ECE 375 Lab 4

Large Number Arithmetic

Lab Time: Wednesday 5-7

Jacques Uber
Riley Hickman

TA Signature

1 Introduction

Adding and multiplying numbers greater than 1 eight bit byte.

2 What We did and Why

We went through the given code for add16 and mul16 to understand what they are doing. To do that we got on the AVR studio and ran the code in debug mode to see what each line was doing. Once the loop functions were identified we were able to make it perform mul24 by increasing the inner and outer loop iterations by one and adjusting Z by an additional location after the loop ceases.

To implement add16 we followed the instructions laid out in the pre-lab. We also drew out what the code would look like before we coded it.

3 Difficulties

We forgot to initialize the stack pointer. When we simulated the execution of our code, we noticed that it was returning from functions to odd places. We could not figure out what was wrong. Once we realized that we had forgotten to initialize the stack pointer things become more clear.

4 Conclusion

We have more respect for calculators.

5 Source Code

```
;***********************************************************
;*
;*  Enter Name of file here
;*
;*  Enter the description of the program here
;*
;*  This is the skeleton file Lab 4 of ECE 375
;*
;***********************************************************
;*
;*   Author: Jacques Uber, Riley Hickman
;*     Date: 2/1/2012
;*
;***********************************************************

.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 calcula
.def    A = r3                  ; An operand
.def    B = r4                  ; Another operand

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

.equ    addrAS = $0100          ; Beginning Address of Operand B data
.equ    addrBS = $0106

.equ    addrAM = $010C          ; Beginning Address of Operand B data
.equ    addrBM = $010C

.equ    LAddrP = $011C          ; Beginning Address of Product Result
.equ    HAddrP = $011E          ; End Address of Product Result

.equ    LAddrS = $010C
.equ    HAddrS = $010E
;.equ   LAddrS = $0112
;.equ   HAddrS = $0114
;***********************************************************
;*  Start of Code Segment
;***********************************************************
.cseg                           ; Beginning of code segment

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

.org    $0046                   ; End of Interrupt Vectors

;-----------------------------------------------------------
; Program Initialization
;-----------------------------------------------------------
INIT:                           ; The initialization routine
        ; Initialize Stack Pointer
        ; TODO                  ; Init the 2 stack pointer registers
        ldi     mpr, High(RAMEND)
        out     sph, mpr
```

```
        ldi     mpr, Low(RAMEND)
        out     spl, mpr

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

;-----------------------------------------------------------
; Main Program
;-----------------------------------------------------------
MAIN:                           ; The Main program
        ; Setup the add funtion
        ; Add the two 16-bit numbers
        rcall   ADD16           ; Call the add function

        ; Setup the multiply function

        ; Multiply two 24-bit numbers
        rcall   MUL24           ; Call the multiply function

        ;rcall  MUL16           ; Call the multiply function
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:
        ; Save variable by pushing them to the stack

        ; Execute the function here

        ; Restore variable by popping them from the stack in reverse order\
        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(addrBS) ; Load low byte
        ldi     YH, high(addrBS)    ; Load high byte

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

        ; Begin outer for loop
        ldi     oloop, 1        ; Load counter
;ADD16_OLOOP:
        ; Set X to beginning address of A
        ldi     XL, low(addrAS) ; Load low byte
        ldi     XH, high(addrAS)    ; Load high byte

        ; Begin inner for loop
        ldi     iloop, 2        ; Load counter
ADD16_ILOOP:
        ld      A, X+           ; Get byte of A operand
        ld      B, Y+           ; Get byte of B operand
        adc     A,B             ; Multiply A and B
        st      Z+, A

        ld      A, X+           ; Get byte of A operand
        ld      B, Y+           ; Get byte of B operand
        adc     A,B             ; Multiply A and B
        st      Z+, A
        clr     B
        adc     B, zero         ; Add carry to A

        st      Z+, B

        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
        ;ret                        ; End a function with RET

;-----------------------------------------------------------
; Func: MUL24
; Desc: Multiplies two 24-bit numbers and generates a 48-bit
;       result.
;-----------------------------------------------------------
MUL24:
        ; Save variable by pushing them to the stack

        ; Execute the function here

        ; Restore variable by popping them from the stack in reverse order\
        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(addrBM) ; Load low byte
        ldi     YH, high(addrBM)    ; 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(addrAM) ; Load low byte
        ldi     XH, high(addrAM)    ; 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 third 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: 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(addrBM) ; Load low byte
        ldi     YH, high(addrBM)    ; 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(addrAM) ; Load low byte
        ldi     XH, high(addrAM)    ; 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

;-----------------------------------------------------------
; 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

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