پاسخ کوییز اول سیستم دیجیتال ۲

۱- در دستور اول نمی توان از رجیستر R5 استفاده کرد و برنامه بصورت ذیل اصلاح می شود :

```
LDI  R16,0x55    ;   R16 = 55H
MOV  R5, R16     ;   R5 = 55H
```

در دستور سوم ، آدرس 005E که ۱۶ بیتی بوده مربوط به SFR است که نمب توان با دستور IN آدرس ۱۶ بیتی بکار برد دو روش برای اصلاح آن وجود دارد :

روش اول : `LDS  R17, 0x005E`

روش دوم : `IN  R17, 0x3E`

در هر صورت مقدار آدرس 3EH در R17 قرار می گیرد.

دستور EOR روی عدد ثابت وجود ندارد. پس بصورت ذیل اصلاح می شود :

(چون قبلا R17 قبلا صفر بوده است)

```
Loop : EOR  R17, R16     ;   R17 = 55H
```

پس برنامه با اجرای دستور BRNE next به دستور COM می رود.

```
COM  R17      ;   R17 = AAH
```

دستور ADDI نداریم. البته دفعه اول اجرا نمی شود و بصورت ذیل باید اصلاح شود :

```
LDI  R18, 0x08
ADD  R17, R18
```

پس از اجرای JMP Loop و برگشت به Loop داریم :

```
Loop : EOR  R17, R16     ;   R17 = FFH
```

پس برنامه با اجرای دستور BRNE next به دستور COM می رود.

```
COM  R17      ;   R17 = 00H
```

۲- تعداد تفاضلها ۱۶ یا 10H تا است و اگر همگی مثبت باشند و حداکثر برابر FFH ، آنگاه حدکثر جمع آنها برابر FF0H خواهد بود. پس به دو پورت احتیاج داریم (پورت A و C)

اگر بخواهیم از اشاره گر استفاده کنیم ، ابتدا بایستی مقدار دو رجیستر را در آدرسی در حافظه SRAM داخلی ذخیره کنیم. از اشاره گر Z استفاده شده است. مقدار دو رجیستری که باید از هم تفریق شوند، در R28 و R29 قرار می گیرد. حاصل جمع ابتدا در رجیستر R26 و R27 و سپس در R1 و R2 قرار می گیرد. مقدار ماکریمم اعداد مثبت ابتدا در رجیستر R25 و سپس در رجیستر R0 قرار می گیرد.

یک نمونه از این برنامه در ذیل نوشته شده است.

```
        STS  0x0500, R25
        STS  0x0501, R26
        STS  0x0502, R27
        STS  0x0503, R28
        STS  0x0504, R29
        STS  0x0505, R30
        STS  0x0506, R31
        LDS  R25, 0x00
        LDS  R26, 0x00
        LDS  R27, 0x00
        LDS  R30, 0x00
        LDS  R31, 0x00
next :  CP  R30, 0x18
        BRE0  next1
        LD   R28, Z+
        LD   R29, Z+
        SUB  R29, R28
        BRCS  next
        ADD  R26, R29
        BRCC  nextt
        INC  R27
nextt :  CP   R29, R25
        BRCS  next
        MOV  R25, R29
        JMP  next
next1 :  MOV R0, R25
         MOV  R1, R26
         MOV  R2, R27
         LDS R25 , 0x0500
         LDS R26 , 0x0501
         LDS  R27, 0x0502
         LDS  R28, 0x0503
         LDS  R29, 0x0504
         LDS  R30, 0x0505
         LDS  R31, 0x0506
         SUB  R25, R24
         BRCS  next2
         ADD  R1, R25
         BRCC  nextt1
         INC  R2
nextt1 :  CP   R25, R0
         BRCS  next2
         MOV  R0, R25
next2 :  SUB  R27, R26
         BRCS  next3
         ADD  R1, R27
         BRCC  nextt2
         INC  R2
nextt2 :  CP   R27, R0
         BRCS  next3
         MOV  R0, R27
next3 :  SUB  R29, R28
         BRCS  next4
         ADD  R1, R29
         BRCC  nextt3
         INC  R2
nextt3 :  CP   R29, R0
         BRCS  next4
         MOV  R0, R29
next4 :  SUB  R31, R30
        BRCS  next3
        ADD  R1, R31
         BRCC  nextt4
         INC  R2
nextt4 :  CP   R31, R0
        BRCS  nextfinish
        MOV  R0, R31
nextfinish :  OUT  PortA, R1
              OUT  PortC, R2
              OUT  PortB, R0
finish :  JMP  finish
```