ECE 375 LAB 5

Large Number Arithmetic

Lab Time: Tuesday 5-7pm

Bianca Beauchamp

Drake Vidkjer

INTRODUCTION

In this lab we worked on multi-byte addition by implementing 16 bit addition, multi-byte subtraction by implementing 16 bit subtraction, multibyte multiplication by implementing 24 bit multiplication and a compound function that evaluates an expression using the addition, subtraction and multiplication all at once. The order in which we did this lab made it easier because once the addition was understood, the subtraction and multiplication were easier to understand and implement because the use addition in order to do the operations.

PROGRAM OVERVIEW

In All the Arithmetic functions written, the first things the functions do is store any values in the registers to be used onto the stack, and then when the function is finished it places those values back into their respective registers. After pushing the values the program loads the inputs from program memory and locates the output memory address.

ADD16

ADD16 takes in two 16 bit numbers and outputs a 24 bit result. First it loads the two inputs from program memory and then adds the first byte of each number; it then adds the second byte of each number and the carry if any from the first addition. Finally it places the carry of the second addition operation into the last byte of the result.

SUB16

SUB16 starts similar to ADD16 in that it loads the two inputs from program memory. It then performs a subtraction operation on the lower bytes of the inputs, and then does a subtract with carry on the higher bytes. If there is another carry, then it adds the carry to the higher byte.

MUL24

MUL24 first loads the two inputs from program memory. It then goes into a sequence of nested loops which multiplies the two inputs byte by byte. It does this by first multiplying the two numbers and then storing any carry in the next output byte. It does this three times, once for each byte.

Compound

The Compound function does just as the description says; it performs the arithmetic described in the function provided. For each operation it loads the data from memory and places it in the correct program memory location and then calls the function needed for that arithmetic operation.

ADDITIONAL QUESTIONS

1. The V flag is the two's complement overflow indicator. An example of two 8-bit binary values that will cause the V flag to be set are -5 and 15 because when you add these two two's complement numbers, the carry overflows.

2. The .BYTE command allocates memory in data memory and allows the programmer to refer to that location with a user defined name. This is beneficial because it makes it easy to keep track of memory locations by adding

all of the previous bytes declared to the specified starting location to determine the starting location of any of the defined names.

DIFFICULTIES

One major difficulty was with trying to create a standard format of what to do at the beginning and end of each function based on the skeleton code given as well as the ADD16 that was provided as a challenge in a previous Lab assignment. Using a combination of both of these, we were able to come up with a general form for storing previous numbers, loading data, and restoring numbers that could be used in each function.

Another difficulty was when trying to simulate the program. The data allocation for the skeleton code seems to start allocating program memory for variables too soon, and so in writing the program it seems to overwrite and interfere with those allocated bytes. This was a major source of frustration while attempting to check the program though the simulator.

CONCLUSION

In this lab we created multiple functions each performing some sort of large number arithmetic. We used our knowledge and examples of smaller implementations to write these functions to be able to perform operations no found natively in the AVR assembly code. We built 16 bit addition, and subtraction programs, as well as a 24 bit multiplication program. Finally, we wrote a subroutine that used all of these functions to perform a complex mathematical formula.

SOURCE CODE

```
;***********************************************************
;*
;*       main.asm
;*
;*       Large number arithmetic functions
;*
;*       This is the skeleton file for Lab 5 of ECE 375
;*
;***********************************************************
;*
;*        Author: Drake Vidkjer
;*        Author: Bianca Beauchamp
;*          Date: 07/02/17
;*
;***********************************************************

.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


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

```
;*      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)
                out             SPL, mpr
                ldi             mpr, high(RAMEND)
                out             SPH, mpr        ; Initialize Stack Pointer
                                                        ; 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

                                ; (IN SIMULATOR) Enter values 0xA2FF and 0xF477 into data
                                ; memory locations where ADD16 will get its inputs from
                                ; (see "Data Memory Allocation" section below)

                                ; Call ADD16 function to test its correctness
                                ; (calculate A2FF + F477)
                                rcall ADD16

                                ; Observe result in Memory window

                ; Setup the SUB16 function direct test

                                ; (IN SIMULATOR) Enter values 0xF08A and 0x4BCD into data
                                ; memory locations where SUB16 will get its inputs from

                                ; Call SUB16 function to test its correctness
                                ; (calculate F08A - 4BCD)
                                rcall SUB16

                                ; Observe result in Memory window

                ; Setup the MUL24 function direct test

                                ; (IN SIMULATOR) Enter values 0xFFFFFF and 0xFFFFFF into data
                                ; memory locations where MUL24 will get its inputs from

                                ; Call MUL24 function to test its correctness
                                ; (calculate FFFFFF * FFFFFF)
                                rcall MUL24


                                ; Observe result in Memory window

                ; Call the COMPOUND function
                                rcall COMPOUND

                                ; 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:

                clr             zero
                push    A
                push    B
                ; 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
                ;add first byte
                ld              A, X+
                ld              B, Y+
                add             A, B
                st              Z+, A

                ;add second byte
                ld              A, X
                ld              B, Y
                adc             B, A
                st              Z+, B

                ;store carry in last byte
                brcc    EXIT
                st              Z, XH
                ;clear carry bit

                pop             B
                pop             A
                clc

EXIT:
                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: SUB16
; Desc: Subtracts two 16-bit numbers and generates a 16-bit
;                result.
;-----------------------------------------------------------
SUB16:
                clr zero
                push A
                push B

                ; 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 here
                ;sub low byte
                ld              A, X+
                ld              B, Y+
                sub             A, B
                st              Z+, A

                ;sub high byte
                ld              A, X
                ld              B, Y
                sbc             A, B
                st              Z, B

                clr             A
                adc             A, A
                st              Z, A

                pop             B
                pop             A

                ;clear carry bit
                clc

                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: MUL24
; Desc: Multiplies two 24-bit numbers and generates a 48-bit
;               result.
;-----------------------------------------------------------
MUL24:

                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

                ldi oloop, 3
MUL24_OLOOP:
                ldi iloop, 3
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

                clr             B
                ld              B, Z                    ;add carry to Z
                adc             B, zero
                st              Z, B

                st              -Z, A
                st              -Z, rhi
                st              -Z, rlo
                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
                sbiw    XH:XL, 3        ; X <= X - 3
                dec             oloop                   ; Decrement counter
                brne    MUL24_OLOOP             ; Loop if oLoop != 0

                ; End outer for loop

                ; Restore variable by popping them from the stack in reverse order
                pop             iloop
                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

                ; Execute the function here


                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:

                ; Setup SUB16 with operands D and E
                ldi             ZL, low(OperandD << 1)
                ldi             ZH, high(OperandD << 1)
                ldi             YL, low(SUB16_OP1)
                ldi             YH, high(SUB16_OP1)

                lpm             mpr, Z+
                st              Y+, mpr
                lpm             mpr, z
                st              Y, mpr

                ldi             ZL, low(OperandE << 1)
```

```
                ldi             ZH, high(OperandE << 1)
                ldi             YL, low(SUB16_OP2)
                ldi             YH, high(SUB16_OP2)

                lpm             mpr, Z+
                st              Y+, mpr
                lpm             mpr, z
                st              Y, mpr

                ; Perform subtraction to calculate D - E
                rcall SUB16

                ; Setup the ADD16 function with SUB16 result and operand F
                ; Setup SUB16 with operands D and E
                ldi             ZL, low(OperandF << 1)
                ldi             ZH, high(OperandF << 1)
                ldi             YL, low(ADD16_OP1)
                ldi             YH, high(ADD16_OP1)

                lpm             mpr, Z+
                st              Y+, mpr
                lpm             mpr, z
                st              Y, mpr

                ldi             ZL, low(OperandE << 1)
                ldi             ZH, high(OperandE << 1)
                ldi             YL, low(ADD16_OP2)
                ldi             YH, high(ADD16_OP2)

                lpm             mpr, Z+
                st              Y+, mpr
                lpm             mpr, z
                st              Y, mpr

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

                ; Setup the MUL24 function with ADD16 result as both operands
                ldi             ZL, low(LAddrP)
                ldi             ZH, high(LAddrP)
                ldi             YL, low(addrB)
                ldi             YH, high(addrB)
                ldi             XL, low(addrA)
                ldi             XH, high(addrA)

                ; Loading the answer into both inputs and zeroing answer memory
                ld              mpr, Z
                st              Z+, zero
                st              Y+, mpr
                st              X+, mpr
                ld              mpr, Z
                st              Z+, zero
                st              Y+, mpr
                st              X+, mpr
                ld              mpr, Z
                st              Z+, zero
                st              Y, mpr
                st              X, mpr

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

                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 third 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

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

; Enter any stored data you might need here

OperandD:
        .DW     0xFD51                          ; test value for operand D
OperandE:
        .DW     0x1EFF                          ; test value for operand E
OperandF:
        .DW     0xFFFF                          ; test value for operand F

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

.dseg
.org    $0170                           ; data memory allocation for MUL16 example
addrA:  .byte 3
addrB:  .byte 3
LAddrP: .byte 6

; 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   $0138                           ; data memory allocation for operands
SUB16_OP1:
                .byte 2                         ; allocate two bytes for first operand of ADD16
SUB16_OP2:
                .byte 2                         ; allocate two bytes for second operand of ADD16

;.org   $0148                           ; data memory allocation for results
SUB16_Result:
                .byte 2                         ; allocate 2 bytes for SUB16 result

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