ECE 375 LAB 5

Introduction to AVR Development Tools

Lab Time: Friday 2-4

Hao Truong

INTRODUCTION

The purpose of this lab is to understand and use the arithmetic instructions. We will create subroutines that perform arithmetic operations with large numbers (numbers that are larger than 8 bits, e.g. 16- and 24-bit numbers) such as addition, subtraction, and multiplication. The numbers are manipulated and handled using registers X, Y, and Z.

PROGRAM OVERVIEW

The Arithmetic program provides the basic arithmetic operations that allows performance for addition, subtraction, and multiplication with 16-bit numbers and 24-bit numbers. These numbers are stored in program memory. Since there is no direct path to move data from program memory to data memory, we will need to use pointers to load data from program memory and then store the data in data memory as operands, and then perform the calculations.

Besides the standard INIT and MAIN routines within the program, the additional routines will be created and used such as ADD16, SUB16, MUL24, and COMPOUND

INITIALIZATION ROUTINE

For this lab, only stack pointer will be initialized to allow the proper use of function and subroutine calls.

MAIN ROUTINE

The Main function sets up the function direct tests for ADD16, SUB16, MUL24, and COMPOUND. Data will be moved from program memory to data memory for used as input operands before performing addition, subtraction, or multiplication. Checkpoints will be set before and after function calls to test for correctness.

SUBROUTINES

1. ADD16 Routine

The ADD16 routine will perform addition with two 16-bit numbers. First we will need to load the address of the addition operands using X and Y pointers, perform the addition, and then store the result in memory.

2. SUB16 Routine

The SUB16 routine will perform unsigned subtraction with two 16-bit numbers. First we will need to load the address of the addition operands using X and Y pointers, perform the addition, and then store the result in memory.

3. MUL24 Routine

The MUL24 routine will perform multiplication for two 24-bit numbers. First we will need to load the address of the addition operands using X and Y pointers, perform the addition, and then store the result in memory. The result will be a 48-bit number

4. COMPOUND Routine

The COMPOUND routine perform a combination of ADD16, SUB16, and MUL24

ADDITIONAL QUESTIONS

1) Although we dealt with unsigned numbers in this lab, the ATmega128 micro- controller also has some features which are important for performing signed arithmetic. What does the V flag in the status register indicate? Give an example (in binary) of two 8-bit values that will cause the V flag to be set when they are added together.

The V flag in the SREG indicates the overflow resulted from an operation. V flag is set when overflow occurs, cleared otherwise.

1000 0000 + 1000 0000 = 0000 0000

2) In the skeleton file for this lab, the .BYTE directive was used to allocate some data memory locations for MUL16's input operands and result. What are some benefits of using this directive to organize your data memory, rather than just declaring some address constants using the .EQU directive?

With .BYTE directive, we can reserve byte to a variable. If we know how many bytes data is, it's easier to do this way.

With .EQU directive, it is used to assign a value to label. The down side of declaring some address constants using this directive is that we will need to do some calculation to figure out where to store data.

CONCLUSION

In this lab, we learned to create routines to perform arithmetic operations such as addition, subtraction, and multiplication with 16-bit and 24-bit numbers. These numbers were stored in program memory and required to be moved to data memory in order to perform the operations. We also learned how to allocate memory to store data in data memory.

SOURCE CODE

```
;***********************************************************
;*
;*      ;***********************************************************
;*
;*      Hao_Truong_Lab5_sourcecode.asm
;*
;*      Large Number Arithmetic
;*
;*      This is the skeleton file for Lab 5 of ECE 375
;*
;***********************************************************
;*
;*       Author: Hao Truong
;*         Date: 11/5/21
;*
;***********************************************************

.include "m128def.inc"               ; Include definition file

;***********************************************************
```

```
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                               ; Multipurpose register
.def    rlo = r0                                ; Low byte of MUL result
.def    rhi = r1                                ; High byte of MUL result
.def    zero = r2                               ;  Zero  register,  set  to  zero  in  INIT,  useful  for
calculations
.def    A = r3                                  ; A variable
.def    B = r4                                  ; Another variable

.def    oloop = r17                             ; Outer Loop Counter
.def    iloop = r18                             ; Inner Loop Counter
.def    count = r19                             ; counter


;***********************************************************
;*      Start of Code Segment
;***********************************************************
.cseg                                           ; Beginning of code segment

;-----------------------------------------------------------
; Interrupt Vectors
;-----------------------------------------------------------
.org    $0000                                   ; Beginning of IVs
                rjmp    INIT                    ; Reset interrupt

.org    $0046                                   ; End of Interrupt Vectors

;-----------------------------------------------------------
; Program Initialization
;-----------------------------------------------------------
INIT:                                           ; The initialization routine
                ; Initialize Stack Pointer
                ldi             mpr, low(RAMEND)
                out             SPL, mpr                ; Load SPL with low byte of RAMEND
                ldi             mpr, high(RAMEND)
                out             SPH, mpr                ; load SPH with high byte of RAMEND
                ; TODO                                  ; Init the 2 stack pointer registers

                clr             zero                    ; Set the zero register to zero, maintain
                                                        ; these semantics, meaning, don't
                                                        ; load anything else into it.

;-----------------------------------------------------------
; Main Program
;-----------------------------------------------------------
MAIN:                                           ; The Main program

                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ; Setup the ADD16 function direct test
                                ; Move values 0xFCBA and 0xFFFF in program memory to data memory
                                ; memory locations where ADD16 will get its inputs from
                                ; (see "Data Memory Allocation" section below)

                                ; move 0xFCBA from PM to DM
                                ldi             ZL, low(OperandD << 1)        ; extract low byte of
OperandD
                                ldi             ZH, high(OperandD << 1)               ; extract high
byte of OperandD
                                ; make Y point to low byte of ADD16_OP1
                                ldi             YL, low(ADD16_OP1)                    ;    load    low
byte of ADD16_OP1
                                ldi             YH, high(ADD16_OP1)                   ;   load   high
byte of ADD16_OP1
                                ; store OperandD to ADD16_OP1
                                lpm             mpr, Z+                                       ;   load
Z to mpr and then increment Z
                                st              Y+, mpr                                       ; store
mpr to Y (low byte of ADD16_OP1) and increment Y
                                lpm             mpr, Z+
```

```
                            st              Y+, mpr                                         ; store
high byte of OperandD to high byte of ADD16_OP1

                            ; move 0xFFFF from PM to DM
                            ldi             ZL, low(Operand2 << 1)          ; extract low byte of
Operand2
                            ldi             ZH, high(Operand2 << 1)                 ; extract high
byte of Operand2

                            ; make Y point to low byte of ADD16_OP2
                            ldi             YL, low(ADD16_OP2)                      ;    load    low
byte of ADD16_OP1
                            ldi             YH, high(ADD16_OP2)                     ;   load   high
byte of ADD16_OP1
                            lpm             mpr, Z+                                         ;   load
the first 8 bits (low byte) to mpr and then increment Z
                            st              Y+, mpr                                         ; store
low byte of Operand2 to low byte of ADD16_OP2
                            lpm             mpr, Z+                                         ;   load
high byte of Operand2 to mpr
                            st              Y+, mpr                                         ; store
high byte of Operand2 to high byte of ADD16_OP2



                nop ; Check load ADD16 operands (Set Break point here #1)
                            ; Call ADD16 function to test its correctness
                            rcall   ADD16
                            ; (calculate FCBA + FFFF) = 1FCB9

                nop ; Check ADD16 result (Set Break point here #2)
                            ; Observe result in Memory window
                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


                ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
                ; Setup the SUB16 function direct test
                            ; Move values 0xFCB9 and 0xE420 in program memory to data memory
                            ; memory locations where SUB16 will get its inputs from
                            ; (see "Data Memory Allocation" section below)

                            ; move 0xFCB9 from PM to DM
                            ldi             ZL, low(Operand3 << 1)          ; extract low byte of
Operand3
                            ldi             ZH, high(Operand3 << 1)                 ; extract high
byte of Operand3
                            ; make Y point to low byte of SUB16_OP1
                            ldi             YL, low(SUB16_OP1)                      ;    load    low
byte of SUB16_OP1
                            ldi             YH, high(SUB16_OP1)                     ;   load   high
byte of SUB16_OP1
                            ; store Operand3 to SUB16_OP1
                            lpm             mpr, Z+                                         ;   load
Z to mpr and then increment Z
                            st              Y+, mpr                                         ; store
mpr to Y (low byte of SUB16_OP1) and increment Y
                            lpm             mpr, Z+
                            st              Y+, mpr                                         ; store
high byte of Operand3 to high byte of SUB16_OP1

                            ; move 0xE420 from PM to DM
                            ldi             ZL, low(Operand4 << 1)          ; extract low byte of
Operand4
                            ldi             ZH, high(Operand4 << 1)                 ; extract high
byte of Operand4
                            ; make Y point to low byte of SUB16_OP2
                            ldi             YL, low(SUB16_OP2)                      ;    load    low
byte of SUB16_OP1
                            ldi             YH, high(SUB16_OP2)                     ;   load   high
byte of SUB16_OP1
```

```
                            lpm             mpr, Z+                                         ;  load
the first 8 bits (low byte) to mpr and then increment Z
                            st              Y+, mpr                                         ; store
low byte of Operand4 to low byte of SUB16_OP2
                            lpm             mpr, Z+                                         ;  load
high byte of Operand4 to mpr
                            st              Y+, mpr                                         ; store
high byte of Operand4 to high byte of SUB16_OP2



                nop ; Check load SUB16 operands (Set Break point here #3)
                            ; Call SUB16 function to test its correctness
                            rcall   SUB16
                            ; (calculate FCB9 - E420)     = 0x1899

                nop ; Check SUB16 result (Set Break point here #4)
                            ; Observe result in Memory window
               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
               ; Setup the MUL24 function direct test

                            ;  Move  values  0xFFFFFF  and  0xFFFFFF  in  program  memory  to  data
memory
                            ; memory locations where MUL24 will get its inputs from
                            ; (see "Data Memory Allocation" section below)

                            ; move 0xFFFFFF from PM to DM
                            ldi             ZL, low(Operand5 << 1)        ; extract low byte of
Operand5
                            ldi             ZH, high(Operand5 << 1)               ; extract high
byte of Operand5
                            ; make Y point to low byte of SUB16_OP1
                            ldi             YL, low(MUL24_OP1)                    ;    load    low
byte of MUL24_OP1
                            ldi             YH, high(MUL24_OP1)                   ;    load    high
byte of MUL24_OP1
                            ; store Operand5 to MUL24_OP1
                            lpm             mpr, Z+                                         ;  load
Z to mpr and then increment Z
                            st              Y+, mpr                                         ; store
mpr to Y (first byte of MUL24_OP1) and increment Y
                            lpm             mpr, Z+
                            st              Y+, mpr                                         ; store
second byte of Operand5 in second byte of MUL24_OP1
                            lpm             mpr, Z
                            st              Y, mpr                                          ; store
last byte of Operand5 in last byte of MUL24_OP1

                            ; move 0xFFFFFF from PM to DM
                            ldi             ZL, low(Operand5 << 1)        ; extract low byte of
Operand5
                            ldi             ZH, high(Operand5 << 1)               ; extract high
byte of Operand5
                            ; make Y point to low byte of SUB16_OP1
                            ldi             YL, low(MUL24_OP2)                    ;    load    low
byte of MUL24_OP2
                            ldi             YH, high(MUL24_OP2)                   ;    load    high
byte of MUL24_OP2
                            ; store Operand5 to MUL24_OP2
                            lpm             mpr, Z+                                         ;  load
Z to mpr and then increment Z
                            st              Y+, mpr                                         ; store
mpr to Y (first byte of MUL24_OP2) and increment Y
                            lpm             mpr, Z+
                            st              Y+, mpr                                         ; store
second byte of Operand5 in second byte of MUL24_OP2
                            lpm             mpr, Z
```

```
                            st              Y, mpr                                          ; store
last byte of Operand5 in last byte of MUL24_OP2

                nop ; Check load MUL24 operands (Set Break point here #5)
                            ; Call MUL24 function to test its correctness
                            rcall   MUL24
                            ; (calculate FFFFFF * FFFFFF) = FFFFFE000001

                nop ; Check MUL24 result (Set Break point here #6)
                            ; Observe result in Memory window
               ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


                nop ; Check load COMPOUND operands (Set Break point here #7)
               ; Call the COMPOUND function
                            rcall   COMPOUND

                nop ; Check COMPUND result (Set Break point here #8)
                            ; Observe final result in Memory window

DONE:   rjmp    DONE                    ; Create an infinite while loop to signify the
                                                        ; end of the program.

;***********************************************************
;*      Functions and Subroutines
;***********************************************************

;-----------------------------------------------------------
; Func: ADD16
; Desc: Adds two 16-bit numbers and generates a 24-bit number
;               where the high byte of the result contains the carry
;               out bit.
;-----------------------------------------------------------
ADD16:
                push    mpr
                push    A                                       ; save A
                push    B                                       ; save B
                push    XL                                      ; save X ptr
                push    XH
                push    YL                                      ; save Y ptr
                push    YH
                push    ZL                                      ; save Z ptr
                push    ZH

                ; Load beginning address of first operand into X
                ldi             XL, low(ADD16_OP1)      ; Load low byte of address
                ldi             XH, high(ADD16_OP1)     ; Load high byte of address

                ; Load beginning address of second operand into Y
                ldi             YL, low(ADD16_OP2)      ; load low byte of address
                ldi             YH, high(ADD16_OP2)     ; load high byte of address

                ; Load beginning address of result into Z
                ldi             ZL, low(ADD16_Result)
                ldi             ZH, high(ADD16_Result)

                ; Execute the function
                ld              A, X+                   ;  load  low  byte  of  ADD16_OP1  to  A  and
increment X
                ld              B, Y+                   ;  load  low  byte  of  ADD16_OP2  to  B  and
increment Y
                add             B, A                    ; perform addition and store result back to
B
                st              Z+, B                   ;  store  result  in  Z  (first  byte  of
ADD16_Result) and increment Z
                ld              A, X                    ; load high byte of ADD16_OP1 to A
                ld              B, Y                    ; load high byte of ADD16_OP2 to B
                adc             B, A                    ; perform addition with carry
                st              Z+, B                   ; store result in Z  and increment Z
                brcc    EXIT                    ; jump to EXIT if carry cleared
                ldi             mpr, $01                ; load 1 to mpr if carry is set
```

```
                st              Z, mpr                  ; store carry to Z
EXIT:

                ; restore all registers in reverse order
                pop             ZH
                pop             ZL
                pop             YH
                pop             YL
                pop             XH
                pop             XL
                pop             B
                pop             A
                pop             mpr
                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: SUB16
; Desc: Subtracts two 16-bit numbers and generates a 16-bit
;               result.
;-----------------------------------------------------------
SUB16:
                push    mpr
                push    A                                       ; save A
                push    B                                       ; save B
                push    XL                                      ; save X ptr
                push    XH
                push    YL                                      ; save Y ptr
                push    YH
                push    ZL                                      ; save Z ptr
                push    ZH

                ; Load beginning address of first operand into X
                ldi             XL, low(SUB16_OP1)      ; Load low byte of address
                ldi             XH, high(SUB16_OP1)     ; Load high byte of address

                ; Load beginning address of second operand into Y
                ldi             YL, low(SUB16_OP2)      ; load low byte of address
                ldi             YH, high(SUB16_OP2)     ; load high byte of address

                ; Load beginning address of result into Z
                ldi             ZL, low(SUB16_Result)
                ldi             ZH, high(SUB16_Result)

                ; Execute the function
                ld              A, X+                   ;  load  low  byte  of  ADD16_OP1  to  A  and
increment X
                ld              B, Y+                   ;  load  low  byte  of  ADD16_OP2  to  B  and
increment Y
                sub             A, B                    ; perform addition and store result back to
A
                st              Z+, A                   ;   store   result   in   Z   (first   byte   of
ADD16_Result) and increment Z
                ld              A, X                    ; load high byte of ADD16_OP1 to A
                ld              B, Y                    ; load high byte of ADD16_OP2 to B
                sbc             A, B                    ; perform addition with carry
                st              Z+, A                   ; store result in Z  and increment Z
;               brcc    EXIT                    ; jump to EXIT if carry cleared
;               ldi             mpr, $01                ; load 1 to mpr if carry is set
;               st              Z, mpr                  ; store carry to Z
;EXIT:

                ; restore all registers in reverse order
                pop             ZH
                pop             ZL
                pop             YH
                pop             YL
                pop             XH
                pop             XL
                pop             B
                pop             A
                pop             mpr
```

```
                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: MUL24
; Desc: Multiplies two 24-bit numbers and generates a 48-bit
;               result.
;-----------------------------------------------------------
MUL24:
                ; Execute the function here
                push    A                               ; Save A register
                push    B                               ; Save B register
                push    rhi                             ; Save rhi register
                push    rlo                             ; Save rlo register
                push    zero                    ; Save zero register
                push    XH                              ; Save X-ptr
                push    XL
                push    YH                              ; Save Y-ptr
                push    YL
                push    ZH                              ; Save Z-ptr
                push    ZL
                push    oloop                   ; Save counters
                push    iloop

                clr             zero                    ; Maintain zero semantics

                ; Set Y to beginning MUL24_OP2
                ldi             YL, low(MUL24_OP2)      ; Load low byte
                ldi             YH, high(MUL24_OP2)     ; Load high byte

                ; Set Z to begginning address of resulting Product
                ldi             ZL, low(MUL24_Result)  ; Load low byte
                ldi             ZH, high(MUL24_Result); Load high byte


                ; Begin outer for loop
                ldi             oloop, 3                ; Load counter
MUL24_OLOOP:
                ; Set X to beginning address of A
                ldi             XL, low(MUL24_OP1)      ; Load low byte
                ldi             XH, high(MUL24_OP1)     ; Load high byte

                ; Begin inner for loop
                ldi             iloop, 3                ; Load counter
MUL24_ILOOP:
                ld              A, X+                   ; Get byte of A operand
                ld              B, Y                    ; Get byte of B operand
                mul             A,B                             ; Multiply A and B
                ld              A, Z+                   ; Get a result byte from memory
                ld              B, Z+                   ; Get the next result byte from memory
                add             rlo, A                  ; rlo <= rlo + A
                adc             rhi, B                  ; rhi <= rhi + B + carry
                ld              A, Z                    ; Get a third byte from the result
                adc             A, zero                 ; Add carry to A
                st              Z, A                    ; Store third byte to memory
                st              -Z, rhi                 ; Store second byte to memory
                st              -Z, rlo                 ; Store first byte to memory
                adiw    ZH:ZL, 1                ; Z <= Z + 1
                dec             iloop                   ; Decrement counter
                brne    MUL24_ILOOP             ; Loop if iLoop != 0
                ; End inner for loop

                sbiw    ZH:ZL, 2                ; Z <= Z - 2
                adiw    YH:YL, 1                ; Y <= Y + 1
                dec             oloop                   ; Decrement counter
                brne    MUL24_OLOOP             ; Loop if oLoop != 0
                ; End outer for loop

                pop             iloop                   ; Restore all registers in reverves order
                pop             oloop
```

```
                pop             ZL
                pop             ZH
                pop             YL
                pop             YH
                pop             XL
                pop             XH
                pop             zero
                pop             rlo
                pop             rhi
                pop             B
                pop             A


                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: COMPOUND
; Desc: Computes the compound expression ((D - E) + F)^2
;               by making use of SUB16, ADD16, and MUL24.
;
;               D, E, and F are declared in program memory, and must
;               be moved into data memory for use as input operands.
;
;               All result bytes should be cleared before beginning.
;-----------------------------------------------------------
COMPOUND:
                push    mpr
                push    A                                       ; save A
                push    B                                       ; save B
                push    XL                                      ; save X ptr
                push    XH
                push    YL                                      ; save Y ptr
                push    YH
                push    ZL                                      ; save Z ptr
                push    ZH

                ; Setup SUB16 with operands D and E
                                ; move OperandD from PM to DM
                                ldi             ZL, low(OperandD << 1)          ; extract low byte of
OperandD
                                ldi             ZH, high(OperandD << 1)                 ; extract high
byte of OperandD
                                ; make Y point to low byte of SUB16_OP1
                                ldi             YL, low(SUB16_OP1)                      ;    load    low
byte of SUB16_OP1
                                ldi             YH, high(SUB16_OP1)                     ;   load    high
byte of SUB16_OP1
                                ; store OperandD to SUB16_OP1
                                lpm             mpr, Z+                                         ;   load
Z to mpr and then increment Z
                                st              Y+, mpr                                         ; store
mpr to Y (low byte of SUB16_OP1) and increment Y
                                lpm             mpr, Z+
                                st              Y+, mpr                                         ; store
high byte of Operand3 to high byte of SUB16_OP1

                                ; move OperandE from PM to DM
                                ldi             ZL, low(OperandE << 1)          ; extract low byte of
OperandE
                                ldi             ZH, high(OperandE << 1)                 ; extract high
byte of OperandE
                                ; make Y point to low byte of SUB16_OP2
                                lpm             mpr, Z+                                         ;   load
the first 8 bits (low byte) to mpr and then increment Z
                                st              Y+, mpr                                         ; store
low byte of Operand4 to low byte of SUB16_OP2
                                lpm             mpr, Z+                                         ;   load
high byte of Operand4 to mpr
                                st              Y+, mpr                                         ; store
high byte of Operand4 to high byte of SUB16_OP2
                ; Perform subtraction to calculate D - E
```

```
                        rcall SUB16

            ; Setup the ADD16 function with SUB16 result and operand F
                        ldi         ZL, low(SUB16_Result)       ; extract low byte of
SUB16_Result
                        ldi         ZH, high(SUB16_Result)      ; extract high byte of
SUB16_Result
                        ; make Y point to low byte of ADD16_OP1
                        ldi         YL, low(ADD16_OP1)                  ;   load   low
byte of ADD16_OP1
                        ldi         YH, high(ADD16_OP1)                 ;   load   high
byte of ADD16_OP1
                        ; store SUB16_Result to ADD16_OP1
                        ld          mpr, Z+
                        st          Y+, mpr                                     ; store
mpr to Y (low byte of SUB16_OP1) and increment Y
                        ld          mpr, Z+
                        st          Y+, mpr                                     ; store
high byte of Operand3 to high byte of SUB16_OP1

                        ; move OperandF from PM to DM
                        ldi         ZL, low(OperandF << 1)      ; extract low byte of
OperandF
                        ldi         ZH, high(OperandF << 1)             ; extract high
byte of OperandF
                        ; make Y point to low byte of ADD16_OP2
                        lpm         mpr, Z+                                     ;  load
the first 8 bits (low byte) to mpr and then increment Z
                        st          Y+, mpr                                     ; store
low byte of Operand4 to low byte of SUB16_OP2
                        lpm         mpr, Z+                                     ;  load
high byte of Operand4 to mpr
                        st          Y+, mpr                                     ; store
high byte of Operand4 to high byte of SUB16_OP2


            ; Perform addition next to calculate (D - E) + F
                        rcall ADD16

            ; Setup the MUL24 function with ADD16 result as both operands
                        ; move ADD16_Result from PM to DM
                        ldi         ZL, low(ADD16_Result)       ; extract low byte of
ADD16_Result
                        ldi         ZH, high(ADD16_Result)      ; extract high byte of
ADD16_Result
                        ; make Y point to low byte of MUL24_OP1
                        ldi         YL, low(MUL24_OP1)                  ;   load   low
byte of MUL24_OP1
                        ldi         YH, high(MUL24_OP1)                 ;   load   high
byte of MUL24_OP1
                        ; store Operand5 to MUL24_OP1
                        ld          mpr, Z+                                     ;  load
Z to mpr and then increment Z
                        st          Y+, mpr                                     ; store
mpr to Y (first byte of MUL24_OP1) and increment Y
                        ld          mpr, Z+
                        st          Y+, mpr                                     ; store
second byte of Operand5 in second byte of MUL24_OP1
                        ld          mpr, Z
                        st          Y, mpr                                      ; store
last byte of Operand5 in last byte of MUL24_OP1

                        ; move ADD16_Result from PM to DM
                        ldi         ZL, low(ADD16_Result)       ; extract low byte of
ADD16_Result
                        ldi         ZH, high(ADD16_Result)      ; extract high byte of
ADD16_Result
                        ; make Y point to low byte of MUL24_OP2
                        ldi         YL, low(MUL24_OP2)                  ;   load   low
byte of MUL24_OP2
```

```
                            ldi             YH, high(MUL24_OP2)                 ;   load   high
byte of MUL24_OP2
                            ; store Operand5 to MUL24_OP2
                            ld              mpr, Z+                                     ;   load
Z to mpr and then increment Z
                            st              Y+, mpr                                     ; store
mpr to Y (first byte of MUL24_OP2) and increment Y
                            ld              mpr, Z+
                            st              Y+, mpr                                     ; store
second byte of Operand5 in second byte of MUL24_OP2
                            ld              mpr, Z
                            st              Y, mpr                                      ; store
last byte of Operand5 in last byte of MUL24_OP2

                ; Perform multiplication to calculate ((D - E) + F)^2
                            rcall   MUL24

                ; restore all registers in reverse order
                pop             ZH
                pop             ZL
                pop             YH
                pop             YL
                pop             XH
                pop             XL
                pop             B
                pop             A
                pop             mpr

                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: MUL16
; Desc: An example function that multiplies two 16-bit numbers
;                       A - Operand A is gathered from address $0101:$0100
;                       B - Operand B is gathered from address $0103:$0102
;                       Res - Result is stored in address
;                                       $0107:$0106:$0105:$0104
;               You will need to make sure that Res is cleared before
;               calling this function.
;-----------------------------------------------------------
MUL16:
                push    A                               ; Save A register
                push    B                               ; Save B register
                push    rhi                             ; Save rhi register
                push    rlo                             ; Save rlo register
                push    zero                    ; Save zero register
                push    XH                              ; Save X-ptr
                push    XL
                push    YH                              ; Save Y-ptr
                push    YL
                push    ZH                              ; Save Z-ptr
                push    ZL
                push    oloop                   ; Save counters
                push    iloop

                clr             zero                    ; Maintain zero semantics

                ; Set Y to beginning address of B
                ldi             YL, low(addrB) ; Load low byte
                ldi             YH, high(addrB)         ; Load high byte

                ; Set Z to begginning address of resulting Product
                ldi             ZL, low(LAddrP)         ; Load low byte
                ldi             ZH, high(LAddrP); Load high byte

                ; Begin outer for loop
                ldi             oloop, 2                ; Load counter
MUL16_OLOOP:
                ; Set X to beginning address of A
                ldi             XL, low(addrA) ; Load low byte
                ldi             XH, high(addrA)         ; Load high byte
```

```
                ; Begin inner for loop
                ldi             iloop, 2                ; Load counter
MUL16_ILOOP:
                ld              A, X+                   ; Get byte of A operand
                ld              B, Y                    ; Get byte of B operand
                mul             A,B                             ; Multiply A and B
                ld              A, Z+                   ; Get a result byte from memory
                ld              B, Z+                   ; Get the next result byte from memory
                add             rlo, A                  ; rlo <= rlo + A
                adc             rhi, B                  ; rhi <= rhi + B + carry
                ld              A, Z                    ; Get a third byte from the result
                adc             A, zero                 ; Add carry to A
                adc             A, zero                 ; Add carry to A
                st              Z, A                    ; Store third byte to memory
                st              -Z, rhi                 ; Store second byte to memory
                st              -Z, rlo                 ; Store first byte to memory
                adiw    ZH:ZL, 1                ; Z <= Z + 1
                dec             iloop                   ; Decrement counter
                brne    MUL16_ILOOP             ; Loop if iLoop != 0
                ; End inner for loop

                sbiw    ZH:ZL, 1                ; Z <= Z - 1
                adiw    YH:YL, 1                ; Y <= Y + 1
                dec             oloop                   ; Decrement counter
                brne    MUL16_OLOOP             ; Loop if oLoop != 0
                ; End outer for loop

                pop             iloop                   ; Restore all registers in reverves order
                pop             oloop
                pop             ZL
                pop             ZH
                pop             YL
                pop             YH
                pop             XL
                pop             XH
                pop             zero
                pop             rlo
                pop             rhi
                pop             B
                pop             A
                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: Template function header
; Desc: Cut and paste this and fill in the info at the
;               beginning of your functions
;-----------------------------------------------------------
FUNC:                                                   ; Begin a function with a label
                ; Save variable by pushing them to the stack

                ; Execute the function here

                ; Restore variable by popping them from the stack in reverse order
                ret                                             ; End a function with RET


;***********************************************************
;*      Stored Program Data
;***********************************************************

; Enter any stored data you might need here

; ADD16 operands
Operand2:
        .DW     0xFFFF

; SUB16 operands
Operand3:
        .DW     0xFCB9
Operand4:
```

```
        .DW 0xE420

; MUL24 operands
Operand5:
        .DD 0xFFFFFF


; Compoud operands
OperandD:
        .DW     0xFCBA                      ; test value for operand D
OperandE:
        .DW     0x2019                      ; test value for operand E
OperandF:
        .DW     0x21BB                      ; test value for operand F

;***********************************************************
;*      Data Memory Allocation
;***********************************************************

.dseg
.org    $0100                       ; data memory allocation for MUL16 example
addrA:  .byte 2
addrB:  .byte 2
LAddrP: .byte 4

; Below is an example of data memory allocation for ADD16.
; Consider using something similar for SUB16 and MUL24.

.org    $0110                       ; data memory allocation for operands
ADD16_OP1:
                .byte 2                     ; allocate two bytes for first operand of ADD16
ADD16_OP2:
                .byte 2                     ; allocate two bytes for second operand of ADD16
SUB16_OP1:
                .byte 2                     ; allocate 2 bytes for first operand of SUB16
SUB16_OP2:
                .byte 2                     ; allocate 2 bytes for second operand of SUB16

.org    $0120                       ; data memory allocation for results
ADD16_Result:
                .byte 3                     ; allocate three bytes for ADD16 result
SUB16_Result:
                .byte 2                     ; allocate 2 bytes for SUB16 result

.org    $0130                       ; data memory allocation for MUL24 operands
MUL24_OP1:
                .byte 3                     ; allocate 3 bytes for first operand of MUL24
MUL24_OP2:
                .byte 3                     ; allocate 3 bytes for second operand  of MUL24

.org    $0140                       ; data memory allocation for MUL24 Result
MUL24_Result:
                .byte 6                     ; allocate 6 bytes for MUL24 Result

;***********************************************************
;*      Additional Program Includes
;***********************************************************
; There are no additional file includes for this program
```