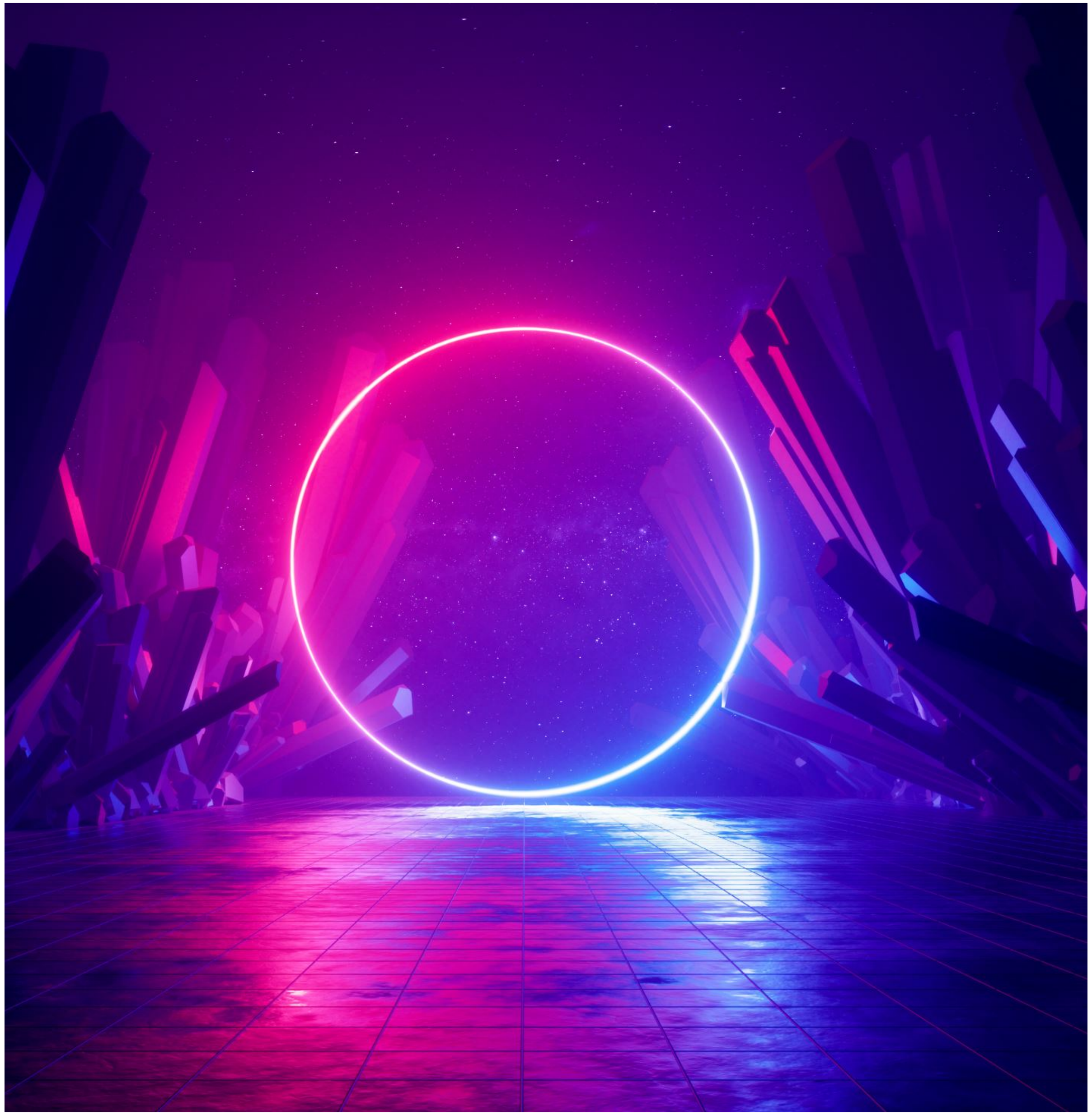


Applied Cryptography

SHA Implementation

Done by :

Ramnarayanan Sankar - 801409708

Poojitha Jayareddygari - 801426875

Shashank Kolluru - 801421839

PROBLEM STATEMENT

Improving the performance of the SHA program which is written in C.

OVERVIEW OF THE FILES ATTACHED IN THE ZIP FOLDER

shaO.c - Original File which was given by Professor.

shaM.c - Modified File which contains the Optimized C program Code.

README.txt – This Readme File contains the Details of the Program in Depth.

How to Run the shaM.c program ?

There are two Steps for the Running the shaM.c file program:

Step 1: Compilation Part for shaM.c file

Give the command as **gcc shaM.c -o programM**

where,

shaM is the C program file.

programM is the Compiled File.

Step 2: Execution Part for shaM.c file

Give the command as **./programM**

where,

programM is the file which is going to show output.

How to Run the shaO.c program ?

There are two Steps for the Running the shaO.c file program:

Step 1: Compilation Part for shaO.c file

Give the command as **gcc shaO.c -o programO**

where,

shaO is the C program file.

programO is the Compiled File.

Step 2: Execution Part for shaO.c file

Give the command as **./programO**

where,

programO is the file which is going to show output.

How we approached the Optimization for the program ?

When we Saw the program, We decided to change only the FOR Loops.

Because Loops has O(n) Time Complexity and Function blocks without the Looping Statements is going to have O(1).

Where ‘(1)' Constant time taken for running the Function block

Where ‘(n)' represents the Number of times the loop is going to run.

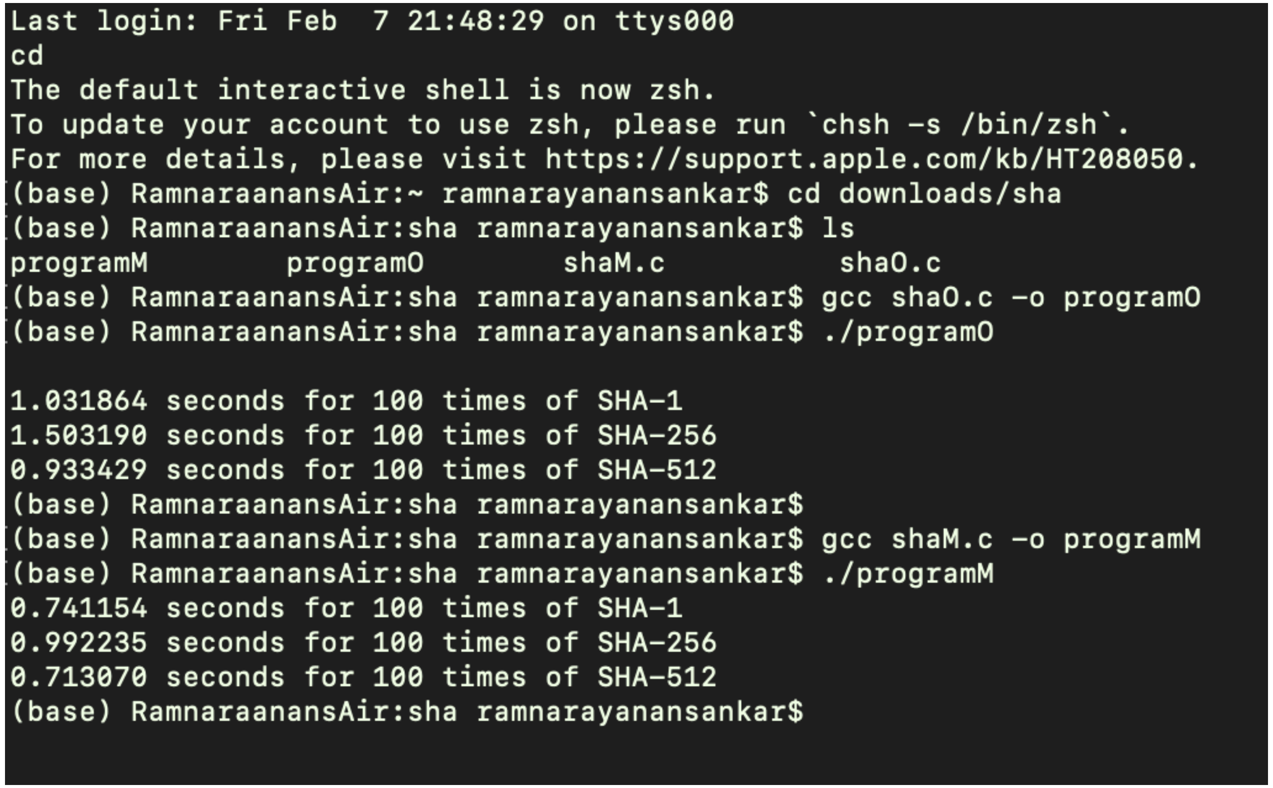
What we did in the program to get the Optimized one ?

We removed For Loops from the functions in the program.

We then manually run the looping statements as the number of times required for the function Block of the program.

Finally, We reduced the Running time of the program.

SOURCE CODE RUNTIME BEFORE AND AFTER THE CHANGES

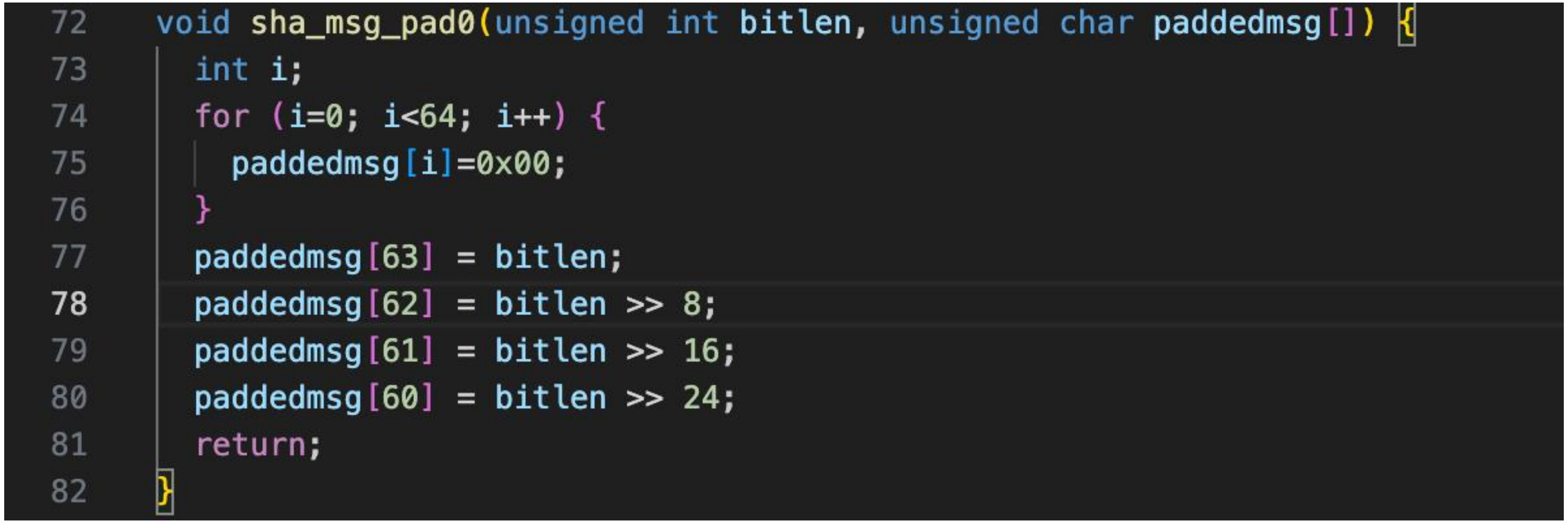


```
Last login: Fri Feb  7 21:48:29 on ttys000
cd
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) RamnaraanansAir:~ ramnarayanansankar$ cd downloads/sha
(base) RamnaraanansAir:sha ramnarayanansankar$ ls
programM        programO        shaM.c          shaO.c
(base) RamnaraanansAir:sha ramnarayanansankar$ gcc shaO.c -o programO
(base) RamnaraanansAir:sha ramnarayanansankar$ ./programO

1.031864 seconds for 100 times of SHA-1
1.503190 seconds for 100 times of SHA-256
0.933429 seconds for 100 times of SHA-512
(base) RamnaraanansAir:sha ramnarayanansankar$
(base) RamnaraanansAir:sha ramnarayanansankar$ gcc shaM.c -o programM
(base) RamnaraanansAir:sha ramnarayanansankar$ ./programM
0.741154 seconds for 100 times of SHA-1
0.992235 seconds for 100 times of SHA-256
0.713070 seconds for 100 times of SHA-512
(base) RamnaraanansAir:sha ramnarayanansankar$
```

Source Code before changes -

void sha_msg_pad0



```
72  void sha_msg_pad0(unsigned int bitlen, unsigned char paddedmsg[]) {
73      int i;
74      for (i=0; i<64; i++) {
75          paddedmsg[i]=0x00;
76      }
77      paddedmsg[63] = bitlen;
78      paddedmsg[62] = bitlen >> 8;
79      paddedmsg[61] = bitlen >> 16;
80      paddedmsg[60] = bitlen >> 24;
81      return;
82  }
```

The 74 to 76 lines are commented and modified

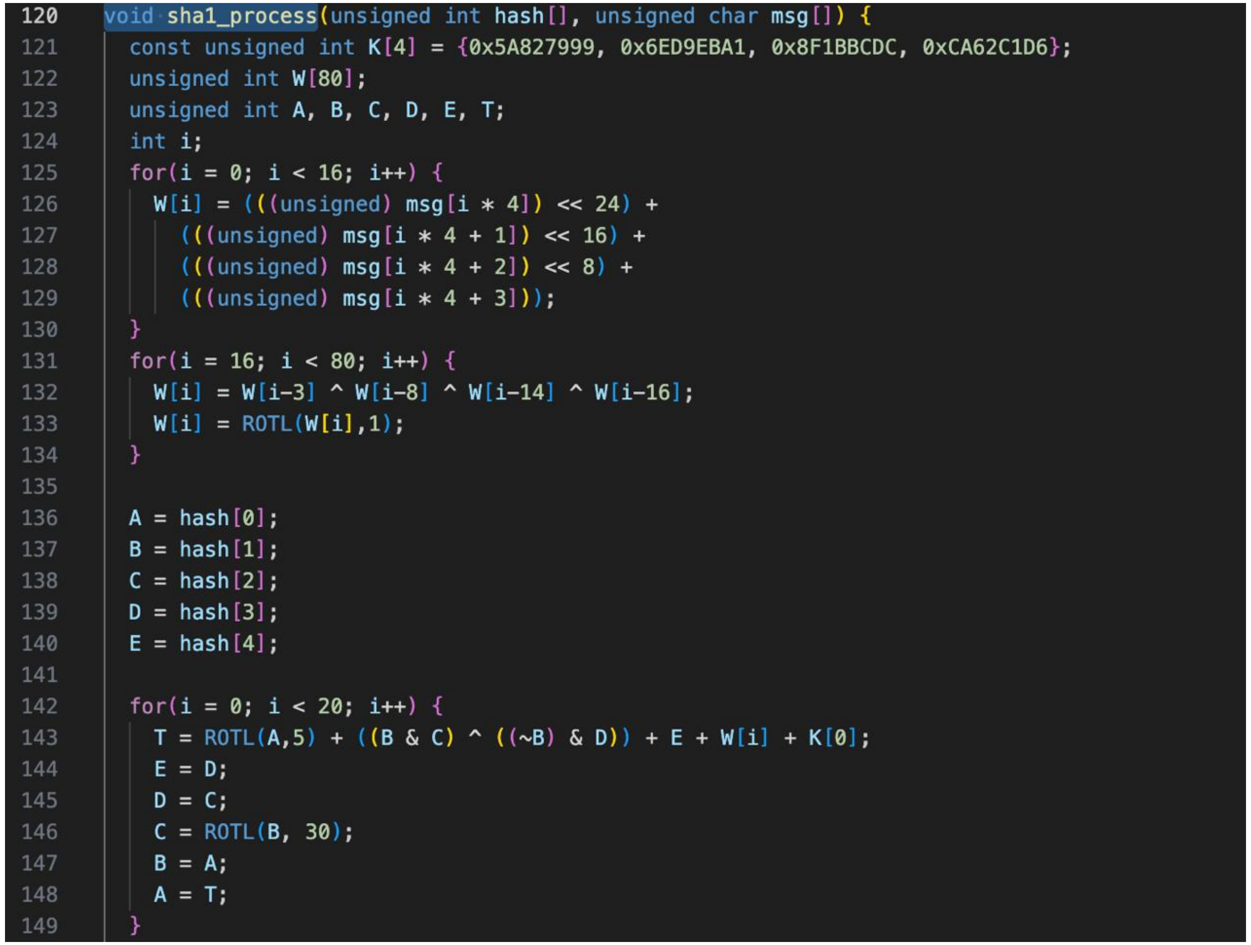
Source Code after changes -

void sha_msg_pad0

Line 77 to line 140 has been added in the file shaM.c

Source Code before changes -

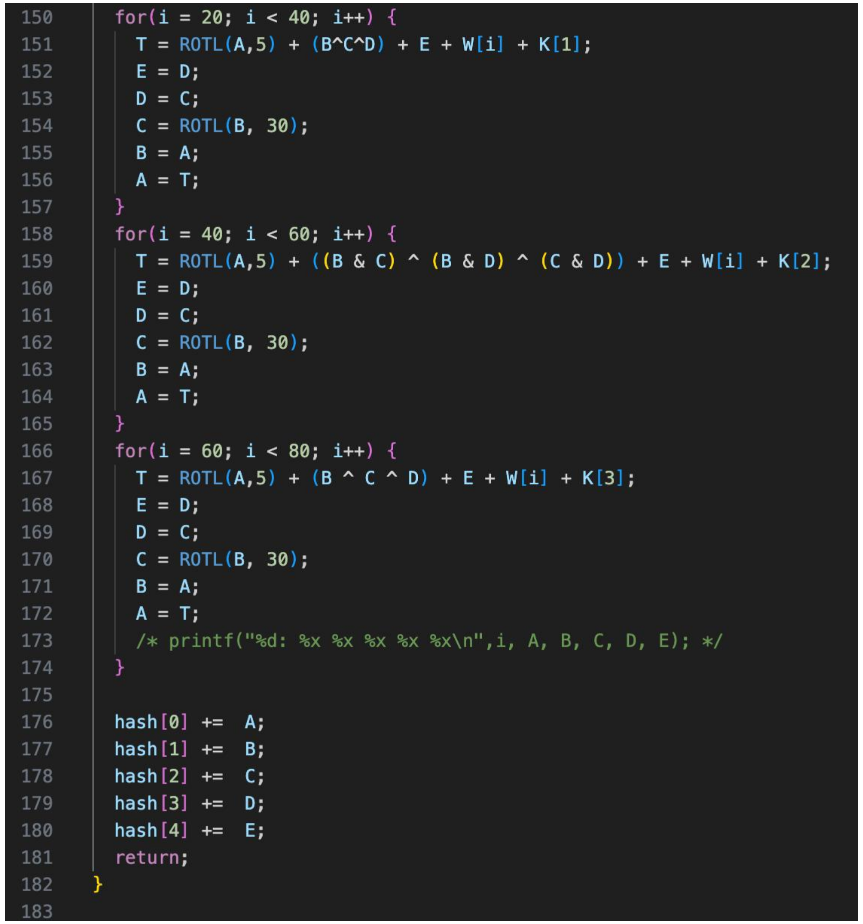
void sha1_process



```c
void sha1_process(unsigned int hash[], unsigned char msg[]) {
  const unsigned int K[4] = {0x5A827999, 0x6ED9EBA1, 0x8F1BBCDC, 0xCA62C1D6};
  unsigned int W[80];
  unsigned int A, B, C, D, E, T;
  int i;
  for(i = 0; i < 16; i++) {
    W[i] = (((unsigned) msg[i * 4]) << 24) +
      (((unsigned) msg[i * 4 + 1]) << 16) +
      (((unsigned) msg[i * 4 + 2]) << 8) +
      (((unsigned) msg[i * 4 + 3]));
  }
  for(i = 16; i < 80; i++) {
    W[i] = W[i-3] ^ W[i-8] ^ W[i-14] ^ W[i-16];
    W[i] = ROTL(W[i],1);
  }

  A = hash[0];
  B = hash[1];
  C = hash[2];
  D = hash[3];
  E = hash[4];

  for(i = 0; i < 20; i++) {
    T = ROTL(A,5) + ((B & C) ^ ((~B) & D)) + E + W[i] + K[0];
    E = D;
    D = C;
    C = ROTL(B, 30);
    B = A;
    A = T;
  }
```



```c
  for(i = 20; i < 40; i++) {
    T = ROTL(A,5) + (B^C^D) + E + W[i] + K[1];
    E = D;
    D = C;
    C = ROTL(B, 30);
    B = A;
    A = T;
  }
  for(i = 40; i < 60; i++) {
    T = ROTL(A,5) + ((B & C) ^ (B & D) ^ (C & D)) + E + W[i] + K[2];
    E = D;
    D = C;
    C = ROTL(B, 30);
    B = A;
    A = T;
  }
  for(i = 60; i < 80; i++) {
    T = ROTL(A,5) + (B ^ C ^ D) + E + W[i] + K[3];
    E = D;
    D = C;
    C = ROTL(B, 30);
    B = A;
    A = T;
    /* printf("%d: %x %x %x %x %x\n",i, A, B, C, D, E); */
  }

  hash[0] +=  A;
  hash[1] +=  B;
  hash[2] +=  C;
  hash[3] +=  D;
  hash[4] +=  E;
  return;
}
```

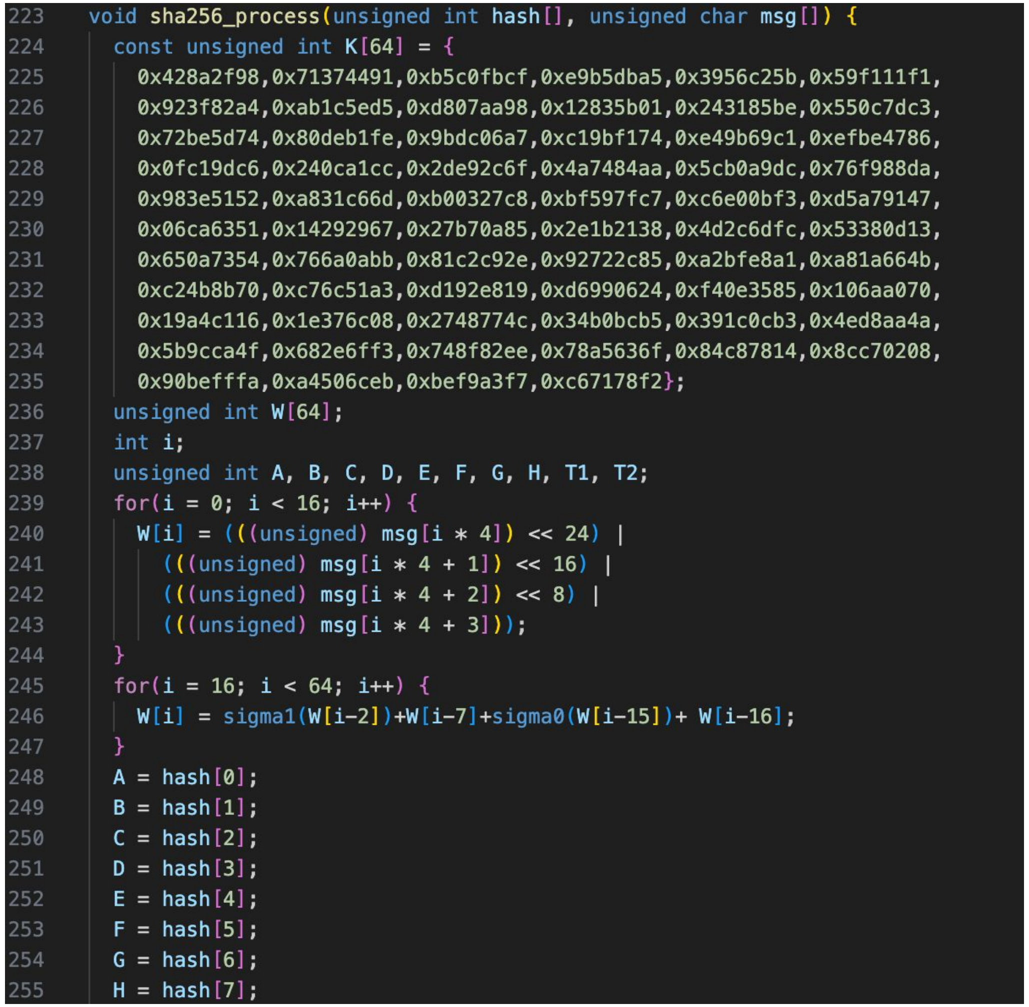
Source Code after changes -

void sha1_process

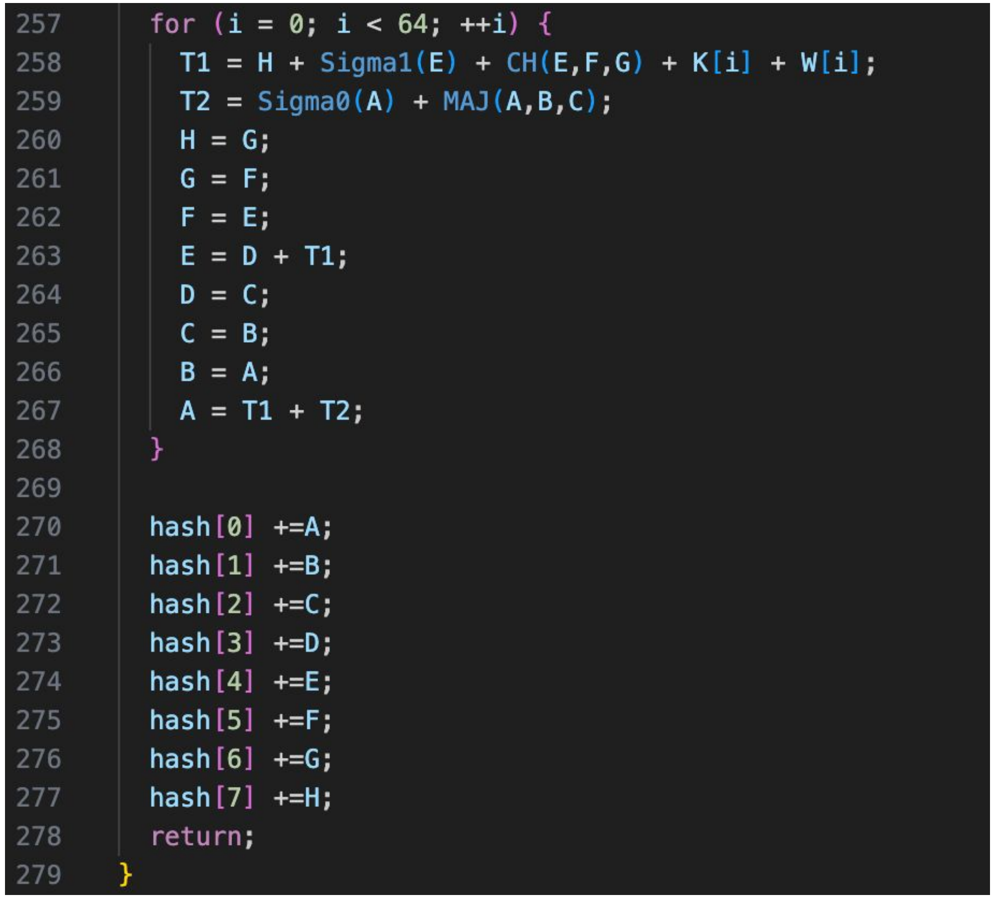
Line 197 to line 1070 has been added in the file shaM.c

Source Code before changes -

void sha256_process



```c
void sha256_process(unsigned int hash[], unsigned char msg[]) {
  const unsigned int K[64] = {
    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};
  unsigned int W[64];
  int i;
  unsigned int A, B, C, D, E, F, G, H, T1, T2;
  for(i = 0; i < 16; i++) {
    W[i] = (((unsigned) msg[i * 4]) << 24) |
      (((unsigned) msg[i * 4 + 1]) << 16) |
      (((unsigned) msg[i * 4 + 2]) << 8) |
      (((unsigned) msg[i * 4 + 3]));
  }
  for(i = 16; i < 64; i++) {
    W[i] = sigma1(W[i-2])+W[i-7]+sigma0(W[i-15])+ W[i-16];
  }
  A = hash[0];
  B = hash[1];
  C = hash[2];
  D = hash[3];
  E = hash[4];
  F = hash[5];
  G = hash[6];
  H = hash[7];
```



```c
  for (i = 0; i < 64; ++i) {
    T1 = H + Sigma1(E) + CH(E,F,G) + K[i] + W[i];
    T2 = Sigma0(A) + MAJ(A,B,C);
    H = G;
    G = F;
    F = E;
    E = D + T1;
    D = C;
    C = B;
    B = A;
    A = T1 + T2;
  }

  hash[0] +=A;
  hash[1] +=B;
  hash[2] +=C;
  hash[3] +=D;
  hash[4] +=E;
  hash[5] +=F;
  hash[6] +=G;
  hash[7] +=H;
  return;
}
```

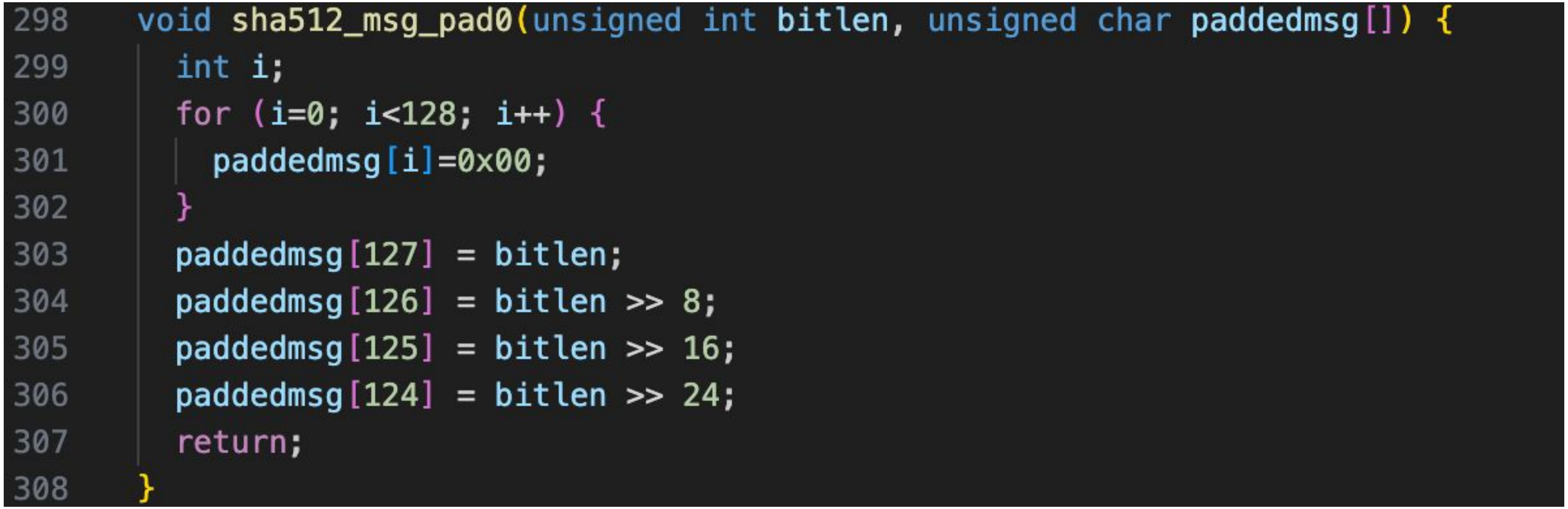
Source Code after changes -

void sha256_process

Line 1133 to line 1987 has been added in the file shaM.c

Source Code before changes -

sha512_msg_pad0



```c
void sha512_msg_pad0(unsigned int bitlen, unsigned char paddedmsg[]) {
  int i;
  for (i=0; i<128; i++) {
    paddedmsg[i]=0x00;
  }
  paddedmsg[127] = bitlen;
  paddedmsg[126] = bitlen >> 8;
  paddedmsg[125] = bitlen >> 16;
  paddedmsg[124] = bitlen >> 24;
  return;
}
```

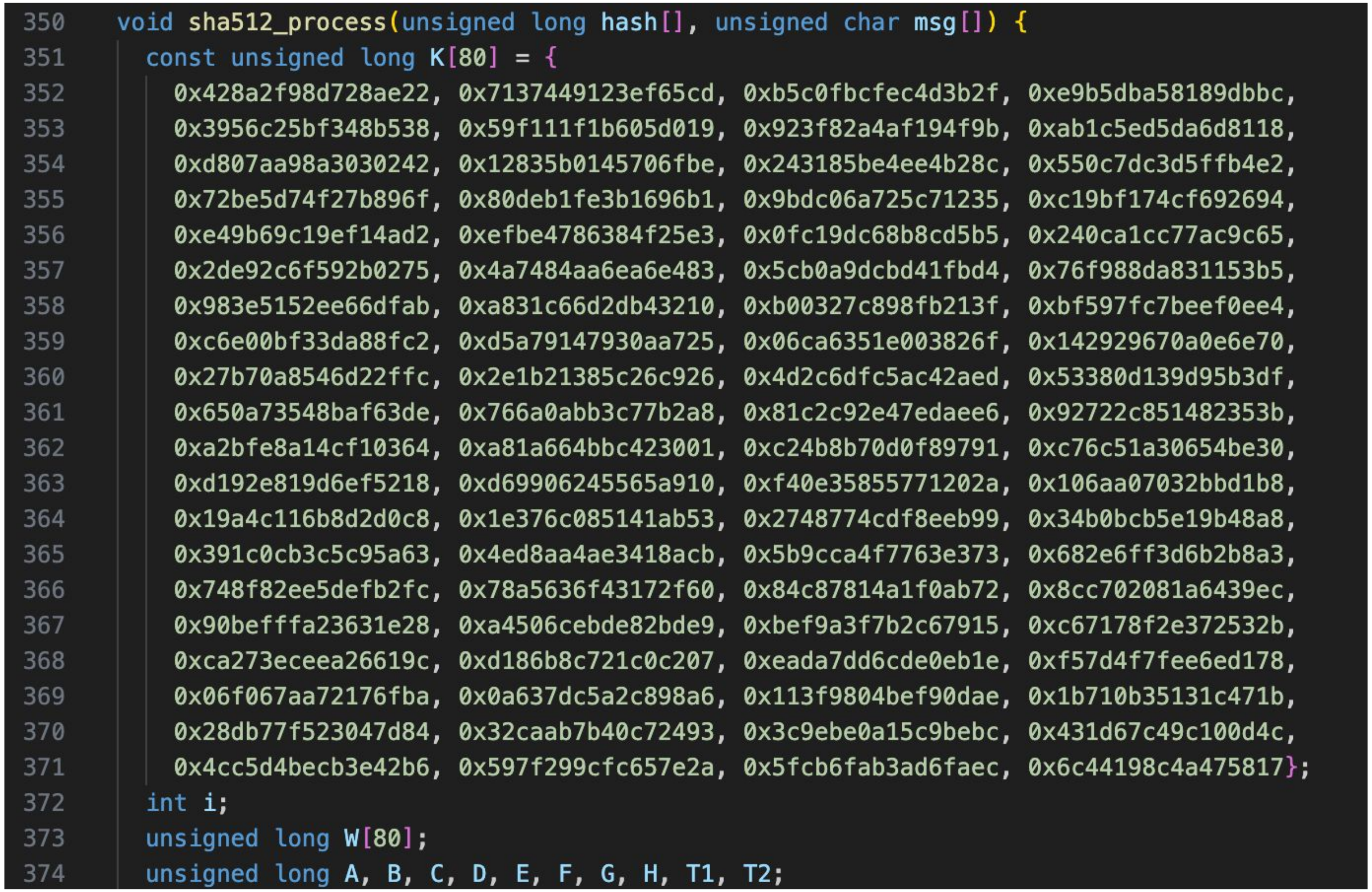
Source Code after changes -

sha512_msg_pad0

Line 2011 to line 2144 has been added in the file shaM.c

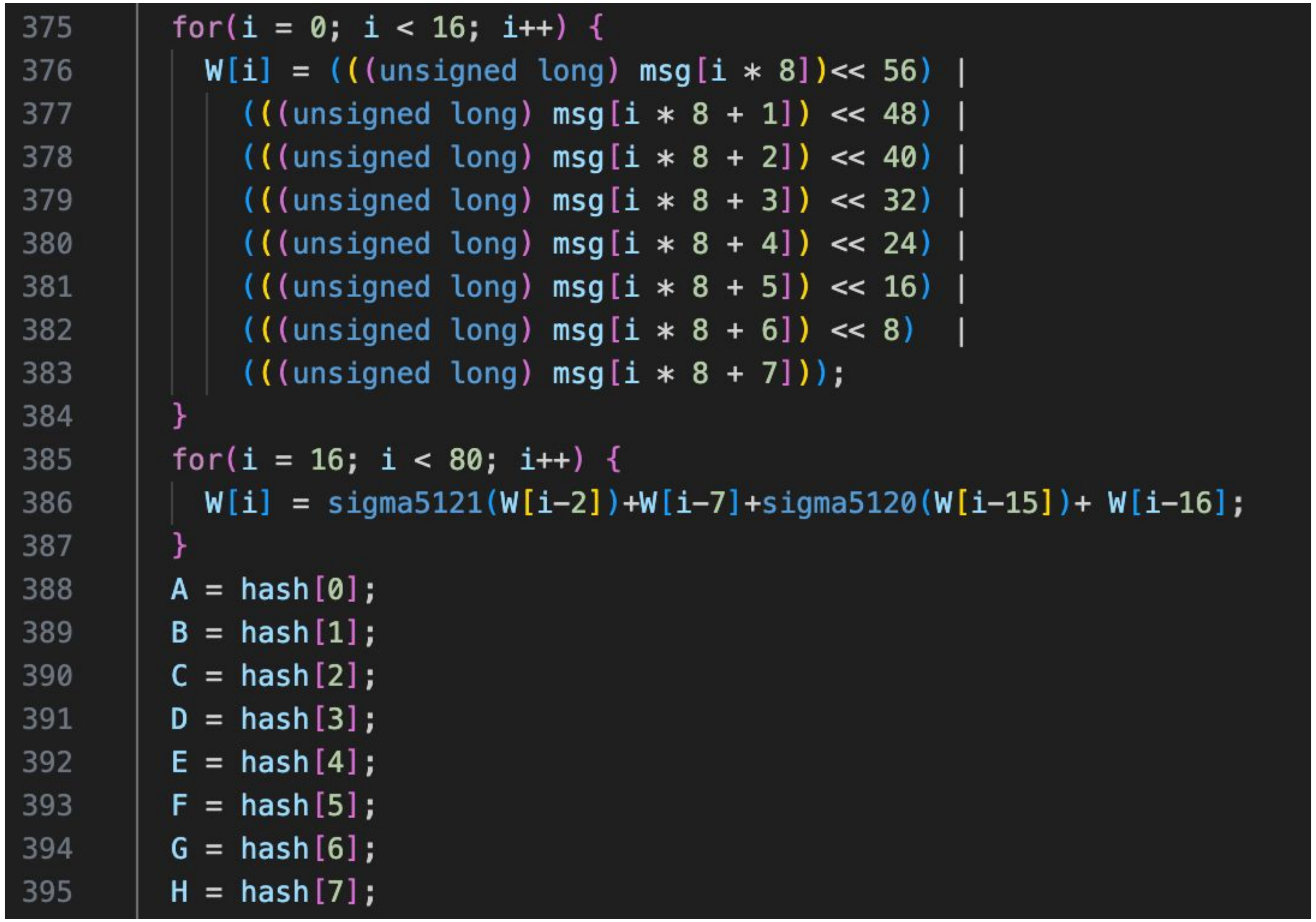
Source Code before changes -

void sha512_process

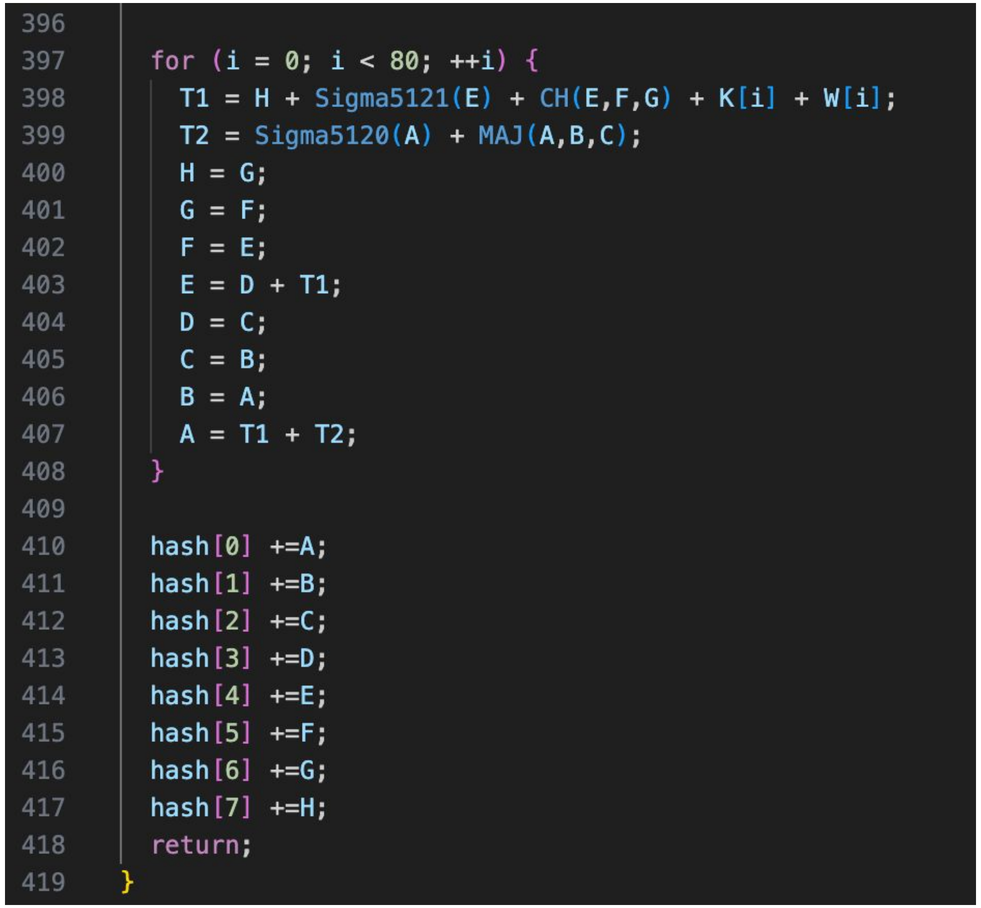


```c
void sha512_process(unsigned long hash[], unsigned char msg[]) {
  const unsigned long K[80] = {
    0x428a2f98d728ae22, 0x7137449123ef65cd, 0xb5c0fbcfec4d3b2f, 0xe9b5dba58189dbbc,
    0x3956c25bf348b538, 0x59f111f1b605d019, 0x923f82a4af194f9b, 0xab1c5ed5da6d8118,
    0xd807aa98a3030242, 0x12835b0145706fbe, 0x243185be4ee4b28c, 0x550c7dc3d5ffb4e2,
    0x72be5d74f27b896f, 0x80deb1fe3b1696b1, 0x9bdc06a725c71235, 0xc19bf174cf692694,
    0xe49b69c19ef14ad2, 0xefbe4786384f25e3, 0x0fc19dc68b8cd5b5, 0x240ca1cc77ac9c65,
    0x2de92c6f592b0275, 0x4a7484aa6ea6e483, 0x5cb0a9dcbd41fbd4, 0x76f988da831153b5,
    0x983e5152ee66dfab, 0xa831c66d2db43210, 0xb00327c898fb213f, 0xbf597fc7beef0ee4,
    0xc6e00bf33da88fc2, 0xd5a79147930aa725, 0x06ca6351e003826f, 0x142929670a0e6e70,
    0x27b70a8546d22ffc, 0x2e1b21385c26c926, 0x4d2c6dfc5ac42aed, 0x53380d139d95b3df,
    0x650a73548baf63de, 0x766a0abb3c77b2a8, 0x81c2c92e47edaee6, 0x92722c851482353b,
    0xa2bfe8a14cf10364, 0xa81a664bbc423001, 0xc24b8b70d0f89791, 0xc76c51a30654be30,
    0xd192e819d6ef5218, 0xd69906245565a910, 0xf40e35855771202a, 0x106aa07032bbd1b8,
    0x19a4c116b8d2d0c8, 0x1e376c085141ab53, 0x2748774cdf8eeb99, 0x34b0bcb5e19b48a8,
    0x391c0cb3c5c95a63, 0x4ed8aa4ae3418acb, 0x5b9cca4f7763e373, 0x682e6ff3d6b2b8a3,
    0x748f82ee5defb2fc, 0x78a5636f43172f60, 0x84c87814a1f0ab72, 0x8cc702081a6439ec,
    0x90befffa23631e28, 0xa4506cebde82bde9, 0xbef9a3f7b2c67915, 0xc67178f2e372532b,
    0xca273eceea26619c, 0xd186b8c721c0c207, 0xeada7dd6cde0eb1e, 0xf57d4f7fee6ed178,
    0x06f067aa72176fba, 0x0a637dc5a2c898a6, 0x113f9804bef90dae, 0x1b710b35131c471b,
    0x28db77f523047d84, 0x32caab7b40c72493, 0x3c9ebe0a15c9bebc, 0x431d67c49c100d4c,
    0x4cc5d4becb3e42b6, 0x597f299cfc657e2a, 0x5fcb6fab3ad6faec, 0x6c44198c4a475817};
  int i;
  unsigned long W[80];
  unsigned long A, B, C, D, E, F, G, H, T1, T2;
```

[redacted]



```
for(i = 0; i < 16; i++) {
    W[i] = (((unsigned long) msg[i * 8])<< 56) |
        (((unsigned long) msg[i * 8 + 1]) << 48) |
        (((unsigned long) msg[i * 8 + 2]) << 40) |
        (((unsigned long) msg[i * 8 + 3]) << 32) |
        (((unsigned long) msg[i * 8 + 4]) << 24) |
        (((unsigned long) msg[i * 8 + 5]) << 16) |
        (((unsigned long) msg[i * 8 + 6]) << 8)  |
        (((unsigned long) msg[i * 8 + 7]));
}
for(i = 16; i < 80; i++) {
    W[i] = sigma5121(W[i-2])+W[i-7]+sigma5120(W[i-15])+ W[i-16];
}
A = hash[0];
B = hash[1];
C = hash[2];
D = hash[3];
E = hash[4];
F = hash[5];
G = hash[6];
H = hash[7];
```



```

    for (i = 0; i < 80; ++i) {
        T1 = H + Sigma5121(E) + CH(E,F,G) + K[i] + W[i];
        T2 = Sigma5120(A) + MAJ(A,B,C);
        H = G;
        G = F;
        F = E;
        E = D + T1;
        D = C;
        C = B;
        B = A;
        A = T1 + T2;
    }

    hash[0] +=A;
    hash[1] +=B;
    hash[2] +=C;
    hash[3] +=D;
    hash[4] +=E;
    hash[5] +=F;
    hash[6] +=G;
    hash[7] +=H;
    return;
}
```

Source Code after changes -

void sha512_process

Line 2225 to line 3201 has been added in the file shaM.c