Faris Zahrah
S181710@win.dtu.dk

Embedded AES_128
Exercise sheet 02
Exercise 3

Goal: cycle minimization

Specifications with -O3 optimization
Program initialization cycle count: 4039
Key and initial state declarations: 12
Key Scheduling cycle count: 1380
Encryption cycle count:4823
Entire Program cycle count: 10254
Program memory: 2346 bytes 14.3%
Data memory Usage: 451 bytes, 44.0%

The code is well documented to support the following implementation choices.

The overall architecture of the code is quite inline with the structure of AES, the key scheduling is computed all at once so that repeated function calling does not incur excessive cycles. From there the program pre-whiten with the add round key. The next 9 rounds follow the protocol or sub_bytes, shift_rows, mix_columns, add_roundkey. The final round is the same as the prior nine without the mix_columns function.

Strategies for cycle reduction:

Unfolding loops, and 'jamming loops'(as referred to in lecture slides) mean the program is more efficient in terms of what items are cached based on spatial locality. The optimizer setting of -O3 seemed to do a level of this optimization automatically because when certain loops were flattened, (the *subByetsState* for loop for example) it actually increased the cycle count. Also flattening the last for loop in the *computRoundKey* function did not reduce cycle counts on -O3 either.

Declaring variables to be static at every possible with the intent on making compilation faster and reduce the cycles it takes to fetch and write to these variables. this is because they are stored lower in the stack which means accessing them costs less cycles. This will result in greater improvement for variables that are accessed with high frequency, so the SBOX is a prime example of an array that sees performance increase when declared static. Additionally, removing *const* from any variables even if they were only being read from was chosen to reduce cycles(based on lecture).