Danh sách tập lệnh thường xuyên sử dụng của 8086

I, CÁC LỆNH TRAO ĐỔI DỮ LIỆU

1, MOV: lệnh gán giá trị

- cú pháp: MOV đích, gốc
- chức năng: đích = gốc
- VD:

MOV AL, BL;	AL = BL
MOV CX, 123FH;	CX = 123FH
MOV DL, [DI];	DL = [DS:DI]

2, LEA: lệnh gán địa chỉ hiệu dụng

- cú pháp: LEA đích, gốc
- chức năng: nạp địa chỉ của gốc vào đích
- trong đó:
  - + đích thường là một trong số các thanh ghi: BX, DX, BP, SI, DI
  - + gốc là tên biến trong đoạn DS được chỉ rõ trong lệnh hoặc ô nhớ cụ thể
- VD:

LEA SI, a;	nạp địa chỉ biến a vào thanh ghi SI
LEA CX, [BX];	nạp địa chỉ ô nhớ có địa chỉ [DS:BX] vào CX (hay CX = BX)

3, PUSH: đẩy giá trị vào ngăn xếp

- cú pháp: PUSH gốc
- chức năng: đẩy giá trị của gốc vào ngăn xếp
- VD:

PUSH AX;	đẩy giá trị của AX vào ngăn xếp
PUSH 0ah;	đẩy giá trị 0ah vào ngăn xếp

4, POP: lấy giá trị từ ngăn xếp

- cú pháp: POP gốc
- chức năng: lấy giá trị trên cùng của ngăn xếp và gán vào gốc
- VD:

POP AX;	lấy giá trị trên cùng của ngăn xếp và gán vào AX

II, CÁC PHÉP TOÁN

1, Add: cộng 2 toán hạng

- cú pháp: add đích, gốc
- chức năng: Đích = Đích + gốc
- VD:

Add AL, 74H;	AL = AL + 74H
Add CL, AL;	CL = CL + AL
Add DL, [SI]	DL = DL + [DS:SI]

2, SUB: trừ hai toán hạng

- cú pháp: sub đích,gốc
- chức năng: đích = đích - gốc
- VD:

```
sub AL, 74H;        AL = AL - 74H
sub CL, AL;         CL = CL - AL
sub DL, [SI]        DL = DL - [DS:SI]
```

3, MUL: nhân hai toán hạng

- cú pháp: mul gốc
- chức năng:
  - + với gốc là số 8 bit: AX = AL x Gốc
  - + với gốc là số 16 bit: DXAX = AX x gốc
- VD:

```
mul CL;             AX = AL x CL
mul BX;             DXAX = AX x BX
```

- Lấy số 8 bit nhân với số 16 bit: Giả sử muốn lấy ô nhớ có địa chỉ DS:SI 8bit nhân với thanh ghi BX 16 bit, ta có thể làm bằng cách để số 16 bit tại gốc, số 8 bit vào AL, sau đó mở rộng sang AH để thành 16 bit

```
MOV [SI], 74;       [DS:SI] = 74
MOV BX, 123FH;      BX = 123FH
MOV AL, [SI];       AL = [DS:SI]
MOV AH, 00H;        AH = 00H -> AX = AL
MUL BX;             DXAX = AX x BX
```

4, DIV: chia hai toán hạng

- cú pháp: div gốc
- chức năng:
  - + với gốc là số 8 bit: AL = AX / Gốc
    AH = AX % Gốc
  - + với gốc là số 16 bit: AX = DXAX / gốc
    DX = DXAX % gốc
- Thương được làm tròn theo số nguyên dưới (VD: AX / gốc = 4,9 -> AL = 4)
- nếu gốc = 0 hoặc thương lớn hơn FFH (với phép chia 8 bit) hoặc FFFFH (với phép chia 16 bit) thì CPU thực hiện lệnh ngắt INT 0
- VD:

```
div BL;             AL = AX / BL;  AH = AX % BL
div [SI];           AL = AX / [DS:SI];      AH = AX % [DS:SI]
div BX;             AX = DXAX / BX;         DX = DXAX % BX
```

5, DEC: trừ đi 1

- cú pháp: DEC đích
- chức năng: đích = đích - 1
- VD:

```
DEC AL;             AL = AL - 1
DEC BX;             BX = BX - 1
```

6, INC: tăng thêm 1

- cú pháp: INC đích
- chức năng: đích = đích + 1
- VD:
  - INC AL; AL = AL + 1

- INC BX; BX = BX + 1

7, NEG: đảo dấu
- cú pháp: NEG đích
- chức năng: đích = 0 - đích
- VD:

NEG AL; AL = -AL;

III, MỘT SỐ LỆNH SO SÁNH BIT:

1, AND:
- cú pháp: and đích, gốc
- chức năng: đích = đích & gốc
- thường dùng để lấy đi một số bit nhất định của một toán hạng
- VD:

AND AL, 0FH; lấy 4 bit cao của AL
(VD: AL = 1011 0111 -> AL = 0000 0111)

2, OR:
- cú pháp: or đích, gốc
- chức năng: đích = đích | gốc
- thường dùng để lập 1 số bit của toán hạng
- VD:

OR AL, F0H; biến 4 bit đầu của AL thành 1
(VD: AL = 1001 1011 -> AL = 1111 1011)

3, XOR:
- cú pháp : xor đích, gốc
- chức năng: đích = đích ^ gốc
- thường dùng để xoá một toán hạng về 0 bằng cách XOR với chính nó
- có thể dùng để đảo bit
- VD:

XOR BL, BL; BL = BL ^ BL = 0000 0000
XOR AL, BL; AL = AL ^ BL
(VD: AL = 1011 0110, BL = 1111 1111 -> AL = 0100 1001)

4, CMP:
- cú pháp: CMP đích, gốc
- chức năng: so sánh hai toán hạng đích và gốc
- sau khi so sánh hai toán không thay đổi, không lưu kết quả so sánh, lệnh chỉ tác động đến các cờ của thanh ghi FR
- thường dùng để tạo cờ cho các lệnh nhảy

IV, CÁC LỆNH DỊCH VÀ QUAY

1, ROL: quay trái (chiều ngược kim đồng hồ) (Rotate left)
- cú pháp: ROL đích, CL
- chức năng: quay toán hạng sang trái CL bit
- trong trường hợp chỉ quay 1 bit thì có thể viết như sau:

ROL đích, 1

- VD:

ROL BL, CL; quay trái BL CL bit
(VD: BL = 1000 000, CL = 2 -> BL = 0000 0010)
ROL AL, 1; quay trái AL 1 bit
(VD: AL = 1000 0000 -> AL = 0000 0001)

2, ROR: quay phải (chiều kim đồng hồ)

Right

- cú pháp: RỎ đích, CL
- chức năng: quay phải toán hạng sang phải CL bit
- trong trường hợp chỉ quay 1 bit thì có thể viết như sau:
  ROR đích, 1
- VD:
  ROR BL, CL; quay phải BL CL bit
  (VD: BL = 1000 1001, CL = 1 -> BL = 1100 0100)
  ROR AL, 1; quay phải AL 1 bit
  (VD: AL = 1000 0000 -> AL = 0100 0000)

3, SHL: dịch trái
- cú pháp: SHL đích, CL
- chức năng: dịch trái toán hạng CL bit
- trong trường hợp chỉ dịch 1 bit thì có thể viết như sau:
  SHL đích, 1
- VD:
  SHL AL, CL; dịch trái AL CL bit
  (VD: AL = 1111 1111, CL = 5 -> AL = 1110 0000)
  SHL BL, 1; dịch trái BL 1 bit
  (VD: BL = 1111 1111 -> BL = 1111 1110)

4, SHR: dịch phải
- cú pháp: SHr đích, CL
- chức năng: dịch phải toán hạng CL bit
- trong trường hợp chỉ dịch 1 bit thì có thể viết như sau:
  SHR đích, 1
- VD:
  SHR AL, CL; dịch phải AL CL bit
  (VD: AL = 1111 1111, CL = 5 -> AL = 0000 0111)
  SHR BL, 1; dịch phải BL 1 bit
  (VD: BL = 1111 1111 -> BL = 0111 1111)

V, CÁC LỆNH NHẢY
- cú pháp: <tên lệnh> Nhãn
- chức năng: IP = IP + Dịch chuyển (nhảy đến nhãn nếu phù hợp với điều kiện của lệnh)
- thường dùng với lệnh CMP

1, JMP (Jump): nhảy không điều kiện
2, JG (Jump if Greater): nhảy nếu lớn hơn
3, JNG (Jump if Not Greater): nhảy nếu không lớn hơn (nhảy nếu bé hơn hoặc bằng)
4, JLE (Jump if Lower or Equal): tương tự JNG
5, JL (Jump if Lower): nhảy nếu bé hơn
6, JNL (Jump if Not Lower): Nhảy nếu không bé hơn (nhảy nếu lớn hơn hoặc bằng)
7, JGE (Jump if Greater or Equal): tương tự JNL
8, JE (Jump if Equal): nhảy nếu bằng
9, JNE (Jump if Not Equal): nhảy nếu không bằng
10, JZ (Jump if Zero): nhảy nếu bằng 0 (lệnh này tương tự JE)
11, JNZ (Jump if Not Zero): nhảy nếu khác 0 (lệnh này tương tự JNE)
12, JS (Jump if Signed): nhảy nếu có dấu (nhảy nếu SF == 1)
13, JNS (Jump if Not Signed): nhảy nếu không có dấu (nhảy nếu kết quả dương) (nhảy nếu SF == 0)

12, JC (Jump if Carry): nhảy nếu có nhớ (nhảy nếu CF == 1)
13, JNC (Jump if Not Carry): nhảy nếu không có nhớ (nhảy nếu CF == 0)
12, JO (Jump if Overflow): nhảy nếu tràn (nhảy nếu OF == 1)
13, JNO (Jump if Not Overflow): nhảy nếu không tràn (nhảy nếu OF == 0)
VD:

```
CMP AL, BL;        so sánh AL và BL
JE bangnhau;       nhảy đến nhãn bangnhau nếu kết quả bằng nhau (AL == BL)

CMP [SI], CL;      so sánh [DS:SI] và CL
JG lonhon          nhảy đến nhãn lonhon nếu kết quả lớn hơn ([DS:SI] > CL)

SUB AL, AH;        AL = AL - AH
JZ bangkhong       nhảy đến nhãn bangkhong nếu kết quả bằng 0 (AL - AH == 0)

CMP AL, 00H;       so sánh AL với 00H
JS am;             nhảy đến nhãn am nếu có dấu (SF == 1) (AL < 0)

ADD AL, AH;        AL = AL + AH
JO tran            nhảy đến nhãn tran nếu có tràn (nhảy nếu giá trị AL + AH vượt
```

quá 8 bit) (OF == 1)

VI, LỆNH ĐIỀU KHIỂN CỜ:

1, CLD: xoá cờ hướng
- cú pháp: CLD
- chức năng: DF = 0

2, STD:lập cờ hướng
- cú pháp: STD
- chức năng: DF = 1

3, CLC: xoá cờ nhớ
- cú pháp: CLC
- chức năng: CF = 0

4, STC: lập cờ nhớ
- cú pháp: STC
- chức năng: CF = 1

5, CMC: đảo cờ nhớ
- cú pháp: CMC
- chức năng: CF = !CF (CF = 0 -> CF = 1; CF = 1 -> CF = 0)

VII, CÁC LỆNH Di CHUYỂN CHUỖI

1, LODSB/LODSW:
- cú pháp:
- chức năng:
- VD:

2, STOSB/STOSW:
- cú pháp:
- chức năng:
- VD:

3, MOVSB/MOVSW:
- cú pháp:
- chức năng:

- VD:

VIII, LỆNH NGẮT INT 21H

- cú pháp: INT 21H
- chức năng của lệnh dựa theo giá trị của AH

1, Ngắt loại 1: đọc một ký tự từ bàn phím

- thực hiện khi AH = 1
- chức năng: 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
- VD:

```
MOV AH, 1;    AH = 1
INT 21H;      c/trình lúc này sẽ ngừng lại đến khi bạn nhập vào một phím
```

2, Ngắt loại 2: hiện một ký tự lên màn hình

- thực hiện khi AH = 2
- chức năng: hiện một ký tự có mã ASCII là giá trị của DL lên màn hình
- VD:

```
MOV AH, 2;          AH = 2
MOV DL = 30h;       DL = 30h (30h là mã ASCII của '0')
INT 21H;            màn hình sẽ in ra ký tự '0'
```

3, Ngắt loại 9: hiện xâu ký tự có ký tự '$' ở cuối

- thực hiện khi AH = 9
- chức năng: hiện xâu ký tự có địa chỉ lệch là giá trị của DX
- VD:

```
tb BD 'co lam thi moi co an$';    khai báo xâu ký tự tb
MOV AH, 9;                        AH = 9
LEA DX, tb;                       gán địa chỉ lệch của tb vào DX
INT 21H;                          màn hình in ra xâu tb (ko hiện ký tự '$')
```

4, Ngắt chương trình (ngắt 4CH): dừng chương trình

- thực hiện khi AH = 4CH
- chức năng: dừng chương trình
- VD:

```
MOV AH, 4CH;      AH = 4CH
INT 21H;          dừng chương trình
```

Chương trình hợp ngữ của 8086

IX, KHAI BÁO BIẾN, HẰNG, MẢNG, CHUỖI KÝ TỰ

1. biến

- cú pháp: <tên biến> <kiểu dữ liệu> <giá trị khởi đầu>
- trong đó:
  - + tên biến: là tên của biến
  - + kiểu dữ liệu: là miền giá trị của biến, có 3 kiểu dữ liệu:
    DB (define byte): biến byte, độ dài 8 bit
    DW (define word): biến word, độ dài 16 bit
    DD (define double word): biến double word, độ dài 32 bit
  - + giá trị khởi đầu: là giá trị được gán vào khi biến được khởi tạo. Nếu muốn khởi tạo biến mà không có giá trị ban đầu, sử dụng ký tự '?'
- VD:

```
B1      DB      16;     khởi tạo biến B1 có giá trị là 16
x       DW      ff0ah;  khởi tạo biến x có giá trị là ff0ah
y       DB      ?;      khởi tạo biến y không có giá trị ban đầu
```

2, mảng

- cú pháp <tên mảng> <kiểu dữ liệu> <giá trị 1>,<giá trị 2>,...
- trong đó:
  - + tên mảng: là tên của mảng
  - + kiểu dữ liệu: là miền giá trị của các phần tử trong mảng, có 3 kiểu dữ liệu:
    DB (define byte): biến byte, độ dài 8 bit
    DW (define word): biến word, độ dài 16 bit
    DD (define double word): biến double word, độ dài 32 bit
  - + giá trị 1, giá trị 2,...: là giá trị được gán vào khi các biến được khởi tạo. Nếu muốn khởi tạo biến mà không có giá trị ban đầu, sử dụng ký tự '?'. Nếu muốn khởi tạo nhiều biến có cùng 1 giá trị, sử dụng lệnh:
    <số lượng phần tử> DUP(<giá trị>)
    lệnh DUP có thể lồng nhau.
- VD:

```
mang    DB      1,2,3,4,5,6,7;   khởi tạo mảng có 7 phần tử có gtrị từ 1 đến 7
dl      DB      100 DUP(?);      khởi tạo mảng có 100 phần tử chưa có giá trị
d1      DB      1, 2, 3 DUP(4);khởi tạo mảng có 5 phần tử có gtrị là: 1,2,4,4,4
d2      DB      1,2, 2 DUP(4 , 3 DUP(5), 6);  d2 = 1, 2, 4, 5, 5, 5, 6, 4, 5, 5, 5,6
```

3, chuỗi

- cú pháp: <tên xâu> <kiểu dữ liệu> <xâu>
- là một kiểu đặc biệt của mảng, trong đó, mỗi ký tự của xâu là 1 phần tử của mảng. giá trị của mỗi phần tử chính là mã ASCII của ký tự đó.
- ký tự '$' báo hiệu đã hết xâu.

- VD:

```
xau1   DB    'can lao vi tien thu$';   xau1 = 'can lao vi tien thu$'
xau2   DB    30h, 31h, 32h, 33h,'$';          xau2 = '0123$'
CRLF   DB    13,10,'$';       đây là xâu dùng để xuống dòng và về đầu dòng
(13 là ký tự về đầu dòng (CR - carriage return), 10 là ký tự thêm dòng mới
(LF - line feed), hiểu đơn giản CRLF có tác dụng như "\n" trong c\c++)
```

4, hằng

- cú pháp: <tên hằng> EQU <giá trị>
- chức năng: tạo hằng có tên là <tên hằng> và có giá trị là <giá trị>. giá trị của hằng không thể thay đổi.
- VD:

```
CRLF   EQU   13, 10, '$';      khai báo hằng CRLF
a1     EQU   19;               khai báo hằng a1 = 19
a2     EQU   'Hello '          khai báo hằng a2 = 'Hello '
```

- có thể sử dụng hằng để khai báo biến mảng:

```
MSG    DB    a2, 'World$';    khai báo MSG = 'Hello World$'
```

X, CẤU TRÚC MỘT CHƯƠNG TRÌNH HỢP NGỮ (.EXE)

- Khung cơ bản của 1 chương trình:

```
.Model
.Stack
.Data
.Code
```

1, Khai báo quy mô sử dụng bộ nhớ (.Model):

- cú pháp: .model <kiểu kích thước bộ nhớ>
- Dùng để mô tả kích thước đoạn mã và đoạn dữ liệu của chương trình:
  - + Tiny: Mã lệnh và mã dữ liệu gói gọn trong một đoạn
  - + Small: Mã lệnh gói gọn trong một đoạn, dữ liệu nằm trong một đoạn
  - + Medium: Mã lệnh không gói gọn trong một đoạn, dữ liệu nằm trong một đoạn
  - + Compact: Mã lệnh không gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn
  - + Large: Mã lệnh không gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn, không có mảng nào lớn hơn 64KB
  - + Huge: Mã lệnh không gói gọn trong một đoạn, dữ liệu không gói gọn trong một đoạn, các mảng có thể lớn hơn 64KB
- Với các bài tập hiện tại thì cứ auto small
- VD:

```
.model small;   khai báo kiểu kích thước bộ nhớ là small
.model tiny;    khai báo kiểu kích thước bộ nhớ là tiny
```

2, Khai báo ngăn xếp (.stack):

- cú pháp: .stack <kích thước>
- dùng để khai báo độ lớn ngăn xếp dùng cho chương trình
- nếu không khai báo kích thước, chương trình dịch sẽ tự động gán cho kích thước là 1KB, kích thước này là quá lớn so với nhu cầu bình thường, của các bài toán. Trong thực tế, chỉ cần dùng từ 100-256 byte là đủ nên ta có thể khai báo như sau:

```
.stack 100
```

3, khai báo đoạn dữ liệu (.data):

- cú pháp:

```
.data
<khai báo biến 1>
<khai báo biến 2>
…
```

- tất cả các biến, mảng, xâu đều phải khai báo ở đây. Nên khai báo hằng ở đây dù có thể khai báo hằng trong đoạn mã.
- VD:

```
.data
tb1     DB      ‘moi nhap xau:$’
tb2     DB      ‘xau ma ban vua nhap la:$’
x       DB      ?
CR      DB      13
LF      EQU     10
```

4, khai báo đoạn mã (.code):

- là nơi chứa mã lệnh của chương trình. gồm chương trình chính và các chương trình con (nếu có).
- Trong đoạn mã sẽ bao gồm các thủ tục, được khai báo bởi 2 lệnh giả PROC và ENDP. Lệnh PROC để bắt đầu và ENDP để kết thúc. Một chương trình chính được khai báo theo mẫu như sau:

```
<tên CTC> PROC
; code nằm ở đây
CALL <tên chương trình con>; gọi chương trình con
<tên CTC> ENDP
```

- Một chương trình con được khai báo như sau:

```
<tên ctc> PROC
;code nằm ở đây
RET;    kết thúc chương trình con
<tên ctc> ENDP
```

5, Khung cơ bản của một chương trình dịch ra chương trình .EXE

- VD:

```
.model small
.stack 100
.data
        ;khai báo các biến và hằng
.code
MAIN PROC
        ;khởi tạo DS
        MOV AX, @data
        MOV DS, AX
        ;code nằm ở đây
        ;kết thúc chương trình
        MOV AH, 4CH
        INT 21H
MAIN ENDP
        ;các chương trình con (nếu có) nằm ở đây
```

END MAIN

- trong đó, đoạn:

```
MOV AX, @data
MOV DS, AX
```

có tác dụng nạp các biến đã được khai báo trong đoạn dữ liệu (.code) vào thanh ghi đoạn DS. Vì lý do kỹ thuật, không thể gán trực tiếp @data vào trong DS. vì vậy, cần phải sử dụng thanh ghi đa năng AX làm trung gian (có thay thay AX bằng thanh ghi khác).

XI, MỘT SỐ BÀI TẬP VÍ DỤ (BẮT ĐẦU TỪ TRANG SAU)

VD1: Nhập vào 1 xâu từ bàn phím và in ra xâu đó:

```
.model small
.stack 100
.data
    tb1 DB 'moi ban nhap xau: $'
    tb2 DB 'xau ban vua nhap la: $'
    CRLF DB 13, 10, '$'
    xau DB ?
.code
main proc
    mov ax, @data
    mov ds, ax
    mov ah, 9
    lea dx, tb1
    int 21h
    lea si, xau
    mov ah, 1
doc:
    int 21h
    cmp al, 13
    je tiep
    mov [si], al
    inc si
    jmp doc
tiep:
    mov [si], '$'
    mov ah, 9
    lea dx, CRLF
    int 21h
    lea dx, tb2
    int 21h
    lea dx, xau
    int 21h
    mov ah, 4ch
    int 21h
main endp
end main
```

VD2: Nhập vào một xâu, kiểm tra xem xâu đó có đối xứng hay không

```
.model small
.stack 100
.data
    tb1 DB 'moi ban nhap xau: $'
    dxung DB 'xau ban vua nhap doi xung$'
    kdxung DB 'xau ban vua nhap khong doi xung$'
    CRLF DB 13, 10, '$'
    xau DB ?
.code
main proc
    mov ax, @data
    mov ds, ax
    mov ah, 9
    lea dx, tb1
    int 21h
    mov cx, 0
    lea si, xau
    mov ah, 1
doc:
    int 21h
    cmp al, 13
    je tiep
    mov [si], al
    inc si
    inc cx
    jmp doc
tiep:
    mov ah, 9
    lea dx, CRLF
    int 21h
    lea di, xau
    add di, cx
    dec di
    lea si, xau
kiemtra:
    mov al, [si]
    mov bl, [di]
    cmp al, bl
    jne khongdoixung
    inc si
    dec di
    cmp si, di
    jge doixung
    jmp kiemtra
doixung:
    mov ah, 9
```

```
    lea dx, dxung
    int 21h
    jmp ketthuc
khongdoixung:
    mov ah, 9
    lea dx, kdxung
    int 21h
    jmp ketthuc
ketthuc:
    mov ah, 4ch
    int 21h
main endp
end main
```

VD3: Nhập vào một xâu, in ra xâu đã in hoa các chữ cái thường (số và ký tự đặc biệt giữ nguyên)

```
.model small
.stack 100
.data
    tb1 DB 'moi ban nhap xau: $'
    tb2 DB 'xau cua ban sau khi in hoa: $'
    CRLF DB 13, 10, '$'
    xau DB ?
.code
main proc
    mov ax, @data
    mov ds, ax
    mov ah, 9
    lea dx, tb1
    int 21h
    mov cx, 0
    lea si, xau
    mov ah, 1
doc:
    int 21h
    cmp al, 13
    je tiep
    mov [si], al
    inc si
    inc cx
    jmp doc
tiep:
    mov ah, 9
    lea dx, CRLF
    int 21h
    lea dx, tb2
    int 21h
    mov ah, 2
    lea si, xau
duyet:
    mov bl, [si]
    cmp bl, 'a'
    jl tieptuc
    cmp bl, 'z'
    jg tieptuc
    sub bl, 32
tieptuc:
    mov dl, bl
    int 21h
    dec cx
    cmp cx, 0
    je ketthuc
```

```
    inc si
    jmp duyet
ketthuc:
    mov ah, 4ch
    int 21h
main endp
end main
```

VD4: nhập vào 2 xâu A, B. Kiểm tra xem xâu B có tồn tại trong xâu A không.

```
.model small
.stack 100
.data
    tb1 DB 'nhap xau A: $'
    tb2 DB 'nhap xau B: $'
    tt DB 'xau B co ton tai trong xau A$'
    ktt DB 'xau B khong ton tai trong xau A$'
    a DB 100 DUP(?)
    b DB 100 DUP(?)
    CRLF DB 13, 10, '$'
.code
main proc
    mov ax, @data
    mov ds, ax
    lea dx, tb1
    call announce
    lea si, a
    call doc
    call endl
    lea dx, tb2
    call announce
    lea si, b
    call doc
    call endl
    lea si, a
chuathay:
    lea di, b
dathay:
    mov bl, [si]
    mov bh, [di]
    cmp bh, 24h
    je tontai
    cmp bl, 24h
    je khongtontai
    inc si
    cmp bl, bh
    jne chuathay
    inc di
    jmp dathay
khongtontai:
    lea dx, ktt
    call announce
    jmp ketthuc
tontai:
    lea dx, tt
    call announce
ketthuc:
```

```
    mov ah, 4ch
    int 21h
main endp
announce proc
    mov ah, 9
    int 21h
    ret
announce endp
endl proc
    mov ah, 9
    lea dx, CRLF
    int 21h
    ret
endl endp
doc proc
    mov ah, 1
doctiep:
    int 21h
    cmp al, 13
    je het
    mov [si], al
    inc si
    jmp doctiep
het:
    mov [si], '$'
    ret
doc endp
end main
```

VD5: Nhập vào một số nhị phân (BCD), in ra dạng thập lục phân (HEXA) của số đó.

```
.model small
.stack 100
.data
    tb1 DB 'moi nhap so (BCD): $'
    tb2 DB 'dang thap luc phan cua so do la: $'
    tb3 DB 'sai dinh dang. Yeu cau nhap lai.$'
    CRLF DB 13, 10, '$'
.code
main proc
batdau:
    mov ax, @data
    mov ds, ax
    mov ah, 9
    lea dx, tb1
    int 21h
    mov ah, 1
    xor bx, bx
doc:
    int 21h
    cmp al, 13
    je docxong
    cmp al, 30h
    jl saidinhdang
    cmp al, 31h
    jg saidinhdang
    sub al, 30h
    shl bx, 1
    or bl, al
    jmp doc
saidinhdang:
    mov ah, 9
    lea dx, CRLF
    int 21h
    lea dx, tb3
    int 21h
    lea dx, CRLF
    int 21h
    jmp batdau
docxong:
    mov ah, 9
    lea dx, CRLF
    int 21h
    lea dx, tb2
    int 21h
    mov cx, 4
matna:
    rol bx, 1
```

```
    rol bx, 1
    rol bx, 1
    rol bx, 1
    mov ax, bx
    and ax, 000fh
    cmp al, 10
    jl laso
    add al, 37h
    jmp inra
laso:
    add al, 30h
    jmp inra
inra:
    mov ah, 2
    lea dl, al
    int 21h
    dec dh
    cmp dh, 00h
    loop matna
ketthuc:
    mov ah, 4ch
    int 21h
main endp
end main
```

VD6: Nhập vào xâu X và 2 số a, b. in ra xâu con của X từ vị trí a đến vị trí b (x[a] -> x[b - 1]) (chưa hoàn thành)

```
.model small
.stack 100
.data
    tb1 DB 'nhap xau x: $'
    tb2 DB 'nhap so a: $'
    tb3 DB 'nhap so b $'
    tb4 DB 'ket qua: $'
    CRLF 13, 10, '$'
    a DB ?
    b DB ?
.code
main proc
    mov ax, @data
    mov ds, ax
    mov ah, 9
    lea dx, tb1
    int 21h


    call endl
    lea dx, tb2
    int 21h

main endp
endl proc
    push ah
    mov ah, 9
    lea dx, CRLF
    int 21h
    pop ah
endl endp
end main
```

XII, BÀI TẬP ÔN TẬP:

B1:

1. Viết chương trình hợp ngữ cho phép nhập số nhị phân (<16 bit) chứa vào trong BX. Chương trình phải kiểm tra ký tự nhập có hợp lệ hay không. Việc nhập kết thúc khi nhấn Enter hoặc đủ 16 bit. Xuất ra số đã nhập dưới dạng thập phân.

- Lưu đồ thuật toán:

Flowchart Maker & Online Diagram Software

- Code:

```
.model small
.stack 100
.data
    str1 DB 'Nhap so nhi phan: $'
    str2 DB 'Ket qua: $'
    str3 DB 'Sai dinh dang roi ban ei$'
    CRLF DB 10, 13, '$'
.code
main proc
    mov ax, @data
    mov ds, ax
    mov bx, 0
    mov dl, 0
    mov ah, 9
    lea dx, str1
    int 21h
DOC:
    mov ah, 1
    int 21h
    cmp al, 13
    je DOCXONG
    cmp al, '0'
    jb ERROR
    cmp al, '1'
    ja ERROR
    sub al, '0'
    inc dl
    shl bx, 1
    or bl, al
    cmp dl, 16
    je DOCXONG
    jmp DOC
DOCXONG:
    mov ah, 9
    lea dx, CRLF
```

```
    int 21h
    lea dx, str2
    int 21h
    mov ax, bx
    mov bx, 10
    mov cx, 0
    xor dx, dx
XULY:
    div bx
    inc cx
    push dx
    xor dx, dx
    cmp ax, 0
    jz INRA
    jmp XULY
INRA:
    mov ah, 2
    pop dx
    add dx, '0'
    int 21h
    loop INRA
    jmp KETTHUC
ERROR:
    mov ah, 9
    lea dx, CRLF
    int 21h
    lea dx, str3
    int 21h
KETTHUC:
    mov ah, 4ch
    int 21h
main endp
end main
```

B2:

2. Viết chương trình hợp ngữ đếm số lần xuất hiện của chuỗi “vixuly” trong một chuỗi. In kết quả ra màn hình dưới dạng số thập phân.

- Lưu đồ thuật toán: Trang 2 của link bài 1
- Code:

```
.model small
.stack 100
.data
    str1 DB 'hom nay troi dep muon hoc vixuly qua di, nhung ma chi duoc 2d vixulu$'
    str2 DB 'vixuly$'
    dem DB 0
.code
main proc
    mov ax, @data
    mov ds, ax

    lea si, str1
    lea di, str2
DUYET:
    mov bl, [si]
    cmp bl, [di]
    je TIMTHAY
KOTIMTHAY:
    inc si
    cmp [si], '$'
    je XONG
    jmp DUYET

TIMTHAY:
    inc di
    cmp [di], '$'
    je TANG
    inc si
    cmp [si], '$'
    je XONG
    mov bl, [si]
    cmp bl, [di]
    je TIMTHAY
    lea di, str2
    jmp KOTIMTHAY
TANG:
    inc dem
    lea di, str2
    jmp DUYET
XONG:
```

```
    xor ax, ax
    mov al, dem
    mov bl, 10
    mov cx, 0
TIEPTUC:
    div bl
    inc cx
    push ax
    xor ah, ah
    cmp ax, 0
    jz INRA
    jmp TIEPTUC
INRA:
    pop ax
    add ah, '0'
    mov dl, ah
    mov ah, 2
    int 21h
    loop INRA
KETTHUC:
    mov ah, 4ch
    int 21h
main endp
end main
```

B3:

3. Viết chương trình hợp ngữ cho phép tính ước số chung lớn nhất và bội số chung nhỏ nhất của hai số nhập vào từ bàn phím. In kết quả thu được ra màn hình dưới dạng thập phân.

-Lưu đồ: người viết nhác éo vẽ nữa
- code:

```
.model small
.stack 100
.data
    a DW ?
    b DW ?
    uc DW ?
    CRLF DB 10, 13, '$'
    inputA DB 'moi nhap so thu nhat: $'
    inputB DB 'moi nhap so thu hai: $'
    gcd DB 'UCLN = $'
    lcm DB 'BCNN = $'
    er DB 'Sai dinh dang so$'
.code
main proc
    mov ax, @data
    mov ds, ax

    mov ah, 9
    lea dx, inputA
    int 21h
    call nhap
    mov a, ax
    call endline
    mov ah, 9
    lea dx, inputB
    int 21h
    call nhap
    mov b, ax
    call endline
    mov ah, 9
    lea dx, gcd
    int 21h
    call UCLN
    call endline
    mov ah, 9
    lea dx, lcm
    int 21h
    call BCNN
```

```
    mov ah, 4ch
    int 21h
main endp
;-------------------------
nhap proc
    ;Vao : ban phim (thap phan -> nhi phan)
    ;Ra :thanh ghi ax
    push bx
    mov bx, 0
tieptuc:
    mov ah, 1
    int 21h
    cmp al, 13
    je xong
    cmp al, '0'
    jb ERROR
    cmp al, '9'
    ja ERROR
    and ax, 000fh
    push ax
    mov ax, 10
    mul bx
    mov bx, ax
    pop ax
    add bx, ax
    jmp tieptuc
ERROR:
    call endline
    mov ah, 9
    lea dx, er
    int 21h

    mov ah, 4ch
    int 21h
xong:
    mov ax, bx
    pop bx
    ret
nhap endp
;-------------------------
UCLN proc
    ;Vao : a, b
    ; Ra :  Man hinh + thanh ghi ax
    mov ax, a
    mov bx, b
    cmp ax, 0
    jz bangkhong
    cmp bx, 0
```

```
    jz bangkhong
trutiep:
    cmp ax, bx
    ja atrub
    jb btrua
    mov uc, ax
    call printNumber
    jmp thoat
bangkhong:
    add ax, bx
    mov uc, ax
    call printNumber
    jmp thoat
atrub:
    sub ax, bx
    jmp trutiep
btrua:
    sub bx, ax
    jmp trutiep
thoat:
    ret
UCLN endp
;------------------------
BCNN proc
    ;Vao : a, b, UCLN
    ;Ra: Man hinh
    mov ax, a
    mov bx, b
    mul bx
    mov cx, uc
    div cx
    call printNumber
    ret
BCNN endp
;------------------------
printNumber proc
    ;Vao: ax chua gia tri can bien doi
    ;Ra: Man hinh (ax duoi dang nhi phan)
    push bx
    push cx
    push dx

    mov bx, 10
    mov cx, 0
    xor dx, dx
XULY:
    div bx
    inc cx
```

```
    push dx
    xor dx, dx
    cmp ax, 0
    jz INRA
    jmp XULY
INRA:
    mov ah, 2
    pop dx
    add dx, '0'
    int 21h
    loop INRA
    pop dx
    pop cx
    pop bx
    ret
printNumber endp
;-------------------------
endline proc
    push ax
    push dx

    mov ah, 9
    lea dx, CRLF
    int 21h

    pop dx
    pop ax
    ret
endline endp
end main
```

B4:

4. Viết chương trình hợp ngữ biến đổi chuỗi ký tự thường thành chuỗi ký tự hoa. In ra màn hình cả hai chuỗi thu được.

- Code:

```
.model small
.stack 100
.data
    str1 DB 'moi nhap xau: $'
    str2 DB 'ket qua: $'
    str3 DB 100 DUP(?)
    CRLF DB 10, 13, '$'
.code
main proc
    mov ax, @data
    mov ds, ax

    lea si, str3
    lea dx, str1
    call printString
doctiep:
    mov ah, 1
    int 21h
    cmp al, 13
    je docxong
    cmp al, 'a'
    jb khongphai
    cmp al, 'z'
    ja khongphai
    sub al, 20h ; al = al - 'a' + 'A'
khongphai:
    mov [si], al
    inc si
    jmp doctiep
docxong:
    mov [si], '$'
    call endline
    lea dx, str2
    call printString
    lea dx, str3
    call printString
;ketthuc
    mov ah, 4ch
    int 21h
main endp
;--------------------
```

```
endline proc
    push ax
    push dx
    mov ah, 9
    lea dx, CRLF
    int 21h
    pop dx
    pop ax
    ret
endline endp
;--------------------
printString proc
    push ax
    mov ah, 9
    int 21h
    pop ax
    ret
printString endp
end main
```

B5:

5. Viết chương trình hợp ngữ tính giai thừa của một số nhập vào từ bàn phím. In kết quả ra màn hình dưới dạng thập phân.

- Code:

```
.model small
.stack 100
.data
    str1 DB 'moi nhap x: $'
    str2 DB 'x! = $'
    Er DB 'Sai dinh dang so$'
    CRLF DB 10, 13, '$'
.code
main proc
    mov ax, @data
    mov ds, ax
    lea dx, str1
    call printString
    call nhap
    call endline
    lea dx, str2
    call printString
    call calculate
    call printNumber
    mov ah, 4ch
    int 21h
main endp
;---------------------
nhap proc
    ;vao: ban phim
    ;ra: ax
    push bx
    push dx
    xor bx, bx
tieptuc:
    mov ah, 1
    int 21h
    cmp al, 13
    je thoat
    cmp al, '0'
    jb error
    cmp al, '9'
    ja error
    and ax, 000fh
    push ax
    mov ax, 10
```

```
    mul bl
    mov bx, ax
    pop ax
    add bx, ax
    jmp tieptuc
error:
    mov ah, 9
    lea dx, Er
    int 21h
    mov ah, 4ch
    int 21h
thoat:
    mov ax, bx
    pop dx
    pop bx
    ret
nhap endp
;--------------------
calculate proc
    ;vao: ax
    ;ra: ax
    push cx
    push dx
    xor dx, dx
    mov cx, ax
    mov ax, 1
tinhtiep:
    mul cx
    loop tinhtiep
    pop dx
    pop cx
    ret
calculate endp
;--------------------
endline proc
    push ax
    push dx
    mov ah, 9
    lea dx, CRLF
    int 21h
    pop dx
    pop ax
    ret
endline endp
;--------------------
printNumber proc
    ;vao: ax
    ;ra: man hinh
```

```
    push bx
    push cx
    push dx
    mov bx, 10
    mov cx, 0
    xor dx, dx
XULY:
    div bx
    inc cx
    push dx
    xor dx, dx
    cmp ax, 0
    jz INRA
    jmp XULY
INRA:
    mov ah, 2
    pop dx
    add dx, '0'
    int 21h
    loop INRA
    pop dx
    pop cx
    pop bx
    ret
printNumber endp
;--------------------
printString proc
    push ax
    mov ah, 9
    int 21h
    pop ax
    ret
printString endp
end main
```

B6:

6. Viết chương trình hợp ngữ tính tổng các số chia hết cho 7 trong một mảng cho trước. In giá trị thu được ra màn hình dưới dạng số hexa.

- Code:

```
.model small
.stack 100
.data
   CRLF DB 10, 13, '$'
   mang DW 10, 14 , 28, 6, 7 , 98, 1, 7
.code
main proc
   mov ax, @data
   mov ds, ax
   mov cx, 8   ;so luong phan tu cua mang
   mov bx, 0   ;tong gia tri cac ptu % 7 == 0
   lea si, mang
   mov dl, 7
   cld         ;DF = 0 -> si += 1
tieptuc:
   lodsw       ;al = [si]; si += 1
   push ax
   div dl
   cmp ah, 0
   jnz boqua
   pop ax
   add bx, ax
boqua:
   loop tieptuc
   mov ax, bx
   call printNumberHex
;ket thuc
   mov ah, 4ch
   int 21h
main endp
;-----------------------
printNumberHex proc
   ;vao: ax
   ;ra: man hinh
   push bx
   push cx
   push dx
   mov cl, 4
   mov ch, 4
tachtiep:
   mov bx, ax
```

```
    and bx, 000fh
    cmp bl, 9
    ja khongphaiso
    add bl, '0'
    push bx
    jmp kiemtra
khongphaiso:
    sub bl, 10
    add bl, 'A'
    push bx
    jmp kiemtra
kiemtra:
    ror ax, cl
    dec ch
    cmp ch, 0
    jnz tachtiep

    mov cx, 4
    mov ah, 2
inra:
    pop bx
    mov dl, bl
    int 21h
    loop inra
    pop dx
    pop cx
    pop bx
    ret
printNumberHex endp

end main
```