سری سوم:
سوال ۱)

```
.include "M64IDEF.inc"

.ORG 0x0000
    JMP main
.ORG 0x000C
    JMP INT5_ISR
.ORG 0x0050
main: LDI R16,High(RAMEND)   ; تنظیم Stack
    Out SPH, R16
    LDI R16,low(RAMEND)
    out SPL,R16

    CLR R16                  ; تنظیم پورت های
    Out DDRC, R16            B,C,D به عنوان ورودی
    Out DDRB, R16
    out DDRD , R16

    LDI R24, 0x00            ; تنظیم اشاره گر
    LDI R25, 0x30            برای آدرس 0x3000

    LDI R24, 0x0V            ; فعال کردن وقفه پین
    Out EICRB, R16

    LDI R16, 0x20            ; فعال کردن وقفه شماره
    Out EIMSK, R16
    CLR R20                  ; مقدار دهی برای مقایسه دو عدد متوالی فرد
    SEI                      ; فعال سازی وقفه

Loop:
    IN R21, PINC             ; خواندن پورت C
    SBRS R21,0               ; اگر بیت صفر، صفر باشد (زوج)، خط بعدی پرش می شود
    JMP NOT_Both_odd
    SBRS R20,0               ; اگر عدد قبلی زوج بود، شرط دو عدد
    JMP Update_pre           متوالی فرد برقرار نیست

    CP R21, R20              ; مقایسه عدد قدیم و عدد جدید R21
    BRLO STORE_21            اگر R21 < R20 بود، پرش به زدن R21
    BREQ Loop
    ST X+, R20
    JMP Update_PRE
```

روش برنامه اصلی:

```
STORE_R21:
    ST X+, R21 ;  R21 , افزایش اشاره‌گر
                  ذخیره در حافظه

UPDATE_PREV:
    MOV R20, R21 ; عدد فعلی برای دور بعد، عدد قبلی محسوب
                   می‌شود

NOT_Both_odd:
    MOV R20, R21 ; عدد قبلی update و تکرار حلقه
    JMP Loop

                                  روتین برنامه وقفه
INT5_ISR:
    PUSH R16
    IN   R16, SREG   ; دستوراتی هستند که flag هارا
    PUSH R16           تغییر می‌دهند پس برای اینکه تغییر نکند
                       واطلاعات حفظ شود این SREG را
                       PUSH , pop می‌کنیم

    IN  R16, PINB ; خواندن پورت B
    IN  R17, PIND ; خواندن پورت D

    CP   R16, R17
    BRLO B_smaller ; مقایسه دو مقدار برای محاسبه قدرمطلق

    SUB R16, R17 ; R18 = B - D  ←  B > D اگر
    MOV R18, R16
    JMP INT5_end

B_smaller:        R18 = D - B  ←  D > B اگر
    SUB R17, R16
    MOV R18, R17

INT5_end:
    POP R16
    OUT SREG, R16
    POP R16
    RETI
```

سوال ۲)

```
.include "M64DEF.inc"

.ORG 0x0000
    JMP main
.ORG 0x0010      ; INT7 وقفه برای (کلید ۱)
    JMP INT7_ISR

.ORG 0x0004      ; INT1 وقفه برای (کلید ۲)
    JMP INT1_ISR
.ORG 0x0050
main: LDI R16, High(RAMEND)      ; تنظیم و مقداردهی stack
    Out SPH, R16
    LDI R16, low(RAMEND)
    out SPL, R16

    CLR R16                      ; تنظیم پورت‌های B,E,C,F به عنوان ورودی
    Out DDRB, R16
    Out DDRC, R16
    Out DDRF, R16
    Out DDRE, R16

    LDI R16, 0xC0                ; rising edge رها کردن = لبه بالا رونده
    Out EICRB, R16

    LDI R16, 0X00                ; Falling edge نگه داشتن فرض شده
    STS EICRA, R16

    LDI R16, 0x82                ; فعال سازی وقفه INT7 و INT1
    out EIMSK, R16

SEI

Loop: IN R16, PINB               ; خواندن پورت B
    LDI R17, 15
    MUL R16, R17
    STS 0x2000, R0               ; بایت باارزش در R1 و کم ارزش در R0 قرار می‌گیرد برا ضرب R1, R0
    STS 0x2001, R1
JMP loop
```

```
INT1_ISR :                       ; درصورت فرد بودن شمارش بیت های
                                 ; 1 پورت
    PUSH R16
    IN   R16, SREG   ; بعضی از دستورالعمل ها flag ها را تغییر می دهد.
    PUSH R16
    PUSH R18
    PUSH R16

    IN   R16, Pinc   ; خواندن پورت C
    SBRS R16, 0      ; (اگر بیت صفر یک باشد فرد) ادامه می دهد در غیر این
    JMP  INT1_end    ; صورت خارج می شود

    CLR  R17         ; شمارش بیت های ۱ (تعداد یکها) در R17
    LDI  R19, 8      ; شمارنده حلقه برای ۸ بیت

Count: SBRC R16, 0   ; اگر بیت صفر، 0 بود خط بعدی را رد کن
    INC  R17
    LSR  R16         ; شیفت به راست برای بررسی بیت بعدی
    DEC  R19
    BRNE count

INT1_end: pop R19
    pop  R18
    pop  R16
    out  SREG, R16
    pop  R16
    RETI

INT7_ISR : PUSH R16
    IN   R16, SREG
    PUSH R16
    PUSH R17

    IN   R16, PINE
    IN   R17, PINF
    ADD  R16, R17
    STS  0x2500, R16   ; ذخیره در آدرس خواسته شده
    pop  R17
    pop  R16
    out  SREG, R16
    POP  R16
    RETI
```

فرض کردیم جمع بیشتر از ۸ بیت نشده است ؛
درصورت بیشتر شدن باید carry ذخیره شود

سوال ۳)

```
.INclude "M64IDEF.inc"

.ORG 0x0000
   JMP main
.ORG 0x000A
   JMP INT4_ISR
.ORG 0x0050
main: LDI R16, Low(RAMEND)
   out SPL, R16
   LDI R16, high(RAMEND)
   out SPH, R16

   CLR R16          ; تنظیم پورت دما
   out DDRA, R16      A خروجی

   LDI R16, 0x0FF   ; F خروجی
   out DDRF, R16      (متصل به LED)

   LDI R16, 0x01    ;      وقفه INT4
   out EICRB, R16     تنظیم برای هردولبه

   LDI R16, 0x10
   Out EIMSK, R16

   CLR R19          ; Falling , Rising برای Flag

   SEI

;

loop: IN R16, PINA
   LSR R16          ;      تقسیم بر۲
   out PORTF, R16   ; نمایش مقدار روی LED
   JMP loop           متصل به پورت F
```

```
INT4_ISR : PUSH  R16
           IN    R16, SREG
           PUSH  R16
           CPI   R19, 0
           BRNE  Rising

Falling:   LDI   R16, 0XFF
           out   PORTF, R16
           LDI R19, 1          ; آماده برای Rising
           JMP  INT4_end

Rising:    LDI  R16, 0x55  ; 01010101 → روشن شدن پورت
           out  PORTF, R16      های فرد
           CLR  R19            ; صفر شدن R19 برای
                                 اینکه دوباره بتواند
                                 وارد Falling بشود

INT4_end : pop  R16
           Out  SREG, R16
           pop  R16
           RETI
```