# SHA2 해시함수

https://youtu.be/J7ZojhDyIZA

IT융합공학부 송경주

# SHA-2

- NSA( National Security Agency )에서 2001년 공개한 해시함수
- SHA-1을 대체하기 위해 새로 고안됨
- 블록 암호의 Davies–Meyer 구조를 사용하여 설계됨
- SHA2 해시함수군:
  : SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/ 224, SHA-512/256

| 구분 | 출력 크기 | 내부 상태 | 블록 크기 | 최대 길이 | 워드 | 라운드 수 |
|---|---|---|---|---|---|---|
| SHA-224 | 224-bit | 256-bit | 512-bit | $2^{64}-1$ bit | 32-bit | 64 |
| SHA-256 | 256-bit | 256-bit | 512-bit | $2^{64}-1$ bit | 32-bit | 64 |
| SHA-384 | 384-bit | 512-bit | 1024-bit | $2^{128}-1$ bit | 64-bit | 80 |
| SHA-512 | 512-bit | 512-bit | 1024-bit | $2^{128}-1$ bit | 64-bit | 80 |

| Published in | Year | Attack method | Attack | Variant | Rounds | Complexity |
|---|---|---|---|---|---|---|
| *New Collision Attacks Against Up To 24-step SHA-2*[32] | 2008 | Deterministic | Collision | SHA-256 | 24/64 | $2^{28.5}$ |
| | | | | SHA-512 | 24/80 | $2^{32.5}$ |
| *Preimages for step-reduced SHA-2*[33] | 2009 | Meet-in-the-middle | Preimage | SHA-256 | 42/64 | $2^{251.7}$ |
| | | | | | 43/64 | $2^{254.9}$ |
| | | | | SHA-512 | 42/80 | $2^{502.3}$ |
| | | | | | 46/80 | $2^{511.5}$ |
| *Advanced meet-in-the-middle preimage attacks*[34] | 2010 | Meet-in-the-middle | Preimage | SHA-256 | 42/64 | $2^{248.4}$ |
| | | | | SHA-512 | 42/80 | $2^{494.6}$ |
| *Higher-Order Differential Attack on Reduced SHA-256*[2] | 2011 | Differential | Pseudo-collision | SHA-256 | 46/64 | $2^{178}$ |
| | | | | | 33/64 | $2^{46}$ |
| *Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family*[1] | 2011 | Biclique | Preimage | SHA-256 | 45/64 | $2^{255.5}$ |
| | | | | SHA-512 | 50/80 | $2^{511.5}$ |
| | | | Pseudo-preimage | SHA-256 | 52/64 | $2^{255}$ |
| | | | | SHA-512 | 57/80 | $2^{511}$ |
| *Improving Local Collisions: New Attacks on Reduced SHA-256*[35] | 2013 | Differential | Collision | SHA-256 | 31/64 | $2^{65.5}$ |
| | | | Pseudo-collision | SHA-256 | 38/64 | $2^{37}$ |
| *Branching Heuristics in Differential Collision Search with Applications to SHA-512*[36] | 2014 | Heuristic differential | Pseudo-collision | SHA-512 | 38/80 | $2^{40.5}$ |
| *Analysis of SHA-512/224 and SHA-512/256*[37] | 2016 | Differential | Collision | SHA-256 | 28/64 | practical |
| | | | | SHA-512 | 27/80 | practical |
| | | | Pseudo-collision | SHA-512 | 39/80 | practical |

**취약점 발견**

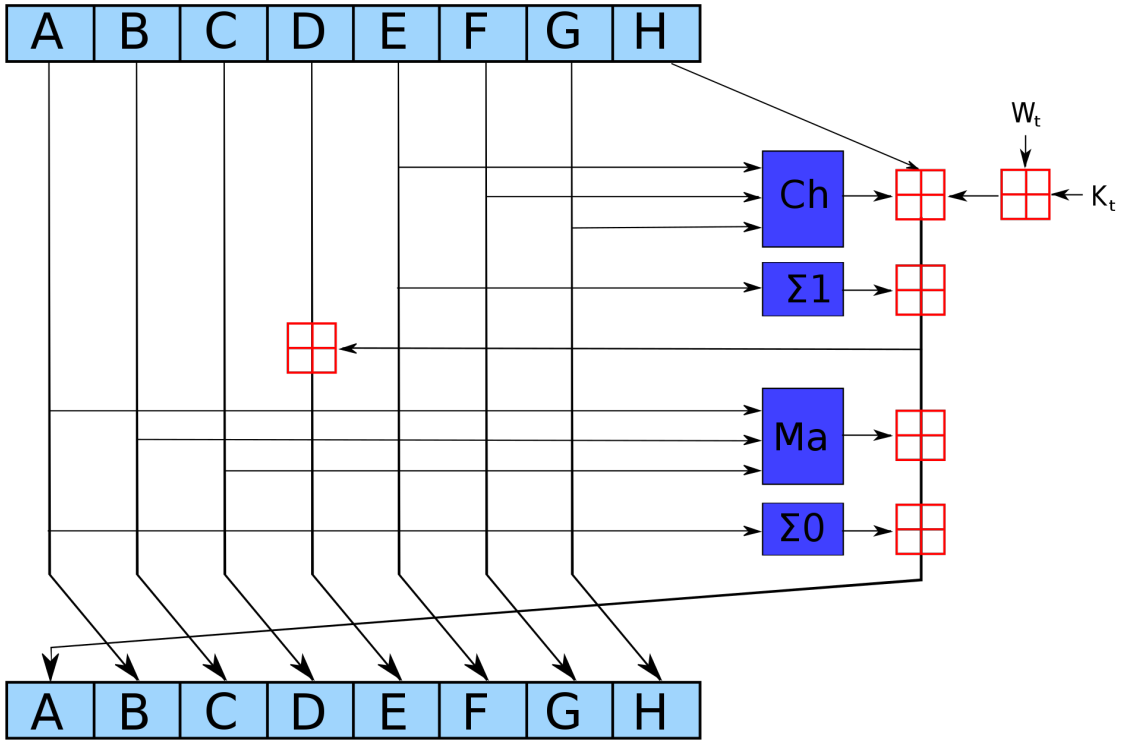
# SHA2

- 임의의 길이 메시지에 대해 224-bit, 256-bit, 384-bit, 512-bit 의 고정 길이 해시를 출력함

- 입력된 메시지는 512의 배수로 패딩되어 512비트의 chunk 들로 나뉨 (16개의 32bit 워드)

- A, B, C, D, E, F, G, H : 32비트의 워드 → 총 256 비트 상태에서 동작

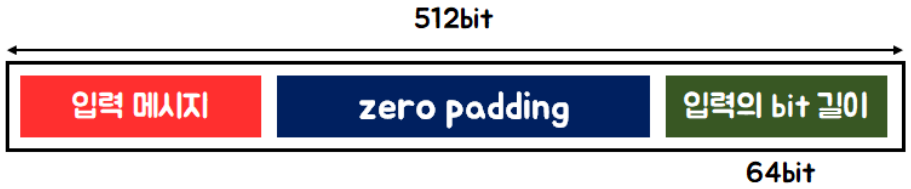
- A, B, C, D, E, F, G, H 레지스터는 Choice, Sigma, Majority, ADD를 통해 연산이 진행됨



# SHA-256

<메시지 패딩>

메시지를 512bit의 배수 단위로 패딩




Input : ‘abc’ → a = 0110 0001, b = 0110 0010, c = 0110 0011

입력의 bit 길이 : 8bit×3 =24bit → 0001 1000

```
01100001 01100010 01100011 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00011000
```


<메시지 패딩>

메시지를 512bit의 배수 단위로 패딩

Input : ‘abc’ → a = 0110 0001, b = 0110 0010, c = 0110 0011

입력의 bit 길이 : 8bit×3 =24bit → 0001 1000

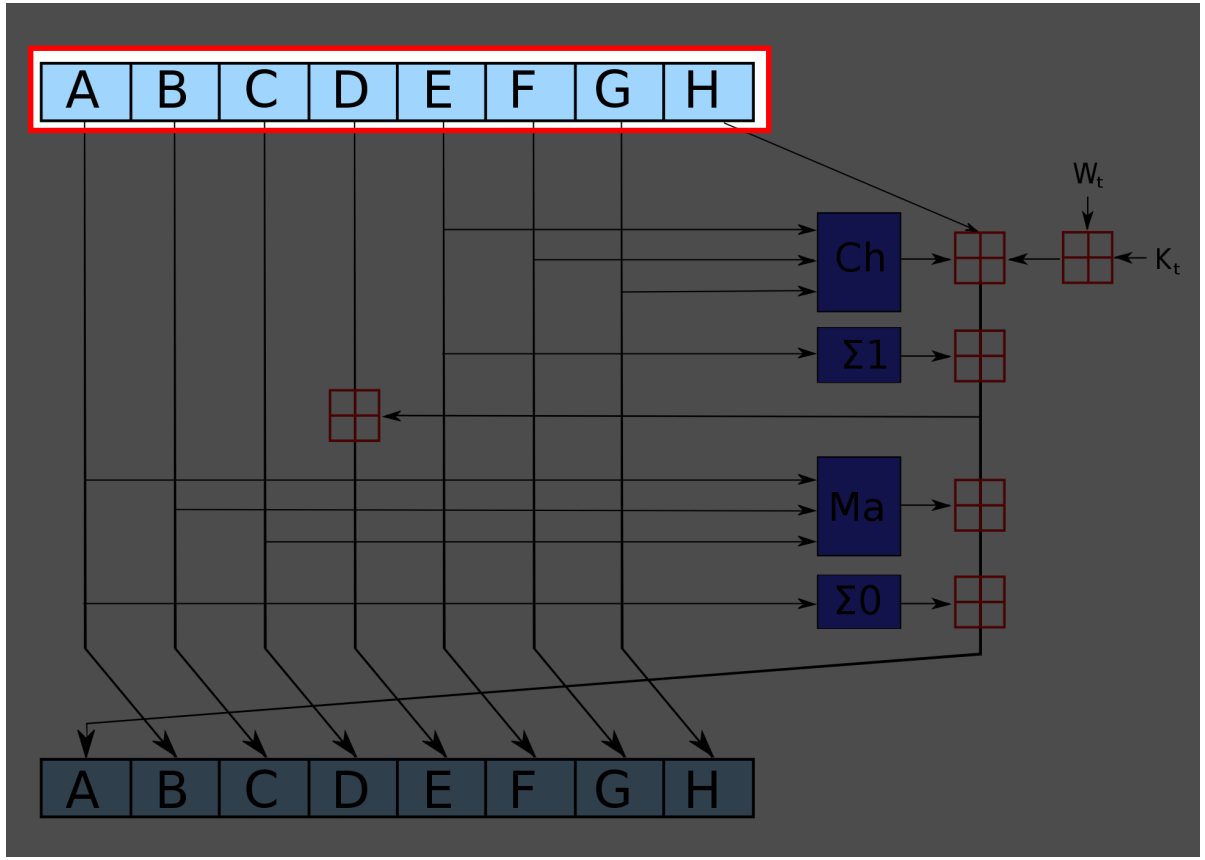
# SHA-256

<초기값 설정>

- A, B, C, D, E, F, G, H : 각 32비트 워드

**가장 작은 8개의 소수** : 2, 3, 5, 7, 11, 13, 17, 19 **의 제곱근의 소수점 이하 32bit 사용**

```
Initialize hash values:
(first 32 bits of the fractional
h0 := 0x6a09e667
h1 := 0xbb67ae85
h2 := 0x3c6ef372
h3 := 0xa54ff53a
h4 := 0x510e527f
h5 := 0x9b05688c
h6 := 0x1f83d9ab
h7 := 0x5be0cd19
```



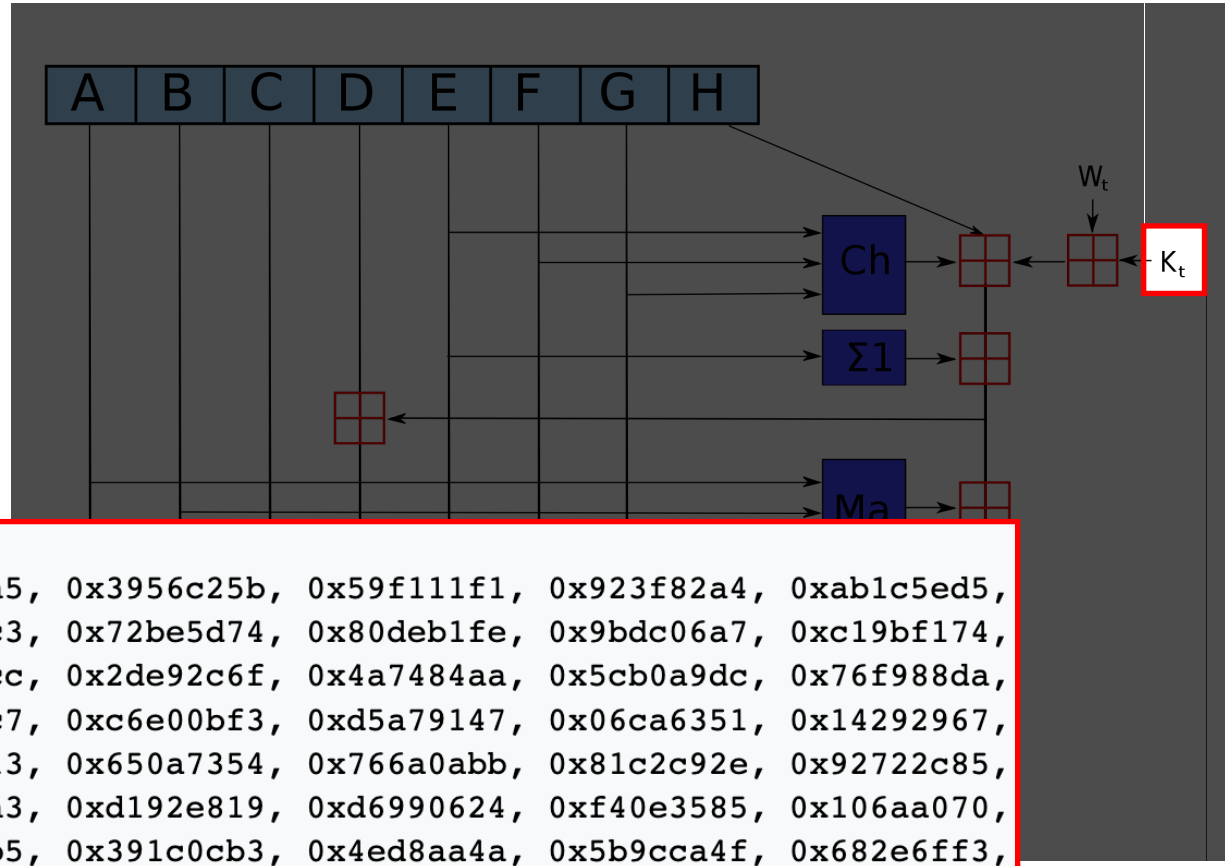
# SHA-256

## <초기값 설정>

**가장 작은 64개의 소수 :**

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59,
61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127,
131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191,
193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257,
263, 269, 271, 277, 281, 283, 293, 307, 311]

**의 세제곱근 소수점 이하 32bit 사용**




```
k[0..63] :=
   0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
   0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
   0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
   0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
   0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
   0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
   0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
   0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2
```

# SHA-256

## <내부 연산>

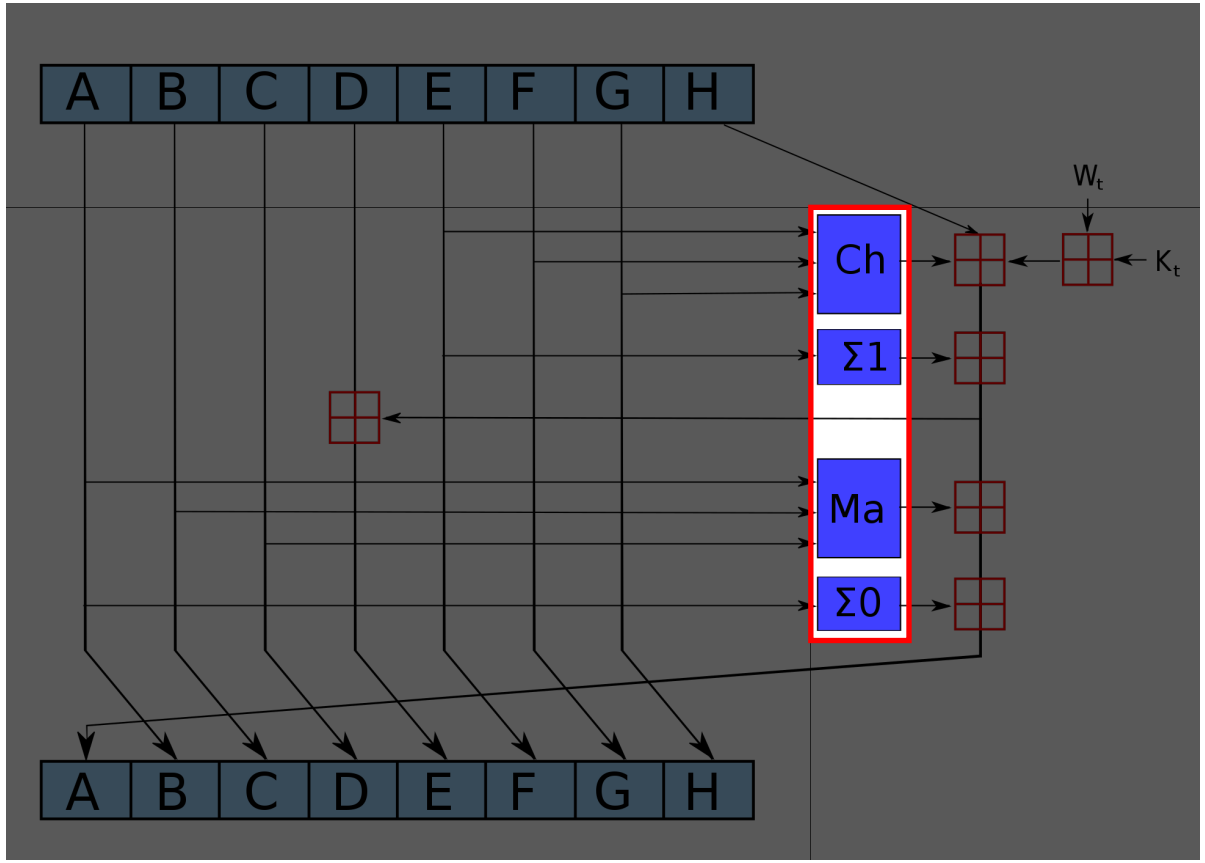
```
#define RR(x, n)        ROTR_ULONG(x, n)
#define SS(x, n)        (x >> n)

#define Ch(x, y, z)        ((x & y) ^ ((~x) & z))
#define Maj(x, y, z)    ((x & y) ^ (x & z) ^ (y & z))
#define Sigma0(x)        (RR(x,  2) ^ RR(x, 13) ^ RR(x, 22))
#define Sigma1(x)        (RR(x,  6) ^ RR(x, 11) ^ RR(x, 25))

#define RHO0(x)            (RR(x,  7) ^ RR(x, 18) ^ SS(x,  3))
#define RHO1(x)            (RR(x, 17) ^ RR(x, 19) ^ SS(x, 10))
```

| SHA-256 |
| --- |
| Ch (E, F, G) = (E & F) ⊕ (~E & G)<br>Ma (A, B, C) = (A & B) ⊕ (A & C) ⊕ (B & C)<br>(A) = (A >>> 2) ⊕ (A >>> 13) ⊕ (A >>> 22)<br>(E) = (E >>> 6) ⊕ (E >>> 11) ⊕ (E >>> 25) |

| SHA-512 |
| --- |
| Ch (E, F, G) = (E & F) ⊕ (~E & G)<br>Ma (A, B, C) = (A & B) ⊕ (A & C) ⊕ (B & C)<br>(A) = (A >>> 14) ⊕ (A >>> 18) ⊕ (A >>> 41)<br>(E) = (E >>> 28) ⊕ (E >>> 34) ⊕ (E >>> 39) |

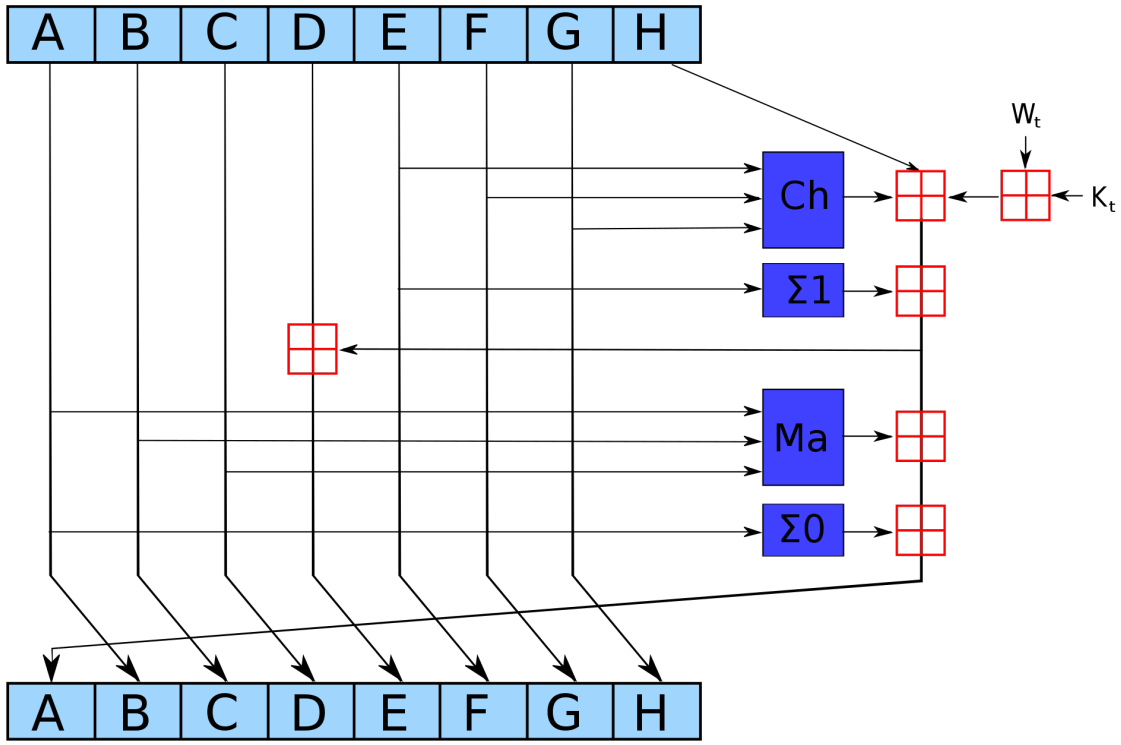


# SHA-256

## <내부 연산>

```
for (j = 0; j < 64; j += 8)
{
    FF(a, b, c, d, e, f, g, h, j + 0);
    FF(h, a, b, c, d, e, f, g, j + 1);
    FF(g, h, a, b, c, d, e, f, j + 2);
    FF(f, g, h, a, b, c, d, e, j + 3);
    FF(e, f, g, h, a, b, c, d, j + 4);
    FF(d, e, f, g, h, a, b, c, j + 5);
    FF(c, d, e, f, g, h, a, b, j + 6);
    FF(b, c, d, e, f, g, h, a, j + 7);
}
```

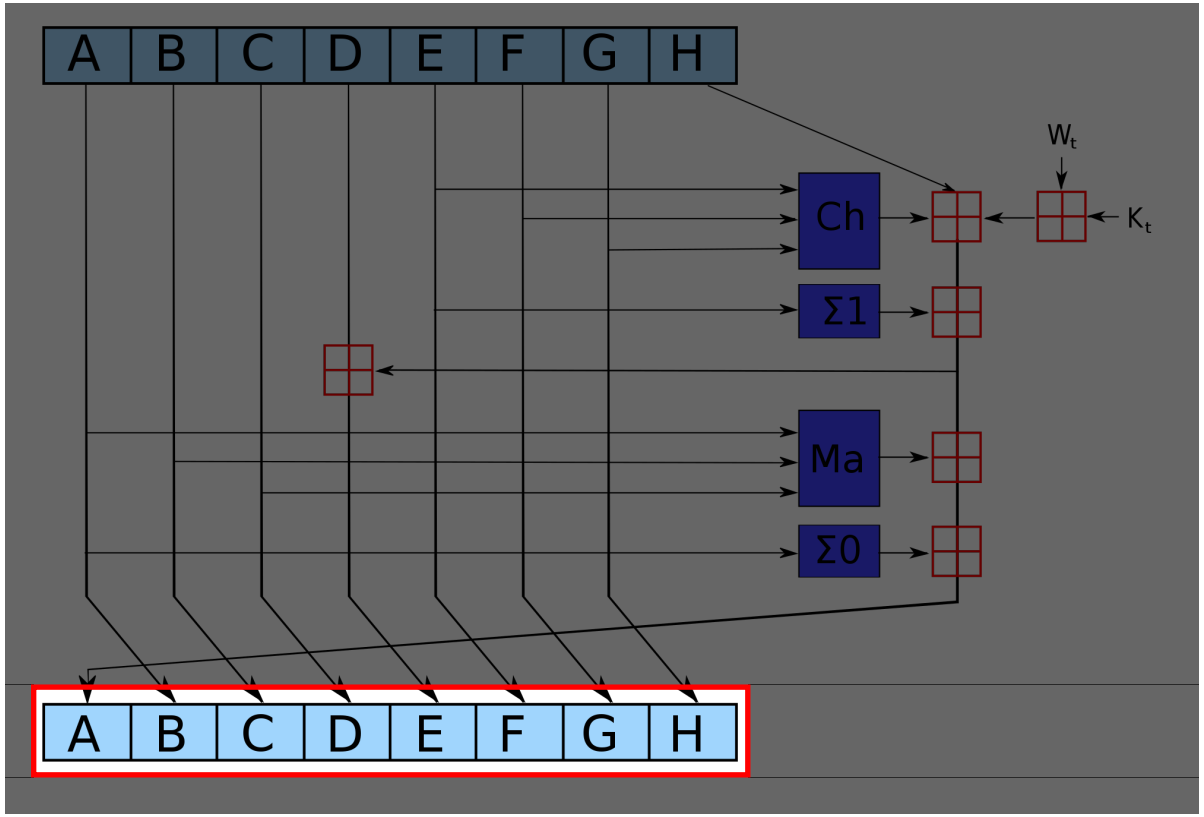
```
#define FF(a, b, c, d, e, f, g, h, j) {
    T1 = h + Sigma1(e) + Ch(e, f, g) + SHA256_K[j] + X[j];
    d += T1;
    h = T1 + Sigma0(a) + Maj(a, b, c);
}
```



# SHA-256

<해시 출력>

```
ChainVar[0] += a;
ChainVar[1] += b;
ChainVar[2] += c;
ChainVar[3] += d;
ChainVar[4] += e;
ChainVar[5] += f;
ChainVar[6] += g;
ChainVar[7] += h;
```



# Q & A