



## File BT Kiến trúc máy tính

kiến trúc máy tính (Học viện Công nghệ Bưu chính Viễn thông)



Scan to open on Studocu

# Tổng hợp bài tập KTMT

## Mục lục

<i>Phần 1: Phân tích một đoạn code</i>	<i>2</i>
<i>Phần 2: Bài tập bộ nhớ Cache</i>	<i>6</i>
<i>Phần 3: Lập trình hợp ngữ</i>	<i>8</i>
<i>Phần 4: Giải mã địa chỉ bộ nhớ</i>	<i>36</i>

## **Phần 1: Phân tích một đoạn code (chế độ địa chỉ, ý nghĩa, xác định giá trị thanh ghi, pipeline )**

- Chế độ địa chỉ: tức thì, trực tiếp, gián tiếp qua thanh ghi, gián tiếp qua ô nhớ, địa chỉ chỉ số, địa chỉ tương đối
- Ý nghĩa của đoạn mã: dựa vào nội dung của từng dòng lệnh và mối liên hệ giữa các dòng lệnh
- Xác định giá trị thanh ghi: thực hiện các phép tính toán dựa vào nội dung câu lệnh
- Pipeline: hiểu 5 giai đoạn thực hiện lệnh: Đọc lệnh (IF), giải mã & đọc toán hạng (ID), thực hiện (EX), truy nhập bộ nhớ (MEM), và lưu kết quả (WB)

**Bài 1 :** Cho đoạn chương trình sau (R1, R2 là các thanh ghi và lệnh quy ước theo dạng LỆNH <ĐÍCH> <GỐC>):

- (1) STORE -100(R2), R1
- (2) LOAD R1, (00FF)
- (3) COMPARE R3, R4
- (4) JUMP-IF-EQUAL Label
- (5) ADD R3, R4
- (6) ADD R2, 2
- (7) Label:

1. Xác định chế độ địa chỉ và ý nghĩa của từng lệnh;
2. Nêu hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình trên biết mỗi lệnh được chia thành 5 giai đoạn.
3. Giả thiết  $R3 \neq R4$  và mỗi giai đoạn thực hiện lệnh đều thực hiện trong thời gian là 0.1ns, so sánh thời gian CPU chạy hết 6 lệnh đầu tiên trong trường hợp không sử dụng cơ chế pipeline và có sử dụng cơ chế pipeline trong ý 2.

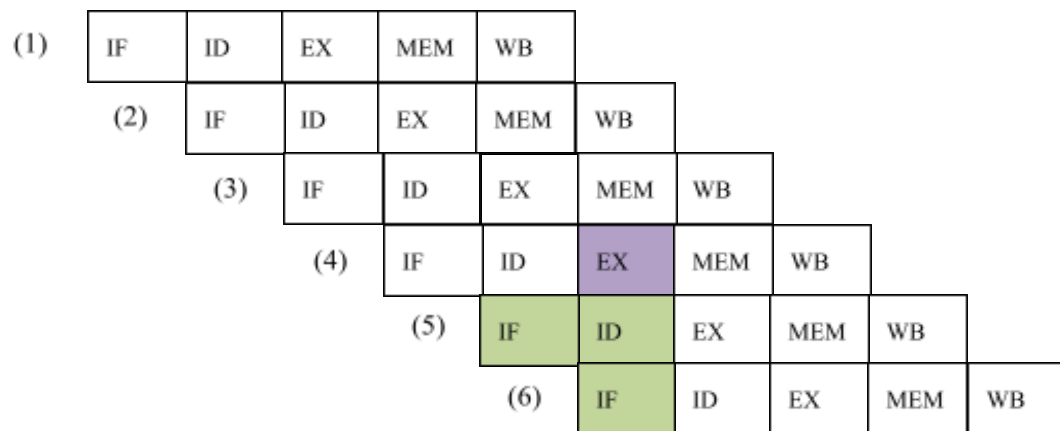
### **Giải**

1. Chế độ địa chỉ và ý nghĩa của từng lệnh
  - (1) STORE -100(R2), R1 :
    - Chế độ địa chỉ chỉ số do toán hạng đích tạo bởi phép cộng giữa 1 hằng số và thanh ghi  $(-100(R2) \square M[R2 - 100])$
    - Ý nghĩa câu lệnh: Gán trực tiếp giá trị của R1 vào ô nhớ  $M[R2-100]$
  - (2) LOAD R1, (00FF) :
    - Chế độ địa chỉ gián tiếp qua ô nhớ do toán hạng nguồn có địa chỉ lưu trong ô nhớ 00FF
    - Ý nghĩa câu lệnh: Gán giá trị trong ô nhớ có địa chỉ lưu ở ô nhớ 00FF vào thanh ghi R1
  - (3) COMPARE R3, R4
    - Chế độ địa chỉ trực tiếp do 2 toán hạng là 2 thanh ghi
    - Ý nghĩa câu lệnh: So sánh R3 và R4.

- (4) JUMP-IF-EQUAL Label  
 - Ý nghĩa câu lệnh: Nếu  $R3 == R4$  thì nhảy đến nhãn Label
- (5) ADD R3, R4  
 - Chế độ địa chỉ trực tiếp do 2 toán hạng là 2 thanh ghi (R3, R4)  
 - Ý nghĩa câu lệnh: Gán giá trị thanh  $R3 = R3 + R4$
- (6) ADD R2, 2  
 - Chế độ địa chỉ trực tiếp do có toán hạng đích là thanh ghi (R2) và toán hạng nguồn là địa chỉ ô nhớ (M[2])  
 - Ý nghĩa câu lệnh: Gán giá trị thanh  $R2 = R2 + M[2]$

## 2. Hướng giải quyết xung đột dữ liệu:

- Branch Hazard ở (4) (liên quan đến lệnh nhảy)
- Cách khắc phục: chèn (1) và (2) dưới câu lệnh (4).



## 3. Thời gian thực hiện 6 sử dụng pipeline

$$5 * n * t = 5 * 6 * 0.1 = 3 \text{ ns}$$

Thời gian thực hiện 6 câu lệnh khi sử dụng pipeline

$$(n+4) * t = (6+4) * 0.1 = 1 \text{ ns}$$

câu lệnh khi không

dùng pipeline

## Bài 2: Cho đoạn chương trình hợp ngữ sau

```

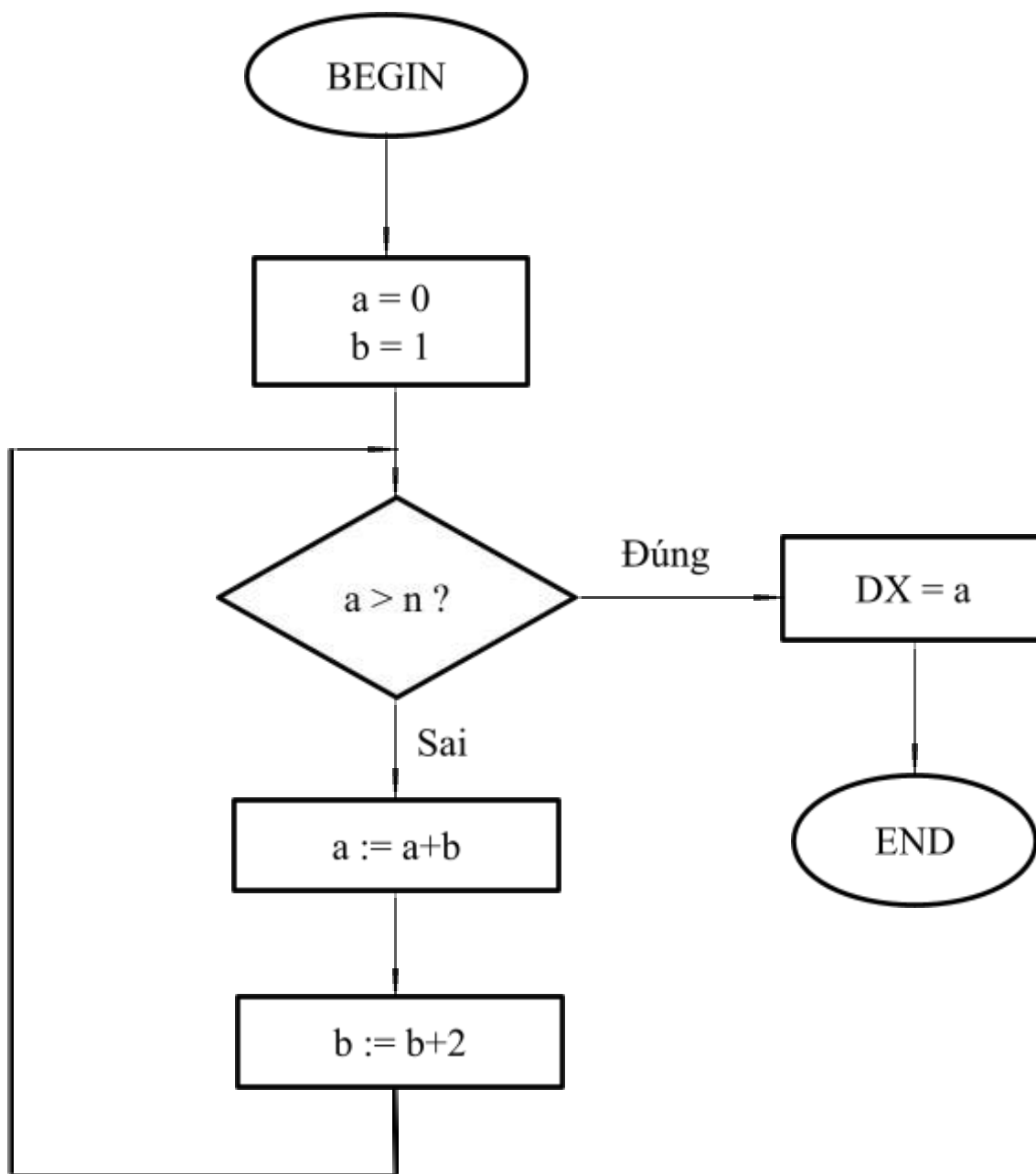
BEGIN:    ADD AX, 0
          ADD BX, 1
LOOP:    CMP AX, CX
          JG FINISH
          ADD AX, BX
          ADD BX, 2
          JMP LOOP
FINISH:   ADD DX, AX
  
```

Giả thiết thanh ghi CX chứa biến số nguyên n và thanh ghi DX để lưu kết quả, thanh ghi AX và BX lưu các biến a, b của chương trình.

1. Vẽ lưu đồ và giải thích ý nghĩa đoạn mã hợp ngữ trên theo các biến a, b, n
2. Tính tổng số lệnh và giá trị thanh ghi DX dưới dạng mã Hexa khi chạy đoạn hợp ngữ trên biết  $CX = 0005H$

## Giải

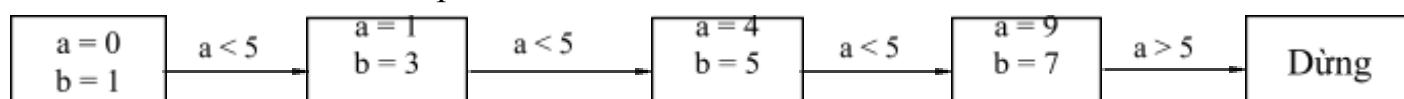
### 1. Vẽ lưu đồ



Ý nghĩa đoạn mã hợp ngữ: Tính tổng các số nguyên dương lẻ ( 1,3,5,...) sao cho tổng của chúng nhỏ nhất và tổng phải lớn hơn n. (hay tìm số chính phương nhỏ nhất và lớn hơn n).

### 2. CX = 0005H = 5<sub>10</sub>

\* Tính số lần lặp



Tổng số lệnh đã thực hiện:

$$2 + 3 \cdot 5 + 2 + 1 = 20 \text{ (lệnh)}$$

ADD AX, 0	3 vòng	CMP AX, CX	ADD DX, AX
ADD BX, 1	lặp	JG FINISH	

=

Giá trị thanh ghi DX = a  
9<sub>10</sub> = 0009H

**Bài 3:** Cho đoạn chương trình sau (R1, R2 là các thanh ghi và lệnh quy

ước theo dạng LỆNH <ĐÍCH> <GỐC>):

- (1) LOAD R2, #400
- (2) LOAD R1, #1200
- (3) STORE (R1), R2
- (4) SUBTRACT R2, #20
- (5) ADD 1200, #10
- (6) ADD R2, (R1)

1. Nêu ý nghĩa của từng lệnh;
2. Xác định giá trị của thanh ghi R2 sau khi thực hiện xong lệnh số (6)
3. Nêu một hướng giải quyết xung đột dữ liệu trong pipeline khi thực hiện đoạn chương trình trên biết rằng mỗi lệnh được chia thành 5 giai đoạn trong pipeline: Đọc lệnh (IF), giải mã & đọc toán hạng (ID), thực hiện (EX), truy nhập bộ nhớ (MEM), và lưu kết quả (WB).

### Giải

1. Ý nghĩa các câu lệnh

LOAD R2, #400 : gán giá trị thanh R2 = 400

LOAD R1, #1200 : gán giá trị thanh R1 = 1200

STORE (R1), R2 : gán trực tiếp giá trị ô nhớ M[R1] = R2  
(M[1200] = 400)

SUBTRACT R2, #20 : gán giá trị thanh R2 := R2 - 20  
( R2 = 400 - 20 = 380 )

ADD 1200, #10 : cộng 10 vào ô nhớ M[1200] rồi lưu vào M[1200]  
(M[1200] = 400 + 10 = 410)

ADD R2, (R1) : cộng giá trị ô nhớ M[1200] vào R2 rồi lưu vào R2  
( R2 = 380 + M[1200] = 380 + 410 = 790 )

2. Giá trị thanh R2 sau khi thực hiện xong lệnh số (6) R2 = 790

3. Giải quyết xung đột dữ liệu:

- Xung đột dữ liệu giữa (1) và (3) ; giữa (2) và (3) ; giữa (3) và (5) ; giữa (5) và (6).
- Một hướng giải quyết xung đột dữ liệu trong pipeline:

- + Chèn 3 NO-OP giữa (2) và (3)
- + Chèn 2 NO-OP giữa (4) và (5)
- + Chèn 3 NO-OP giữa (5) và (6)

## **Phần 2: Bài tập bộ nhớ Cache**

(by Hữu Sơn)

**1.** Tính tổng số bit cần thiết cho một bộ nhớ cache tổ chức theo ánh xạ trực tiếp kích thước là 64 KByte dữ liệu và một dòng cache có kích thước là 1 từ nhớ, giả sử bus địa chỉ 32 bit?

Giải:

- + Ta có 1 dòng cache 32bit = 4 byte =  $2^2$  byte  $\Rightarrow$  word = 2 bit
- + Số dòng Cache là  $L = \frac{M_{Cache}}{[Word]} = \frac{64 \cdot 2^{10} \text{ byte}}{4 \text{ byte}} = \frac{2^{16} B}{2^2 B} = 2^{14} \Rightarrow \text{line} = 14\text{bit}$
- + tag = address\_size - line - word = 32-14-2 = 16bit
- + Số lượng bit cần thiết cho một dòng Cache là:  
bits/block = data bits + tag bits + valid bit = 32 + 16 + 1 = 49 bit
- + Tổng số bit cần thiết cho bộ nhớ Cache là = L x bits/block =  $2^{14} \times 49 \text{ bit} = 98 \text{ Kbytes}$

**2.** Tổng số bit sẽ cần là bao nhiêu cho bộ nhớ cache sử dụng tập kết hợp được thiết lập 4 way kích thước là 64 KByte dữ liệu và một dòng cache có kích thước là 1 từ nhớ, giả sử bus địa chỉ 32 bit?

Giải:

- + Ta có 1 dòng cache 32bit = 4 byte =  $2^2$  byte  $\Rightarrow$  word = 2 bit
- + Số dòng Cache là  
$$L = \frac{M_{Cache}}{[Word] \cdot Way} = \frac{64 \cdot 2^{10} \text{ byte}}{4 \cdot 4 \text{ byte}} = \frac{2^{16}}{2^4} = 2^{12} \Rightarrow \text{line} = 12\text{bit}$$
- + tag = address\_size - line - word = 32-12-2 = 18bit
- + Số lượng bit cần thiết cho một dòng Cache là: bits/block = data bits + tag bits + valid bit = 32 + 18 + 1 = 51bit
- + Tổng số bit cần thiết cho bộ nhớ Cache là = L x bits/block =  $2^{12} \cdot 2^2 \times 51 \text{ bit} = 2^{14} \times 51 \text{ bit} = 102 \text{ Kbytes}$

**3.** Tính tổng số bit cần thiết cho một bộ nhớ cache tổ chức theo ánh xạ trực tiếp kích thước là 64 KB và một dòng cache có kích thước là 8 từ nhớ, giả sử

bus địa chỉ 32 bit?

Giải:

- + Ta có 1 dòng cache  $8 \times 32 \text{ bit} = 32 \text{ byte} = 2^5 \text{ byte} \Rightarrow \text{word} = 5 \text{ bit}$
- + Số dòng Cache là  $L = \frac{M_{\text{Cache}}}{[\text{Word}]} = \frac{64 \times 2^{10} \text{ byte}}{32 \text{ byte}} = \frac{2^{16}}{2^5} = 2^{11} \Rightarrow \text{line} = 11 \text{ bit}$
- +  $\text{tag} = \text{address\_size} - \text{line} - \text{word} = 32 - 11 - 5 = 16 \text{ bit}$
- + Số lượng bit cần thiết cho một dòng Cache là:  $\text{bits/block} = \text{data bits} + \text{tag bits} + \text{valid bit} = 8 \times 32 + 16 + 1 = 273 \text{ bit}$
- + Tổng số bit cần thiết cho bộ nhớ Cache là  $= L \times \text{bits/block} = 2^{11} \times 273 \text{ bit} = 68.25 \text{ Kbytes}$

4. Cho máy có dung lượng bộ nhớ chính: 64KB, dòng cache 8 Bytes, bộ nhớ cache được gồm 32 dòng (lines) được tổ chức ánh xạ trực tiếp.

1. Xác định số bit các thành phần địa chỉ của ô nhớ.
2. Ô nhớ có địa chỉ 0D20H trong bộ nhớ chính được nạp vào dòng (lines) nào của cache

Giải

1.  $M_{\text{Memory}} = 64 \text{ KB} = 2^6 * 2^{10} \text{ B} = 2^{16} \text{ B}$ 
  - +  $\text{Tag} + \text{line} + \text{word} = \log_2 64 * 2^{10} = \log_2 2^{16} = 16 \text{ bit}$
  - + Ta có 1 dòng Cache  $= 8 \text{ B} = 2^3 \text{ B} \Rightarrow \text{word} = 3 \text{ bit}$
  - + Số dòng Cache  $= 32 \text{ dòng} = 2^5 \Rightarrow \text{line} = 5 \text{ bit}$
  - +  $\text{tag} = \text{address\_size} - \text{line} - \text{word} = 16 - 5 - 3 = 8 \text{ bit}$

2. Ô nhớ 0D20H

0	D	2	0
0 0 0 0	1 1 0 1	0 0 1 0	0 0 0 0
tag		line	word
13		4	0

$\Rightarrow$  ô nhớ 0D20H nằm ở trang 13, dòng 4, ô 0



### **Phần 3: Lập trình hợp ngữ**

*(by Tuấn Đạt)*

#### **Một số bài tập tham khảo**

##### **I. Lập trình chương trình hợp ngữ Assembly**

- 1.** Viết chương trình hợp ngữ in ra lời chào Tiếng Anh và Tiếng Việt.
- 2.** Viết chương trình hợp ngữ Assembly cho phép nhập 1 ký tự và in ra màn hình ký tự đó.
- 3.** Viết chương trình hợp ngữ Assembly cho phép nhập 1 chuỗi ký tự và in ra màn hình chuỗi ký tự đó.
- 4.** Viết chương trình hợp ngữ Assembly cho phép nhập 1 ký tự viết thường và in ra màn hình chữ hoa của ký tự đó.
- 5.** Viết chương trình hợp ngữ Assembly cho phép nhập 1 chuỗi ký tự, in ra màn hình chuỗi ký tự đó theo dạng viết hoa và viết thường.
- 6.** Viết chương trình hợp ngữ Assembly cho phép nhập một chuỗi các ký tự kết thúc bởi “#” và yêu cầu in ra màn hình chuỗi ký tự đó theo thứ tự ngược lại.
- 7.** Viết chương trình hợp ngữ Assembly chuyển một số từ hệ cơ số 10 sang hệ nhị phân.
- 8.** Viết chương trình hợp ngữ Assembly chuyển một số từ hệ cơ số 10 sang hệ cơ số 16 (Hexa).
- 9.** Viết chương trình hợp ngữ Assembly cho phép nhập số nhị phân (8 bit) chứa vào trong thanh ghi BL. Chương trình phải kiểm tra

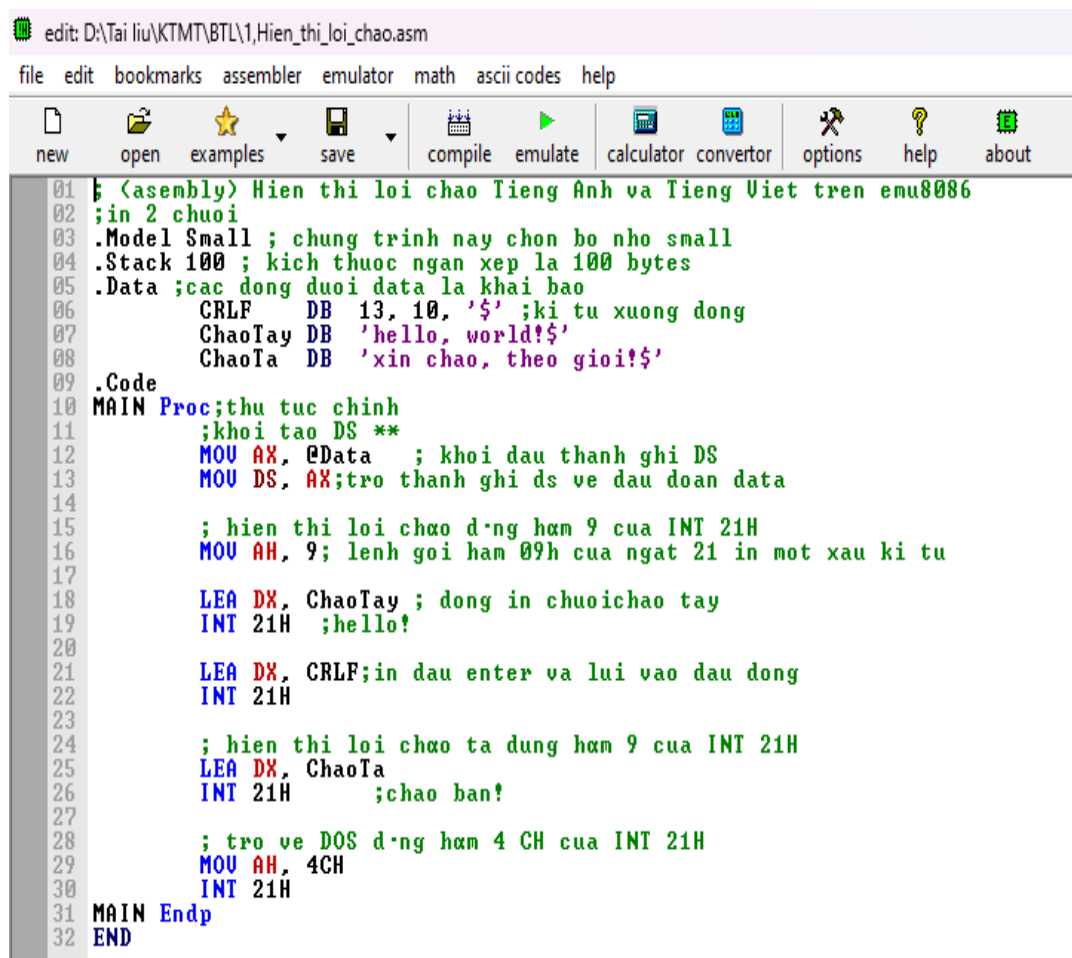
ký tự nhập có hợp lệ hay không (ký tự “0” hoặc ký tự “1”). Việc nhập kết thúc khi nhấn # hoặc đủ 8 bit. Xuất ra số đã nhập dưới dạng hệ thập lục phân (hệ 16).

- 10.** Viết chương trình hợp ngữ Assembly yêu cầu đếm chiều dài của một chuỗi ký tự cho trước.
- 11.** Viết chương trình hợp ngữ Assembly tìm giá trị lớn nhất và nhỏ nhất của một mảng số.
- 12.** Viết chương trình hợp ngữ Assembly cho phép nhập vào một số và in ra màn hình giai thừa của số đó.
- 13.** Viết chương trình hợp ngữ Assembly cho phép nhập vào các số và in ra màn hình tổng của các số đó.
- 14.** Viết chương trình hợp ngữ Assembly cho phép nhập vào 2 số và in ra màn hình UCLN và BCNN của hai số đó.
- 15.** Viết chương trình hợp ngữ Assembly cho phép in ra số lượng các số chia hết cho 11 và tính tổng của các số đó từ một mảng cho trước.
- 16.** Viết chương trình hợp ngữ Assembly tính tổng 2 số kiểu word.
- 17.** Viết chương trình hợp ngữ cho phép nhập vào một mảng gồm 10 số có hai chữ số. Tính tổng các số chia hết cho 7. In tổng thu được ra màn hình dưới dạng thập phân.
- 18.** Viết chương trình hợp ngữ đếm số lần xuất hiện của chuỗi con “ktmt” trong một chuỗi. In kết quả dưới dạng số thập phân.
- 19.** Viết chương trình hợp ngữ cho hai chuỗi ký tự A và B có độ dài là n và m ( $n > m$ ), chỉ ra xâu B có phải là xâu con của xâu A không?  
Nếu xâu B là xâu con của xâu A thì chỉ ra vị trí xâu B ở xâu A.
- 20.** Viết chương trình hợp ngữ cho hai chuỗi ký tự A và B có độ dài là n và m ( $n > m$ ), chỉ ra xâu A chứa mấy xâu B.

## **II. Lập trình điều khiển đèn Led, điều khiển nhiệt kế**

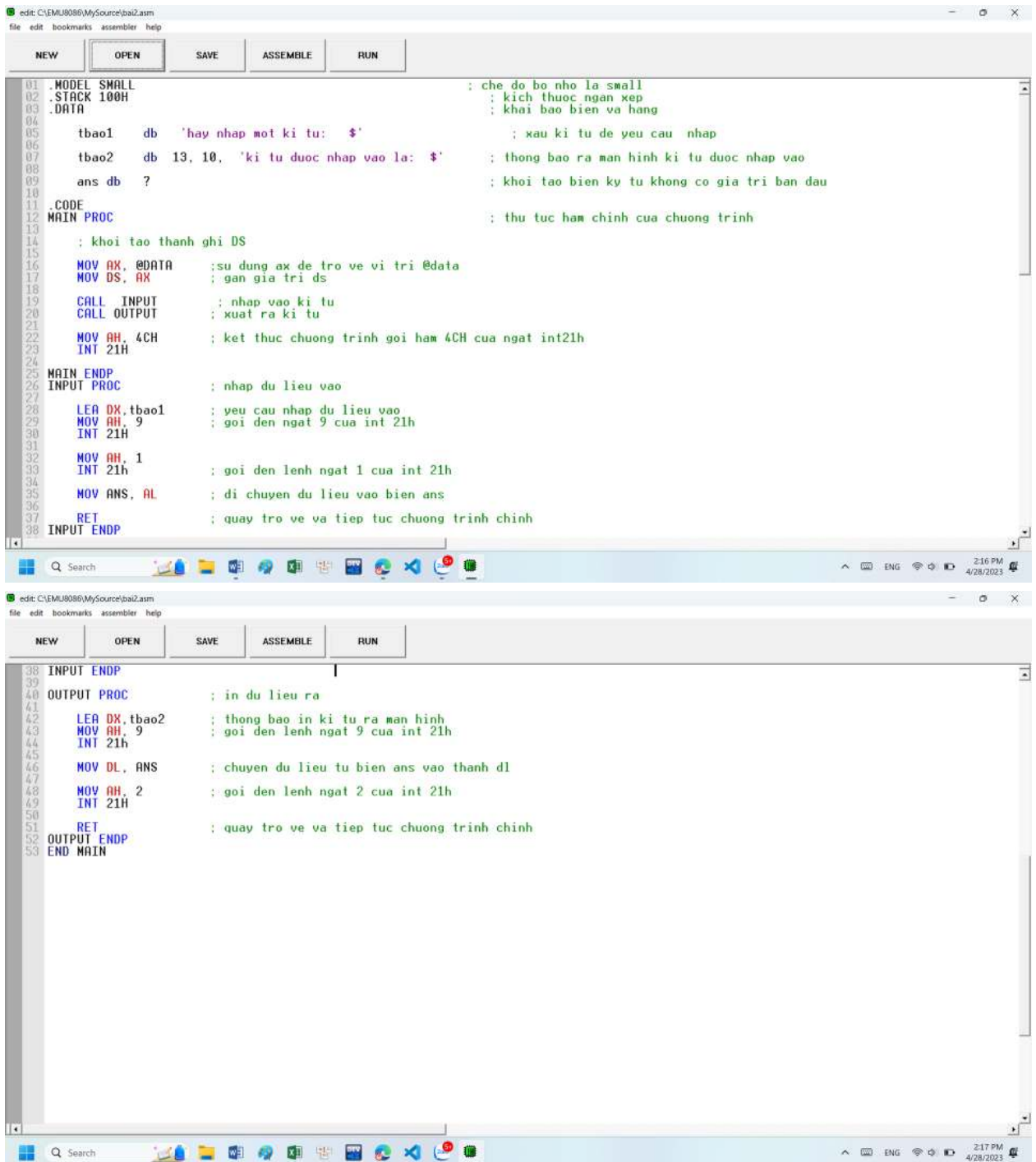
## I. Lập trình chương trình hợp ngữ Assembly

### Câu 1:



```
edit: D:\Tai lieu\KTMT\BTL\1,Hien_thi_loi_chao.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
01 ; (assembly) Hien thi loi chao Tieng Anh va Tieng Viet tren emu8086
02 ; in 2 chuoai
03 .Model Small ; chung trinh nay chon bo nho small
04 .Stack 100 ; kich thuoc ngan xep la 100 bytes
05 .Data ; cac dong duoi data la khai bao
06     CRLF DB 13, 10, '$' ; ki tu xuong dong
07     ChaoTay DB 'hello, world!$'
08     ChaoTa DB 'xin chào, theo giới!$'
09 .Code
10 MAIN Proc; thu tuc chinh
11     ; khai tao DS **
12     MOV AX, @Data ; khai dau thanh ghi DS
13     MOV DS, AX; tro thanh ghi ds ve dau doan data
14
15     ; hien thi loi chao d'ng ham 9 cua INT 21H
16     MOV AH, 9; lenh goi ham 09h cua ngat 21 in mot xau ki tu
17
18     LEA DX, ChaoTay ; dong in chuoichao tay
19     INT 21H ;hello!
20
21     LEA DX, CRLF; in dau enter va lui vao dau dong
22     INT 21H
23
24     ; hien thi loi chao ta dung ham 9 cua INT 21H
25     LEA DX, ChaoTa
26     INT 21H ;chao ban!
27
28     ; tro ve DOS d'ng ham 4 CH cua INT 21H
29     MOV AH, 4CH
30     INT 21H
31 MAIN Endp
32 END
```

## Câu 2:



```
01 .MODEL SMALL ; che do bo nho la small
02 .STACK 100H ; kích thước ngăn xếp
03 .DATA ; khai báo biến và hằng
04
05 tbao1 db 'hay nhap mot ki tu: $' ; xau ki tu de yeu cau nhap
06
07 tbao2 db 13, 10, 'ki tu duoc nhap vao la: $' ; thông báo ra màn hình ki tu duoc nhap vao
08
09 ans db ? ; khai tạo biến ký tự không có giá trị ban đầu
10
11 .CODE ; thu tục hàm chính của chương trình
12 MAIN PROC
13
14 ; khởi tạo thanh ghi DS
15
16 MOV AX, @DATA ; sử dụng ax để trở về vị trí @data
17 MOV DS, AX ; gán giá trị ds
18
19 CALL INPUT ; nhập vào ký tự
20 CALL OUTPUT ; xuất ra ký tự
21
22 MOV AH, 4CH ; kết thúc chương trình gọi hàm 4CH của ngắt int21h
23 INT 21H
24
25 MAIN ENDP
26 INPUT PROC ; nhập dữ liệu vào
27
28 LEA DX, tbao1 ; yêu cầu nhập dữ liệu vào
29 MOV AH, 9 ; gọi đến ngắt 9 của int 21h
30 INT 21H
31
32 MOV AH, 1 ; gọi đến lệnh ngắt 1 của int 21h
33 INT 21h
34
35 MOV ANS, AL ; di chuyển dữ liệu vào biến ans
36
37 RET ; quay trở về và tiếp tục chương trình chính
38 INPUT ENDP
39
40 OUTPUT PROC ; in dữ liệu ra
41
42 LEA DX, tbao2 ; thông báo in ký tự ra màn hình
43 MOV AH, 9 ; gọi đến lệnh ngắt 9 của int 21h
44 INT 21h
45
46 MOV DL, ANS ; chuyển dữ liệu từ biến ans vào thanh dl
47
48 MOV AH, 2 ; gọi đến lệnh ngắt 2 của int 21h
49 INT 21h
50
51 RET ; quay trở về và tiếp tục chương trình chính
52 OUTPUT ENDP
53 END MAIN
```

## Câu 3:

```
edit: D:\Ki2Nam2\KTMT\Emu8086\Project\c3.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
01 ;Nhap mot chuoai ky tu va in ra man hinh chuoai do.
02
03 .MODEL SMALL ;che do cua bo nho la small
04 .STACK 100H ;kich thuoc ngan xep
05 .DATA
06 CRTF DB 13, 10, '$' ;10 la xuong dong, 13 lui vao dau dong
07 MS1 DB "Nhap vao mot chuoai ky tu: $"
08 MS2 DB 13, 10, "Chuoai ky tu da nhap la: $"
09 STR DB 100 DUP('$') ;khai bao mot xau ki tu
10 .CODE
11 ;phan chuong trinh chinh
12 MAIN PROC
13 ;khởi tạo thanh ghi DS
14 MOV AX, @data ;su dung AX de tro ve @data
15 MOV DS, AX ;DS gan bang AX
16
17 LEA DX, MS1 ;hien thi ms1
18 MOV AH, 9 ;su dung ham 9 cua ngat INT21H
19 INT 21H
20
21 LEA DX, STR ;su dung ham 10 cua ngat INT 21H
22 MOV AH, 10 ;nhap vao 1 chuoai ki tu
23 INT 21H
24
25 LEA DX, MS2 ;hien thi ms2
26 MOV AH, 9
27 INT 21H
28 ;in ra chuoai ki tu vua nhap
29 LEA DX, STR + 2 ;dua DX tro ve phan tu thu 2 cua mang
30 MOV AH, 9
31 INT 21H
32
33 MOV AH, 4CH ;ket thuc chuong trinh bang ham 4CH cua ngat INT21H
34 INT 21H
35 MAIN ENDP
36 END MAIN
line: 34 col: 16 drag a file here to open
9:40 AM 4/27/2023
```

Câu 4:

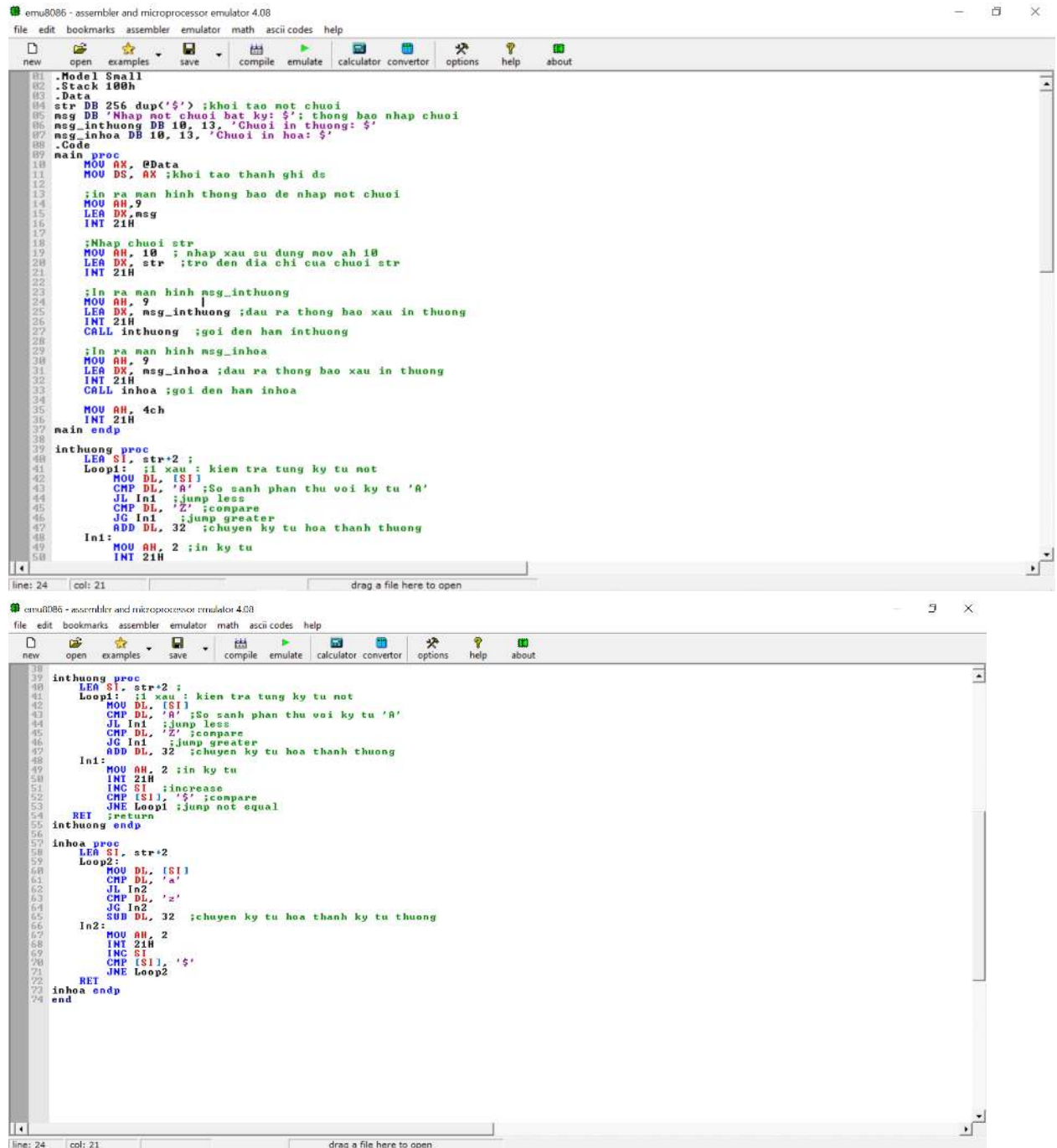
```

emu8086 - assembler and microprocessor emulator 4.03
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

01 .Model $small
02 .Stack 100h
03 .Data
04 str DB 256 dup('$') ;khởi tạo chuỗi str
05 tb1 DB 10, 13, 'Chuyen sang ky tu in hoa: $'
06 .Code
07 main proc
08     MOV AX, @Data
09     MOV DS, AX ;khởi tạo thanh ghi dữ
10
11     ;Nhập chuỗi:
12     LEA DX, str ;trở về địa chỉ chuỗi str
13     MOV AH, 10 ;nhập vào ngay 10
14     INT 21h
15
16     ;In chuỗi in hoa:
17     MOV AH, 9
18     LEA DX, tb1 ;hiển thông báo tb2
19     INT 21h
20     CALL inhoa
21
22     MOV AH, 4ch
23     INT 21h
24 main endp
25
26 inhoa proc
27     LEA SI, str+2
28     Lap2:
29         MOV DL, [SI]
30         CMP DL, 'a'
31         JL In2
32         CMP DL, 'z'
33         JG In2
34         SUB DL, 32 ;chuyển ký tự hoa thành ký tự thường
35     In2:
36         MOV AH, 2
37         INT 21h
38         INC SI
39         CMP [SI], '$'
40         JNE Lap2
41     RET
42 inhoa endp
43 end
44
45

```

## Câu 5:



The image shows two screenshots of the emu8086 - assembler and microprocessor emulator 4.08 interface. The top screenshot displays the assembly code for a program that reads a string and converts it to uppercase. The bottom screenshot shows the continuation of the code, including the 'inthuong' and 'inhua' procedures.

```
0001 .Model Small
0002 .Stack 100h
0003 .Data
0004 str DB 256 dup('$') ;khởi tạo một chuỗi
0005 msg DB 'Nhập một chuỗi bất kỳ: $'; thông báo nhập chuỗi
0006 msg_inthuong DB 10, 13, 'Chuỗi in thường: $'
0007 msg_inhoa DB 10, 13, 'Chuỗi in hoa: $'
0008 .Code
0009 main proc
0010     MOV AX, @Data
0011     MOV DS, AX ;khởi tạo thanh ghi ds
0012
0013     ;in ra màn hình thông báo để nhập một chuỗi
0014     MOV AH, 9
0015     LEA DX, msg
0016     INT 21h
0017
0018     ;Nhập chuỗi str
0019     MOV AH, 10 ;nhập vào số dùng mov ah 10
0020     LEA DX, str ;trỏ đến địa chỉ của chuỗi str
0021     INT 21h
0022
0023     ;in ra màn hình msg_inthuong
0024     MOV AH, 9
0025     LEA DX, msg_inthuong ;đầu ra thông báo vào in thường
0026     INT 21h
0027     CALL inthuong ;gọi đến hàm inthuong
0028
0029     ;in ra màn hình msg_inhoa
0030     MOV AH, 9
0031     LEA DX, msg_inhoa ;đầu ra thông báo vào in thường
0032     INT 21h
0033     CALL inhua ;gọi đến hàm inhua
0034
0035     MOV AH, 4ch
0036     INT 21h
0037 main endp
0038
0039 inthuong proc
0040     LEA SI, str+2 ;
0041     Loop1: ;l xau : kiểm tra từng ký tự một
0042     MOV DL, [SI]
0043     CMP DL, 'A' ;So sánh phần tử với ký tự 'A'
0044     JL In1 ;jump less
0045     CMP DL, 'Z' ;compare
0046     JG In1 ;jump greater
0047     ADD DL, 32 ;chuyển ký tự hoa thành thường
0048     In1:
0049     MOV AH, 2 ;in ký tự
0050     INT 21h
0051     INC SI ;increase
0052     CMP SI, '$' ;compare
0053     JNE Loop1 ;jump not equal
0054     RET ;return
0055 inthuong endp
0056
0057 inhua proc
0058     LEA SI, str+2
0059     Loop2:
0060     MOV DL, [SI]
0061     CMP DL, 'a'
0062     JL In2
0063     CMP DL, 'z'
0064     JG In2
0065     SUB DL, 32 ;chuyển ký tự hoa thành ký tự thường
0066     In2:
0067     MOV AH, 2
0068     INT 21h
0069     INC SI
0070     CMP SI, '$'
0071     JNE Loop2
0072     RET
0073 inhua endp
0074 end
```

## Câu 6:

```
edit: D:\Tai lieu\KTMT\BTL\6\In_ra_chuoi_ki_tu_nguoc_ket_thuc_bang_dau_#.asm
file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

01 ; <assembly> in chuoi d?o ngu?c khi di?n d?u # tr?n emu8086
02 ;enter la 13 , esc la 27
03 .Model small
04 .Stack 100h
05 .Data
06     str db 50 dup ('$')    ; khoi tao chuoi 50 bytes
07 .code
08 main proc
09     mov ax,0Data
10     mov ds,ax ;khoi tao thanh ghi ds
11     mov cx,0;gan gia tri cho thanh ghi cx=0
12 START:
13     inc cx;increase tang cx len 1
14     mov ah,1 ;nhap 1 ki tu
15     int 21h
16     cmp al,'#';so sanh al voi ki tu # => enter la 13
17     je end ;jump equal :nhay den 'end' neu bang
18     push ax ; them phan tu vao ngan xep
19     jmp START;nhay k dieu kien
20
21     end;;ket thuc
22     mov ah,2 ;in ky tu
23     mov dl,0; in dau cach
24     int 21h
25
26     dec cx;decrease:tru cx di 1
27     pop dx;lay phan tu o dinh ngan xep dua vao dx
28     mov ah,2 ;in ky tu
29     int 21h
30
31     cmp cx,1;so sanh cx voi 1
32     jne end;jump not equal :nhay den 'end' neu khong bang
33     ;ve dos
34     Mov ah, 4Ch
35     Int 21h
36 main endp
37 end
```

## Câu 7:



```

edit: C:\EMU0808\MySource\bai7.asm
file edit bookmarks assembler help

NEW OPEN SAVE ASSEMBLE RUN

01 .MODEL SMALL           ; che do bo nho la small
02 .STACK 100H           ; kich thuoc ngan xep
03 .DATA
04                       ; khai bao cac bien va hang
05     tb db 10, 13, 'So da nhap dang nhi phan: $'
06     tb1 db 13, 10, 'moi ban nhap so : $'
07     str db 5 dup ('$') ; nhap vao 1 chuoai toi da 5 ky tu
08
09 .code
10 MAIN PROC
11     MOV AX, @DATA       ;
12     MOV DS, AX          ;khởi tạo thanh ghi ds
13
14     MOV AX, 'H'
15     PUSH AX             ;push dấu H (ascii) vào trong ngan xep
16
17     lea dx, tb1          ; in ra yêu cầu nhập số
18     mov ah, 9
19     int 21h
20
21
22     ;Nhập số dạng chuỗi:
23
24     MOV AH, 10
25     LEA DX, str          ; nhập chuỗi str
26     INT 21h
27
28     ;Chuyển chuỗi thành số:
29
30     MOV CL, [str+1]      ; lấy số ký tự của chuỗi
31     MOV CH, 0            ; trả về 0
32     LEA SI, str+2        ; trỏ đến địa chỉ của ký tự đầu tiên của chuỗi str
33     MOV AX, 0            ; khởi tạo giá trị ban đầu bằng 0
34     MOV BX, 10           ; bx=10 ;hệ số nhân
35
36     ;chuyển chuỗi thành số
37
38
39     MOV BX, 10           ;bx=10 ;hệ số nhân
40
41     ;chuyển chuỗi thành số
42
43     thapphan:
44     MUL BX               ;nhân 10
45     MOV DL, [SI]
46     SUB DL, '0'          ; chuyển từ ký tự thành số
47     ADD AX, DX            ; ax=ax+dx
48     INC SI               ; tăng si
49     LOOP thapphan
50
51     MOV CL, 2            ; hệ số chia
52
53     ;chuyển số thập phân sang nhị phân và đẩy các số vào ngan xep
54     nhiphan:
55     MOV AH, 0            ;phan dư =0
56     DIV CL               ; chia ax cho 2
57     PUSH AX              ; đẩy ax vào ngan xep (al·ah)
58     CMP AL, 0            ;so sánh thương#0 tiếp tục chia
59     JNE nhiphan          ;jump not equal
60
61     MOV AH, 9
62     LEA DX, tb;
63     INT 21h
64
65     MOV AH, 2
66
67     output:
68     POP DX               ;lấy từng phần tử trong ngan xep
69     CMP DX, 'H'
70     JE Done              ;jump equal
71     MOV DL, DH
72     ADD DL, '0'
73     INT 21h

```

```
edit: C:\EMU0808\MySource\bai7.asm
file edit bookmarks assembler help

NEW OPEN SAVE ASSEMBLE RUN

64
65 output:
66 POP DX ;lay tung phan tu trong ngan xep
67 CMP DX, '#'
68 JE Done ;jump equal
69 MOV DL, DH ;lay duoc so tu ngan xep
70 ADD DL, '0'
71 INT 21h
72 JMP output
73 Done:
74 MOV AH, 4Ch
75 INT 21h
76 MAIN ENDP
77 END
```

Câu 8:

```

edit: D:\Ki2Nam2\KTMT\Emu8086\Project\C8.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

001 ;Chuyen tu he DEC sang he HEX
002 .MODEL SMALL
003 .DATA
004
005 CRTF DB 13, 10, '$'
006 MS1 DB "DEC: $"
007 MS2 DB 13, 10, "Convert to HEX: $"
008 str DW 100 dup('$')
009 tmp DW 0 ;bien luu tru tam thoi
010 muoi DW 10 ;luu gia tri 10
011
012 .CODE
013 ;chuong trinh chinh
014 Main Proc
015 ;khởi tạo thanh ghi DS
016 MOV AX, @data ;su dung AX de tro ve @data
017 MOV DS, AX ;gan DS bang AX
018
019 LEA DX, MS1 ;hien thi MS1
020 MOV AH, 9
021 INT 21H
022
023 CALL Input ;nhap vao DEC
024 CALL Convert16 ;chuyen thanh HEX
025
026 ;ket thuc chuong trinh
027 MOV AH, 4CH
028 INT 21H
029
030 Main Endp
031

```

line: 36 col: 25

```

Input Proc ;nhap vao str va chuyen sang so
;xu li so co nhieu hon 2 chu so
MOV AH, 10 ;nhap vao str
LEA DX, str
INT 21H

LEA SI, str + 2 ;bat dau xau str
MOV CX, [str + 1] ;do dai str nhap vao
XOR CH, CH ;CH = 0

;Loop den khi CX=0->duyet hat xau str
Lap:

MOV AX, tmp ;dua gia tri tu tmp ve AX
MOV BX, [SI] ;ki tu c của si
SUB BX, 48 ;chuyen sang so

MUL muoi ;AX = AX * 10
ADD AX, BX ;AX = AX + BX
MOV tmp, AX ;luu vao tmp
INC SI ;tang index của xau str
LOOP Lap

ret ;return

Input Endp

```

col: 5

```
060 Convert16 Proc
061 ;chuyen so da nhap tu he DEC sang HEX
062 LEA DX, MS2 ;hien thi MS2
063 MOV AH, 9
064 INT 21H
065
066 MOV AX, tmp ;AX dang luu tru so can chuyen
067 MOV AH, 0 ;AH = 0
068 MOV BX, 16 ;BX = 16
069 MOV CX, 0 ;CX = 0
070
071 chia:
072 DIV BL ;lay so hien tai chia cho 16
073 PUSH AX ;push AX vao stack
074 INC CX ;tang bien dem CX
075 CMP AL, 0 ;neu thuong AL = 0
076 JE hienthi ;jump to hienthi
077 MOV AH, 0 ;xoaphan du tai AH
078 JMP chia ;continue
079
080 hienthi:
081
082 POP AX ;lay ra tu stack
083 MOV DL, AH
084 ;truong hop la ki tu tu A->F
085
086 CMP DL, 10
087 JE caseA ;chuyen den caseA
088
089 CMP DL, 11
090 JE caseB ;chuyen den caseB
091
092 CMP DL, 12
093 JE caseC ;chuyen den caseC
```

line: 94 col: 10 drag a file here to open

Zalo DEV Google Chrome Word 10:29 AM 4/30/2023

```

094
095     CMP DL, 13
096     JE caseD      ;chuyen den caseD
097
098     CMP DL, 14
099     JE caseE      ;chuyen den caseE
100
101     CMP DL, 15
102     JE caseF      ;chuyen den caseF
103     ;neu khong phai ki tu tu A->F
104     ADD DL, '0'    ;chuyen so thanh ky tu
105     JMP print
106
107     ;chuyen doi sang ki tu phu hop
108 caseA:
109     MOV DL, 'A'
110     JMP print
111 caseB:
112     MOV DL, 'B'
113     JMP print
114 caseC:
115     MOV DL, 'C'
116     JMP print
117 caseD:
118     MOV DL, 'D'
119     JMP print
120 caseE:
121     MOV DL, 'E'
122     JMP print
123 caseF:
124     MOV DL, 'F'
125     JMP print
126

```

ne: 94 col: 10

drag a file here to open

Zalo DEV B W 10:31 AM 4/30/2023

```

126
127     ;in ra ki tu vua xu li
128 print:
129     MOV AH, 2
130     INT 21H
131
132     LOOP hienthi  ;lap den khi da in het
133     ret          ;return
134
135 Convert16 Endp
136 END
137
138

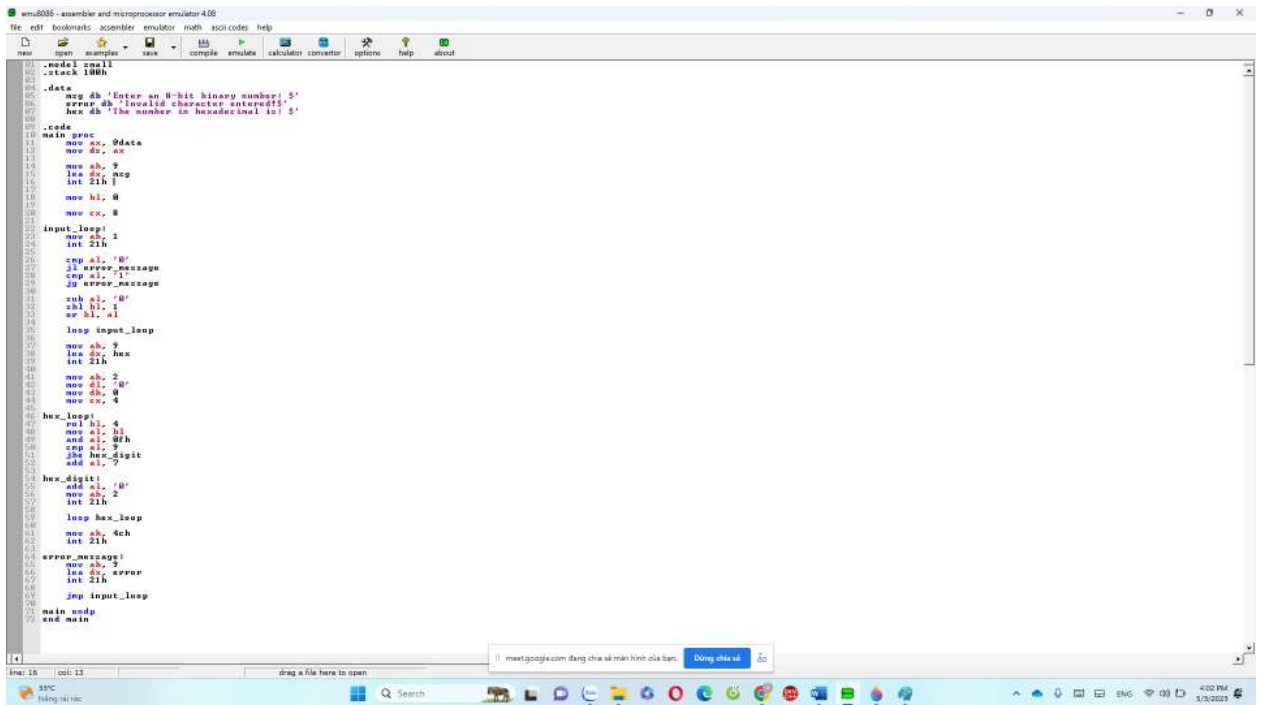
```

ne: 123 col: 20

drag a file here to open

Zalo DEV B W 10:00 AM 4/27/2023

## Câu 9:

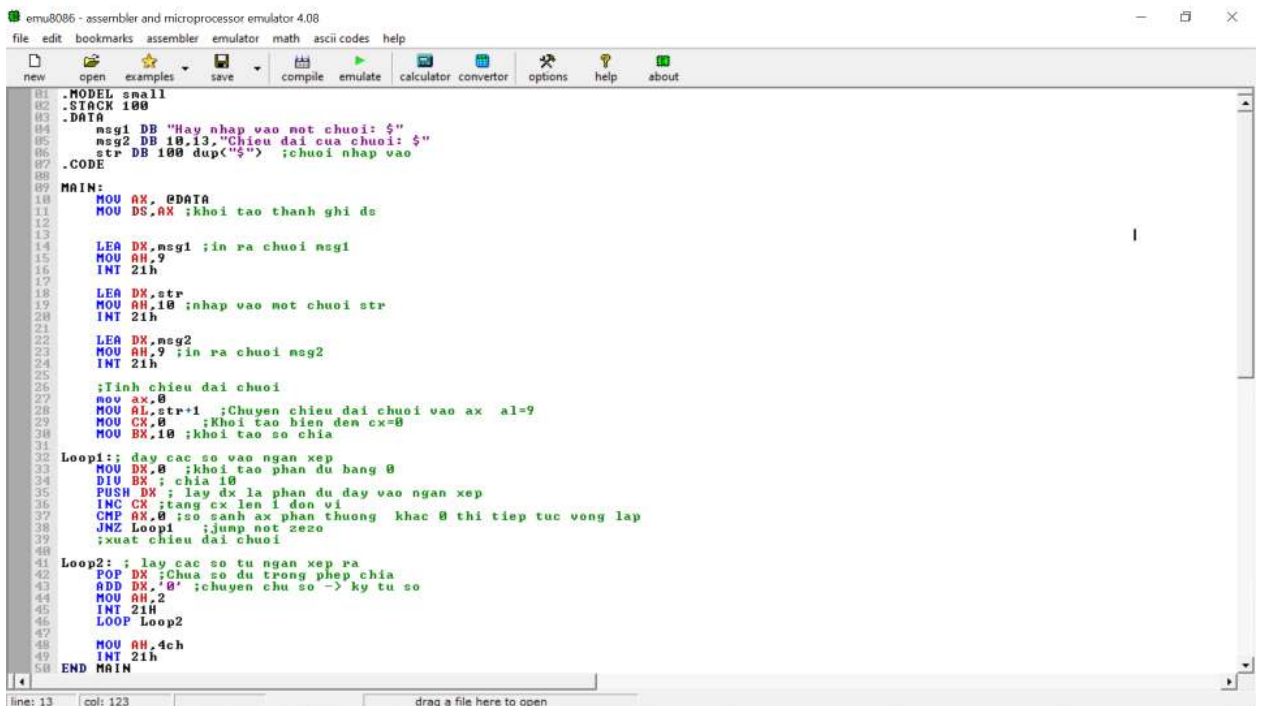


```
.model small
.stack 100h

.data
msg db 'Enter an 8-bit binary number: $'
error db 'Invalid character entered! $'
hex db 'The number in hexadecimal is: $'

.code
main proc
    mov ax, @data
    mov dx, ax
    mov ah, 9
    lea dx, msg
    int 21h
    mov hl, 0
    mov cx, 8
input_loop:
    mov ah, 1
    int 21h
    cmp al, '0'
    jl error_message
    cmp al, '1'
    jl error_message
    sub al, '0'
    shl hl, 1
    or hl, al
    loop input_loop
    mov ah, 9
    lea dx, hex
    int 21h
    mov ah, 2
    mov dl, al
    mov dx, 0
    mov cx, 4
hex_loop:
    rol hl, 4
    mov al, hl
    and al, 0fh
    cmp al, 9
    jbe hex_digit
    add al, 7
hex_digit:
    add si, 2
    int 21h
    loop hex_loop
    mov ah, 4ch
    int 21h
error_message:
    mov ah, 9
    lea dx, error
    int 21h
    jmp input_loop
main endp
end main
```

## Câu 10:



```
.MODEL small
.STACK 100h
.DATA
msg1 DB "Hay nhap vao mot chuo: $"
msg2 DB 10,13,"Chieu dai cua chuo: $"
str DB 100 dup("$") ;chuo nhap vao
.CODE
MAIN:
    MOV AX, @DATA
    MOV DS, AX ;khởi tạo thanh ghi ds
    LEA DX, msg1 ;in ra chuo msg1
    MOV AH, 9
    INT 21h
    LEA DX, str
    MOV AH, 10 ;nhap vao mot chuo str
    INT 21h
    LEA DX, msg2
    MOV AH, 9 ;in ra chuo msg2
    INT 21h
    ;Tinh chieu dai chuo
    mov ax, 0
    MOV AL, str+1 ;Chuyen chieu dai chuo vao ax al=9
    MOV CX, 0 ;Khởi tạo biến đếm cx=0
    MOV BX, 10 ;khởi tạo số chia
Loop1: ;day cac so vao ngan xep
    MOV DX, 0 ;khởi tạo phần dư bằng 0
    DIV BX ; chia 10
    PUSH DX ; lay dx laphan du day vao ngan xep
    INC CX ;tang cx len 1 don vi
    CMP AX, 0 ;so sanh axphan thuong khac 0 thi tiep tục vòng lặp
    JNZ Loop1 ;jump not zero
    ;xuat chieu dai chuo
    Loop2: ; lay cac so tu ngan xep ra
    POP DX ;Chua so du trong phép chia
    ADD DX, '0' ;chuyen chu so -> ky tu so
    MOV AH, 2
    INT 21h
    LOOP Loop2
    MOV AH, 4ch
    INT 21h
END MAIN
```



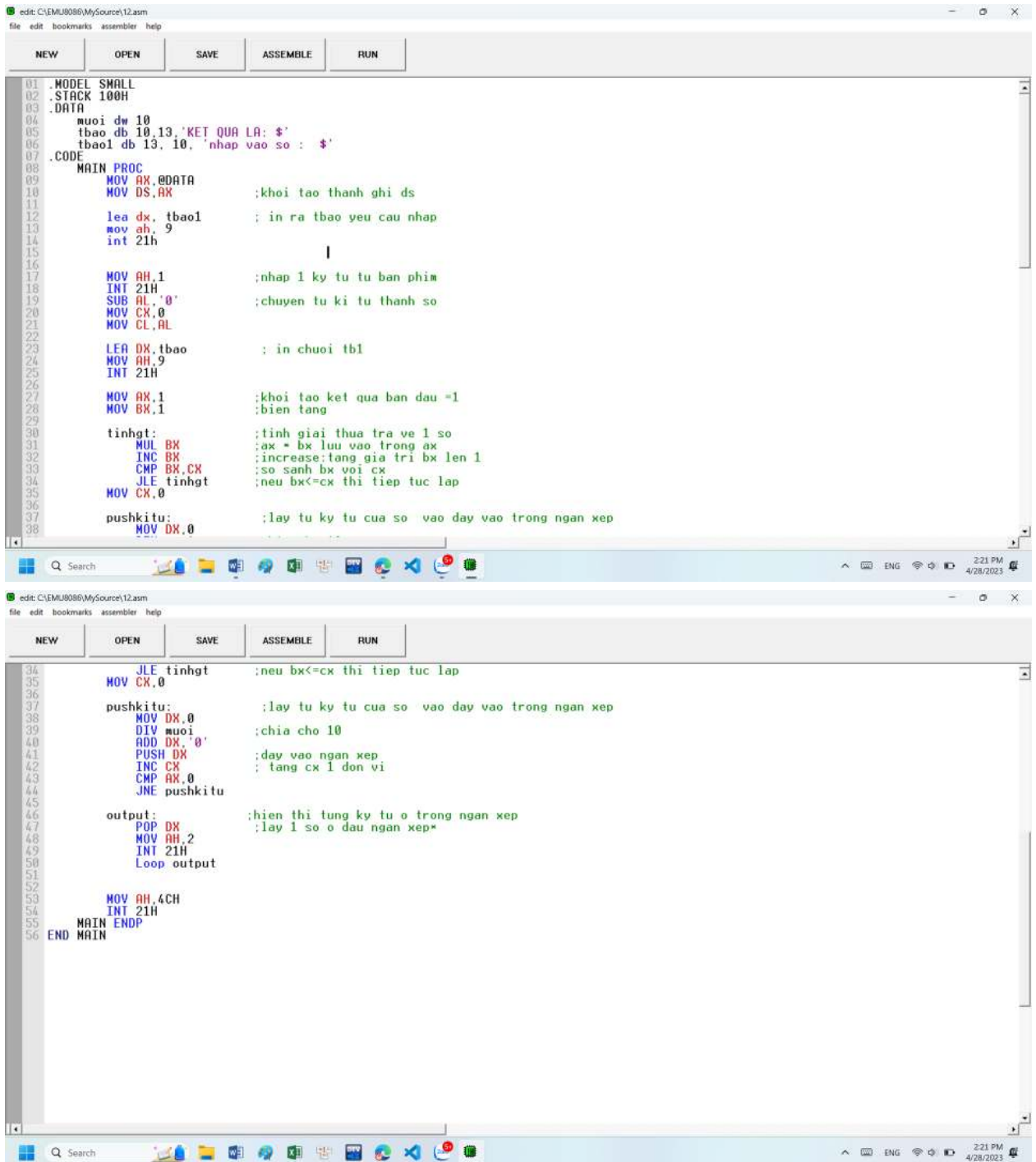
## Câu 11:

```
edit: D:\Tai lieu\KTMT\BTL\11,Tim_gia_tri_lon_nhat_va_nho_nhat_cua_mot_day_so.asm
file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

01 ; <assembly> Tim gia tri lon nhat va nho nhat cua mot day so
02 .Model small
03 .Stack 100H
04 .Data
05 list DB 1,2,3,4,5,6,7,8,0
06 .code
07 main proc
08     ; initilize the ds and es registers
09     mov ax, @Data ;
10     mov ds,ax ;
11     mov cx, 9
12     lea si, list    ; dua gia tri dau tien cua chuoi vao si
13     mov bl, [si]    ; dua dia chi si vao bl
14     inc si          ; tang gia tri si them 1
15 Start:
16     lodsb
17     cmp bl, al ; so sanh al va bl
18     jge BYPASS; nhay denn BYPASS
19     mov bl, al; neu al > bl thi gan bl = al;
20     BYPASS:
21     loop Start ; lap
22
23     ; print the max
24     add bl, '0' ; ep kieu so ve kieu ke tu
25     mov dl,bl ; dua gia tri max bl vao dl;
26     mov ah, 2 ; in ra màn hình
27     int 21H
28
29     mov ah, 4CH ; ket thuc chuong trinh
30     int 21H
31 main endp
32 End Main
```

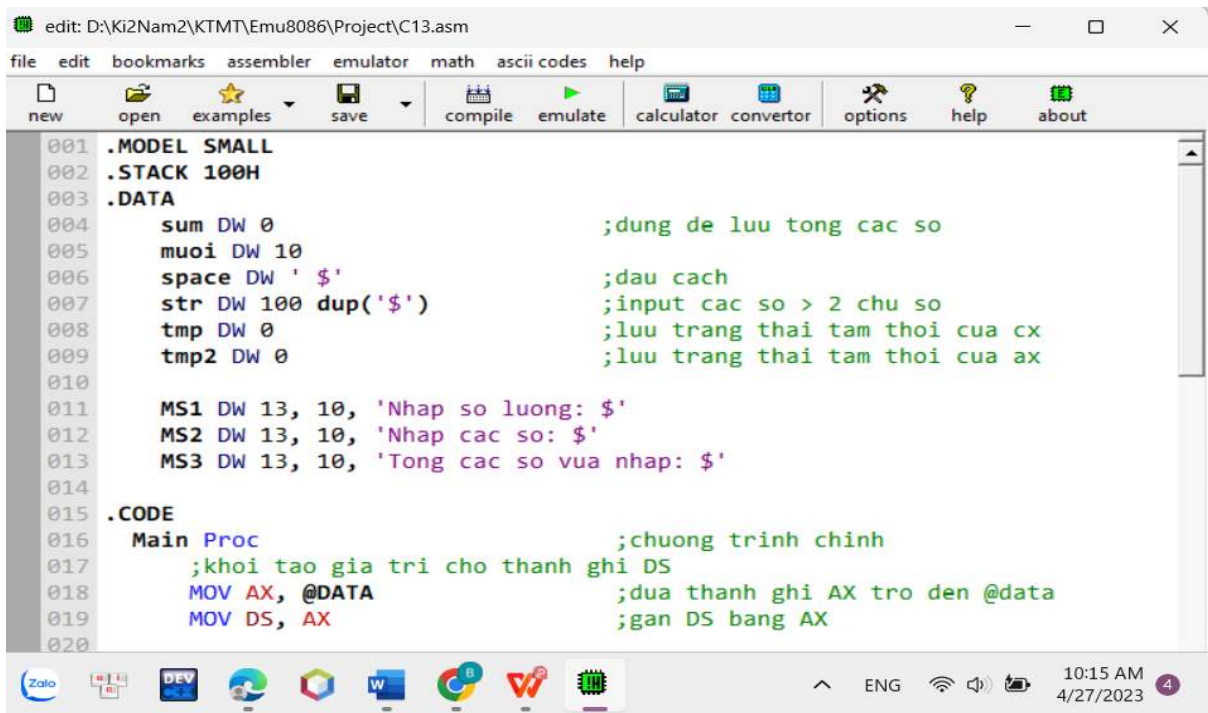
## Câu 12:



```
01 .MODEL SMALL
02 .STACK 100H
03 .DATA
04     muoi dw 10
05     tbao db 10,13,'KET QUA LA: $'
06     tbaol db 13,10,'nhap vao so : $'
07 .CODE
08     MAIN PROC
09         MOV AX,@DATA
10         MOV DS,AX           ;khởi tạo thanh ghi ds
11
12         lea dx,tbaol        ; in ra tbao yêu cầu nhập
13         mov ah,9
14         int 21h
15
16         |
17         MOV AH,1           ;nhập 1 ký tự từ bàn phím
18         INT 21H
19         SUB AL,'0'         ;chuyển từ ký tự thành số
20         MOV CX,0
21         MOV CL,AL
22
23         LEA DX,tbao         ; in chuỗi tb1
24         MOV AH,9
25         INT 21H
26
27         MOV AX,1           ;khởi tạo kết quả ban đầu =1
28         MOV BX,1           ;biến tăng
29
30         tinhgt:            ;tính giai thừa trả về 1 số
31             MUL BX         ;ax = bx lưu vào trong ax
32             INC BX         ;increase: tăng giá trị bx lên 1
33             CMP BX,CX      ;so sánh bx với cx
34             JLE tinhgt     ;nếu bx<=cx thì tiếp tục lặp
35             MOV CX,0
36
37         pushkitu:          ;lấy từ ký tự của số vào đây vào trong ngăn xếp
38             MOV DX,0
39             DIV muoi        ;chia cho 10
40             ADD DX,'0'
41             PUSH DX        ;đẩy vào ngăn xếp
42             INC CX          ;tăng cx 1 đơn vị
43             CMP AX,0
44             JNE pushkitu
45
46         output:           ;hiện thị từng ký tự ở trong ngăn xếp
47             POP DX         ;lấy 1 số ở đầu ngăn xếp*
48             MOV AH,2
49             INT 21H
50             Loop output
51
52
53         MOV AH,4CH
54         INT 21H
55     MAIN ENDP
56 END MAIN
```



### Câu 13:



```
edit: D:\Ki2Nam2\KTMT\Emu8086\Project\C13.asm
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
0001 .MODEL SMALL
0002 .STACK 100H
0003 .DATA
0004     sum DW 0 ;dung de luu tong cac so
0005     muoi DW 10
0006     space DW ' '$' ;dau cach
0007     str DW 100 dup('$') ;input cac so > 2 chu so
0008     tmp DW 0 ;luu trang thai tam thoi cua cx
0009     tmp2 DW 0 ;luu trang thai tam thoi cua ax
0010
0011     MS1 DW 13, 10, 'Nhap so luong: '$'
0012     MS2 DW 13, 10, 'Nhap cac so: '$'
0013     MS3 DW 13, 10, 'Tong cac so vua nhap: '$'
0014
0015 .CODE
0016 Main Proc ;chuong trinh chinh
0017     ;khoi tao gia tri cho thanh ghi DS
0018     MOV AX, @DATA ;dua thanh ghi AX tro den @data
0019     MOV DS, AX ;gan DS bang AX
0020
```



```

055 Input Proc
056 ;nhap vao str va chuyen sang so
057 MOV AH, 10 ;nhap vao str
058 LEA DX, str
059 INT 21H
060
061 LEA SI, str + 2 ;bat dau xau str
062 MOV CX, [str + 1] ;do dai str nhap vao
063 XOR CH, CH ;CH = 0
064 MOV AX, 0
065 MOV tmp2, AX
066 Lap2:
067 MOV BX, 0
068 MOV AX, tmp2 ;dua gia tri tu tmp2 ve AX
069 MOV BX, [SI] ;ki tu c của str
070 MOV BH, 0
071 SUB BX, '0' ;chuyen sang so
072
073 MUL muoi ;AX = AX * 10
074 ADD AX, BX ;AX = AX + BX
075 MOV tmp2, AX ;luu vao tmp2
076 INC SI ;tang index của xau str
077 LOOP Lap2
078 ret ;return
079
080 Input Endp
081

```

line: 83 col: 9

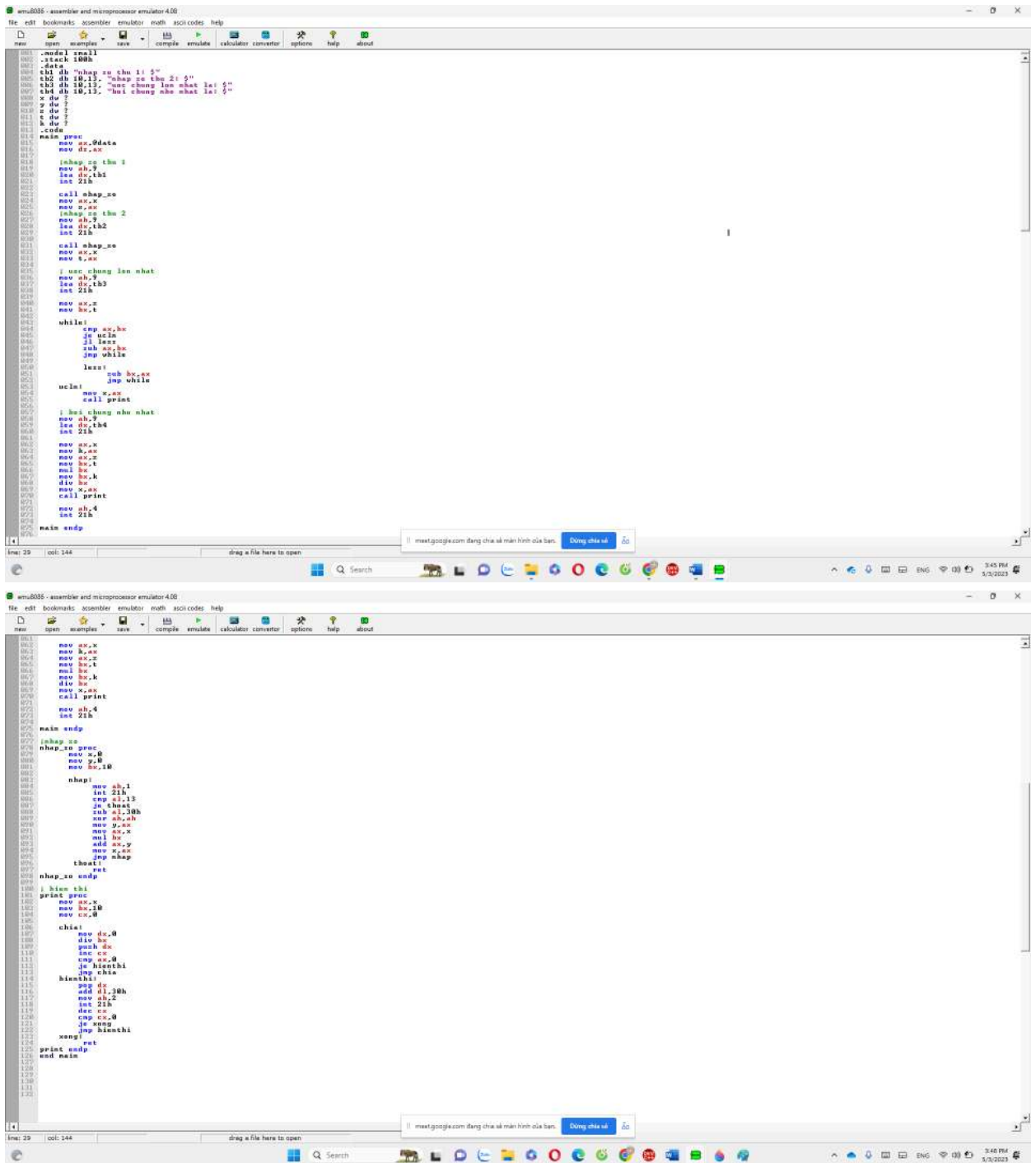
```

083 Output Proc
084
085 LEA DX, MS3 ;hien thi MS3
086 MOV AH, 9
087 INT 21H
088
089 MOV AX, SUM ;AX = sum
090
091 MOV BX, 10 ;BX = 10
092 MOV CX, 0 ;CX = 0 : dem so luong push vao stack
093
094 chia:
095
096 MOV DX, 0
097 DIV BX ;AX/10 phan du luu tai DX
098 PUSH DX ;push DX vao stack
099 INC CX ;CX++
100 CMP AX, 0 ;neu AX = 0 thi jump den hienthi
101 JE hienthi
102 JMP chia ;tiếp tục chia
103
104
105 hienthi:
106 POP DX ;lay DX ra khỏi stack
107 ADD DL, 48 ;chuyen tu so sang ki tu
108 MOV AH, 2 ;in ra ki tu vừa xử lý
109 INT 21H
110 DEC CX ;CX--
111 CMP CX, 0 ;neu CX = 0 thi dung
112 JNE hienthi
113 ret
114 Output Endp
115
116

```

line: 116 col: 3

## Câu 14:



```

0001 .model intel
0002 .stack 100h
0003
0004 ;chỉ số nhập số thứ i là
0005 ;chỉ số 10,11, nhập số thứ 2 là
0006 ;chỉ số 10,12, nhập số thứ 3 là
0007 ;chỉ số 10,13, nhập số thứ 4 là
0008 x du ?
0009 y du ?
0010 z du ?
0011 k du ?
0012
0013 .code
0014 main proc
0015     mov ax,data
0016     mov dx,ax
0017
0018     ;nhập số thứ 1
0019     mov ah,7
0020     lea dx,th1
0021     int 21h
0022
0023     call nhap_so
0024     mov ax,x
0025     mov dx,ax
0026     ;nhập số thứ 2
0027     mov ah,7
0028     lea dx,th2
0029     int 21h
0030
0031     call nhap_so
0032     mov ax,x
0033     mov t,ax
0034
0035     ; nhập chung lại
0036     mov ah,7
0037     lea dx,th3
0038     int 21h
0039
0040     mov ax,x
0041     mov bx,t
0042
0043     while:
0044         cmp ax,bx
0045         ja ucla
0046         jll lea
0047         sub bx,bx
0048         jmp while
0049
0050     lea:
0051         sub bx,ax
0052         jmp while
0053
0054     ucla:
0055         mov x,ax
0056         call print
0057
0058     ; nhập chung lại
0059     mov ah,7
0060     lea dx,th4
0061     int 21h
0062
0063     mov ax,x
0064     mov dx,ax
0065     mov bx,t
0066     mov cx,k
0067     div bx
0068     mov x,ax
0069     call print
0070
0071     mov ah,4
0072     int 21h
0073
0074 main endp
0075
0076

```

```

0001     mov ax,x
0002     mov bx,ax
0003     mov dx,ax
0004     mov bx,t
0005     mov cx,k
0006     div bx
0007     mov x,ax
0008     call print
0009
0010     mov ah,4
0011     int 21h
0012
0013 main endp
0014
0015
0016 ;nhập số
0017 nhap_so proc
0018     mov x,0
0019     mov y,0
0020     mov z,0
0021
0022     nhap:
0023         mov ah,1
0024         int 21h
0025         cmp al,13
0026         ja thuat
0027         sub al,30h
0028         mov y,al
0029         lea dx,x
0030         mul bx
0031         add ax,y
0032         mov x,ax
0033         jmp nhap
0034     thuat:
0035         ret
0036 nhap_so endp
0037
0038 ; hiển thị
0039 print proc
0040     mov ax,x
0041     mov bx,10
0042     mov cx,0
0043
0044     chia:
0045         mov dx,0
0046         div bx
0047         push dx
0048         inc cx
0049         cmp ax,0
0050         je hienthi
0051         jmp chia
0052     hienthi:
0053         pop dx
0054         add dx,30h
0055         mov ah,2
0056         int 21h
0057         dec cx
0058         cmp cx,0
0059         je xong
0060         jmp hienthi
0061     xong:
0062         ret
0063 print endp
0064 end main

```

```

File  edit  bookmarks  assembler  emulator  math  ascii codes  help
new  open  examples  save  compile  emulate  calculator  convertor  options  help  about

0001 .model small
0002 .stack 100h
0003 .data
0004 th1 db "nhap so thu 1: %"
0005 th2 db 10,13, "nhap so thu 2: %"
0006 th3 db 10,13, "so chung lon nhat la: %"
0007 th4 db 10,13, "boi chung nho nhat la: %"
0008 x du ?
0009 y du ?
0010 z du ?
0011 t du ?
0012 h du ?
0013 .code
0014 main proc
0015     mov ax,@data
0016     mov ds,ax
0017
0018     ;nhap so thu 1
0019     mov ah,9
0020     lea dx,th1
0021     int 21h
0022
0023     ;call nhap so
0024     mov ax,x
0025     mov t,ax
0026
0027     ;nhap so thu 2
0028     mov ah,9
0029     lea dx,th2
0030     int 21h
0031
0032     ;call nhap so
0033     mov ax,x
0034     mov t,ax
0035
0036     ; so chung lon nhat
0037     mov ah,9
0038     lea dx,th3
0039     int 21h
0040
0041     mov ax,z
0042     mov h,x
0043
0044     while:
0045         cmp ax,hx
0046         je ucln
0047         jl less
0048         sub ax,hx
0049         jmp while
0050
0051     less:
0052         sub hx,ax
0053         jmp while
0054
0055     ucln:
0056         mov x,ax
0057         call print
0058
0059     ; boi chung nho nhat
0060     mov ah,9

```

## Câu 15:

```

emu8086 - assembler and microprocessor emulator 4.08
file  edit  bookmarks  assembler  emulator  math  ascii codes  help
new  open  examples  save  compile  emulate  calculator  convertor  options  help  about

0001 .model small
0002 .stack 100h
0003 .data
0004 msg1 db 7,11,66,233,77,43,55,123,121
0005 msg1 db "Tong cac so chia het cho 11 la: %"
0006 msg2 db 10,13, "So luong cac so chia het cho 11 la: %"
0007 x du ?
0008 y du ?
0009 .code
0010 main proc
0011     mov ax,@data
0012     mov ds,ax ;khai tao thanh ghi ds
0013
0014     lea dx,msg1 ;lea chuoai msg vao thanh ghi dx
0015     mov ah,9 ;su dung han move ah,9 de in ra chuoai msg1
0016     int 21h
0017
0018
0019
0020     ;in ra tong cua cac so chia het cho 11
0021     mov x,0 ;khai tao bien x = 0
0022     mov cx,9 ;khai tao bien den cx bang voi so phan tu trong mang
0023     lea si,msg1 ;su dung thanh ghi si de tro toi dia chi cua mang
0024     mov bx,11 ;khai tao so chia
0025
0026
0027     while:
0028         mov dx,0 ;khai tao thanh ghi dx = 0
0029         mov ax,[si] ;khai tao thanh ghi ax voi dia chi si
0030         xor ah,ah ;xoa thanh ghi ah
0031         div bx ;chia cho bien bx da khai tao ben tren la bang 11
0032         cmp dx,0 ;so sanh neu chia het thi phan du la 0
0033         je tang
0034         jmp tieptuc
0035     tang:
0036         mov ax,[si]
0037         xor ah,ah
0038         add x,ax ;cong x voi phan tu trong mang duoc luu o ax
0039         inc y ;tang bien y len 1 don vi
0040     tieptuc:
0041         inc si ;tang bien si len 1 don vi
0042         loop while
0043
0044     mov bx,10 ;khai tao bx = 10 de chia lay so du
0045     mov ax,x ;khai tao ax = x
0046     mov cx,0
0047     deni:
0048         mov dx,0 ;khai tao phan du = 0
0049         div bx ;chia 10 lay du;
0050

```

```

emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help

new open examples save compile emulate calculator convertor options help about

41
42      tieptuc:
43          inc si ;tang bien si len 1 don vi
44          loop while
45
46      mov bx,10; khoi tao bx = 10 de chia lay so du
47      mov ax,x; khoi tao ax = x
48      mov cx,0;
49      den1:
50          mov dx,0; khoi taophan du = 0
51          div bx ;chia 10 lay du;
52          push dx; lay dx la phan tu day vao ngan xep
53          inc cx; tang cx len 1 don vi
54          cmp ax,0; so sanh ax phan thuong khac 0 thi tiep tục
55          jnz den1
56      den2:
57          pop dx; chua so du top trong phep chia
58          add dx,'0'; cong dx voi ky tu '0' de chuyen so thanh not ky tu
59          int 21h
60          loop den2
61
62      ;in ra tong so luong cac so chia het cho 11;
63      lea dx,msg2;
64      mov ah,9
65      int 21h
66
67      mov bx,10; khoi tao bx = 10 de chia lay so du
68      mov ax,y; khoi tao ax = y
69      mov cx,0;
70
71      check1:
72          mov dx,0; khoi taophan du = 0
73          div bx ;chia 10 lay du;
74          push dx; lay dx la phan tu day vao ngan xep
75          inc cx; tang cx len 1 don vi
76          cmp ax,0; so sanh ax phan thuong khac 0 thi tiep tục
77          jnz check1
78      check2:
79          pop dx; chua so du top trong phep chia
80          add dx,'0'; cong dx voi ky tu '0' de chuyen so thanh not ky tu
81          mov ah,2; su dung han ah2 de in ra 1 ky tu
82          int 21h
83          loop check2
84
85      mov ah,4
86      int 21h
87
88      main endp
89
90
line: 89 col: 30 drag a file here to open

```

**Câu 16:**



```

10 .code
11 main proc
12 mov ax,@data
13 mov ds,ax
14 ;in thong bao nhap so thu nhat
15 Page 34
16 lea dx,tb1
17 mov ah,9
18 int 21h
19 nhap1:
20 mov ah,1
21 int 21h
22 cmp al,13 ;so sanh ky tu vua nhap voi 13
23 je nhap2 ;neu bang nhap so thu 2
24 sub al,30h ;doi ky tu sang so
25 mov ah,0 ;xoa bit cao
26 mov cx,ax ;cat so vua nhap vao cx
27 mov ax,so1 ;dua bien sf 1 vao kiou byte de chuan bi nhann vao 10
28 mov bx,10 ;gan bx =10
29 mul bx ;nhin ax voi 10
30 add ax,cx ;cng ket qua vua nhan voi so vua nhap ket qua c't vao ax
31 mov so1,ax ; cat ket qua vao bien sf1
32 jmp nhap1
33 nhap2:
34 lea dx,tb2 ;hin thng bpo nhp sf thu 2
35 mov ah,9
36 int 21h
37 nhap: mov ah,1 ;nhap sf thu 2
38 int 21h
39 cmp al,13 ;so snh k2 tu vua nhp voi 13
40 je tinhcong ;neu bang th tnh tng
41 sub al,30h ;chuyrn k2 tu sang dang sf
42 mov ah,0 ;xob bst cao
43 mov cx,ax ;c't ket qua vua nhap vao cx
44 mov ax,so2 ;dua bien sf 2 vao kiou byte
45 mov bx,10 ;gan bx=10
46 mul bx ;nhin ket qua vua nhap voi 10
47 add ax,cx ;cng ket qua vua nhin vao sf vua nhp
48 mov so2,ax ;c't ket qua vao bien sf 2
49 jmp nhap
50 tinhcong:
51 mov dx,tong
52 mov ax,so1 ;dua bien sf 1 ra thanh ghi ax
53 mov bx,so2 ;dua bien sf 2 ra thanh ghi bx
54 add ax,bx ;cng ax voi bx ket qua cat vao ax
55 mov tong,ax ;dua ket qua tu ax vao bien tng
56 inso: mov ah,9 ;hin thng bpo in tng
57 lea dx,tb3
58 int 21h
59 mov ax,tong ;dua ket qua trongv bien tng ra thanh ghi ax
60 mov dx,0 ;xoa bit cao dx
61 mov bx,10 ;gn bx=10
62 mov cx,0 ;khai tao bien dm
63 chia: div bx ;l'y ket qua chia cho 10
64 push dx ;du o dx d'y vao ngan xrp
65 inc cx ;tang bien dm
66 cmp ax,0 ;so sanh thuong voi 0
67 je hienkq ;neu bang th hin ket qua
68 xor dx,dx ;xoa bit cao trong dx
69 jmp chia
70 hienkq: pop dx ;l'y du trong ngan xrp ra khai dx
71 add dl,30h ;chuyrn sf thanh dang k2 tu
72 mov ah,2 ;in tng
73 int 21h
74 loop hienkq
75 ra: mov ah,4ch
76 int 21h
77 Main endp
78 End mai

```

Câu 17:





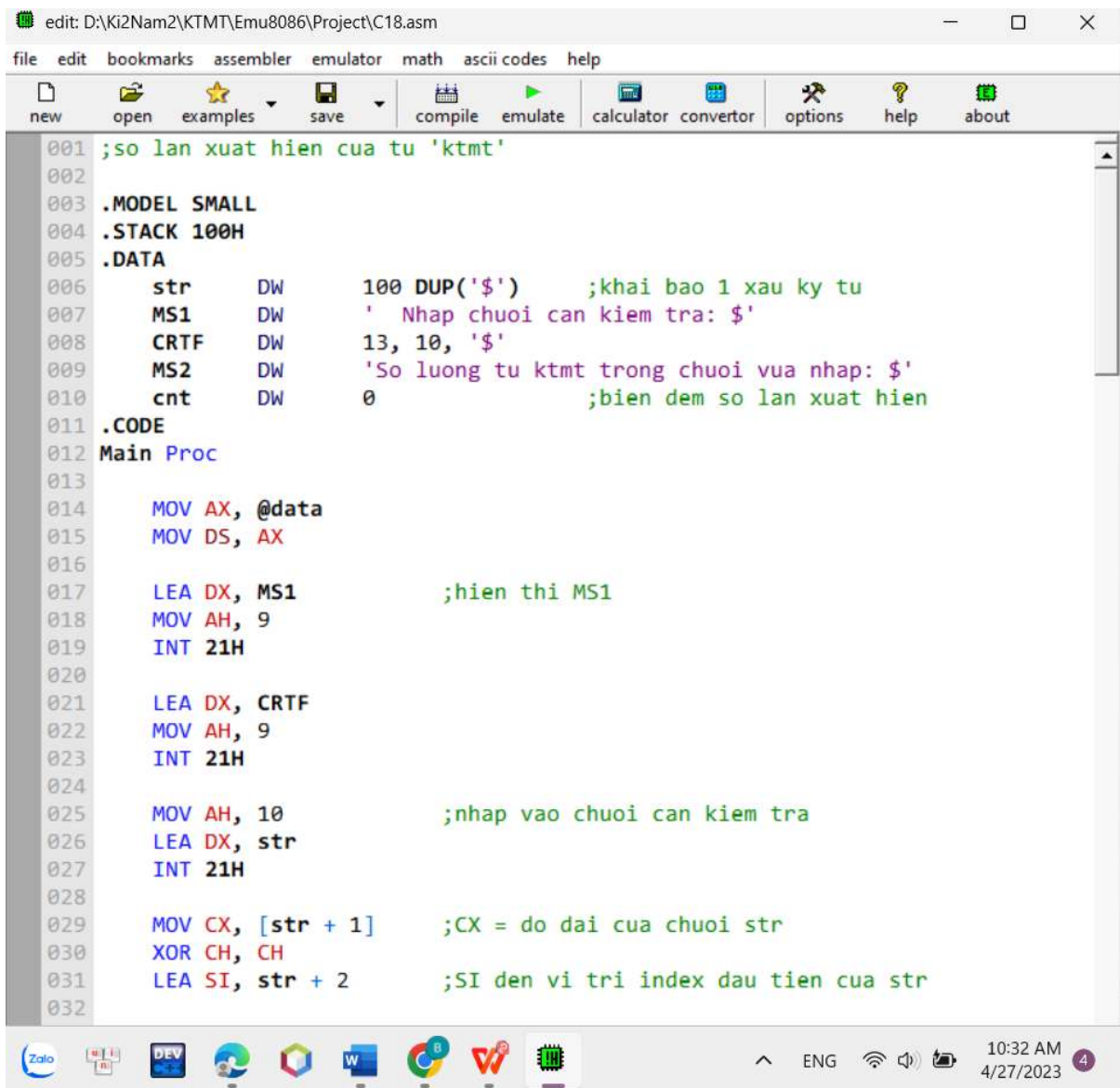
```

edit: C:\EMU0808\MySource\17.asm
file edit bookmarks assembler help

NEW OPEN SAVE ASSEMBLE RUN

064      INT 21H
065  MAIN ENDP
066
067  INPUT PROC                                :NHAP VAO SO DUOI DANG KI TU VA CHUYEN SANG SO
068
069
070      MOV AH, 10                            :NHAP VAP XAU STR
071      LEA DX, str
072      INT 21H
073
074      LEA SI, str + 2                        :BAT DAU CUA CHUOI KI TU
075      MOV CX, [str + 1]                    :DO DAI STR
076      MOV CH, 0                            :CH = 0
077      MOV AX, 0
078      MOV TMP2, AX
079
080  Lap2:
081      MOV BX, 0
082      MOV AX, TMP2                          :DUA GIA TRI TMP2 VE AX
083      MOV BX, [SI]                          :KI TU THU SI TRONG XAU
084      MOV BH, 0
085      SUB BX, '0'                            :CHUYEN SANG SO
086
087      MUL muoi                              :AX = AX * 10
088      ADD AX, BX                            :AX = AX + BX
089      MOV TMP2, AX                          :LUU VAO TMP2
090      INC SI                                :SI = SI + 1
091      LOOP Lap2
092
093      RET
094  INPUT ENDP
095
096  OUTPUT PROC
097
098      LEA DX, TB3                            :HIEN THI THONG BAO 3
099
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2
```

## Câu 18:



The screenshot shows an assembly editor window titled "edit: D:\Ki2Nam2\KTMT\Emu8086\Project\C18.asm". The menu bar includes "file", "edit", "bookmarks", "assembler", "emulator", "math", "ascii codes", and "help". The toolbar contains icons for "new", "open", "examples", "save", "compile", "emulate", "calculator", "convertor", "options", "help", and "about".

The assembly code is as follows:

```
001 ;so lan xuat hien cua tu 'ktmt'
002
003 .MODEL SMALL
004 .STACK 100H
005 .DATA
006     str      DW      100 DUP('$')      ;khai bao 1 xau ky tu
007     MS1      DW      '  Nhap chuoi can kiem tra: $'
008     CRTF     DW      13, 10, '$'
009     MS2      DW      'So luong tu ktmt trong chuoi vua nhap: $'
010     cnt      DW      0                  ;bien dem so lan xuat hien
011 .CODE
012 Main Proc
013
014     MOV AX, @data
015     MOV DS, AX
016
017     LEA DX, MS1          ;hien thi MS1
018     MOV AH, 9
019     INT 21H
020
021     LEA DX, CRTF
022     MOV AH, 9
023     INT 21H
024
025     MOV AH, 10           ;nhap vao chuoi can kiem tra
026     LEA DX, str
027     INT 21H
028
029     MOV CX, [str + 1]    ;CX = do dai cua chuoi str
030     XOR CH, CH
031     LEA SI, str + 2      ;SI den vi tri index dau tien cua str
032
```

The Windows taskbar at the bottom shows the system clock as 10:32 AM on 4/27/2023, along with various application icons and system status icons.

```

031
032 | LAP:
033     ;kiem tra lan luot cac ki tu 'k','t','m','t' dung canh nhau
034
035     MOV DL, [SI]           ;so sanh voi ki tu 'k'
036     CMP DL, 'k'
037     JNE Continue          ;neu khong phai jump toi Continue
038     INC SI                ;SI++
039     DEC CX                ;CX--
040     CMP CX, 0             ;neu CX = 0 thi jump den Stop
041     JE Stop
042     ;so sanh voi ki tu 't'
043     MOV DL, [SI]
044     CMP DL, 't'
045     JNE Continue
046     INC SI
047     DEC CX
048     CMP CX, 0
049     JE Stop
050     ;so sanh voi ki tu 'm'
051     MOV DL, [SI]
052     CMP DL, 'm'
053     JNE Continue
054     INC SI
055     DEC CX
056     CMP CX, 0
057     JE Stop
058     ;so sanh voi ki tu 't'
059     MOV DL, [SI]
060     CMP DL, 't'
061     JNE Continue
062     INC SI
063     DEC CX
064     CMP CX, 0             ;neu CX = 0 thi ktmt la chuoai cuoi cua day
065     JE Cong               ;nen jump toi Cong
066

```

```

-- Stop
;neu jump den Stop
MOV DL, [SI]           ;kiem tra dau cach dam bao ktmt la mot tu riêng biet
CMP DL, ' '
JNE Continue

Cong:

```

drag a file here to open

```

070
071 Cong:
072     MOV AX, 1           ;Cnt ++
073     ADD Cnt, AX
074     CMP CX, 0
075     JE Stop
076
077 Continue:               ;tiếp tục duyệt lại từ kí tự k
078     INC SI              ;SI ++
079     DEC CX              ;CX--
080     CMP CX, 0
081     JNE LAP             ;jump to Lap
082 Stop:
083
084     CALL Output          ;jump to Output
085
086     MOV AH, 4CH          ;kết thúc chương trình
087     INT 21H
088
089 Main Endp
090

```



## Câu 19:

The image shows two screenshots of the emu8086 assembler and microprocessor emulator 4.06 interface. The top screenshot displays the assembly code for a program that compares two strings, 'Xau A' and 'Xau B'. The code includes data declarations, variable initialization, and logic for finding the end of the strings and comparing them byte by byte. The bottom screenshot shows the continuation of the code, including the 'outer\_loop' and 'inner\_loop' labels, and the final program termination instructions.

```
0001 .model small
0002 .stack 100h
0003
0004 .data
0005     0 db 100 dup(0)
0006     0 db 100 dup(0)
0007     n du ?
0008     m du ?
0009     i du ?
0010     j du ?
0011     found db ?
0012     msg_found db 'Xau A la xau con cua xau A', 0
0013     msg_not_found db 'Xau B khong phai la xau con cua xau A', 0
0014     result_msg db 00h, 'Vi tri cua xau B trong xau A la ', 0
0015
0016 .code
0017     mov ax, 0data
0018     mov dx, ax
0019
0020     ;Nhap xau A
0021     mov ah, 0ah
0022     int 21h
0023
0024     ;Nhap xau B
0025     mov ah, 0ah
0026     int 21h
0027
0028     ; Tinh dai cua xau A va B
0029     mov si, 0
0030     mov cx, 0
0031     strlen_A:
0032     cmp byte ptr [si], 0
0033     je strlen_A_done
0034     inc si
0035     inc cx
0036     jmp strlen_A
0037
0038     strlen_A_done:
0039     mov n, cx
0040
0041     mov si, 0
0042     mov cx, 0
0043     strlen_B:
0044     cmp byte ptr [si], 0
0045     je strlen_B_done
0046     inc si
0047     inc cx
0048     jmp strlen_B
0049
0050     strlen_B_done:
0051     mov m, cx
0052
0053     ; Kiem tra xau B co phai la con cua xau A khong
0054     mov di, 0
0055     mov found, 0
0056     mov pc, 0
0057     mov cx, n
0058     cmp cx, 0
0059     jle not_found
0060
0061     outer_loop:
0062     mov cx, 0
0063     mov i, 0
0064     add i, 1
0065
0066     inner_loop:
0067     cmp si, 0
0068     je found_it
0069     mov si, byte ptr [0 + di]
0070     cmp al, byte ptr [B + di]
0071     jne next_outer
0072     inc si
0073     inc i
0074     jmp inner_loop
0075
0076     found_it:
0077     mov found, 1
0078     mov pc, i
0079     jmp end
0080
0081     next_outer:
0082     add di, 1
0083     dec cx
0084     cmp cx, 0
0085     jge outer_loop
0086
0087     not_found:
0088     mov found, 0
0089
0090     end:
0091     ; In ket qua
0092     mov ah, 09h
0093     cmp found, 1
0094     je found_msg
0095     jmp not_found_msg
0096
0097     found_msg:
0098     mov dx, found_msg
0099     jmp print_msg
0100
0101     not_found_msg:
0102     mov dx, not_found_msg
0103     jmp print_msg
0104
0105     print_msg:
0106     int 21h
0107
0108     mov ah, 09h
0109     cmp found, 1
0110     je end_program
0111     mov ax, pc
0112     mov dx, result_msg
0113     jmp print_result
0114
0115     print_result:
0116     int 21h
0117
0118     end_program:
0119     mov ah, 4Ch
0120     int 21h
0121
0122 end
0123
```



```

new open examples save compile emulate calculator convertor options help about
0001 .model small
0002 .stack 100h
0003
0004 .data
0005     db 100 dup('5')
0006     db 100 dup('5')
0007     a db 7
0008     a db 7
0009     l db 7
0010     j db 7
0011     found db ?
0012     pos db ?
0013     msg_found db 'Xau B la con con cua xau A', '5'
0014     not_found_msg db 'Xau B khong phai la con con cua xau A', '5'
0015     result_msg db 00h, 'Vi tri cua xau B trong xau A la: 5'
0016
0017 .code
0018     mov ax, 2data
0019     mov dx, ax
0020
0021     ;Nhap xau A
0022     mov ah, 0Ah
0023     lea dx, a
0024     int 21h
0025
0026     ;Nhap xau B
0027     mov ah, 0Ah
0028     lea dx, l
0029     int 21h
0030
0031     ;Tinh do dai cua xau A va B
0032     lea si, a
0033     mov cx, 0
0034     strlen_A:
0035         cmp byte ptr [si], '5'
0036         je strlen_A_done
0037         inc cx
0038         inc si
0039         jmp strlen_A
0040     strlen_A_done:
0041         mov a, cx
0042
0043     lea si, l
0044     mov cx, 0
0045     strlen_B:
0046         cmp byte ptr [si], '5'
0047         je strlen_B_done
0048         inc cx
0049         inc si
0050         jmp strlen_B
0051     strlen_B_done:
0052         mov b, cx
0053
0054     ;Kiem tra xau B co phai la con cua xau A khong
0055     lea di, 0
0056     mov found, 0

```

```

0090
0091 Output Proc
0092
0093     LEA DX, CRTF                ;hien thi xuong dong
0094     MOV AH, 9
0095     INT 21H
0096
0097     LEA DX, MS2
0098     MOV AH, 9
0099     INT 21H
0100
0101     MOV AX, Cnt
0102     MOV BX, 10
0103     MOV CX, 0
0104     chia:
0105     MOV DX, 0
0106     DIV BX
0107     PUSH DX
0108     INC CX
0109     CMP AX, 0
0110     JE hienthi
0111     JMP chia
0112     hienthi:
0113     POP DX
0114     ADD DL, 48
0115     MOV AH, 2
0116     INT 21H
0117     DEC CX
0118     CMP CX, 0
0119     JNE hienthi
0120     ret
0121 Output Endp
0122
0123

```

## Câu 20:

```
emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

0094
0095 .model small
0096 .stack 100h
0097 .data
0098 msg1 db "Hay Nhap Vao Chuoi A: $"
0099 msg2 db 10,13,"So lan xuất hiện của chuỗi 'ktnt' trong chuỗi A là: $"
0100 strA db 256 dup('$')
0101
0102 x dw ?
0103 .code
0104
0105 main proc
0106     mov ax,0data
0107     mov ds,ax ;khai tạo thanh ghi ds
0108
0109     lea dx,msg1 ;thông báo nhập chuỗi
0110     mov ah,9 ;sử dụng ah9 để in ra thông báo
0111     int 21h
0112
0113     ;Nhập chuỗi A
0114     lea dx,strA
0115     mov ah,10 ;sử dụng ah10 để nhập chuỗi A
0116     int 21h
0117
0118     xor cx,cx;
0119     lea si,strA+2; đưa con trỏ đến với địa chỉ của mảng
0120     mov cl,[si]
0121     while:
0122         mov dl,[si] ;lưu ký tự vào dl
0123         cmp dl,'k' ;so sánh ký tự đã lưu ở dl với ký tự 'k'
0124         jne continue ;nếu không thì nhảy đến continue
0125         inc si ;tăng si lên một đơn vị
0126         dec cx ;giảm cx
0127         cmp cx,0 ;so sánh cx với 0;
0128         je print
0129
0130         mov dl,[si] ;lưu ký tự vào dl
0131         cmp dl,'t' ;so sánh ký tự đã lưu ở dl với ký tự 't'
0132         jne continue ;nếu không thì nhảy đến continue
0133         inc si ;tăng si lên một đơn vị
0134         dec cx ;giảm cx
0135         cmp cx,0 ;so sánh cx với 0;
0136         je print
0137
0138         mov dl,[si] ;lưu ký tự vào dl
0139         cmp dl,'n' ;so sánh ký tự đã lưu ở dl với ký tự 'n'
0140         jne continue ;nếu không thì nhảy đến continue
0141         inc si ;tăng si lên một đơn vị
0142         dec cx ;giảm cx
0143         cmp cx,0 ;so sánh cx với 0;
0144         je print
0145
0146     main endp
```

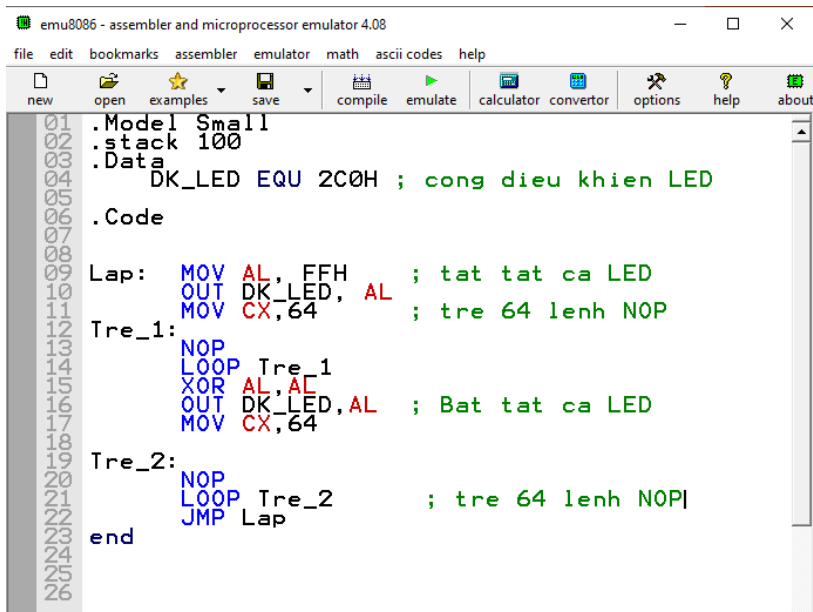
```
emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about

0146
0147     mov dl,[si]
0148     cmp dl,' '
0149     jne continue
0150
0151     count:
0152     mov ax,1 ;khai tạo ax = 1
0153     add x,ax ; x = x + ax
0154     cmp cx,0
0155     je print
0156
0157     continue:
0158     inc si
0159     dec cx
0160     cmp cx,0
0161     jne while
0162
0163     print:
0164     ; in ra thông báo số lần xuất hiện
0165     mov ah,9
0166     lea dx,msg2
0167     int 21h
0168
0169     ;in ra số lần xuất hiện
0170
0171     mov bx,10; khai tạo bx = 10
0172     mov ax,x; khai tạo ax = x
0173     mov cx,0
0174
0175     check1:
0176     mov dx,0; khai tạo phần dư 0
0177     div bx; chia 10 lấy dư
0178     push dx; đẩy dx vào ngăn xếp
0179     inc cx; tăng cx lên 1 đơn vị
0180     cmp ax,0; so sánh phần thương ! 0
0181     jnz check1
0182
0183     check2:
0184     pop dx; lấy dx ra khỏi ngăn xếp
0185     add dx,'0'; đổi số sang ký tự
0186     mov ah,2; sử dụng ah2 để in ra ký tự
0187     int 21h
0188     loop check2
0189
0190     mov ah,4
0191     int 21h
0192
0193     main endp
```



## II. Lập trình điều khiển đèn Led, điều khiển nhiệt kế

1. Chương trình hợp ngữ để 8 LED nối với cổng ra 2C0H sáng rồi tắt led. đèn được bật sáng nếu bit điều khiển tương ứng nhận giá trị 0. Ngược lại khi bit điều khiển bằng 1 thì đèn sẽ tắt:



```
01 .Model Small
02 .stack 100
03 .Data
04     DK_LED EQU 2C0H ; cổng điều khiển LED
05
06 .Code
07
08 Lap:  MOV AL, FFH      ; tắt tất cả LED
09       OUT DK_LED, AL
10       MOV CX, 64      ; chờ 64 lệnh NOP
11       Tre_1:
12           NOP
13           LOOP Tre_1
14           XOR AL, AL
15           OUT DK_LED, AL ; Bật tất cả LED
16           MOV CX, 64
17       Tre_2:
18           NOP
19           LOOP Tre_2    ; chờ 64 lệnh NOP
20       JMP Lap
21
22 end
23
24
25
26
```

2. Lập trình hợp ngữ điều khiển thiết bị ảo Nhiệt kế và bếp (Therometer & Heater) sử dụng 2 cổng:

- Cổng điều khiển số 127 để nhận byte điều khiển bếp: 0 để tắt bếp, 1 để bật bếp.
- Cổng dữ liệu số 125 để đọc dữ liệu là nhiệt độ đo được bởi nhiệt kế.

```

emu8086 - assembler and microprocessor emulator 4.08
file edit bookmarks assembler emulator math ascii codes help
new open examples save compile emulate calculator convertor options help about
01 .Model Small
02 .stack 100
03 .Data
04
05
06 .Code
07
08 start:
09     in al,125 ;doc nhiet do hien tai
10     cmp al,60 ; so sanh nhiet do hien tai voi 60 do
11     jl low ; neu thap hon, nhay den nhan low
12
13     cmp al,80 ; so sanh nhiet do hien tai voi 80 do
14     jle ok ; nhay den ok neu <=
15     jg high ; nhay den high neu >
16
17 low:
18     mov al,1
19     out 127, al ; tang nhiet
20     jmp ok
21
22 high:
23     mov al,0
24     out 127, al ; giam nhiet (tat)
25
26 ok: jmp start ; lap lai tu dau
27
28
29 end
30
31
32

```

## Phần 4: Giải mã địa chỉ bộ nhớ

(by Hoàng Anh Vũ)

### 1. Giải mã địa chỉ bộ nhớ sử dụng mạch logic cơ bản:

(chỉ sử dụng các cổng NOT, NAND, ...)



#### Các bước xây dựng:

**Bước 1** : Xác định số bit cho địa chỉ nội bộ chip và mạch giải mã.

**Bước 2** : Phân giải địa chỉ cơ sở của các chip.

**Bước 3** : Vẽ sơ đồ bit.

**Bước 4** : Vẽ hình mạch giải mã.

**Ví dụ** : Xây dựng mạch giải mã địa chỉ cho 1 bộ nhớ ROM có dung lượng 4KB bằng phương pháp sử dụng mạch logic cơ bản; Biết rằng kích thước 1 vi mạch nhớ là 2Kx8 và địa chỉ cơ sở là 03800H.

□ Phân tích đề:

- + dung lượng bộ nhớ ROM :  $C = 4KB$ .
- + kích thước một vi mạch nhớ :  $IC = 2K \times 8$
- + địa chỉ cơ sở :  $DCCS = 03800H$

**Bài làm :**

**Bước 1 : Xác định số bit cho địa chỉ nội bộ chip và mạch giải mã.**

Ta có : Chip nhớ IC 2Kx8 chiếm không gian  $2KB = 2^1 \times 2^{10} = 2^{11} B$

□ cần 11 bit địa chỉ nội bộ chip ( $A_0 - A_{10}$ )

Vì xử lý 8086 có 20 bit địa chỉ nên ta có  $20 - 11 = 9$

□ cần 9 bit cho mạch giải mã.

**Bước 2 : Phân giải địa chỉ cơ sở của các chip.**

Ta có:

Bộ nhớ có dung lượng 4KB và mỗi chip nhớ có dung lượng 2KB.

□ cần phải có 4KB :  $2KB = 2$  chip nhớ.

Mà  $2KB = 2^{11} B = 0000\ 0000\ 1000\ 0000\ 0000\ (B) = 00800\ (H)$ .

□ Dung lượng của một chip nhớ là 00800(H).

Phân giải địa chỉ cơ sở: Địa chỉ cuối = Địa chỉ đầu + Dung lượng – 1.

+ Địa chỉ của IC 1: Từ **03800H** đến (**03800H + 00800H – 1H**)

= Từ **03800H** đến (**04000H – 1H**)

= Từ **03800H** đến **03FFFH**.

+ Địa chỉ của IC 2: Từ **04000H** đến **047FFH**.

**Bước 3 : Vẽ sơ đồ bit.**

		9 bit cao									11 bit địa chỉ nội bộ chip ( $A_0 - A_{10}$ )										
IC1	<b>03800H</b>	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
IC2	<b>04000H</b>	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		A <sub>19</sub>	A <sub>18</sub>	A <sub>17</sub>	A <sub>16</sub>	A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>	A <sub>12</sub>	A <sub>11</sub>	A <sub>10</sub>	A <sub>9</sub>	A <sub>8</sub>	A <sub>7</sub>	A <sub>6</sub>	A <sub>5</sub>	A <sub>4</sub>	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>

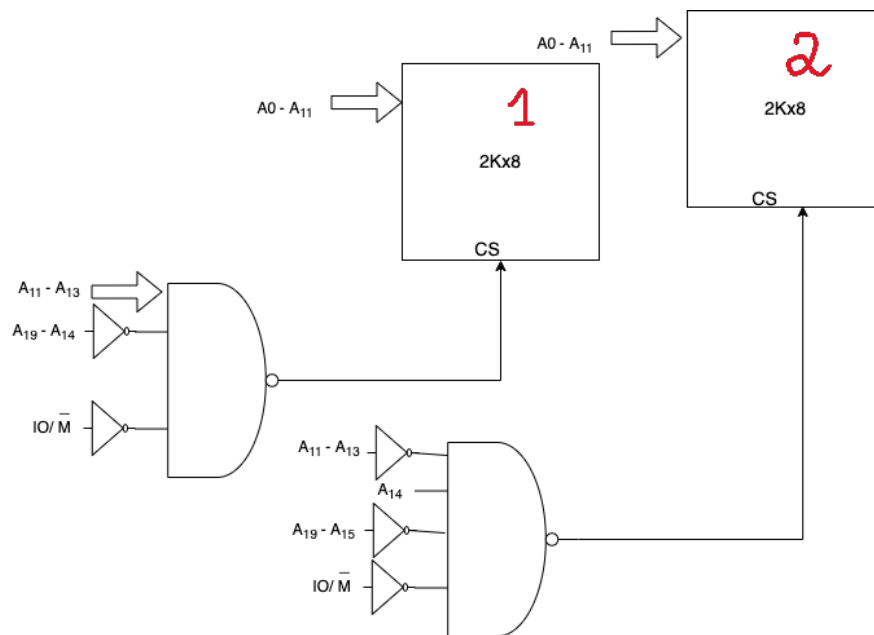
Giải thích :

03800 H = 000000**11**100000000000 B

04000 H = 00000**1**0000000000000000 B

Ta phân biệt 9 bit cao dành cho mạch giải mã ( $A_{19} - A_{10}$ ) và 11 bit dành cho địa chỉ nội bộ chip ( $A_0 - A_{10}$ ).

**Bước 4 : Vẽ mạch giải mã.**



Giai thích :

+ Các bit 0 được đưa vào qua cổng **NOT** trong khi các bit 1 được truyền thẳng vào cổng NAND.

+ Đầu ra từ cổng NAND đi vào đi vào **IC1** và **IC2** tương ứng như hình vẽ.

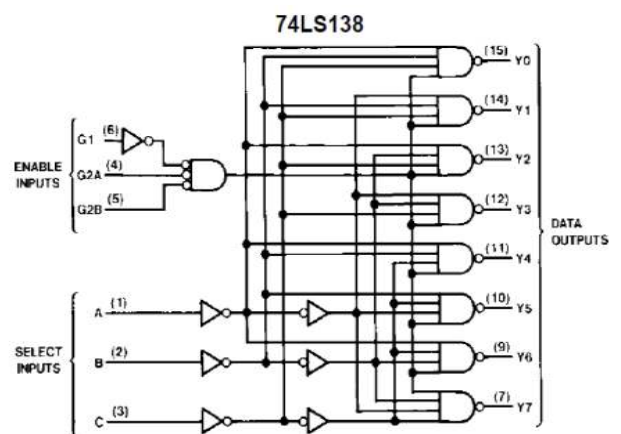
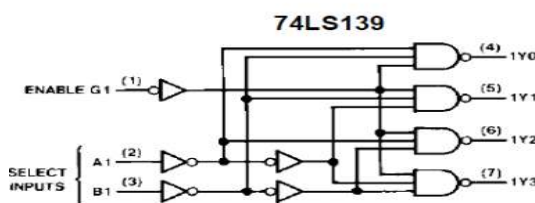
+  $IO / \overline{M}$  : tín hiệu CPU chọn làm việc với thiết bị vào ra hay bộ nhớ. Qua cổng NOT nhận  $IO / \overline{M} = 0$   $\square$  CPU chọn làm việc với bộ nhớ.

## 2. Giải mã địa chỉ bộ nhớ sử dụng mạch tích hợp 74LS138 và 74LS139.

Mình họa về 74LS138 và 74LS139:

74LS138 mạch giải mã  $3 \rightarrow 8$

74LS139 mạch giải mã  $2 \rightarrow 4$



**Ví dụ:** Xây dựng bộ giải mã địa chỉ bộ nhớ có dung lượng 8KB có địa chỉ bắt đầu là 0F800H với các chip nhớ có dung lượng 2Kx8. Chỉ được sử dụng các chip giải mã địa chỉ **74LS139** (Là các chip giải mã có 2 đầu vào và 4 đầu ra).

$\square$  Phân tích đề:

+ dung lượng bộ nhớ ROM : **C = 8KB.**

+ kích thước một vi mạch nhớ :  $IC = 2K \times 8$

+ địa chỉ cơ sở :  $DCCS = 0F800H$

## Bài làm:

### Bước 1 : Xác định số bit cho địa chỉ nội bộ chip và mạch giải mã.

Chip nhớ IC  $2K \times 8$  chiếm không gian  $2KB = 2^1 \times 2^{10} = 2^{11}$  B

□ Cần 11 bit địa chỉ nội bộ chip ( $A_0 - A_{10}$ ).

Vi xử lý 8086 có 20 bit địa chỉ nên ta có  $20 - 11 = 9$

□ cần 9 bit cho mạch giải mã.

### Bước 2 : Phân giải địa chỉ cơ sở của các chip.

Bộ nhớ có dung lượng 8KB và mỗi chip nhớ có dung lượng 2KB □ cần  $8/2 = 4$  IC.

Mà  $2KB = 2^{11} = 0000\ 0000\ 1000\ 0000\ 0000(B) = 00800(H)$  □ Dung lượng của một chip nhớ là 00800(H).

Ta có : Địa chỉ cuối = Địa chỉ đầu + Dung lượng - 1. (tính toán tương tự phần 1)

+ Địa chỉ của IC 1: Từ  $0F800H$  đến  $0FFFFH$ .

+ Địa chỉ của IC 2: Từ  $10000H$  đến  $107FFH$ .

+ Địa chỉ của IC 3: Từ  $10800H$  đến  $10FFFH$ .

+ Địa chỉ của IC 4: Từ  $11000H$  đến  $117FFH$ .

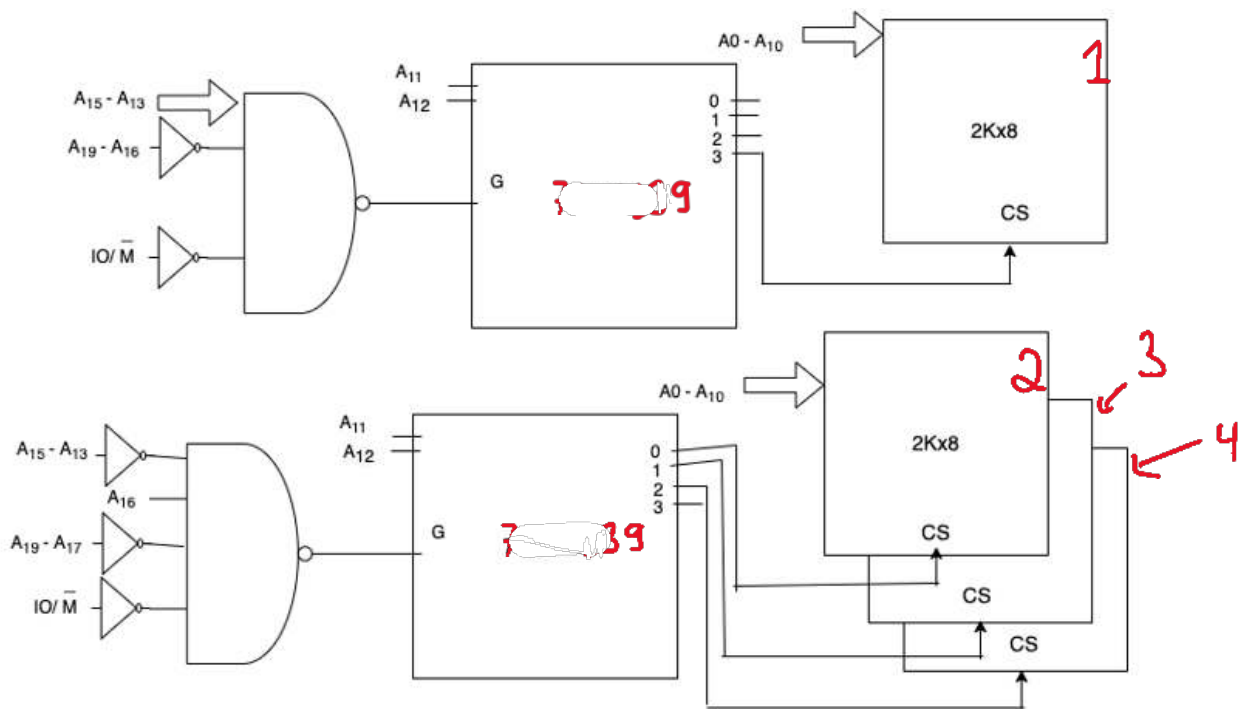
### Bước 3 : Vẽ sơ đồ bit.

		7 bit cao							2 bit vào		11 bit địa chỉ nội bộ chip ( $A_0 - A_{10}$ )										
IC1	<i>0F800H</i>	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
IC2	<i>10000H</i>	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC3	<i>10800H</i>	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
IC4	<i>11000H</i>	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
		A <sub>19</sub>	A <sub>18</sub>	A <sub>17</sub>	A <sub>16</sub>	A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>	A <sub>12</sub>	A <sub>11</sub>	A <sub>10</sub>	A <sub>9</sub>	A <sub>8</sub>	A <sub>7</sub>	A <sub>6</sub>	A <sub>5</sub>	A <sub>4</sub>	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>

Giải thích :

Ta phân biệt 9 bit cao dành cho mạch giải mã ( $A_{19} - A_{10}$ ), trong đó có 2 bit vào ( $A_{11}$  và  $A_{12}$ ) cho  $74LS139$  và 11 bit dành cho địa chỉ nội bộ chip ( $A_0 - A_{10}$ ).

### Bước 4 : Vẽ mạch giải mã.



Giải thích :

+ Xét 7 bit cao ( $A_{13} - A_{19}$ ) cần 2 chip giải mã 74LS139:

- Một cho 0F800H
- Một cho 10000H, 10800H, 11000H (vì 7 bit giống nhau)

+ Các bit 0 được đưa vào qua cổng **NOT** trong khi các bit 1 được truyền thẳng vào cổng NAND.

+  $IO / \overline{M}$  : tín hiệu CPU chọn làm việc với thiết bị vào ra hay bộ nhớ. Qua cổng NOT nhận  $IO / \overline{M} = 0$   $\square$  CPU chọn làm việc với bộ nhớ.

+  $A_{11}$  và  $A_{12}$  dành cho đầu vào của **74LS139**.

**Ví dụ :** Xây dựng bộ giải mã địa chỉ bộ nhớ có dung lượng 8KB có địa chỉ bắt đầu là 0F800H với các chip nhớ có dung lượng 2Kx8. Chỉ được sử dụng các chip giải mã địa chỉ **74LS138** (Là các chip giải mã có 3 đầu vào và 8 đầu ra).

$\square$  Phân tích đề:

- + dung lượng bộ nhớ ROM :  **$C = 8KB$** .
- + kích thước một vi mạch nhớ :  **$IC = 2Kx8$**
- + địa chỉ cơ sở :  **$DCCS = 0F800H$**

**Bước 1 :** Xác định số bit cho địa chỉ nội bộ chip và mạch giải mã.

Chip nhớ IC 2Kx8 chiếm không gian  $2KB = 2^1 \times 2^{10} = 2^{11} B$

□ Cần 11 bit địa chỉ nội bộ chip (A0 – A10).

Vì xử lý 8086 có 20 bit địa chỉ nên ta có  $20 - 11 = 9$

□ cần 9 bit cho mạch giải mã.

### **Bước 2 : Phân giải địa chỉ cơ sở của các chip.**

Bộ nhớ có dung lượng 8KB và mỗi chip nhớ có dung lượng 2KB □ cần  $8/2 = 4$  IC.

Mà 2KB = 211 = 0000 0000 1000 0000 0000(B) = 00800(H) □ Dung lượng của một chip nhớ là 00800(H).

Ta có : Địa chỉ cuối = Địa chỉ đầu + Dung lượng – 1. (tính toán tương tự phần 1)

+ Địa chỉ của IC 1: Từ **0F800H** đến **0FFFFH**.

+ Địa chỉ của IC 2: Từ **10000H** đến **107FFH**.

+ Địa chỉ của IC 3: Từ **10800H** đến **10FFFH**.

+ Địa chỉ của IC 4: Từ **11000H** đến **117FFH**.

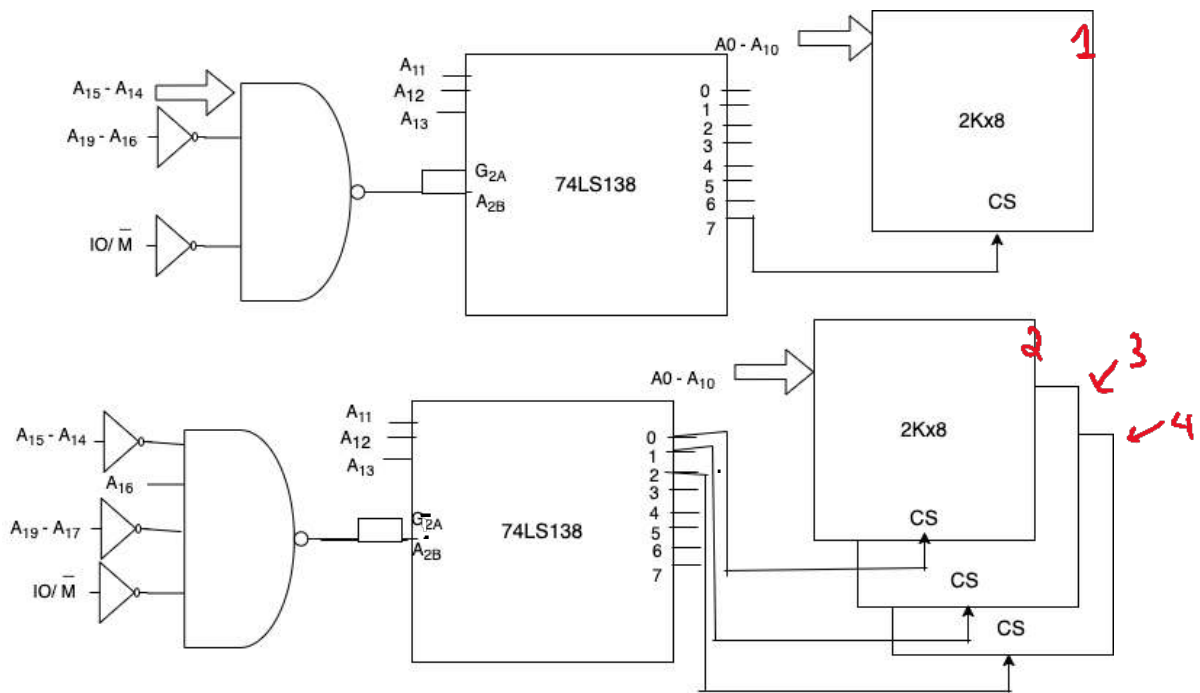
### **Bước 3 : Vẽ sơ đồ bit.**

		6 bit cao						3 bit vào			11 bit địa chỉ nội bộ chip (A <sub>0</sub> – A <sub>10</sub> )										
IC1	<b>0F800H</b>	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
IC2	<b>10000H</b>	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
IC3	<b>10800H</b>	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
IC4	<b>11000H</b>	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
		A <sub>19</sub>	A <sub>18</sub>	A <sub>17</sub>	A <sub>16</sub>	A <sub>15</sub>	A <sub>14</sub>	A <sub>13</sub>	A <sub>12</sub>	A <sub>11</sub>	A <sub>10</sub>	A <sub>9</sub>	A <sub>8</sub>	A <sub>7</sub>	A <sub>6</sub>	A <sub>5</sub>	A <sub>4</sub>	A <sub>3</sub>	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>

Giải thích :

Ta phân biệt 9 bit cao dành cho mạch giải mã (A19 – A10), trong đó có 3 bit vào (A11, A12 và A13) cho **74LS138** và 11 bit dành cho địa chỉ nội bộ chip (A0 – A10).

### **Bước 4 : Vẽ mạch giải mã.**



Giải thích:

Xét 6 bit cao (A14 – A19) cần 2 chip giải mã 74LS138:

- Một cho 0F800H
- Một cho 10000H, 10800H, 11000H (vì 6 bit giống nhau).

+ Các bit 0 được đưa vào qua cổng **NOT** trong khi các bit 1 được truyền thẳng vào cổng NAND.

+  $IO / \overline{M}$  : tín hiệu CPU chọn làm việc với thiết bị vào ra hay bộ nhớ. Qua cổng NOT nhận  $IO / \overline{M} = 0$  □ CPU chọn làm việc với bộ nhớ.

+ A11, A12 và A13 dành cho đầu vào của **74LS138**.

**So sánh ưu điểm và nhược điểm giữa 2 cách xây dựng bộ giải mã địa chỉ bộ nhớ.**

	<b>Mạch logic cơ bản</b>	<b>Mạch tích hợp</b>
<b>Ưu điểm</b>	<ul style="list-style-type: none"> <li>+ Cho phép tạo mạch giải mã đầy đủ.</li> <li>+ Tương đối đơn giản rẻ tiền khi chỉ cần 1 hoặc ít đầu ra</li> </ul>	<ul style="list-style-type: none"> <li>+ Cho phép tạo mạch giải mã đầy đủ.</li> <li>+ Cho phép tạo mạch giải mã chấp nhận một số hạn chế đầu vào và tạo ra một số hạn chế tín hiệu chọn mạch đầu ra.</li> </ul>



<p><i>Nhược điểm</i></p>	<p>+ Cộng kênh khi cần giải mã cho nhiều đầu ra do số mạch tăng nhanh.</p>	<p>+ Không thích hợp với mạch giải mã cần chấp nhận một số lượng lớn tín hiệu đầu vào và sinh ra nhiều tín hiệu đầu ra.</p> <p>□ Cần sử dụng bổ sung mạch logic phụ thì mạch tích hợp mới có thể cho phép giải mã đầy đủ.</p>
--------------------------	--	---