دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر

دستورکار آزمایشگاه ریزپردازنده
دستورکار آزمایشگاه طراحی سیستم‌های دیجیتال2

(مبتنی بر ریزپردازنده ATMEGA16/32)

تهیه کننده:
زهرا محمدزاده

بررسی کننده:
دکتر امیر خورسندی

شهریور 1401

2 جلسه دوم

زیربرنامه‌نویسی و ایجاد فایل‌های جانبی

2.1 هدف

در این جلسه، نحوه زیربرنامه‌نویسی به همراه آرگومان‌های ورودی/خروجی و هم‌چنین ایجاد فایل‌های جانبی شامل فایل‌های پیاده‌سازی زیربرنامه با قالب C. و فایل‌های سرآیند زیربرنامه‌ها با قالب h. بررسی می‌گردند.

2.2 مقدمه

امروزه در اکثر پروژه‌های عملیاتی به منظور افزایش قابلیت انعطاف سیستم و رفع نیازهای متنوع کاربران، برنامه-های پیچیده و با حجم زیادی از کد به کار گرفته می‌شوند که عموماً به روش کار گروهی توسعه می‌یابند. لذا منطقی به نظر می‌رسد که جهت توسعه این برنامه‌ها راهکاری دنبال شود تا ضمن تسهیل انجام کار گروهی، با ایجاد ساختار و نظم مناسب در کدهای نوشته شده، از یک سو خوانایی (Readability) و قابلیت ردیابی (Tracability) این کدها افزایش یابد و از سوی دیگر امکان استفاده بخش‌هایی از برنامه‌های قبلی در پروژه‌های جدید فراهم گردد (Reusability).

راه‌حل مناسب برای این منظور، کدنویسی ساختار یافته و استفاده از زیربرنامه‌های مجزا برای پیاده‌سازی بخش‌های مختلف کد می‌باشد. به عبارت دقیق‌تر بخش‌هایی از کد که هدف مشخصی دارند و احتمالاً در جاهای مختلف، استفاده از آن‌ها تکرار می‌گردد، در قالب یک زیربرنامه مجزا توسعه داده شده و هر جا که نیاز به آن‌ها هست این زیربرنامه‌ها فراخوانی می‌شوند. هم چنین می‌توان برای هر زیربرنامه تعدادی آرگومان ورودی و خروجی در نظر گرفت تا پویایی و قابلیت انعطاف لازم برای به‌کارگیری آن در شرایط گوناگون فراهم گردد.

برای جلوگیری از افزایش حجم کد در برنامه اصلی و نیز ایجاد قابلیت استفاده مجدد از کد زیربرنامه‌ها در پروژه‌های مختلف، عموماً زیربرنامه‌های توسعه داده شده در فایل‌های جانبی ذخیره می‌گردند. در هر مجموعه فایل جانبی دسته‌ای از زیربرنامه‌ها برای یک کاربرد خاص جمع‌آوری و به صورت یک کتابخانه قابل استفاده خواهند بود. به صورت کلی هر مجموعه فایل جانبی، شامل دو فایل با قالب‌های c. و h. است. فایل c. حاوی بدنه اصلی زیربرنامه‌ها می‌باشد و فایل h. سرایند زیربرنامه های تعریف شده در فایل c. و برخی تعاریف اولیه را در بر می‌گیرد.

2.3 ایجاد زیربرنامه با آرگومان ورودی و خروجی

برنامه‌ای ساده را در نظر بگیرید که در آن ریزپردازنده باید داده‌ها را از یکی از درگاه‌های ورودی بخواند و همان داده را روی یک درگاه خروجی نمایش دهد. برنامه 0-1 داده ورودی را از درگاه A خوانده و بر روی درگاه B نمایش می‌دهد.

برنامه 0-1

```
#include <mega16.h>

void main(void)
{
   DDRA=0x00;    // as input
   DDRB=0xFF; // as output

 while (1)
  {
    PORTB=PINA;
   }
}
```

اکنون می‌خواهیم همین کار را با استفاده از زیربرنامه انجام دهیم. برای این منظور برنامه 0-1 را به صورت برنامه 0-2 اصلاح می‌کنیم.

برنامه 0-2

```
#include <mega16.h>
/*************************************/
void portfun(void)
{
DDRA=0x00;    // as input
DDRB=0xFF; // as output
   PORTB=PINA;
}
/* ************************************/
void main(void)
{
while (1){
Portfun();
}
}
```

در مرحله‌ی بعدی، برای این که در برنامه انعطاف بیشتری ایجاد شود و بتوان داده را از هر درگاه دلخواهی بخواند و بر روی یک درگاه دلخواه دیگر نمایش دهد، لازم است درگاه‌های مورد نظر به صورت آرگومان‌های ورودی به تابع معرفی گردند.

```
#include <mega16.h>
#define port_A  1
#define port_B  2
#define port_C  3
#define port_D  4
/*********************************/
```

برنامه 0-3

```
void portfun(unsigned int port_in,unsigned int port_out)
{
char data_in;

switch( port_in)
{
  case port_A:
    DDRA=0x00;    // as input
    data_in=PINA;
  break ;
  case port_B:
    DDRB=0x00;    // as input
    data_in=PINB;
  break;
  case port_C:
    DDRC=0x00;    // as input
    data_in=PINC;
  break;
  case port_D:
    DDRD=0x00;    // as input
    data_in=PIND;
  break;
}
switch( port_out)
{
  case port_A:
    DDRA=0xFF;    // as output
    PORTA=data_in;
  break ;
  case port_B:
    DDRB=0xFF;    // as output
    PORTB=data_in;
  break;
  case port_C:
    DDRC=0xFF;    // as output
    PORTC=data_in;
  break;
  case port_D:
    DDRD=0xFF;    // as output
    PORTD=data_in;
  break;
}
}
/*   /*********************************
void main(void)
{
while(1){
portfun(port_A,port_B);
}

}
```

برای ایجاد آرگومان خروجی نیز می‌توان تغییراتی را مانند برنامه 0-4 اعمال کرد.

برنامه 4-0

```
#include <mega16.h>
#define port_A  1
#define port_B  2
#define port_C  3
#define port_D  4
/*********************************/
unsigned int  portfun(unsigned int port_in,unsigned int
port_out)
{
char data_in;

switch( port_in)
{
  case port_A:
    DDRA=0x00;    // as input
    data_in=PINA;
  break;
  case port_B:
    DDRB=0x00;    // as input
    data_in=PINB;
  break;
  case port_C:
    DDRC=0x00;    // as input
    data_in=PINC;
  break;
  case port_D:
    DDRD=0x00;    // as input
    data_in=PIND;
  break;
}
switch( port_out)
{
  case port_A:
    DDRA=0xFF;    // as output
    PORTA=data_in;
  break;
  case port_B:
    DDRB=0xFF;    // as output
    PORTB=data_in;
  break;
  case port_C:
    DDRC=0xFF;    // as output
    PORTC=data_in;
  break;
  case port_D:
    DDRD=0xFF;    // as output
    PORTD=data_in;
  break;
}
return 1; // can be any data with unsigned int  format
}
/*   /**********************************
void main(void)
{
char data_out;
```

```
while(1){
data_out=portfun(port_A,port_B);
}
 }
```

2.4 ایجاد فایل جانبی با قالب c.

برای ایجاد یک فایل جانبی با قالب c. می‌توان از طریق منوی File/New/ Source File اقدام کرد. با انجام این کار فایل جدیدی ایجاد گردیده که می‌توان بدنه اصلی زیربرنامه‌های مورد نظر را درون آن پیاده‌سازی نمود. در این فایل مطابق برنامه 0-5 کلیه‌ی تعاریف و فایل‌های سرآیند مورد نیاز در همان ابتدای فایل معرفی می‌گردند.

برنامه 0-5

```
#include <mega16.h>

#define port_A  1
#define port_B  2
#define port_C  3
#define port_D  4

unsigned int  portfun(unsigned int port_in,unsigned int port_out)
{
……
……
}
```

اکنون فایل جدید را با استفاده از منوی پیکربندی پروژه (Project/Configure)، پنجره‌ی Configuration Project<project_ name>.prj، سربرگ Files/input files و دکمه add به پروژه اضافه می‌کنیم.

2.5 ایجاد فایل جانبی با قالب h.

این کار از طریق منوی File/New/ Source File قابل انجام است که در نتیجه آن فایل جدیدی ایجاد گردیده و با قالب h. ذخیره می‌شود. بهتر است فایل‌های ایجاد شده دارای نام یکسانی با فایل نظرشان در قالب c. بوده و هر دو در مسیر پروژه ذخیره شده باشند.

در این فایل مطابق برنامه 0-6 سرایند زیربرنامه های پیاده‌سازی شده در فایل جانبی با قالب c. آورده می‌شود.

برنامه 0-6

```
#ifndef _portio_INCLUDED_
#define _portio_INCLUDED_

unsigned int  portfun(unsigned int port_in,unsigned int port_out);

#endif
```

بعد از ایجاد فایل‌های کمکی، همانند برنامه 0-7 در ابتدای فایل اصلی و پیش از تعریف تابع main، فایل سرآیند ایجاد شده فراخوانی می‌گردد و در ادامه می‌توان از زیربرنامه‌های تعریف شده در آن استفاده نمود. همان گونه که مشاهده می‌شود، با این کار حجم فایل اصلی به صورت قابل توجهی کاهش پیدا کرده است.

برنامه 0-7

```
#include <mega16.h>
#include <portio.h>

#define port_A  1
#define port_B  2
#define port_C  3
#define port_D  4

void main(void)
{
char data_out;
while(1){
data_out=portfun(port_A,port_B);
}
 }
```

2.6 ایجاد فایل سرآیند برای کل پروژه

تا این جا نحوه ایجاد فایل‌های سرایند برای کاهش حجم برنامه اصلی و ایجاد قابلیت استفاده از کد نشان داده شد. اما برخی تعاریف اولیه مورد نیاز زیربرنامه‌ها هستند که به صورت همزمان در فایل‌های حانبی مختلف و نیز فایل اصلی پروژه مورد نیاز هستند. مثلاً تعاریف انجام شده در ابتدای فایل c. مربوط به بخش2.4 هم در فایل جانبی مورد استفاده هستند و هم برای فراخوانی زیربرنامه در فایل اصلی مورد نیاز می‌باشند. پس ناگزیر به اضافه کردن محدد آن‌ها به فایل اصلی هستیم. بدیهی است که با افزایش حجم این تعاریف، میزان نظم و بهینه بودن کدهای پروژه کاهش می‌یابد. لذا برای حل این مشکل از فایل جداگانه ای با قالب h. که حاوی تمام فراخوانی‌ها و تعاریف اولیه است استفاده می‌شود.

در برنامه 0-8 محتوای فایل سرآیند مذکور مشاهده می‌شود. این فایل در ابتدای فایل اصلی و تمام فایل‌های جانبی که به آن نیاز دارند، فراخوانی می‌شود.

برنامه 0-8

```
#ifndef _header_INCLUDED_
#define _header_INCLUDED_

 #include <mega16.h>
#include <portio.h>

#define port_A  1
#define port_B  2
#define port_C  3
#define port_D  4

#endif
```

در برنامه 0-9 تغییرات مربوط به فایل‌های اصلی و جانبی آورده شده است.

فایل جانبی

برنامه 0-9

```
#include <header.h>

unsigned int  portfun(unsigned int port_in,unsigned int
port_out)
```

```
{
..
...
}
```

فایل اصلی

```
#include <header.h>

void main(void)
{
…
}
```

2.7 تعریف متغیر سراسری

در یک پروژه گاهی به تعدادی متغیر سراسری نیاز است که بتوان از آن در تعدادی از زیربرنامه‌ها استفاده نمود. بدین منظور متغیر عمومی مورد نظر در فایل اصلی و قبل از تابع main مانند برنامه 2-10 تعریف می‌گردد.[1]

برنامه 0-10

```
#include <header.h>

int data=120;   //global variable

void main(void)
{
…}
```

همچنین این متغیر data را می‌توان در فایل سرآیند بدون مقداردهی اولیه و با افزودن عبارت «extern» مانند برنامه 0-11 تعریف نمود.

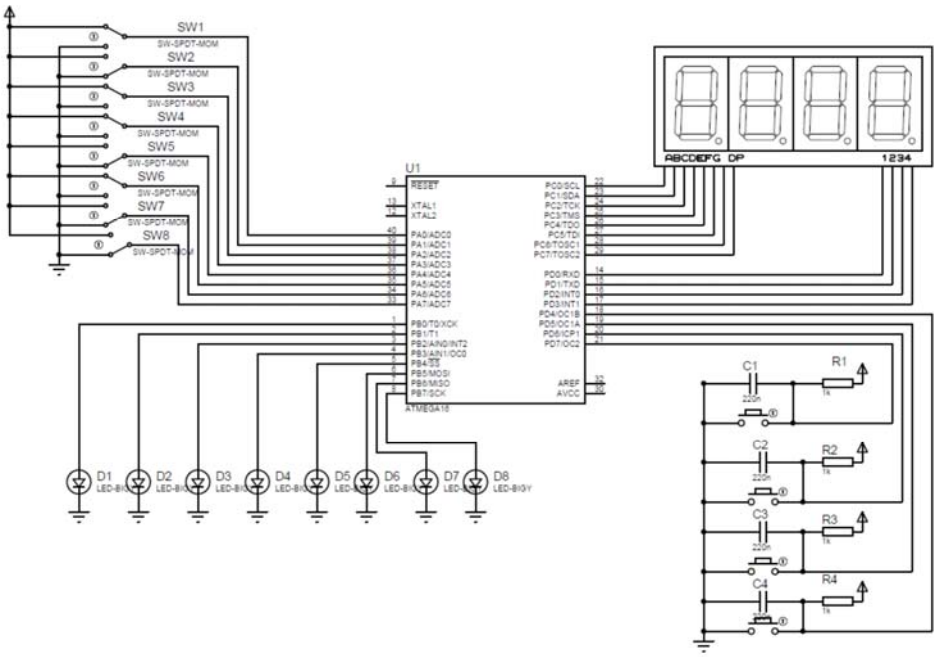
برنامه 0-11

```
#ifndef _header_INCLUDED_
#define _header_INCLUDED_

 #include <mega16.h>
#include <portio.h>

#define port_A  1
#define port_B  2
#define port_C  3
#define port_D  4

extern int data;

#endif
```

[1] استفاده از متغیرهای سراسری در برنامه‌نویسی کامپیوتر چندان توصیه نمی‌شود. اما در ریزپردازنده با توجه به محدود بودن حجم و ساده‌تر بودن کد و هم‌چنین محدودیت‌های سخت‌افزاری این کار با حساسیت کمتری صورت می‌پذیرد.

2.8 برنامه‌های اجرایی مربوط به زیربرنامه‌نویسی و فایل‌های جانبی

برای سخت‌افزار نشان داده شده در شکل 0-1 برنامه‌های زیر را بنویسید.



شکل 0-1

1) زیربرنامه‌ای بنویسید که داده‌ای را از یکی از درگاه‌ها بخواند. در این زیربرنامه باید درگاه مورد نظر به عنوان آرگومان ورودی و داده خوانده شده به عنوان آرگومان خروجی در نظر گرفته شود.
2) زیربرنامه‌ای بنویسید که داده‌ای را بر روی یکی از درگاه‌ها بنویسد. در این زیربرنامه درگاه مورد نظر و داده، هر دو به عنوان آرگومان ورودی در نظر گرفته می‌شوند.
3) با استفاده از نتایج بند یک و دو، زیربرنامه‌ای بنویسید که با استفاده از آن همه‌ی LEDها خاموش و روشن شوند. دفعات و فاصله زمانی بین خاموش روشن شدن به عنوان آرگومان ورودی در نظر گرفته شود.
4) با استفاده از نتایج بند یک و دو زیربرنامه‌ای بنویسید که مقدار تعیین شده توسط سوییچ‌ها را بر رویLED ها نمایش دهد.

5) با استفاده از نتایج بند یک و دو، زیربرنامه‌ای بنویسید که عددی 4 رقمی را روی 7-segment نشان دهد. عدد مورد نظر و درگاه متصل به خطوط Enable و داده‌ی 7-segment به صورت آرگومان ورودی دریافت شوند. (فرض نمایید همواره خطوط Enable به چهار بیت پایین درگاه مورد نظر متصل است.)

6) تمام زیر برنامه‌های توسعه داده شده را در یک فایل جداگانه به پروژه اضافه نمایید و همه‌ی بندها به ترتیب اجرا شوند.