پودمان ۲

برنامه‌نویسی به زبان C

درکتاب مونتاژ و دمونتاژ SMD با میکرو کنترلر AVR و پروژه‌های ساده الکترونیکی آشنا شدید. همچنین مثال‌ها، تمرین‌ها و برنامه‌های آماده را که از طریق نرم‌افزار کد ویژن نوشته شده بود، در میکروکنترلر بارگذاری و آزمایش کردید. در این پودمان با چگونگی شکل‌گیری و نوشتن برنامه‌ها به زبان C آشنا می‌شوید تا بتوانید توانایی نوشتن برنامه‌های پیچیده‌تر را کسب کنید. در فرایند آموزش این پودمان، علائم الگوریتم، رسم فلوچارت و تبدیل آن به برنامه را فرا خواهید گرفت. یکی دیگر از اهداف آموزشی، آشنایی با قطعات و تجهیزات ورودی و خروجی است که به‌وسیله آن می‌توانید داده‌های دیجیتال و آنالوگ را به میکروکنترلر وارد کنید یا از آن دریافت نمایید. فرایند پردازش داده‌ها و اطلاعات توسط میکروکنترلر از مواردی است که هر متخصص الکترونیک باید با آن آشنایی داشته باشد که در ادامه آموزش به این موارد نیز می‌پردازیم.

واحد یادگیری ۳

کسب شایستگی در طراحی الگوریتم (فلوچارت) مدارپروژه ساده الکترونیکی

آیا تا به‌حال فکر کرده‌اید:

- با چه زبان‌ها و کامپایلرهایی می‌توان اقدام به برنامه‌نویسی میکروکنترلرها نمود؟
- اولین زبان برنامه‌نویسی اختراع شده چه بوده است؟
- فلوچارت یا روند نما چیست و چه کاربردی دارد؟
- در فلوچارت هر یک از علائم چه معنایی دارد و با چه نرم‌افزاری رسم می‌شود؟
- از توابع و عملگرها در زبان C چگونه استفاده می‌شود؟
- به‌کارگیری توابع و عملگرها چگونه باعث افزایش سرعت برنامه‌نویسی و تفهیم بهتر آن می‌شود؟
- اتصال و راه‌اندازی صفحه کلید به میکروکنترلر چگونه صورت می‌گیرد؟
- راه‌اندازی سنسورهایی مانند سنجش دما و شدت نور به کمک واحد ADC در میکروکنترلرها چگونه انجام می‌شود؟

از زمان پیدایش و توسعه زبان برنامه‌نویسی C تا به امروز، این زبان همواره در حال توسعه بوده و در سیستم‌های نرم‌افزاری و سخت‌افزاری به‌کار می‌رود. با توجه به قدیمی بودن این زبان هیچ‌گاه دستور زبان به‌کار رفته در آن منسوخ نشده است. امروزه زبان‌های بسیار زیادی ایجاد شده‌اند که شباهت به C دارند و به شکل گسترده‌ای توسعه یافته‌اند. نفوذ زبان C در صنعت به قدری است که توسعه‌دهندگان نرم‌افزارهای میکروکنترلر، در حد گسترده از آن استفاده می‌کنند. امروزه زبان‌های برنامه‌نویسی متنوعی برای میکروکنترلرهایی مانند AVR، PIC و سری ARM با استفاده از کامپایلرهای مبتنی‌بر زبان C مانند Codeviosion، PICC، keil و IAR طراحی شده‌اند که بخش عظیمی از صنایع دنیا را پوشش می‌دهند.
در این واحد یادگیری ابتدا علائم و چگونگی ترسیم الگوریتم یا روند برای حل یک مسئله را مطرح می‌کنیم. سپس با مفاهیم ساختار کلی برنامه‌نویسی به زبان C آشنا خواهیم شد. همچنین به‌کارگیری برنامه C در برنامه‌نویسی میکروکنترلرهای AVR و توسعه سخت‌افزار مبتنی بر آن را مورد بحث قرار خواهیم داد.

استاندارد عملکرد

ترسیم فلوچارت برنامه‌های مختلف و پیاده‌سازی آن روی میکروکنترلر

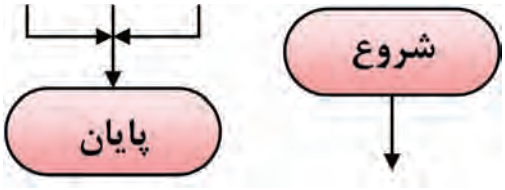
۱ـ۳ـ طراحی الگوریتم (روند) برنامه

عناصر اصلی برای حل یک مسئله در کامپیوتر، ورودی‌ها و خروجی‌ها هستند. برای مثال، برای جمع دو عدد a و b ورودی‌ها a و b هستند. این برنامه یک خروجی دارد که حاصل جمع این دو عدد است. بنابراین نوشتن این برنامه بسیار ساده است، اما برای نوشتن برخی از برنامه‌ها به تفکر بیشتری نیاز داریم. در چنین مرحله‌ای لازم است ابتدا گام‌های برنامه مورد نظر را به‌دست آورید تا بتوانید برنامه مورد نظر را پیاده‌سازی کنید. غالباً اجرای این گام‌ها چندین ساعت طول می‌کشد. همچنین نیاز به محاسبات ریاضی فراوانی دارد. برای رسیدن به این هدف، ترسیم فلوچارت برای مسئله مورد نظر، امری کاربردی است.

به مجموعه‌ای از تصاویر و نمادها که الگوریتم (روند) برنامه را به‌صورت تصویری و نموداری نشان می‌دهد فلوچارت یا روندنما می‌گویند. رسم فلوچارت تا حد زیادی درک مسئله را برای ذهن ما ساده‌تر و قابل فهم‌تر می‌کند.

■ **علائم استاندارد برای طراحی فلوچارت:** برای رسم یک فلوچارت لازم است ابتدا با نمادهای آن آشنا شویم. نمادهایی که در روند نما به‌کار می‌رود شامل بیضی، مستطیل، متوازی‌الاضلاع و لوزی است که هر یک مفهومی به شرح زیر دارند.

■ **نماد شروع و پایان:** برای شروع و پایان هر الگوریتم در فلوچارت از نماد بیضی مطابق شکل ۱ـ۳ استفاده می‌شود. معمولاً از نماد شروع یک پیکان خارج و به نماد پایان یک یا چند پیکان وارد می‌شود.

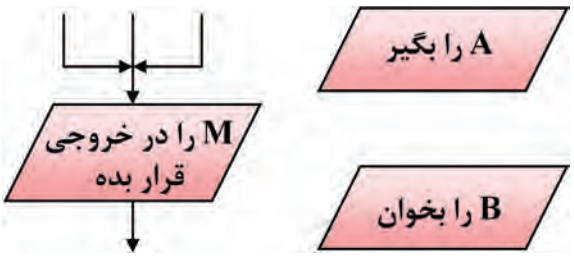


شکل ۱ـ۳

■ **علامت پردازش یا اجرای عملیات:** برای نمایش انجام یک عملیات مانند عملیات محاسباتی و پردازشی از علامت مستطیل مطابق شکل ۲ـ۳ استفاده می‌شود، مثلاً شکل ۲ـ۳ می‌گوید در A مقدار ۲، در B مقدار ۴ و حاصل‌ضرب محتوای A در محتوای B در C قرار می‌گیرد. به این نماد یک یا چند پیکان وارد شده و یک پیکان از آن خارج می‌شود.

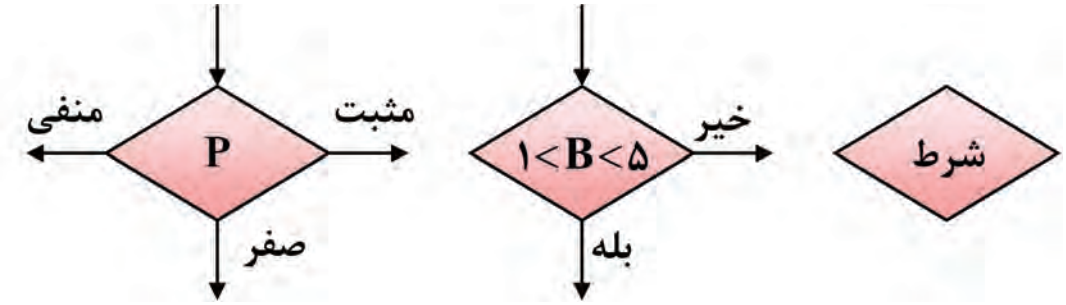
شکل ۲ـ۳

■ **نماد ورودی و خروجی:** از علامت متوازی‌الاضلاع برای نمایش عملیات مربوط به مقادیر ورودی و خروجی داده‌ها استفاده می‌شود. در شکل ۳ـ۳ این نماد و مثال‌هایی از آن را مشاهده می‌کنید.



شکل ۳ـ۳

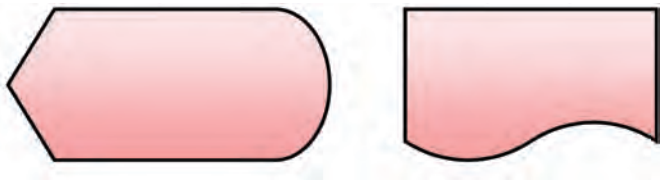
■ **علامت شرط یا if:** شرط یا شرط‌ها را طبق شکل ۴ـ۳ داخل لوزی قرار می‌دهیم. در عبارات شرطی، انجام عملیات منوط به برقرار بودن شرط یا شروط مورد نظر است. طبق شکل ۴ـ۳ به این نماد یک پیکان وارد و متناسب با نیاز دو یا سه پیکان خارج می‌شود. مثلاً در شکل پ ـ۴ـ۳ اگر P، مثبت، منفی یا صفر باشد در هر حالت عملیات مشخصی انجام می‌گیرد.



الف ب) دو فلش به عنوان خروجی پ) سه فلش به عنوان خروجی

شکل ۴ـ۳

علامت چاپ: برای چاپ مقدار مورد نظر روی کاغذ یا صفحه نمایش، می‌توان از دو علامت شکل ۵ـ۳ استفاده کرد. مقدار مورد نظر برای چاپ، در داخل نماد نوشته می‌شود.



برای چاپ روی کاغذ چاپ روی صفحه نمایش

شکل ۵ـ۳

فعالیت

■ تحقیق کنید چه کامپایلرهای دیگری برای AVR به زبان C و پاسکال (PASCAL) وجود دارد.
■ با مراجعه به رسانه‌های مختلف انواع نرم‌افزارهای رسم فلوچارت را پیدا کنید و با پرسش از اهل فن، نرم‌افزار مناسب را انتخاب نمایید و در ساعت‌های غیر درسی با آن کار کنید.

الگوی پرسش:

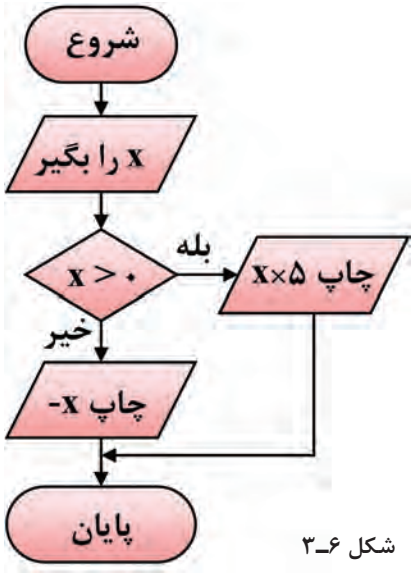
۱ـ به مجموعه‌ای از تصاویر و نمادها که الگوریتم (روند) یک برنامه را به‌صورت تصویری نشان می‌دهد ..... یا ..... می‌گویند.
۲ـ از نماد ..... برای شروع و پایان الگوریتم استفاده می‌شود.
۳ـ از نماد ..... برای پردازش استفاده می‌شود.
۴ـ از نماد لوزی تعداد ..... پیکان داخل و ..... پیکان خارج می‌شود.

مثال ۱: فلوچارت برنامه‌ای را ترسیم کنید که عدد x را به عنوان ورودی دریافت، سپس اگر عدد x مثبت بود آن را در ۵ ضرب کرده و چاپ کند در غیراین صورت آن را در یک منفی ضرب کرده و سپس آن را چاپ نماید، شکل ۶ـ۳.

مثال ۲: فلوچارتی رسم کنید که دو عدد a و b را بگیرد و حاصل جمع آنها را چاپ کند. در شکل ۷ـ۳ الگوریتم و فلوچارت برنامه ترسیم شده است.

پاسخ مثال ۱

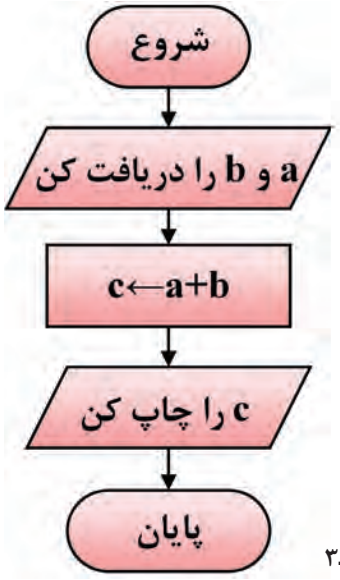
الگوریتم
۱ـ شروع
۲ـ x را بگیر
۳ـ اگر x > ۰ ، آنگاه x × ۵ را چاپ کن
و برو به پایان
۴ـ x - را چاپ کن
۵ـ پایان



شکل ۶ـ۳

پاسخ مثال ۲

الگوریتم
۱ـ شروع
۲ـ a و b را دریافت کن
۳ـ c ⟵ a + b
۴ـ c چاپ کن
۵ـ پایان



شکل ۷ـ۳

مثال ۳: فلوچارتی رسم کنید که عدد حقیقی a را دریافت و مقدار تابع چندضابطه‌ای داخل کادر را محاسبه کند و در نهایت نمایش دهد، شکل ۸ـ۳.

الگوریتم

۱ـ شروع

۲ـ a را دریافت کن

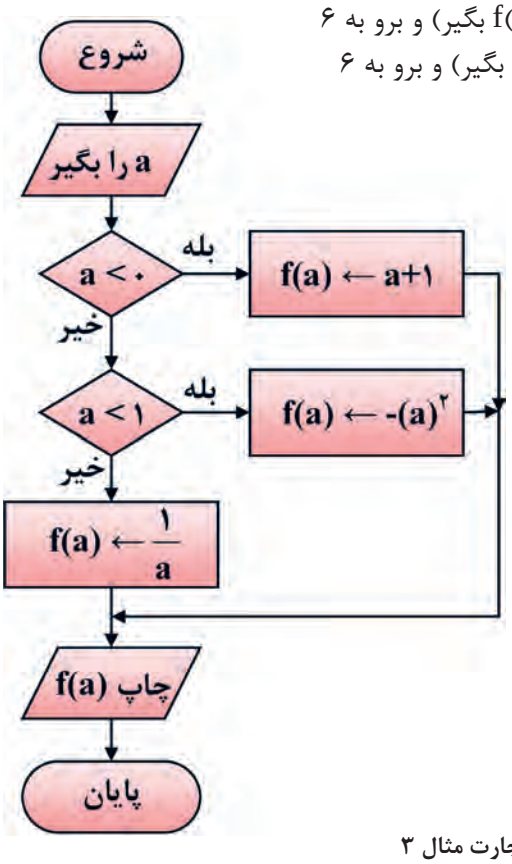
۳ـ اگر $a < ۰$ سپس $f(a) \leftarrow a + ۱$، ($a + ۱$ را مساوی $f(a)$ بگیر) و برو به ۶

۴ـ اگر $a < ۱$ سپس $f(a) \leftarrow -(a^۲)$ ، ($-a^۲$ را مساوی $f(a)$ بگیر) و برو به ۶

۵ـ $f(a) \leftarrow \frac{۱}{a}$، ($\frac{۱}{a}$ را مساوی $f(a)$ بگیر)

۶ـ چاپ $f(a)$

۷ـ پایان



$$f(a)=\begin{cases} a+1 & a<0 \\ -a^2 & 0\le a<1 \\ \frac{1}{a} & 1\le a \end{cases}$$

شکل ۸ـ۳ـ فلوچارت مثال ۳

فعالیت

■ برای روند (الگوریتم) زیر یک فلوچارت رسم کنید.

۱ـ شروع

۲ـ a و b را دریافت کن

۳ـ d=a.b (d را مساوی a ضربدر b بگیر)

۴ـ d را چاپ کن

۵ـ پایان

■ برای رسم فلوچارت‌های ذکر شده از نرم‌افزاری که در مراحل قبل انتخاب کرده‌اید استفاده کنید. سپس فلوچارت رسم شده با نرم‌افزار را با فلوچارت‌های داده شده مقایسه کنید، در صورتی که تعارضی دارد درباره آن بحث کنید.

پژوهش

با مراجعه به رسانه‌های مختلف، جست‌وجو کنید آیا نرم‌افزاری وجود دارد که با دادن الگوریتم، فلوچارت آن را مورد استفاده قرار دهد.

الگوی پرسش

۱ـ فلوچارت برنامه‌ای را رسم کنید که طول و عرض یک مستطیل را دریافت و مساحت و محیط آن را محاسبه و چاپ کند.

۲ـ روندنمای برنامه‌ای را رسم کنید که سه عدد را دریافت کرده و آنها را با هم مقایسه کند، سپس به ترتیب صعودی چاپ نماید.

۳ـ فلوچارت برنامه‌ای را رسم کنید که ضرایب یک معادله درجه دوم را دریافت کند و در صورت داشتن ریشه‌های حقیقی آنها را محاسبه و چاپ نماید. در غیر این صورت با پیغام مناسب مشخص نماید که معادله ریشه حقیقی ندارد.

۴ـ فلوچارت پرسش‌های ۱ تا ۳ را با نرم‌افزار مناسب رسم کنید.

۲ـ۳ـ یادآوری ساختار برنامه‌نویسی به زبان C

زبان برنامه‌نویسی C در سال ۱۹۷۳ در آزمایشگاه بل طراحی و ارائه شد. خالق این زبان برنامه‌نویسی دنیس ریچی نام دارد. وی در کنار کن تامپسون علاوه‌بر خلق زبان C سیستم عامل یونیکس را توسعه داد. لذا در زمره افراد نامدار در زمینه محاسبات جدید و به عنوان یک فرد مشهور و نامی شناخته شده است.

از ویژگی‌های زبان برنامه‌نویسی C، امکان دسترسی به سخت‌افزار و حافظه‌ها، امکان برنامه‌نویسی مستقل (ماژولار ـ Modular)، استفاده از برنامه‌های یک میکروکنترلر با کمی تغییر برای میکروکنترلرهای دیگر به خاطر استاندارد بودن این زبان نام برد.

در کتاب کارگاهی مونتاژ و دمونتاژ پایه یازدهم با ساختار و چگونگی برنامه‌نویسی به زبان C و کامپایلر کد ویژن آشنا شدید. علت استفاده از کدویژن فراگیر بودن آن در بین کاربران AVR، قابل اجرا بودن آن با حداقل سخت‌افزار رایانه‌ای و ساده و سبک بودن نرم‌افزار است. در این بخش مباحث تکمیلی و پیشرفته‌تری را درباره این موضوع آموزش می‌دهیم.

فعالیت

- با استفاده از رسانه‌های موجود، در مورد زبان‌های برنامه‌نویسی قبل از پدید آمدن زبان C گزارشی تهیه و به کلاس ارائه دهید.
- کامپایلر دیگری علاوه‌بر کد ویژن را بررسی و سعی کنید یک برنامه ساده را در آن محیط بنویسید و با میکروکنترلر اجرا کنید.

■ نرم‌افزار

پس از طراحی و ساخت سخت‌افزارهای میکروکنترلری، لازم است برنامه‌ای برای کنترل آن نوشته شود. تنها زبان قابل فهم برای پردازنده‌ها، زبان ماشین است که از کدهای دو دویی یا باینری (۱۱۰۰۰۱۰۰) تشکیل می‌شود. نوشتن برنامه و رفع اشکال آن به زبان ماشین (low level)کاری دشوار و طاقت فرسا است؛ لذا برای حل این مشکل، از زبان‌های سطح بالا (high level) مانند بیسیک، پاسکال و C استفاده می‌کنیم.

زبان‌های سطح بالا زبان‌هایی هستند که از نظر ساختاری به زبان محاوره انسان شباهت بیشتری دارند. هرقدر زبان برنامه‌نویسی به زبان ماشین که همان صفر و یک است نزدیک‌تر شود، آن زبان را اصطلاحاً زبان سطح پایین می‌گویند.

فعالیت

در مورد زبان اسمبلی Assembly تحقیق کنید و نتیجه را در قالب یک گزارش ارائه دهید.

ساختار زبان C: برای برنامه‌نویسی میکروکنترلرها به زبان C باید از ساختار کلی این زبان که در جدول ۱ـ۳ نشان داده شده است، پیروی کنیم:

جدول ۱ـ۳ـ ساختار و تعاریف مربوط به زبان C همراه با یک نمونه از برنامه

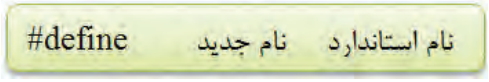
نمونه‌ای از برنامه زبان C	ساختار تعاریف مربوط به زبان C
#include <mega۸.h> #define key PINB.۳ unsigned char i=۰; void wait (void); void main (void) { DDRD=۰xff; while (۱) { if (key==۰) { i++; PORTD=i; wait( ); } } } void wait (void) { int j; for(j=۰;j<۳۰۰۰۰;j++); }	#include (هدر فایل) ماکروها متغیرهای عمومی معرفی توابع void main (void) { .... برنامه اصلی .... } بدنه تابع { .... برنامه تابع .... } ادامه در ستون مقابل ⇒

بارش فکری

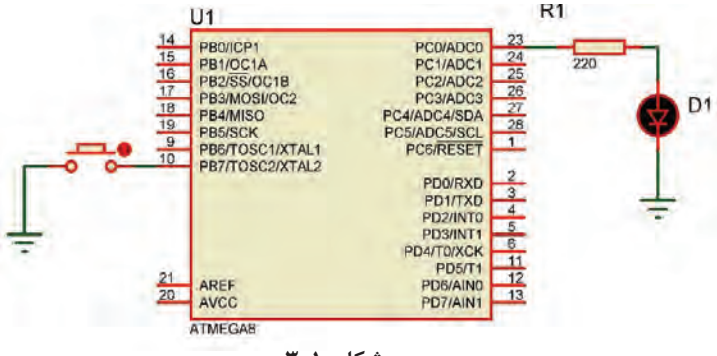
جدول ۱ـ۳ را از طریق بارش فکری به بحث بگذارید و نتیجه را جمع‌بندی کنید و در قالب یک گزارش ارائه دهید.

□ **هدر (سرتیتر) فایل (Header):** برای تعریف ثابت‌ها، متغیرهای عمومی، دستورات پیش پردازنده و همچنین تعریف اولیه توابع از هدر فایل ها استفاده می‌کنیم. در نمونه ارائه شده در جدول ۱ـ۳ عبارت #include <mega8.h> ویژگی‌ها و مشخصات سخت‌افزاری قطعه ATMEGA8 و آدرس حافظه‌ها و رجیسترهای آن را برای کامپایلر معرفی می‌کند.

□ **ماکرو (Macro):** ماکرو رشته‌ای است که می‌تواند شامل حرف، عدد، مقادیر ثابت، توابع و مانند آن باشد. برای تعریف ماکرو از پیش پردازنده #define استفاده می‌شود. همچنین در انجام تعاریف جدید به‌جای مقادیر استاندارد و از پیش تعیین شده در مترجم نیز به‌کار می‌رود. استفاده از ماکروها باعث می‌شود تا با نام‌گذاری‌های جدید به‌جای نام‌های استاندارد، برنامه‌نویسی ساده تر و قابل فهم‌تر شود و از مراجعه پی در پی به نقشه سخت‌افزار جلوگیری به عمل آید. پیش پردازنده #define به‌صورت شکل ۹ـ۳ نوشته می‌شود: به مثال ۴ توجه کنید.



شکل ۹ـ۳ـ روش نوشتن پیش‌پردازنده در ماکرو



شکل ۱۰ـ۳

جدول ۲ـ۳ـ برنامه مثال ۴

با استفاده از ماکرو	بدون استفاده از ماکرو
#define key PINB.۷ #define led PORTC.۰ if (key==۰) led=۱;	if (PINB.۷==۰) PORTC.۰=۱;

مثال ۴: در شکل ۱۰ـ۳ می‌خواهیم با اتصال کلید PINB.7، LED متصل به PORTC.0 روشن شود، برنامه را یک بار بدون استفاده از ماکرو و سپس با استفاده از ماکرو می‌نویسیم.

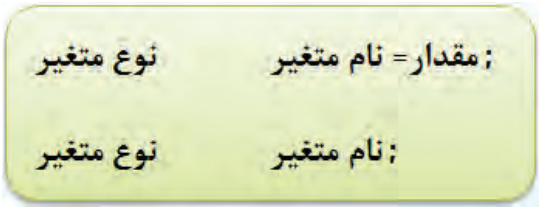
حل: در جدول ۲ـ۳ برنامه نوشته شده با استفاده از ماکرو و بدون استفاده از ماکرو را ملاحظه می‌کنید.

□ **متغیر Variable:** در هر زبان برنامه‌نویسی لازم است محل‌هایی از حافظه را برای نگهداری اعداد،کاراکترها و رشته‌ها در نظر بگیریم تا بتوانیم در هنگام اجرای برنامه آنها را بخوانیم یا روی آن بنویسیم. در نمونه ارائه شده در جدول ۱ـ۳ این متغیر با عبارت unsigned char i=0 نوشته شده که متغیر آن i و مقدار اولیه صفر اعلان شده است.

□ **تابع (Function):** در برنامه‌های طولانی و پیچیده که شامل چندین بخش منطقی و مستقل از هم هستند، بهتر است برای هر قسمت منطقی، برنامه جداگانه‌ای نوشته شود. برنامه‌هایی که برای هر یک از

بخش‌ها نوشته می‌شود را تابع می‌نامند. با استفاده از تابع، از نوشتن تکراری دستورها تا حد زیادی جلوگیری به‌عمل می‌آید. معمولاً توابع در ابتدای برنامه معرفی و بعد از تابع main تعریف می شوند. در نمونه ارائه شده در جدول ۱ـ۳ یک تابع با نام wait معرفی و در انتهای برنامه تعریف شده است. این تابع با هر بار اجرا یک تاخیر ایجاد می‌کند.

□ **تابع main**: ساختار زبان C بر پایه توابع بنا شده است. برای ترجمه و اجرای برنامه لازم است حتماً یکی از توابع را به نام main معرفی کنیم تا به عنوان تابع اصلی شناخته شود. اجرای برنامه با دستورهای نوشته شده در تابع main آغاز می‌شود.



شکل ۱۱ـ۳ـ ساختار چگونگی اعلان یک متغیر در برنامه C

■ **روش‌های بیان و نوشتن متغیرVariable**:
همان‌طور که گفته شد، در برنامه‌نویسی لازم است محل‌هایی از حافظه را برای نگهداری اعداد کاراکتر و رشته‌ها درنظر بگیریم. در زبان C برای اعلان یک متغیر از ساختار شکل ۱۱ـ۳ استفاده می کنیم.
مثال ۵: نمونه‌هایی از معرفی و اعلان متغیر در برنامه را در جدول ۳ـ۳ آورده‌ایم.

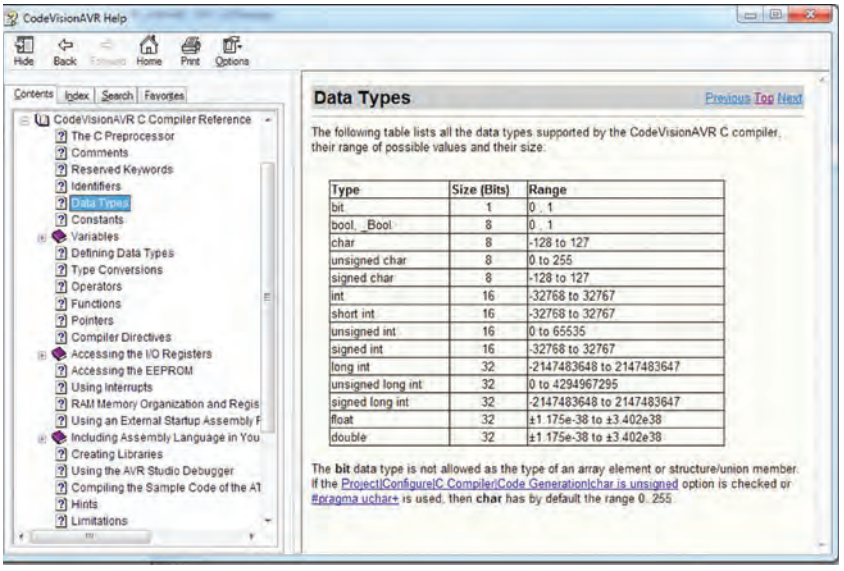
جدول ۳ـ۳

unsigned char	a=۵ ;
int	b,c,d=۲۰۰۰;
float	pi=۳.۱۴;
char	t='R';
char	s[ ]="REZA";

نکته

مقدار دهی اولیه اختیاری است.

■ با توجه به یافته‌های خود این مثال را بررسی کنید و با انطباق با جدول ۱ـ۳ فرایند آن را تعریف نمایید.
■ انواع متغیرها و رنج اعداد قابل نمایش به‌وسیله آنها در Help برنامه و در بخش Data Types طبق شکل ۱۲ـ۳ در دسترس است.



شکل ۱۲ـ ۳ـ جدول متغیرهای C در کدویژن

■ بازه اعداد قابل ذخیره در متغیرهای بدون علامت و علامت‌دار از روابط داده شده در جدول ۴ـ۳ محاسبه می‌شود.

جدول ۴ـ۳

علامت‌دار	بدون علامت
$-۲^{n-۱} \le N \le ۲^{n-۱}-۱$	$۰ \le N \le ۲^{n}-۱$
در روابط بالا n تعداد بیت و N رنج عدد قابل نمایش می باشد	

مثلاً برای دو متغیر signed char و unsigned char که هشت بیتی هستند، بازه اعداد قابل ذخیره را در جدول ۵ـ۳ ملاحظه می‌کنید:

جدول ۵ـ۳

signed char	unsigned char
$-۲^{۸-۱} \le N \le ۲^{۸-۱}-۱$ $-۱۲۸ \le N \le ۱۲۷$	$۰ \le N \le ۲^{۸}-۱$ $۰ \le N \le ۲۵۵$

فعالیت

با اعضاء گروه خود محاسبات مربوط به بازه اعداد قابل نمایش را برای یک متغیر ۱۶ بیتی انجام دهید.

۳ـ۳ـ نوشتن اعداد در مبناهای مختلف در زبان C

در کد ویژن می‌توانید اعداد را در مبناهای ۲، ۸، ۱۰ و ۱۶ بنویسید. در جدول ۶ـ۳، اعداد ۰ تا ۱۵ در مبناهای ۲، ۸، ۱۰ و ۱۶ جهت یادآوری نوشته شده است.

جدول ۶ـ۳ـ تبدیل مبنا

مبنای ۱۰	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵
مبنای ۲	۰۰۰۰	۰۰۰۱	۰۰۱۰	۰۰۱۱	۰۱۰۰	۰۱۰۱	۰۱۱۰	۰۱۱۱	۱۰۰۰	۱۰۰۱	۱۰۱۰	۱۰۱۱	۱۱۰۰	۱۱۰۱	۱۱۱۰	۱۱۱۱
مبنای ۱۶	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	A	B	C	D	E	F
مبنای ۸	۰	۱	۲	۳	۴	۵	۶	۷	۱۰	۱۱	۱۲	۱۳	۱۴	۱۵	۱۶	۱۷

مثال ۶: می‌خواهیم عدد ۱۲ را در مبناهای مختلف به PORTD ارسال کنیم. برنامه مربوط به مثال ۶ به‌صورت جدول ۷ـ۳ نوشته می‌شود. همان‌طور که

جدول ۷ـ۳

```
(۱۲)۱۰=(۱۱۰۰)۲=(۱۴)۸=(C)۱۶
PORTD=۱۲;          مبنای ۱۰
PORTD=۰b۱۱۰۰;      مبنای ۲
PORTD=۰O۱۴;        مبنای ۸
PORTD=۰xC;         مبنای ۱۶
```

در جدول ۶ـ۳ دیده می‌شود برای نوشتن اعداد در مبنای ۲ از پیشوند ۰b، برای مبنای ۸ از پیشوند ۰O و برای مبنای ۱۶ از پیشوند ۰x استفاده می‌شود. مبنای ۱۰ نیازی به پیشوند ندارد. توجه داشته باشید هر سه پیشوند با صفر شروع می‌شود.

مثال ۷: طبق شکل ۹ـ۳ هشت عدد LED را به PORTB متصل کنید و یک شمارنده بالا شمار بر روی آن ایجاد نمایید. در جدول ۸ـ۳ برنامه مربوط به مثال ۷ را ملاحظه می‌کنید.

جدول ۸ـ۳

```
#include <mega۸.h>
#include <delay.h>
unsigned char i=۰;
void main(void)
{
DDRB=۰xFF;
while(۱)
{
 PORTB=i;
 delay_ms(۵۰۰);
 i++;
}
}
```

نکته

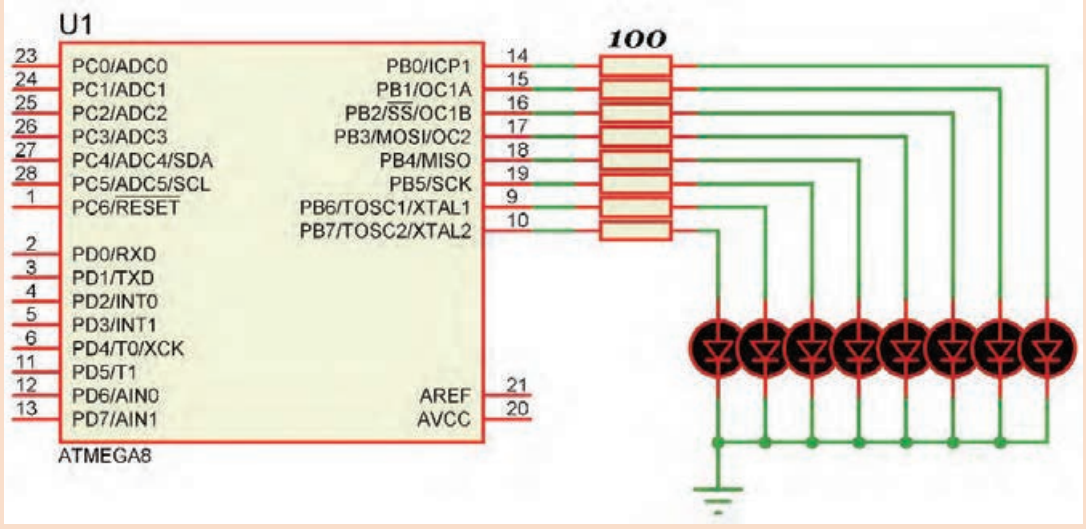
در نوشتن برنامه به زبان C باید دستورها با حروف کوچک و نام رجیسترها با حروف بزرگ نوشته شود.

فعالیت

برنامه مثال ۷ را در گروه خود بررسی کنید و فرایند اجرای آن را طبق الگوی برنامه داده شده به بحث بگذارید. نتیجه را در قالب یک گزارش ارائه دهید.

فعالیت

با توجه به مدار شکل ۱۳ـ۳ برنامه‌ای بنویسید که روی PORTB یک شمارنده پایین شمار ایجاد کند.



شکل ۱۳ـ۳

۴ـ۳ـ عملگرها (Operators)

■ تعریف عبارت

در ریاضیات، به مجموعه‌ای مرکب از اعداد، متغیرها و عملگرها، عبارت گفته می‌شود. برای مثال هر یک از جملات زیر یک عبارت ریاضی است.

$$-x \quad ۳/۱۴\times R\times R \quad ۷\times ۲+۴ \quad ۳+۶$$

درجمله ۳+۶ علامت + را عملگر و دو عدد ۳ و ۶ را عملوند می‌گویند. از آنجایی که عملگر جمع، بر روی دو عملوند عمل می‌کند به آن عملگر دوتایی گفته می‌شود.

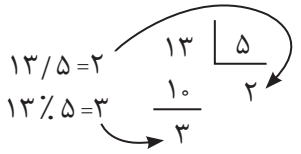
در جمله x- عملگر قرینه‌ساز فقط یک عملگر دارد و به آن عملگر یکتایی گفته می‌شود.

عملگرهایی که می‌توانید در زبان C و در برنامه کدویژن استفاده کنید، طبق شکل ۱۴ـ۳ در Help برنامه و در بخش Operators آورده شده است.

در جدول ۹ـ۳ فهرست عملگرهای ریاضی را به ترتیب اولویت مشاهده می‌کنید:

عملگرهای جمع (+)، تفریق (-) و ضرب (*): این عملگرها در واقع همان محاسبات ریاضی هستند که تاکنون آموخته‌اید.

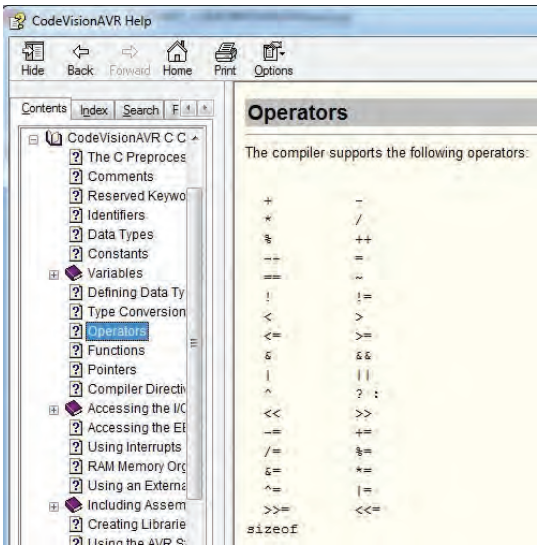
عملگرهای تقسیم معمولی (/)، باقی‌مانده (٪) و اعشار (.): عملکرد این عملگرها به‌صورت زیر تعریف می‌شود.



در عملگر تقسیم معمولی اگر هر دو عملوند از نوع عدد صحیح باشند، نتیجه نیز عددی صحیح خواهد بود. اگر یکی یا هر دو عملوند از نوع عدد اعشاری باشند، نتیجه عدد اعشاری خواهد شد.

۷/۲=۳ , ۷/۲.۰=۳.۵ , ۷.۰/۲=۳.۵ , ۷.۰/۲.۰=۳.۵

عملگرهای افزایش به مقدار یک واحد (++) و کاهش به مقدار یک واحد (--): در جدول ۱۰ـ۳ عملکرد این عملگرها با مثال نشان داده شده است.



شکل ۱۴ـ۳ـ عملگرها در کدویژن

جدول ۹ـ۳ـ اولویت عملگرهای ریاضی در زبان C همراه با مثال

الویت	نام عملگر	نشانه	مثال	نوع عملگر
۱	قرینه	-	-۷	یکتایی
۲	ضرب تقسیم باقیمانده تقسیم	* / ٪	۷*۱۲ ۱۷/۴ ۱۳٪۵	دوتایی
۳	جمع تفریق	+ -	۶۷+۹ ۵۶-۷۸	دوتایی

جدول ۱۰ـ۳ـ عملگرهای افزایش و کاهش همراه با مثال

نتیجه	عملگر	مقدار اولیه
a=۶	a++	a=۵
a=۴	a--	a=۵

عملگرهای انتساب (=)، شرط مساوی (==) و شرط نامساوی (=!): نماد = مقداری را به یک متغیر نسبت می‌دهد و نماد == بررسی می‌کند که آیا دو مقدار با هم مساوی هستند یا خیر. همچنین نماد =! بررسی می‌کند که آیا دو مقدار با هم نامساوی هستند یا خیر.

- در دستور ;K=۵ مقدار ۵ در متغیر K قرار می‌گیرد.
- در دستور ;K==۵ مقدار K با عدد ۵ مقایسه می‌شود. اگر برابر باشند، نتیجه دستور مقدار یک یا TRUE است. در صورتی که مقدار K با عدد ۵ برابر نباشد، نتیجه دستور مقدار صفر یا FALSE خواهد بود.
- در دستور ;K!=۵ مقدار K با عدد ۵ مقایسه می‌شود. اگر برابر نباشند، نتیجه دستور مقدار یک یا TRUE و در صورتی که برابر باشند مقدار برابر با صفر یا FALSE است.

عملگرهای بیتی و منطقی: نماد این عملگرها به‌صورت !، ~، &، &&، |، || و ^ است. در جدول ۱۱ـ۳ عملگرهای بیتی و منطقی و نماد آنها را ملاحظه می‌کنید.

جدول ۱۱ـ۳ـ عملگرهای بیتی و منطقی

عملگر	بیتی	منطقی
NOT	~	!
AND	&	&&
OR	|	||
XOR	^	ندارد

در عملگرهای بیتی ابتدا عملوندها به‌صورت باینری نوشته و سپس بیت به بیت عمل مورد نظر بر روی بیت‌ها انجام می‌شود.

مثال ۸: اگر a=۰x۵۶ و b=۰x۹C باشد، مقادیر زیر را به‌دست آورید.

PORTD=a & b; PORTD=a | b; PORTD=a ^ b; PORTD=~a;

در جدول ۱۲ـ۳ عملگرها، نوع عملیات و نتیجه حاصل شده را ملاحظه می‌کنید.

در عملگرهای منطقی ابتدا درستی یا نادرستی هرعبارت مشخص و سپس عمل مورد نظر بر روی آنها انجام می‌شود.

جدول ۱۲ـ۳ـ پاسخ مثال ۷ عملگرهای بیتی

عملگر	عملیات	نتیجه
AND	a=۰x۵۶ ۰۱۰۱۰۱۱۰ b=۰x۹C ۱۰۰۱۱۱۰۰ a&b ۰۰۰۱۰۱۰۰	PORTD =۰۰۰۱۰۱۰۰=۰x۱۴
OR	a=۰x۵۶ ۰۱۰۱۰۱۱۰ b=۰x۹C ۱۰۰۱۱۱۰۰ a|b ۱۱۰۱۱۱۱۰	PORTD =۱۱۰۱۱۱۱۰=۰xDE
XOR	a=۰x۵۶ ۰۱۰۱۰۱۱۰ b=۰x۹C ۱۰۰۱۱۱۰۰ a^b ۱۱۰۰۱۰۱۰	PORTD =۰۰۰۱۰۱۰۰=۰xCA
NOT	a=۰x۵۶ ۰۱۰۱۰۱۱۰ ~a ۱۰۱۰۱۰۰۱	PORTD =۱۰۱۰۱۰۰۱=۰xA۹

فعالیت

در گروه کاری خود، جدول ۱۲ـ۳ را به بحث بگذارید و نتیجه آن را جمع‌بندی کرده و در قالب گزارش ارائه دهید.

مثال ۹: در جدول ۱۳ـ۳ که قطعه‌ای از یک برنامه است، اگر a=۵ و b=۸ باشد، بر روی پورت D چه عددی نمایش داده خواهد شد؟

در این مثال نتیجه عبارت a>۶ نادرست (false) و نتیجه عبارت b<۱۰ درست (true) می‌باشد.پس نتیجه عملگر منطقی AND نادرست خواهد بود و به دلیل برقرار نبودن شرط، دستور مربوط به else اجرا و روی PORTD عدد ۵۵ نمایش داده می‌شود.

شیفت به چپ و راست: در شکل ۱۵ـ۳ علامت شیفت یا جابه‌جایی به چپ و راست را مشاهده می‌کنید.

شکل ۱۵ـ۳ شیفت به چپ و راست

□ با اجرای عملگرهای شیفت، از یک سمت «صفر» وارد و از سمت دیگر «یک بیت» خارج می‌شود. نماد شیفت به چپ با دو علامت کوچکتر « << » و نماد شیفت به راست با دو علامت بزرگتر « >> » نشان داده می‌شود. در عبارت a>>۳ یعنی عدد موجود در متغیر a سه بار به سمت راست انتقال یابد.

مثال ۱۰: اگر b=۰xFE و a=۰xCF باشد، حاصل عبارت PORTD =((a>>۳) & (b<<۲)) را به‌دست آورید.

در جدول ۱۳ـ۳ پاسخ مربوط به مثال ۱۰ را مشاهده می‌کنید.

a=۱۱۰۰۱۱۱۱ b=۱۱۱۱۱۱۱۰

جدول ۱۳ـ۳ـ کد مثال ۹

```
if ((a>۶) && (b<۱۰))
PORTD =۹۹;
else
PORTD = ۵۵;
```

فعالیت

درباره جدول ۱۴ـ۳ و چگونگی کار عملگرها در آن بحث کنید و نتیجه را در یک پاراگراف بنویسید.

جدول ۱۴ـ۳ـ پاسخ مثال ۹ عملگرهای شیفت و AND

عملگر	نتیجه
a>>۳	۰۰۰۱۱۰۰۱
b<<۲	۱۱۱۱۱۰۰۰
&	۰۰۰۱۱۰۰۰

جدول ۱۵ـ۳ـ کد مثال ۱۱

```
if (a<b)
PORTD =۹۹;
else
PORTD =۵۵;
```

عملگرهای شرط بزرگ‌تر (>) و شرط کوچک‌تر (<):

مثال۱۱: در قطعه برنامه‌ای که در جدول ۱۵ـ۳ نشان داده شده است، اگر a=۵ و b=۸ باشد، پورت D چه عددی را نمایش می‌دهد؟

در این مثال به دلیل برقرار بودن شرط، دستور PORTD =۹۹ اجرا می‌شود.

بارش فکری

جدول ۱۵ـ۳ را از طریق بارش فکری به بحث بگذارید و نتیجه را با پاسخ داده شده مقایسه کنید.

عملگر شرطی «: ? »: این عملگر مانند دستور if else عمل می‌کند و به‌صورت زیر نوشته می‌شود:

(دستور دوم) : (دستور اول) ؟ (عبارت شرطی)

□اگر شرط موجود در عبارت شرطی برقرار باشد دستور اول و در غیر این صورت، دستور دوم اجرا می‌شود.

(a>b) ? (PORTD =۹۹) : (PORTD =۵۵)

عملگرهای تخصیص مرکب

توسط این روش می‌توان عبارات محاسبه ای را مانند جدول ۱۶ـ۳ به‌صورت خلاصه نوشت.

جدول ۱۶ـ۳ـ عملگرهای تخصیص مرکب

شکل خلاصه شده	شکل اصلی
a+ = 5	a = a+5
b* =7	b = b * 7
c& = 8	c = c & 8
d << =4	d = d<<4

پژوهش

با مراجعه به رسانه‌های مختلف، درباره عملگرهای تخصیص مرکب تحقیق کنید و نتیجه را در قالب گزارش ارائه دهید.

عملگر sizeof:

خروجی عملگر sizeof، مقدار حافظه‌ای است که یک متغیر، برحسب بایت اشغال می‌کند.در جدول ۱۷ـ۳ مثال‌هایی آورده شده است.

جدول ۱۷ـ۳ مثال‌های مربوط به عملگر sizeof و نتیجه آنها

متغیر	عملیات	نتیجه
char a;	x= sizeof (a)	x =۱
int b;	x= sizeof (b)	x =۲
float c;	x= sizeof (c)	x =۴

الگوی پرسش

۱ـ برنامه‌ای بنویسید که عددی را از PORTB دریافت کند، اگر عدد دریافت شده، فرد بود روی PORTD عدد ۱ و اگر زوج بود عدد ۲ را روی PORTD نمایش دهد.

۲ـ برنامه‌ای بنویسید که یک عدد باینری را مشابه جدول ۱۸ـ۳ از PORTB دریافت کند، و تعداد صفرهای آن را روی PORTC و تعداد یک های آن را روی PORTD نمایش دهد.

۳ـ برنامه‌ای بنویسید که مشابه جدول ۱۹ـ۳ یک عدد باینری را از PORTB دریافت کند، در صورتی که خروجی دو سنسور به این پورت متصل شده باشند، عدد ارسالی هر سنسور را جدا کرده و به پورت‌های C و D ارسال نماید و آن را نمایش دهد.

جدول ۱۸ـ۳

PORTB	PORTC	PORTD
۱۰۱۱۰۱۱۰	۳	۵

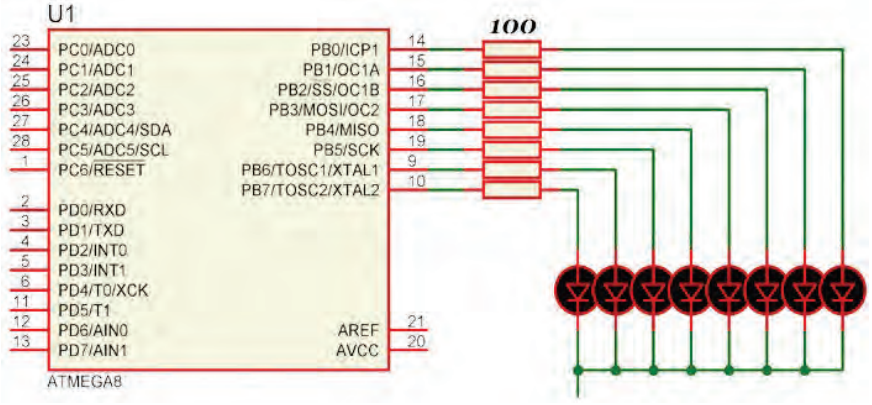
جدول ۱۹ـ۳

PORTB		PORTC	PORTD
S۱	S۲	S۱	S۲
۱۰۰۱	۰۱۱۱	۹	۷

مثال۱۲: با توجه به شکل ۱۶ـ۳ که هشت عدد LED به PORTB متصل است برنامه‌ای بنویسید که LEDها به‌صورت چشمک‌زن عمل کنند.

جدول ۲۰ـ۳

```
#include <mega۸.h>
#include <delay.h>
void main(void)
{
 DDRB=۰xff;
 while(۱)
 {
 PORTB=۰b۰۰۰۰۱۱۱۱;
 delay_ms(۵۰۰);
 PORTB=۰b۱۱۱۱۰۰۰۰;
 delay_ms(۵۰۰);
}
}
```



شکل ۱۶ـ۳ مدار مثال ۱۲

فعالیت

در ساعات غیر درسی، برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی نمایید. در صورت امکان، مدار را به‌صورت سخت‌افزاری نیز اجرا کنید. نتیجه را در قالب یک گزارش ارائه دهید.

مثال ۱۳: برای شکل ۱۶ـ۳ برنامه‌ای بنویسید که یک شمارنده حلقوی روی آن ایجاد شود.

جدول ۲۱ـ۳

```
#include <mega۸.h>
#include <delay.h>
void main(void)
{
 DDRB=۰xff;
 while(۱)
 {
 PORTB=۰b۰۰۰۰۰۰۰۱;
 delay_ms(۵۰۰);
 PORTB=۰b۰۰۰۰۰۰۱۰;
 delay_ms(۵۰۰);
 PORTB=۰b۰۰۰۰۰۱۰۰;
 delay_ms(۵۰۰);
 PORTB=۰b۰۰۰۱۰۰۰۰;
 delay_ms(۵۰۰);
 PORTB=۰b۰۰۱۰۰۰۰۰;
 delay_ms(۵۰۰);
 PORTB=۰b۰۱۰۰۰۰۰۰;
 delay_ms(۵۰۰);
 PORTB=۰b۱۰۰۰۰۰۰۰;
 delay_ms(۵۰۰);
 }
}
```

ادامه در ستون مقابل ⇒

در مثال ۱۳ چون برنامه با دستورهای ساده نوشته شد تعداد خطوط برنامه نیز بسیار زیاد شده است.

مثال۱۴: برنامه مثال ۱۳ را به کمک حلقه و دستور انتقال به چپ << بنویسید. این کار باعث کاهش تعداد خطوط برنامه می‌شود. برنامه نوشته شده را با برنامه داده شده در جدول ۲۲ـ۳ مقایسه کنید.

جدول ۲۲ـ۳

```
#include <mega۸.h>
#include <delay.h>
unsigned char i;
void main(void)
{
 DDRB=۰xff;

ادامه در ستون مقابل ⇒
```

```
while(۱)
{
for (i=۰ ; i<۸ ; i++)
{
PORTB=۱<< i;
delay_ms(۵۰۰);
  }
 }
}
```

نکته

در این برنامه متغیر i با هر بار اجرای حلقه for یک واحد افزایش می‌یابد. در این حالت، در دستور PORTB=۱<< i; عدد یک به اندازه عدد متغیر i به سمت چپ انتقال می‌یابد.

فعالیت

برای شکل ۱۲ـ۳ برنامه‌ای بنویسید که یک شمارنده جانسون بر روی آن ایجاد شود. این فعالیت را یک بار با دستورهای ساده و بار دیگر با استفاده از حلقه و دستور انتقال انجام دهید.

راهنمایی: از عملگر ^ (XOR) استفاده کنید.

□در شمارنده جانسون LEDها یکی پس از دیگری روشن می‌شوند و روشن باقی می مانند تا همه آنها روشن شوند. سپس به همان ترتیب که روشن شده بودند خاموش خواهند شد.

۵ـ۳ـ دستور شرطی if

اگر قرار باشد دستور یا دستورهایی بنا به شرایط خاص انجام شود، از دستورهای شرطی استفاده می‌کنیم.

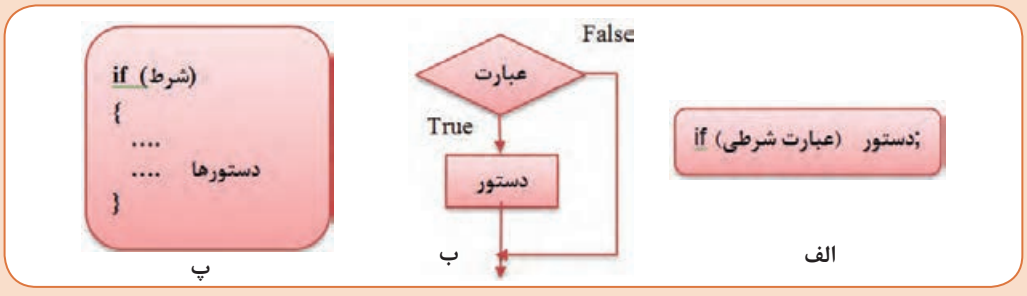
یکی از دستورهای شرطی پرکاربرد دستور if است که در شکل الف ـ ۱۷ـ ۳ مشاهده می‌کنید.

□توجه داشته باشید که نتیجه عبارت شرطی همواره یکی از دو حالت (true) یا (false) خواهد بود.

فلوچارت دستور if را در شکل ب ـ ۱۷ـ ۳ مشاهده می‌کنید.

نکته

اگر تعداد دستورهای بعد از if بیش از یک دستور باشد، طبق شکل پ ـ ۱۷ـ ۳ باید بین دو آکولاد {} قرار گیرد.



شکل ۱۷ـ۳

جدول ۲۳ـ۳

```
#include < mega۸.h >
#include < delay.h >
signed char i=۰;
void main (void)
{
 DDRB =۰xff;
 while (۱)
⇒ ادامه در ستون مقابل
{
PORTB =i;
delay_ms (۵۰۰);
i++;
if (i==۱۰) i=۰;
}
}
```

مثال ۱۵: برای شکل ۱۶ـ۳ برنامه‌ای بنویسید که یک شمارنده باینری بالاشمار صفر تا ۹ ساخته شود.

در جدول ۲۳ـ۳ برنامه داده شده است. این برنامه را بررسی کنید و درباره آن بحث کنید.

فعالیت

در ساعات غیر درسی، برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی نمایید. در صورت امکان، مدار را به‌صورت سخت‌افزاری نیز اجرا کنید. نتیجه را در قالب یک گزارش ارائه دهید.

فعالیت

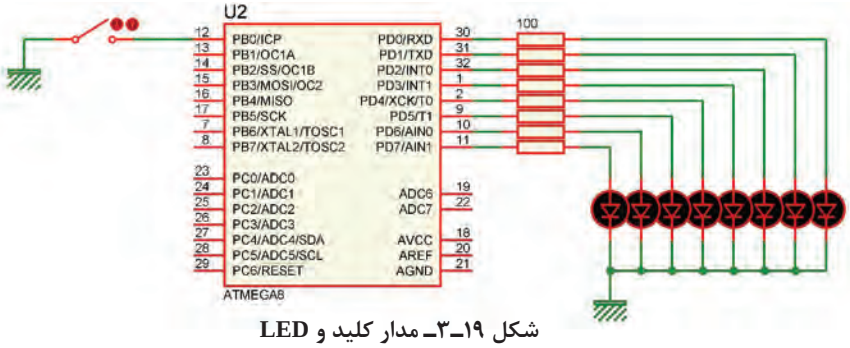
برنامه مثال ۱۵ را طوری بنویسید که یک شمارنده باینری پایین شمار ۹ تا صفر داشته باشیم.

۶ـ۳ـ دستور if - else

اگر لازم باشد که در صورت برقرار بودن یک شرط، دستور یا دستورهایی اجرا شود، و در صورت برقرار نبودن آن شرط، دستورهای دیگری اجرا شود، از دستور if_else استفاده می‌کنیم. ساختار این دستور در شکل ۱۸ـ۳ نشان داده شده است.

مثال۱۶: هشت عدد LED به PORTD و یک کلید به PINB.۰ مانند شکل ۱۹ـ۳ متصل است برنامه‌ای بنویسید که اگر کلید باز بود تمام LEDهای روی PORTD روشن و اگر کلید بسته بود همه LEDها خاموش شوند.

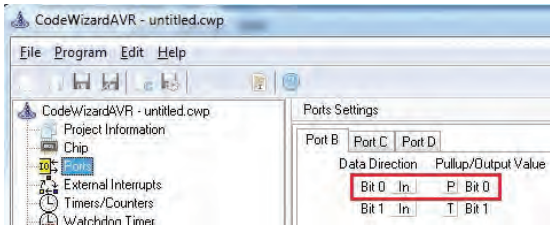
شکل ۱۸ـ۳ـ دستور if_else



شکل ۱۹ـ۳ـ مدار کلید و LED

توجه داشته باشید هنگام تنظیم wizard باید مانند شکل ۲۰ـ۳ بیت صفر مربوط به PORTB، یعنی PB۰ را در حالت ورودی قرار دهیم و Pull_Up داخلی آن را فعال کنیم.

در جدول ۲۴ـ۳ برنامه داده شده است. این برنامه را بررسی و درباره آن بحث کنید.



شکل ۲۰ـ۳ تنظیم

جدول ۲۴ـ۳

```
#include <mega۸. h>
#include <delay. h>
void main (void)
{
while(۱)
{
 if ( PINB.۰= =۱)
{
 ⇒ ادامه در ستون مقابل
```

```
PORTD =۰b۱۱۱۱۱۱۱۱;
 }
else
 {
PORTD =۰b۰۰۰۰۰۰۰۰;
 }
}
}
```

فعالیت

برای سخت‌افزار شکل ۱۹ـ۳ برنامه‌ای بنویسید که اگر کلید باز بود روی PORTD یک شمارنده بالا شمار و اگر کلید بسته بود شمارنده پایین شمار داشته باشیم.

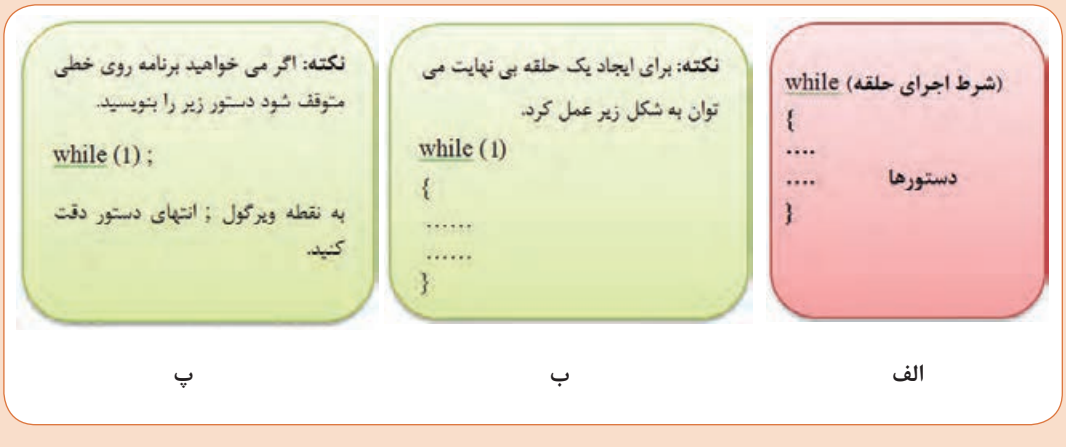
۷ـ۳ـ حلقه (Loop)

در برنامه‌نویسی حالت هایی پیش می‌آید که لازم است دستور یا دستورهایی چندین بار به‌صورت حلقه و تکراری اجرا شود. در این شرایط باید آنها را درون یک حلقه قرار دهیم تا به تعداد دفعات مورد نیاز تکرار شوند. در هر حلقه، یک شمارنده (Counter) وجود دارد که آن را با عدد حلقه پر می‌کنیم و با هر بار اجرا یک واحد از آن کم می‌کنیم. وقتی محتوای شمارنده صفر شود، از حلقه خارج می‌شویم.

■ **دستور while** : یکی از روش‌های ایجاد حلقه استفاده از دستور while است که در شکل الف ـ ۲۱ـ ۳ آن را ملاحظه می‌کنید.

بحث کنید

به نکات ذکر شده در شکل‌های ب ـ ۲۱ـ ۳ و پ ـ ۲۱ـ ۳ توجه کنید و آنها را مورد بحث قرار دهید. سپس نتیجه را جمع‌بندی کنید.



الف ب پ

شکل ۲۱ـ۳

مثال ۱۷: برای مدار شکل ۱۵ـ۳ با استفاده از دستور while برنامه‌ای بنویسید که LEDها را پنج بار روشن و خاموش کند.
در جدول ۲۵ـ۳ برنامه داده شده است. این برنامه را بررسی کنید و درباره آن بحث کنید.

جدول ۲۵ـ۳

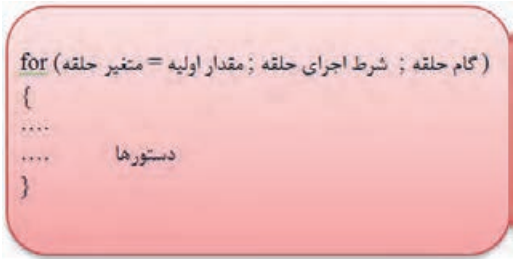
```
#include <mega۸. h>
#include <delay. h>
signed char i;
void main (void)
{
DDRB=۰xff;
i=۵;
while(i >۰) //
تا زمانی که i بزرگ‌تر از صفر است
حلقه انجام می‌شود
⇒ ادامه در ستون مقابل
{
PORTB =۰xff;
delay_ms(۵۰۰);
PORTB =۰x۰۰;
delay_ms(۵۰۰);
i_ _;
}
while (۱);}
```

فعالیت

با استفاده از دستور while برنامه‌ای بنویسید که بر روی PORTB یک شمارنده باینری بالا شمار صفر تا ۹ داشته باشیم.

■ **حلقه for:** هرگاه تعداد دفعات تکرار حلقه مشخص باشد، بهتر است از دستور for با ساختار داده شده در شکل ۲۲ـ۳ استفاده کنیم.

مثال ۱۸: برای مدار شکل ۱۵ـ۳ با استفاده از حلقه for برنامه‌ای بنویسید که LEDها را پنج بار روشن و خاموش کند.
در جدول ۲۶ـ۳ برنامه داده شده است. این برنامه را بررسی و درباره آن بحث کنید.



شکل ۲۲ـ۳

جدول ۲۶ـ۳

```
#include <mega۸.h>
#include <delay.h>
unsigned char i;
void main(void)
{
 DDRB=۰xff;
for(i=۰;i<۵;i++)
⇒ ادامه در ستون مقابل
{
PORTB=۰xFF;
delay_ms(۵۰۰);
PORTB=۰x۰۰;
delay_ms(۵۰۰);
}
while(۱);
}
```

فعالیت

برای مدار شکل ۱۹ـ۳ با استفاده از دستور for برنامه‌ای بنویسید که بر روی PORTB یک شمارنده باینری بالا شمار از صفر تا ۹ داشته باشیم.

۸ـ۳ـ تابع (Function)

■ ساختار زبان C بر پایه توابع است. کاربر می‌تواند هر بخش از برنامه را به‌صورت یک تابع بنویسد و زمانی که لازم بود آن را فراخوانی و اجرا نماید. نوشتن برنامه به‌صورت توابع دارای مزایایی به شرح زیر است:

☑ خواندن برنامه و درک آن آسان‌تر است.

☑ رفع اشکال از برنامه ساده‌تر است.

☑ اصلاح و ارتقای برنامه ساده‌تر صورت می‌گیرد.

☑ می‌توان از توابع نوشته شده در برنامه‌های دیگر هم استفاده کرد.

■ بسته به این که تابع ورودی یا خروجی داشته باشد چهار حالت زیر به‌وجود می‌آید:

بدون ورودی ـ بدون خروجی `void f1(void)`

با ورودی ـ بدون خروجی `void f2(unsigned char x)`

بدون ورودی ـ با خروجی `char f3(void)`

با ورودی ـ با خروجی `int f4(float y)`

نکته

■ اگر تابعی دارای چند ورودی از یک نوع باشد باید مانند نمونه زیر تک تک آنها معرفی شوند:

درست `void f1(int x , int y , int z)` نادرست `void f1(int x , y , z)`

■ هر تابع فقط می‌تواند یک خروجی داشته باشد. برای خروج دیتا از دستور return استفاده می‌شود.

`return` مقدار ثابت ; یا `return` نام متغیر ;

■ انواع توابع در زبان C

☑ توابع کتابخانه ای استاندارد (standard library function)

☑ توابع تعریف شده توسط کاربر (user defined function)

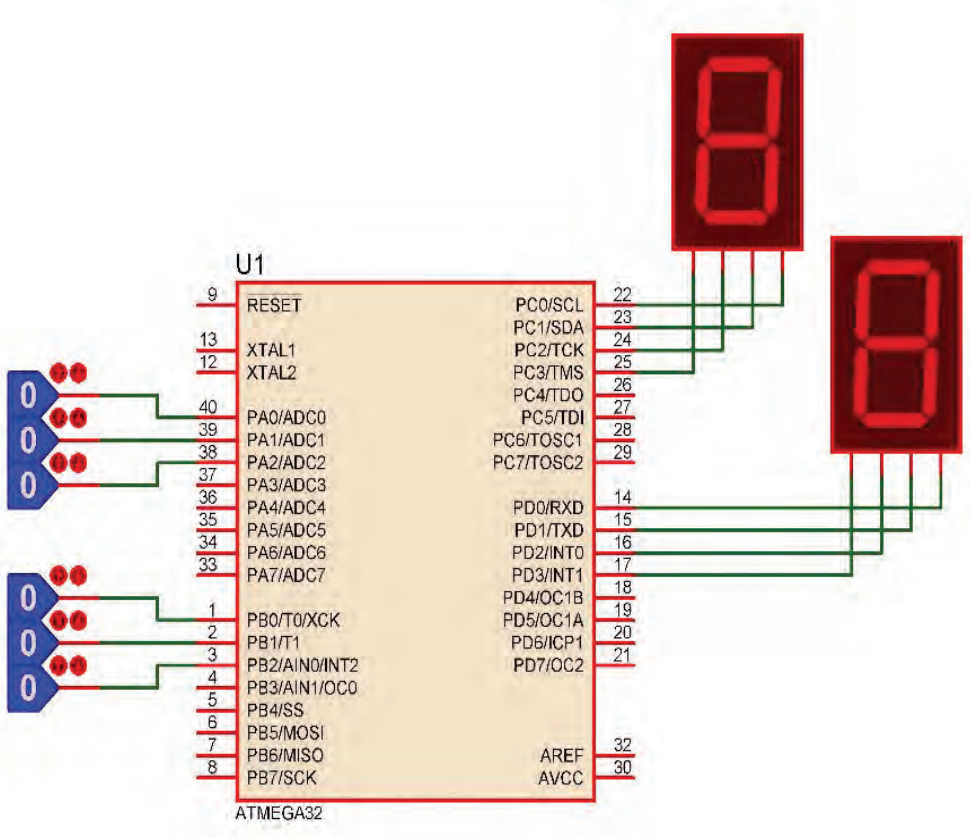
برخی از توابع پرکاربرد که در اغلب برنامه‌ها مورد استفاده قرار می‌گیرند، از قبل نوشته شده و در فایل‌هایی با پسوند h هنگام نصب برنامه در کنار بقیه فایل‌های کامپایلر C ذخیره می‌شوند. این توابع را توابع کتابخانه‌ای می‌نامند. برای مثال توابع ریاضی مانند sin() و cos() در کتابخانه math.h قرار دارند. فایل های دیگری نیز وجود دارند که در آرشیوهای دیگر تجمیع می‌شوند. و در صورت لزوم می‌توانیم از آنها استفاده کنیم.

■ ایجاد تابع جدید

□ محل قرار گرفتن تابع می‌تواند به یکی از شکل‌های نشان داده شده در جدول ۲۷ـ۳ باشد.

اگر توابع مستقل باشند، و از داخل یک تابع، تابع دیگری فراخوانی نشود، انتخاب شکل (الف) یا (ب) تفاوتی ندارد. در غیر این صورت باید روش الف را استفاده کنید.

مثال۱۹: برنامه‌ای بنویسید که مانند شکل ۲۳ـ۳ دو عدد سه بیتی را از PORTA و PORTB دریافت کند، سپس حاصل جمع آنها را روی PORTC و حاصل تفریق آنها را روی PORTD نمایش دهد. این برنامه را یک بار بدون استفاده از تابع و یک بار با استفاده از تابع بنویسید. این مثال را در نرم‌افزار پروتئوس انجام دهید. برای ورود اعداد باینری از کتابخانه Debugging Tools ابزار Logicstate شکل‌الف ـ ۲۴ـ۳ و برای نمایش اعداد از کتابخانه Optoelectronics قطعه BCD ـ SEG ۷، شکل‌ب ـ ۲۴ـ۳ را بردارید. در جدول ۲۸ـ۳ برنامه داده شده است. این برنامه را بررسی کنید و فرآیند اجرای آن را با دقت بیاموزید و درباره آن بحث کنید.



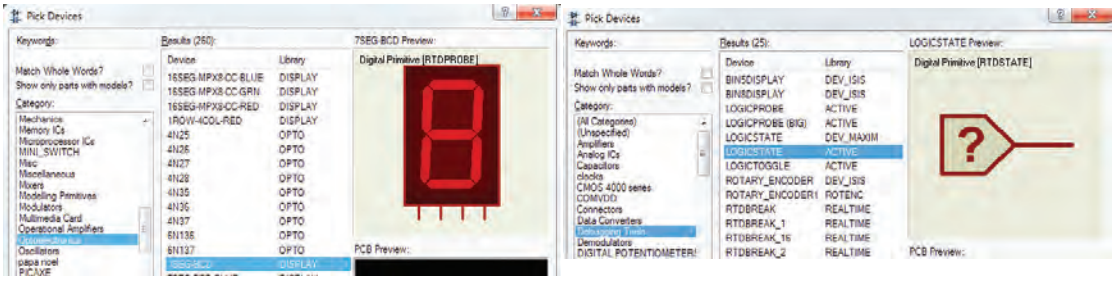
شکل ۲۳ـ۳

جدول ۲۷ـ۳ محل قرارگیری تابع

الف	ب
; معرفی تابع void main (void) { .... فراخوانی تابع ..... } بدنه تابع { .... }	بدنه تابع { .... } void main (void) { .... فراخوانی تابع ..... }

جدول ۲۸ـ۳

```
#include <mega۳۲. h >                    {
unsigned char a,b,c,d;                   a=PINA & ۰x۰۷;        ۱
void main (void)                         b=PINB & ۰x۰۷;        ۲
{                                        c=a+b;
DDRA=۰x۰۰;                               d=a_b;
DDRB=۰x۰۰;                               PORTC=c;
DDRC=۰xff;                               PORTD=d;
DDRD=۰xff;                                }
While (۱)                                }
⇒ ادامه در ستون مقابل
```



الف ـ انتخاب LOGICSTATE

ب ـ انتخاب ۷SEG_BCD

شکل ۲۴ـ۳

□ در خطوط مشخص شده با شماره‌های ۱ و ۲ به عمل انجام شده ماسک (Mask) کردن می‌گویند با این کار فقط سه بیت اول پورت خوانده شده و بقیه بیت ها صفر می شوند. (در پودمان سوم به این نکته اشاره می‌شود که A.0=0 و A.1=A)

□ حال طبق جدول ۲۹ـ۳ هر قطعه از برنامه را به‌صورت یک تابع مستقل می نویسیم و به ترتیب فراخوانی می کنیم.

جدول ۲۹ـ۳ـ مثال برای تابع همراه با توضیح هر بخش

#include <mega32. h >	**معرفی میکرو MEGA۳۲**
void init_port (void) ; void input (void) ; unsigned char sum (unsigned char x, unsigned char y) ; unsigned char sub (unsigned char x, unsigned char y) ; void output (unsigned char x, unsigned char y) ;	معرفی توابع نوشته شده در برنامه و در صورت وجود، همراه با مشخص کردن نوع ورودی و خروجی آنها، توابعی که ورودی یا خروجی ندارند با کلمه void مشخص شده‌اند.
void main (void) { init_port(); while (۱) { input(); c=sum(a,b); d=sub(a,b); output(c,d); } }	تابع اصلی برنامه که توابع در هنگام لزوم در آن فراخوانی شده‌اند.
void init_port (void) { DDRA=۰x۰۰; DDRB=۰x۰۰; DDRC=۰xff; DDRD=۰xff; }	بدنه تابع init_port که در برنامه اصلی یک بار فراخوانی شده و با اجرای آن پورت ها به‌صورت ورودی یا خروجی پیکربندی شده‌اند. init مخفف کلمه Initialization به معنی مقدار دهی اولیه می‌باشد. این تابع، ورودی و خروجی ندارد.

ادامه جدول ۲۹ـ۳

بدنه تابع input که هر بار فراخوانی شود دو عدد از پورت‌های A و B را خوانده و بعد از ماسک کردن، آنها را در متغیرهای a وb قرار می‌دهد. این تابع، ورودی و خروجی ندارد.	void input (void) { a=PINA & 0x0f; b=PINB & 0x0f; }
بدنه تابع sum که هر بار فراخوانی شود دو عدد ورودی خود را با هم جمع ونتیجه را که در z قرار دارد با استفاده از دستور return به برنامه اصلی برمی گرداند. این تابع، هم ورودی و هم خروجی دارد.	unsigned char sum (unsigned char x, unsigned char y) { unsigned char z; z=x+y; return z; }
بدنه تابع sub که هر بار فراخوانی شود دو عدد ورودی خود را از هم تفریق ونتیجه را که در z قرار دارد با استفاده از دستور return به برنامه اصلی برمی‌گرداند. این تابع، هم ورودی و هم خروجی دارد.	unsigned char sub (unsigned char x, unsigned char y) { unsigned char z; z=x_y; return z; }
بدنه تابع output که هر بار فراخوانی شود دو عدد ورودی خود را به پورت‌های C و D ارسال می‌کند. این تابع ورودی دارد ولی خروجی ندارد.	void output (unsigned char x, unsigned char y) { PORTC=x; PORTD=y }

فعالیت

فعالیت‌های زیر را در محیط پروتئوس و با میکرو ATMEGA۳۲، یک بار بدون تابع و یک بار با تابع انجام دهید.

- یک عدد چهار بیتی را از پورت A دریافت کنید. اگر عدد ورودی زوج بود، روی پورت C عدد ۲ و اگر فرد بود عدد ۳ را نمایش دهید.
- یک عدد هشت بیتی را از پورت A دریافت و تعداد یک‌های آن را روی پورت C و تعداد صفرهای آن را روی پورت D نمایش دهید.

الگوی آزمون نرم‌افزاری واحد یادگیری ۳

مشابه یکی از مثال‌ها یا فعالیت های داده شده در متن واحد یادگیری را به‌صورت نرم‌افزاری اجرا کنید.

الگوی آزمون نظری واحد یادگیری ۳

۱ـ فلوچارت برنامه‌ای را بنویسید که بتواند از رابطه $y=(x+1)^2$ جذر بگیرد و مقدار y را به ازاء مقادیر بزرگ‌تر از یک محاسبه کند.

۲ـ اصلاحات Variable، Fuction و main را تعریف کنید.

۳ـ هر یک از دستورات زیر را تفسیر کنید.

```
unsigned char a = ۶ ;
int b,c,d = ۱۰۰۰ ;
float pi =۰,۲۸;
char s[] =”ALI”;
```

۴ـ هریک از دستورات PORTD =12; و PORTD =0b11000; چه مفهومی دارند.
۵ـ دستور a++ نشانه ..... و دستور a-- نشانه ..... است.
۶ـ دردستور شرطی if برای اجرای چند دستور، دستورها همواره باید در بین دو آکولاد قرار گیرد.
درست ☐ نادرست ☐
۷ـ در ویزارد، پول آپ داخلی چه کاربردی دارد؟ شرح دهید. چنانچه فعال نشود چه اشکالی برای برنامه پدید می‌آورد؟

ارزشیابی واحد یادگیری ۳:کسب شایستگی در طراحی الگوریتم(فلوچارت) مدار پروژه ساده الکترونیکی

شرح کار:
۱ـ طراحی الگوریتم و فلوچارت ۲ـ عملگرها و کاربرد آن در برنامه C ۳ـ نوشتن چند برنامه ساده با دستورهای حلقه، شرطی و حلقه شرطی در زبان C ۴ـ تشریح توابع و توابع کتابخانه‌ای استاندارد و استفاده از آنها در نوشتن برنامه ساده به زبان C ۵ـ استفاده از شبیه‌سازی در نرم‌افزار در برنامه C

استاندارد عملکرد: ترسیم اجرای فلوچارت و نوشتن چند برنامه ساده با انواع دستورات اجرای آن به‌صورت نرم‌افزاری

شاخص‌ها:
انتخاب فضای مورد نظر و مناسب بودن آن (میز کار)
تدوین و طراحی الگوریتم و فلوچارت (۲۰ دقیقه)
استفاده از عملگر در برنامه‌های ساده به زبان C (۲۰ دقیقه)
نوشتن برنامه با دستورهای مختلف (۲۵ دقیقه)
استفاده از توابع کتابخانه در برنامه C (۱۵ دقیقه)
اجرای برنامه در نرم‌افزار (۳۰دقیقه)

شرایط انجام کار و ابزار و تجهیزات: شرایط انجام کار مشابه بقیه واحدهای یادگیری

معیار شایستگی:

ردیف	مراحل کار	حداقل نمره قبولی از ۳	نمره هنرجو
۱	طراحی الگوریتم، فلوچارت با استفاده از عملگرها	۱	
۲	نوشتن برنامه ساده،تبدیل مبناها وتحلیل و اجرای آن با نرم‌افزار	۲	
۳	نوشتن برنامه ساده بادستورهای مختلف و اجرای آن با نرم‌افزار	۲	
	شایستگی‌های غیرفنی، ایمنی، بهداشت، توجهات زیست‌محیطی و نگرش: ۱ـ رعایت نکات ایمنی دستگاه‌ها ۲ـ دقت و تمرکز در اجرای کار ۳ـ شایستگی تفکر و یادگیری مادام‌العمر ۴ـ اخلاق حرفه‌ای	۲	
	میانگین نمرات		*

* حداقل میانگین نمرات هنرجو برای قبولی و کسب شایستگی، ۲ می‌باشد.

واحد یادگیری ۴

کسب شایستگی در برنامه‌نویسی به زبان C (یا هر زبان به روز دیگر) و تحلیل برنامه‌های آماده پروژه‌های الکترونیکی

آیا تا به‌حال فکر کرده‌اید:

- برنامه‌نویسی مدارهای الکترونیکی چگونه انجام می‌شود؟
- چگونه می‌توان یک بارگراف (نمودار میله‌ای) را با استفاده از میکروکنترلر به نمایش درآورد؟
- برنامه‌نویسی برای نمایش‌گر LCD چگونه انجام می‌شود؟
- کد اسکی ASCII چیست و چه کاربردی دارد؟
- برای صفحه کلیدها از چه کدهایی استفاده می‌شود؟
- برنامه‌نویسی صفحه‌کلیدها چگونه انجام می‌شود؟
- برای اجرای پروژه‌های الکترونیکی برنامه‌های آماده وجود دارد؟

دنیای برنامه‌نویسی به‌قدری گسترده است که هرگز نمی‌توان برای آن انتهایی در نظر گرفت. امروزه تمام دستگاه‌های الکترونیکی الکتریکی خانگی، اداری، تجاری یا صنعتی که دارای کنترل‌هایی مانند دما، سرعت، نور و حرکت هستند از میکروکنترلر و برنامه‌های آن استفاده می‌کنند. کمی به‌دستگاه‌هایی که در اطراف خود می‌بینید توجه کنید. تقریباً در تمام آنها میکروکنترلر به‌کار گرفته شده است، ساده‌ترین این نوع دستگاه‌ها، دستگاه چای‌ساز است که برای کنترل دما طی سه مرحله برنامه‌ریزی شده است، زمانی که آب به جوش می‌آید و زمانی که باید جوشیدن آب متوقف شود و دما در محدوده ۹۰ درجه ثابت بماند. در این واحد یادگیری بررسی نظری و نرم‌افزاری چند نمونه از پروژه‌های الکترونیکی که در آنها از برنامه‌نویسی و میکروکنترلر استفاده کرده‌اند می‌پردازیم.

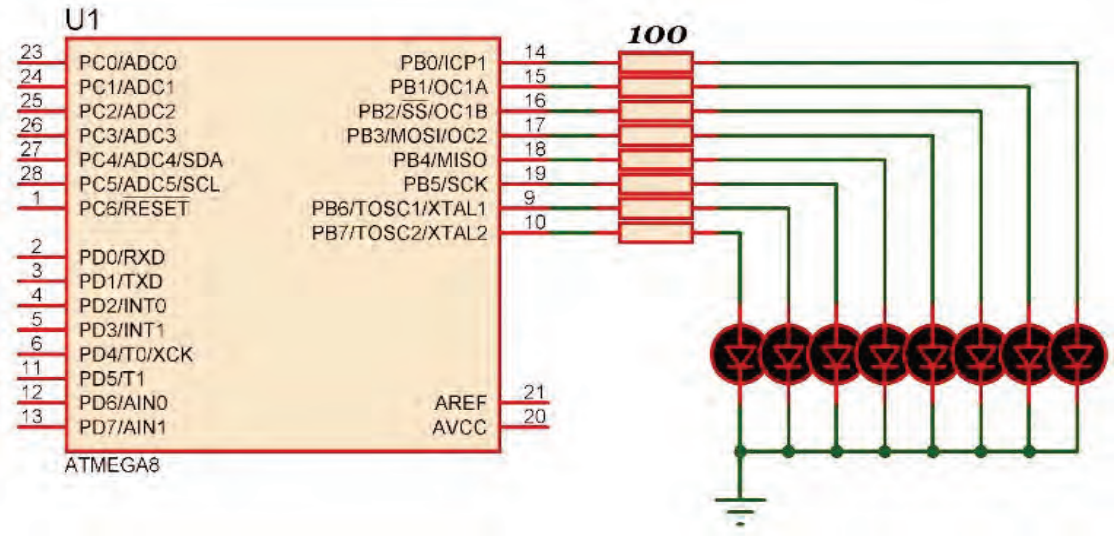
استاندارد عملکرد

برنامه‌نویسی به زبان C (یا هر زبان به‌روز دیگر) برای پروژه‌های عملی و تحلیل برنامه‌های سامانه‌های کنترلی

۱ـ۴ـ برنامه‌نویسی برای مدارهای الکترونیکی با LED

یکی از خروجی‌های پرکاربرد، ساده و متنوع از نظر شکل و رنگ LED است.دراین بخش چند پروژه ساده را با این قطعه انجام می‌دهیم.

مثال ۱: هشت عدد LED مانند شکل ۱ـ۴ به پورت B متصل کنید و یک چشمک زن بسازید. برنامه این مدار در جدول ۱ـ۴ آمده است.



شکل ۱ـ۴ـ مدار چشمک‌زن

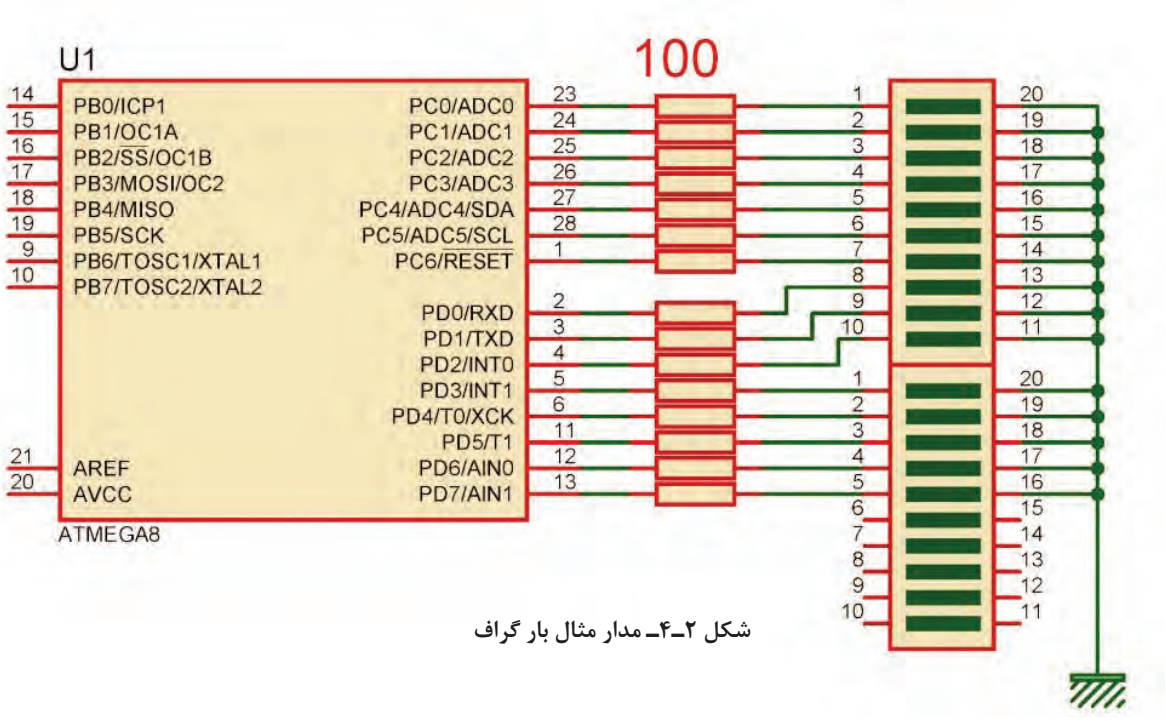
جدول ۱ـ۴

```
#include < mega۸. h >   //   معرفی کتابخانه میکروکنترلر
#include < delay. h >   //   معرفی کتابخانه تأخیر
void main (void)        //   شروع تابع اصلی
{
 DDRB=۰xff;             //   قرار دادن پورت در حالت خروجی
 while(۱)               //   شروع حلقه while
 {
 PORTB=۰b ۱۱۰۰۰۰۱۱;    //   روشن کردن چهار بیت کناری و خاموش کردن چهار بیت وسط
 delay_ms(۵۰۰);         //   ایجاد تاخیر به اندازه نیم ثانیه
 PORTB=۰b۰۰۱۱۱۱۰۰;      //   روشن کردن چهار بیت وسط و خاموش کردن چهار بیت کناری
 delay_ms(۵۰۰);         //   ایجاد تاخیر به اندازه نیم ثانیه
 }                      //   انتهای حلقه while
 }                      //   انتهای تابع اصلی
```

فعالیت

در ساعت‌های غیردرسی، برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی کنید. نتیجه را در قالب یک گزارش ارائه دهید.

مثال ۱: با اتصال پانزده عدد LED به میکروکنترلر مانند شکل ۲ـ۴ یک نمودارمیله‌ای (bar graph) بسازید.



شکل ۲ـ۴ـ مدار مثال بار گراف

جدول ۲ـ۴

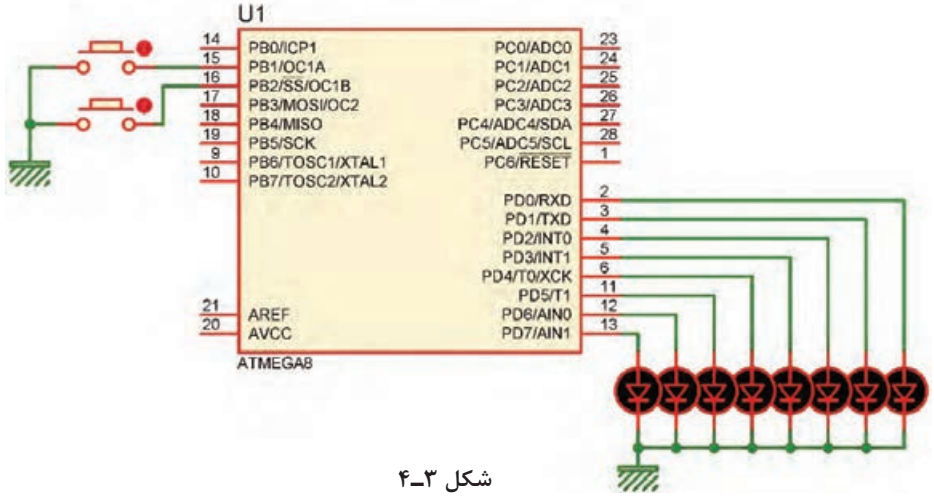
```
#include <mega۸. h > // معرفی کتابخانه میکروکنترلر
#include <delay. h > // معرفی کتابخانه تاخیر
signed char i; // تعریف متغیر علامت‌دار
void main(void) // شروع تابع اصلی
{
DDRC=۰xff; // قرار دادن پورت در حالت خروجی
DDRD=۰xff; // قرار دادن پورت در حالت خروجی
while(۱) // while شروع حلقه
{ // ------ LED ON ------
for(i=۷;i>=۰;i--) // for شروع حلقه
{
                    روشن کردن با استفاده از شیفت و عملگر xor
PORTD=PORTD^(۱<<i); //
delay_ms(۱۰۰); // ایجاد تأخیر
}
 for(i=۶;i>=۰;i--) // for شروع حلقه
{
⇒ ادامه برنامه در ستون مقابل از بالا به پایین
```

```
                    روشن کردن با استفاده از شیفت و عملگر xor
PORTC=PORTC^(۱<<i); //
 delay_ms(۱۰۰); // ایجاد تاخیر
 }
 //------ LED OFF ------
 for(i=۰;i<۷;i++) // for شروع حلقه
 {
                    خاموش کردن با استفاده از شیفت و عملگر xor
 PORTC=PORTC^(۱<<i); //
 delay_ms(۱۰۰); // ایجاد تاخیر
 }
 for(i=۰;i<۸;i++) // for شروع حلقه
 {
                    خاموش کردن با استفاده از شیفت و عملگر xor
 PORTD=PORTD^(۱<<i); //
 delay_ms(۱۰۰); // ایجاد تاخیر
 }
 }// while انتهای حلقه
} // انتهای تابع اصلی
```

۴ـ۲ـ استفاده از کلید (KEY)

یکی از ساده‌ترین و پرکاربردترین قطعات ورودی کلید فشاری است که در اکثر دستگاه‌های الکترونیکی مانند ریموت کنترل تلویزیون، کنترل دمای یخچال و کولر و مانند آنها به‌کار می‌رود.

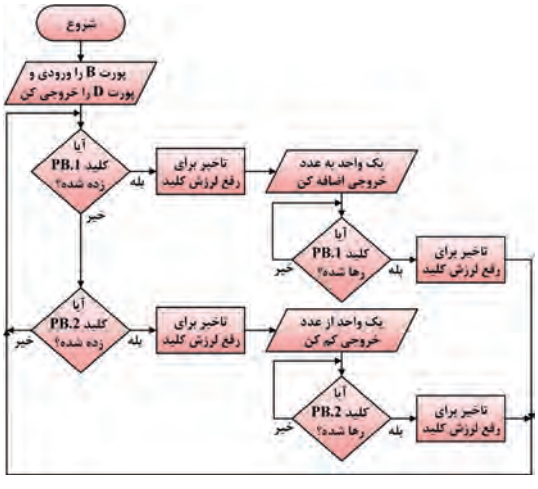
مثال ۳: می‌خواهیم با استفاده از دو کلید، عدد باینری خروجی روی LEDهای متصل به پورت D را کم و زیاد کنیم. هشت عدد LED به PORTD و دو عدد کلید به پین‌های PINB.۱ و PINB.۲ متصل شده است، شکل۳ـ۴. برنامه‌ای بنویسید که با هر بار زدن کلید PINB.۱ عدد روی PORTD افزایش و با هر بار زدن کلید PINB.۲ عدد روی PORTD کاهش یابد. (ورودی کلیدها pull_up داخلی شوند)



شکل ۳ـ۴

■ ابتدا فلوچارت برنامه را به‌صورت شکل ۴ـ۴ رسم می‌کنیم:

■ برنامه را طبق جدول ۳ـ۴ می‌نویسیم و پس از بارگذاری راه‌اندازی می‌کنیم.



شکل ۴ـ۴ـ فلوچارت خواندن کلید و شمارش اعداد

جدول ۳ـ۴

```
#include < mega۸. h >
#include < delay. h >
unsigned char i=۰;
void main (void)
{
DDRB=۰x۰۰;
DDRD=۰xff;
while (۱)
{
if (PINB.۱==۰) // کلید اول
{
delay_ms(۲۵);
i++;
ادامه برنامه در ستون مقابل از بالا به پایین ⇒
```

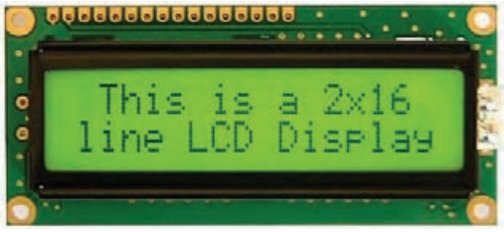
```
PORTD=i;
while (PINB.۱= = ۰);
delay_ms(۲۵);
}
if (PINB.۲==۰) // کلید دوم
{
delay_ms(۲۵);
i--;

PORTD=i;
while (PINB.۲= = ۰);
delay_ms(۲۵);
}
}}
```

فعالیت

در ساعت‌های غیردرسی، برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی کنید. نتیجه را در قالب یک گزارش ارائه دهید.

۳ـ۴ـ راه‌اندازی LCD

یکی دیگر از دستگاه‌های خروجی، LCD کاراکتری است. روی LCD می‌توان متن، اعداد و علائم را نمایش داد. در LCD کاراکتری پارامترهای مهم، تعداد خط و تعداد کاراکترها در هر خط است. نوع ۲×۱۶ آن کاربرد بیشتری در مقایسه با سایر LCDها دارد.



شکل ۵ـ۴ـ LCD کاراکتری ۲×۱۶

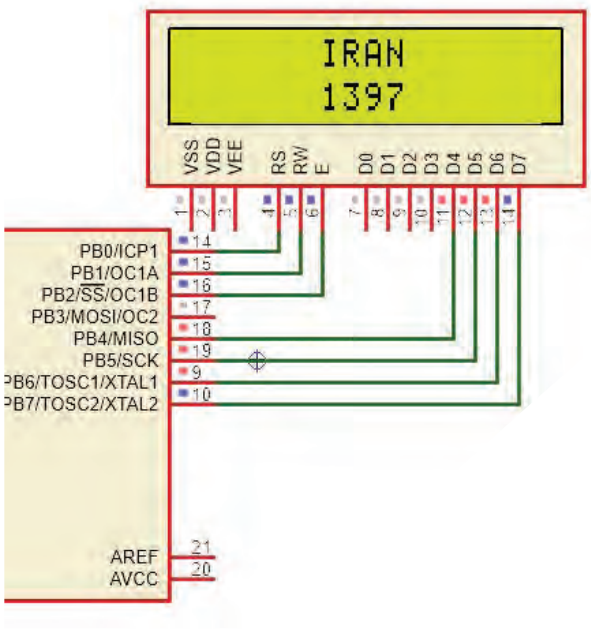
نکته

■ LCDهای کاراکتری را می‌توان با ۴ خط دیتا راه‌اندازی کرد (خطوط D۴ تا D۷). برای مثال در شکل۶ـ۴ طرز اتصال یک LCD به PORTB با ۴ خط دیتا نشان داده شده است.

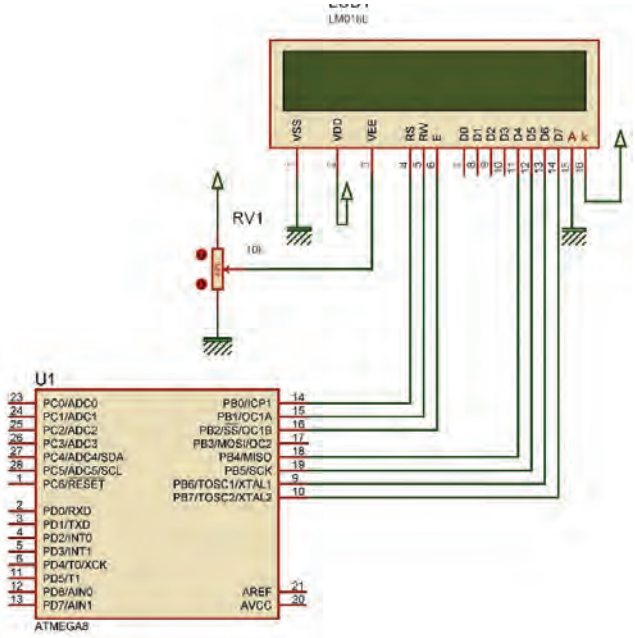
■ کد ویژن از این نوع اتصال پشتیبانی می‌کند.

مثال۳: یک LCD را مانند شکل ۶ـ۴ به پورت B متصل کنید.

مانند شکل ۷ـ۴ برنامه‌ای بنویسید که وسط خط اول کلمه IRAN و وسط خط دوم ۱۳۹۷ نوشته شود.



شکل ۷ـ۴



شکل ۶ـ۴ـ نحوه اتصال LCD به میکروکنترلر

■ **مراحل اجرای کار**

ابتدا در ویزارد برگه Alphanumeric LCD را باز و آن را فعال می‌کنیم.

□ در قسمت Character/Line مشخص می‌کنیم که در هر خط چند کاراکتر وجود دارد.

جدول ۴ـ۴

سطرهای LCD	ستون‌های LCD			
Y=۰	X=۰	X=۱	.....	X=۱۵
Y=۱			.....	

□ در قسمت Connections مانند شکل ۸ـ۴ چگونگی اتصال LCD به میکروکنترلر مشخص شده و قابل ویرایش است.

□ پس از تولید کد،کتابخانه <alcd.h> به برنامه اضافه می‌شود که توابع کار با LCD در این کتابخانه قرار دارد. این توابع عبارت‌اند از:

lcd_clear(); پاک کردن صفحه نمایش

lcd_gotoxy (x,y); مشخص کردن ستون و سطری که نوشتن از آنجا شروع می‌شود

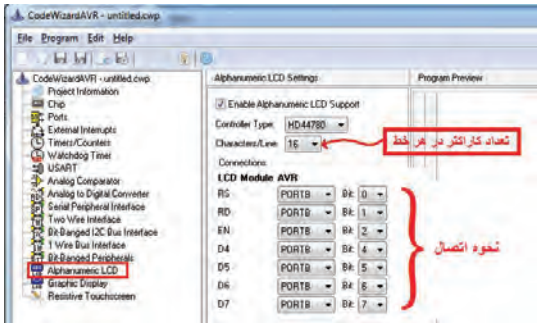
□ در جدول ۴ـ۴ شماره سطرها و ستون‌های LCD نشان داده شده است.

□ برای نوشتن متن روی LCD می‌توانید یکی از توابع موجود در جدول ۵ـ۴ را به‌کار ببرید.

□ کدهای مندرج در جدول ۶ـ۴ زیر را به کدهای تولید شده توسط ویزارد اضافه کنید.

مثال۴: برنامه‌ای بنویسید که کلمه IRAN را از ابتدای خط اول LCD تا انتهای خط اول حرکت دهد.

□ کدهای وارد شده در جدول ۷ـ۴ را به کدهای تولید شده توسط ویزارد، اضافه کنید.



شکل ۸ـ۴ـ تنظیم ویزارد برای اتصال LCD به میکروکنترلر

جدول ۵ـ۴ توابع نوشتن بر روی LCD همراه با مثال

مثال	تابع
lcd_putchar('A');	lcd_putchar('کاراکتر');
lcd_putsf("ARASH");	lcd_putsf("رشته");
char s[ ]="IRAN"; lcd_puts(s);	lcd_puts(متغیر رشته‌ای);

جدول ۶ـ۴

```
char s[ ]="IRAN";
void main(void)
{
lcd_init(۱۶);
lcd_clear();
lcd_gotoxy(۶,۰);
lcd_puts(s);
lcd_gotoxy(۶,۱);
lcd_putsf("۱۳۹۷");
while (۱)
{
}
}
```

جدول ۷ـ۴

```
#include <delay.h>
unsigned char x;
void main(void)
{
while (۱)
{
for(x=۰;x<۱۳;x++)
{
lcd_clear();
lcd_gotoxy(x,۰);
lcd_putsf("IRAN");
delay_ms (۳۰۰);
}
}
}
```

■ **کد اسکی** American Standard Code for Information Interchange: (ASCII)

☑ کدهای اسکی توسط انجمن استاندارد آمریکا برای تبادل اطلاعات در سیستم‌های کامپیوتری و مخابراتی به‌وجود آمد. این کدها به‌صورت استاندارد برای همه دستگاه‌های ارتباطی درنظر گرفته شده‌اند تا تمامی سیستم‌های مخابره اطلاعات از استاندارد یکسانی جهت ارتباط با یکدیگر تبعیت کنند.

☑ ASCII کاراکترهای مورد استفاده برای ارسال و دریافت اطلاعات را به‌صورت کدهایی تعریف می‌کند که برای همه به‌صورت یکسان از قبل تعریف شده است. مثلاً حرف A با کد اسکی ۶۵ به‌صورت یک مشخصه استاندارد تعریف شده است، حال اگر سیستمی این کد را دریافت و یا ارسال کند همواره این کد برای سامانه‌های مختلف به عنوان حرف A تلقی خواهد شد.

☑ در LCDهای کاراکتری نیز از کد ASCII استفاده می‌شود.در ادامه، کدهای ASCII برخی کاراکترهای ASCII را ملاحظه می‌کنید:

A =۶۵ a =۹۷ c =۴۰ + =۴۳ @ =۶۴ ? =۲۳

☑ در جدول ۸ـ۴ کدهای اسکی مربوط به‌کاراکترها را ملاحظه می‌کنید.

جدول ۸ ـ ۴ـ کدهای اسکی

Character	Binary Code	Character	Binary Code	Character	Binary Code
A	01000001	a	01100001	!	00100001
B	01000010	b	01100010	"	00100010
C	01000011	c	01100011	#	00100011
D	01000100	d	01100100	$	00100100
E	01000101	e	01100101	%	00100101
F	01000110	f	01100110	&	00100110
G	01000111	g	01100111	'	00100111
H	01001000	h	01101000	(	00101000
I	01001001	i	01101001	)	00101001
J	01001010	j	01101010	*	00101010
K	01001011	k	01101011	+	00101011
L	01001100	l	01101100	,	00101100
M	01001101	m	01101101	-	00101101
N	01001110	n	01101110	.	00101110
O	01001111	o	01101111	/	00101111
P	01010000	p	01110000	0	00110000
Q	01010001	q	01110001	1	00110001
R	01010010	r	01110010	2	00110010
S	01010011	s	01110011	3	00110011
T	01010100	t	01110100	4	00110100
U	01010101	u	01110101	5	00110101
V	01010110	v	01110110	6	00110110
W	01010111	w	01110111	7	00110111
X	01011000	x	01111000	8	00111000
Y	01011001	y	01111001	9	00111001
Z	01011010	z	01111010	?	00111111
				@	01000000

☑ برای ارسال اعداد به LCD لازم است که ابتدا آنها را توسط توابع موجود به کد ASCII تبدیل کرده و سپس ارسال نماییم. برای این کار توابعی وجود دارد که یکی از آنها تابع sprintf موجود در کتابخانه ی <stdio.h> است. ساختار این تابع مانند شکل ۹ـ۴ است.

```
sprintf ( متغیر رشته‌ای , "متن و نوع متغیر هاونحوه چاپ " , متغیر عددی)
```

شکل ۹ـ۴ـ ساختار تابع sprintf

☑ برای تعریف متغیر رشته‌ای طبق جدول ۹ـ۴ عمل کنید.

☑ برای مشخص کردن نوع و چگونگی چاپ اعداد روی LCD از علامت ٪ به همراه کاراکترهای خاص استفاده می‌شود. مثلاً از علامت ٪d یا ٪i برای مشخص کردن اعداد صحیح و از ٪f برای اعداد اعشاری استفاده می‌شود.

☑ برای مشخص کردن چگونگی چاپ، علاوه بر نوع متغیر، از اعداد نیز در کنار آنها استفاده می‌شود، مثلاً ٪۰۲d یعنی این که عدد به‌صورت دو رقمی چاپ شود حال اگر عدد یک رقمی باشد با قرار دادن یک ۰ قبل از عدد، آن را در دو رقم چاپ می‌کند (نمایش می‌دهد). برای مثال اگر مقدار متغیر ۳ باشد روی LCD عددورقمی ۰۳ دیده خواهد شد.

جدول ۹ـ۴

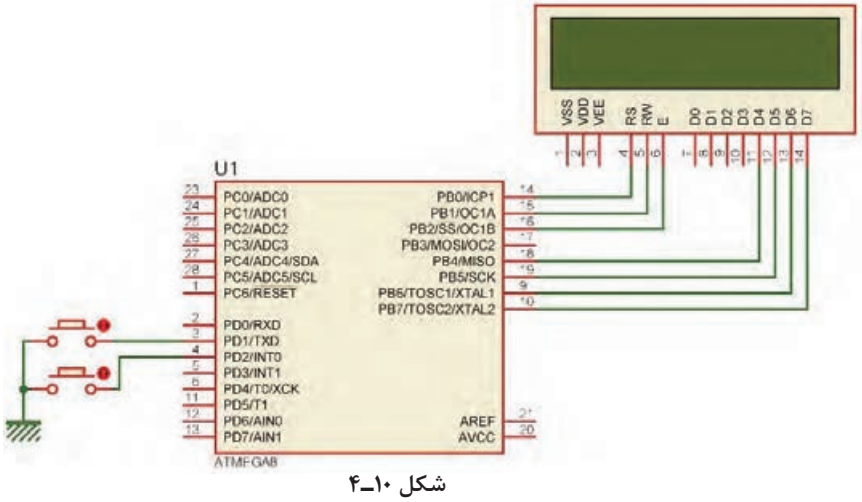
مثال	تعریف متغیر رشته‌ای
char s[۶];	char نام رشته [طول رشته];

جدول ۱۰ـ۴

```
i=۳;
char s[۶];
sprintf (s , "٪۰۲d" , i);
lcd_puts(s);
```

☑ در جدول ۱۰ـ۴ یک نمونه برنامه مربوط به چاپ یا نمایش اعداد روی LCD را ملاحظه می‌کنید.

مثال ۵: دو عدد کلید و یک LCD را مانند شکل ۱۰ـ۴ به میکروکنترلر متصل کنید، سپس برنامه‌ای بنویسید که با نگه داشتن یکی از کلیدها، عدد روی lcd زیاد و با کلید دیگر کاهش یابد و بازه تغییر عدد صفر تا ۲۰ باشد.



شکل ۱۰ـ۴

پاسخ: در جدول ۱۱ـ۴ برنامه مربوط به مثال ۵ آمده است.

جدول ۱۱ـ۴

```
#include <mega۸.h>
#include <alcd.h>
#include <delay.h>
#include <stdio.h>
unsigned char i=۰;
char s[۴];
void main(void)
{
lcd_init(۱۶);
lcd_gotoxy(۷,۰);
lcd_putsf("۰۰");
while (۱)
{

ادامه برنامه در ستون مقابل از بالا
به پایین ⇒
```

```
if(PIND.۱==۰ && i<۲۰) // اگر کلید اول زده‌شده و i کوچک‌تر از ۲۰ باشد
{
delay_ms(۲۵);
i++;
sprintf(s,"%۰۲d",i);
 lcd_gotoxy(۷,۰);
 lcd_puts(s);
 delay_ms(۳۰۰);
 }
 if(PIND.۲==۰ && i>۰) // اگر کلید دوم زده شده و i بزرگ‌تر از صفر باشد
 {
 delay_ms(۲۵);
 i - -;
 sprintf(s,"%۰۲d",i);
 lcd_gotoxy(۷,۰);
 lcd_puts(s);
 delay_ms(۳۰۰);
 }
 }
}
```

فعالیت

برنامه داده شده در جدول ۱۱ـ۴ را تحلیل کنید و فرایند اجرای آن را در قالب یک گزارش ارائه دهید.

مثال۶: با توجه به مدار شکل ۱۰ـ۴ یک ساعت بر روی LCD طراحی کنید.

جدول ۱۲ـ۴

```
#include <stdio.h>
#include <delay.h>
unsigned char h,m,s;
char time[۱۶];
void main(void)
{
lcd_clear();
while (۱)
{
s++;
if(s==۶۰)
{
ادامه برنامه در ستون مقابل از بالا
به پایین ⇒
```

```
 s=۰;
m++;
if(m==۶۰)
{
m=۰;
h++;
if(h==۲۴) h=۰;
}}
sprintf(time,"%۰۲d: %۰۲d: %۰۲d",h,m,s);
lcd_gotoxy(۴,۰);
lcd_puts(time);
delay_ms(۱۰۰۰);
}
}
```

■برنامه‌این مثال در جدول ۱۲ـ۴ آمده است.
■دراین برنامه با گذشت هر ثانیه به مقدار متغیر s که ثانیه را در خود نگه می دارد یک واحد اضافه می‌شود.
■هر گاه s=۶۰ شود شرط موجود مقدار s را صفر و به مقدار m که دقیقه را در خود نگه می‌دارد یک واحد اضافه می‌کند.
■هر گاه m=۶۰ شود، شرط موجود m را صفر و به مقدار h که ساعت را در خود نگه می‌دارد یک واحد اضافه می‌کند.
■هر گاه h=۲۴ شود این متغیر نیز صفر خواهد شد.
■در تابع sprintf هر سه مقدار ساعت، دقیقه و ثانیه به‌صورت دو رقمی همراه با جدا کننده ":" به‌صورت یک رشته در متغیر رشته‌ای time قرار می‌گیرد؛ سپس وسط خط اول چاپ می‌شود.
■دستور delay_ms (۱۰۰۰) نیز باعث می‌شود هر ثانیه یک بار متغیر s زیاد شود.

نکته

توجه داشته باشید که برنامه ساعت نوشته شده دقیق نیست و برای داشتن یک ساعت دقیق لازم است از واحد تایمر و یا RTC استفاده شود.

پژوهش

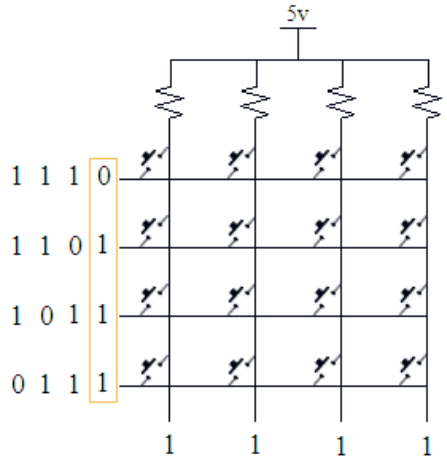
تحقیق کنید و دریابید که RTC چه مفهوم وچه کاربردهایی دارد.

فعالیت

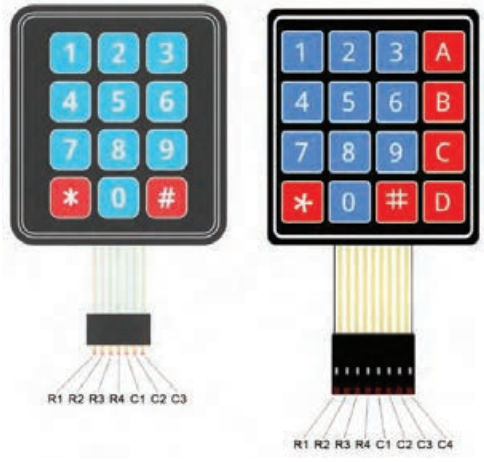
با توجه به مدار شکل ۱۰ـ۲ یک ساعت بر روی LCD طراحی کنید، که بتوان توسط دو کلید جداگانه، ساعت و دقیقه را تنظیم کرد.

۴ـ۴ـ صفحه کلید

■اگر در مداری به تعداد زیادی کلید نیاز داشته باشیم و بخواهیم هر یک را به یک پین از پورت‌ها متصل کنیم، تعداد زیادی از پورت‌ها اشغال می‌شود. برای رفع این مشکل از صفحه کلید که یکی دیگر از وسایل ورودی است استفاده می‌کنیم. در این دستگاه ورودی‌ها، کلیدها هستند که طبق شکل الف ـ ۱۱ـ۴ به‌صورت ماتریسی چیده می‌شوند. در شکل الف و ب ۱۱ـ۴ دو نمونه صفحه کلید با سوکت‌های آنها را ملاحظه می‌کنید.



شکل الف ۱۱ـ۴ـ مدار صفحه کلید



شکل ب ۱۱ـ۴ـ دو نمونه صفحه کلید

■روش خواندن صفحه کلید

ابتدا یک سطر را صفر و بقیه را یک می‌کنیم، سپس ستون‌ها را می‌خوانیم.
اگر همهٔ آنها یک باشند، یعنی در آن سطر کلیدی فعال نشده است.
آن سطر را یک و سطر بعدی را صفر می‌کنیم و دوباره ستون‌ها را می‌خوانیم. این کار را برای همهٔ سطرها انجام می‌دهیم.
اگر سطری را صفر کردیم و ستونی صفر شد با توجه به سطر و ستونی که صفر شده، کلید فعال مشخص می‌شود.
برای مثال اگر سطر دوم را صفر کنیم و ستون سوم صفر شود یعنی، کاربر کلید ۶ را زده است.

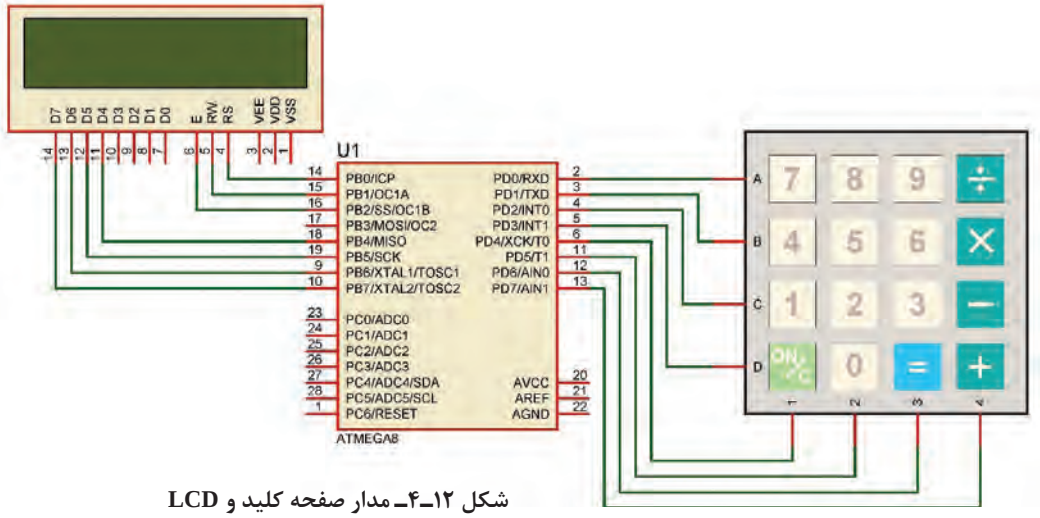
مثال ۷: یک صفحه کلید ۴×۴ و یک LCD را مطابق شکل ۱۲ـ۴ به میکروکنترلر متصل کنید. برنامه‌ای بنویسید که با فعال کردن هر کلید، عدد نوشته شده روی کلید در خروجی ظاهر شود. برای کلیدهای غیر عددی یعنی نمادهایی مانند جمع و تفریق، کدهای داده شده در جدول ۱۳ـ۴ روی نمایشگر به نمایش درآید، چاپ کنید.

جدول ۱۳ـ۴ نمایش کدهای عددی متناظر برای کاراکترها

÷	×	−	+	=	on/c
۱۰	۱۱	۱۲	۱۳	۱۴	۱۵

نکته

توجه داشته باشید که برای ستون‌ها از Pull_Up داخلی استفاده شده است.



شکل ۱۲ـ۴ـ مدار صفحه کلید و LCD

توجه

توجه: جهت خواناتر شدن (کاربر پسندتر شدن User Friend) برنامه، با استفاده از ماکرو برای سطر و ستون‌ها از نام‌های جدید استفاده کرده‌ایم. همچنین برنامه خواندن صفحه کلید را به‌صورت یک تابع نوشته‌ایم.

جدول ۱۴ـ۴ برنامه مربوط به مثال ۷ را نشان می‌دهد.

جدول ۱۴ـ۴

```
#include <mega۸.h>
#include <delay.h>
#include <stdio.h>
#include <alcd.h>
#define R۱ PORTD.۰
#define R۲ PORTD.۱
#define R۳ PORTD.۲
#define R۴ PORTD.۳
#define C۱ PIND.۴
#define C۲ PIND.۵
#define C۳ PIND.۶
#define C۴ PIND.۷
unsigned char kb(void);
unsigned char key;
char s[۴];
void main (void)
{
DDRD=۰x۰f;
lcd_init(۱۶);
while(۱)
{
 key=kb();
 if (key !=۱۶)
 {
 sprintf (s,"٪d",key);
 lcd_puts(s);
 }
}
}
//--------------- kb function ----------
unsigned char kb (void)
 {
⇒ ادامه برنامه در ستون مقابل از بالا به پایین
```

```
 unsigned char k=۱۶;
 PORTD=۰xFF;
 ////------------- ROW۱ //-------------
R۱=۰;
delay_ms(۳);
if (C۱==۰) {k=۷ ; while (C۱==۰);}
if (C۲==۰) {k=۸ ; while (C۲==۰);}
if (C۳==۰) {k=۹ ; while (C۳==۰);}
if (C۴==۰) {k=۱۰ ; while (C۴==۰);}
R۱=۱;
 ////------------- ROW۲ //-------------
R۲=۰;
delay_ms(۳);
if (C۱==۰) {k=۴ ; while (C۱==۰);}
if (C۲==۰) {k=۵ ; while (C۲==۰);}
if (C۳==۰) {k=۶ ; while (C۳==۰);}
if (C۴==۰) {k=۱۱ ; while (C۴==۰);}
R۲=۱;
 ////------------- ROW۳ //-------------
R۳=۰;
delay_ms(۳);
if (C۱==۰) {k=۱ ; while (C۱==۰);}
if (C۲==۰) {k=۲ ; while (C۲==۰);}
if (C۳==۰) {k=۳ ; while (C۳==۰);}
if (C۴==۰) {k=۱۲; while (C۴==۰);}
R۳=۱;
 //------------- ROW۴ //-------------
R۴=۰;
delay_ms(۳);
if (C۱==۰) {k=۱۵ ; while (C۱==۰);}
if (C۲==۰) {k=۰ ; while (C۲==۰);}
if (C۳==۰) {k=۱۴ ; while (C۳==۰);}
if (C۴==۰) {k=۱۳ ; while (C۴==۰);}
R۴=۱;
return k;
}
```

فعالیت

در ساعت‌های غیر درسی برنامه را در نرم‌افزار بارگذاری کنید و مدار را راه‌اندازی نمایید. نتیجه را در قالب یک گزارش ارائه دهید.

مثال ۸: در مثال ۷ برنامه را طوری تغییر دهید که با فعال کردن کلیدهای مربوط به علائم، به‌جای اعداد ۱۰ تا ۱۴ علامت مشخصه آن کلید مانند تقسیم روی صفحه نمایشگر ظاهر شود و با فعال کردن کلید ON/C صفحه LCD پاک شود.

پاسخ: در برنامه مثال ۷ در حلقه while تغییرات داده شده در جدول ۱۵ـ۴ را انجام دهید.

جدول ۱۵ـ۴

```
while(۱)
{
key=kb();
if (key < ۱۰)
{
sprintf (s,"٪d",key);
lcd_puts(s);
}
ادامه برنامه در ستون مقابل
⇒ از بالا به پایین
```

```
if (key==۱۰) lcd_putchar('/');
if (key==۱۱) lcd_putchar('*');
if (key==۱۲) lcd_putchar('-');
if (key==۱۳) lcd_putchar('+');
if (key==۱۴) lcd_putchar('=');
if (key==۱۵) lcd_clear();
}
```

فعالیت

برای مدار شکل ۱۳ـ۴ برنامه‌ای بنویسید که کاربر بتواند:

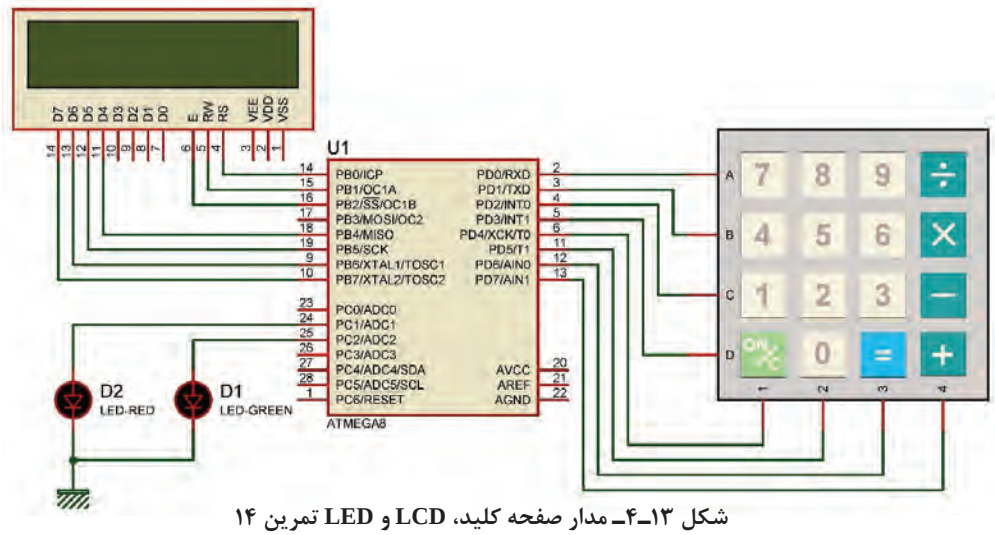
- یک رمز چهار رقمی را وارد کند.
- با فعال کردن کلید مساوی، عدد وارد شده با رمز تعیین شده مقایسه شود.
- اگر رمز صحیح بود LED سبز (متصل به PC.۲) و اگر رمز اشتباه بود LED قرمز (متصل به PC.۱) روشن و بعد از یک ثانیه خاموش شود.

راهنمایی: برای نوشتن برنامه قفل رمز لازم است موارد زیر را اجرا کنید:

- یک آرایه تعریف کنید.
- شماره کلیدهای انتخاب شده را در داخل آرایه ذخیره کنید.
- اگر کاربر کلید مساوی = را فشار داد، اعداد ذخیره شده در آرایه با ارقام رمز از پیش تعیین شده مقایسه شود.
- چنانچه عدد وارد شده با رمز برابر بود، LED۱ سبز رنگ و درغیر این صورت LED۲ قرمز رنگ روشن شود.

نکته

تعریف یک آرایه با نام d و تعداد ۱۰ خانه به‌صورت ;[۱۰]Char d بیان می‌شود.



شکل ۱۳ـ۴ـ مدار صفحه کلید، LCD و LED تمرین ۱۴

جدول ۱۶ـ۴

```
unsigned char i;
while(۱)
{
for(i=۰;i<۱۰;i++)
{
lcd_gotoxy(۷,۰);
lcd_puts(i);
delay_ms(۵۰۰);
}
}
```

الگوی پرسش

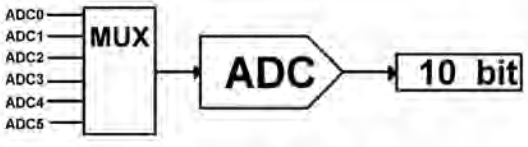
۱ـ اگر در برنامه داده شده در جدول ۱۶ـ۴ متغیر i از نوع unsigned char تعریف شده باشد، چه خطایی وجود دارد و برای تصحیح آن چه باید کرد. با مراجعه به جدول شرح دهید و جدول را اصلاح کنید.

۲ـ یک LCD و دو عدد کلید به میکروکنترلر متصل است، برنامه‌ای بنویسید که با فشار دادن کلیدها یک نمودار میله ای افقی بر روی LCD ترسیم شده و کوتاه و بلند شود.

راهنمایی: دستور lcd_putchar(۲۵۵) باعث چاپ یک مستطیل تو پر ■ بر روی LCD می‌شود.

۴ـ۵ـ ADC (Analog to Digital Converter)

■ تمام کمیت‌های اطراف ما مانند دما، فشار و رطوبت کمیت‌های آنالوگ هستند. برای اندازه‌گیری و پردازش آنها لازم است ابتدا آنها را به یک کمیت دیجیتال تبدیل و سپس پردازش شوند. در ATMEGA۸ یک مبدل ۶ کاناله و ۱۰ بیتی پیش‌بینی شده است، شکل ۴ـ۱۴. توسط Multiplexer یکی از ۶ کانال ورودی انتخاب و کمیت آنالوگ آن کانال به مبدل وارد و در نهایت به یک عدد ۱۰ یا ۸ بیتی تبدیل می‌شود.

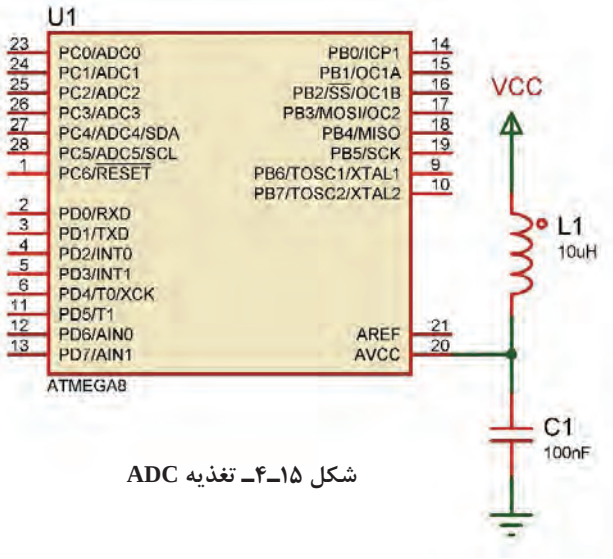


شکل ۱۴ـ۴ـ واحد ADC

■ **تغذیه ADC:** برای بالا بردن دقت ADC و کاهش تأثیر نویزهای احتمالی که ممکن است از ولتاژ تغذیه اصلی به بخش ADC وارد شود، تغذیه این قسمت را از تغذیه سایر مدارهای مرتبط با میکروکنترلر جدا می‌کنند. کاربر می‌تواند تغذیه این بخش را جداگانه تأمین کرده یا با استفاده از یک فیلتر LC مانند شکل ۴ـ۱۵ به ولتاژ اصلی متصل نماید. برای تغذیه میکروکنترلر و بخش ADC مطابق جدول ۴ـ۱۷ عمل کنید.

جدول ۴ـ۱۷ـ پایه‌های تغذیه اصلی و بخش ADC

عملکرد	شماره پایه
تغذیه اصلی VCC	۷
تغذیه اصلی GND	۸
تغذیه AVCC ADC	۲۰
تغذیه AGNG ADC	۲۲
ولتاژ مرجع خارجی Aref	۲۱



شکل ۱۵ـ۴ـ تغذیه ADC

■**ولتاژ مرجع:** در این نوع مبدل نیاز به یک ولتاژ مرجع داریم که می‌توانیم آن را از سه طریق تأمین کنیم:
☑ ولتاژ تغذیهٔ روی پایه AVCC علاوه‌بر تغذیه واحد ADC می‌تواند به عنوان ولتاژ مرجع نیز درنظر گرفته شود.
☑ ولتاژ روی پایه Aref، که می‌تواند بین ۰ تا ۵ ولت باشد و به عنوان ولتاژ مرجع خارجی درنظر گرفته شود.
☑ ولتاژ مرجع داخلی، یعنی ۲/۵۶ v ولت نیز می‌تواند ولتاژ مرجع باشد.

■**ضریب تفکیک:** پارامتری (مشخصه‌ای) است که مشخص می‌کند حساسیت یا دقت ADC چقدر است و از رابطه زیر محاسبه می‌شود.

$$\text{ضریب تفکیک} = \frac{V_{ref}}{۲^n - ۱}$$

در این رابطه n تعداد بیت خروجی مبدل است که می‌تواند با توجه به تنظیمات انجام شده در بخش ADC ۸ یا ۱۰ باشد.

■**عدد خروجی:** عدد خروجی مبدل را نیز می‌توان از رابطه زیر محاسبه کرد:

$$\text{عدد خروجی مبدل} = \frac{V_{in}}{\text{ضریب تفکیک}}$$

در این رابطه V_{in} ولتاژ ورودی به مبدل است. توجه داشته باشید که حداکثر ولتاژ ورودی به یک مبدل برابر با ولتاژ مرجع است.

مثال۹: اگر ولتاژ ورودی به مبدل، یک ولت و ولتاژ مرجع، از نوع داخلی و برابر ۲/۵۶ v باشد، مطلوب است:
الف) ضریب تفکیک
ب) عدد خروجی مبدل

$$\text{ضریب تفکیک} = \frac{۲/۵۶v}{۲۰^{۱۰} - ۱} = ۲/۵mv \qquad \text{عدد خروجی مبدل} = \frac{۱v}{۲/۵mv} = ۴۰۰$$

نکته

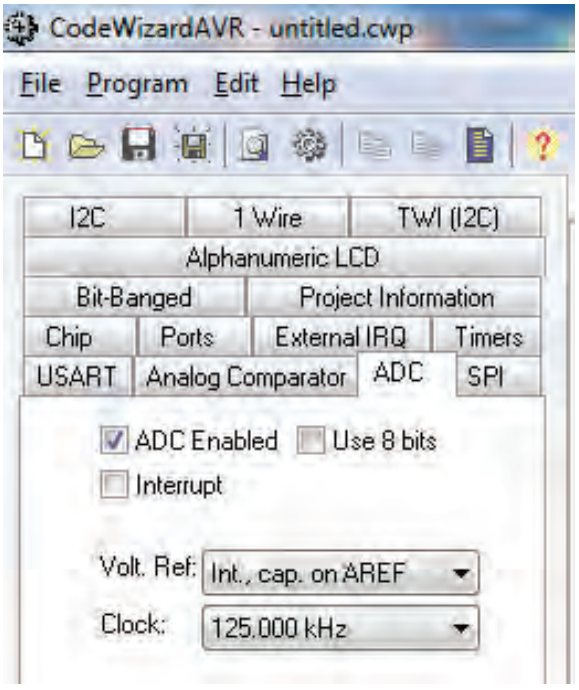
برای تنظیم ADC در ویزارد مطابق شکل ۱۶ـ۴، برگهٔ ADC را باز کنید و آن را فعال نمایید. در قسمت volt. ref می‌توانیم مشخص کنیم که ولتاژ مرجع از کدام منبع تأمین شود. در این مثال ولتاژ مرجع را داخلی و برابر با ۲/۵۶ v انتخاب کرده‌ایم.

نکته

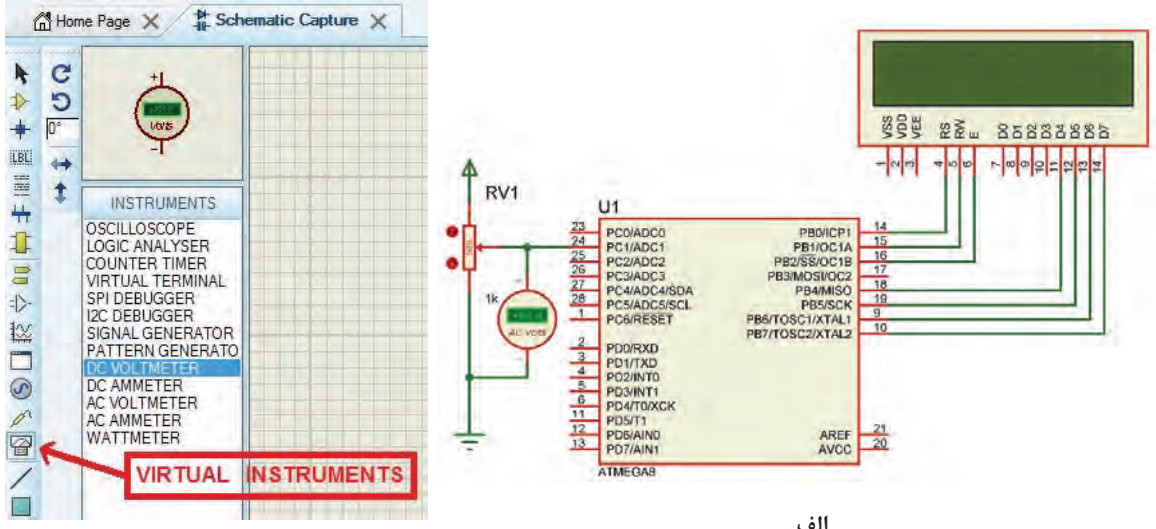
برای استفاده از مرجع داخلی لازم است، یک خازن در حد ۱uF، که طرف دیگر آن به زمین متصل است، بر روی پایه Aref قرار گیرد. در قسمت Clock فرکانس ورودی به قسمت ADC را مشخص می‌کنیم. این فرکانس باید بین ۵۰k تا ۲۰۰k هرتز باشد.

مثال۱۰: یک پتانسیومتر و یک ولت‌متر DC را مانند شکل ۱۶ـ۴ الف به کانال ADC۱ متصل کنید، برنامه‌ای بنویسید که عدد خروجی مبدل بر روی LCD نمایش داده شود. با توجه به روابط مربوط به ضریب تفکیک و عدد خروجی، محاسبات را برای ولتاژ ورودی به‌دست آورید و نتایج را با اعداد نمایش داده شده بر روی LCD مقایسه کنید.

راهنمایی: در نرم‌افزار Proteus مطابق شکل الف و ب ۱۷ـ۴ می‌توانید پتانسیومتر را با تایپ عبارت POT_HG در قسمت Keywords و ولت‌متر را در صفحه اصلی از بخش VIRTUAL INSTRUMENT بردارید.



شکل ۱۶ـ۴ـ تنظیم ADC در ویزارد



ب الف

شکل ۱۷ـ۴

■ پس از تولید کد ملاحظه می‌شود که تابع (read_adc) به برنامه اضافه شده، ورودی این تابع شماره کانال و خروجی آن که از نوع unsigned int می باشد، عدد خروجی مبدل ADC را نشان می‌دهد.

```
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
```

بنابراین نوع متغیری که برای دریافت عدد خروجی تعریف می‌شود را unsigned int انتخاب می‌کنیم. اگر کدهای نوشته شده در جدول ۱۸ـ۴ را در محل مناسب به برنامه اضافه کنید می‌توانید نتیجه را بر روی lcd ببینید.

جدول ۱۸ـ۴

```
#include <stdio.h>
unsigned int a;
char s[۶];
void main(void)
{
while (۱)
{
a=read_adc(۱);
sprintf (s,"٪۰۴d",a);
lcd_gotoxy(۰,۰);
lcd_puts(s);
delay_ms(۱۰۰);
}}
```

فعالیت

با توجه به مدار شکل ۱۸ـ۴ برنامه را طوری تغییر دهید که یک ولت‌متر داشته باشیم.

مثال ۱۱: کار با سنسور دمای LM۳۵

■در کتاب مونتاژ و دمونتاژ پایه یازدهم با این سنسور آشنا شدید. حساسیت آن ۱۰mV/°C است. یعنی اگر دمای محیط ۱°C باشد خروجی آن ۱۰ میلی‌ولت خواهد بود.

■می‌خواهیم عدد خروجی مبدل ADC را با ولتاژ مرجع داخلی ۲/۵۶v محاسبه کنیم و برنامه‌ای بنویسیم که دمای محیط را بر روی LCD نمایش دهد.

■همچنین یک موتور FAN را که به میکروکنترلر متصل است در دمای ۳۰ درجه سانتی‌گراد روشن و در دمای ۲۵ درجه سانتی‌گراد خاموش کند. (چنانچه در محیطی که این آزمایش را انجام می دهید این دو دما قابل دسترس نیست می‌توانید آنها را تغییر دهید).

پاسخ:

☑ موارد مطابق شکل ۱۸ـ۴ است. در شکل ۱۹ـ۴ پایه‌های سنسور LM۳۵ وترانزیستور BDX۵۳C آمده است.

☑ مقدار عدد خروجی و ضریب تفکیک را محاسبه می‌کنیم.

$$\text{ضریب تفکیک} = \frac{۲/۵۶v}{۱۰۲۴} = ۲/۵mv$$

$$\text{عدد خروجی} = \frac{۱۰mv}{۲/۵mv} = ۴$$

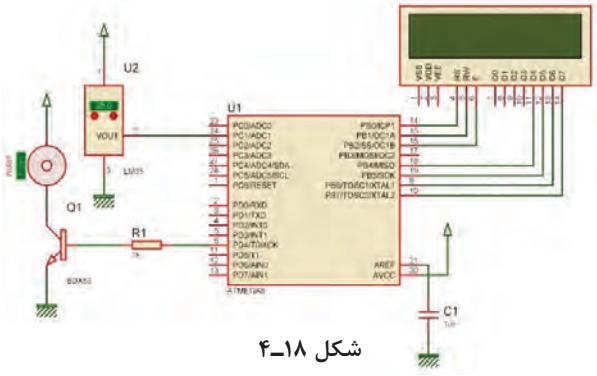
☑ با توجه به این که به ازای دمای ۱°C عدد خروجی مبدل ۴ است پس برای نمایش دما روی LCD باید عدد خروجی مبدل را بر ۴ تقسیم کنیم.

☑ کدهای داده شده در جدول ۱۹ـ۴ را به کدهای تولید شده توسط ویزارد اضافه کنید.

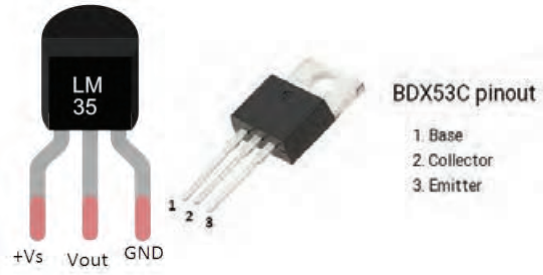
جدول ۱۹ـ۴

```
#include <stdio.h>
unsigned int a;
unsigned char t;
char s[۱۶];
void main(void)
{
while (۱)
{
a=read_adc(۱);
t=a/۴;
sprintf (s,"Temp=٪۰۲d",t);
lcd_gotoxy(۰,۰);
lcd_puts(s);
if(t>=۳۰)
{
ادامه برنامه در ستون مقابل از بالا به پایین ⇒
```

```
PORTD.۴=۱;
lcd_gotoxy(۰,۱);
 lcd_putsf("MOTOR  ON
");
}
if(t<=۲۵)
{
PORTD.۴=۰;
lcd_gotoxy(۰,۱);
 lcd_putsf("MOTOR  OFF
");
}
delay_ms(۱۰۰);
}
}
```



شکل ۱۸ـ۴

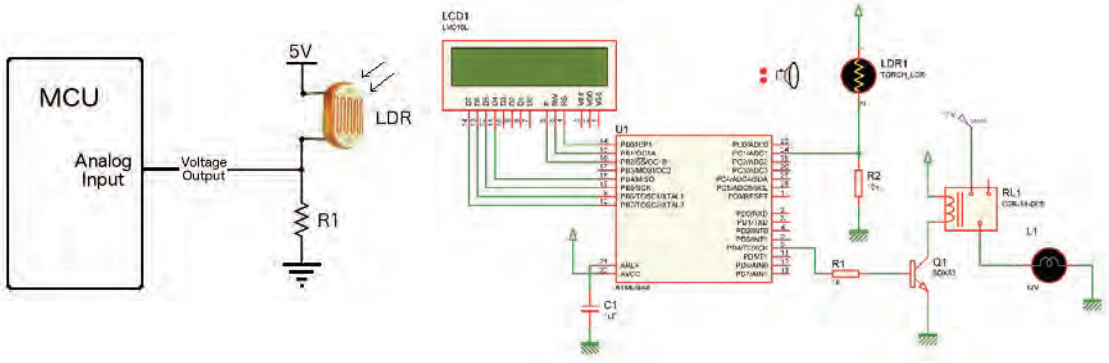


شکل ۱۹ـ۴ـ پایه‌های سنسور LM۳۵ و BDX۵۳

مثال ۱۲: کار با LDR

□ LDR (light dependent resistor) یک مقاومت متغیر با تغییر شدت نور است. مقاومت این قطعه در تاریکی حدود ۱۰۰kΩ و در نور شدید در حد ۱۰۰Ω است.

■می‌خواهیم با توجه به مدار شکل ۲۰ـ۴ برنامه‌ای بنویسیم که با کم شدن نور محیط، لامپ متصل به رله روشن و با زیاد شدن نور محیط، لامپ خاموش شود.



شکل ۲۰ـ۴

■در این پروژه نیز از مبدل آنالوگ به دیجیتال استفاده شده است. LDR با یک مقاومت سری شده، در نتیجه با کم و زیاد شدن نور محیط، مقاومت آن و در نتیجه ولتاژ ورودی به ADC تغییر می‌کند.

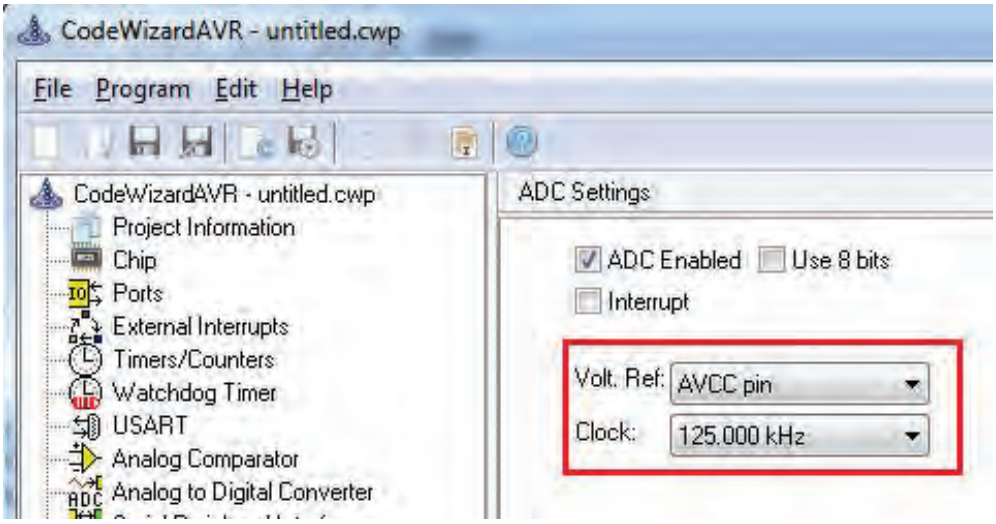
■برای کنترل لامپ، عدد خروجی ADC را خوانده و روی LCD نمایش می‌دهیم.

■با کم و زیاد کردن نور محیط و خواندن عدد روی LCD، تصمیم می‌گیریم که بر روی چه عددی لامپ روشن و یا خاموش شود.

■توجه داشته باشید که دو عدد انتخابی باید از هم فاصله مناسبی داشته باشند تا تغییر جزیی نور محیط، لامپ را به‌صورت پی‌درپی روشن و خاموش نکند.

توجه

مطابق شکل ۲۱ـ۴ ولتاژ مرجع را AVCC درنظر بگیرید. کدهای داده شده در جدول ۲۰ـ۴ را به کدهای تولید شده توسط ویزارد اضافه کنید.



شکل ۲۱ـ۴ـ تنظیم ولتاژ مرجع

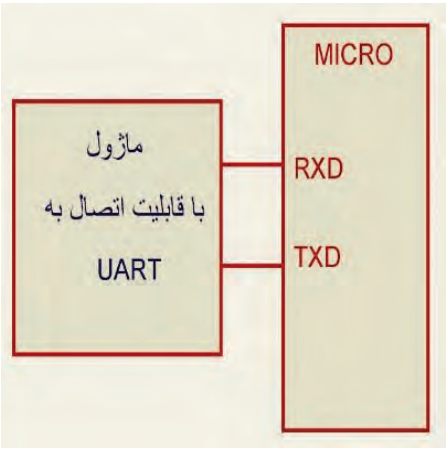
جدول ۲۰ـ۴

```
#include <stdio.h>
unsigned int a;
char s[۱۶];
void main(void)
{
while (۱)
{
a=read_adc(۱);
sprintf (s,"Light=٪۰۴d",a);
lcd_gotoxy(۰,۰);
lcd_puts(s);
ادامه برنامه در ستون مقابل از بالا
به پایین ⇒
```

```
if(a>=۵۵۰)
{
 PORTD.۴=۱;
 Lcd_gotoxy(۰,۱);
 lcd_putsf("LAMP ON ");
}
if(a<=۵۰۰)
{
 PORTD.۴=۰;

 Lcd_gotoxy(۰,۱);
lcd_putsf("LAMP OFF ");
}
delay_ms(۱۰۰);}}
```

تحقیق کنید

یکی از درگاه‌های بسیار پرکاربرد در میکروکنترلرها UART است.درباره چگونگی اتصال سخت‌افزاری ماژول‌های آماده شکل ۲۲ـ۴ از قبیل حسگر اثر انگشت، کارت‌خوان‌های RFID، GPS و GSM به میکروکنترلر تحقیق کنید و نتیجه را در قالب یک گزارش به کلاس ارائه دهید.



شکل ۲۲ـ۴ـ اتصال ماژول به میکروکنترلر

الگوی آزمون نرم‌افزاری واحد یادگیری ۴

مشابه یکی از مثال‌ها یا فعالیت‌های داده شده در متن واحد یادگیری را به‌صورت نرم‌افزاری اجرا کنید.

الگوی آزمون نظری واحد یادگیری ۴

۱ـ دستور<mega۸.h > include # در برنامه، بارگراف کتابخانه میکرو را معرفی می‌کند.

درست ☐ نادرست ☐

۲ـ دستور مربوط به شرط حلقه for و دستور تأخیر را بنویسید.

۳ـ LCD کاراکتری ۱۶×۲ نسبت به سایر LCDها پرکاربردتر است.

درست ☐ نادرست ☐

۴ـ برای اتصال LCD کاراکتری به میکروکنترلر، باید پایه‌های دیتای ..... تا ..... متصل شود.

۵ـ برای نوشتن کلمات در LCD با استفاده از میکروکنترلر حتماً باید از نرم‌افزاری مانند ویزارد استفاده کنیم.

درست ☐ نادرست ☐

۶ـ در مثال ;[۶] chars حرف s و عدد ۶ چه مفهومی دارد؟

۷ـ کد اسکی یک کد ..... که در ..... به‌کار می‌رود و در سطح جهان کاربرد یکسانی دارد.

۸ـ خروجی ۸ بیت یا ۱۰ بیت در ADC چه تفاوتی با هم دارد وروی چه مشخصه‌ای تأثیر می‌گذارد؟

ارزشیابی واحد یادگیری ۴: کسب شایستگی در برنامه‌نویسی به زبان C

شرح کار:
۱ـ تحلیل برنامه‌های نوشته شده به زبان C برای شمارنده ۲ـ تحلیل برنامه‌های نوشته شده به زبان C برای LCD ۳ـ تحلیل برنامه‌های نوشته شده به زبان C برای کنترل دما ونور ۴ـ اجرای انواع برنامه‌های کنترلی با نرم‌افزار

استاندارد عملکرد: تحلیل انواع برنامه‌های آماده به زبان C برای مدارهای کنترلی ساده با استفاده از خروجی‌های LCD و LED و ورودی‌های حسگر دما، رطوبت و نور

شاخص‌ها:
انتخاب فضای مورد نظر و مناسب بودن آن (میز کار)
تحلیل برنامه نوشته شده به زبان C برای شمارنده یا LCD (۳۰ دقیقه)
تحلیل برنامه نوشته شده به زبان C کنترل دما،رطوبت و نور (۳۰ دقیقه)
اجرای نرم‌افزاری انواع برنامه‌ها (۴۰ دقیقه)

شرایط انجام کار و ابزار و تجهیزات: شرایط انجام کار مشابه بقیه واحدهای یادگیری

معیار شایستگی:

ردیف	مراحل کار	حداقل نمره قبولی از ۳	نمره هنرجو
۱	تحلیل انواع برنامه‌های آماده ساده	۱	
۲	تحلیل انواع برنامه‌ها برای شمارنده با LCD	۲	
۳	تحلیل انواع برنامه‌ها برای کنترل دما،نور و رطوبت	۲	
	شایستگی‌های غیرفنی، ایمنی، بهداشت، توجهات زیست محیطی و نگرش: ۱ـ رعایت نکات ایمنی دستگاه‌ها ۲ـ دقت و تمرکز در اجرای کار ۳ـ شایستگی تفکر و یادگیری مادام‌العمر ۴ـ اخلاق حرفه‌ای	۲	
	میانگین نمرات		*

* حداقل میانگین نمرات هنرجو برای قبولی و کسب شایستگی، ۲ می‌باشد.