ECE 375 LAB 5

Large Number Arithmetic

Lab Time: Wednesday 10-12

Tu Lam

INTRODUCTION

The purpose of the fifth lab is to write different types of function where it is doing arithmetic on basic function such as addition, subtraction, and multiplication. In this program you would deal with number larger than 8-bit value and how to handle the data as the data is being manipulated. The program also performs a compound that combine input from other function to make it compound.

PROGRAM OVERVIEW

The Arithmetic program overall will perform basic arithmetic function such as addition, subtraction, multiplication. There is data manipulation on how to handle bigger than 8-bit data. The program will also use compound function to set to use other function of ADD16, SUB16, and etc.. to build on this compound function.

Besides the standard MAIN routines within the program, there will be an INIT initialize stage where we will set up the stack pointer from it.

INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of the stack pointer to set up where to point in the program.

MAIN ROUTINE

The Main routine executes a simple statement of calling other functions such as ADD16, MUL24, SUB16, and etc… and it will set up the direct test by passing data from program memory into data memory for testing to see if the result is correct. There are breakpoints for the TA to check to see the result is correct.

ADD16 ROUTINE

The ADD16 routine will look at the data memory that was initialize in the MAIN program and it maniputlate data and do the addition arithmetic. Then it will store it in the result operand in IRAM.

SUB16 ROUTINE

The SUB16 routine is identical to the ADD16 routine, except that it will perform the task a little bit differently. This function will subtract two operands from each other and store the result on data memory.

MUL24 ROUTINE

The MUL24 routine will take two 24-bit value and multiplying it into a 48-bit value as the result. The routine is almost the same reflection off the MUL16 Routine.

COMPOUND ROUTINE

The COMPOUND routine is utilizing the other functions that was created in the program to add, subtract, and multiply different operand value. This program needs to also set up a program to data memory.

ADDITIONAL QUESTIONS

1) Although we dealt with unsigned numbers in this lab, the ATmega128 microcontroller 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.

V flag is indicated that an overflow flag is active and an example could cause an V flag to activate is doing 0100 + 0100 = 1000.

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?

Using directive to organize your data memory as it is easier for you as the programmer to know where your data is store and you get to choose where to store it and know the location to manipulated. With .EQU, the data can change toward that directive and it is hard to store it as data can be wipe out with another data by overwriting it.

CONCLUSION

For Lab #5, the lab helps to demonstrate on how AVR assembly works to do simple arithmetic and how we can handle more bit data than just 16 bit that the AVR can handle. The lab was simple and easy to follow along and require past lab knowledge to implement the direct test from the MAIN program. Overall, this lab taught how the step of arithmetic is performing in AVR assembly which help us understand the background work it is doing if we were coding in other language such as C/C++.

SOURCE CODE

Provide a copy of the source code. Here you should use a mono-spaced font and can go down to 8-pt in order to make it fit. Sometimes the conversion from standard ASCII to a word document may mess up the formatting. Make sure to reformate the code so it looks nice and is readable.

```
;***********************************************************
;*
;*      Tu_Lam_Lab5_SourceCode.asm
;*
;*      Description: This file contains function that help to do
;*                          addition, subtraction, and multiplying
;*                   more than 8-bit value.
;*
;***********************************************************
;*
;*       Author: Tu Lam
;*         Date: November 1st, 2020
;*
;***********************************************************

.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    counter = r20                                   ; Counter variable

.def    oloop = r17                                             ; Outer Loop Counter
.def    iloop = r18                                             ; Inner Loop 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
                ldi             mpr, low(RAMEND)        ; Initialize Stack Pointer
                out             SPL, mpr
                ldi             mpr, high(RAMEND)
                out             SPH, mpr
                ; 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
                ldi             ZL, low(OperandD << 1)  ; Move values 0xFCBA and 0xFFFF in program memory
to data memory
                ldi             ZH, high(OperandD << 1) ; memory locations where ADD16 will get its
inputs from
                ldi             YL, low(ADD16_OP1)              ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(ADD16_OP1)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 2                              ; Make counter hold value of 2

LOOP1:
                lpm             mpr, Z+                                 ; Load the value 0xFCBA into mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOP1

                ldi             ZL, low(OperandA << 1)  ; Move 0xFFFF in program memory to data memory
                ldi             ZH, high(OperandA << 1) ; where ADD16 will get its inputs from
```

```
                ldi             YL, low(ADD16_OP2)              ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(ADD16_OP2)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 2                              ; Make counter hold value of 2
for looping two times

LOOP2:
                lpm             mpr, Z+                                 ; Load the value 0xFFFF into mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOP2

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

                nop ; Check ADD16 result (Set Break point here #2)

                                                                                ; Observe result in
Memory window

                ; Setup the SUB16 function direct test
                ldi             ZL, low(OperandB << 1)   ; Move values 0xFCB9 in program memory to data
memory
                ldi             ZH, high(OperandB << 1)  ; memory locations where SUB16 will get its
inputs from
                ldi             YL, low(SUB16_OP1)              ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(SUB16_OP1)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 2                              ; Make counter hold value of 2

LOOP3:
                lpm             mpr, Z+                                 ; Load the value 0xFCB9 into mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOP3

                ldi             ZL, low(OperandC << 1)   ; Move 0xE420 in program memory to data memory
                ldi             ZH, high(OperandC << 1)  ; where SUB16 will get its inputs from
                ldi             YL, low(SUB16_OP2)              ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(SUB16_OP2)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 2                              ; Make counter hold value of 2
for looping two times

LOOP4:
                lpm             mpr, Z+                                 ; Load the value 0xE420 into mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOP4

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

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

```
                ; Setup the MUL24 function direct test
                ldi             ZL, low(OperandG << 1)   ; Move values 0xFFFFFF in program memory to data
memory
                ldi             ZH, high(OperandG << 1)  ; memory locations where MUL24 will get its
inputs from
                ldi             YL, low(addrC)                  ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(addrC)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 3                              ; Make counter hold value of 3

LOOP5:
                lpm             mpr, Z+                                 ; Load the value 0xFFFFFF into
mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOP5

                ldi             ZL, low(OperandG << 1)   ; Move 0xFFFFFF in program memory to data memory
                ldi             ZH, high(OperandG << 1)  ; where MUL24 will get its inputs from
                ldi             YL, low(addrD)                  ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(addrD)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 3                              ; Make counter hold value of 3
for looping three times

LOOP6:
                lpm             mpr, Z+                                 ; Load the value 0xFFFFFF into
mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOP6

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

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

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

                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:
                ; 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 into Y-
register
                ldi             YH, high(ADD16_OP2)             ; Load high byte of address into Y-
register

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

                ; Execute the function
                ld              A, X+                                   ; Load the low byte X into A and
move to the high byte
                ld              B, Y+                                   ; Load the low byte Y into B and
move to the high byte
                add             A, B                                    ; Add value A and B (low byte)
together and store it in B
                st              Z+, A                                   ; Store value of low byte into Z
and increment to the high byte

                ld              A, X                                    ; Load high byte X into A
                ld              B, Y                                    ; Load high byte Y into B
                adc             A, B                                    ; Add the value of A & B (high
byte) together and store in B w/ carry
                st              Z+, A                                   ; Store value into the high byte
of Z and increment to the next register

                brcc    DONE_JOB                        ; Check if the carry flag is clear, if so, jump
to exit
                ldi             mpr, 1                                  ; If not load 1 into the mpr
                st              Z, mpr                                  ; Store that carry into the Z-
register

DONE_JOB:
                ret                                                             ; End a function with
RET

;-----------------------------------------------------------
; Func: SUB16
; Desc: Subtracts two 16-bit numbers and generates a 16-bit
;               result.
;-----------------------------------------------------------
SUB16:
                ; 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 into Y-
register
                ldi             YH, high(SUB16_OP2)             ; Load high byte of address into Y-
register

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

                ; Execute the function
                ld              A, X+                                   ; Load the low byte X into A and
move to the high byte
                ld              B, Y+                                   ; Load the low byte Y into B and
move to the high byte
                sub             A, B                                    ; Subtract value A from B (low
byte) together and store it in A
                st              Z+, A                                   ; Store value of low byte into Z
and increment to the high byte
```

```
                ld              A, X                                    ; Load high byte X into A
                ld              B, Y                                    ; Load high byte Y into B
                sbc             A, B                                    ; Subtract the value of A from B
(high byte) together and store in A w/ carry
                st              Z+, A                                   ; Store value into the high byte
of Z and increment to the next register


                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
                clr             zero                                    ; Maintain zero semantics

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

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

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

                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:
                ; Clear the Result
                clr             mpr
                ldi             XL, low(SUB16_Result)
                ldi             XH, high(SUB16_Result)
                clr             counter
                ldi             counter, 2
CLEAR1:
                st              X+, mpr
                dec             counter
                brne    CLEAR1

                clr             mpr
                ldi             XL, low(ADD16_Result)
                ldi             XH, high(ADD16_Result)
                clr             counter
                ldi             counter, 3
CLEAR2:
                st              X+, mpr
                dec             counter
                brne    CLEAR2

                clr             mpr
                ldi             XL, low(LAddrE)
                ldi             XH, high(LAddrE)
                clr             counter
                ldi             counter, 6
CLEAR3:
                st              X+, mpr
                dec             counter
                brne    CLEAR3

                ldi             ZL, low(OperandD << 1)   ; Move values 0xFCBA in program memory to data
memory
                ldi             ZH, high(OperandD << 1)  ; memory locations where SUB16 will get its
inputs from
                ldi             YL, low(SUB16_OP1)              ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(SUB16_OP1)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 2                              ; Make counter hold value of 2

LOOPD:
                lpm             mpr, Z+                                 ; Load the value 0xFCBA into mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOPD

                ldi             ZL, low(OperandE << 1)   ; Move 0x2019 in program memory to data memory
                ldi             ZH, high(OperandE << 1)  ; where SUB16 will get its inputs from
                ldi             YL, low(SUB16_OP2)              ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(SUB16_OP2)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 2                              ; Make counter hold value of 2
for looping two times

LOOPE:
                lpm             mpr, Z+                                 ; Load the value 0x2019 into mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
```

```
                dec             counter                                 ; Decrement counter
                brne    LOOPE

                rcall   SUB16                                   ; Setup SUB16 with operands D and E
                                                                        ; Perform subtraction to
calculate D - E

                ldi             ZL, low(SUB16_Result << 1)      ; Move values result from sub in program
memory to data memory
                ldi             ZH, high(SUB16_Result << 1)     ; memory locations where ADD16 will get
its inputs from
                ldi             YL, low(ADD16_OP1)              ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(ADD16_OP1)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 2                              ; Make counter hold value of 2

LOOPS:
                lpm             mpr, Z+                                 ; Load the value result from
subtraction into mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOPS

                ldi             ZL, low(OperandF << 1)   ; Move 0x21BB in program memory to data memory
                ldi             ZH, high(OperandF << 1)  ; where ADD16 will get its inputs from
                ldi             YL, low(ADD16_OP2)              ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(ADD16_OP2)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 2                              ; Make counter hold value of 2
for looping two times

LOOPF:
                lpm             mpr, Z+                                 ; Load the value 0x21BB into mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOPF

                rcall   ADD16                                   ; Setup the ADD16 function with SUB16
result and operand F
                                                                        ; Perform addition next
to calculate (D - E) + F

                ldi             ZL, low(ADD16_Result << 1)      ; Move values of ADD result in program
memory to data memory
                ldi             ZH, high(ADD16_Result << 1)     ; memory locations where MUL24 will get
its inputs from
                ldi             YL, low(addrC)                  ; (see "Data Memory Allocation" section
below)
                ldi             YH, high(addrC)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 3                              ; Make counter hold value of 3

LOOPA:
                lpm             mpr, Z+                                 ; Load the value of ADD result
into mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOPA

                ldi             ZL, low(ADD16_Result << 1)      ; Move addition result in program memory
to data memory
                ldi             ZH, high(ADD16_Result << 1)     ; where MUL24 will get its inputs from
                ldi             YL, low(addrD)                  ; (see "Data Memory Allocation" section
below)
```

```
                ldi             YH, high(addrD)
                clr             counter                                 ; Make sure counter is at 0
                ldi             counter, 3                              ; Make counter hold value of 3
for looping three times

LOOPB:
                lpm             mpr, Z+                                 ; Load the value of addtion into
mpr
                st              Y+, mpr                                 ; Store the value in the Y-
register
                dec             counter                                 ; Decrement counter
                brne    LOOPB

                rcall   MUL24                                   ; Setup the MUL24 function with ADD16
result as both operands
                                                                        ; Perform multiplication
to calculate ((D - E) + F)^2

                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
                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
OperandA:
        .DW 0xFFFF                                              ; test value for ADD16

; SUB16 operands
OperandB:
```

```
        .DW 0xFCB9                                              ; test value for SUB16
OperandC:
        .DW     0xE420

; MUL24 operands
OperandG:                                                       ; test value for MUL24
        .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

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

.org    $0130                                           ; data memory allocation for operands
SUB16_OP1:
                .byte 2                                         ; allocate two bytes for first
operand of SUB16
SUB16_OP2:
                .byte 2                                         ; allocate two bytes for second
operand of SUB16

.org    $0140                                           ; data memory allocation for results
SUB16_Result:
                .byte 2                                         ; allocate two bytes for SUB16
result

.org    $0200                                           ; data memory allocation for MUL24
example
addrC:  .byte 3
addrD:  .byte 3
LAddrE: .byte 6

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