Homework 5 Solution

Comment the code given on the following page and answer the questions given below. Remember that your comments should describe the function of the code in the program, not simply state what the instruction does. Be sure to include in your comments for the subroutine a list of what registers it uses and modifies.

1. What does the instruction “dbne b, addloop” do? (We have not seen this instruction in lecture. Check the instruction reference in the appendix of your textbook.)

As indicated on page 446 of Appendix A of the text, this instruction, having the form “DBNE cntr, rel”, decrements the counter “cntr” (the value in accumulator B in this case) and branches if the counter value is nonzero. Here, B counts down the bytes of data to be summed. If bytes remain to be added, execution branches back to location addloop.

2. What are the contents of memory locations 0x09f0 through 0x0a00 after the jsr instruction is executed?

xx,xx,xx,xx,08,14,0A,09,08,07,06,05,04,03,02,01, where “xx” means whatever was there before we executed our program.

3. What is the sum of the first two numbers that are added by the subroutine?

The program sums the data starting with the last values so the sum of the first two numbers added is 10+9=19 or $13.

4. What is the value of of the stack pointer after these two numbers have been added?

After these first two values are added, the stack pointer points to the next value to add, which is the value 8 at location 0x09F8.

5. What are the contents of the A, B, X, Y, CCR, and SP registers when execution reaches the swi instruction?

The value in accumulator A is $37 = 3*16+7 = 55 = 11*10/2. (\sum_{i=1}^{n} i = n(n+1)/2, if you don’t feel like doing the addition.)

The value in accumulator B is zero because the dbne statement decrements the counter in B to zero.

The value in the X index register is $090A because the X register was used as an index to the data values stored startnig at location buf.

The value in the Y index register is zero because we did not use it and hence it retained its initial value.

The value in the CCR is %1101 0000. Working backward from the SWI instruction, the leas num,sp, rts, leas \{0-num-2\},sp, and dbne b,addloop instructions do not affect the flags (see Appendix A) so the last instruction that affects the flags is the adda 1,sp+ which adds to the value in register A the value pointed by SP and then increments the stack pointer. The last time this instruction is executed, it adds 1 to $36 to get $37 so H,
N, Z, V, and C are all cleared. (There is no carry out of bit 3 nor bit 7; the result is not zero; both addends and the result are positive so no overflow occurs.)

```
num: equ !10 ; number of data values in the ‘‘buffer’’
      ; which starts at location ‘‘buf’’.
prog: equ $0800
data: equ $0900
stack: equ $0a00

org prog
lds #stack ; initialize stack pointer
ldab #num ; load counter for data values
ldx #buf ; use X as a pointer into the ‘‘buffer’’
ldloop: ldaa 0,x
        psha ; push the current data value to the stack
        inx
        dbne b,ldloop ; decrement the counter and loop if there are
        ; still values to be read
        ldab #num ; reinitialize the counter for the summation step
        jsr CalcSum
        leas num,sp ; set the stack pointer back to wherever it was
        ; before we pushed the data
        swi

; subroutine CalcSum uses and modifies registers A and B

CalcSum: leas 2,sp ; skip over the return address
         pula ; pull a data value from the stack
decb ; decrement the counter
addloop: adda 1,sp+ ; add to the value in A the value pointed
         ; to by the stack pointer and then increment
         ; the stack pointer
dbne b,addloop ; decrement the counter and if not done branch
         ; back to add another value
        leas {0-num-2},sp
         ; set the stack pointer back to the return
         ; address for the subroutine call.
        rts

org data
buf:   db 1,2,3,4,5,6,7,8,9,!'10
       ; data values
```