HW 3 Solution

1. N is divisible by M, if N/M yields no remainders. Write a subroutine called NdivM that accepts as input 2 unsigned byte values (R16=N and R17=M), and returns R18=1 or 0 if N is divisible by M.

```
NdivM:
;N = R16
;M = R17
;divisible = R18 = 0 OR 1

                CLR R18           ;ASSUME NOT DIVISIBLE
WHILE:          CP R16, R17       ;N > D?
                BRLO ENDW         ;N<D
                SUB R16,R17      ;N-D
                RJMP WHILE

ENDW:           CPI R16, 0
                BRNE NOTDIV
                LDI R18, 1        ;IS DIVISIBLE
NOTDIV:         RET
```

2. Repeat problem 1 as a MACRO code. Macro call format:
NdivM @0, @1, @2
@0=1 if N is divisible by M, 0 otherwise
@1=N
@2=M

```
.MACRO          NdivM
                CLR @0        ;ASSUME NOT DIVISIBLE
WHILE:          CP @1, @2        ;N > D?
                BRLO ENDW       ;N<D
                SUB @1,@2     ;N-D
                RJMP WHILE

ENDW:           CPI @1, 0
                BRNE NOTDIV
                LDI @0, 1          ;IS DIVISIBLE
NOTDIV:
.ENDM
```

3. Given a signed byte array stored in the program memory. Write a subroutine called checksum that takes as input: the address of the array and its length, then returns the sum of all bytes as a byte value.

```
checksum:
;R31:R30 = Z = address of array
;R16 = length
;R17 = sum
                CLR R17         ;SUM=0
L1:             LPM R18, Z+     ;ARRAY[I]
                ADD R17, R18    ;SUM = SUM + ARRAY[I]
                DEC R16
                TST R16
                BRNE L1

                RET
```

Redo the code with another extra instruction: PUSH/POP registers that are used inside procedure.

```
checksum:
                PUSH R18
;R31:R30 = Z = address of array
;R16 = length
;R17 = sum
                CLR R17         ;SUM=0
L1:             LPM R18, Z+     ;ARRAY[I]
                ADD R17, R18    ;SUM = SUM + ARRAY[I]
                DEC R16
                TST R16
                BRNE L1

                POP R18
                RET
```

4. Repeat problem 3 as a MACRO code. Macro call format:
   checksum @0, @1, @2
   @0=sum
   @1=array
   @2=array length

```
.MACRO checksum
                LDI ZH, HIGH(2*@1)
                LDI ZL, LOW(2*@1)

                CLR @0                  ;SUM=0
L1:             LPM R18, Z+             ;ARRAY[I]
                ADD @0, R18             ;SUM = SUM + ARRAY[I]
                DEC @2
                TST @2
                BRNE L1
.ENDM
```

Redo the code with another extra instruction: PUSH/POP registers that are used inside procedure.

```
.MACRO checksum
                PUSH R18
                PUSH ZH
                PUSH ZL

                LDI ZH, HIGH(2*@1)
                LDI ZL, LOW(2*@1)

                CLR @0                  ;SUM=0
L1:             LPM R18, Z+             ;ARRAY[I]
                ADD @0, R18             ;SUM = SUM + ARRAY[I]
                DEC @2
                TST @2
                BRNE L1
                PUSH ZL
                PUSH ZH
                PUSH R18
.ENDM
```

5. Given a signed byte array stored in the program memory. Write a subroutine called medval that takes as input: the address of the array, its length, and returns the median value of the array.

Idea: for each element in the array count how many elements are greater or higher. If the there are exactly array_length/2 elements then that element is the median.

```
Median_array:
;Z=array address
;assume the array is in the DM
;R16=MEDIAN VALUE
;R18= ARRAY LENGTH


                    PUSH R17
                    PUSH R19
                    PUSH R20
                    PUSH R21
                    PUSH XL
                    PUSH XH
                    PUSH YL
                    PUSH YH

                    MOVW X, Z
                    MOV R21, R18
                    LSR R21                 ;LENGTH/2

L1:                 MOV R19,R18             ;COUNT FOR ARRAY LENGTH
                    CLR R20                 ;COUNT FOR ARRAY[I] GTE
                    MOVW Y, X
                    LD R16, Z+              ;ARRAY[I]
L2:                 LD R17, Y+              ;ARRAY[J]
                    CP R16, R17
                    BRLT NEXTJ
                    INC R20

NEXTJ:              DEC R19
                    TST R19
                    BRNE L2

                    CP R20,R21              ;GTE COUNT == LENGTH/2 ?
                    BREQ FOUND
                    RJMP L1             ;IF NOT MEDIAN CHECK NEXT ARRAY[I]

FOUND:              POP YL
                    POP YH
                    POP XL
                    POP XH
                    POP R21
                    POP R20
                    POP R19
                    POP R17
                    RET
```

6. Repeat problem 5 as a MACRO code. Macro call format:
medval @0, @1, @2
@0=max value
@1=array
@2=array len

```
.MACRO                  Median_array:
;@1=array
;assume the array is in the DM
;@0=MEDIAN VALUE
;@2= ARRAY LENGTH


                    PUSH R16
                    PUSH R17
                    PUSH R18
                    PUSH R19
                    PUSH R20
                    PUSH R21
                    PUSH XL
                    PUSH XH
                    PUSH YL
                    PUSH YH
                    PUSH ZL
                    PUSH ZH

                    LDI ZH, HIGH(@0)
                    LDI ZL, LOW(@0)
                    LDI R18, @2

                    MOVW X, Z
                    MOV R21, R18
                    LSR R21                 ;LENGTH/2

L1:                 MOV R19,R18             ;COUNT FOR ARRAY LENGTH
                    CLR R20                     ;COUNT FOR ARRAY[I] GTE
                    MOVW Y, X
                    LD R16, Z+              ;ARRAY[I]
L2:                 LD R17, Y+              ;ARRAY[J]
                    CP R16, R17
                    BRLT NEXTJ
                    INC R20
```

```
NEXTJ:          DEC R19
                TST R19
                BRNE L2

                CP R20,R21              ;GTE COUNT == LENGTH/2 ?
                BREQ FOUND
                RJMP L1                 ;IF NOT MEDIAN CHECK NEXT ARRAY[I]

FOUND:          MOV @0, R16
                POP ZH
                POP ZL
                POP YL
                POP YH
                POP XL
                POP XH
                POP R21
                POP R20
                POP R19
                POP R18
                POP R17
                POP R16

.ENDM
```

7. Write a subroutine called median that takes as input 3 unsigned bytes and returns a byte value that represents the median value of the 3 bytes.

```
median:
;r16 = val1
;r17 = val2
;r18 = val3
;R19 = MEDIAN
;IDEA: FORCE R16 TO MAX, R18 TO BE MIN. This leaves r17 as median

            CP R16, R17       ;R16 >= R17     make r16 max
            BRSH NEXTMAX
            PUSH R17          ;IF R16< R17 => SWAP THE VALUES
            PUSH R16
            POP R17
            POP R16

NEXTMAX:    CP R16, R18       ;R16 >= R18     make r16 max
            BRSH CHKMIN
            PUSH R18          ;IF R16< R18 => SWAP THE VALUES
            PUSH R16
            POP R18
            POP R16

CHKMIN:     CP R18, R17       ;R18 < R17  make r18 min
            BRLO DONE
            PUSH R18          ;IF R18>= R17 => SWAP THE VALUES
            PUSH R17
            POP R18
            POP R17

DONE:       MOV R19, R17
            RET
```

8. Repeat problem 7 as a MACRO code. Macro call format:
median @0, @1, @2, @3
@0=max value
@1=val1
@2=val2
@3=val3

```
.MACRO      median
                  CP @1, @2           ;R16 >= R17      make r16 max
                  BRSH NEXTMAX
                  PUSH @2             ;IF R16< R17 => SWAP THE VALUES
                  PUSH @1
                  POP @2
                  POP @1

NEXTMAX:          CP @1, @3           ;R16 >= R18      make r16 max
                  BRSH CHKMIN
                  PUSH R18            ;IF R16< R18 => SWAP THE VALUES
                  PUSH @1
                  POP R18
                  POP @1

CHKMIN:           CP @3, @2           ;R18 < R17  make r18 min
                  BRLO DONE
                  PUSH @3             ;IF R18>= R17 => SWAP THE VALUES
                  PUSH @2
                  POP @3
                  POP @2

DONE:             MOV @0, @2
.ENDM
```

9. Write a subroutine called changesign that will take as input: data memory array address, and length. It will negate each value in the array

```
changesign:
;Z = ADDRESS
;R16 = LENGTH

L1:             LD R17, Z
                NEG R17
                ST Z+, R17
                DEC R16
                TST R16
                BRNE L1
                RET
```

10. Repeat problem 9 as a MACRO code. Macro call format:
    changesign @0, @1
    @0=ARRAY
    @1=length

```
.MACRO          changesign

                LDI ZH, HIGH(@0)
                LDI ZL, LOW (@0)

L1:             LD R17, Z
                NEG R17
                ST Z+, R17
                DEC R16
                TST R16
                BRNE L1
.ENDM
```