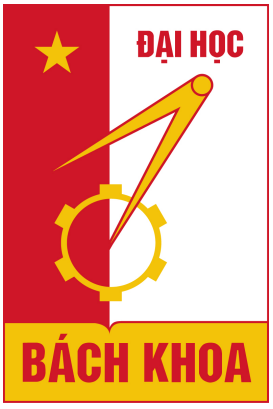
Trường đại học bách khoa hà nội



Báo cáo giữa kì môn học:

Vi xử lý

LẬP TRÌNH HÀM BĂM MD5 TRÊN 8086

Sinh viên thực hiện:
Tạ Quang Tùng
MSSV: 20154280

Giáo viên hướng dẫn:
TS. Ngô Lam Trung

Hà Nội, Ngày 13 tháng 6 năm 2018

Mục lục

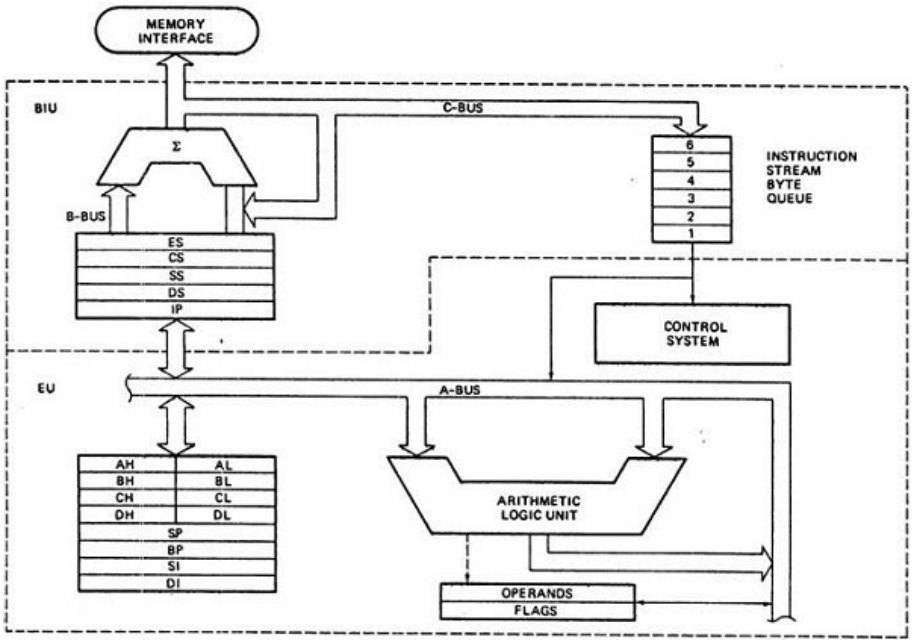
1 Giới thiệu về 8086

Vi xử lý 8086 là phiên bản nâng cấp của vi xử lý 8085 được phát triển từ năm 1976. Nó là vi xử lý 16 bit, có một tập lệnh mạnh mẽ và cung cấp phép toán nhân, chia trực tiếp trên phần cứng. Bao gồm 20 chân địa chỉ và 16 chân dữ liệu. Do đó nó có thể truy cập tới 1MB bộ nhớ.

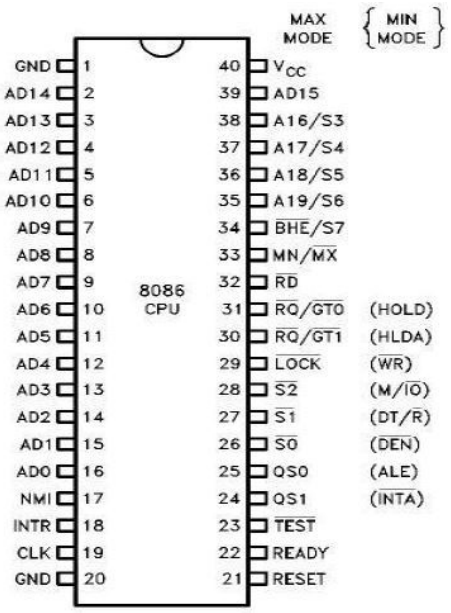
Nó có 2 chế độ hoạt động là chế độ Maximum và chế độ Minimum. Chế độ Maximum phù hợp với hệ thống có nhiều vi xử lý và ngược lại chế độ Minimum phù hợp với hệ thống chỉ có một vi xử lý. Ở đây ta chỉ quan tâm tới chế độ Minimum.

Những đặc điểm nổi bật của vi xử lý 8086:

- Nó có hàng đợi lệnh, cho phép lưu trữ tới 6 byte lệnh giúp tăng tốc xử lý.
- Là vi xử lý 16 bit đầu tiên có 16 bit ALU, thanh ghi 16 bit, bus nội 16 bit, và 16 bit bus ngoại.
- Nó có 2 giai được xử lý đường ống, Fetch và Execute giúp tăng hiệu năng.
- Bảng vector ngắt 256 phần tử.
- Được tạo thành bởi 29.000 transitor.



Hình 1: Kiến trúc của 8086

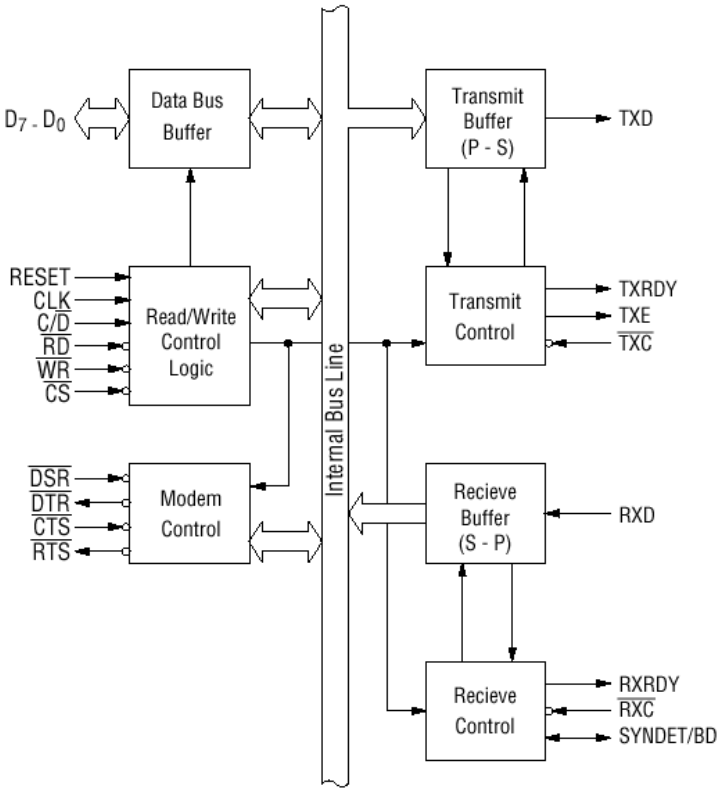


Hình 2: Các chân của 8086

Các chân của 8086 ở chế độ Min bao gồm:

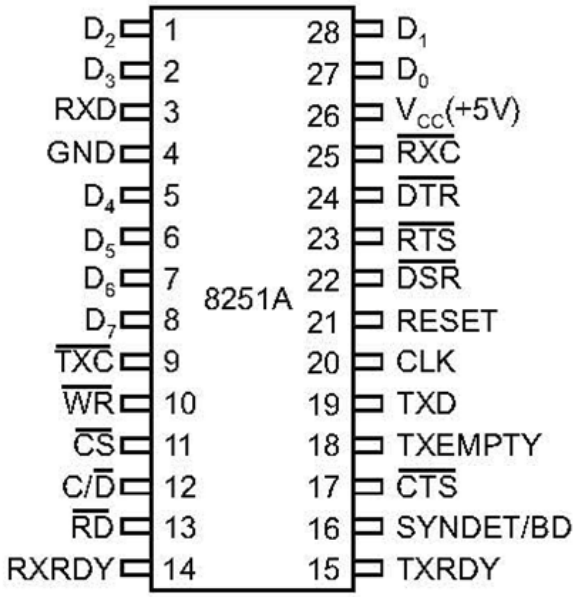
- Nguồn: Sử dụng ngồn 5V DC với chân nguồn tại chân 40, chân đất tại chân 1 và 20.
- Tín hiệu clock: chân 19.
- Địa chỉ / Dữ liệu: AD0-AD15.
- Địa chỉ / Trạng thái: A16-A19/S3-S6.
- Tín hiệu đọc: Chân 32.
- Tín hiệu sẵn sàng: Chân 22.
- Tín hiệu khởi động vi xử lý: Chân 21.
- Tín hiệu báo ngắt: Chân 18.
- Tín hiệu ngắt không che được: Chân 17.
- Tín hiệu kiểm tra: Chân 23. CPU đợi nếu tín hiệu này là cao.
- Tín hiệu phân biệt Minimum / Maximum: Chân 33.
- Tín hiệu chấp nhận ngắt: Chân 24.
- Tín hiệu chốt địa chỉ: Chân 25.
- Tín hiệu chỉ định hướng của dữ liệu: Chân 27.
- Tín hiệu chỉ định bộ nhớ / vào ra: Chân 28.
- Tín hiệu ghi: Chân 29.

2 Truyền nối tiếp sử dụng 8251 USART



Hình 3: Kiến trúc của 8251

2.1 Các chân của 8251

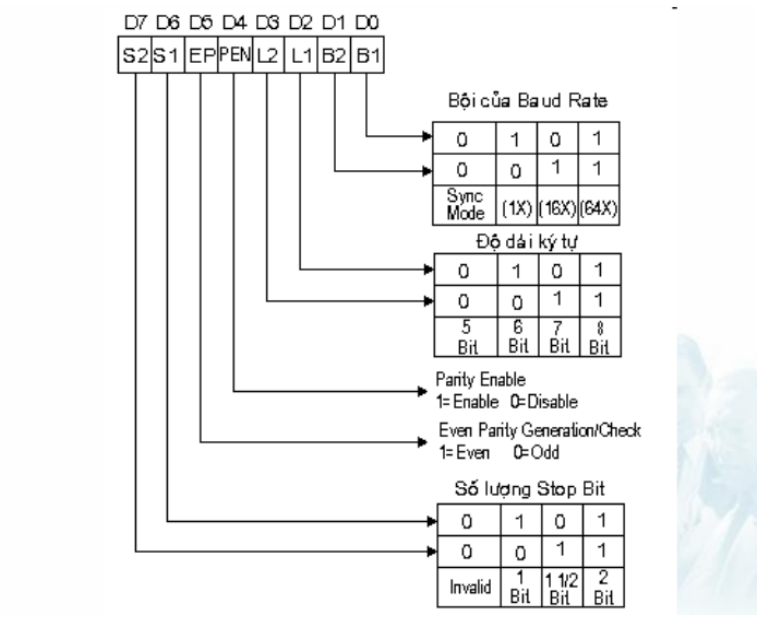


Hình 4: Các chân của 8251

- CLK: Chân nối tới xung đồng hồ của hệ thống.
- TxRDY: Tín hiệu báo bộ đệm giữ rỗng (sẵn sàng nhận dữ liệu mới từ CPU).
- RxRDY: Tín hiệu báo bộ đệm thu đầy (có kí tự nằm chờ CPU đọc vào).
- TxEMPTY: Báo cả đệm giữ và đệm thu đều rỗng.
- C/D: CPU chọn thanh ghi lệnh hay thanh ghi dữ liệu.
- RxC và TxC: Xung đồng hồ cung cấp cho thanh ghi dịch của phần thu và phát.
- RxD và TxD: Nhận và gửi tín hiệu nối tiếp.
- D0-D7: Nối với bus dữ liệu của hệ thống.

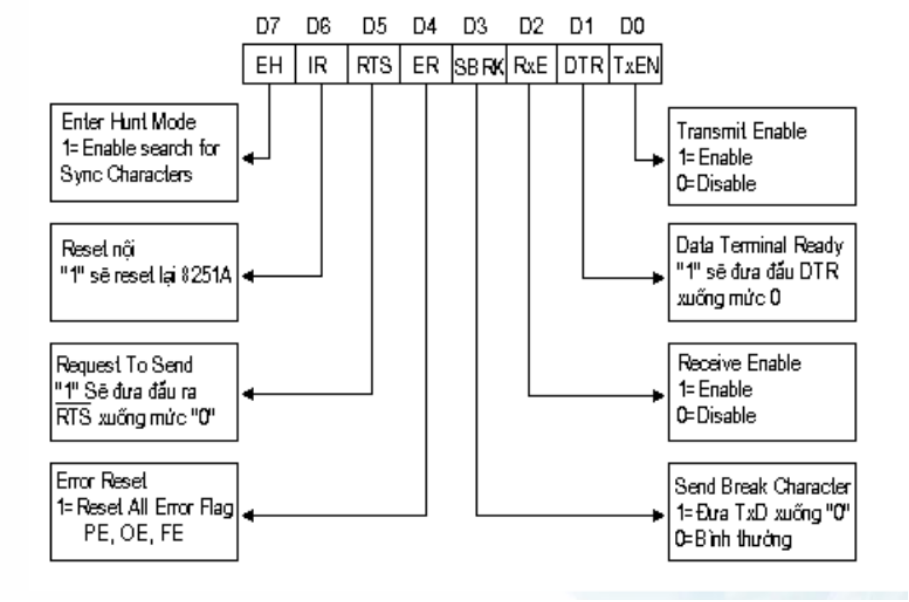
2.2 Chế độ được sử dụng

Trong bài này ta sử dụng 2 giá trị của thanh ghi chế độ và thanh ghi lệnh lần lượt là 01001101b và 00000111b. Tương ứng:



Thanh ghi chế độ:

- Bội của Baurd Rate: 1x.
- Độ dài kí tự: 8 bit.
- Parity: Disable.
- Even Parity Generation/Check: Odd.
- Số lượng Stop Bit: 1 bit.



Thanh ghi lệnh:

- Transmit Enable: Enable.
- Data Terminal Ready: 1.
- Receive Enable: Enable.
- Send Break Character: Bình thường.
- Error Restet: Disable.
- Request To Send: 0.
- Reset nội: 0.
- Enter Hunt Mode: Disable.

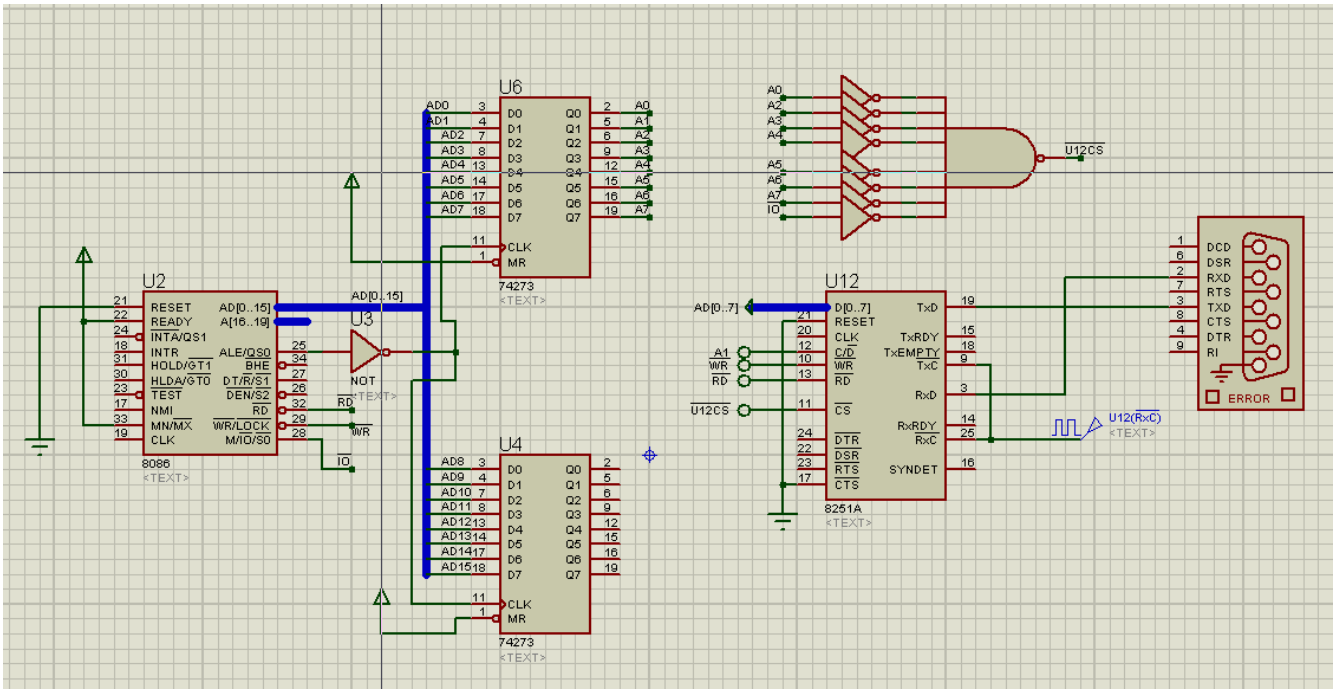
2.3 Các bước ghi dữ liệu

1. Đọc 1 byte từ thanh ghi trạng thái vào AL.
2. And giá trị nhận được với 0x1.
3. Nếu giá trị thu được bằng 0, quay lại bước 1.
4. Gán giá trị cần gửi vào AL và ghi ra thanh ghi đệm phát của 8251.

2.4 Các bước đọc dữ liệu

1. Đọc 1 byte từ thanh ghi trạng thái vào AL.
2. And giá trị nhận được với 0x2.
3. Nếu giá trị thu được bằng 0, quay lại bước 1.
4. Đọc giá trị từ thanh ghi đệm phát của 8251 vào AL.

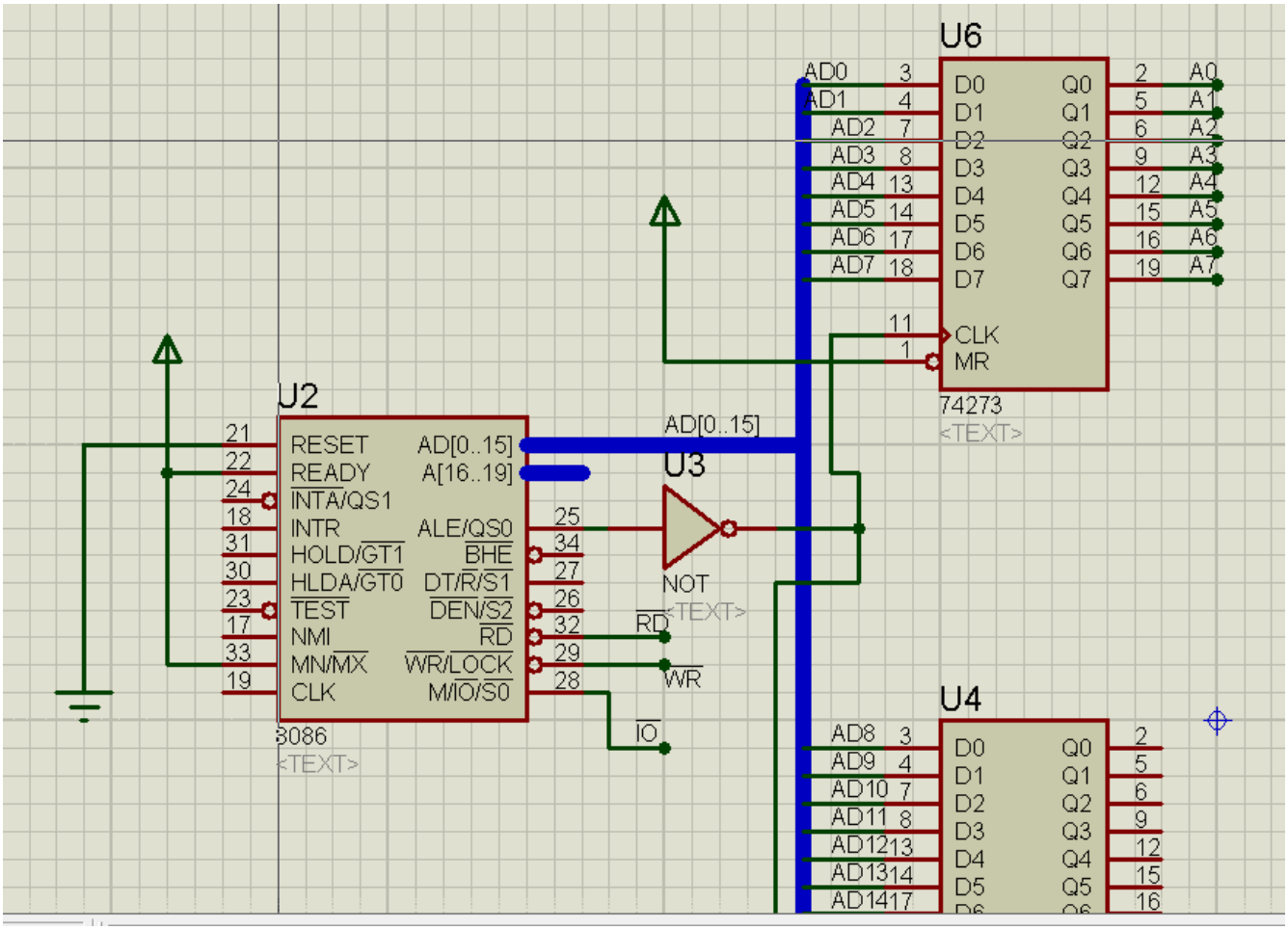
3 Sơ đồ mắc nối mạch trên Proteus



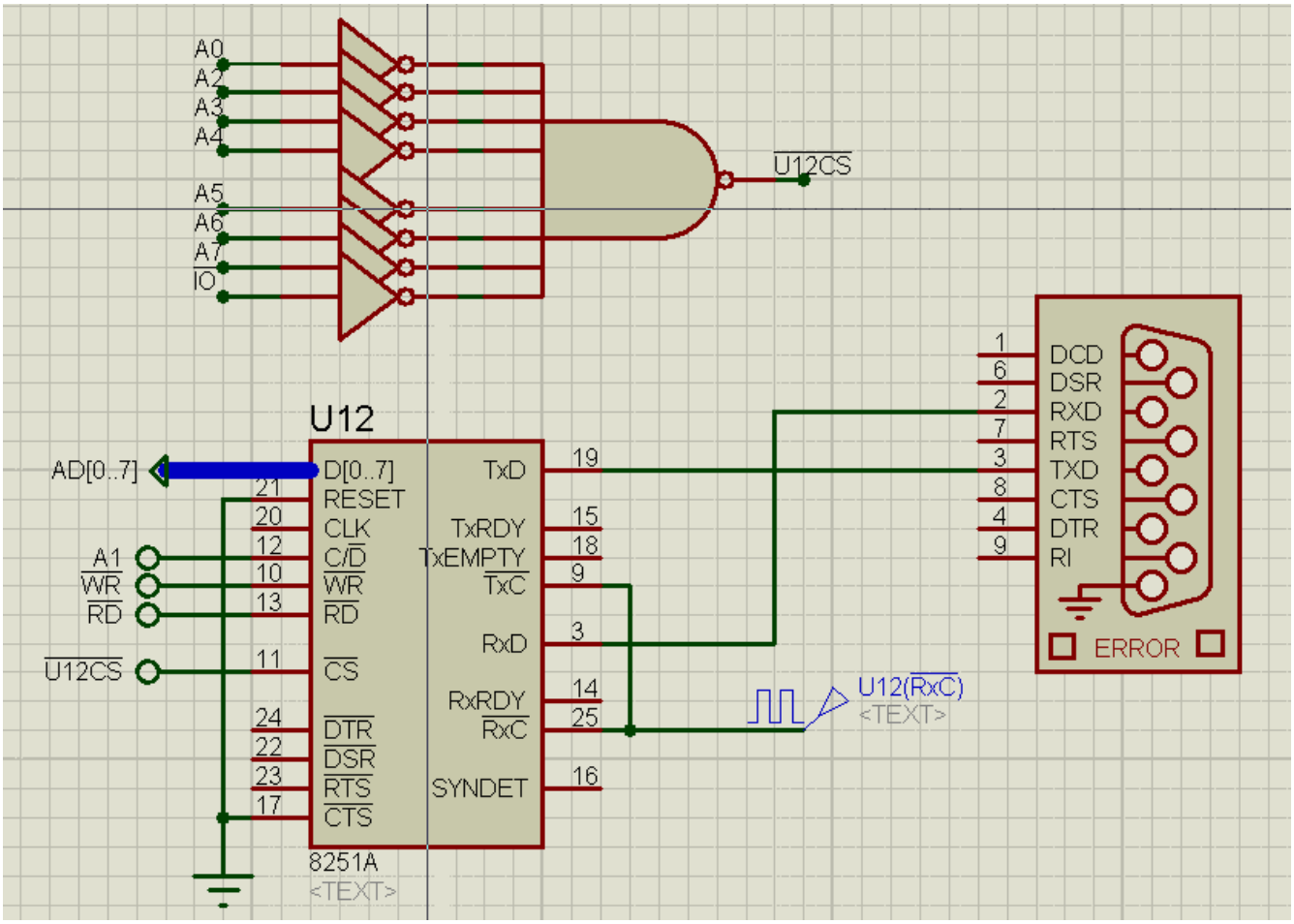
Hình 5: Sơ đồ mắc nối mạch (tổng thể).

Mạch bao gồm:

- Một CPU 8086.
- Hai bộ chốt địa chỉ 74273.
- Một chip 8251.
- Một cổng Compim.
- Một mạch giải mã địa chỉ.



Hình 6: Sơ đồ mắc nối mạch (trái).



Hình 7: Sơ đồ mắc nối mạch (phải).

- Mạch giải mã địa chỉ cho phép 2 địa chỉ là 0x00 và 0x02.

- Xung đồng hồ được sử dụng có tần số 19200Hz

4 Hàm băm MD5

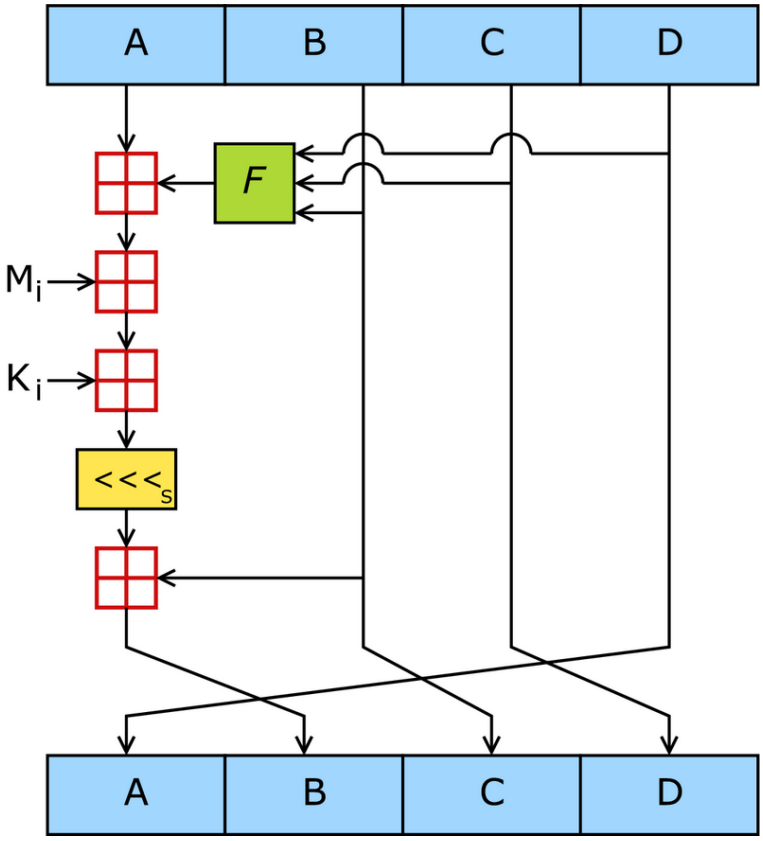
4.1 Giới thiệu về hàm băm mật mã MD5

MD5 là một hàm băm mật mã học được sử dụng phổ biến với giá trị băm dàm 128 bit. Là một chuẩn Internet (Được mô tả trong RFC 1321). MD5 đã được dùng nhiều trong ứng dụng bảo mật, và được sử dụng để kiểm tra tính toàn vẹn của tệp tin.

MD5 chuyển một đoạn thông tin chiều dài bất kì thành một kết quả có chiều dài không đổi 128 bit. MD5 chia tin đầu vào thành từng đoạn 512 bit, được padding thêm để chiều dài nó chia chẵn cho 512. Bit 1 được gắn vào cuối mẩu tin đầu vào, sau đó là một dãy số 0, và cuối cùng là một giá trị nguyên không dấu 64 bit theo kiểu little endian.

Giải thuật MD5 hoạt động trên trạng thái 128 bit, được chia thành 4 từ 32 bit, kí hiệu là A, B, C, D. Chúng được khởi tạo bằng những hằng số cố định. Từng gói tin 512 bit được xử lý thông

qua bốn giai đoạn, mỗi gian đoạn lại bao gồm 16 tác vụ với hàm phi tuyến F giống nhau.



Hình 8: Một tác vụ trong 64 tác vụ của MD5, trong đó F là một hàm phi tuyến (có 4 hàm phi tuyến), ô vuông là phép cộng modulo 2^{32}, M_j là khối tin đầu vào 32 bit, K_i là hằng số 32 bit, mỗi tác vụ có một i j khác nhau, $\lll$ là phép xoay trái.

4.2 Mã C++ của thuật toán MD5

```
#include <iostream>
#include <string>

using word = unsigned int;
using byte = unsigned char;

#define MESSAGE_SIZE 64
#define WORD_COUNT (MESSAGE_SIZE / 4)

byte input[MESSAGE_SIZE];

byte output[WORD_COUNT];

word T[] = {
    0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdceee,
    0xf57c0faf, 0x4787c62a, 0xa8304613, 0xfd469501,
```

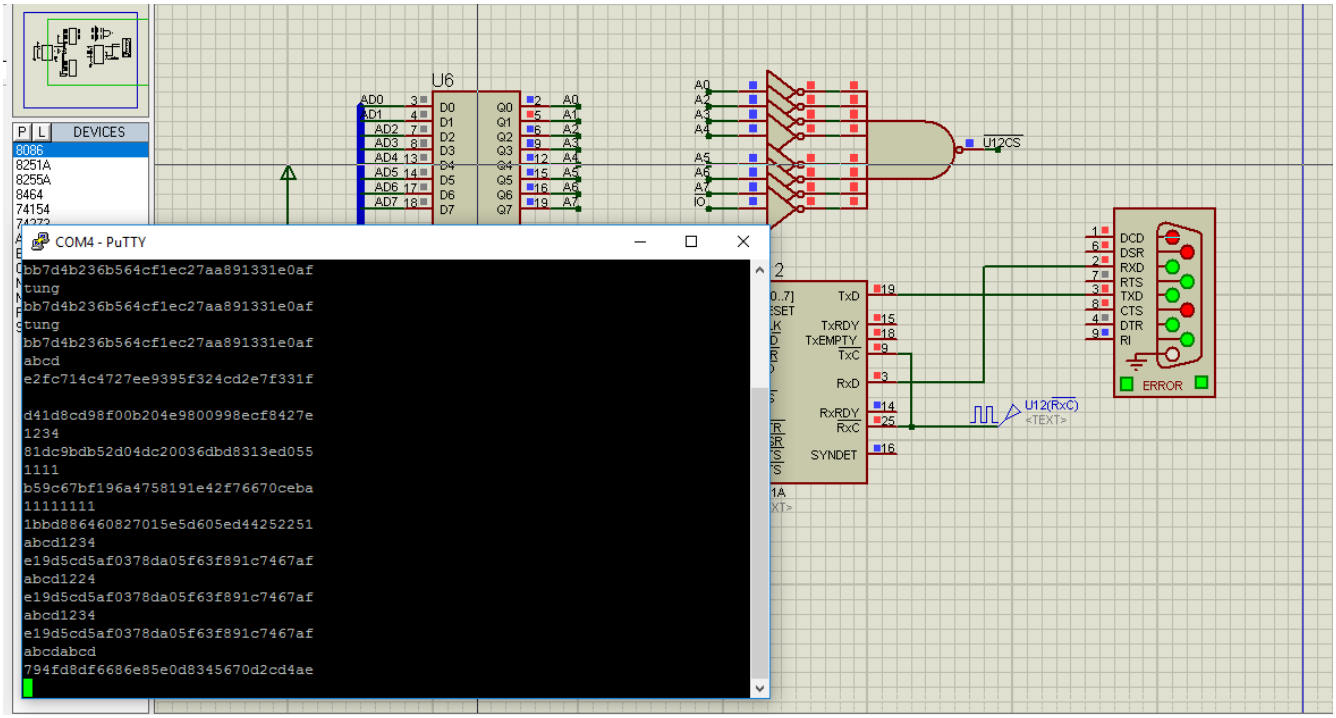
```
    0x698098d8, 0x8b44f7af, 0xffff5bb1, 0x895cd7be,
    0x6b901122, 0xfd987193, 0xa679438e, 0x49b40821,
    0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa,
    0xd62f105d, 0x02441453, 0xd8a1e681, 0xe7d3fbc8,
    0x21e1cde6, 0xc33707d6, 0xf4d50d87, 0x455a14ed,
    0xa9e3e905, 0xfcefa3f8, 0x676f02d9, 0x8d2a4c8a,
    0xfffa3942, 0x8771f681, 0x6d9d6122, 0xfde5380c,
    0xa4beea44, 0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70,
    0x289b7ec6, 0xeaa127fa, 0xd4ef3085, 0x04881d05,
    0xd9d4d039, 0xe6db99e5, 0x1fa27cf8, 0xc4ac5665,
    0xf4292244, 0x432aff97, 0xab9423a7, 0xfc93a039,
    0x655b59c3, 0x8f0ccc92, 0xffeff47d, 0x85845dd1,
    0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1,
    0xf7537e82, 0xbd3af235, 0x2ad7d2bb, 0xeb86d391,
};

void init_input(const std::string& s) {
    size_t i = 0;
    size_t input_size = s.size();

    for (; i < input_size; i++)
        input[i] = s[i];
    for (; i < MESSAGE_SIZE; i++)
        input[i] = 0;

    input[input_size] = 128;
    word *len = (word *)&input[MESSAGE_SIZE - 8];
    *len = input_size * 8;
}

#define ROTATE_LEFT(x, n) \
    ((x) << (n) | (x) >> (32 - (n)))

void md5() {
    word A = 0x67452301;
    word B = 0xefcdab89;
    word C = 0x98badcfe;
    word D = 0x10325476;

    word *X = (word *)input;

    word AA = A;
    word BB = B;
    word CC = C;
    word DD = D;

    const word s[] = {
```

```
        7, 12, 17, 22,
        5,  9, 14, 20,
        4, 11, 16, 23,
        6, 10, 15, 21,
    };

    for (int i = 0; i < 64; i++) {
        word F, g;
        if (i < 16) {
            F = (B & C) | (~B & D);
            g = i % 16;
        }
        else if (i < 32) {
            F = (D & B) | (~D & C);
            g = (5*i + 1) % 16;
        }
        else if (i < 48) {
            F = B ^ C ^ D;
            g = (3*i + 5) % 16;
        }
        else {
            F = C ^ (B | ~D);
            g = (7*i) % 16;
        }

        F = F + A + X[g] + T[i];
        A = D;
        D = C;
        C = B;
        B += ROTATE_LEFT(F, s[(i / 16) * 4 + i % 4]);
    }
    A += AA;
    B += BB;
    C += CC;
    D += DD;

    ((word *)output)[0] = A;
    ((word *)output)[1] = B;
    ((word *)output)[2] = C;
    ((word *)output)[3] = D;
}

void print_output() {
    for (byte b: output)
        std::cout << std::hex << (word)b << std::endl;
}
```

```
int main() {
    init_input("abcd");
    md5();
    print_output();

    return 0;
}
```

Code thực hiện trên assembly dựa theo code C++ này.

5 Kết quả



Hình 9: Kết quả chạy chương trình trên Proteus