Lab 10

Submit your program before the deadline.

1. The average memory access time, AMAT, can be used to measure the memory performance and it includes hit time, miss ratio, and miss penalty (refer to page 478 of textbook/slide 38 of chapter 5).

   ```
   AMAT = Hit time + Miss rate * Miss penalty
   ```

   In this task you will write a program in MIPS assembly language that first defines an array of bytes of size 1024, where each byte is initially set to 0, then somehow 'updates' this memory such that after the update the memory looks as in the table below:

   Initial contents of 1024 bytes (values are in hexadecimal):

Address	value(+0)	value(+4)	value(+8)	value(+c)	...
0x1001000	0x00000000	0x00000000	0x00000000	0x00000000	...

   After update:

Address	value(+0)	value(+4)	value(+8)	value(+c)	...
0x1001000	0x01010101	0x01010101	0x01010101	0x01010101	...

   For example, one of the ways to get as above result is to add 1 to each byte in the array.

   Then using the Data Cache Simulator tool of MARS you need to determine the memory access count, hit rate, miss rate, and calculate AMAT. Then multiply AMAT by the number of memory accesses to get your *metric score*. Assume that hit time is 1 ns and miss penalty is equal to 1 ns + 15 * (# of words) + (# of words) * 1 ns. For example, if the block size is 1 word then, the miss penalty is 1 + 15*1 + 1 = 17 ns (refer to page 471 of textbook/ slide 34 of Chapter 5).

   Note: Bonus points may be given to the top 3 students, who have the lowest *metric score*.

   To activate the Cache and Memory related Tools click on Tools ->Data Cache Simulator. Enable the Runtime Log and then click 'Connect to MIPS'.
   You may change *only* the number of blocks and the cache block size in the Data Cache Simulator Tool. Other parameters like Cache size and Placement policy should be set to the default values.

   Write your optimized parameters next to .text as comments like below (if there are no parameters, it will be graded with the default values):

   ```
   .data
   array: .space 1024
   ...
   .text
   #Number of blocks:
   #Cache block size:
   #YOUR METRIC SCORE:
   #The reasons for my optimization
   #In Assembly code:
   #In the configurations of cache parameters:
   ...
   your code goes here
   ```