH3	T3L RL_TL3	T2H RL_TH2	T2L RL_TL2	0xD7
0xC8	P5	P5M1	P5M0	P6M1	P6M0	SPSTAT	SPCTL	SPDAT	0xCF
0xC0	P4	WDT_CONTR	IAP_DATA	IAP_ADDRH	IAP_ADDRL	IAP_CMD	IAP_TRIG	IAP_CONTR	0xC7
0xB8	IP	SADEN	P_SW2		ADC_CONTR	ADC_RES	ADC_RESL		0xBF
0xB0	P3	P3M1	P3M0	P4M1	P4M0	IP2	IP2H	IPH	0xB7
0xA8	IE	SADDR	WKTCL WKCTL_CNT	WKTCH WKTCH_CNT	S3CON	S3BUF		1E2	0xAF
0xA0	P2	BUS_SPEED	AUXR1 P_SW1						0xA7
0x98	SCON	SBUF	S2CON	S2BUF		P1ASF			0x9F
0x90	P1	P1M1	P1M0	P0M1	P0M0	P2M1	P2M0	CLK_DIV PCON2	0x97
0x88	TCON	TMOD	TL0 RL_TL0	TL1 RL_TL1	TH0 RL_TH0	TH1 RL_TH1	AUXR	INT_CLK0 AUXR2	0x8F
0x80	P0	SP	DPL	DPH	S4CON	S4BUF		PCON	0x87