SignX and SignY need to be defined as bits.
All parameters in user-set register bank
DIV32 requires the use of register bank 1 as well

subroutine **CONV816** 8-bit Signed number to 16-Bit Signed number conversion
format: 2's Complement
input: r0 = X
output: r1, r0 = X with sign extended to 16 bits
alters:

subroutine **MAG16** 16-Bit Signed number to Magnitude conversion
format: 2's Complement
input: r1, r0 = X
output: r1, r0 = |X|
alters: PSW

subroutine **NEG16** 16-Bit Negate
format: 2's Complement
input: r1, r0 = X
output: r1, r0 = -X
alters: PSW

subroutine **LADD8** 8-Bit Signed Addition with Range limiter
format: 2's Complement
input: r0 = X r1 = Y
output: r0 = signed sum S = X + Y
80h < S < 7Fh
alters: PSW

subroutine **ADD16** 16-Bit Signed Addition
format: 2's Complement
input: r1, r0 = X r3, r2 = Y
output: r1, r0 = signed sum S = X + Y

SignX is set if the result has a carry out

SignY is set if the result is out of range

alters: PSW

subroutine **LADD16** 16-Bit Signed Addition with Range limiter

format: 2's Complement

input: r1, r0 = X r3, r2 = Y

output: r1, r0 = signed sum S = X + Y

8000h < S < 7FFFh

alters: PSW

subroutine **ADD32** 32-Bit Signed Addition

format: 2's Complement

input: r3, r2, r1, r0 = X r7, r6, r5, r4 = Y

output: r3, r2, r1, r0 = signed sum S = X + Y

SignX is set if the result has a carry out

SignY is set if the result is out of range

alters: PSW

subroutine **SUB16** 16-Bit Signed Subtraction

format: 2's Complement

input: r1, r0 = X r3, r2 = Y

output: r1, r0 = signed difference D = X - Y

SignX is set if the result has a carry out

SignY is set if the result is out of range

alters: PSW

subroutine **SUB32** 32-Bit Signed subtraction

format: 2's Complement

input: r3, r2, r1, r0 = X r7, r6, r5, r4 = Y

output: r3, r2, r1, r0 = signed difference D = X - Y

SignX is set if the result has a carry out

SignY is set if the result is out of range

alters: PSW

subroutine **MUL8** 8-Bit × 8-Bit to 16-Bit Product Multiply

format: 2's Complement

input: r0 = multiplicand X r1 = multiplier Y

output: r1, r0 = product P = X × Y.

alters: PSW, SignX and SignY

subroutine **UMUL8** 8-Bit × 8-Bit to 16-Bit Product Multiply

format: unsigned

input: r0 = multiplicand X r1 = multiplier Y

output: r1, r0 = product P = X × Y.

alters:

subroutine **MUL816** 8-Bit × 16-Bit to 32-Bit Product Multiply

format: 2's Complement

input: r0 = multiplicand X r3, r2 = multiplier Y

output: r3, r2, r1, r0 = product P = X × Y (r3 = Sign Extension)

alters: PSW, r4, SignX and SignY

subroutine **MUL16** 16-Bit × 16-Bit to 32-Bit Product Multiply

format: 2's Complement

input: r1, r0 = multiplicand X r3, r2 = multiplier Y

output: r3, r2, r1, r0 = product P = X × Y.

alters: acc, PSW, SignX and SignY

subroutine **UMUL16** 16-Bit × 16-Bit to 32-Bit Product Multiply

format: unsigned

input: r1, r0 = multiplicand X r3, r2 = multiplier Y

output: r3, r2, r1, r0 = product P = X × Y.

alters: acc, PSW

subroutine **MAC16** 16-Bit × 16-Bit to 32-Bit Multiply-Accumulate

format: 2's Complement

input: r1, r0 = multiplicand X r3, r2 = multiplier Y

r7, r6, r5, r4 = accumulator Ar

output: r7, r6, r5, r4 = accumulated result Ar = Ar + (X × Y)
r3, r2, r1, r0 = product P = X × Y
Carry C is set if overflow
alters: acc, PSW, SignX and SignY

subroutine **DIV8** 8-Bit / 8-Bit to 8-Bit Quotient & Remainder Divide
format: 2's Complement
input: r0 = Dividend X r1 = Divisor Y
output: r0 = quotient Q of division Q = X / Y
r1 = remainder
alters: PSW, SignX and SignY

subroutine **UDIV8** 8-Bit / 8-Bit to 8-Bit Quotient & Remainder Divide
format: unsigned
input: r0 = Dividend X r1 = Divisor Y
output: r0 = quotient Q of division Q = X / Y r1 = remainder
alters: PSW

subroutine **DIV16** 16-Bit / 16-Bit to 16-Bit Quotient & Remainder Divide
format: 2's Complement
input: r1, r0 = Dividend X r3, r2 = Divisor Y
output: r1, r0 = quotient Q of division Q = X / Y r3, r2 = remainder
Carry C is set if Y = 0, i.e. divide by 0 attempted
alters: acc, PSW, r4, r5, r6, r7, SignX and SignY

subroutine **UDIV16** 16-Bit / 16-Bit to 16-Bit Quotient & Remainder Divide
format: unsigned
input: r1, r0 = Dividend X r3, r2 = Divisor Y
output: r1, r0 = quotient Q of division Q = X / Y r3, r2 = remainder
Carry C is set if Y = 0, i.e. divide by 0 attempted
alters: acc, PSW, B, dpl, dph, r4, r5, r6, r7

subroutine **DIV32** 32-Bit / 16-Bit to 32-Bit Quotient & Remainder Divide
format: 2's Complement
input: r3, r2, r1, r0 = Dividend X r5, r4 = Divisor Y

output: r3, r2, r1, r0 = quotient Q of division Q = X / Y r7, r6, r5, r4 = remainder
Carry C is set if Y = 0, i.e. divide by 0 attempted
alters: acc, PSW, SignX and SignY

subroutine **UDIV32** 32-Bit / 16-Bit to 32-Bit Quotient & Remainder Divide
format: unsigned
input: r3, r2, r1, r0 = Dividend X r5, r4 = Divisor Y
output: r3, r2, r1, r0 = quotient Q of division Q = X / Y r7, r6, r5, r4 = remainder
alters: acc, PSW

subroutine **MULDIV** 16-Bit × 16-Bit to 32-Bit Product Multiply followed by
32-Bit / 16-Bit to 32-Bit Quotient & Remainder Divide
format: 2's Complement
input: r1, r0 = multiplicand X r3, r2 = multiplier Y r5, r4 = divisor Z
output: r3, r2, r1, r0 = quotient Q of division Q = (X × Y) / Z r7, r6, r5, r4 = remainder
Carry C is set if Z = 0, i.e. divide by 0 attempted
alters: acc, PSW, SignX and SignY

subroutine **MACD16** 16-Bit × 16-Bit to 32-Bit Multiply-Accumulate & Data Move
y(n) = x(n) × h0 + x(n-1) × h1 + x(n-2) × h2 + ...
x(n-1) = x(n)
Note: Assumes shared program/data space. i.e. PSEN and RD are OR-ed together on the microcontroller board.
format: 2's Complement
input: B = No. of 16-Bit Items in Data Table (Max = 63)
DPTR New Input Data (e.g. from ADC)
DPTR+2 Base of Data Table (x)
DPTR+128 Base of Multiplier Table (h)
output: r7, r6, r5, r4 = 32-bit accumulated result Ar
alters: acc, PSW

subroutine **DELAY** Waits for a specified period
input: r0, r1, r2 = delay loop constants, r0 = coarse loop