Embedded systems – Exercise #0

Exercise purpose

Get acquainted with the development environment: compiler, linker and debugger.

Exercise description

1. Download the program source code (ex0.c)
2. Compile ex0.c with full speed optimization to assembly (ex0.s) and then link (ex0.out) and generate a memory-map listing file (ex0.map)
3. Compile ex0.c with source level debug information to assembly (ex0.dbg.s) and then link (ex0.dbg.out) and generate a memory-map listing file (ex0.dbg.map)
4. Using the debugger, do the following:
    a. Trace the code and find out what the “modify” function does.
    b. Use the memory window to watch the “data” array
    c. Set a breakpoint in the “replace” function and see how the array changes any time the array stops
    d. Use a data watchpoint to follow the flow of the program instead of a code breakpoint.
    e. Display the call stack when in the “replace” function
    f. Change the values in the array without recompiling the code.
    g. What happens when the program ends? (Hint - what is the very last instruction executed by the processor [that change the state of the processor] and what does it do? use the disassembly window and the “instruction history” menu option).
    h. Count how many instructions the program executed (with full speed optimization and without) (use the Profiling windows).
5. Examine ex0.dbg.s
    a. What register implements the “out” variable?
    b. What register implements for the “in” variable?

c. Copy ex0.dbg.s to ex0_tag.s

d. In ex0_tag.s, change the code for the “modify” function such that it does not call the “replace” function; instead, it implements it inline

e. Compile ex0_tag.s and test its correctness

6. Submit the following files:

a. ex0.c, ex0_tag.s

b. A 'makefile' (with that name) that will generate the following files:

   i. ex0.s, ex0.out, ex0.map

   ii. ex0.dbg.s, ex0.dbg.out, ex0_tag.out

c. ‘README’ containing answers to the following questions:

   i. What does the function “modify” do?

   ii. What is the last instruction executed by the processor and what does it do? (question 4g. above)

   iii. Look at the memory-map listing file you created in Q.3 and answer the following:

      1. What is the size of your functions (replace, modify, main) in memory when the program runs? (look at “SECTION DETAILS” section)

      2. What is the total size of your executable code in memory? (look at “SECTION SUMMARY” section)

      3. Why there is a difference between the total size of the code you wrote to the actual size of the executable code?

   iv. What registers implement the “in” and “out” variables?

   v. What does the assembly command “asl” do? Check out the ARC6 manual

   vi. What is the machine code for the command: `asl %r0,%r5,2`? Answer with two methods:

      1. Check in the ARC6 manual

2. Check in the debugger by locating the command in the code and viewing the memory
3. Why they aren't the same?