Lab 3: Software Simulator for the M68HC11

Introduction

A 68HC11 software simulator is useful because programs for the microcontroller can be tested and debugged without using the hardware. "Wookie" is a software simulator for Windows that can simulate the function of a real 68HC11 microcontroller. Through a graphical user interface, the programmer can monitor program execution and change the status of the simulated microcontroller. It runs without any specialized hardware so it can be used on any Windows PC, including those outside the ELE 205 laboratory.

We will also be using the AS11 assembler, which is much more powerful than the inline assembler `ASM` in BUFFALO. The AS11 assembler can process labels (for loops, branches, and variables) and assembler directives.

This software will be used in our future prelab exercises. Before running your program during your scheduled lab section, you should type it in using the Windows Notepad editor (or any text editor), assemble it with the AS11 assembler, and test/debug it using the Wookie simulator. In this lab, we will practice using the simulator with a simple program. You should download and read the Wookie Simulator manual from the ELE 205 web page:

`http://www.ele.uri.edu/faculty/vetter/ELE205/the6811wookie.pdf`

or the Wookie Simulator home page:

`http://people.msoe.edu/~barnicks/courses/cs400/19978/hc11/the6811wookie.pdf`

To do the prelab work on a PC outside of the ELE 205 lab, you will need the executable Wookie program and the AS11 cross assembler; links to these programs and their documentation are on the ELE 205 web page.

Related Reading

Chapters 2, 3, and 4 in the *8-Bit Cross Assemblers User's Manual* (on the ELE 205 web page); skip sections 2.2.3.1 through 2.2.3.4 in Chapter 2

Pages 161-172 in Spasov's textbook (note that assembler directives are called "pseudo-operations" by Spasov)

Reference notes for the AS11 cross assembler: AS11.txt (on ELE 205 web page)

A note about the Stack

The "stack" is a small portion of memory reserved for the MCU. The MCU stores information on the stack using the Stack Pointer, the 16-bit SP register. On the EVB, the BUFFALO monitor initializes the SP register to $0041. The Wookie simulator, however, does not initialize SP, so

we must do it using the `LDS` instruction. Be sure to always initialize the SP register when running programs using the Wookie simulator. We will discuss the stack later in the semester.

Lab Work

We will use the following program to learn the Wookie simulator:

```
* A 68HC11 program to compute the sum and
* difference of two unsigned 8-bit numbers.
SUM  EQU $0103      ; assign address to label SUM
DIFF EQU $0104      ; assign address to label DIFF
ABSA EQU $0105      ; assign address to label ABSA

     ORG $0100      ; Variables start here
NUM1 FCB $1C
NUM2 FCB $0C

     ORG $C000      ; Executable code starts here

     LDS #$0041     ; initialize stack pointer

     LDAA NUM1
     LDAB NUM2
     JSR GET_SUM
     STAA SUM

     LDAA NUM1
     JSR GET_DIFF
     STAA DIFF
     BGT ST_ABS
     NEGA
ST_ABS:
     STAA ABSA
     SWI

GET_SUM:            ; Subroutine GET_SUM
     ABA
     RTS            ; return the sum in ACCA

GET_DIFF:           ; Subroutine GET_DIFF
     SUBA NUM2
     RTS            ; return the difference in ACCA
```

1. Examine the program and determine the expected result of each instruction. Note that `EQU`, `FCB`, and `ORG` are *assembler directives*, not instructions.

2. Type the program into a text editor and save it as `sim.asm`. Note that AS11 requires hexadecimal numbers to have a leading dollar sign, as shown in the listing above. Use the following command to assemble the program:

   ```
   AS11 sim.asm -l s c cre > sim.lst
   ```

   Correct the code to resolve any warnings or error messages. The assembler will generate two new files, `sim.s19` and `sim.lst`. What do the command line options "`-l s c cre`" do?

3. Look at the listing file `sim.lst`. What is the address of the variable `NUM1`? What is stored at this address?

   Now look at the opcode bytes for the instruction `LDAB NUM2`. Decode these bytes to determine the addressing mode of the instruction. What did the assembler do to the `NUM2` label in this instruction? How did the assembler convert the `GET_SUM` and `GET_DIFF` labels in the two `JSR` instructions?

4. Run the simulator program `Wookie167.exe`. Load the assembled machine code by the menu commands *File - Load .s19 File...* Open `sim.s19`, then choose *Brief Case* mode; the *Start Address* should be `C000` since this is where the executable code begins.

5. Click on the buttons *M68HC11 CPU*, *Browse Mem*, *View Code*, and *Registers* to open the corresponding windows. In the *View Code* window, click *Load LST file* to load the assembler listing `sim.lst` (you may need to specify *Files of type* as "Listing files"). Choose File Type *AS11M*. The assembled code should appear in the window.

6. Check and record (in your report) the contents of the registers and verify that *PC = C000.*

7. **Step-by-step Execution:** Click the "walking boy" button (*Step*) to step through the program one instruction at a time. After each step, check the contents of the registers and memory (especially addresses $0100 through $0104) to see if they agree with your expected results. Reset the microcontroller (via the *Reset* button), change the value of the memory at $0100, and run the program again.

8. **Continuous Execution:** Reset the microcontroller. Change the values stored in memory at $0100 and $0101. Enter the address of the `SWI` instruction into the *Break Point* text box (on the main Wookie window). Click on the "red light" button (*Run*) to start continuous execution of the program; it should stop at the breakpoint you set at the `SWI` instruction. When it stops, check the contents of the registers, memory, and ports to see if they agree with what you expected.

9. What does the `RTS` instruction do? How much memory is used by the entire program (i.e., how many bytes are needed to store the executable code and variables)?

In your lab report include any prelab analysis, the assembler listing, and a comparison of your expected and actual results. Answer any questions asked above in your report, too (do this for all the laboratory exercises). Be sure to include the values you entered at memory locations $0100 and $0101, and the results of all the step-by-step and continuous execution runs.

For the subsequent lab assignments, you should write and test your programs *before* your regular lab session using the AS11 assembler and Wookie simulator. Be sure to **bring your source code to lab on a floppy disk** so you can run it on the 68HC11 EVB hardware.