EE 308 – LAB 2

Further Use of ZAP and the EVBU

This laboratory will give you more experience with the tools we will use this semester. Be sure to read through the entire lab and do the pre-lab for each section **before** coming to lab.

1. Consider the program in Figure 1:

```
prog:   equ     $0800

CODE:   section .text
        org     prog
        ldab    #23
        ldaa    #142
        aba
        std     $980
        swi
```

Figure 1. Demo program for Part 1 of Lab 2.

Do (a), (b), (c) and (d) **before** coming to lab.

(a) Hand-assemble this program. I.e., determine the op-codes the HC12 will use to execute this program.

(b) How many instruction cycles will it take the HC12 to execute this program? How long (in time) will this take on the HC12? (Do not consider the `swi` instruction.)

(c) What will be the state of the `N`, `Z`, `V` and `C` bits after each instruction has been executed. (Ignore the `swi` instruction.)

(d) What will be in address `0x0980` after the program executes?

(e) Assemble the program using `ca6812`, `clnk` and `chex`. Look at the `S19` file. You should be able to relate the op-codes from part (a) to the data in the `S19` file. Appendix A of the **Evaluation Board User's Manual** describes the format of the `S19` file.

(f) Now execute the program in the `ZAP` simulator. After starting `ZAP`, set the cycle counter to zero. You can do this by going to the `COMMAND` window and giving the instruction

```
eval $time=0
```

. Trace through the program using `ZAP`. To find out how many cycles it took to execute, go to the `COMMAND` window an give the instruction

```
eval $time
```

. Compare this to the answer of part (b).

(g) Look at the condition code register after you have traced over the `aba` instruction. Verify that the `Z`, `N`, `V` and `C` bits are what you expect.

(h) In the `MEMORY` window look at the contents of address `0x0980`. Does the value agree with your answer of part (c)? (Note: `ZAP` does not automatically update the memory window. You will need to load the memory window again using the `Address` menu of the `MEMORY` window.)

(i) You could change this program to subtract rather than add by changing the `aba` instruction to a `sba` instruction. Rather than going back to the text editor, modifying the program, assembling it and loading the new program into the simulator, you can easily change the one instruction of the simulator.

i. Find the address of the `aba` instruction. Do this by looking at the program code in the `ASSEMBLE` window of the simulator.

ii. In the **CPU12 Reference Manual**, find the op code for the `sba` instruction.

iii. In the `MEMORY` window, go to the address of the `aba` instruction, and change the op code to that of the `sba` instruction. You can change a value in the `MEMORY` window by double-clicking on it and typing in the new value.

iv. Run the program again, and verify that the program now subtracts rather than adds.

(j) Repeat (f) through (h) on your EVBU. For part (e) you will not be able to determine the number of cycles used on the EVBU. For part (h), use the `ASM` command of BUFFALO to look at and change the program instruction on the EVBU. This is easier than changing the op code in memory.

2. Consider the program in Figure 2, which is a program to divide a table of ten values by 2.

```
; 68HC11 demo program
; Bill Rison
; 1/26/99

; This is a program to take a table of data, and create a new table
; which is the original table divided by 2

         title   "LAB 2 Demo Program"

prog:    equ     $0800    ;put program at address 0x0800
data:    equ     $0900    ;put data at address 0x0900
count:   equ     10       ;number of entries in table

CODE:    section .text
         org     prog     ;set program counter to 0x0800
         ldab    #count   ;ACC B holds number of entries left to process
         ldx     #table1  ;Reg X points to entry to process
repeat:  ldaa    0,x      ;Get table1 entry into ACC A
         asra              ;Divide by 2
         staa    count,x  ;Save in table2
         inx               ;REG X points to next entry in table1
         decb              ;Decrement number left to process
         bne     repeat   ;If not done, process next table1 entry
         swi               ;Done -- Exit


DATA:    section .data
         org     data
;initialize table1 (COUNT bytes long)
```

```
table1:  dc.b    $07,$ae,$4a,$f3,$6c,$30,$7f,$12,$67,$cf
table2:  ds.b    count    ;reserve count bytes for table2.
```

Figure 2. Demo program for part 2 of lab 2.

(a) Use a text editor to enter this program, or download it from `http://www.ee.nmt.edu/~rison/ee308/labs/lab02/demoprog.s`. Assemble the program into an `s19` file.

(b) Load the program into the simulator. Run the program. How many cycles does it take to execute the entire program? How long would this take on the EVBU?

(c) Use the `fill` submenu of the `memory` window simulator to change the values in addresses `0x0900` through `0x9020` to `0xff`. Reload the `s19` file.

(d) Set a break point at `repeat`.

(e) Execute the program again. The program should stop the first time it reaches the `repeat` label, with `0x0a` in `acc b`, and `0x0000` in `x`.

(f) Continue running the program. It should stop each time it gets to the `repeat` label – `b` should be decremented by one, `x` should be incremented by one, and there should be a new entry in `table2`. (Note: `ZAP` does not automatically update the memory window. You will need to load the memory window again using the `Address` menu of the `MEMORY` window.)

(g) Repeat the above using the EVBU.

3. Consider the code of Figure 3. Do parts (a) and (b) below `before` coming to lab.

```
          ldab    #200
loop1:    ldx     #50000
loop2:    dex
          bne     loop2
          decb
          bne     loop1
          swi
```

Figure 3. Demo program for part 3 of lab 2.

(a) **Answer as part of pre-lab:** How many cycles will it take to execute this program? How long will this take on the EVBU? (Again, ignore the `swi` instruction.)

(b) Use a text editor to enter the code into a program – you will have to add `org` statements and other assembler directives to make the program work.

(c) Assemble the program and run it on the HC12. How long does it take to run? This time should match your answer to part (a).