یا لطیف

دانشکده مهندسی برق

تمرین های شبیه سازی ریز پردازنده

تمرین سری ۱

تهیه کننده و نویسنده:

رضا آدینه پور

استاد مربوطه:

جناب اقای دکتر حسین خسروی

تاریخ تهیه و ارائه:

مهر ماه ۱۴۰۰

۱) **برنامه ای به زبان اسمبلی بنویسید که:**

الف) جمع ۸ بیتی دو عد 0xF0 و 0xA5 را حساب کرده و نتیجه را در R20 قرار دهد.

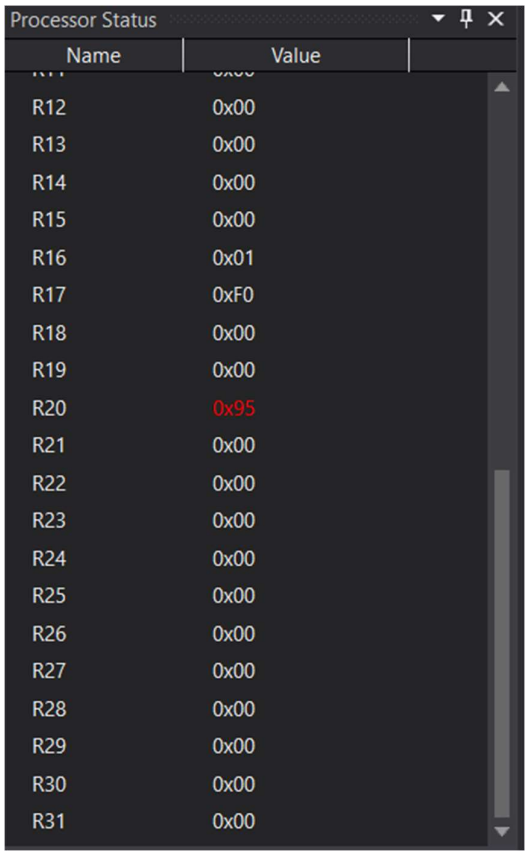
ب) سپس جمع ۱۶ بیتی دو عدد 0x20CA و 0x4BF8 را حساب کرده و نتیجه را در آدرسهای 0x341 (بایت پایین) و 0x342 (بایت بالا) قرار دهد.

پس از اجرای بخش الف و ب، وضعیت ثبات SReg چگونه است؟

راهنمایی: برای جمع ۱۶ بیتی، باید دو جمع ۸ بیتی انجام دهید، اولی روی بایتهای کم ارزش و دومی روی بایتهای باارزش. برای احتساب رقم نقلی، جمع دوم را باید با دستور ADC(Cary with ADD)به جای ADD انجام دهید. اختیاری: جمع ۱۶ بیتی ۳ عدد را حساب کنید (عدد 0x3D20 را به اعداد قبلی اضافه کنید)

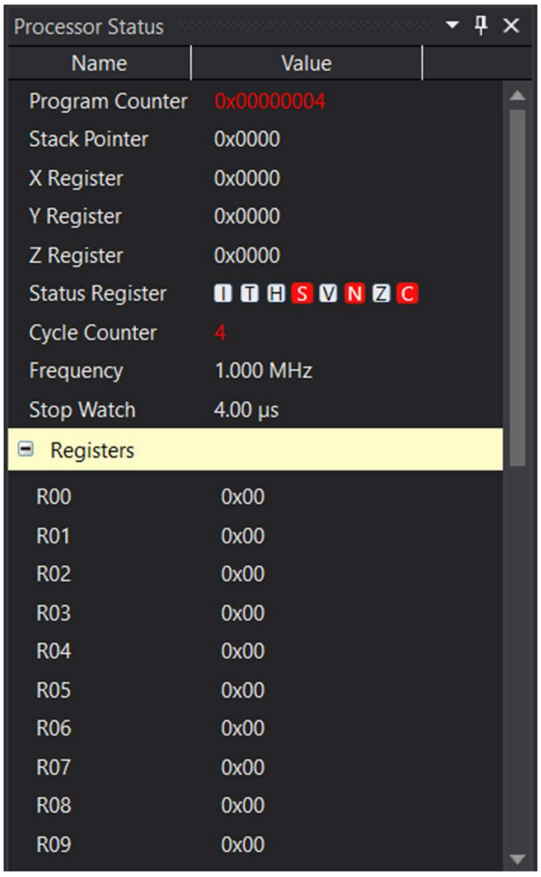
الف

```
start:
    inc r16

       ldi r17, 0xf0
       ldi r20, 0xa5
       add r20, r17

    rjmp start
```



وضعیت رجیستر ها بعد از انجام ۵ سیکل

رجیستر SReg در قسمت (الف) ۱ میشود، زیرا در جمع دو عدد 0xA5 و 0xF0 رقم نقلی تولید می شود.

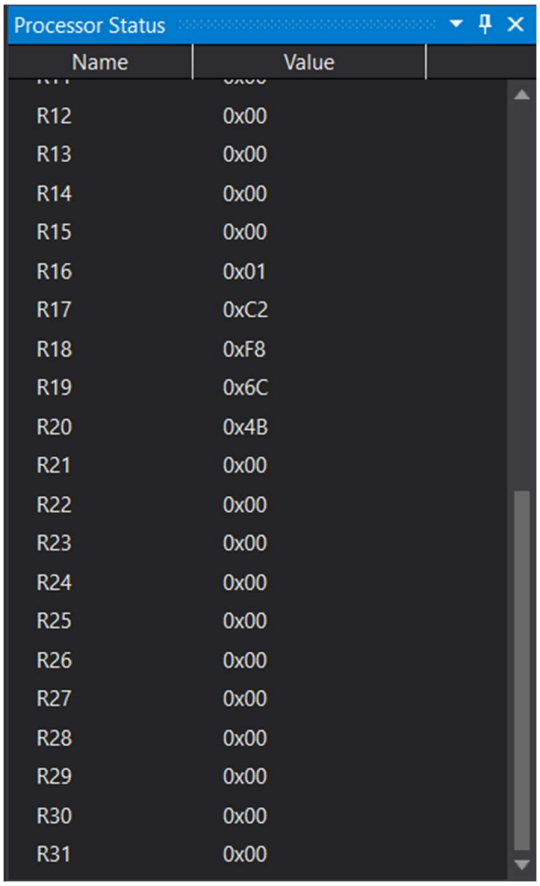


وضعیت Status Register بعد از انجام ۵ سیکل

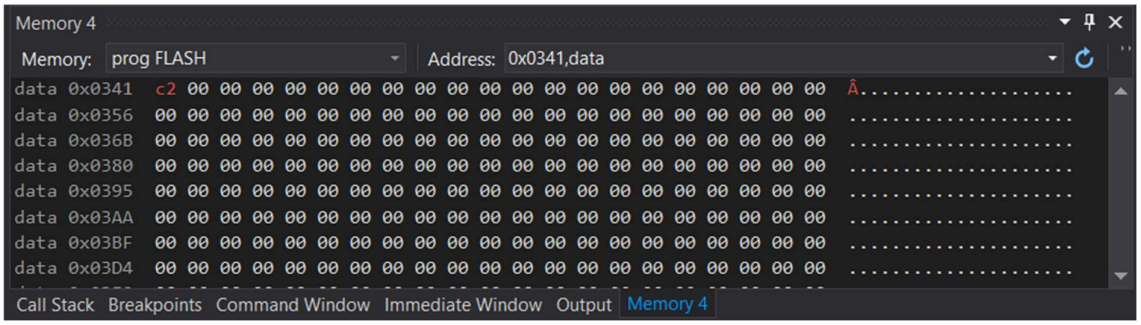
ب

```
start:
    inc r16

       ldi r17, 0xca
       ldi r18, 0xf8
       ldi r19, 0x20
       ldi r20, 0x4b
       add r17, r18
       adc r19, r20
       sts 0x341, r17
       sts 0x342, r19


    rjmp start
```

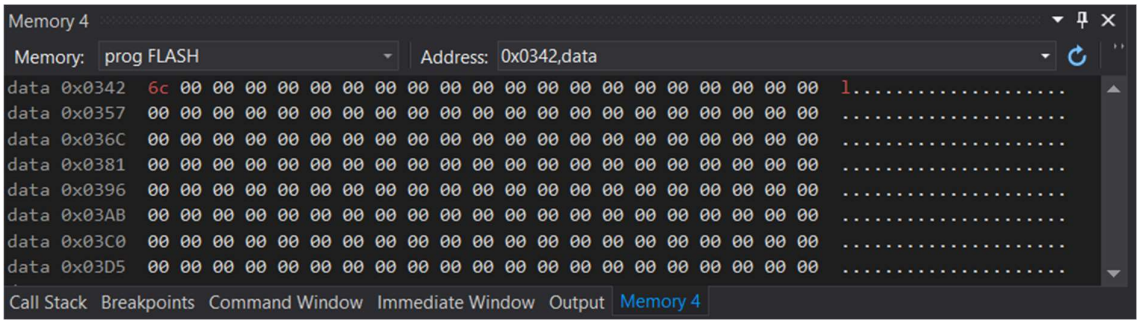
می دانیم نتیجه جمع دو عدد 0x20CA و 0x4BF8 ، 0x6CC2 می شود. به این منظور ۴ بیت LSB هر عدد را در یک رجیستر مجزا ذخیره کردیم و ۴ بیت MSB را هم همینطور و سپس LSB ها را باهم و MSB ها را باهم جمع و در ادرس های گفته شده ذخیره کردیم.



وضعیت رجیستر ها بعد از انجام ۹ سیکل



جمع ۴ بیت LSB ذخیره شده در ادرس 0X341

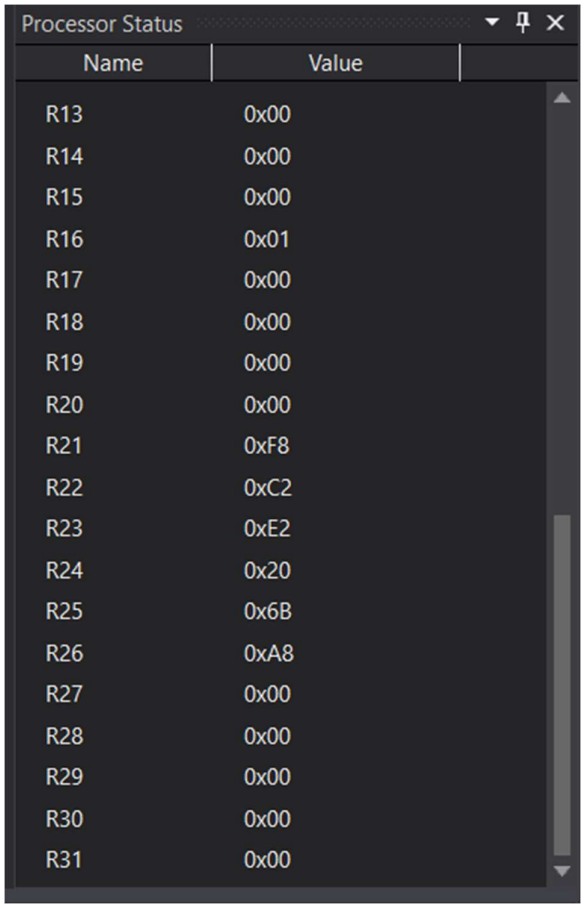


جمع ۴ بیت LSB ذخیره شده در ادرس 0X342

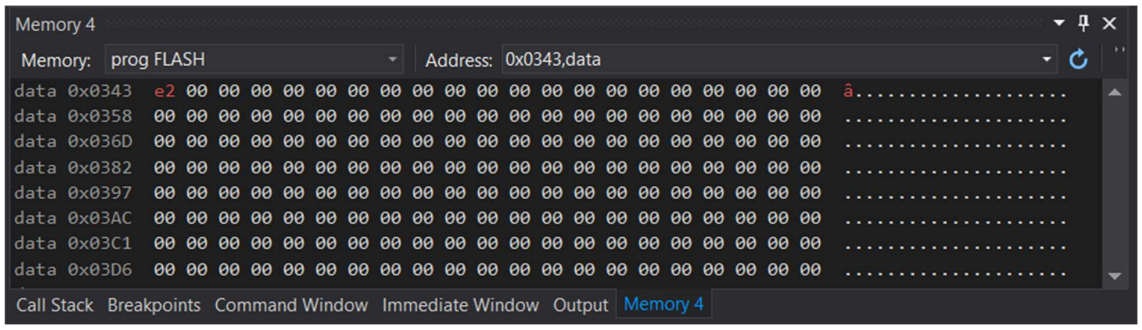
رجیستر SReg در قسمت (ب) ۱ میشود، زیرا در جمع بیت های کم رقم نقلی تولید می شود.

اختیاری

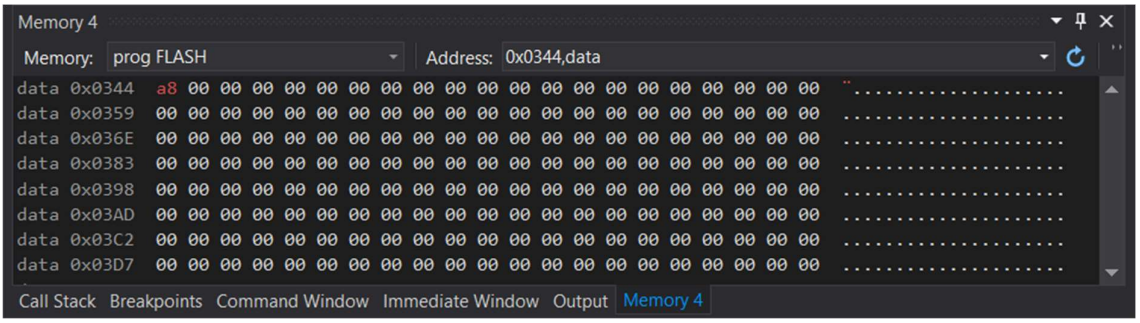
```
start:
    inc r16

       ldi r21, 0xf8
       ldi r22, 0xca
       ldi r23, 0x20
       add r22, r21
       add r23, r22

       ldi r24, 0x20
       ldi r25, 0x4b
       ldi r26, 0x3d
       add r25, r24
       adc r26, r25

       sts 0x343, r23
       sts 0x344, r26


    rjmp start
```



وضعیت رجیستر ها بعد از انجام ۱۳ سیکل



جمع ۴ بیت LSB ذخیره شده در ادرس 0X343



جمع ۴ بیت MSB ذخیره شده در ادرس 0X344

۲) برنامه ای به زبان C بنویسید که:

پورت A را ورودی و پورت B و C را خروجی تعریف کند. ۸ کلید به پایه های پورت A و ۸ عدد LEDبه پایه های پورت B وصل کنید. هرگاه کلیدی روی پورت A فشرده شد، LED متناظر آن تغییر وضعیت دهد (اگر روشن است خاموش شود و برعکس). یک LED هم به PORTC.5 وصل کنید که با تاخیر مختصری، مرتبا روشن و خاموش شود (طوریکه فشرده شدن کلیدها هم از دست نرود).

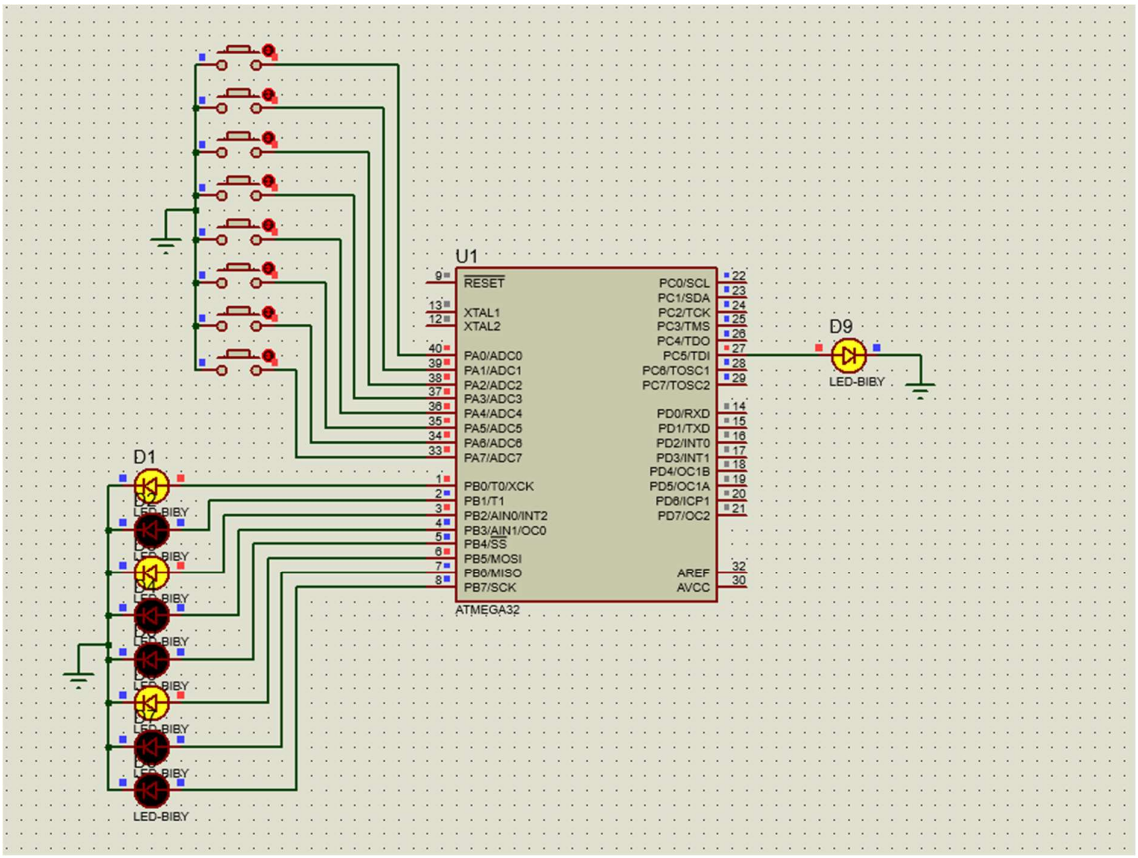
کد برنامه به صورت زیر است:

```
#include <mega32.h>
#include <delay.h>

#define oneLED PORTC.5
#define LED0 PORTB.0
#define LED1 PORTB.1
#define LED2 PORTB.2
#define LED3 PORTB.3
#define LED4 PORTB.4
#define LED5 PORTB.5
#define LED6 PORTB.6
#define LED7 PORTB.7
#define btn0 PINA.0
#define btn1 PINA.1
#define btn2 PINA.2
#define btn3 PINA.3
#define btn4 PINA.4
#define btn5 PINA.5
#define btn6 PINA.6
#define btn7 PINA.7
#define on 1
#define off 0

unsigned char t1 = 0, flag = 0;

void main(void)
{

    DDRA = 0x00;
    PORTA = 0xff;

    DDRB = 0xff;
    PORTB = 0x00;

    DDRC = 0xff;
    PORTC = 0x00;

    while (1)
    {
        oneLED = on;
        delay_ms(100);
        oneLED = off;
        delay_ms(100);
        t1++;
        if(flag == 0)
        {
```

```
            if(btn0 == 0)
            {
                flag = 1;
                t1 = 0;
                LED0 = ~LED0;
            }

            if(btn1 == 0)
            {
                flag = 1;
                t1 = 0;
                LED1 = ~LED1;
            }
            if(btn2 == 0)
            {
                flag = 1;
                t1 = 0;
                LED2 = ~LED2;
            }
            if(btn3 == 0)
            {
                flag = 1;
                t1 = 0;
                LED3 = ~LED3;
            }
            if(btn4 == 0)
            {
                flag = 1;
                t1 = 0;
                LED4 = ~LED4;
            }
            if(btn5 == 0)
            {
                flag = 1;
                t1 = 0;
                LED5 = ~LED5;
            }
            if(btn6 == 0)
            {
                flag = 1;
                t1 = 0;
                LED6 = ~LED6;
            }
            if(btn7 == 0)
            {
                flag = 1;
                t1 = 0;
                LED7 = ~LED7;
            }
        }
        else
        {
            if(t1 > 1)
            {
                flag = 0;
            }
        }

    }//End while(1)
}//End main()
```

مقدار فرکانس کاری میکرو هم در CodeVision و هم در Proteus، ۸ مگاهرتز در نظر گرفته شده است.

نتیجه شبیه سازی:



- تمام فایل ها (شبیه سازی و کد و...) در پیوست ارائه شده است.