Benjamin Linam

Bml42@students.uwf.edu

EEL 4744L: Microprocessor Applications Laboratory

Lab 3: Writing and Testing a Simple Program

2/7/2018

Objective

Introduce students to writing and testing a program in HC11 assembly language and using the BUFFALO I/O routines to display/verify the results.

Introduction/Background/Theory

This lab required construction of an assembly program which would compute the 4-byte product of a 3-byte number and a 1-byte number. At first glance, it appeared that a singular usage of the MUL command would provide the desired result; however, MUL is restricted by only being able to multiply two 2-byte numbers saved in both registers A and B. Since one of the numbers, the multiplicand, will be 3-bytes long, it is impossible to utilize only one MUL command to compute the product. As such, a complex program must be constructed to utilize MUL several times to multiply each individual byte of the multiplicand and arrive at the desired value.

Provided for the lab, was the following algorithm which lists the operations required to compute the product:

	MSB	MID	LSB	
	XX	XX	XX	**Multiplicand**
			XX	**Multiplier**
		MSB	LSB	
		XX	XX	**P1**
	MSB	LSB		
	XX	XX		**P2**
MSB	LSB			
XX	XX			**P3**
MSB	MID-H	MID-L	LSB	
XX	XX	XX	XX	**Product**

Figure 1: 3-byte by 1-byte multiplication algorithm

Procedure

1. By using the algorithm shown in Figure 1, the following assembly language program was designed to formulate the 4-byte product ([A], [B], and [D] represent accumulators A, B, and D respectively).

```
        org     $00      ;data allocation begin
M       rmb     3        ;allocate 3 bytes for multiplicand
N       rmb     1        ;allocate 1 byte for mulipliier
P       rmb     4        ;allocate 4 bytes for final product
P1      rmb     2        ;allocate 2 bytes for product1
P2      rmb     2        ;allocate 2 bytes for product2
P3      rmb     2        ;allocate 2 bytes for product3

        org     $0100    ;code data begin
        ldaa    M + 2    ;read multiplicand LSB to [A]
        ldab    N        ;read multiplier to [B]
        mul              ;mulitiply [A] * [B]
        stab    P + 3    ;store [B] in product LSB
        staa    P1       ;store [A] to P1 MSB
        ldaa    M + 1    ;read mulitplicand MID to [A]
        ldab    N        ;read multiplier to [B]
        mul              ;multiply [A] * [B]
        std     P2       ;store [D] to P2
        ldaa    M        ;read multiplicand MSB to [A]
        ldab    N        ;read multiplier to [B]
        mul              ;multiply [A] * [B]
        std     P3       ;store [D] to P3
        ldaa    P1       ;read P1 MSB to [A]
        adda    P2 + 1   ;add P2 LSB
        staa    P + 2    ;store sum to product MID-L
        ldaa    P2       ;read P2 MSB to [A]
        adca    P3 + 1   ;add with carry P3 LSB
        staa    P + 1    ;store sum to product MID-H
        ldaa    P3       ;read P3 MSB to [A]
        adca    #$00     ;add with carry, zero
        staa    P        ;store sum to product MSB
        swi              ;exit program
```

Figure 2: Assembly language code designed to compute the product of a 3-byte number and a 1-byte number

2. The program retrieves a 3-byte multiplicand from memory locations $00-$02 and a 1-byte multiplier from memory location $03. Each set of 2-byte terms shown in Figure 1 are then stored in accumulators A and B, multiplied by use of the MUL command, and finally saved at memory locations $04-$07 from most significant bit to least significant bit.

3. The following is a list of multiplicands and multipliers, of which the 4-bit product was to be determined:

Multiplicand	**Multiplier**	**Product**
$A4 B1 92	$74	$4A A0 72 28
$F2 84 C7	$E1	$D5 26 B2 E7
$19 65 E9	$F4	$18 35 22 14
$19 65 E9	$00	$00 00 00 00
$19 65 E9	$01	$00 19 65 E9
$19 65 E9	$08	$00 CB 2F 48
$FF FF FF	$FF	$FE FF FF 01

Table 1: List of test multiplicands, multipliers, and their respective products.

4. Once the product is calculated, that value can be displayed by using BUFFALO I/O routines to search for the specific memory location where the product was stored. The Buffalo commands used to enter the multiplicands and multipliers is “MM 00”, followed by the desired values represented in hexadecimal (0-F). That command will save the multiplicand to memory locations $00-02 and multiplier to memory location $03. After running the program using the

command “G 0100” (0100 being the location of the starting byte of the program’s code), the product will be saved to memory locations $04-$07. Entering the command “MD 00 1” displays memory locations $00-$0F to enable all three values to be viewed at once.

```
>md 00 1

0000 A4 B1 92 74 4A A0 76 28 42 FF 50 34 4A 50 FF FF    tJ v(B P4JP
>
```

Figure 3: Memory locations $00-$0F after calculating first product.

```
>g 0100

P-0129 Y-FFFF X-FFFF A-D5 B-B2 C-D8 S-0041
>md 00 1

0000 F2 84 C7 E1 D5 26 B2 E7 AE FF 74 04 D4 B2 FF FF      &    t
>
```

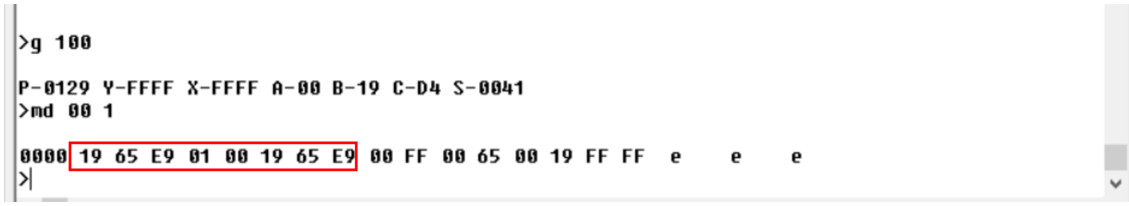
Figure 4: Memory locations $00-$0F after calculating second product.

```
>g 100

P-0129 Y-FFFF X-FFFF A-18 B-D4 C-D0 S-0041
>md 00 1

0000 19 65 E9 F4 18 35 22 14 DE FF 60 44 17 D4 FF FF  e   5"   `D
>
```

Figure 5: Memory locations $00-$0F after calculating third product.

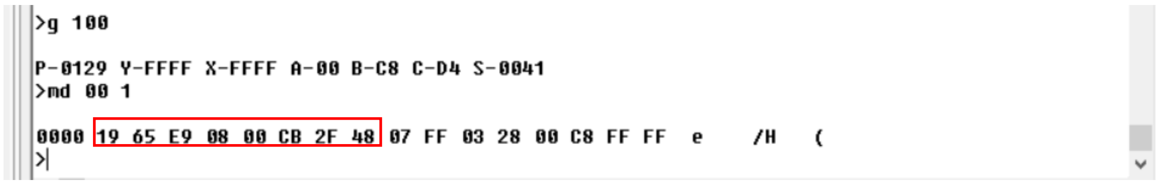
```
>md 00 1

0000 19 65 E9 00 00 00 00 00 00 FF 00 00 00 00 FF FF  e
>
```

Figure 6: Memory locations $00-$0F after calculating fourth product.



```
>g 100

P-0129 Y-FFFF X-FFFF A-00 B-19 C-D4 S-0041
>md 00 1

0000 19 65 E9 01 00 19 65 E9 00 FF 00 65 00 19 FF FF  e   e   e
>
```

Figure 7: Memory locations \$00-\$0F after calculating fifth product.



```
>g 100

P-0129 Y-FFFF X-FFFF A-00 B-C8 C-D4 S-0041
>md 00 1

0000 19 65 E9 08 00 CB 2F 48 07 FF 03 28 00 C8 FF FF  e   /H   (
>
```

Figure 8: Memory locations \$00-\$0F after calculating sixth product.

```
>g 100

P-0129 Y-FFFF X-FFFF A-FE B-01 C-D8 S-0041
>md 00 1

0000 FF FF FF FF FE FF FF 01 FE FF FE 01 FE 01 FF FF
>
```

Figure 9: Memory locations \$00-\$0F after calculating seventh product.

Conclusions

The program and lab were completed quickly and efficiently. Having the algorithm as well as the algorithm steps made it easy to decide which assembly commands were needed to complete the lab. The only problems encountered while completing the lab were that the mode jumpers were in the wrong location for testing the program and two screws had to be removed from the serial connector on the HC11.