ECE 375 LAB 5

Large number arithmetic

Lab Time: Tuesday 6-7:50

Adit Kotharii

Mac Santa Cruz

INTRODUCTION

As the name of this lab suggests, in this lab we perform large number arithmetic. The purpose of this lab was to learn how to manipulate numbers larger than 8 bits and perform arithmetic operations like addition, subtraction and multiplication on them. In addition to that we also learned how to handle functions and subroutines.

PROGRAM OVERVIEW

The program we write multiple subroutines. First, we write 2, 16-bit subroutines, addition and subtraction that stores result in the memory. Next, we write 24-bit Multiplication subroutine that performs 24-bit multiplication and stores the 6 bytes result in the memory. Finally, we use all the three functions to create the compound subroutine, that evaluates expression ((D-E)+F)^2.

INITIALIZATION ROUTINE

First the Stack Pointer is initialized and the “zero” register (R2) is cleared.

MAIN ROUTINE

We load operands for all the four subroutines from the program to the data memory. In addition to that we make calls to all the subroutines.

ADD16 ROUTINE

In this subroutine we add the two 16-bit operands to produce 24-bit result. We first add the two lower bytes of the operands and store the result in first byte. Next two higher bytes are added with carry and result is stored in second byte. Lastly, we store the carry from the higher byte addition In the third byte (24-bit result).

SUB16 ROUTINE

In this subroutine we subtract the two 16-bit operands to produce 16-bit result. We first subtract the two lower bytes of the operands and store the result in first byte. Next two higher bytes are subtracted along with carry and result is stored in second byte.

MUL24 ROUTINE

In this subroutine we multiply the two 24-bit operands to produce 48-bit result. We multiply every byte in the first operand with every byte of the second operand. We implement sum of products technique using doubly nested loop to obtain the final result.

COMPOUND ROUTINE

In this subroutine we evaluate the given equation: ((D-E)+F)^2 by using functions ADD16, SUB16 and MUL24. The operands D, E and F are defined in the program memory to us as the input operands.

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 also known as overflow flag. It indicates if the two operands added or subtracted are larger than 8 bits. For instance when the signed binary numbers 0b10000000 and 0b01000000 are subtracted, results in a larger number than 8 bits appropriately setting the v flag.

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?

.BYTE preceded by the label, reserves space in data memory in a consecutive manner, which makes it easier for us to access these locations using pointers and their post increment and pre decrement capability. This enable us to access and manipulate operands larger than 8-bits with ease. While using .EQU would only let us declare one address constant in data memory. Depending on our manner of declaring these address constants they may or may not be in a consecutive fashion. This in turn would make it hard for us to use pointers to access operands larger than 8 bits, since we won't be able to use pointers their post increment and pre decrement capability

CONCLUSION

In this lab I learned how to perform arithmetic operations such as addition, subtraction and multiplication on numbers larger than 8 bits. In addition to that I also learnt how to call subroutines to create my own function. . This lab really helped me get better at coding in assembly

SOURCE CODE

```
;***********************************************************
;*
;*      Enter Name of file here
;*
;*      Enter the description of the program here
;*
;*      This is the skeleton file for Lab 5 of ECE 375
;*
;***********************************************************
;*
;*       Author: Enter your name
;*         Date: Enter Date
;*
;***********************************************************

.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 R16, LOW(RAMEND)    ; Initialize Stack Pointer
               OUT SPL, R16
               LDI R16, HIGH(RAMEND)
               OUT SPH, R16                 ; 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
;-----------------------------------------------------------
; Loading in Add operands from Program to Data memory
;-----------------------------------------------------------
                               LDI ZL, low(AddOperand1 << 1)
               ;Shift Bit due to 16 Bit program memory
                               LDI ZH, high(AddOperand1 << 1)
                       ;The locations are seqential in both program
                               LDI YL, low(add16_op1)
                       ;and data memory so a post increment is used
                               LDI YH, high(add16_op1)
                               ;to move to different locations
                               LPM R16, Z+
                                       ;DATA MEMORY LOCATION: $0120-$0123
                               LPM R17, Z+

                               ST Y+, R16
                               ST Y+, R17
//Opperand 2 Below
                               LDI ZL, low(AddOperand2 << 1)
                               LDI ZH, high(AddOperand2 << 1)

                               LPM R16, Z+
                               LPM R17, Z+

                               ST Y+, R16
                               ST Y+, R17

                nop
                               ; Check load ADD16 operands (Set Break point here #1)
rcall ADD16
                                       ;result should be at $0130-$0132

                                       ; (calculate A2FF + F477) = 19776
```

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

                                                ; Observe result in Memory window
;-----------------------------------------------------------
; Loading in Sub operands from Program to Data memory
;-----------------------------------------------------------
                                LDI ZL, low(SubOperand1 << 1)
               ;DATA MEMORY LOCATION:$0124-$0127
                                LDI ZH, high(SubOperand1 << 1)
                        ;Where opperands are found
                                LDI YL, low(sub16_op1)
                                LDI YH, high(sub16_op1)
                                LPM R16, Z+
                                LPM R17, Z+

                                ST Y+, R16
                                ST Y+, R17
//Opperand 2 Below
                                LDI ZL, low(SubOperand2 << 1)
                                LDI ZH, high(SubOperand2 << 1)

                                LPM R16, Z+
                                LPM R17, Z+

                                ST Y+, R16
                                ST Y+, R17



                nop
                                ; Check load SUB16 operands (Set Break point here #3)

rcall SUB16;
                                ;result should be at $0133-$0134


                                                ; (calculate F08A - 4BCD)= A4BD

                nop
                                ; Check SUB16 result (Set Break point here #4)

;-----------------------------------------------------------
; Loading in Mult24 operands from Program to Data memory
;-----------------------------------------------------------
                                LDI ZL, low(MUL24OP1 << 1)
                        ;Same as add and sub but opperands here are 3 Bytes
                                LDI ZH, high(MUL24OP1 << 1)
                        ;So they have one more location necessary to load from
                                LDI YL, low(addrA)
                                ;They were stored in program memory as Bytes which means
                                LDI YH, high(addrA)
                                ;They were entered in backwards due to the order in which they will
be grabbed
                                LPM R16, Z+
                                LPM R17, Z+
                                LPM R18, Z
                                ST Y+, R16
                                ST Y+, R17
                                ST Y+, R18
//Opperand 2 Below
                                ;DATA MEMORY LOCATION:$0100-$0105 split at $0102
                                LDI ZL, low(MUL24OP2 << 1)
                                LDI ZH, high(MUL24OP2 << 1)

                                LPM R16, Z+
                                LPM R17, Z+
                                LPM R18, Z

                                ST Y+, R16
                                ST Y+, R17
```

```
                              ST Y+, R18
                              ldi             ZL, low(LAddrP)
                                     ;Set Z register before MUL24 CAll
                              ldi             ZH, high(LAddrP)
                              ;The function as written expects $00

                                              ;at memory location of result

                                              ;which means each multiply call needs a new location
for the product


                  nop
                              ; Check load MUL24 operands (Set Break point here #5)
rcall MUL24;
                              ; result should be at $0106-0111

                                              ; (calculate FFFFFF * FFFFFF) =  FFFFFE000001

                  nop
                              ; Check MUL24 result (Set Break point here #6)
;-----------------------------------------------------------
; Loading in Compund operands from Program to Data memory
;-----------------------------------------------------------
                              LDI ZL, low(OperandD << 1)
                              LDI ZH, high(OperandD << 1)
                              LDI YL, low(compoundopp1)
                         ;data memory location needs to be called only one time
                              LDI YH, high(compoundopp1)
                         ;as the opperands are stored next to one another in memory
                              LPM R16, Z+
                                     ;DATA MEMORY: $0140-$0145. Each opperand takes 2 bytes
                              LPM R17, Z+
                                     ;0xFD51
                              ST Y+, R16
                              ST Y+, R17
//Opperand 2 Below   (E)
                              LDI ZL, low(OperandE << 1)
                         ;0x1EFF
                              LDI ZH, high(OperandE << 1)

                              LPM R16, Z+
                              LPM R17, Z+

                              ST Y+, R16
                              ST Y+, R17
//Opperand 3 Below (F)
                              LDI ZL, low(OperandF << 1)
                         ;0xFFFF
                              LDI ZH, high(OperandF << 1)

                              LPM R16, Z+
                              LPM R17, Z+

                              ST Y+, R16
                              ST Y+, R17
                  nop
                              ; Check load COMPOUND operands (Set Break point here #7)
rcall compound
                              ; result should be at $0106-0111

                  nop
                              ; Check COMPUND result (Set Break point here #8)


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
                clr R20

                                        ;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
                ld          R16, X+
                                        ;low of first opperand
                ld              R17, X+
                                        ;high of first opperanc
                ld              R18, X+
                                        ;low of second opperand
                ld              R19, X+
                                        ;high of second opperand



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

                ADD R16, R18;
                                ;add lower bytes of each opperand
                ST      Z+, R16;


                ADC R17, R19;
                                ;add upper bytes of each opperand along with carry
                ST      Z+, R17;
                ADC R20, zero;
                                ;get carry
                ST      Z+, R20;




                                        ; Execute the function

                ret
                                        ; End a function with RET

;-----------------------------------------------------------
; Func: SUB16
; Desc: Subtracts two 16-bit numbers and generates a 16-bit
;               result.
;-----------------------------------------------------------
SUB16:

                clr zero

                                        ;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
                ld          R16, X+
                                        ;low of first opperand
                ld              R17, X+
                                        ;high of first opperanc
```

```
                ld              R18, X+
                                        ;low of second opperand
                ld              R19, X+
                                        ;high of second opperand


                                                ;Load beginning address of result into Z
                ldi             ZL, low(Sub16_Result)
                ldi             ZH, high(Sub16_Result)

                SUB R16, R18;
                                ;Subtract Lower bytes of each opperand
                ST      Z+, R16;
                SBC R17, R19;
                                ;Subtract High Bytes of each opperand along with carry
                ST      Z+, R17;


                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

                ; Begin outer for loop
                ldi             oloop, 3
                                        ; Load counter
MUL24_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, 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
                ld              B,Z
                adc             b, zero
                st              Z, B

                ; 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 - 1
                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:

                ; Setup SUB16 with operands D and E
                LDI YL, low(compoundopp1)
                LDI YH, high(compoundopp1)
                LD R16, Y+
                LDI XL, low(sub16_op1)
                        ;DATA MEMORY Location: $0124-$0127
                LDI XH, high(sub16_op1)

                ldi r24, 4
                compoundLoopSub:
                        ST X+, R16
                         ; low of Op1
                        LD R16, Y+
                        dec r24
                        brne compoundLoopSub

rcall sub16
                                ; result should be at $0133

                                        ; Setup the ADD16 function with SUB16 result and operand F
                LDI YL, low(Sub16_Result)
                LDI YH, high(Sub16_Result)
                LD R16, Y+
                LDI XL, low(add16_op1)
                ;DATA MEMORY Locatoin: $0120-$0123
                LDI XH, high(add16_op1)

                ldi r24, 2
                compoundLoopAdd1:
                        ST X+, R16
                        LD R16, Y+
                        dec r24
                        brne compoundLoopAdd1
                        LDI YL, low(compoundopp3)
                ;location of f in data memory
                        LDI YH, high(compoundopp3)
                        LD R16, Y+
                                ; first byte of F
                        ldi r24, 2
                compoundLoopAdd2:
                        ST X+, R16
                        LD R16, Y+
                        dec r24
                        brne compoundLoopAdd2

rcall add16
                                ; result should be at $0130-$0132

                                        ;  Setup  the  MUL24  function  with  ADD16  result  as  both
operands
                LDI YL, low(add16_Result)
                LDI YH, high(add16_Result)
                LD R16, Y+
                LDI XL, low(addrA)
                        ;DATA MEMORY: $0100-$0102
                LDI XH, high(addrA)
                LDI ZL, low(addrB)
                        ;DATA MEMORY: $0103-$0105
```

```
                LDI ZH, high(addrB)

                ldi r24, 3
        compoundLoopMult1:
                        ST X+, R16
                        ST Z+, R16
                        LD R16, Y+
                        dec r24
                        brne compoundLoopMult1
                        ldi             ZL, low(Compound_Result)
                                                ; Load low byte

                                        ;Set Z register before MUL24 CAll
                        ldi             ZH, high(Compound_Result)
                        ;The function as written expects $00

                                        ;at memory location of result

                                        ;which means each multiply call needs a new location
for the product
rcall mul24
                                        ; result should be at $0106-0111

                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
AddOperand1:
        .DW 0xA2FF
AddOperand2:
        .DW 0xF477
; SUB16 operands
SubOperand1:
        .DW 0xF08A
SubOperand2:
        .DW 0x4BCD
; MUL24 operands
MUL24OP1:
        .DB 0xFF, 0xFF, 0xFF, 0x00//.DB 0x51, 0xDE, 0x01, 0x00//
```

```
MUL24OP2:
        .DB 0xFF, 0xFF, 0xFF, 0x00//.DB 0x51, 0xDE, 0x01, 0x00//
; Compoud operands
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    $0100
addrA:  .byte 3
addrB:  .byte 3
LAddrP: .byte 6

; Consider using something similar for SUB16 and MUL24.

.org    $0120                           ; 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 two bytes for first operand of Sub16
Sub16_OP2:
                .byte 2                         ; allocate two bytes for second operand of Sub16

.org    $0130                           ; data memory allocation for results
ADD16_Result:
                .byte 3                         ; allocate three bytes for ADD16 result
Sub16_Result:
                .byte 3                         ; allocate three bytes for Sub16 result

.org    $0140
CompoundOpp1:
                .byte 2
CompoundOpp2:
                .byte 2
CompoundOpp3:
                .byte 2

.org    $0150
Compound_Result:
                .byte 6

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