gcd 함수

a가 0이면 b가, b가 0이면 a가 최대공약수 이므로 a == 0 || b == 0일 경우 a+b를 return해 주고 (a나 b 둘 중 하나는 무조건 0이므로 더해도 상관 없음), a나 b가 0이 아닐 경우 a는 b가 되고, b는 a % b 되도록 temp 변수에 a 값을 저장해서 계산해 준다.

Xgcd 함수

변수 d0, d1, x0, x1, y0, y1, q를 정의하고, while 문 안에서 q = d0 div d1, d2 = d0 – q * d1, x2 = x0 – q * x1, y2 = y0 – q * y1를 계산한다. 만약 d1이 0이라면 최대공약수 d0값(d1 == 0이 되기 전 단계가 최대 공약수이기 때문이다.)을 return하고 x값과 y값을 전달한다.

mul_inv 함수

while문 안에서 q = d0 / d1, d0 = d0 – q * d1, SWAP(d0, d1), x0 = x0 – q * x1, SWAP(x0, x1)를 계산하고, (SWAP은 temp 변수를 이용해 계산한다) 만약 d1이 1이라면(a와 m이 서로소라면) x1을 return한다. 이 때, x1이 음수이면 m을 더해 양수로 변환해 준 후 return한다. d1이 1이 아닌 경우 역이 없으므로 0을 return한다.

umul_inv 함수

umul_inv 함수의 경우 입출력이 64비트의 양수 정수임을 고려한 후 mul_inv 함수를 응용하여 함수를 제작하였다. Input 변수를 모두 unit64_t로 정의하였고, While 문 안의 계산 과정은 mul_inv의 코드를 이용하였다. Output인 return 값의 경우 x1이 64비트가 존재할 경우 x1+m, 존재하지 않을 경우 x1을 return한다.

gf16_mul 함수

x^16 = x^5+x^3+x+1 (mod x^16+x^5+x^3+x+1) 특성을 이용하였다.

Xtime 함수의 경우 만약 x >> 15(x^15)가 존재한다면 1이므로 1 & 1 = 1이 될 것이고, 존재하지 않는다면 0 & 1 = 0이 될 것이다. 만약 1이 된다면 x << 1(x^15를 제외한 나머지)와 0x2B(x^5+x^3+x+1를 16진수로 바꿔준 값)를 XOR 한 후 return하고, 0이 된다면 x << 1을 return 한다. Gf16_mul 함수는 비트를 하나씩 shift하고 xtime 함수를 거쳐 b가 0이 될 때까지 반복한다.

gf16_pow 함수

gf16_mul함수를 이용하였다. B가 0이 될 때까지 b가 1이면 r XOR a를 gf16_mul 함수를 이용해 계산한 후 b를 하나씩 shift하고, a값 또한 gf16_mul 함수를 이용해 계산해 주었다.

컴파일 과정

```
lejinhy@DESKTOP-IPANOF9:/mnt/c/Users/82103/Downloads/proj#1$ make clean
rm -rf *.o
rm -rf test
lejinhy@DESKTOP-IPANOF9:/mnt/c/Users/82103/Downloads/proj#1$ make all
gcc -Wall -O3 -c test.c
gcc -Wall -O3 -c euclid.c
gcc -o test test.o euclid.o  -lbsd
```

Test.c 실행 결과

```
lejinhy@DESKTOP-IPANOF9:/mnt/c/Users/82103/Downloads/proj#1$ ./test
========== 기본 gcd 시험 ==========
gcd(28,0) = 28
gcd(0,32) = 32
gcd(41370,22386) = 42
gcd(22386,41371) = 1
========== 기본 xgcd, mul_inv 시험 ==========
42 = 41370 * -204 + 22386 * 377
41370^-1 mod 22386 = 0, 22386^-1 mod 41370 = 0
1 = 41371 * 4285 + 22386 * -7919
41371^-1 mod 22386 = 4285, 22386^-1 mod 41371 = 33452
========== 무작위 xgcd, mul_inv 시험 ==========
................................................................................................
................................................................................................
.....................................................PASSED
========== 기본 GF(2^16) a*b 시험 ==========
3 * 7 = 9
65535 * 12345 = 41504
========== 기본 umul_inv 시험 ==========
a = 5, m = 9223372036854775808, a^-1 mod m = 5534023222112865485.....PASSED
a = 17, m = 9223372036854775808, a^-1 mod m = 8138269444283625713.....PASSED
a = 85, m = 9223372036854775808, a^-1 mod m = 9006351518340545789.....PASSED
========== 무작위 umul_inv 시험 ==========
................................................................................................
................................................................................................
.....................................................PASSED
========== 전체 GF(2^16) a*b 시험 ==========
................................................................................................
................................................................................................
.....................................................PASSED
```

설명&소감

최대공약수, 곱의 역, GF를 구하는 알고리즘에도 여러가지가 있고, 알고리즘 값을 비교하며 같은 값이 나오는지, 혹은 잘못된 구현으로 다른 값이 나오는지 확인하였다.

수업 시간에 배운 이론적인 내용들을 직접 구현하여 올바른 결과가 나오는 것이 신기하기도 했고 굉장히 뿌듯했다. 수업을 들을 때 이해했던 내용이더라도 문제가 조금만 바뀌면 그 문제를 제대로 해결하지 못한 적이 많았는데, 이번 과제에서 8비트를 16비트로 바꾸어 구현하는 함수를 다루며 더욱 정확하게 알고리즘을 이해할 수 있었다. 또한 test 코드는 내가 구현한 함수들이 정확한지 파악할 수 있어 편리했다.

문제점

현재 수업시간에 배운 알고리즘을 다루는데 무리가 없었기 때문에 큰 문제점은 없다고 생각한다. 한가지 보완점이 있다면, input값을 유동적으로 바꾸어도 오류 없이 실행될 수 있도록 함수를 수정하여 다양한 type의 input을 받을 수 있게 보완할 수 있다. Proj1에서 mul 함수의 input은 16비트로 고정되어 있는데, 16비트 이외에도 8비트 등등 다양한 비트의 input이 들어올 수 있다면 더욱 정확한 함수가 될 것이다.