Due on 11/27, submit in hard copy.

1. (20%)

Read the linker script of pic24ep in hw5.q1. Run a pic24ep program in MPLab X, and watch memory of the program in debugger.

1. Fill the table in hw5.q1.docx.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memory Section | Start Address | End Address | Length | The First 4 Bytes in HEX |
| data | 0x1000 | 0xE000 | 0xD000 | FF FF FF FF |
| reset | 0x0 | 0x4 | 0x4 | 00 00 00 00 |
| ivt | 0x4 | 0x200 | 0x1FC | 00 01 02 03 |
| program | 0x200 | 0x557FE | 0x555FE | 6F 96 20 0E |

1. Show a screen shot of watching memory in debugger.



2. (20%) Use avrgcc to compile the code in hw5.q2 with different optimizations: O0, O1, O2, O3, Os.

Makefile is provided, and you may run "make" to get the results.

Otherwise, you can refer to Makefile to choose proper compiler command and options to get results.

Copy and paste your results in the table in hw5.q2.docx.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Optimization | Text | Data | Bss | Dec |
| O0 | 1456 | 0 | 77 | 1533 |
| O1 | 828 | 0 | 77 | 905 |
| O2 | 814 | 0 | 77 | 891 |
| O3 | 814 | 0 | 77 | 891 |
| Os | 818 | 0 | 77 | 895 |

3. (20%) Run the code of hw5.q3 in the simulator of AVR Studio.

Use the stopwatch to profile the time of ledToggle, \_delay\_ms, and configureTimer.

1. Describe how you profile their times, such as where to set break points and how many profiling trials.

The way we achieved the profile times, by adding break points in the main function, where the ledToggle, \_delay\_ms, and configureTimer are called and configured. Also what was implemented was two break point for ledToggle within each while loop. Additionally for the ledToogle one had to change the if condition so it would go into the if statement, otherwise the code portion would be skipped. Another thing that was required in order for the code to work was to comment out the TimmerInt function so it would step into the else condition of the main function.

1. Show the profiling results in report.

**configureTimer**: 34.00 µs

**ledToggle** (first): 93.00 µs

**ledToggle** (second): 40,042.00 µs

**\_delay\_ms:** 40,020.00 µs

4. (20%)

a) Implement the bar function in hw5.q4.c. The bar function should do the same job as the foo function, but unroll the while loop of the foo function.



1. Compile hw5.q4 with no optimization option O0 and run the program in AVR Studio. Use stopwatch to measure the times of running the foo function and the bar function. Report the times of the two functions.

Foo time – 3567 µs

Bar time – 212 µs

I only unrolled the loop 5 times, so I can’t say for sure what the time would be for Bar. Assuming that each set of 5 loops take the same amount of time the Bar function should be 4240 µs

c) Compile hw5.q4 with optimization option O1 and run the program in AVR Studio. Use stopwatch to measure the times of running the foo function and the bar function. Report the times of the two functions.

Foo time – 1518 µs

Bar time – 77 µs

Again, I only unrolled the loop 5 times, so I can’t say for sure what the time would be for Bar. But assuming that every 5 loops take the same amount of time I can calculate that Bar would take 1540﻿μs.

5. (20%)

a) Run the program in hw5.q5 with no optimization option O0 in AVR Stduio. Use stopwatch to measure the times of the crc16\_foo function and the crc16\_bar function. Report the times of the two functions.

Crc16\_foo time for function to complete.

Time: 673µs

Crc16\_bar time for function to complete.

Time: 1173µs

b) Run the program in hw5.q5 with optimization option O1 in AVR Stduio. Use stopwatch to measure the times of the crc16\_foo function and the crc16\_bar function. Report the times of the two functions.

Crc16\_foo time for function to complete.

Time: 75 µs

Crc16\_bar time for function to complete.

Time: 249 µs

c) You may observe that using table is faster than no table with O0, but using table is slower than no table with O1. Analyze the reason for this conflicting result.

The reason for the possible conflicting results is that due to the machine or device requiring excess RAM that it doesn’t have access to. This is related to the register entries not being able to fit in the register memory. At this point what we find is that the compiler is forced to read from RAM, and only if the computer has a big enough cache size it wouldn’t be affected in this case.