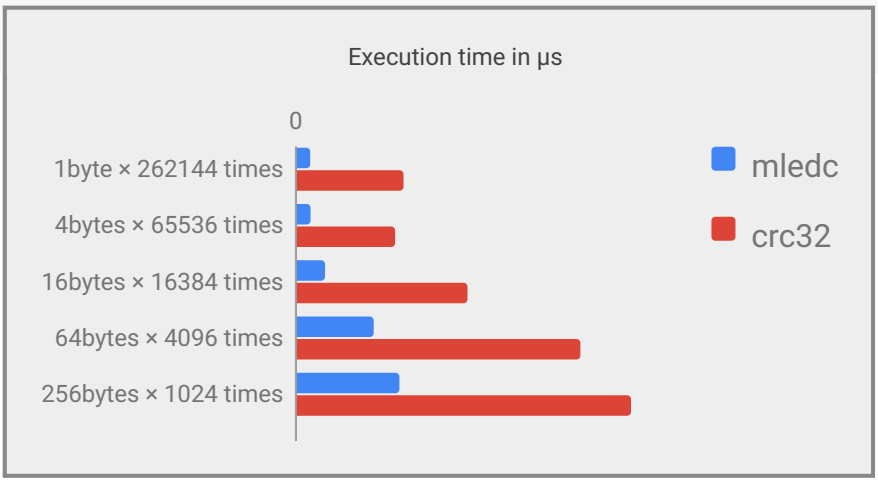
計算量



計算内容が簡単な上にメモリアクセスが少ないので、計算量は CRC32 と比べるとだいぶ少ない。
データのフェッチが 16bit なのも大きいと思う。
手元での測定(MacBook Pro (13-inch, 2017, Two Thunderbolt 3 ports), 右図)の結果を見ると、CRC32 よりだいぶ速い。

計算内容

データの取得

データは 2byte ずつ取得する。端数がある場合は末尾にもう 1byte ゼロがあることにする。
→ `0x12, 0x34, 0x56` と `0x12, 0x34, 0x56, 0x00` の区別はつかない。

というわけで、入力バイト数の半分(端数切り上げ)個の、符号なし16ビット整数が手に入る

計算

計算に必要な定数

以下の定数を必要とする。

変数名	説明	補足
`init`	初期値	2進数で0と1がいい感じに混ざっている値がいいんじゃないかと思う。
`mul`	乗数	2進数で0と1がいい感じに混ざっている **素数** がいいんじゃないかと思う。

計算に必要な変数

実質的に 32bit 符号なし整数1個。この変数の名前を `c` とする。

初期化

`c` を `init` で初期化する

更新

符号なし16bit整数の入力データ x を受け取り、以下の計算をする:

$$c \leftarrow rol(c) \times mul + x$$

関数 rol は、1bit 左ローテート。数学っぽく書くと以下の通り:

$$rol(x) = mod(\lfloor x \times 2 + x \div 2^{31} \rfloor,\ 2^{32})$$

関数 mod は剰余関数。数学っぽく書くと以下の通り:

$$mod(a,\ n) = a - n \lfloor a \div n \rfloor$$

誤り検出性能

正直言って、よくわからない。

1〜255 バイトまでのデータを乱数で作り、1〜32bit ぐらいのノイズを付加し、誤り検出符号の値が変わるかどうか、1200万回ぐらい調べたが、全てのケースでエラーを検出した。

実験結果は悪くないので、そんなに酷いってことはないわけだけど、どれほど有意義な計算なのかはわからない。