Embedded System & Computer Organization

*Assignment 1*

This assignment is worth 20% of your total score for this course.

Demonstrate your solutions and get them evaluated by one of the TAs in the respective lab session.

Four points for the 4th problem, each of the other problems is worth 3 points. No points if you miss the submission date

1. Write an ARM assembly program to WRITE to a file. Write 1 line of text to this file. Make sure you close the file after you write to it. **Submission due Jan 24th or 25th**
2. Write an ARM assembly program to read from the file that you created in Problem 1. Open this file in READ mode, READ and dump its contents to STDOUT. **Submission due on Jan 24th or 25th.**
3. Write an ARM assembly program to receive 3 different Integers (Comma separated) inputs from a file. After receiving all the three values, print them on STDOUT on separate lines. **Submission due on Jan 24th or 25th.**
4. In some single threaded embedded system applications, we use timers built using assembler sub-routine. This type of routines is sufficient and reasonably accurate for many applications. Develop a subroutine “*Stimer*” that can create 1000 (approximately) Clock cycles delay. Using this subroutine write another subroutine “*Ltimer*” that can create delay which are multiples of 10000 (approximately) cycles. The *Ltimer* is simple and no need to other timers. The *Ltimer* can be used to create periodic scan of I/O devices or any periodic activity. **Submission due on Jan 31st or Feb 1st.**
5. Write ARM assembly code that can make the two red LEDs in *Embest Plugin of ARMSIM* to glow alternatively at observable rate. Your code should be user configurable for various rate. User timer developed in problem 4. **Submission due on Feb 21st or 22nd.**
6. Write ARM assembly code that can make the 8-segment display to go from 0 to 9 at an observable rate. Your code should be user configurable for various rate. **Submission due on Feb 21st or 22nd.**

*Assignment 2*

This assignment is worth 20% of your total score for this course.

Demonstrate and get it evaluated by one of the TAs before the end of the respective lab session.

Each problem is worth 3 points except for the first; it is worth 11 points. No points if you miss the submission date

1. Write an ARM assembler program that increments the elements of a vector of size 100. After incrementing, copy the vector to a different part of the memory and add the two vectors and store the result where first vector was in. Repeat the above procedure three more times once with Multiplication (instead of ADD) by 5, once with divide by 4, and finally by adding 16384. Finally copy the vector to a file. Optimize code for execution time using appropriate cache configuration and choosing the appropriate assembly instruction. Solution with best execution time will be recognized suitably.

**Due on Feb 28th or March 1st**

Plot your observations using Excel Graphs for problem 2, 3, and 4

1. Run the above program (1) with ***unified cache with direct mapping and Write back****.* For the following combination of cache size and block size measure the *miss rate*. Plot your results and provide your observation. **Due on March 3rd or 4th.**

|  |  |  |
| --- | --- | --- |
| cache Size | Test 1 | Test 2 |
| Bock Size | Block Size |
| 128 | 16 | 32 |
| 256 | 16 | 32 |
| 512 | 16 | 32 |
| 1024 | 16 | 32 |
| 2K | 16 | 32 |
| 4K | 16 | 32 |
| 8K | 16 | 32 |

1. Run the above program (1) with ***split cache with direct mapping and Write back.***For the following combination of cache size and block size measure the hit and *miss rate*. Plot your results and provide your observation. **Due on March 3rd or 4th.**
2. Run the above program (1) with ***split cache with associative mapping and Write back****.* For the following combination of cache size and block size measure the hit and miss. Plot your results and provide your observation. **Due on March 3rd or 4th.**

**OR (Either Problem 4 or Problem 5 - one will do, however, problem 5 includes bonus point of 5!)**

1. **Develop a set of ARM subroutines to ADD and SUBTRACT floating point IEEE 754 numbers. Read the inputs from a file, write your results to STDOUT.**