

Contents

| | |
|------------------------------------------------------------------------------|----|
| 1; (assembly) Hiển thị lời chào Tiếng Anh và Tiếng Việt trên emu8086 | 8 |
| 2: (assembly) Nhập 1 ký tự và xuất ra màn hình trên emu8086..... | 9 |
| 3. (assembly) nhập kí tự thường và chuyển sang kí tự hoa trên emu8086 | 11 |
| 4. (assembly) Nhập một chuỗi và in ra trên emu8086..... | 12 |
| 5. (assembly) Nhập chuỗi thành chuỗi thường và chuỗi hoa trên emu8086 | 13 |
| 6. (assembly) IN CHUỖI ĐẢO NGƯỢC trên emu8086..... | 15 |
| 7. (assembly) in chuỗi đảo ngược khi điền dấu # trên emu8086 | 16 |
| 8. (assembly) Tìm giá trị lớn nhất của mảng | 17 |
| 16. (assembly) Đếm chiều dài 1 chuỗi trên emu8086 | 18 |
| 9. (assembly) Nhập vào một số và tính giai thừa của số đó trên emu8086 | 20 |
| 10. (assembly) Chuyển một số từ hệ 10 sang hệ 2 trên emu8086 | 21 |
| 11. (assembly) Chuyển một số từ hệ 10 sang hệ 16 trên emu8086 | 23 |
| 12.TỔNG CÁC CHỮ SỐ | 25 |
| 13.UCLN, BCNN: | 27 |
| 14.tổng chia hết cho 7:..... | 30 |
| 15.Tổng 2 số kiểu word..... | 33 |

Kiến thức :

Cấu trúc chương trình dạng EXE:*

.Model <Chế độ bộ nhớ>

.Stack 100h

.Data

<Khai báo dữ liệu đặt tại đây>

.Code

<Thủ tục chính> PROC

<Các lệnh của chương trình đặt tại đây>

<Thủ tục chính> Endp

<Các thủ tục khác đặt tại đây>

END

Cấu trúc bài làm đi thi :

;nhap chuoi truoc , in thong bao và in chuoi sau

.model small

.stack 100

.data

tb1 DB 10,13, 'nhap chuoi: \$'

tb2 DB 10,13, 'chuoi da nhap la: \$';10 xuong dong , 13 lui dau dong*

str DB 100 dup('\$') ;

.code

main proc

mov ax, @data

mov ds,ax ;khởi tạo thanh ghi ds

mov ah,9 ;in một xâu ký tự

lea dx,tb1 ;

int 21h ;in ra

;nhập chuỗi ký tự

mov ah,10 ; 10=0ah

lea dx,str ;trở đến địa chỉ đầu str

int 21h

mov ah,9 ;in một xâu ký tự

lea dx,tb2 ;

int 21h ;in ra

call xuatchuoi

mov ah,4CH

int 21h

main endp

xuatchuoi proc

```

;dua dx chỉ ve phan tu thu 2 cua mang la
; ky tu dau tien duoc nhap vao
lea dx,str +2 ;256,5,h,e,l,l,o
mov ah,9
int 21h ;in na
ret
xuatchuoi endp

```

end

CÁC LỆNH TRAO ĐỔI DỮ LIỆU

| Lệnh | Cú pháp | Chức năng | Ví dụ |
|------|-------------------------|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MOV | MOV đích, gốc | Lệnh gán giá trị của gốc vào đích | MOV AL, BL; AL = BL MOV CX, 123FH; CX = 123FH |
| LEA | LEA đích, gốc | Lệnh gán địa chỉ của gốc vào đích | LEA SI, a; nạp địa chỉ biến a vào thanh ghi SI ; Load Effective Address LEA CX, [BX]; nạp địa chỉ ô nhớ có địa chỉ [DS:BX] vào CX (hay CX = BX) |
| SUB | SUB đích, gốc | Từ hai toán hạng | SUB AL, 74H ;AL = AL - 74H SUB CL, AL; ;CL = CL - AL SUB DL, [SI] ;DL = DL - [DS:SI] |

....

LỆNH NGẮT INT 21H*

- Cú pháp: **INT 21H**
- Chức năng của lệnh dựa theo giá trị của AH

| Ngắt | Chức năng | Thực hiện khi AH=? | Cách emu8086 xử lý | Ví dụ |
|-------------|---------------------------|--------------------|----------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------|
| Ngắt loại 1 | Đọc một ký tự từ bàn phím | AH = 1 | Đọc một ký tự được nhập vào từ bàn phím, AL sẽ lưu mã ASCII của phím vừa nhập. Nếu phím vừa nhập là phím chức năng, AL = 0 | MOV AH, 1; INT 21H; => chương trình sẽ ngừng lại đến khi bạn |

| | | | | |
|-------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|-----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| | | | | nhập vào một phím |
| Ngắt loại 2 | Hiện một kí tự lên màn hình | AH = 2 | Hiện một ký tự có mã ASCII là giá trị của DL lên màn hình | MOV AH, 2; MOV DL, 30h; (30h là mã ASCII của '0') INT 21H; => màn hình sẽ in ra ký tự '0' |
| Ngắt loại 9 | Hiện xâu ký tự | AH = 9 | Hiện xâu ký tự có địa chỉ lệch là giá trị của DX | tb BD 'co lam thi moi co an\$'; MOV AH, 9; LEA DX, tb; INT 21H; => màn hình in ra xâu tb (ko hiện ký tự '\$') |
| Ngắt chương trình | Dừng chương trình | AH = 4 CH | Dừng chương trình | MOV AH, 4CH |
| Hàm 0Ah AH=10 | Nhập một xâu ký tự vào một biến đệm cho trước trong chương trình, biến này phải được chỉ bởi cặp thanh ghi DS:DX. Và biến đệm phải có dạng như sau: -Byte 0: chứa số ký tự tối đa của xâu nhập vào -Byte 1: chứa một trị không (= 0) -Byte 2, byte 3, byte 4, ... chứa một trị rỗng (để trống), để chứa các ký tự sẽ được nhập vào sau này (khi chương trình thực hiện). | | | |

Để có được một biến như trên chương trình phải khai báo biến (tên biến là Xau_Nhap) như sau:

Xau_Nhap DB 256, 0, 256 Dup (' ')

Như vậy Xau_Nhap là một biến kiểu byte, gồm 258 byte. Byte đầu tiên (byte) chứa trị 256, byte 1 chứa trị 0, 256 byte tiếp theo chứa ký tự trống, được sử

dụng để chứa các kí tự sẽ được nhập sau này. Xau_Nhap chứa tối đa 256 kí tự.

Cũng có thể sử dụng hàm 0Ah/21h để nhập một xâu kí tự vào vùng nhớ có địa chỉ xác định trong bộ nhớ.

Sử dụng:

Vào: Ah = 0Ah

DS:DX = <Địa chỉ Segment:Offset của xâu nhập>

Ra: DS:DX không thay đổi

Biến đếm bây giờ có dạng như sau:

- Byte 0: không thay đổi
- Byte 1: chứa tổng số kí tự đã được nhập vào
- Byte 2, byte 3, byte 4, ... chứa các kí tự đã được nhập vào.

Ví dụ 1: Với khai báo biến đếm Xau_Nhap như trên, nếu sau này chương trình nhập vào xâu: “Tin hoc” thì://7

- Byte 0: vẫn chứa số 256
- Byte 1: chứa giá trị 7, đó chính là 7 kí tự trong xâu “Tin hoc”
- Từ byte 2 đến 8 chứa lần lượt các kí tự trong xâu “Tin hoc.”

<https://www.scribd.com/doc/50996286/M%E1%BB%98T-S%E1%BB%90-CH%E1%BB%A8C-N%C4%82NG-C%E1%BB%A6A-NG%E1%BA%AET-21H>

1.MOV

- **Mov** [Toán hạng đích], [Toán hạng nguồn]

Tác dụng: Lấy nội dung (giá trị) của [Toán hạng nguồn] đặt vào [Toán hạng đích]. Nội dung của [Toán hạng nguồn] không bị thay đổi.

2. Inc/ADD/DEC/SUB

- **Inc** [Toán hạng đích]
- **Add** [Toán hạng đích],[Toán hạng nguồn]
- **Dec** [Toán hạng đích]
- **Sub** [Toán hạng đích],[Toán hạng nguồn]

Lệnh **Inc (Increment)**: làm tăng giá trị của [Toán hạng đích] lên 1 đơn vị.

Lệnh **Dec (Decrement)**: làm giảm giá trị của [Toán hạng đích] xuống 1 đơn vị.

Lệnh **Add (Addition)**: lấy giá trị/nội dung của [Toán hạng nguồn] cộng vào giá trị/nội dung của [Toán hạng đích], kết quả này đặt vào lại [Toán hạng đích].

Lệnh **Sub (Subtract)**: lấy giá trị/nội dung của [Toán hạng đích] trừ đi giá trị/nội dung của [Toán hạng nguồn], kết quả này đặt vào lại [Toán hạng đích].

3. Lệnh LOOP

Loop <Nhãn đích>

Tác dụng: Khi gặp lệnh này chương trình sẽ lặp lại việc thực hiện các lệnh sau <Nhãn lệnh> đủ n lần, với n được đặt trước trong thanh ghi CX. Sau mỗi lần lặp CX tự động giảm 1 đơn vị ($Cx = Cx - 1$) và lệnh lặp sẽ dừng khi $Cx = 0$.

| Instruction | Description | Flags test |
|-------------|-------------------------------------|------------|
| JE/JZ | Jump Equal or Jump Zero | ZF |
| JNE/JNZ | Jump not Equal or Jump Not Zero | ZF |
| JG/JNLE | Jump Greater or Jump Not Less/Equal | OF, SF, ZF |
| JGE/JNL | Jump Greater/Equal or Jump Not Less | OF, SF |
| JL/JNGE | Jump Less or Jump Not Greater/Equal | OF, SF |
| JLE/JNG | Jump Less/Equal or Jump Not Greater | OF, SF, ZF |

XỬ LÝ CHUỖI

- Lệnh : LODS
- Dạng lệnh : LODSB
LODSW
- Chức năng : Nạp chuỗi nguồn byte vào thanh ghi AL hay chuỗi nguồn word vào thanh ghi AX. Cập thanh ghi DS:SI giữ địa chỉ chuỗi nguồn. Địa chỉ chuỗi nguồn được tự động tăng hay giảm sau mỗi lần nạp. Chiều tăng giảm địa chỉ tùy thuộc cờ định hướng DF. DF=0 xử lý tăng địa chỉ. DF=1 xử lý giảm địa chỉ.

❖ LODSB AL \leftarrow [DS:SI]

Nếu DF=0 thì : SI \leftarrow SI + 1

ngược lại thì : SI \leftarrow SI - 1

❖ LODSW AX \leftarrow [DS:SI+1,DS:SI]

Nếu DF=0 thì : SI \leftarrow SI + 2

4. Lệnh LEA (LoadEffectiveAddress)

Cú pháp:

LEA [Toán hạng đích],[Toán hạng nguồn]

Trong đó: [Toán hạng đích]: Là các thanh ghi 16 bit. [Toán hạng nguồn]: Là địa chỉ của một vùng nhớ hay tên của một biến.

Tác dụng: Lệnh LEA có tác dụng chuyển địa chỉ offset của [Toán hạng nguồn] vào [Toán hạng đích]. Lệnh này thường được sử dụng để lấy địa chỉ offset của một biến đã được khai báo trong chương trình. Thanh ghi được sử dụng trong trường hợp này là thanh ghi cơ sở (BX) và thanh ghi chỉ mục (SI và DI).

5. Lệnh Mul và Div

- **Mul** [Toán hạng nguồn]
- **IMul** [Toán hạng nguồn]
- **Div** [Toán hạng nguồn]
- **IDiv** [Toán hạng nguồn]

Tác dụng:

- Lệnh **Mul** (**M**ultiply): Thực hiện phép nhân trên số không dấu.
 $AX = AL * \text{nguồn } 8\text{bit}.$

*Nếu nguồn là 16bit thì kết quả lưu vào DX $AX = AX * \text{nguồn } 16\text{bit}$. Phần thấp ở AX, phần cao ở DX.*

- Lệnh **IMul** (**I**nteger **M**ultiply): tương tự như MUL nhưng là số có dấu.

DIV :

Chức năng: Thực hiện phép chia trên số không dấu. Nếu [Toán hạng nguồn] là toán hạng 8 bit thì lệnh sẽ lấy giá trị của thanh ghi Ax (số bị chia) chia cho [Toán hạng nguồn] (số chia),

kết quả thương số chứa trong thanh ghi Al, số dư chứa trong thanh ghi Ah.

Nếu [Toán hạng nguồn] là toán hạng 16 bit thì lệnh sẽ lấy giá trị của cặp thanh ghi Dx:Ax (số bị chia) chia cho [Toán hạng nguồn] (số chia),
kết quả thương số chứa trong thanh ghi Ax, số dư chứa trong thanh ghi Dx.

Nếu phép chia cho 0 xảy ra hay thương số vượt quá giới hạn của thanh

ghi AL (chia 8 bit) hay Ax (chia 16 bit) thì CPU sẽ phát sinh lỗi “Divice overflow”.

6. Lệnh logic: NOT – AND – OR – XOR – TEST

Cú pháp:

- **Not** [Toán hạng đích]
- **And** [Toán hạng đích], [Toán hạng nguồn]
- **Or** [Toán hạng đích], [Toán hạng nguồn]
- **Xor** [Toán hạng đích], [Toán hạng nguồn]
- **Test** [Toán hạng đích], [Toán hạng nguồn]

Lệnh TEST : tương tự lệnh AND nhưng không ghi kết quả nó chỉ ảnh hưởng đến các cờ CF,OF,ZF

8. Lệnh dịch bit

SHL: dịch trái bit ảnh hưởng đến cờ

SHR: dịch phải bit ảnh hưởng đến cờ

SAL: Dịch trái (quay)

SAR: Dịch phải(quay)

RCR: quay phải.

RCL: quay trái.

9. Lệnh CMP

Cú pháp: **Cmp** [Toán hạng đích], [Toán hạng nguồn]

- **Tác dụng:** Lệnh Cmp (**Compare**) được sử dụng để so sánh giá trị/nội dung của [Toán hạng đích] so với [Toán hạng nguồn]. Tương tự như lệnh Sub, nó lấy [Toán hạng đích] trừ đi [Toán hạng nguồn] nhưng kết quả không làm thay đổi [Toán hạng đích] mà chỉ làm thay đổi giá trị của một số cờ hiệu: CF, ZF, OF,...

Assembly, bài tập assembly, assembly,emu8086, lập trình hợp ngữ , kiến trúc máy tính, ktxvl, vi xử lý, kiến trúc vi xử lý

1; (assembly) Hiển thị lời chào Tiếng Anh và Tiếng Việt trên emu8086

;in 2 chuỗi

.Model Small ; chương trình này chọn bộ nhớ small

.Stack 100 ; kích thước ngăn xếp là 100 bytes

.Data ;các dòng dưới data là khai báo

CRLF DB 13, 10, '\$' ;ki tu xuống dòng


```
ChaoTay DB 'hello!$'  
ChaoTa DB 'chao ban!$'
```

.Code

MAIN Proc;thu tuc chinh

```
;khởi tạo DS **
```

```
MOV AX, @Data ; khởi đầu thanh ghi DS
```

```
MOV DS, AX;trở thành ghi ds về đầu đoạn data
```

```
; hiển thị lời chào dùng hàm 9 của INT 21H
```

```
MOV AH, 9; lệnh gọi hàm 09h của ngắt 21 in một chuỗi kí tự
```

```
LEA DX, ChaoTay ; đóng in chuỗi chào tay
```

```
INT 21H ;hello!
```

```
LEA DX, CRLF;in dấu enter và lùi vào đầu dòng
```

```
INT 21H
```

```
; hiển thị lời chào ta dùng hàm 9 của INT 21H
```

```
LEA DX, ChaoTa
```

```
INT 21H ;chào bạn!
```

```
; trở về DOS dùng hàm 4 CH của INT 21H
```

```
MOV AH, 4CH
```

```
INT 21H
```

MAIN Endp

END

2: (assembly) Nhập 1 ký tự và xuất ra màn hình trên emu8086

```
.model small
```

```
.stack
```

```
.data
```

TBao1 db "Hay nhap mot ky tu: \$" ;khai tao xau

TBao2 db 13,10,"Ky tu da nhap: \$"

output db ? ;khai tao bien KyTu khong co gia tri ban dau*

.code

main proc

Mov ax,@data ;khai dau thanh ghi DS

Mov ds,ax ;tro thanh ghi ds ve dau doan data

;in ra man hinh xau TBao1

Lea dx, TBao1 ;dua con tro toi dia chi cua TBao1

Mov ah, 9 ;su dung ham ngat 9 cua ngat INT 21h

int 21h

;nhap vao 1 ki tu tu ban phim *

Mov ah, 1 ;su dung ham ngat 1 cua ngat INT 21h

Int 21h

Mov output, al;gan gia tri vua nhap (duoc luu o AL) vao bien KyTu

;in ra man hinh xau TBao2

lea dx, TBao2

mov ah, 9

int 21h

;hien thi ky tu da nhap

Mov ah, 2 ;su dung ham ngat 2 cua ngat INT 21h

Mov dl, output ;Hien thi ra man hinh ky tu da luu o DL
Int 21h

;ve dos

Mov ah, 4Ch

Int 21h

main endp

End

3. (assembly) nhập kí tự thường và chuyển sang kí tự hoa trên emu8086

;nhap ky tu truooc sau do in chuoi và ky tu hoa sau

.MODEL SMALL

.STACK 100h

.DATA

tb2 DB 13,10,'Chuyen sang ki tu hoa la : \$'

Char DB ?, '\$'

.CODE

Main PROC

MOV AX,@DATA

```

MOV DS,AX
;Nhap vao 1 ky tu thuong

MOV AH,1      ;Su dung ngat 1 cua ngat INT 21h de doc 1 ki tu va luu
vao AL
INT 21h
SUB AL,20h    ;Doi thanh ki tu hoa

MOV Char,AL   ;gan gia tri vua nhap (luu o AL) vao bien Char


;In ra thong bao tb2
LEA DX,tb2
MOV AH,9
INT 21h

LEA DX,char
MOV AH,9
INT 21h

;Ket thuc chuong trinh

MOV AH,4Ch
INT 21h
Main ENDP

END

```

4. (assembly) Nhập một chuỗi và in ra trên emu8086

```

;nhap chuoi truoc , in thong bao va in chuoi sau
.model small
.stack 100
.data
    tb1 DB 10,13, 'chuoi da nhap la: $';10 xuong dong , 13 lui dau dong*
    str DB 100 dup('$') ;

```

```

.code
main proc
    mov ax, @data
    mov ds,ax ;khởi tạo thanh ghi ds

    ;nhập chuỗi ký tự
    mov ah,10 ; 10=0ah
    lea dx,str ;trở đến địa chỉ đầu str
    int 21h

    mov ah,9 ;in một chuỗi ký tự
    lea dx,tb1 ;
    int 21h ;in ra

    ;dưa dx chỉ về phần tử thứ 2 của mảng là
    ; ký tự đầu tiên được nhập vào
    lea dx,str +2 ;256,5,h,e,l,l,o
    int 21h ;in ra

    mov ah,4CH
    int 21h
main endp
end

```

5. (assembly) Nhập chuỗi thành chuỗi thường và chuỗi hoa trên emu8086

```

.Model Small
.Stack 100h
.Data
str DB 256 dup('$') ;khởi tạo chuỗi str
tb1 DB 10, 13, 'Chuyển sang chuỗi in thường: $'
tb2 DB 10, 13, 'Chuyển sang chuỗi in hoa: $'
.Code
main proc
    MOV AX, @Data
    MOV DS, AX ;khởi tạo thanh ghi ds

```

```

;Nhap chuoi:
LEA DX, str ;tro den dia chi chuoi str
MOV AH, 10 ; nhap xau ngat 10
INT 21H

;In thong bao 1
MOV AH, 9
LEA DX, tb1 ;hien xau tb1
INT 21H
CALL inthuong ;hien xau str chu thuong

;In chuoi in hoa:
MOV AH, 9
LEA DX, tb2 ;hien thong bao tb2
INT 21H
CALL inhoa

MOV AH, 4ch
INT 21H

```

```
main endp
```

```

inthuong proc
    LEA SI, str+2 ;
    Lap1: ;1 xau : kiem tra tung ky tu mot
        MOV DL, [SI]
        CMP DL, 'A' ;compare
        JL In1 ;jump less
        CMP DL, 'Z' ;compare
        JG In1 ;jump greater
        ADD DL, 32 ;chuyen ky tu hoa thanh thuong
    In1:
        MOV AH, 2 ;in ky tu
        INT 21H
        INC SI ;increase
        CMP [SI], '$' ;compare
        JNE Lap1 ;jump not equal

```

```

    RET ;return
inthuong endp

inhoa proc
    LEA SI, str+2
    Lap2:
        MOV DL, [SI]
        CMP DL, 'a'
        JL In2
        CMP DL, 'z'
        JG In2
        SUB DL, 32 ;chuyen ky tu hoa thanh ky tu thuong
    In2:
        MOV AH, 2
        INT 21H
        INC SI
        CMP [SI], '$'
        JNE Lap2
    RET
inhoa endp
end

```

6. (assembly) IN CHUỖI ĐẢO NGƯỢC trên emu8086

```

.model small
.stack 100
.data
    str DB 50 dup('$') ;str gom 50 bytes chua 50 gia tri khoi dau
    tb1 db 10,13,'Chuoi da duoc dao nguoc: $'
.code
    main proc
        mov ax,@data
        mov ds,ax ;khoi tao thanh ghi ds

        ;nhap xau ki tu
        lea dx,str
        mov ah,10;0Ah
    
```

```
int 21h
```

```
;in chuỗi tb2 ra màn hình
```

```
lea dx, tb1
```

```
mov ah,9
```

```
int 21h
```

```
;vd 123456789 =>len=9 256,9,1,2,3....
```

```
mov cl,[str + 1] ;chuyển chiều dài chuỗi vừa nhập vào cl
```

```
lea si,[str + 2] ;đưa si chỉ về phần tử thứ 2 của mảng là ký tự đầu tiên  
được nhập vào
```

```
Lap::đẩy các ký tự vào ngăn xếp
```

```
push [si];đưa phần tử mà si đang chỉ đến vào đầu ngăn xếp
```

```
inc si ;increase :tăng giá trị của si lên 1
```

```
loop Lap
```

```
mov cl, [str + 1];chuyển chiều dài chuỗi vừa nhập vào cl
```

```
Lap2: ;lấy dữ liệu từ ngăn xếp
```

```
pop dx ;lấy giá trị trên đỉnh ngăn xếp đưa vào dx
```

```
mov ah,2 ;in ký tự vừa lấy ra màn hình
```

```
int 21h
```

```
Loop Lap2
```

```
;trở về DOS dùng hàm 4 CH của INT 21H
```

```
MOV AH, 4CH
```

```
INT 21H
```

```
main endp
```

```
end
```

7. (assembly) in chuỗi đảo ngược khi điền dấu # trên emu8086

```
;enter là 13 , esc là 27
```

```
.Model small
```

```
.Stack 100h
```

```
.Data
```

```
str db 50 dup ('$') ;khởi tạo chuỗi 50 bytes
```

```
.code
```



```

main proc
    mov ax,@Data
    mov ds,ax ;khởi tạo thanh ghi ds
    mov cx,0;gán giá trị cho thanh ghi cx=0
START:
    inc cx;increase tăng cx lên 1
    mov ah,1 ;nhập 1 kí tự
    int 21h
    cmp al,'#';so sánh al với kí tự # => enter là 13
    je end ;jump equal :nhảy đến 'end' nếu bằng
    push ax ; thêm phần tử vào ngăn xếp
    jmp START;nhảy k điều kiện

end::;kết thúc
    mov ah,2 ;in kí tự
    mov dl,0; in dấu cách
    int 21h

    dec cx;decrease:trừ cx đi 1
    pop dx;lay phần tử ở đỉnh ngăn xếp đưa vào dx
    mov ah,2 ;in kí tự
    int 21h

    cmp cx,1;so sánh cx với 1
    jne end;jump not equal :nhảy đến 'end' nếu không bằng
    ;ve dos
    Mov ah, 4Ch
    Int 21h
main endp
end

```

8. (assembly) Tìm giá trị lớn nhất của mảng

.Model small

.Stack 100H

.Data

list DB 1,2,3,4,5,6,7,8,0

.code

```

main proc
    ; initilize the ds and es registers
    mov ax, @Data ;
    mov ds,ax ;
    mov cx, 9
    lea si, list    ; dua gia tri dauu tien cua chuoi vào si
    mov bl, [si]    ; dua dia chi si vào bl
    inc si          ; tang gia tri si them 1
Start:
    lodsb
    cmp bl, al ; so sanh al va bl
    jge BYPASS;  nhay denn BYPASS
    mov bl, al;    neu al > bl thi gan bl = al;
BYPASS:
    loop Start ; lap

    ; print the max
    add bl, '0' ; ep kieu so ve kieu ke tu
    mov dl,bl ; dua gia tri max bl vào dl;
    mov ah, 2 ; in ra màn hình
    int 21H

    mov ah, 4CH ; ket thuc chuong trinh
    int 21H
main endp
End Main

```

16. (assembly) Đếm chiều dài 1 chuỗi trên emu8086

```

.MODEL small
.STACK 100
.DATA
    tb1 DB "Nhap vao 1 chuoi: $"
    tb2 DB 10,13,"Tong chieu dai cua chuoi: $"
    s DB 100 dup("$") ;chuoi nhap vao
    ;byte 0: chua so ki tu toi da cua xau nhap vao
    ;byte 1: chua chieu dai cua xau vd :123456789 => 9
    ;byte 2: Chua dia chi cua ky tu dau tien

```

.CODE

MAIN:

MOV AX, @DATA
MOV DS,AX ;khởi tạo thanh ghi ds

LEA DX,tb1 ;xuất chuỗi tb1
MOV AH,9
INT 21h

LEA DX,s
MOV AH,10 ;nhập chuỗi s
INT 21h

LEA DX,tb2
MOV AH,9 ;xuất chuỗi tb2
INT 21h

;Tính chiều dài chuỗi
mov ax,0
MOV AL,s+1 ;Chuyển chiều dài chuỗi vào ax al=9
MOV CX,0 ;Khởi tạo biến đếm cx=0(count)
MOV BX,10 ;khởi tạo số chia
;123 push 3 2 1 =>pop 1 2 3

LapDem1:: day các số vào ngăn xếp

MOV DX,0 ;khởi tạo phần dư bằng 0
DIV BX ; chia 10
PUSH DX ; lấy dx là phần dư day vào ngăn xếp
INC CX ;tăng cx lên 1 đơn vị
CMP AX,0 ;so sánh ax phần thương khác 0 thì tiếp tục vòng lặp
JNZ LapDem1 ;jump not zero
;xuất chiều dài chuỗi

LapDem2: ; lấy các số từ ngăn xếp ra

POP DX ;Chưa số dư trong phép chia

HUY INIT

```

ADD DX,'0' ;chuyen chu so -> ky tu so
MOV AH,2
INT 21H
LOOP LapDem2

MOV AH,4ch
INT 21h
END MAIN
End

```

9. (assembly) Nhập vào một số và tính giai thừa của số đó trên emu8086

```

;Nhap vao so va tinh giai thua
.model small
.stack 100
.data
    muoi dw 10
    TB1 db 10,13,'KET QUA LA: $'
.code
    main proc
        mov ax,@data
        mov ds,ax ;khoi tao thanh ghi ds

        mov ah,1 ;nhap 1 ky tu tu ban phim
        int 21h
        sub al,'0' ;vd :chuyen ky tu'5' ve so 5
        mov cx,0 ;cx=0
        mov cl,al ;cl=5

        lea dx,TB1 ; in chuoai tb1
        mov ah,9
        int 21h
        ;1*2*3*4*5
        mov ax,1;khoi tao ket qua ban dau =1
        mov bx,1 ;bien tang

        Giaithua: ;tinh giai thua tra ve 1 so

```

```

Mul bx ;ax * bx luu vao trong ax
inc bx ;increase:tang gia tri bx len 1
cmp bx,cx ;so sanh bx voi cx
jle giaithua ;neu bx<=cx thi tiep tục lap
mov cx,0

```

```

Lappush: ;lay tu ky tu cua so 120 vao day vao trong ngan xep
mov dx,0
div muoi ;chia cho 10
add dx,'0'
push dx;day vao ngan xep
inc cx ;increase : tang cx 1 don vi
cmp ax,0
jne Lappush

```

```

Hienthi: ;hien thi tuwng ky tu o trong ngan xep
pop dx ;lay 1 so o dau ngan xep*
mov ah,2
int 21h
Loop HienThi
mov ah,4Ch
int 21h
main endp
end main

```

10. (assembly) Chuyển một số từ hệ 10 sang hệ 2 trên emu8086

```

;10 => # 0 1 0 1 *
.Model Small
.Stack 100
.Data
    tb1 DB 10, 13, 'So da nhap dang nhi phan: $'
    str DB 5 dup ('$'); nhap vao 1 chuoì tôi đã 5 ký tự
.Code
main proc
    MOV AX, @Data
    MOV DS, AX ;khởi tạo thanh ghi ds

```

```

MOV AX, '#' ;so sanh
PUSH AX ;push dau # (acii) vao trong ngan xep
;Nhap so dang chuoi:
MOV AH, 10
LEA DX, str ; nhap chuoi str
INT 21h
;Chuyen chuoi thanh so:
MOV CL, [str+1] ; lay so ky tu cua chuoi ( vd :cl=2 )
LEA SI, str+2 ; tro den dia chi cua ky tu dau tien cua chuoi str
MOV AX, 0 ; ax=0 123
MOV BX, 10 ;bx=10 ;he so nhan
thapphan::; chuyen chuoi thanh so 123;0*10+1 1*10+2; 12*10+3
    MUL BX ;nhan 10
    MOV DL, [SI] ; dl='1'
    SUB DL, '0'; dl=1
    ADD AX, DX ;ax=ax+dx
    INC SI ;increase tang si 1 down vi
    LOOP thapphan
;Chuyen thanh so nhi phan: 10- 1010
MOV CL, 2 ; he so chia

nhiphan: ;chuyen so thap phan sang nhi phan va day cac so vao ngan xep
    MOV AH, 0 ;phan du =0
    DIV CL ; chia ax cho 2
    PUSH AX ; day ax vao ngan xep (al+ah)
    CMP AL, 0 ;so sanh thuong#0 tiep tục chia
    JNE nhiphan ;jump not equal

MOV AH, 9
LEA DX, tb1;in ra thb1
INT 21h

MOV AH, 2
Inra:
    POP DX ;lay tung phan tu trong ngan xep
    CMP DX, '#'

```

```

JE Done ;jump equal
MOV DL, DH ;lay duoc so tu ngan xep :1 0 1 0
ADD DL, '0' ; convert tu 1 so sang ky tu '1' '0' '1' '0'
INT 21h
JMP Inra

```

Done:

```

MOV AH, 4Ch
INT 21h

```

```

main endp
end

```

11. (assembly) Chuyển một số từ hệ 10 sang hệ 16 trên emu8086

```

;11: he 10 sang 16 444=>1BC
;444=1*16^2 + 11*16 + 12*16    *
.model small
.stack 100h
.data
.code
main proc
    mov ax,@data
    mov ds,ax;khoi tao thanh ghi ds
    mov ax,444 ;chuyen 444 he 10v vao thanh ghi ax=> sang he 16
    call printnum16 ;chuyen den ham printnum16
    mov ah,4ch
    int 21h
main endp

printnum16 proc
    mov bx,16 ;khoi tao bl bang 16
    mov cx,0 ;khoi tao bien dem

    hexa: ;xay dung duoc stack
        div bl ;lay so hien tai chia cho 16
        push ax ;day gia tri ax vao ngan xep
        inc cx ;tang bien dem them 1

```

```
    cmp al,0    ;neu thuong (thuong al cua lenh div) bang 0 thi ket thuc (ket
thuc startsplit)
    je ketqua   ;jum equal
    mov ah,0    ;clear ah (xoa phan du cua lenh div) va quay lai tiep tục chia
(quay lai startsplit)
    jmp hexa
```

ketqua:

inkt:

```
    pop ax     ;lay tung gia tri o dinh ngan xep vua day vao o ham startsplit
    mov dl,ah   ;lay phan du chuyen vao dl
```

```
    cmp dl,10   ;neu phan du la 10 thi chuyen den ham in ky tu A
    je hex_a
```

```
    cmp dl,11   ;neu phan du la 11 thi chuyen den ham in ky tu B
    je hex_b
```

```
    cmp dl,12   ;neu phan du la 12 thi chuyen den ham in ky tu C
    je hex_c
```

```
    cmp dl,13   ;neu phan du la 13 thi chuyen den ham in ky tu D
    je hex_d
```

```
    cmp dl,14   ;neu phan du la 14 thi chuyen den ham in ky tu E
    je hex_e
```

```
    cmp dl,15   ;neu phan du la 15 thi chuyen den ham in ky tu F
    je hex_f
```

```
    add dl,'0'  ;(neu khong la ky tu A->F) chuyen cac so tu 0-9 sang ascii
    jmp print    ;nhay den ham in de in so tu 0-9
```

hex_a:

```
    mov dl,'A'
```

```
    jmp print
```

hex_b:

```
    mov dl,'B'
```

```
    jmp print
```

hex_c:

```
    mov dl,'C'
```



```

        jmp print
hex_d:
        mov dl,'D'
        jmp print
hex_e:
        mov dl,'E'
        jmp print
hex_f:
        mov dl,'F'
        jmp print
print: ;in 1 ky tu ra man hinh
        mov ah,2
        int 21h
loop inkt
ret ;return
printnum16 endp
end

```

12.TỔNG CÁC CHỮ SỐ

| | |
|-----------------|------------------------------------|
| .model small | |
| | .stack 50 |
| | .data |
| | so db 10,0,10 dup(\$) |
| | tb1 db 'Nhap so: \$' |
| | tb2 db 10,13,'Tong cac chu so: \$' |
| | Tong db 0 |
| | muoi db 10 |
| | .code |
| | main proc |
| | mov ax,@data |
| | mov ds,ax |
| | |
| | lea dx,tb1 |

| | |
|--|---------------|
| | mov ah,9 |
| | int 21h |
| | ;nhap so vao |
| | lea dx,so |
| | mov ah,0Ah |
| | int 21h |
| | |
| | xor cx,cx |
| | lea si,so+2 |
| | mov cl,[so+1] |
| | |
| | xor ax,ax |
| | Lap: |
| | xor bx,bx |
| | mov bl,[si] |
| | sub bl,30h |
| | add ax,bx |
| | inc si |
| | loop Lap |
| | |
| | |
| | xor cx,cx |
| | lapchia: |
| | xor dx,dx |
| | div muoi |
| | add ah,30h |
| | mov dl,ah |
| | push dx |
| | inc cx |
| | xor ah,ah |
| | cmp ax,0 |

| | |
|--|--------------|
| | jne lapchia |
| | |
| | lea dx,tb2 |
| | mov ah,9 |
| | int 21h |
| | Hienthi: |
| | pop dx |
| | mov ah,2 |
| | int 21h |
| | loop Hienthi |
| | mov ah,4Ch |
| | int 21h |
| | main endp |
| | end main |
| | |

13.UCLN, BCNN:

.model small

.stack 100h

.data

str1 db 'UCLN: \$'

str2 db 10,13,'BCNN: \$'

.code

main proc

mov ax,@data

mov ds,ax

mov cl,7

mov ch,9

mov dl,cl

mov dh,ch

```
uc:
cmp dl,dh
je ucend
jg ucln
sub dh,dl
jmp uc
ucln:
sub dl,dh
jmp uc
```

```
ucend:
push dx
mov ah,9
lea dx,str1
int 21h
```

```
pop dx
xor ah,ah
mov al,dl
push dx
call printnum
```

```
mov ah,9
lea dx,str2
int 21h
```

```
mov al,cl
mul ch
```

```
pop dx
div dl
call printnum
```

```
    mov ah,4ch
    int 21h
main endp

printnum proc
    push ax
    push bx
    push cx
    push dx

    mov bl,10
    mov cx,0

loopcat:
    div bl
    push ax
    inc cx
    cmp al,0
    je catexit
    xor ah,ah
    jmp loopcat

catexit:
startprint:
    pop ax
    mov dl,ah
    add dl,'0'
    mov ah,2
    int 21h
    loop startprint
    pop dx
    pop cx
    pop bx
    pop ax
    ret
```

```
printnum endp
```

```
end main
```

14.tổng chia hết cho 7:

```
.model small
```

```
.stack 100h
```

```
.data
```

```
mang1 db 7,5,60,233,77,43
```

```
str1 db 'Tong: $'
```

```
.code
```

```
main proc
```

```
    mov ax,@data
```

```
    mov ds,ax
```

```
    lea si,mang1
```

```
    mov bx,0
```

```
    mov dl,7
```

```
tong:
```

```
    mov al,[si]
```

```
    xor ah,ah
```

```
    mov cx,ax
```

```
    inc si
```

```
    cmp si,6
```

```
    je endtong
```

```
    div dl
```

```
    cmp ah,0
```

```
    jne tong
```

```
    add bx,cx
```

```
    jmp tong
```

```

    endtong:

    mov ax,bx
    call printNumber

    mov ah,4ch
    int 21h

main endp

;-----
; print an integer number (MAX: 2550)
; input: AX: input number to print
printNumber proc
    push ax
    push bx
    push cx
    push dx

    ; check if AX is negative --> change to positive and print the minus sign (-)
    cmp ax, 0
    jge NOT_NEGATIVE
    ; change to positive
    neg ax
    ; print the minus sign
    mov dl, '-'
    call printCharacter

    ; divide the number by 10 to get the remainder and the quotient
NOT_NEGATIVE:
    mov bl, 10
    mov cx, 0
    StartSplit:

```

```

div bl
push ax ; store the result into stack
inc cx ; count the number of digit
cmp al, 0
jz ExitSplit
xor ah, ah
jmp StartSplit
ExitSplit:

; print each digit
StartPrint:
pop ax
mov dl, ah
call printSingleDigit
loop StartPrint

pop dx
pop cx
pop bx
pop ax
ret
printNumber endp
;-----
; proc to print out a single digit number
; input: dl to contain the digit to print
printSingleDigit proc
push ax
add dl, '0' ; digit to char
mov ah, 2
int 21H
pop ax
ret
printSingleDigit endp
; -----
; proc to print a string
; input: DX to contain the relative address of the string

```



```

printString proc
    push ax    ; store AX into stack
    mov ah, 9
    int 21H
    pop ax    ; restore AX from stack
    ret
printString endp

;-----
; proc to print out a character
; input: dl to contain the character to print
printCharacter proc
    push ax
    mov ah, 2
    int 21H
    pop ax
    ret
printCharacter endp

end main

```

15. Tổng 2 số kiểu word

```

.model tiny
.stack 100h
.data
tb1 db 'nhap so thu 1:$'
tb2 db 13,10,'nhap so thu 2:$'
tb3 db 13,10,'tong 2 so$'
so1 dw 0
so2 dw 0
tong dw 0
.code
main proc
mov ax,@data
mov ds,ax
;in thong bao nhap so thu nhat

```

```

lea dx,tb1
mov ah,9
int 21h
nhap1:
mov ah,1
int 21h
cmp al,13 ;so sanh ky tu vua nhap voi 13
je nhap2 ;neu bang nhap so thu 2
sub al,30h ;doi ky tu sang so
mov ah,0 ;xoa bit cao
mov cx,ax ;cat so vua nhap vào cx
mov ax,so1 ;dua bien so 1 về kiểu byte de chuan bi nhann với 10
mov bx,10 ;gan bx =10
mul bx ; nhân ax voi 10
add ax,cx ;công ket qua vua nhan voi so vua nhap ket qua cât vào ax
mov so1,ax ; cat kêt qua vào biến so1
jmp nhap1
nhap2:
lea dx,tb2 ;hiên thông báo nhâp so thu 2
mov ah,9
int 21h
nhap: mov ah,1 ;nhap so thu 2
int 21h
cmp al,13 ;so sánh ký tu vua nhâp voi 13
je tinhcong ;neu bang thì tính tổng
sub al,30h ;chuyển ký tu sang dạng số
mov ah,0 ;xóa bit cao
mov cx,ax ;cắt kêt qua vua nhap vào cx
mov ax,so2 ;dua biên số 2 về kiểu byte
mov bx,10 ;gan bx=10
mul bx ;nhân kêt qua vua nhap voi 10
add ax,cx ;công kêt qua vua nhân với số vua nhâp
mov so2,ax ;cắt kêt qua vào biên số 2
jmp nhap
tinhcong:
mov dx,tong

```

```

mov ax,so1 ;dua bien so 1 ra thanh ghi ax
mov bx,so2 ;dua bien so 2 ra thanh ghi bx
add ax,bx ;cong ax voi bx ket qua cat vao ax
mov tong,ax ;dua ket qua tu ax vao bien tong
inso: mov ah,9 ;hien thông báo in tong
lea dx,tb3
int 21h
mov ax,tong ;dua ket qua trongv bien tong ra thanh ghi ax
mov dx,0 ;xoa bit cao dx
mov bx,10 ;gán bx=10
mov cx,0 ;khoi tao bien dem
chia: div bx ;lay ket qua chia cho 10
push dx ;du o dx dây vao ngan xep
inc cx ;tang bien dem
cmp ax,0 ;so sanh thuong voi 0
je hienkq ;neu bang thi hien ket qua
xor dx,dx ;xoa bit cao trong dx
jmp chia
hienkq: pop dx ;lay du trong ngan xep ra khoi dx
add dl,30h ;chuyen so thanh dang ky tu
mov ah,2 ;in tong
int 21h
loop hienkq
ra: mov ah,4ch
int 21h
Main endp
End main

```