EE2016 Experiment 4

Group 3 - EE23B039, EE23B033, EE23B027

Task 1

Finding (i) maximum and (ii) minimum of 10 numbers stored in flash.

```
.cseg

.org 0x0000
rjmp RESET

numbers:
    .db 0x32, 0x15, 0x9F, 0x4C, 0x22, 0x5A, 0x1E, 0x7B, 0x12, 0x89

RESET:
    ldi ZH, high(numbers << 1)
    ldi ZL, low(numbers << 1)

    ldi r17, 0x00 // r17 will store the maximum value
    ldi r18, 0xFF // r18 will store the minimum value

    ldi r20, 10 //loop counter

MaxMin:
    lpm r19, Z+
    cp r19, r17 //cp rd, rr = rd-rr, based on the value of rd and rr, one of the flags is chosen
    brcs NotMax //brcs is activated when carry is required to do r19-r17
    mov r17, r19

NotMax:
    cp r19, r18
    brcc NotMin //here brcc is activated when carry is not required i.e, when r19>r18
    mov r18, r19

NotMin:
    dec r20
    brne MaxMin

nop

End:
    rjmp End
```

Debugging

Click here to see debugging in microchip studio for min and max

Task 2

Finding sum of 10 numbers stored in flash.

```
.cseg

.org 0x0000
rjmp RESET

numbers:
    .db 0x01, 0x03, 0x02, 0x04, 0x12, 0x00, 0x10, 0x07, 0x06, 0x09
```

```
RESET:
    ldi ZH, high(numbers << 1)
    ldi ZL, low(numbers << 1)

        ldi r20, 10 //loop counter
        ldi r17, 0x00 //sum initially set to 0


Add_numbers:
        lpm r19, Z+
        add r17, r19
        dec r20
        brne Add_numbers

nop

end:
        rjmp end
```

Debugging

Click here to see debugging in microchip studio for finding sum of 10 numbers

Task 3

Sort 5 numbers stored in flash memory in arbitrary order and write the final results to data memory (show the results in the memory window).

We used 0x01, 0x03, 0x02, 0x04, 0x12 as an example.

```
.cseg

.org 0x0000
rjmp RESET

numbers:
        .db 0x01, 0x03, 0x02, 0x04, 0x12

RESET:

    ldi ZH, high(numbers << 1)
    ldi ZL, low(numbers << 1)


    lpm r19, Z+
    mov r21, r19


    lpm r19, Z+ //load register with number
    cp r19, r21 //compare it with value in r21
    brcc skip_2 // if r19 is greater than r21 go to skip_2
    mov r22, r21 //else move r21 to r22
    mov r21, r19 // move r19 to r21
    rjmp load_3
skip_2:
    mov r22, r19

load_3:
    lpm r19, Z+
    cp r19, r21
    brcc skip_3_1
    mov r23, r22
    mov r22, r21
    mov r21, r19
    rjmp load_4
skip_3_1:
    cp r19, r22
    brcc skip_3_2
```

```
    mov r23, r22
    mov r22, r19
    rjmp load_4
skip_3_2:
    mov r23, r19

load_4:
    lpm r19, Z+
    cp r19, r21
    brcc skip_4_1
    mov r24, r23
    mov r23, r22
    mov r22, r21
    mov r21, r19
    rjmp load_5
skip_4_1:
    cp r19, r22
    brcc skip_4_2
    mov r24, r23
    mov r23, r22
    mov r22, r19
    rjmp load_5
skip_4_2:
    cp r19, r23
    brcc skip_4_3
    mov r24, r23
    mov r23, r19
    rjmp load_5
skip_4_3:
    mov r24, r19

load_5:
    lpm r19, Z+
    cp r19, r21
    brcc skip_5_1
    mov r25, r24
    mov r24, r23
    mov r23, r22
    mov r22, r21
    mov r21, r19
    rjmp end
skip_5_1:
    cp r19, r22
    brcc skip_5_2
    mov r25, r24
    mov r24, r23
    mov r23, r22
    mov r22, r19
    rjmp end
skip_5_2:
    cp r19, r23
    brcc skip_5_3
    mov r25, r24
    mov r24, r23
    mov r23, r19
    rjmp end
skip_5_3:
    cp r19, r24
    brcc skip_5_4
    mov r25, r24
    mov r24, r19
    rjmp end
skip_5_4:
    mov r25, r19

nop

end:
    rjmp end
```

Debugging

Click here to see debugging in microchip studio for sorting