For any given filter H the input signal X, then output is
Y[n] = X[n]*H[n] convolution of input and impulse response
Assuming x[0] = 0;
y[4] = x[4]*h[0] + x[3]*h[1] + x[2]*h[2] + x[1]*h[3]
x is the input sample, it will update every time when ADC conversion complete
In case of the moving average filter y(n)= (x(n)+x(n-1)+c(n-2)+x(n-3))/4
h[0] = h[1] = h[2] = h[3] = 0.25

But presenting 0.25 in binary format is difficult, so multiply the coeff by 256 and at the end divide the output by 256, to get the actual output.

So h[n] = 0.25*256 = 64

```
.EQU H0 = 0x40        ;filter coefficient from MATLAB FDA tool, multiplied by 256
```

____

.

____

Like wise define other variables with coeff values.

Next after the ADC input is available

```
IN R20, ADCL
IN R21, ADCH
```

Use ADCH value to multiply with coefff.
The operation x[4]*h[0]

```
LDI R28, H0             ;Load filter coefficient H0
MOV R22, R21
MUL R28, R22             ; 2 Clock cycle Multiplication R1:R0 = R28*R22
ADD R29, R0
ADC R30, R1
```

Then x[4]*h[0]+ x[3]*h[1]

```
LDI R28, H1             ;Load filter coefficient H1
MUL R28, R23            ; 2 Clock cycle Multiplication R1:R0 = R28*R23
ADD R29, R0
ADC R30, R1
```

Like wise write code for other operations and complete the x[4]*h[0] + x[3]*h[1] + x[2]*h[2] + x[1]*h[3]

____

.

Put result at DAC input port which is connected to PORTB IN this case
The whole result is in R30(Most significant Byte) and R29(Least significant Byte)
Since initially you had multiplied the coefficients by 256, so at last you will have to divide by 256.

For this you can Right shift result 8 positions to divide the result by 256. If you Right shift result by 8 positions, the data in register R30 will come in R29.
Here you need not shift but leave the R29 and take only R30 that contains 8-bit MSB result.

```
OUT PORTB, R30
```

Next get registers ready for next input sample to come so do a sample shift

```
MOV R25, R24
MOV R24, R23
MOV R23, R22

LDI R29, 0
LDI R30, 0

RJMP…….
```