Chương 7:
HỢP NGỮ

Nội dung môn học

- Chương 1: Giới thiệu chung
- Chương 2: Tổng quan về hệ thống máy tính
- Chương 3: Biểu diễn dữ liệu và số học
- Chương 4: Đơn vị xử lý trung tâm
- Chương 5: Bộ nhớ máy tính
- Chương 6: Hệ thống vào ra
- Chương 7: Hợp ngữ

Nội dung chương

- Cơ bản về hợp ngữ
- Lệnh nhảy
- Lệnh lặp
- Ngăn xếp
- Thủ tục

CƠ BẢN VỀ HỢP NGỮ

- Máy ngữ và hợp ngữ
- Thanh ghi CPU 8086/8088
- Viết chương trình hợp ngữ
- Một số lệnh đơn giản

Máy ngữ và hợp ngữ

- Chương trình là một tập lệnh được đưa vào bộ nhớ cho CPU thực hiện
- Dạng cơ bản nhất của lệnh mà máy có thể hiểu ngay gọi là Máy Ngữ hay Ngôn ngữ máy (Machine Languge)
- Đặc điểm ngôn ngữ máy:
  - Thực hiện rất nhanh
  - Chiếm ít bộ nhớ
  - Khó viết, khó nhớ:

Máy ngữ và hợp ngữ

- Một đoạn chương trình máy ngữ thuộc họ 80x86

Ngôn ngữ máy		Hợp ngữ	
5601:0100	B4.02	MOV	AH, 2
5601:0102	80.C2.30	ADD	DL,30
5601:0105	50	PUSH	AX

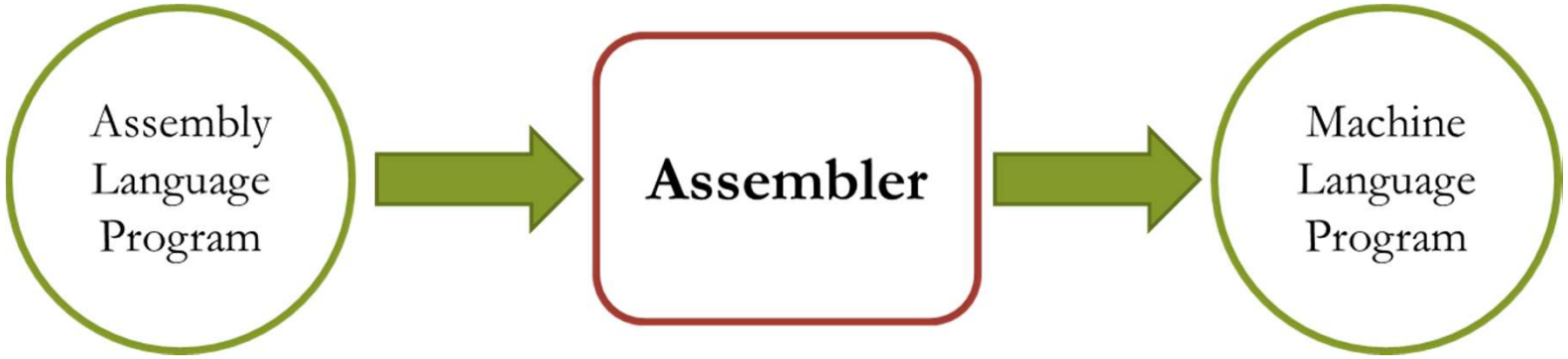
Máy ngữ và hợp ngữ

- Hợp ngữ (Assembly Language)
  - Giúp viết chương trình dễ hơn thay cho ngôn ngữ máy
  - Có dạng giống ngôn ngữ máy

    1 Lệnh hợp ngữ ⇔ 1 Lệnh máy
  - Dùng các ký hiệu tượng trưng dễ nhớ hơn Ngôn ngữ máy

Máy ngữ và hợp ngữ

- Hợp ngữ (Assembly Language)
  - Chương trình hợp ngữ phải được **DỊCH** sang ngôn ngữ máy để máy có thể hiểu và thực hiện
  - Chương trình dịch gọi là Trình hợp dịch (Assembler)



Source Program

One by one translation

Object Program

Máy ngữ và hợp ngữ

Các bước soạn và chạy chương trình hợp ngữ với MS DOS hay Command Prompt		
Bước 1	Soạn chương trình hợp ngữ. Phần mở rộng là .asm	Dùng phần mềm soạn thảo văn bản như notepad,…
Bước 2	Dịch chương trình	Dùng MASM để dịch chương trình nguồn .asm, tạo file object
Bước 3	Liên kết chương trình	Dùng LINK để liên kết object tạo tập tin thực hiện .EXE
Bước 4	Thực hiện chương trình	Gõ lệnh thực hiện tập tin .EXE

Máy ngữ và hợp ngữ

- Chương trình hợp ngữ: Bao gồm các phát biểu hợp ngữ (assembly language instruction)
  - Lệnh hợp ngữ: lệnh nhị phân/mã máy
  - Chỉ dẫn biên dịch (assembler directive)
    - Không có lệnh nhị phân tương ứng
    - Tuân theo cú pháp của assembler

Máy ngữ và hợp ngữ

- Dạng tổng quát của một lệnh hợp ngữ gồm 4 trường sau:

[label:] mnemonic [operand1 , operand2] [; comment]

- Nhãn thay thế địa chỉ câu lệnh.
- Người lập trình tự đặt

- Từ gợi nhớ xác định hành động của câu lệnh
- Tra trong tập lệnh

- Các toán hạng, ngăn cách bởi dấu ,

Ghi chú

Máy ngữ và hợp ngữ

- Ví dụ:

  DoAddition: ADD AX, DX ; AX ← AX + DX

- Ta có:
  - Nhãn là DoAddition với dấu hai chấm theo sau
  - Từ gợi nhớ: ADD
  - Toán hạng: AX và DX
  - Ghi chú: AX ← AX + DX

CƠ BẢN VỀ HỢP NGỮ

- Máy ngữ và hợp ngữ
- Thanh ghi CPU 8086/8088
- Viết chương trình hợp ngữ
- Một số lệnh đơn giản

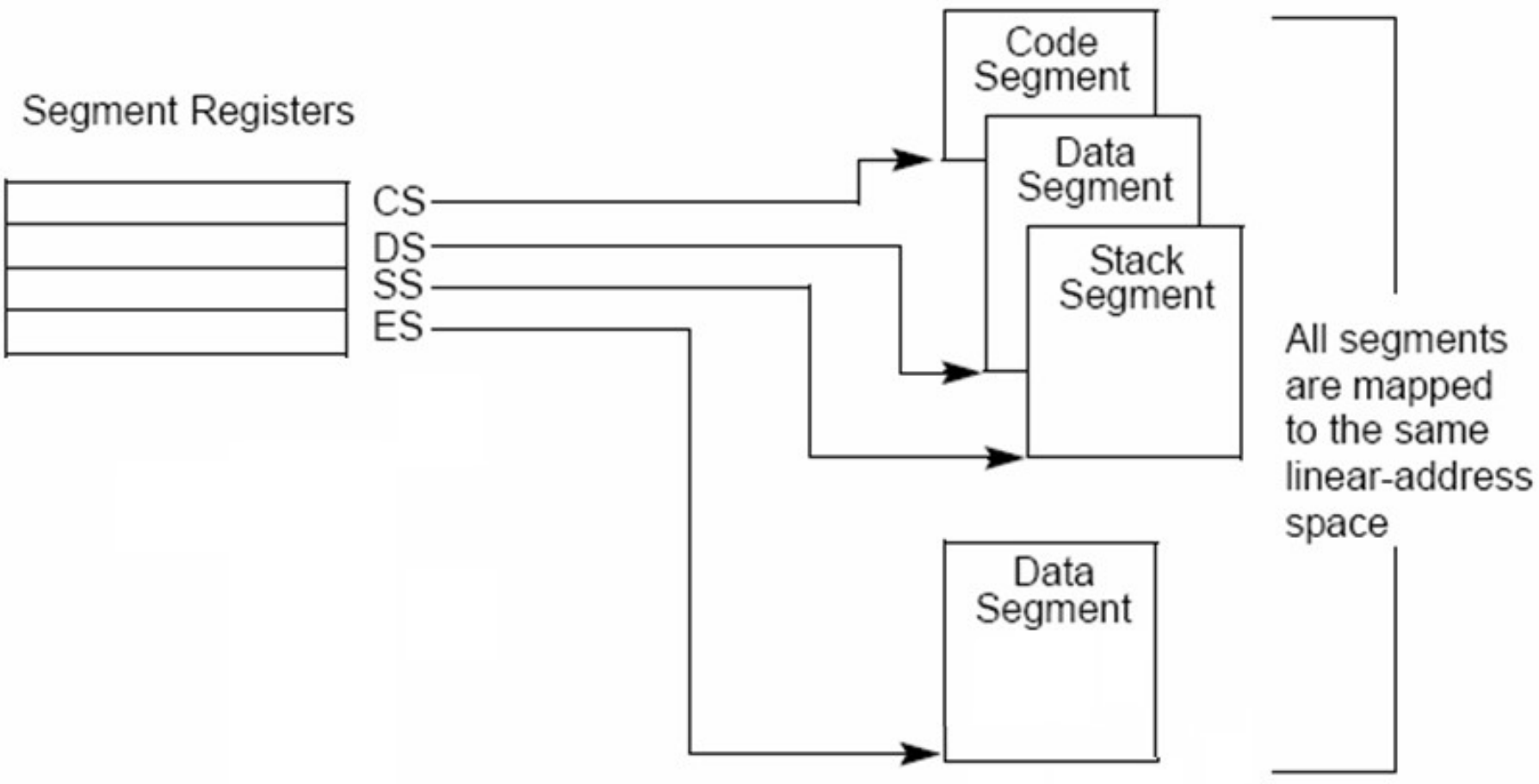
Thanh ghi của CPU 8086

- Gồm 14 thanh ghi, mỗi thanh ghi 16 bit
- Chia làm 5 nhóm:
  - Nhóm thanh ghi đoạn
  - Nhóm thanh ghi đa dụng
  - Nhóm thanh ghi con trỏ và chỉ mục
  - Thanh ghi con trỏ lệnh
  - Thanh ghi cờ hiệu

Nhóm 1: các thanh ghi đoạn

- Segment registers
- Chứa địa chỉ đoạn (segment)
- 4 thanh ghi:
  - CS (Code segment register)
  - DS (Data segment register)
  - ES (Extra data segment register)
  - SS (Stack segment register)

Quản lý bộ nhớ với thanh ghi đoạn



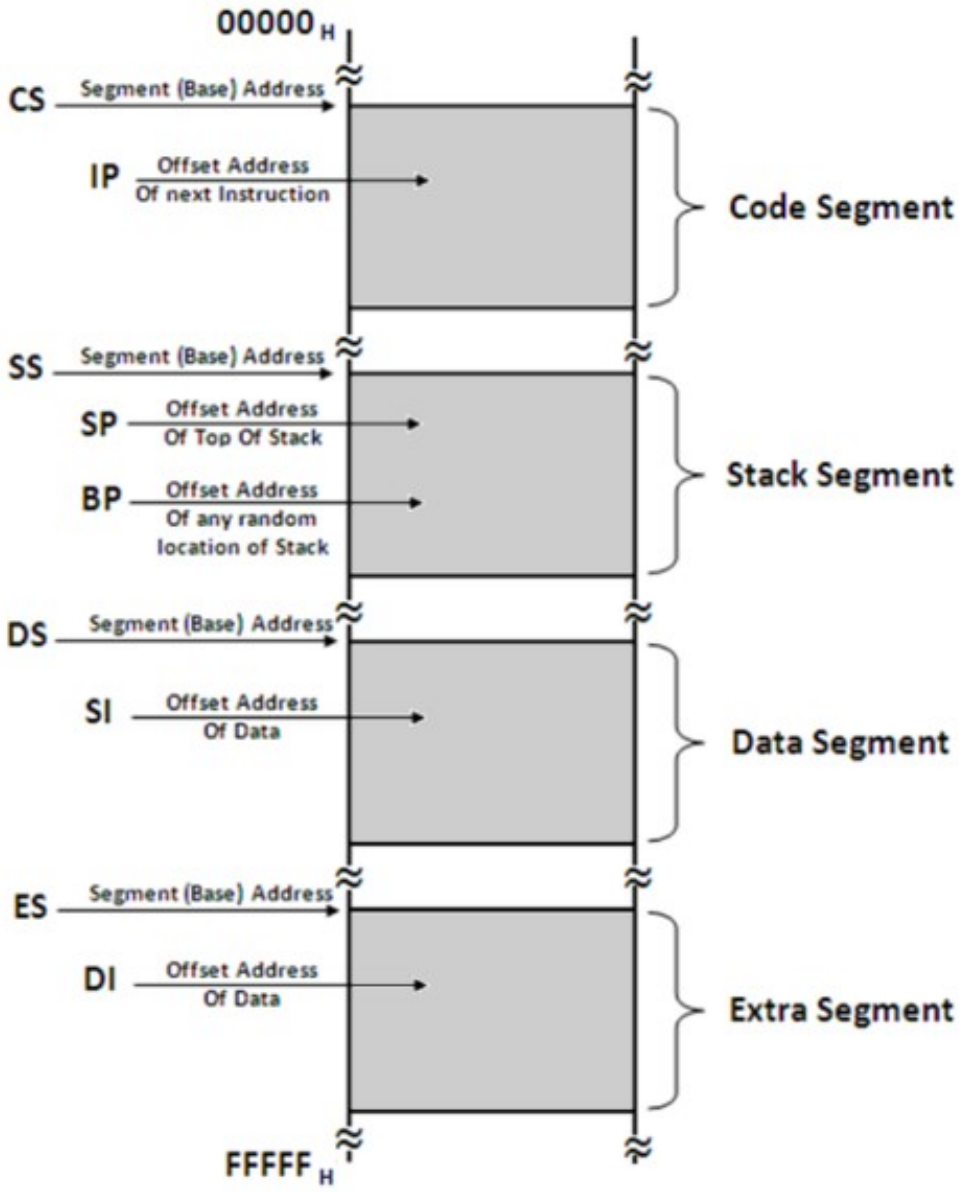
Nhóm 1: các thanh ghi đoạn

- Thanh ghi đoạn : Thanh ghi lệch (segment:offset)

Địa chỉ vật lý = Thanh ghi đoạn x 16 + Thanh ghi lệch

- CS (code segment, 16 bit): phối hợp với con trỏ lệnh IP để ghi địa chỉ mã lệnh trong bộ nhớ (CS:IP).
- DS (data segment, 16 bit): phối hợp với 2 thanh ghi chỉ số SI và DI để đánh địa chỉ cho dữ liệu.
  - Dữ liệu cần đọc vào là DS:SI
  - Dữ liệu cần ghi ra là DS:DI.
- SS (stack segment, 16 bit) : địa chỉ đỉnh của ngăn xếp được biểu diễn cùng với con trỏ ngăn xếp SP là SS:SP.
- ES (extra segment, 16 bit): dùng để đánh địa chỉ một chuỗi.
  - ES:DI là địa chỉ chuỗi cần viết đến (chuỗi đích)
  - ES:SI là địa chỉ chuỗi đọc vào (chuỗi nguồn).

MEMORY SEGMENTATION IN 8086



Nhóm 1: các thanh ghi đoạn

- Thanh ghi đoạn : Thanh ghi lệch (segment:offset)

Địa chỉ vật lý = Thanh ghi đoạn x 16 + Thanh ghi lệch

- Ví dụ:

Theo hệ thập phân (decimal)

Địa chỉ vật lý 123789 có địa chỉ segment là 7736 và địa chỉ offset là 13

Theo hệ thập lục (hexa)

Địa chỉ vật lý 1E38D có địa chỉ segment là 1E38 và địa chỉ offset là D

Nhận xét: theo hệ hexa, để tính địa chỉ vật lý thì chỉ cần thêm 0 vào cuối địa chỉ đoạn sau đó cộng địa chỉ offset

Tính địa chỉ vật lý của ngăn nhớ có địa chỉ segment và offset như sau:

a) segment:offset như sau: 1134h:1023h

b) segment:offset như sau 020Ah:1BCDh

Nhóm 2: các thanh ghi đa dụng

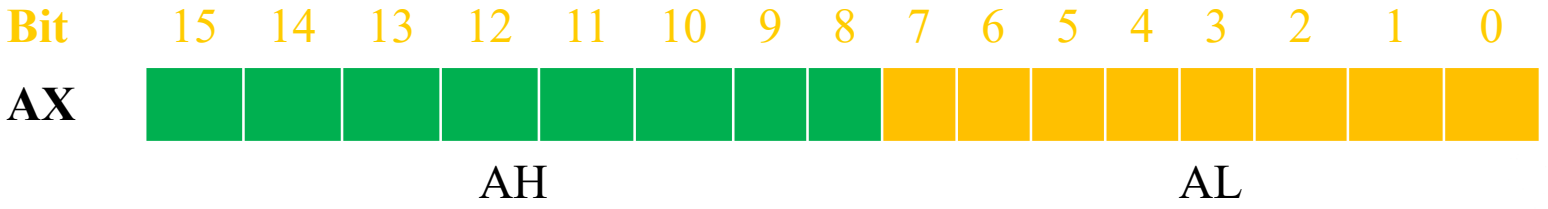
- General purpose registers: Chứa các toán hạng, dữ liệu
- Gồm 4 thanh ghi 16 bit

- AX (accumulator, 16 bit): chứa kết quả các thao tác lệnh.
- BX (base, 16 bit) : chứa địa chỉ cơ sở của một bảng trong lệnh XLAT.
- CX (count, 16 bit): chứa số lần lặp trong trường hợp các lệnh LOOP. *Thanh ghi thấp CL được dùng để chứa (nhớ) số lần quay hoặc dịch của các lệnh quay và dịch.*
- DX (data, 16 bit): tham gia vào các thao tác của phép nhân hoặc cùng thanh ghi các số 16 bit. *DX còn dùng để chứa địa chỉ 16 bit của các cổng cứng (dài hơn 8 bit) trong các lệnh truy nhập các cổng ngoại vi.*

Nhóm thanh ghi đa dụng (t.t)

- Mỗi thanh ghi đa dụng 16 bit có thể được sử dụng như 1 cặp thanh ghi 8 bit

Thanh ghi 16bit	8bit cao	8 bit thấp
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

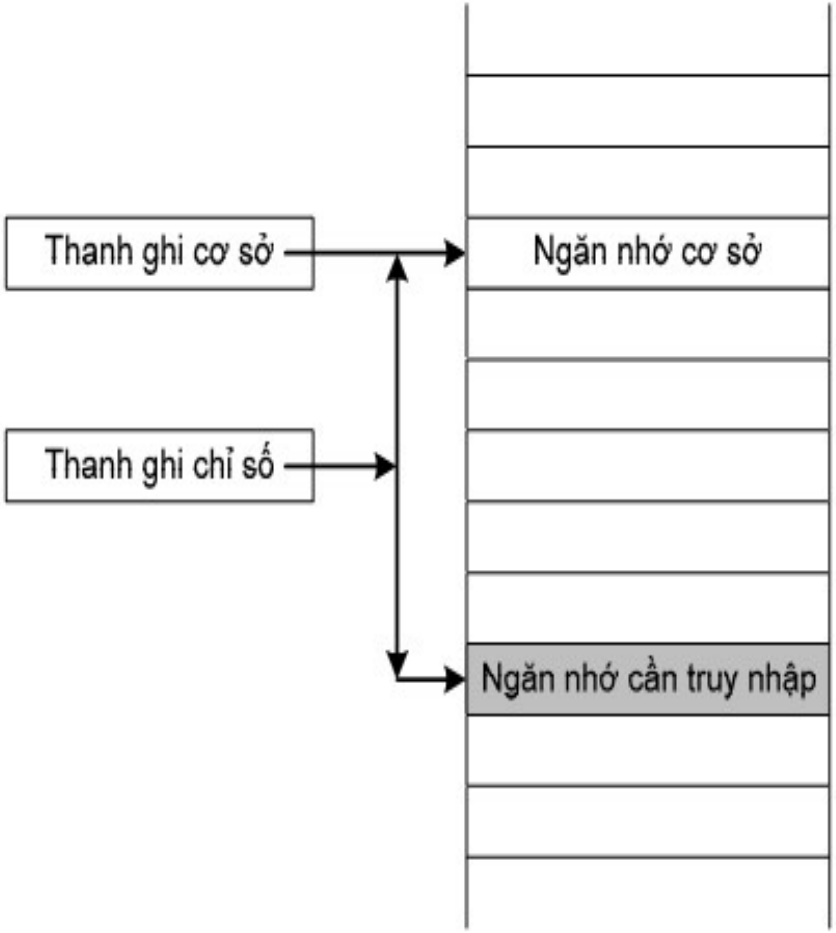


Nhóm 3: Các thanh ghi con trỏ và chỉ mục

- Gồm 4 thanh ghi
  - SP (Stack Pointer register): thanh ghi con trỏ
  - BP (Base Pointer register): thanh ghi cơ sở
  - SI (Source Indexed register): thanh ghi chỉ số nguồn
  - DI (Destination Indexed register): thanh ghi chỉ số đích

Thanh ghi cơ sở và thanh ghi chỉ số

- Thanh ghi cơ sở: chứa địa chỉ của ngăn nhớ cơ sở (địa chỉ cơ sở)
- Thanh ghi chỉ số: chứa độ lệch địa chỉ giữa ngăn nhớ mà CPU cần truy nhập so với ngăn nhớ cơ sở (chỉ số)
- Địa chỉ của ngăn nhớ cần truy nhập = địa chỉ cơ sở + chỉ số



Nhóm 4: Thanh ghi con trỏ chương trình

- IP (Instruction Pointer register)
- Lệnh kế tiếp sẽ thực hiện ở địa chỉ:

  CS:IP

Nhóm 5: Thanh ghi cờ hiệu

- Flag register
- Gồm các cờ hiệu

CF: Carry Flag

AF: Auxiliary Flag

ZF: Zero Flag

OF: Overflow Flag

SF: Signed Flag

PF: Parity Flag

DF: Direction Flag

TF: Trap Flag

IF: Interupt Enable Flag

Viết chương trình hợp ngữ

❖ Trường tên:

- Tên hằng, tên biến, tên nhãn… do người lập trình đặt ra
- Bắt đầu của tên không là số
- Tên không được trùng với từ khóa
- Tên tối đa 31 ký tự
- Không phân biệt hoa – thường

Viết chương trình hợp ngữ

❖ Trường tên:

- Tên hằng, tên biến, tên nhãn… do người lập trình đặt ra
- Bắt đầu của tên không là số
- Tên không được trùng với từ khóa
- Tên tối đa 31 ký tự
- Không phân biệt hoa – thường

Viết chương trình hợp ngữ

❖Trường tên:

Ví dụ:

Nhãn hợp lệ:	Nhãn không hợp lệ
MainLoop	THONG BAO
Calc_long_sum	1TB
Error	X30.1
Iterate	HO&TEN
Delay_100_seconds	

Viết chương trình hợp ngữ

- **Tên gợi nhớ của lệnh (mnemonic):**
  - Nếu là chỉ thị như MOV, ADD,…,thì lệnh sẽ được dịch sang mã máy
  - Nếu là chỉ dẫn như END, PROC,…,thì đó là lệnh hướng dẫn trình hợp dịch trong quá trình dịch chương trình sang mã máy

Viết chương trình hợp ngữ

❖ Trường toán hạng (operand)

- Xác định dữ liệu sẽ được xử lý bởi lệnh
- Lệnh có thể không có toán hạng, có một hoặc nhiều toán hạng
- Các toán hạng cách nhau dấu phẩy “,”
- Trong lệnh có 2 toán hạng: toán hạng thứ nhất (từ trái qua) gọi là toán hạng đích, toán hạng thứ hai gọi là toán hạng nguồn. VD: ADD AX, BX

Viết chương trình hợp ngữ

❖ Trường ghi chú:

- Sau mỗi lệnh có thể viết dòng ghi chú sau dấu chấm phẩy ;

  VD: MOV AH,1 ; Nhập phím kế tiếp

- Ngoài ra có thể dùng nguyên một dòng để ghi chú với dấu chấm phẩy ở đầu. VD:

```
;Tạo byte kiểm tổng cho vùng đệm chuyển
MOV BX, OFFSET TransferBuffer
MOV CX, TRANSFER_BUFFER_LENGTH
```

Viết chương trình hợp ngữ

❖ Cách viết số:

- Các số trong chương trình, bình thường được hiểu là thập phân. Khi cần có thể thêm chữ D hoặc d đằng sau số. VD: 10, 10D, 10d đều có trị là mười
- Số viết theo hệ thập lục kết thúc bằng H hoặc h và phải bắt đầu là một số. VD: 10h, 10H, 2F8H, 2F8h, 0DH, 0Dh
- Số nhị phân kết thúc bằng B hoặc b. VD: 1011b, 1011B
- Số bát phân kết thúc bằng O hoặc o. VD: 100O, 100o

Viết chương trình hợp ngữ

❖ Chuỗi ký tự:

- Ký tự hoặc chuỗi ký tự phải được đặt trong dấu nháy đơn ‘’ hoặc dấu nháy kép “”
- Ví dụ: ‘A’, “A”, ‘ABC’, “ABC”
- Các ký tự sẽ được chuyển thành mã ASCII tương ứng, do đó ‘A’, “A”, 41h hoặc 65 đều có nghĩa như nhau
- Trường hợp ký tự trong chuỗi là ‘ hoặc “ thì dung ký tự rào khác
- Ví dụ: “Don’t forget me!” hoặc “‘Forget me not!”’

Viết chương trình hợp ngữ

❖ Định nghĩa vùng nhớ chứa dữ liệu:

- Các chỉ dẫn thông dụng dùng định nghĩa dữ liệu kiểu Byte, hoặc từ 2 byte (Word) như sau:

```
Nhãn  DB    trị1, trị2,….        ;byte
Nhãn  DW    trị1, trị2,….        ;word
```

Trong đó: Nhãn là tên vùng nhớ (còn gọi là biến)

Trị1, Trị2 sẽ là trị được gán cho vùng nhớ, nếu thay trị bằng ? thì sẽ không gán giá trị. Nếu có nhiều trị liên tiếp thì nhiều vùng nhớ liên tiếp sẽ được cấp phát

Đơn vị: word (2 byte), double word DD (4 byte), quad word (8 byte), paragraph (16 byte)

Viết chương trình hợp ngữ

❖ Định nghĩa vùng nhớ chứa dữ liệu:

Một số ví dụ:

B DB 5 ; cấp phát vùng nhớ có địa chỉ là B, 1 byte, trị là 5

BA DB 1,2,3,4,5 ; cấp phát vùng nhớ có địa chỉ là BA có kích thước 5 byte và có trị lần lượt là 1,2,3,4,5

B DB ? ; cấp phát vùng nhớ có địa chỉ là B, trị không xác định

Viết chương trình hợp ngữ

❖ Định nghĩa vùng nhớ chứa dữ liệu:

- Một số ví dụ:

```
W   DW    10        ; Lưu ý trị cao nằm ở byte cao
W   DW 1234h ; ???
STR   DB    'ABC' ; vùng nhớ được cấp phát có địa chỉ là STR, chiếm 3 byte và có trị là 41h, 42h, 43h
STR   DB    'ABC', 0Dh, 0Ah, '$'          ; ???
```

Viết chương trình hợp ngữ

❖ Định nghĩa vùng nhớ chứa dữ liệu:

- Một số ví dụ:

```
b_array  DB    10h, 20h, 30h
```

Nếu B_ARRAY có địa chỉ offset 0200h thì nội dung bộ nhớ như sau:

SYMBOL	ADDRESS	CONTENTS
b_array	0200h	10h
b_array+1	0201h	20h
b_array+2	0202h	30h

Viết chương trình hợp ngữ

❖ Định nghĩa vùng nhớ chứa dữ liệu:

Một số ví dụ:

b_array DB 6 DUP (0) ; vùng nhớ được cấp phát có địa chỉ là b_array, chiếm 6 byte và đều được gán trị là 0

TIME DW 7 DUP (?) ; vùng nhớ được cấp phát có địa chỉ là TIME, chiếm 14 byte và không được gán trị

Viết chương trình hợp ngữ

❖ Định nghĩa hằng:

- Thay vì viết trực tiếp các hằng số hoặc các chuỗi trong chương trình, ta có thể đặt tên cho hằng ở đầu chương trình. Cú pháp như sau:

  Tên EQU hằng

- Ví dụ:

```
CR    EQU  0Dh
LF    EQU  0Ah
STR   EQU  'Dữ liệu nhập sai!'
```

Viết chương trình hợp ngữ

❖ Cấu trúc chương trình hợp ngữ:

- Chương trình mã máy trong bộ nhớ gồm 3 phần chứa trong 3 đoạn là: đoạn mã, dữ liệu và ngăn xếp.
- Do đó, hợp ngữ cũng được tổ chức tương tự

```
.MODEL SMALL
.STACK 100h
.DATA
        ; phần khai báo dữ liệu
.CODE
        ; phần lệnh
END
```

Viết chương trình hợp ngữ

Đầu tiên là lệnh khai báo kích thước chương trình, gồm đoạn mã và đoạn dữ liệu được xác định bằng chỉ dẫn MODEL như sau:

.MODEL Kiểu

Viết chương trình hợp ngữ

Kiểu	Kích thước
TINY	Mã và dữ liệu nằm trong phạm vi 1 đoạn. Đây là chương trình dạng .COM, lúc đó CS và DS có cùng trị
SMALL	Mã nằm trong một đoạn, dữ liệu nằm trong một đoạn khác. Đây là chương trình dạng .EXE nhỏ nhất
MEDIUM	Mã có thể gồm nhiều đoạn nhưng dữ liệu ở trong phạm vi một đoạn
COMPACT	Mã trong phạm vi một đoạn, dữ liệu có thể gồm nhiều đoạn Mảng dữ liệu liên tục không được lớn hơn một đoạn
LARGE	Mã và dữ liệu có thể gồm nhiều đoạn Mảng dữ liệu liên tục không được lớn hơn một đoạn
HUGE	Mã và dữ liệu có thể gồm nhiều đoạn Mảng dữ liệu liên tục có thể lớn hơn một đoạn

Viết chương trình hợp ngữ

❖ Cấu trúc chương trình hợp ngữ:

- Thứ hai là lệnh khai báo kích thước vùng ngăn xếp với chỉ dẫn STACK như sau:

.STACK Kích thước

Kích thước là độ lớn ngăn xếp tính bằng byte, nếu không ghi thì lấy mặc định là 1024. Ví dụ

```
.STACK   100h   ;khai báo vùng ngăn xếp 256 byte
.STACK          ; khai báo vùng ngăn xếp 1024 byte
```

Viết chương trình hợp ngữ

❖ Cấu trúc chương trình hợp ngữ:

- Thứ ba là lệnh khai báo dữ liệu dung trong chương trình với chỉ dẫn .DATA. Ví dụ:

```
.DATA
    CR          EQU   13
    LF          EQU   10
    Vungnho1    DW    2
    ThongBao    DB    'Chuong trinh ABC'
```

Viết chương trình hợp ngữ

❖ Cấu trúc chương trình hợp ngữ:

- Thứ tư là lệnh khai báo các lệnh của chương trình dùng chỉ dẫn .CODE
- Lệnh cuối cùng là chỉ dẫn END

Một số lệnh đơn giản

MOV Đích, Nguồn

- Lệnh MOV sao chép dữ liệu (định bởi toán hạng nguồn) vào thanh ghi hoặc vùng nhớ (định bởi toán hạng đích)
- Toán hạng nguồn: thanh ghi, vùng nhớ, hằng
- Không được sao chép hằng vào thanh ghi đoạn
- Hai toán hạng không đồng thời là vùng nhớ hoặc thanh ghi đoạn
- Kích thước toán hạng đích phải đủ để nhận dữ liệu

Một số lệnh đơn giản

MOV Đích, Nguồn

- Ví dụ:

```
MOV     AX, VungNho         ;AX ← VungNho
MOV     AX, BX              ;AX ← BX
MOV     AH, 'A'             ;AH ← 41h
MOV     AX, 'A'             ;AX ← 0041h
MOV     DX, OFFSET MSG      ;DX ← địa chỉ ô MSG
MOV     DX, SEG MSG         ;DX ← địa chỉ đoạn MSG
```

Một số lệnh đơn giản

LEA Đích, Nguồn

- Lệnh LEA (Load Effective Address) sao chép địa chỉ offset (định bởi toán hạng nguồn) vào thanh ghi (định bởi toán hạng đích)
- Toán hạng nguồn: tên biến trong đoạn DS
- Toán hạng đích: BX, CX, DX, BP, SI, DI.
- Ví dụ:

LEA DX, MSG; đưa địa chỉ offset của MSG vào DX

Tương đương:

MOV DX, offset MSG

Một số lệnh đơn giản

XCHG Đích, Nguồn

- Lệnh XCHG hoán chuyển dữ liệu của hai toán hạng. Hai toán hạng có thể là thanh ghi hoặc vùng nhớ cùng kích thước nhưng không đồng thời là vùng nhớ
- Ví dụ:

```
XCHG    AX, VungNho
XCHG    AL, BH
```

Một số lệnh đơn giản

ADD Đích, Nguồn

- ADD cộng nội dung toán hạng nguồn với toán hạng đích, kết quả chứa trong toán hạng đích.
- Toán hạng nguồn: hằng, thanh ghi, vùng nhớ
- Toán hạng đích: thanh ghi, vùng nhớ
- Hai toán hạng không đồng thời là vùng nhớ
- Ví dụ:
  - ADD AX, BX ; AX ← AX + BX
  - ADD AX, VungNho ; AX ← AX + VungNho
  - ADD AX, 2 ; AX ← AX + 2

Một số lệnh đơn giản

SUB Đích, Nguồn

- SUB trừ nội dung toán hạng đích với toán hạng nguồn, kết quả chứa trong toán hạng đích.
- toán hạng nguồn: hằng, thanh ghi, vùng nhớ
- toán hạng đích: thanh ghi, vùng nhớ
- Hai toán hạng không đồng thời là vùng nhớ
- Ví dụ:
  - SUB AX, BX ; AX ← AX - BX
  - SUB AX, VungNho ; AX ← AX - VungNho
  - SUB AX, 2 ; AX ← AX - 2

Một số lệnh đơn giản

INC Đích

- Lệnh INC tăng toán hạng (số không dấu) lên 1.
- Toán hạng có thể là thanh ghi hoặc vùng nhớ
- Ví dụ:
  - INC AX ;AX ← AX + 1
  - INC VungNho ;VungNho ← VungNho + 1

Một số lệnh đơn giản

DEC Đích

- Lệnh DEC giảm toán hạng (số không dấu) bớt 1.
- toán hạng có thể là thanh ghi hoặc vùng nhớ
- Ví dụ:
  - DEC AX ;AX ← AX - 1
  - DEC VungNho ;VungNho ← VungNho - 1

Một số lệnh đơn giản

NEG Đích

- Lệnh NEG đổi dấu của toán hạng.
- Toán hạng có thể là thanh ghi hoặc vùng nhớ
- Nếu toán hạng là số âm nhỏ nhất (-128 , -32768)thì trị không thay đổi sau khi thực hiện lệnh
- Ví dụ:
  - NEG AX ;AX ← -AX
  - NEG VungNho ;VungNho ← -VungNho
  - MOV AX, -32768 ; AX = -32768 = 8000h
  - NEG AX ; AX = -32768 = 8000h
  - MOV DL, -128 ; DL = -128 = 80h
  - NEG DL ; DL = -128 = 80h

Bài tập

- Giả sử A được lưu tại địa chỉ offset 0000h. Cho biết C lưu tại địa chỉ offset nào với khai báo sau đây:

```
A       DB    7
B       DW    1ABCh
C       DB    'HAO'
```

- Viết nhóm lệnh tương đương với lệnh gán

$$B = B - A - 1$$

Trong đó A, B là vùng nhớ kiểu từ

Một số lệnh đơn giản

INT 21h

- Lệnh ngắt 21h là lệnh gọi một chương trình con của hệ điều hành. Trình con này có nhiều chức năng khác nhau tùy theo trị của thanh ghi AH.
- Sau đây là một số chức năng (hàm) thông dụng:

Một số lệnh đơn giản

- **Nhập ký tự từ bàn phím**:
  - Nếu AH = 1, ngắt 21h thực hiện nhập một ký tự từ bàn phím
  - Mã ASCII ký tự nhận được chứa trong AL
  - Nếu là ký tự thông thường được gõ thì sẽ hiển thị màn hình.
  - Ví dụ:

```
MOV AH, 1    ; chọn chức năng 1
INT   21h    ; nhập ký tự và chứa vào AL
```

Một số lệnh đơn giản

- **Xuất ký tự ra màn hình**:
  - Nếu AH = 2, ngắt 21h thực hiện xuất ký tự trong DL ra màn hình
  - Ví dụ:

```
MOV AH, 2     ; chọn chức năng 2
MOV DL, 'A'   ; đưa ký tự muốn xuất vào DL
INT   21h     ; hiển thị ký tự
```

Một số lệnh đơn giản

- **Xuất chuỗi ký tự ra màn hình**:
  - Nếu AH = 9, ngắt 21h thực hiện xuất chuỗi ký tự có địa chỉ ô chứa trong thanh ghi DX, địa chỉ đoạn chứa trong DS ra màn hình
  - Chú ý là chuỗi ký tự phải kết thúc bằng dấu '$'

```
.DATA
    MSG   DB     "chuoi can hien thi$"
.CODE
    MOV   AX, @DATA ; lấy địa chỉ đoạn dữ liệu đưa vào DS
    MOV           DS, AX
    MOV           AH, 9   ; chọn chức năng 9
    MOV           DX, OFFSET MSG
    INT           21h     ; hiển thị chuỗi
```

Một số lệnh đơn giản

- **Kết thúc chương trình**:
  - Nếu AH = 4Ch, ngắt 21h thực hiện việc kết thúc chương trình và trả điều khiển về cho hệ điều hành. Lệnh được viết như sau:

```
MOV     AH, 4Ch     ; chọn chức năng 4Ch
INT     21h         ; kết thúc chương trình
```

Nội dung chương

- Cơ bản về hợp ngữ
- Lệnh nhảy
- Lệnh lặp
- Ngăn xếp
- Thủ tục

Lệnh nhảy

- Bình thường: lệnh được thực hiện tuần tự
- Thực tế, tùy tình huống tạo ra những xử lý mà trong đó trật tự các lệnh có thể bị thay đổi
- Lệnh nhảy làm thay đổi trật tự thực hiện các lệnh thông qua các cờ (flag)

Lệnh nhảy

Lệnh **CMP** Đích, Nguồn

- Lệnh CMP (compare): lệnh so sánh
- Lệnh này tương tự lệnh SUB nhưng không làm thay đổi toán hạng đích
- Thường được dùng để tạo cờ cho các lệnh nhảy có điều kiện

Lệnh nhảy

Lệnh **CMP** Đích, Nguồn

- Nếu Đích = Nguồn: ZF = 1
- Nếu Đích # Nguồn: F = 0, và
  - Nếu Đích > Nguồn: SF = 0 (số có dấu), CF = 0 (số không dấu)
  - Nếu Đích < Nguồn: SF = 1 (số có dấu), CF = 1 (số không dấu)

Lệnh nhảy

Lệnh **CMP** Đích, Nguồn

Ví dụ:

Nhóm lệnh	ZF	SF	CF
MOV AL, 0Ah CMP AL, 10	1	0	0
MOV AL, 0Ah CMP AL, 5	0	0	0
MOV AL, 0Ah CMP AL, 15	0	1	1

Lệnh nhảy

Lệnh **JMP** Nhãn

- Lệnh JMP (Jump): lệnh nhảy không điều kiện
- Lệnh này làm thay đổi thứ tự thực hiện lệnh, bất chấp giá trị các cờ
- Tác dụng: ra lệnh cho CPU thực hiện lệnh ngay sau Nhãn

Lệnh nhảy

Lệnh **JMP** Nhãn

- Ví dụ:

```
MOV AL, 1
JMP  Nhan2
Nhan1:
        ADD
        AL, 2
Nhan2:
        ADD
        AL, 3
```

- Sau khi thực hiện lệnh thì AL có giá trị bao nhiêu ?

Lệnh nhảy

- Ví dụ về lệnh JMP

```
JMP     Calc    ; nhảy đến Calc
Back:   JMP     Stop
Calc:   ADD     AX, BX
JMP     Back
Stop:
MOV     AH, 4Ch
INT     21h
```

Lệnh nhảy

- Lệnh nhảy theo điều kiện:
  - Lệnh nhảy theo cờ đơn (test single flag)
  - Nhảy theo kết quả không dấu (compare numbers as unsigned)
  - Nhảy theo kết quả có dấu (compare number as signed)

Lệnh nhảy

CÁC LỆNH NHẢY THEO CỜ ĐƠN

Lệnh	Mô tả	Điều kiện	Lệnh đối
JZ, JE	Jump if Zero (Equal)	ZF = 1	JNZ, JNE
JC, JB, JNAE	Jump if Carry, Below, Not Above Equal	CF = 1	JNC, JNB, JNAE
JS	Jump if Sign	SF = 1	JNS
JO	Jump if Overflow	OF = 1	JNO
JPE, JP	Jump if Parity Even	PF = 1	JPO
JNZ, JNE	Jump if not Zero, Not equal	ZF = 0	JZ, JE
JNC, JNB, JAE	Jump if not Carry, not Below, Above Equal	CF = 0	JC, JB, JNAE
JNS	Jump if not Sign	SF = 0	JS
JNO	Jump if not Overflow	OF = 0	JO
JPO, JNP	Jump if Parity Odd (No Parity)	PF = 0	JPE, JP

Lệnh nhảy

CÁC LỆNH NHẢY THEO SỐ KHÔNG DẤU

Lệnh	Mô tả	Điều kiện	Lệnh đối
JZ, JE	Jump if Zero (Equal)	ZF = 1	JNZ, JNE
JNZ, JNE	Jump if Not Zero, Not equal	ZF = 0	JZ, JE
JA, JNBE	Jump if Above (>0) Jump if Not Below or Equal (not <=)	CF = 0 and ZF = 0	JNA, JBE
JB, JNAE, JC	Jump if Below (<0) Jump if Not Above or Equal (not >=) Jump if Carry	CF = 1	JNB, JAE, JNC
JAE, JNB, JNC	Jump if Above or Equal (>=) Jump if Not Below (not <) Jump if Not Carry	CF = 0	JNAE, JB, JC
JBE, JNA	Jump if Below or Equal (<=) Jump if Not Above (not >)	CF = 1 or ZF = 1	JNBE, JA

Lệnh nhảy

CÁC LỆNH NHẢY THEO SỐ CÓ DẤU

Lệnh	Mô tả	Điều kiện	Lệnh đối
JZ, JE	Jump if Zero (Equal)	ZF = 1	JNZ, JNE
JNZ, JNE	Jump if Not Zero, Not equal	ZF = 0	JZ, JE
JG, JNLE	Jump if Greater (>) Jump if Not Less or Equal (not <=)	ZF = 0 and SF = OF	JNG, JLE
JL, JNGE	Jump if Less (<) Jump if Not Greater or Equal (not >=)	SF <> OF	JNL, JGE
JGE, JNL	Jump if Greater or Equal (>=) Jump if Not Less (not <)	SF = OF	JNGE, JL
JLE, JNG	Jump if Less or Equal (<=) Jump if Not Greater (not >)	ZF = 1 or SF <> OF	JNLE, JG

Lệnh nhảy

Ví dụ 1a: nhảy số không dấu

```
MOV BL, 1
MOV AL, 0h
CMP AL, 0FFh ; so sanh 255
JB  Nhan1  ; nhay vi 0 < 255
INC BL
Nhan1:
INC BL
```

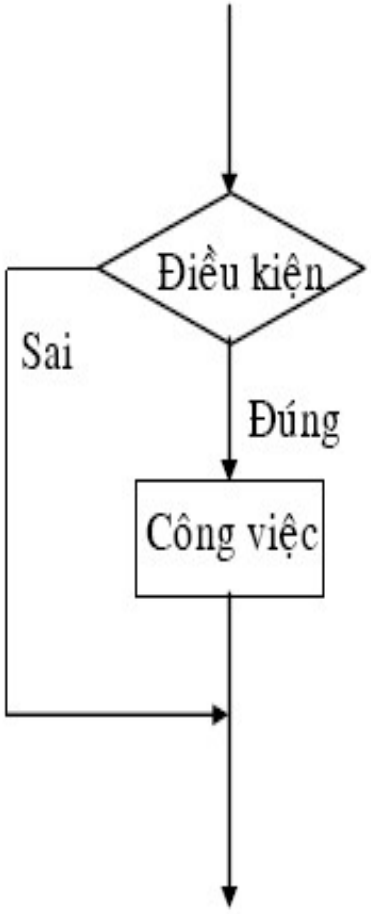
Kết quả đoạn lệnh trên: BL = 02

Ví dụ 1b: nhảy số có dấu

```
MOV BL, 1
MOV AL, 0h
CMP AL, 0FFh ; so sanh -1
JL  Nhan1  ;khong nhay vi 0 > -1
INC BL
Nhan1:
INC BL
```

Kết quả đoạn lệnh trên: BL = 03

Lệnh nhảy



- Cấu trúc IF

Lệnh điều kiện

Nhảy Thoát ; Sai

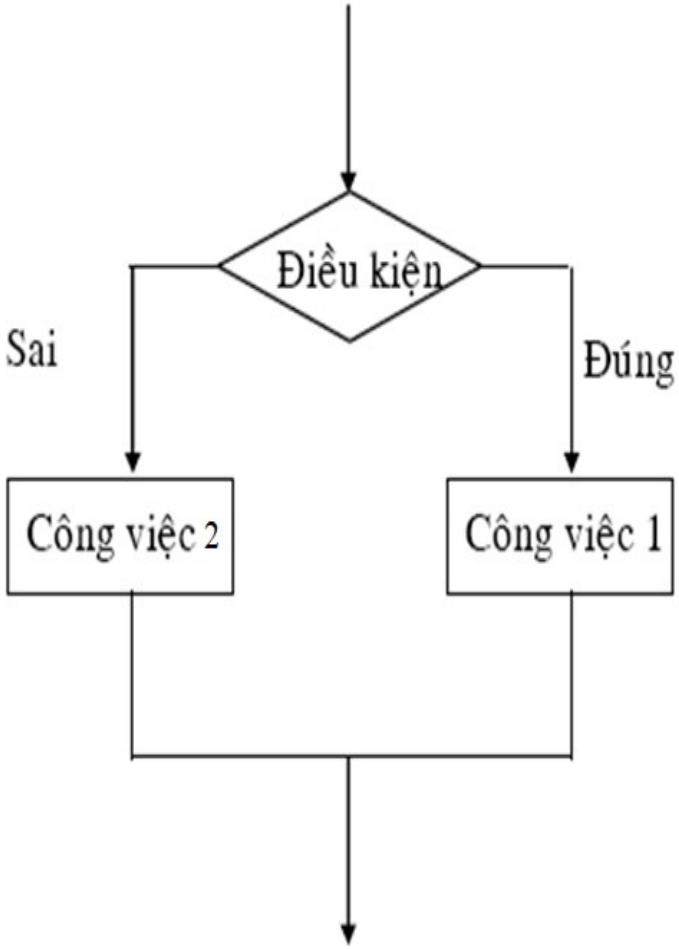
Công việc ; Đúng

Thoát:

Lệnh nhảy

- Ví dụ: Viết chương trình cho phép người dùng nhập vào 1 ký tự. Nếu là ký tự 0 thì báo “Bạn đã nhập số không”, nếu không thì kết thúc chương trình

Lệnh nhảy



- Cấu trúc IF - ELSE

Lệnh điều kiện

Nhảy True:

Công việc 2

JMP Thoát

True:

Công việc 1

Thoát:

Lệnh nhảy

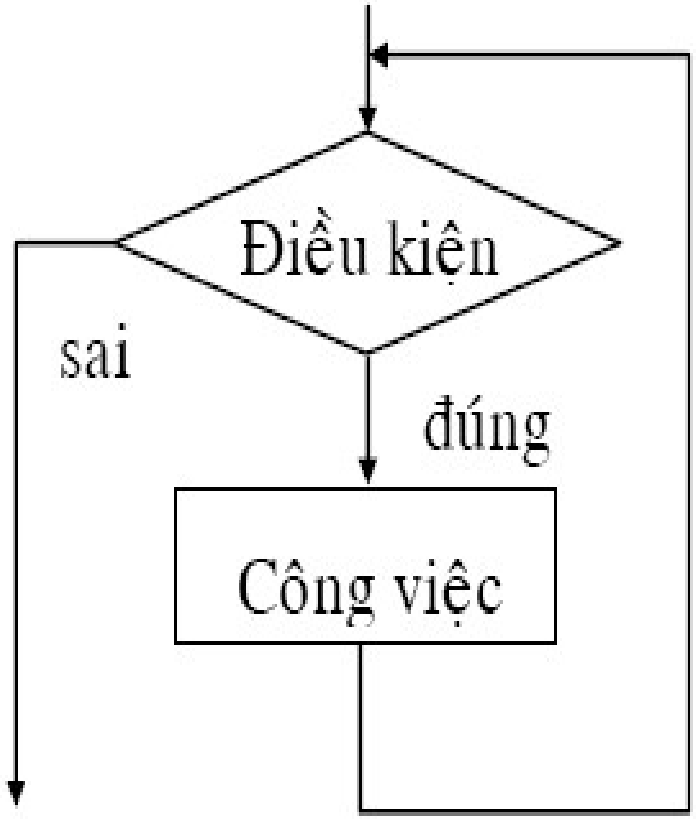
- Ví dụ: Viết chương trình cho phép người dùng nhập vào 1 ký tự. Nếu là ký tự ‘S’ thì báo “Chào buổi sáng”, nếu không thì báo “Chào buổi chiều”

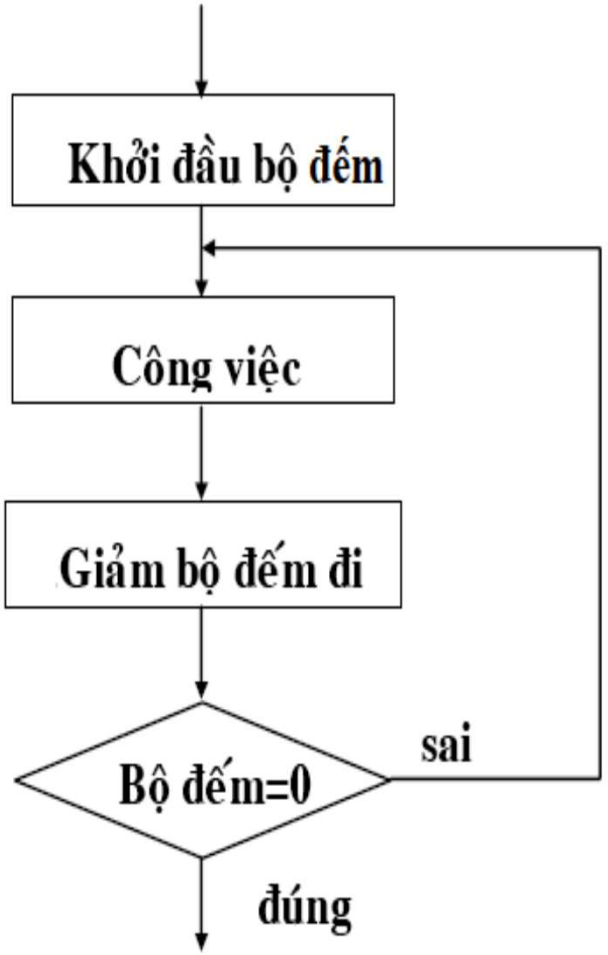
Lệnh lặp

- Vòng lặp là một đoạn lệnh được thực hiện nhiều lần khi còn thỏa một điều kiện nào đó
- Do đó vòng lặp thường được kết thúc bằng một lệnh nhảy có điều kiện
- Ngoài các lệnh nhảy đã biết, ta còn dùng các lệnh nhảy đặc biệt gọi là lệnh lặp: LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ

Lệnh lặp

while	for





Lệnh lặp

- Xét ví dụ: Nhập một chuỗi ký tự, kết thúc bằng enter

```
.CODE
MOV AH, 1    ; chon chuc nang nhap ky tu
KeyLoop:
    INT 21h      ; nhap ky tu va luu trong AL
    CMP AL, 0Dh ; ky tu nhap la enter ?
    JE  Thoat
    JMP KeyLoop  ;
Thoat:
```

Lệnh lặp

Xét ví dụ: Hiển thị 256 ký tự ASCII

```
.CODE
    MOV AH, 2          ; hien thi ky tu
    MOV CX, 256        ; so ky tu hien thi
    MOV DL, 0          ; ky tu dau tien
PrintLoop:             ; Vong lap
    INT 21h            ; hien thi ky tu trong DL
    INC DL             ; lay ky tu ke tiep
    DEC CX             ; giam so lan lap
    JNZ PrintLoop      ; neu CX <> 0 thi lap lai
```

Lệnh lặp

- Thay vì dùng 2 lệnh cuối là DEC và JNZ, ta dùng lệnh LOOP thì sẽ tiết kiệm và nhanh hơn
- LOOP PrintLoop ;Giảm CX, nếu CX <>0 thì lặp lại
- Tổng quát lệnh: LOOP NHÃN
- Ý nghĩa: lệnh này giảm CX đi 1 và kiểm tra CX có bằng 0 hay không. Nếu khác 0 thì nhảy đến NHÃN, nếu bằng 0 thì nhảy đến lệnh kế (nói cách khác, lặp kết thúc khi CX = 0)

Lệnh lặp

- Xét ví dụ: Hiển thị 256 ký tự ASCII dung LOOP

```
.CODE
    MOV AH, 2                   ; hien thi ky tu
    MOV CX, 256         ; so ky tu hien thi
    MOV DL, 0                   ; ky tu dau tien
PrintLoop:                      ; Vong lap
    INT 21h             ; hien thi ky tu trong DL
    INC DL              ; lay ky tu ke tiep
    LOOP PrintLoop      ; neu CX <> 0 thi lap lai
```

Lệnh lặp

- LOOPE/LOOPZ tương tự như LOOP nhưng việc lặp tiếp tục khi CX <>0 và ZF = 1 (hay kết thúc lặp khi CX = 0 hoặc ZF = 0)
- LOOPNE/LOOPNZ kết thúc lặp khi CX = 0 hoặc ZF = 1

5.2Lệnh lặp

Ví dụ:Nhập chuỗi ký tự, kết thúc bằng Enter hoặc nhập đủ 128 ký tự

```
MOV CX, 128 ; chieu dai toi da chuoi nhap
KeyLoop:            ; Vong lap
        MOV AH, 1   ; chon chuc nang nhap ky tu
        INT 21h     ; nhap ky tu va at trong AL
        CMP AL, 0Dh ; ky tu nhap la enter ?
        LOOPNE KeyLoop  ; neu khong thi nhap tiep
hoac CX = 0
```

Một số bài tập

1) Viết CT cho phép người dùng nhập vào một số từ 0 → 9. Nếu người dùng nhập sai thì báo lỗi và yêu cầu nhập lại

2) Viết CT cho phép người dùng nhập vào một số từ 0 → 9. Nếu người dùng nhập sai thì báo lỗi và yêu cầu nhập lại (tối đa 3 lần)

3) Viết CT nhập 1 ký tự là S, C, T.

Nếu nhập S, hiển thị "Chào buổi sáng"

Nếu nhập C, hiển thị "Chào buổi chiều"

Nếu nhập T, hiển thị "Chào buổi tối"

Với yêu cầu là có kiểm tra ký tự nhập, nếu nhập sai thì phải nhập lại

Nội dung chương

- Cơ bản về hợp ngữ
- Lệnh nhảy
- Lệnh lặp
- Ngăn xếp
- Thủ tục

Ngăn xếp (stack)

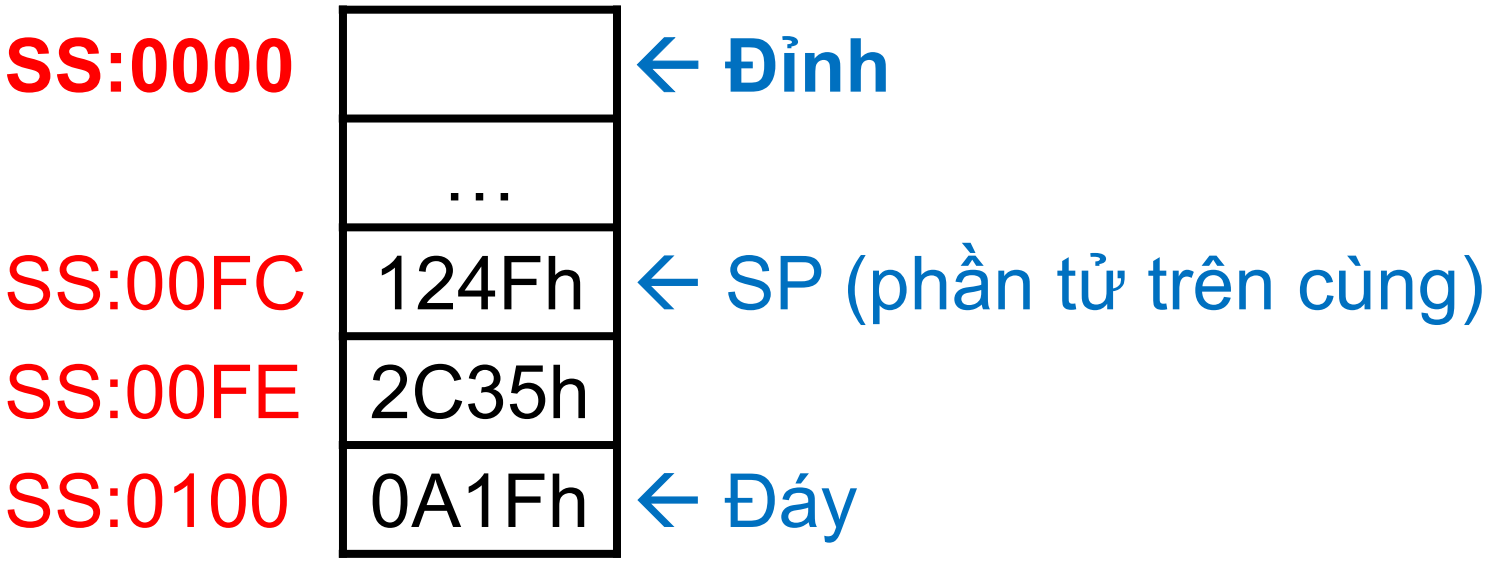
- Là vùng nhớ lưu trữ tạm thời một số dữ liệu dùng cho chương trình hoặc địa chỉ của các trình con (thủ tục)
- Hoạt động theo cơ chế LIFO (Last In First Out)
- Mỗi phần tử của ngăn xếp là một từ

Ngăn xếp

- CPU truy xuất ngăn xếp thông qua cặp thanh ghi SS: SP
- Khi chưa sử dụng, ngăn xếp rỗng, SP trỏ vào đáy ngăn xếp
- Khi đưa dữ liệu vào ngăn xếp, SP giảm bớt 2, lúc đó SP sẽ trỏ vào phần tử trên cùng
- Nếu lấy trị ra khỏi ngăn xếp, SP tăng lên 2

Ngăn xếp

NGĂN XẾP VỚI KHAI BÁO .STACK 100h



Ngăn xếp

- Đưa trị vào/ra
- Muốn đưa trị vào ngăn xếp, ta dùng lệnh

  PUSH Nguồn
- Muốn lấy trị ra khỏi ngăn xếp, ta dùng lệnh

  POP Đích

Với Đích, Nguồn là thanh ghi/vùng nhớ 16 bit

Ngăn xếp

Ví dụ: AX = 1234h, BX = 5678h, CX = 9ABCh

```
PUSH   AX
PUSH   BX
XCHG   AX, CX
POP    CX
PUSH   AX
POP    BX
```

Hãy cho biết nội dung của AX, BX, CX, SP sau khi thực hiện lệnh trên

Ngăn xếp

- Ví dụ dùng ngăn xếp để lưu một chuỗi ký tự, sau đó chuỗi được lấy ra khỏi ngăn xếp và hiển thị theo thứ tự ngược lại

Ngăn xếp

Thuật giải:

- Xóa biến đếm số ký tự nhập (để trị = 0)
- Lặp1:
  - Nhập 1 ký tự
  - Nếu ký tự là enter, kết thúc lặp
  - Không phải là enter, đưa ký tự vào ngăn xếp
  - Tăng biến đếm
  - Về lặp 1

Nhập chuỗi

6.3. Thủ tục

Thuật giải:

- Nếu biến đếm bằng 0 thì kết thúc
- Lặp2:
  - Lấy ký tự trong ngăn xếp và hiển thị
  - Giảm biến đếm
  - Nếu khác không thì về Lặp2

Xuất chuỗi

```
.MODEL     SMALL
.STACK     100h
.DATA
NhacNhap  DB  13,10, 'Nhap chuoi(ket thuc
bang enter): $'
TBKetQua  DB  13, 10, 'Chuoi nguoc: $'
.CODE
;Lấy địa chỉ đoạn dữ liệu vào DS
MOV AX, @DATA
MOV DS, AX
XOR         CX, CX       ; xóa biến đếm
```

```
; Nhap chuoi
; SV tự code đoạn xuất thông báo NhacNhap
; vong lap nhap chuoi va cat vao ngan xep
MOV         AH, 1
LAP1:
        INT   21h
        CMP  AL, 13
        JE              EndLap1
        PUSH         AX
        INC   CX
        JMP  LAP1
EndLap1:
```

```
; Xuat chuoi
; SV tự code đoạn xuất thông báo TBKetQua
; vong lap lay ky tu tu ngan xep va hien thi
JCXZ    EndLap2; neu khong nhap thi ket thuc
MOV     AH, 2
Lap2:
        POP  DX
        INT   21h
        LOOP Lap2
EndLap2:
MOV         AH, 4Ch
INT         21h
END
```

Thủ tục

- Thủ tục (Procedures): là một chương trình con có nhiệm vụ tương đối độc lập. Thực chất, thủ tục cũng chỉ là một đoạn chương trình được viết riêng giúp chương trình dễ đọc, linh hoạt, dễ bảo trì

Thủ tục

- Khai báo thủ tục:
- Thường được viết cuối chương trình, kèm giữa 2 lệnh PROC và ENDP như sau:

```
Tên thủ tục PROC Kiểu
        Các lệnh trong thủ tục
        RET
Tên thủ tục ENDP
```

Thủ tục

- Khai báo thủ tục:
  - Tên thủ tục: Nhãn
  - Kiểu là
    - NEAR (mặc định nếu không khai báo) nếu lệnh gọi thủ tục ở cùng đoạn với thủ tục
    - FAR: nếu lệnh gọi thủ tục ở khác đoạn với thủ tục
- RET là lệnh kết thúc thủ tục và quay trở về trình gọi

Thủ tục

- Lệnh gọi và kết thúc thủ tục:
- Để gọi thủ tục ta dùng một trong hai lệnh sau:

```
CALL    TênThủTục
CALL    ĐịaChỉ      ; thanh ghi/vùng nhớ chứa địa chỉ thủ tục
```

Thủ tục

- Lệnh gọi và kết thúc thủ tục:
  - Khi thực hiện lệnh gọi thủ tục (gần), địa chỉ (ô) của lệnh kế lệnh CALL được cất vào ngăn xếp và địa chỉ (ô) thủ tục được đưa vào IP, do đó lệnh được thi hành sau lệnh CALL chính là lệnh đầu tiên trong thủ tục
  - Khi thủ tục là xa thì trị của CS và IP được cập nhật thay vì chỉ có IP và trị đưa vào ngăn xếp có cả CS (được đưa vào trước)

Thủ tục

- Lệnh gọi và kết thúc thủ tục:
- Ngược lại, khi thực hiện lệnh RET để quay lại trình gọi thì địa chỉ chứa trong ngăn xếp sẽ được lấy ra và đặt vào IP, do đó, lệnh được thi hành kế sau lệnh RET chính là lệnh kế lệnh CALL trong chương trình gọi
- Nếu quay về từ thủ tục xa thì hai trị được lấy ra khỏi ngăn xếp. Trị 1 đưa vào IP, trị 2 đưa vào CS

Thủ tục

- Ví dụ: Cho 2 lệnh

CALL PROC1

MOV AX,BX

Và giả sử MOV nằm ở địa chỉ 08FD:0203, PROC1 là thủ tục NEAR tại địa chỉ 08FD:0300, SP = 010Ah. Hãy cho biết nội dung IP và SP sau khi thực hiện lệnh CALL và sau khi thực hiện RET trong PROC1 ?

Thủ tục

- Ví dụ: Viết lại chương trình dùng ngăn xếp để lưu một chuỗi ký tự, sau đó chuỗi được lấy ra khỏi ngăn xếp và hiển thị theo thứ tự ngược lại **dùng thủ tục**

```
.MODEL    SMALL
.STACK    100h
.DATA
        NhacNhap  DB  13,10, ‘Nhap chuoi(ket
thuc bang enter): $’
        TBKetQua  DB  13, 10, ‘Chuoi nguoc: $’
.CODE
        MOV AX, @DATA
        MOV DS, AX
        CALL Nhap        ; gọi thủ tục nhập chuỗi
        CALL Xuat        ; gọi thủ tục xuất choỗi
        MOV AH, 4Ch
        INT 21h
```

```
; ------------------------------ Nhap chuoi ----------------------------
Nhap PROC
        POP             BX      ;lưu địa chỉ quay về
        MOV     AH,9
        LEA     DX, NhacNhap
        INT     21h
        XOR     CX, CX          ; xoa bien dem
        MOV     AH, 1
LAP1:           ; vong lap nhap chuoi va cat vao ngan xep
        INT     21h
        CMP     AL, 13
        JE      EndLap1
        PUSH    AX
        INC     CX
        JMP     LAP1
EndLap1:
        PUSH    BX              ;trả địa chỉ quay về
        RET
Nhap ENDP
```

```
;  ---------------------------- Xuất chuỗi ----------------------------
Xuat PROC
        POP             BX              ;lưu địa chỉ quay về
        MOV             AH, 9
        LEA             DX,     TBKetQua
        INT     21h
        JCXZ            EndLap2
        MOV             AH,2
Lap2:
        POP     DX
        INT             21h
        LOOP    Lap2
EndLap2:
        PUSH    BX
        RET
        Xuat    ENDP
```