

6812 Assembly Programming

PURPOSE

In this two-week Lab, you will practice 6812 assembly instructions about memory access, loop control, and subroutine call.

PRELAB (20%) Complete the prelab assignments as teams.

(1) (Week 1: 10%) Draw a flow chart, with a format similar to Figure 2.9 on page 68 (page 58 for the 1st edition), for the program to be used in Experiment (1), but with all variables replaced with constants, memory locations, or register names. Write down a program based on the flow chart. Make sure your program does not have any syntax error by using the **as12** assembler. Turn in the flow charts, but not the program. Bring the program to the lab and demonstrate that there is no syntax error at the beginning of Experiment (1).

(2) (Week 2: 10%) Study the finished code for Experiment (1) that was debugged by your team. Turn in a drawing of the stack frame and two flow charts for Experiment (2), one for the main program and one for the subroutine. (For the flow chart of a subroutine, use a “Return” terminal box instead of a “Stop” terminal box.) Write down the whole program, use **as12** to remove syntax errors, and bring it to the lab for demonstration at the beginning of Experiment (2). There is no need to turn in the code.

EXPERIMENT (80%)

(1) (40%) Write a 6812 “memory fill” assembly program that writes (and verifies the success of data writing) a one-byte constant to each location of a 16-byte memory area. (You must use loop control. Repeating instructions 16 times is not acceptable.) The 2-byte starting address of the 16-B memory is not always the same and is specified as a 2-byte number stored at memory location \$0CF0 (and \$0CF1). The one-byte constant is also not always the same and is specified as a number stored at memory location \$0CF2. You specify those three bytes using the monitor command MM prior to the execution of the program. The program should turn off LED1 at the beginning. It should turn on LED1 if the memory fill succeeds. It should also return to the monitor when it terminates. You should use the MD command to examine the memory area after the program execution. Demonstrate your program on the EVB.√√

Hint: LED1 relevant register addresses: (PORTA: \$0000, DDRA: \$0002)

(2) (40%) Modify the previous program as follows.

- The *16-byte memory fill* part becomes a subroutine that takes an argument for the starting memory address and an argument for the one-byte constant. Use the stack, not a register, to pass the input argument.
- The main program first turns on LED1 at the beginning. It then applies the memory fill subroutine two times, first with 3 bytes stored at \$0CF0, as in Experiment (1), and then with the other 3 bytes at \$0CF3. The program turns on LED1 when both memory areas have been successfully filled.

The program should return to the monitor when it terminates, irrespective of the memory fill results. Demonstrate the program execution.√√