فصل ۱۱

برنامه نویسی میکروکنترلر ۸۰۵۱ به زبان C51

مقدمه

در فصول قبل، با برنامه نویسی میکروکنترلر ۸۰۵۱ با زبان اسمبلی آشنا شدیم. زبان اسمبلی کاربر را از کار با کدهای ماشین معاف می کند؛ اما پیاده سازی برنامه های مفصل و پیچیده به زبان اسمبلی کاری دشوار و وقتگیر است (نمونه ساده ای از آن را در پیاده سازی شرطها دیدیم).

امروزه برای برنامه نویسی میکروکنترلرها اغلب از زبانهای سطح بالا استفاده می شود که زبان C51، برای برنامه نویسی میکروکنترلر ۸۰۵۱ به کار می رود.
مزایای استفاده از زبانهای سطح بالا برای برنامه نویسی میکروکنترلر ها عبارتند از :

- سادگی و صرف زمان کمتر برای نوشتن برنامه های بزرگ و پیچیده
- سهولت تغییر و به روز کردن برنامه ها
- قابل فهم تر بودن برنامه های زبانهای سطح بالا نسبت به برنامه های زبان اسمبلی
- قابلیت استفاده از توابع از پیش آماده شده کامپایلر
- قابلیت استفاده از کد نوشته شده برای یک میکروکنترلر در میکروکنترلرهای دیگر پس از تغییرات ناچیز

البته همانطور که در فصل ۳ ذکر شد، برنامه های زبانهای سطح بالا توسط کامپایلر به زبان ماشین قابل فهم میکروکنترلر تبدیل می شوند؛ هرچند تلاشهای زیادی برای بهینه کردن کامپیلرها در روش تبدیل دستورات سطح بالا به دستورات ماشین انجام می شود، اما فایلهای زبان ماشین تولید شده توسط کامپایلرها نسبت به فایلهای زبان ماشین تولید شده توسط اسمبلرها به نحو قابل ملاحظه ای بزرگتر و کندتر است. به بیان دیگر اگر برای انجام یک عملیات واحد، یک برنامه به زبان سطح بالا نوشته و توسط کامپایلر به زبان ماشین تبدیل کنیم و برنامه ای دیگر به زبان اسمبلی نوشته و با اسمبلر کد زبان ماشین معادل آن را به دست آوریم، حجم فایل زبان ماشین اول بین ۱/۵ تا دو برابر حجم فایل زبان ماشین دوم بوده و اجرای آن نیز به همین نسبت بیشتر به طول می انجامد.
کلاً می توان گفت برای کاربردهای معمولی که حجم کد ماشین و زمان اجرای برنامه چندان مهم نیست، استفاده از زبان سطح بالا برای برنامه نویسی میکروکنترلر نسبت به زبان اسمبلی ساده تر و سریعتر است. البته در یک برنامه سطح بالا در مواردی که حجم یا زمان اجرای برنامه مهم است، می

توان مستقیماً از کدهای زبان اسمبلی استفاده کرد که در انتهای این فصل شیوه این کار در زبان C51 را خواهیم دید.

در این فصل ابتدا به مقدمات برنامه نویسی میکروکنترلر ۸۰۵۱ با زبان C51 خواهیم پرداخت و سپس با بازنویسی تعدادی از مثالهای فصول قبل، ویژگیهای مختلف زبان C51 را بررسی خواهیم کرد.

برای مطالعه این فصل، آشنایی با ساختارهای زبان C ضروری است. برای نوشتن برنامه های C51، ترجمه، شبیه سازی و اشکالزدایی آنها از محیط مجتمع µVision استفاده می کنیم که با مشخصات آن در ضمیمه **د** آشنا خواهید شد.

برنامه نویسی به زبان C51

زبان C51 کاملاً شبیه به زبان C است و اکثر تکنیکهای زبان C در این زبان نیز قابل استفاده است. البته هیچگاه فراموش نکنید که یک برنامه C در کامپیوتر و یک برنامه C51 در میکروکنترلر اجرا می شود؛ استفاده از برخی تکنیکهای سطح بالا در برنامه- نویسی C51 ممکن است منجر به ایجاد کدهای ماشین بزرگ با زمان اجرای طولانی شود. در این موارد بهتر است برنامه خود را هرچه ساده تر بنویسید تا کامپایلر بتواند کد ماشین کوتاهتری از آن بسازد.

قالب برنامه های C51

قالب کلی یک برنامه C51 به صورت زیر است :

```
#include <نام فایل سرآیند۱.h>
#include <نام فایل سرآیند۲.h>
...
اعلان متغیرهای عمومی

        اعلان متغیرهای نوع sbit، sfr و sfr16

اعلان الگوی توابع کاربر

void main(void){

اعلان متغیرهای محلی
```

دستورات برنامه (فراخوانی توابع کتابخانه ای و توابع کاربر)
}

تعریف بدنه توابع کاربر

همانطور که مشاهده می کنید، قالب برنامه های C51 به قالب برنامه های C شباهت زیادی دارد.
فایلهای سرآیند[1] که به همراه کامپایلر C51 ارایه می شوند، شامل تعریف توابعی هستند که برای ساده تر شدن برنامه نویسی در اختیار کاربر قرار داده شده است؛ مثلاً در فایل سرآیند reg51.h که در اکثر برنامه هایی که برای ۸۰۵۱ نوشته می شود مورد استفاده قرار می گیرد، نام ثباتها، پورتها، پرچمها و دیگر اجزا داخلی ۸۰۵۱ اعلان شده است؛ با شامل[2] کردن این فایل در برنامه خود می توانید مستقیماً با این اجزا کار کنید. از دیگر فایلهای سرآیند می توان به ctype.h (حاوی توابع و ماکروهای مفید برای امتحان و تبدیل کاراکترها)، intrins.h (حاوی توابع ذاتی C51)، math.h (حاوی توابع ریاضی)، stdlib.h (حاوی توابع با کاربرد عمومی)، string.h (حاوی توابع ویژه رشته ها)، stdio.h (حاوی توابع کار با ورودی/خروجی استاندارد که در C51 پورت سریال است) و ... اشاره کرد که برای آشنایی با توابع آنها می توانید به مراجع C51 که همراه با µVision ارایه شده است، مراجعه کنید.

انواع داده ها در C51

تمام انواع داده های زبان C در برنامه های C51 نیز قابل استفاده هستند؛ اما به دلیل اینکه ۸۰۵۱ یک میکروکنترلر ۸ بیتی است و کامپایلر C51 متغیرهای برنامه C51 را باید به واحدهای ۸ بیتی تبدیل کند، برای تعریف متغیرهای برنامه خود سعی کنید از انواع داده زبان C به ترتیب زیر استفاده کنید تا حجم کد ماشین حاصل از برنامه های شما کوتاهتر شود :

انواع داده یک بایتی : (signed) char – unsigned char
انواع داده دوبایتی : unsigned int یا (signed) int – short int یا (signed) short یا enum
انواع داده ۴ بایتی : float – (signed) long – unsigned long

مثلاً برای تعریف شمارنده یک حلقه for که حداکثر آن ۲۰۰ است، به جای روش معمول در برنامه های C که از داده های int استفاده می کند، از داده نوع unsigned char استفاده کنید.

---
[1] ***Header Files***
[2] ***include***

انواع داده های ویژه C51

کامپایلر C51 علاوه بر انواع داده زبان C، از چند نوع داده جدید نیز حمایت می کند که عبارتند از :

نوع داده bit : برای تعریف متغیرهای **بیتی** در ناحیه قابل آدرس دهی بیتی حافظه RAM میکروکنترلر استفاده می شود؛ مثلاً دستور *bit mybit;* یک **بیت** در ناحیه قابل آدرس دهی بیتی حافظه RAM میکروکنترلر را با نام mybit نامگذاری و در اختیار برنامه نویس قرار می دهد.

نوع داده sfr : برای *نامگذاری* ثباتهای SFR مورد استفاده قرار می گیرد؛ مثلاً دستور *sfr P1 = 0x90;* ثبات شماره 90H (پورت ۱) را P1 نامگذاری می کند. تمام ثباتهای sfr در فایل سرآیند reg51.h تعریف شده اند و با شامل کردن این فایل در برنامه، نیازی به تعریف متغیرهای نوع sfr نیست. توجه کنید که برای اکثر میکروکنترلر های خانواده ۸۰۵۱، فایلی شامل تعاریف sfr مناسب به همراه کامپایلر C51 ارایه شده که هنگام برنامه نویسی برای هر میکروکنترلر، کافی است فایل سرآیند مخصوص آن را در در برنامه خود شامل کنید. این فایلها عبارتند از :

reg51.h	reg52.h	reg515.h	reg517.h	reg451.h	reg452.h
reg252.h	reg152.h	reg552.h	reg51g.h	reg51gb.h	reg51f.h

نوع داده sfr16 : برای تعریف SFRهای ۱۶ بیتی محصولات خانواده ۸۰۵۱ که شامل دو SFR با آدرس متوالی باشند، استفاده می شود.

نوع داده sbit : برای *نامگذاری* بیتهای ثباتهای SFR به کار می رود. دستور
sbit n = SFR^m; بیت شماره m ثبات SFR را n نامگذاری می کند؛ مثلاً برای مراجعه به بیت P1.5، باید آن را با یک نام دلخواه نامگذاری کنیم :

```
sbit mybit = P1^5;
```

و در برنامه تنها از نام mybit استفاده کنیم. مثلاً دستور *mybit = 1;* بیت P1.5 را یک می کند.
توجه کنید که متغیرهای نوع sfr و sfr16 و sbit باید قبل از تابع main تعریف شوند.

مثالهایی از برنامه های C51

اکنون تعدادی از مثالهای برنامه نویسی اسمبلی را که در فصول گذشته دیده ایم، به زبان C51 بازنویسی می کنیم. توصیه می شود این مثالها که حاوی نکاتی مهم درباره برنامه نویسی به زبان C51 هستند را با دقت مطالعه کنید.

مثال) برنامه ای بنویسید که دو عدد ۸ بیتی را بخواند، با هم جمع کند، حاصل را با عدد ۱۰ جمع کند و نتیجه را نمایش دهد.

سخت افزار این مثال در فصل ۳ بررسی شد. در اینجا نیز پورتهای P0 و P2 را ورودی و پورت P1 را خروجی در نظر می گیریم.

```
#include <reg51.h>

void main(void){

        P0 = 255;     // Define P0 as input.
        P2 = 0xFF; // Define P2 as input.
        while (1){
                P1 = P0 + P2 + 10;
        }
}
```

در مورد این برنامه چند نکته قابل توجه است :

۱) همانطور که گفته شد، فایل سرآیند reg51.h حاوی تعریف و نامگذاری ثباتهای داخلی ۸۰۵۱ است و شامل کردن آن در ابتدای برنامه ضروری است. در این فایل کلیه اجزای داخلی ۸۰۵۱ با حروف **بزرگ** انگلیسی نامگذاری شده است. از آنجا که زبان C به بزرگی و کوچکی حروف حساس است، در نوشتن برنامه ها به این نکته دقت کنید؛ مثلاً کامپایلر C51، پورت ۱ را با نام P1 می شناسد، نه به نام p1.

۲) در ابتدای برنامه پورتهای P0 و P2 به عنوان ورودی تعریف شده اند. در زبان C پیشوند **0x** قبل از یک عدد ثابت ، نشانگر بیان آن عدد در دستگاه ۱۶تایی (هگزا دسیمال) و پیشوند **0** نشان دهنده دستگاه ۸تایی (اکتال) است. در زبان C اعداد دودویی مستقیماً قابل استفاده نیستند.

۳) در برنامه های C، بعد از علامت // تا انتهای خط به عنوان توضیح در نظر گرفته می شود. از توضیحات برای واضح کردن قسمتهای مختلف برنامه استفاده می کنیم.

۴) حلقه while (1) به دلیل درست بودن دایمی شرط ادامه حلقه، بی نهایت مرتبه اجرا می شود. چنانچه بخواهیم این عملیات تنها یک مرتبه اجرا شود، برنامه باید به صورت زیر بازنویسی شود :

```
#include <reg51.h>

void main(void){

        P0 = 255; // Define P0 as input.
        P2 = 255; // Define P2 as input.
```

```
    P1 = P0 + P2 + 10;
    while (1);
}
```

پس از اجرای دستورات، برنامه در حلقه بینهایت *while (1);* به دام می افتد و باقی می ماند. این دستور مشابه دستور *SJMP $* است که برای قطع اجرای برنامه در انتهای برنامه های اسمبلی مورد استفاده قرار می گرفت.

برای ایجاد حلقه بینهایت، به جای دستور *while(1)*، می توان از دستور *for(;1;)* نیز استفاده کرد.

توجه کنید که برخی کامپایلرها حتی اگر برنامه را بدون استفاده از حلقه های بینهایت بنویسید، آن رابه نحوی ترجمه می کنند که کد ماشین به شکل خودکار به صورت حلقه بینهایت در می آید و مرتباً تکرار می شود؛ اما برای اطمینان بیشتر و نیز جلوگیری از بروز اشتباه در مقداردهی اولیه متغیرها و نیز بعضی برنامه های خاص، عادت کنید وقتی می خواهید دستورات برنامه بینهایت مرتبه اجرا شوند، خودتان آنها را در حلقه *while (1)* یا *for(;1;)* قرار دهید.

مثال) برنامه ای بنویسید که مرتباً P1 را بخواند و معکوس آن را در P2 نمایش دهد.

```
#include <reg51.h>

void main(void){

    P1 = 255; // Define P1 as input.

    for (;1;) {
        P2 = ~P1;
    }
}
```

علامت ~، باعث معکوس شدن بیتهای P1 می شود.

مثال) برنامه ای بنویسید که دو عدد m و n را بخواند و حاصل 2m-n را در نمایش دهد.

عدد m را از پورت P0 و عدد n را از پورت P2 خوانده و حاصل مورد نظر را در پورت P1 نمایش می دهیم.

```
#include <reg51.h>

#define     m     P0
#define     n     P2
#define     Out   P1
```

```
void main(void){

        m = n = 255; // Define ports as input.

        while (1) {
                Out = 2*m - n;
        }
}
```

می بینید که با استفاده از رهنمود #define می توان پورتها یا ثباتها را نامگذاری و به خوانـاتر شـدن برنامه کمک نمود. بعلاوه این کار باعث می شود بتوانید برنامـه ای را کـه بـرای میکروکنترلـر ۸۰۵۱ نوشته شده است، به راحتی روی میکروکنترلرهای دیگر که از یک زبان شبیه به C حمایت می کنند، اجرا کنید؛ برای این کار تنها کافی است تعاریف #define را مطابق با سـخت افـزار آن میکروکنترلـر تغییر دهید.

مثال) برنامه ای بنویسید که LED متصل به P1.0 را مرتباً خاموش و روشن کند.

```
#include <reg51.h>

sbit OutBit = P1^0;

void Delay(void); // Prototype of Delay() function

void main(void){

        for (;1;) {
                OutBit = 1;
                Delay();
                OutBit = 0;
                Delay();
        }
}
////////////////////////////////////////////////////////////////////////////
void Delay(void){ // Body of Delay() function
        unsigned int i,j;
        for (i = 0;i < 200;i++)
                        for (j = 0;j < 1000;j++);
}
```

در این برنامه از تابع Delay که شامل دو حلقه تودرتو است، برای ایجاد تأخیر استفاده شده است (با اسیلوسکوپ این تأخیر را اندازه بگیرید). راه دیگر ایجاد تأخیر، استفاده از تایمرهاست که در ادامه خواهیم دید.

توجه کنید هرچند می توان بدنه تابع Delay را به همراه الگوی آن قبل از تابع main آورد، برای پیروی از اصول برنامه نویسی ساخت یافته و خواناتر شدن برنامه ها، الگوی توابعی که خود تعریف می کنیم را قبل از تابع mian و بدنه آنها را بعد از تابع main می آوریم.

پرسش) نشان دهید حلقه اصلی برنامه را می توان به صورت زیر بازنویسی کرد :

```
for (;1;) {
        Delay();
        OutBit = ~OutBit;
}
```

مثال) برنامه ای بنویسید که مرتباً P1 را بخواند و تا زمانی که صفر نشده، عدد ۱۰۰ را در P2 بنویسد. با صفر شدن P1، عدد ۲۰۰ در P2 نوشته شده و برنامه به پایان می‌رسد.

```
#include <reg51.h>

void main(void){

        P1 = 255; // Define P1 as input.

        while (P1 != 0) {
                P2 = 100;
        }
        P2 = 200;
        while (1);
}
```

مثال) برنامه ای بنویسید که مرتباً عددی بین صفر تا ۹ را از P1 بخواند و آن را در نمایشگر 7-segment آند مشترک متصل به P2 نمایش دهد. اگر عدد خوانده شده بزرگتر از ۹ باشد، نمایشگر باید خاموش شود.

```
#include <reg51.h>

void main(void){

        unsigned char index,Seven_Seg_Codes[] = {0x03,0x9F, 0x25,0x0D ,0x99,
        0x49,0x41,0x1F,0x01,0x09};

        P1 = 255; // Define P1 as input.
```

```
    while (1) {
        while (P1 >= 10)
            P1 = 0xFF; // Turn off the Common Anode 7-Seg
        index = P1;
        P2 = Seven_Seg_Codes[index];
    }
}
```

کامپایلر C51 برای ذخیره متغیرهای تعریف شده در برنامه از فضای RAM میکروکنترلر استفاده می‌کند؛ یعنی آرایه Seven_Seg_Codes در فضای RAM میکروکنترلر ذخیره می شود.
چون محتویات این آرایه در طول برنامه تغییر نمی کند، بهتر است آن را در فضای ROM میکروکنترلر ذخیره کنیم. برای این کار کافی است آرایه فوق به صورت زیر تعریف شود :

code *unsigned char ,Seven_Seg_Codes[] = {0x03, 0x9F, 0x25, 0x0D, 0x99, 0x49, 0x41,0x1F,0x01,0x09};*

چون متغیر index در طول برنامه تغییر می کند، حتماً باید با دستورتعریف متغیر *unsigned char index;* در RAM میکروکنترلر ذخیره شود.
توجه کنید که در C51 بر خلاف C، نمی توان متغیرها را در هر نقطه دلخواه تعریف کرد؛ بلکه متغیرها تنها در ابتدای تابع یا حلقه می توانند تعریف شوند.

مثال) برنامه ای بنویسید که زوج پورت P0-P2 را با عدد ۱۶ بیتی 127A H جمع کند و حاصل را در زوج پورت P1-P3 را نمایش دهد.

```
#include <reg51.h>

void main(void){
    P0 = P2 = 255; // Define as input
    for (;1;){
        unsigned char carry = 0;
        if (P2 + 0x7A > 255)
            carry = 1;
        P3 = P2 + 0x7A;
        P1 = P0 + 0x12 + carry;
    }
}
```

پرسش) با توجه به اینکه $۲^{۸}$ = ۲۵۶ نشان دهید برنامه زیر نیز همین کار را انجام می دهد.

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

```
void main(void){

	P0 = P2 = 255; // Define as input
	for (;1;){
		unsigned int P0_P2_Value = P0 * 256 + P2;
		unsigned int Result = P0_P2_Value + 0x127A;
		P3 = Result % 256; // Low
		P1 = Result / 256; // High
	}
}
```

مثال) برنامه ای بنویسید که هرگاه کلید فشاری متصل به P0.0 فشرده ("صفر") شود، محتویات P1 را در P2 بنویسد.

```
#include <reg51.h>

void Debounce(void);

sbit Enter = P0^0;

void main(void){

	Enter = 1; // Define as input

	while (1){
		while (Enter == 1); // Wait until push button is pressed.
		P2 = P1;
		Debounce();
		while (Enter ==0); // Wait until push button is released.
		Debounce();
	}
}
////////////////////////////////////////////////////////////////////////////
void Debounce(void){
	unsigned char i,j;
	for (i = 0;i < 50;i++)
		for (j = 0;j < 50;j++);
}
```

مثال) برنامه ای بنویسید که حالت دو درب را کنترل کند؛ اگر یکی از آنها باز شود یک چراغ و اگر هر دو باز شود یک آژیر روشن شود. مدار تشخیص باز و بسته بودن درب در حالت باز بودن درب، سیگنال منطقی "یک" تولید می کند.

```
#include <reg51.h>

sbit Door1 = P1^0;
sbit Door2 = P1^1;
sbit Lamp  = P2^0;
sbit Alarm = P2^1;

void main(void){

        Door1 = Door2 = 1; // Define as input

        while (1){
                if (Door1 & Door2 == 1){
                        Alarm = 1;
                        Lamp = 0;
                }
                else if (Door1 | Door2 ==1 ){
                        Alarm = 0;
                        Lamp = 1;
                }
                else{
                        Alarm = 0;
                        Lamp = 0;
                }
        }
}
```

علامت & نشان دهنده AND بیتی و علامت ^ نـشانگر XOR بیتـی اسـت. دو عملگـر بیتـی دیگـر عبارتند از ~ (NOT بیتی) و | (OR بیتی).

مثال) برنامه ای بنویسید که P1 را بخواند و بـا اسـتفاده از ماسـکهای منطقـی بـدون تغییـر P1.0 و P1.1،

الف) P1.2 و P1.3 را یک کند. ب) P1.4 و P1.5 را صفر کند.

ج) P1.6 و P1.7 را معکوس کند و حاصل را در P2 بنویسد.

```
#include <reg51.h>

void main(void){

        P1 = 255; // Define as input

        for (;1;){
                unsigned char n = P1;
                n = n | 0x0C; // or n |= 0x0C
```

```
        n = n & 0xCF; // or n &= 0xCF
        n = n ^ 0xC0; // or n ^= 0xC0
        P2 = n;
    }
}
```

مثال) برنامه ای بنویسید که P0 و P1 را بخواند و اگر P1 > P0، P2.0 را روشن و در غیر این صورت خاموش کند.

```
#include <reg51.h>

sbit MyBit = P2^0;

void main(void){

    P1 = P0 = 255; // Define as input

    while (1){
        if (P1 > P0)
            MyBit = 1;
        else
            MyBit = 0;
    }
}
```

مثال) برنامه ای بنویسید که P1 را بخواند و
اگر P1 = 0 عدد 11111110، اگر P1 = 1 عدد 11111101، اگر P1 = 2 عدد 11111011، اگر P1 = 3 عدد 11110111، اگر P1 = 4 عدد 11101111، اگر P1 = 5 عدد 11011111، اگر P1 = 6 عدد 10111111، اگر P1 = 7 عدد 01111111 و در غیر این صورت عدد 10101010 را به P2 ارسال کند.

```
#include <reg51.h>

void main(void){

    P1 = 255; // Define as input

    for (;1;){
        switch (P1){
            case 0 : P2 = 0xFE;
                     break;
            case 1 : P2 = 0xFD;
                     break;
            case 2 : P2 = 0xFB;
```

```
                        break;
                case 3 : P2 = 0xF7;
                        break;
                case 4 : P2 = 0xEF;
                        break;
                case 5 : P2 = 0xDF;
                        break;
                case 6 : P2 = 0xBF;
                        break;
                case 7 : P2 = 0x7F;
                        break;
                default: P2 = 0xAA;
                        break;
            }
        }
}
```

مثال) برنامه ای بنویسید که رقص نوری با یک LED چرخان روی P1 بسازد.

```
#include <reg51.h>

void Delay(void); // Prototype of Delay() function

void main(void){

    unsigned char i;

    while (1) {
        P1 = 1;
        for (i = 0;i < 8;i++){
            Delay();
            P1 = P1 << 1;
        }
    }
}
//////////////////////////////////////////////////////////////////
void Delay(void){ // Body of Delay() function
    unsigned int i,j;
    for (i = 0;i < 200;i++)
            for (j = 0;j < 1000;j++);
}
```

دستور k = m << n، عدد m را n واحد به سمت چپ شیفت داده و حاصل را در k ذخیره مـی کنـد. برای شیفت به راست از عملگر >> استفاده می کنیم.

مثال) برنامه ای بنویسید که بیتهای زوج و فرد P1 را به ترتیب روشن و خاموش کند. تـأخیر مـابین روشن و خاموش شدن بیتها با عددی که از P2 خوانده می شود تنظیم می گردد.

تاکنون فقط از توابع بدون پارامتر استفاده کرده ایم. این مثال از یک تابع پارامتردار استفاده می کند.

```
#include <reg51.h>

void Delay(unsigned char time); // 'time' is parameter of Delay function.

void main(void){

	P2 = 255; // Define as input.

	while (1) {
		unsigned char t = P2;
		P1 = 0x55;
		Delay(t);
		P1 = 0xAA;
		Delay(t);
	}
}
//////////////////////////////////////////////////////////////////////////
void Delay(unsigned char time){
	unsigned int i,j;
	for (i = 0;i < time;i++)
			for (j = 0;j < 1000;j++);
}
```

مثال) برنامه ای بنویسید که رشته *This is a message* را به P1 ارسال کند.

```
#include <reg51.h>

void Delay(void);

void main(void){

	code unsigned char Table[18] = "This is a message";
	for (;1;) {
		unsigned char i;
		for (i = 0;i < 18;i++){
			P1 = Table[i];
			Delay();
		}
	}
}
```

```
////////////////////////////////////////////////////////////////////////////////
void Delay(void){
        unsigned int i,j;
        for (i = 0;i < 200;i++)
                for (j = 0;j < 1000;j++);
}
```

مثال) برنامه ای بنویسید که P1 را بخواند و مجموع ارقام دهدهی آن را در P2 بنویسد.

```
#include <reg51.h>

void main(void){

        P1 = 255; // Define as input

        while (1){
                unsigned char n = P1,sum = 0;
                sum = sum + n % 10; // or sum += n % 10;
                n = n / 10;
                sum = sum + n % 10;
                sum = sum + n / 10;
                P2 = sum;
        }
}
```

استفاده از وقفه ها در C51

برای استفاده از وقفه ها، کامپایلر C51 به هرکدام از منابع وقفه ۸۰۵۱ یک شماره به صورت زیر اختصاص داده است :

INT0 ➔ 0	T0 ➔ 1	INT1 ➔ 2
T1 ➔ 3	Serial ➔ 4	T2 ➔ 5

توابع ISR وقفه ها، مانند توابع معمولی C51، تعریف می شوند؛ فقط بعد از الگوی تابع، عبارت interrupt m ذکر می شود که m شماره وقفه آن ISR است. معمولاً بدنه ISR نیز قبل از تابع main تعریف می شود.

مثال) برنامه ای بنویسید که LED متصل به P1.0 را مرتباً خاموش و روشن کند. با فشردن یک کلید فشاری، LED باید دو برابر زمان عادی روشن مانده و سپس به چشمک زدن خود ادامه دهد. برای کنترل فشرده شدن کلید، از وقفه سخت افزاری حساس به لبه INT0 استفاده کنید.

```
#include <reg51.h>

sbit out_bit = P1^0;
```

```
void Delay(void);

void ISR_INT0(void) interrupt 0{
        out_bit = 1;
        Delay();
        Delay();
        out_bit = 0;
}

void main(void){

        IE = 0x81; // Enable INT0 interrupt
        IT0 = 1; // Make INT0 Edge-Triggerd

        while (1) {
                out_bit = ~out_bit;
                Delay();
        }
}
//////////////////////////////////////////////////////////////////////////////
void Delay(void){
        unsigned int i,j;
        for (i = 0;i < 200;i++)
                        for (j = 0;j < 1000;j++);
}
```

توجه کنید اگر پس از الگوی تابع ISR (یا هر تابع دیگری) از عبارت using n
(n = 0,1,2,3) استفاده کنیم، در ISR از بانک رجیستری شماره n استفاده می شود که باعث کاهش سربار ناشی از ذخیره R0 تا R7 در پشته به هنگام فراخوانی وقفه
می شود. مثلاً برای اینکه ISR_INT0 از بانک شماره ۲ استفاده کند، باید آن را به این صورت اعـلان کنیم :

void IST_INT0(void) interrupt 0 using 2

استفاده از تایمرها در C51

مثال) برنامه ای بنویسید که با استفاده از تـایمر مـد یـک (۱۶ بیتـی)، کریـستال 12 MHz و روش سرکشی، P1.0 را هر ۵۰ میلی ثانیه یک بار خاموش و روشن کند.

بر اساس آنچه در فصل ۹ گفته شد، با استفاده از کریستال 12 MHz داریم :

زمان مورد اندازه گیری بر حسب میکروثانیه - ۶۵۵۳۶ = مقدار اولیه تایمر

تابع Delay که پارامتر آن مقدار تأخیر مورد نیاز بر حسب میکروثانیه است، بر اساس این نکته نوشته شده است.

```
#include <reg51.h>

void Delay(unsigned int time); // 'time' is parameter of Delay function.

void main(void){

        TMOD = 0x01; // Timer 0 in Mode 1 (16 bit)

        P1 = 0x55;
        while (1) {
                Delay(50000);
                P1 = ~P1;
        }
}
////////////////////////////////////////////////////////////////////////////
void Delay(unsigned int time){ // 'time' is Delay in micro-seconds.
        unsigned int init_value = 65536 - time;
        TH0 = init_value / 256;
        TL0 = init_value % 256;
        TR0 = 1;
        while (TF0 == 0) // wain until T0 overflows.
        TR0 = 0;
        TF0 = 0;
}
```

به روش مقداردهی TH0 و TL0 دقت کنید.

مثال) برنامه ای بنویسید که دوبرابر P1 را در P2 بنویسد و بعلاوه با استفاده از تایمر ۱ یک در مد دو (۸ بیتی) و کریستال 12 MHz و روش وقفه، یک موج مربعی با فرکانس 10 KHz روی پایه P0.0 تولید کند.

دوره تناوب موج مربعی با فرکانس 10 KHz برابر است با :

$T = 1/f = 1/10000 = 100\ \mu s \rightarrow T/2 = 50\ \mu s$

مقدار اولیه تایمر ۸ بیتی برای ایجاد تأخیر ۵۰ میکروثانیه برابر ۲۰۶ است.

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

```
sbit mybit = P0^0;

void Init_T0(void);

void ISR_T0(void) interrupt 1{
        mybit = ~mybit;
}

void main(void){

        P1 = 255; // Define as input
        Init_T0();

        for (;1;) {
                P2 = 2*P1;
        }
}

void Init_T0(void){

        TMOD = 0x20;
        TH1 = 206;
        IE = 0x88; // Enable T0 interrupt
        TR1 = 1;
}
```

توجه کنید که به دلیل استفاده از تایمرها به شیوه وقفه، نیازی به پـاک کـردن دسـتی TF0 در ISR نیست. به علاوه به دلیل خاصیت بارگذاری مجدد خودکار تایمر در مد ۲ (۸ بیتی)، نیـازی نیـست در ISR تایمر را مجدداً مقداردهی کنیم؛ در حالی که در اگر از تایمر مد ۱ (۱۶ بیتـی) بـه شـیوه وقفـه استفاده کنیم، باید در ISR تایمر را خاموش، مقداردهی اولیه و مجدداً روشن کنیم.

ارتباط سریال در C51

مثال) برنامه ای بنویسید که با استفاده از شیوه سرکشی، عددی بین صفر تا ۹ را با نـرخ داده ۹۶۰۰ بیت در ثانیه از پورت سریال میکروکنترلر دریافت و روی نمایشگر
7-seg آند مشترک متصل به P1 نمایش دهد. اگر عدد بزرگتر از ۹ باشد، نمایشگر باید خاموش شود.

```
#include <reg51.h>

void main(void){
        code unsigned char Seven_Seg_Codes[] = {0x03,0x9F,0x25,0x0D,0x99,
        0x49,0x41,0x1F,0x01,0x09};
        unsigned char index;

        SCON = 0x50;
```

```
	TMOD = 0x20;
	TH1 = 0xFD ; // 9600 bps
	TR1 = 1;

	while (1) {

		while (RI == 0);

		index = SBUF - 48; // Convert ascii code to integer.
		if (index > 9)
			P1 = 0xFF; // Turn off Common Anode 7-seg.
		else
			P1 = Seven_Seg_Codes[index];

		RI = 0;
	}
}
```

مثال) برنامه ای بنویسید که با استفاده از شیوه سرکشی، مرتباً حروف کوچک انگلیسی را با نرخ ۴۸۰۰ بیت در ثانیه به پورت سریال ارسال کند.

```
#include <reg51.h>

void Init_Serial_Port(void);
void Send(unsigned char C);

void main(void){

	Init_Serial_Port();

	while (1) {
		unsigned char ch;
		for (ch = 'a';ch <= 'z';ch++)
			Send(ch);
	}
}
////////////////////////////////////////////////////////////////////////////////
void Init_Serial_Port(void){

	SCON = 0x50;
	TMOD = 0x20;
	TH1 = 0xFA ; // 4800 bps
	TR1 = 1;
}
////////////////////////////////////////////////////////////////////////////////
void Send(unsigned char C){
```

```
        SBUF = C;
        while (TI == 0);
        TI = 0;
}
```

مثال) برنامه ای به شیوه وقفه بنویسید که چنانچه عدد ۱ را از پورت سریال دریافت کند، دو برابر P1 و چنانچه عدد ۲ دریافت شود، دو برابر P2 را روی P0 نمایش دهد. نرخ داده ۹۶۰۰ بیت در ثانیه است.

```
#include <reg51.h>

unsigned char use_P1; // Golabal Variable

void Init_Serial_Port(void);

void Receive(void) interrupt 4 {
        unsigned char m = SBUF;
        RI = 0;
        if (m == '1')
                use_P1 = 1;
        else if (m == '2')
                use_P1 = 0;
}

void main(void){

        IE = 0x90; // Enable Serial Interrupt
        Init_Serial_Port();
        use_P1 = 1;

        while (1) {
                if (use_P1)
                        P0 = 2*P1;
                else
                        P0 = 2*P2;
        }
}
void Init_Serial_Port(void){

        SCON = 0x50;
        TMOD = 0x20;
        TH1 = 0xFD ; // 9600 bps
        TR1 = 1;
}
```

استفاده از دستورات اسمبلی در برنامه های C51

گاهی برای کاهش طول کد ماشین یا زمان اجرای برنامه، در نقاط بحرانی یک نرم افزار سطح بالا از کدهای اسمبلی استفاده می کنیم.

برای استفاده از کدهای اسمبلی در یک برنامه C51، کافی است دو خط زیر را به ابتدای برنامه خود اضافه کنید :

```
#pragma SRC
#pragma SMALL
```

در طول برنامه می توانید بین دو رهنمود *#pragma ASM* و *#pragma ENDASM* از دستورات اسمبلی استفاده کنید.

مثال) برنامه ای بنویسید که LED متصل به P1.0 را مرتباً روشن و خاموش کند؛ برای ایجاد تأخیر از دستورات اسمبلی استفاده کنید.

```
#pragma SRC
#pragma SMALL

#include <reg51.h>

sbit out_bit = P1^0;

void Delay(void);

void main(void){

	while (1) {
		out_bit = ~out_bit;
		Delay();
	}
}
////////////////////////////////////////////////////////////////////////////
void Delay(void){
	#pragma ASM

		MOV R0,#255
		AGAIN:
			NOP
			NOP
			NOP
		DJNZ R0,AGAIN

	#pragma ENDASM
```

```
}
```

در محیط µVision یک فایل C51 حاوی دستورات اسمبلی نمی تواند مستقیماً به فایل hex تبدیل شود؛ در پنجره project روی نام فایل C51 کلیک راست و گزینه properties را انتخاب کنید و گزینه های Generate Assembler SRC File و Assemble SRC File را علامت بزنید. با این کار پس از انجام عمل Build، فایلی با نام فایل C51 مذکور و با پسوند SRC تولید می شود که حاوی کدهای اسمبلی ترجمه شده فایل C51 است. این فایل را جداگانه (در یک project دیگر) با استفاده از اسمبلر ۸۰۵۱ به فایل hex تبدیل کنید.

تعیین محل ذخیره سازی متغیرها در حافظه میکروکنترلر

با استفاده از چند کلمه کلیدی C51 که قبل از اعلان نوع ماغیر به کار می روند، می توان ناحیه ذخیره متغیر در حافظه ۸۰۵۱ را مشخص کرد. این کلمات کلیدی به قرار زیر هستند :

code : نشان دهنده این است که متغیر مذکور در حافظه ROM میکروکنترلر ذخیره می شود. بدیهی است که مقدار این متغیر در برنامه نباید تغییر کند. مثلاً اعلان متغیر *code char Table[5] = {-2,-1,0,1,2};* یک آرایه ۵ عضوی به نام Table را در حافظه ROM میکروکنترلر ذخیره می کند.

data : نشان دهنده آن است که مکان متغیر در RAM داخلی و دسترسی به آن عادی است؛ مثلاً دستورات

```
data char i;
i = 2;
```

در زبان اسمبلی معادل *MOV i,#2* خواهد بود. متغیرهای C51 به صورت پیش فرض از نوع data هستند.

idata : نشان دهنده آن است که مکان متغیر در RAM داخلی و دسترسی به آن به صورت غیرمستقیم است؛ مثلاً دستورات

```
idata char i;
i = 2;
```

در زبان اسمبلی معادل دستورات زیرخواهد بود :

```
MOV R0,#i
MOV @R0,#2
```

bdata : نشان دهنده این است که متغیر فوق در ناحیه قابل آدرس دهی بیتی RAM میکروکنترلر ۸۰۵۱ ذخیره شده و هم به صورت بیتی و هم به صورت بایتی قابل مراجعه است. مثلاً اعلان متغیر *bdata unsigned char m;*، متغیر m را در ناحیه قابل آدرس دهی بیتی RAM میکروکنترلر ۸۰۵۱ تعریف می کند. مراجعه به m معادل مراجعه به بایت m است. برای مراجعه به بیتهای m باید از نوع داده sbit استفاده کنید. مثلاً اعلان متغیر *sbit mybit = m^3;* برای مراجعه به بیت شماره ۳ بایت m با نام mybit به کار می رود.

خلاصه

در این فصل با اصول برنامه نویسی میکروکنترلر ۸۰۵۱ به زبان سطح بالای C51 آشنا شدیم و جنبه های مختلف برنامه نویسی آن از جمله برنامه نویسی ورودی/خروجی، دستورات ریاضی، عملگرهای منطقی و نیز استفاده از وقفه ها، تایمرها، امکانات ارتباط سریال و نیز کدهای اسمبلی را در طی چند مثال بررسی کردیم. برای آشنایی با نحوه استفاده از کامپایلر C51 در محیط مجتمع µVision به ضمیمه **د** مراجعه کنید.

پرسشهای دوره ای

۱) برنامه ای بنویسید که مقادیر صفر تا ۲۵۵ را مرتباً به P1 ارسال کند.

۲) برنامه ای بنویسید که یک رمز ۸ بیتی را بخواند و اگر برابر 10111001 بود، یک درب را باز کند و برنامه به پایان برسد. از یک کلید فشاری برای ورود رمز استفاده کنید. کاربر تا ۳ مرتبه فرصت دارد رمز ورودی را به درستی وارد کند، وگرنه یک آژیر به صدا در خواهد آمد.

۳) برنامه ای بنویسید که P0 را بخواند و اگر بزرگتر از ۱۰۰ بود آن را به P1 و در غیر این صورت به P2 ارسال کند.

۴) الف) نشان دهید که برنامه زیر، محتویات P1 را به صورت سریال با شروع از LSB روی P2.0 ارسال می کند :

```
#include <reg51.h>

bdata unsigned char MyByte;
sbit LSB_bit = MyByte^0;
sbit out_bit = P2^0;

void Delay(void){
        unsigned char i,j;
        for (i = 0;i < 200;i++)
                for (j = 0;j < 1000;j++);
```

```
}

void main(void){

        P1 = 255; // Define as input
        while (1){
                unsigned char index;
                MyByte = P1;
                for (index = 0; index < 8;index++){
                        out_bit = LSB_bit;
                        Delay();
                        MyByte = MyByte >> 1;
                }
        }
}
```

ب) اگر در این برنامه بخواهیم با شروع از MSB ارسال کنیم، چه تغییراتی باید انجام شود؟

ج) با استفاده از برنامه بالا، برنامه ای بنویسید که P1 را بخواند و یک شیفت چرخشی به سمت راست داده و نتیجه را در P2 بنویسد. توجه کنید که عملگر << مربوط به شیفت عادی است؛ یعنی بیت ورودی همیشه صفر است.

د) برنامه ای بنویسید که بایتهای سریال ارسال شده با برنامه قسمت **الف** را روی پین P0.0 میکروکنترلر مقصد دریافت کند.

۵) نشان دهید برنامه زیر، توازن فرد P1 را روی P2.0 نمایش می دهد :

```
#include <reg51.h>

bdata unsigned char MyByte;
sbit LSB_bit = MyByte^0;
sbit out_bit = P2^0;

bit Odd_Parity(unsigned char C){

        bit parity_bit;
        unsigned char i;

        MyByte = C;
        parity_bit = 0;
        for (i = 0;i < 8;i++){
                parity_bit = parity_bit ^ LSB_bit;
                MyByte = MyByte >> 1;
        }
        return parity_bit;
```

```
}

void main(void){

        P1 = 255; // Define as input
        while (1){
                out_bit = Odd_Parity(P1);
        }
}
```

ب) نشان دهید اگر در تابع توازن مقدار اولیه parity_bit صفر باشد، توازن زوج تولید می شود.

۶) با استفاده از وقفه های تایمر، برنامه ای بنویسید که دو موج مربعی یکی با فرکانس ۱ هرتز و دیگری با فرکانس ۵۰ هرتز روی دو پایه میکروکنترلر تولید کند. با اسیلوسکوپ فرکانس موجهای تولید شده را اندازه بگیرید. اختلاف مقدار آنها با مقدار مورد نظر ما ناشی از چیست؟

۷) برنامه ای بنویسید که هرگاه یک کلید فشاری فشرده شود، محتویات P1 را با نرخ ۹۶۰۰ بیت در ثانیه برای پورت سریال ارسال کند.

۸) برنامه ای بنویسید که بایت دریافتی از پورت سریال را دوباره به پورت سریال ارسال کند و در عین حال اگر بایت مذکور یکی از حروف بزرگ انگلیسی (کد اسکی بین ۶۵ تا ۹۰) بود، آن را روی P1 و اگر یکی از حروف کوچک انگلیسی (کد اسکی بین ۹۷ تا ۱۲۲) بود، آن را روی P2 قرار دهد.