| galois |

Background: Assembly program for multiplication

Here is an assembly language program for multiplying two 8 bit numbers. Expanding this to 16, 32, or 64 bits is a relatively simple task. This lab sticks to 8 bit numbers because all that can be learned can be learned from the 8 bit version and proofs will be very fast.

```
step01:       LDX #8      ; load X immediate with the integer 8
step02:       LDA #0      ; load A immediate with the integer 0
step03:       CLC         ; set C to 0
step04: LOOP  ROR F1      ; rotate F1 right circular through C
step05:       BCC ZCOEF   ; branch to ZCOEF if C = 0
step06:       CLC         ; set C to 0
step07:       ADC F2      ; set A to A+F2+C and C to the carry
step08: ZCOEF ROR A       ; rotate A right circular through C
step09:       ROR LOW     ; rotate LOW right circular through C
step10:       DEX         ; set X to X-1
step11:       BNE LOOP    ; branch to LOOP if Z = 0
```

The input to the multiplier is two 8 bit numbers in 8 bit registers `F1` and `F2`. The output is in registers `A` (also known as `HI`) and `LOW`. The result of the multiplication is `HI*256 + LOW`. Flag `C` is the carry bit which can be changed in a number of ways, particularly in addition as in `step07`, or due to a rotation (`ROR`) as in `step08`, `step09` and `step04`, or through the `CLC` instruction which clears the carry bit. The `Z` or zero bit is used to terminate the program. The `X` register holds the number of iterations of the `LOOP` remaining and decrements eventually to 0 due to the `DEX` instruction. When `X` becomes `0` `Z` becomes 1. Then the `BNE LOOP` test fails and the program ends. Comments are detailed to explain the operation of each step. The state of the multiplier at any time during operation is determined from the `X`, `A`, and `LOW` registers, the carry bit, the zero bit, and the inputs `F1` and `F2`.

The algorithm is pretty much like the multiplication scheme one learns in grade school except here bits are the multiplier and multiplicand so multiplication entails a bit addition, then a shift of one bit followed by an addition, and so on for 8 times.