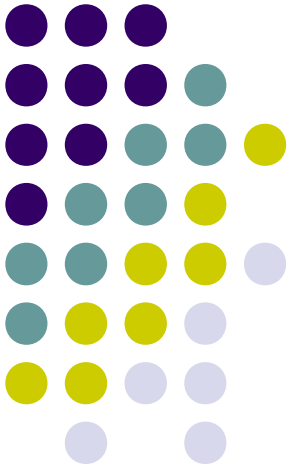
Chương 9
STACK &
CHƯƠNG TRÌNH CON

- Giới thiệu STACK
- Một số ứng dụng của STACK
- Cấu trúc của 1 CTC
- Cơ chế làm việc của 1 CTC
- Vấn đề truyền tham số
- Chương trình gồm nhiều MODULE



GiỚI THIỆU STACK

STACK : là một cấu trúc dữ liệu một chiều. Các phần tử cất vào và lấy ra theo phương thức LIFO (Last In First Out). Mỗi chương trình phải dành ra một khối bộ nhớ để làm stack bằng khai báo STACK. Ví dụ :
.STACK 100H ; Xin cấp phát 256 bytes làm stack

- **Là 1 phần của bộ nhớ, được tổ chức lưu trữ dữ liệu theo cơ chế vào sau ra trước (LIFO).**

LẬP TRÌNH VỚI STACK

- **Trong lập trình có khi cần truy xuất đến các phần tử trong STACK nhưng không được thay đổi trật tự của STACK. Để thực hiện điều này ta dùng thêm thanh ghi con trỏ BP :**
  trỏ BP về đỉnh Stack : MOV BP,SP
  thay đổi giá trị của BP để truy xuất đến các phần tử trong Stack : [BP+2]

- **Phần tử được đưa vào STACK lần đầu tiên gọi là đáy STACK, phần tử cuối cùng được đưa vào STACK được gọi là đỉnh STACK.**

- **Khi thêm một phần tử vào STACK ta thêm từ đỉnh, khi lấy một phần tử ra khỏi STACK ta cũng lấy ra từ đỉnh → địa chỉ của ô nhớ đỉnh STCAK luôn luôn bị thay đổi.**

SS dùng để lưu địa chỉ segemnt của đoạn bộ nhớ dùng làm STACK
SP để lưu địa chỉ của ô nhớ đỉnh STACK (trỏ tới đỉnh STACK)

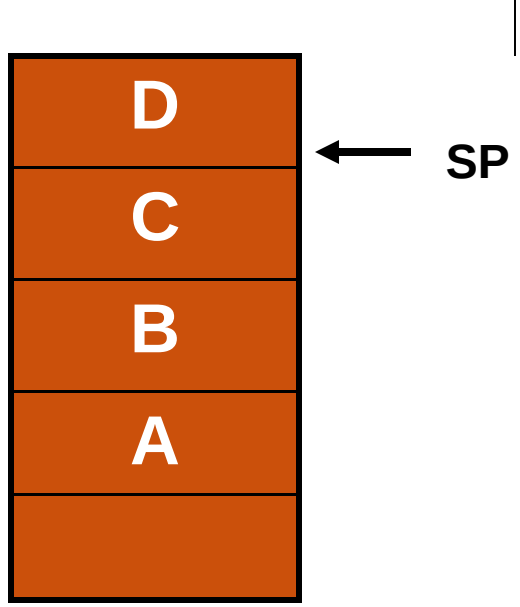
THÍ DỤ

A,B,C là các Word
MOV BP,SP

```
MOV AX,[BP]      ;AX = D

MOV AX,[BP+2]    ;AX = C

MOV AX,[BP+6]    ;AX = A
```



Để lưu 1 phần tử vào Stack ta dùng lệnh PUSH
Để lấy 1 phần tử ra từ Stack ta dùng lệnh POP

PUSH nguồn : đưa nguồn vào đỉnh STACK
PUSHF : cất nội dung thanh ghi cờ vào STACK

- **nguồn là một thanh ghi 16 bit hay một từ nhớ**

POP và POPF : dùng để lấy một phần tử ra khỏi STACK.
Cú pháp : POP đích : đưa nguồn vào đỉnh STACK

POPF : cất nội dung ở đỉnh STACK vào thanh ghi cờ
Chú ý : - Ở đây đích là một thanh ghi 16 bit (trừ thanh ghi IP) hay một từ nhớ
Các lệnh PUSH, PUSHF, POP và POPF không ảnh hưởng tới các cờ

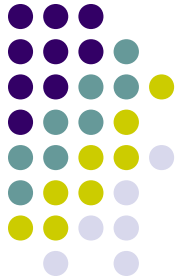
MỘT SỐ ỨNG DỤNG CỦA STACK

- Khắc phục các hạn chế của lệnh MOV
  Ex : MOV CS,DS ; sai
  PUSH DS
  POP CS ; đúng

- **Truyền tham số cho các chương trình con**

- **Lưu tạm thời giá trị thanh ghi hay biến.**

THÍ DỤ 2



- Nhập vào 1 chuổi, in chuổi đảo ngược
  Ex : nhập : Cong nghe thong tin
  xuất : int gnoht ehgn gnoC

Ví dụ minh họa : dùng STACK trong thuật toán đảo ngược thứ tự như sau :

- ; Nhập chuỗi kí tự
- Khởi động bộ đếm
- Đọc một kí tự
- WHILE kí tự <> 13 DO
- Cất kí tự vào STACK
- Tăng biến đếm
- Đọc một kí tự
- END_WHILE
- ; Hiển thị đảo ngược
- FOR biến đếm lần DO
- Lấy một kí tự từ STACK
- Hiển thị nó
- END_FOR

GIỚI THIỆU CHƯƠNG TRÌNH CON

- **CTC là 1 nhóm các lệnh được gộp lại dưới 1 cái tên mà ta có thể gọi từ nhiều nơi khác nhau trong chương trình thay vì phải viết lại các nhóm lệnh này tại nơi cần đến chúng.**

Lợi ích

CTC làm cho cấu trúc logic của của CT dễ kiểm soát hơn, dễ tìm sai sót hơn và có thể tái sử dụng mã ➔ tiết kiệm được công sức và thời gian lập trình.

CẤU TRÚC CỦA CTCON

```
TÊNCTC  PROC  [NEAR|FAR]
    CÁC LỆNH CỦA CTC
   RET
TÊNCTC  ENDP
```

MINH HỌA

- Viết chương trình nhập 1 số n (n nguyên dương và <9). Tính giai thừa của n và xuất ra màn hình dưới dạng số hex (giới hạn kết quả 16 bit).
- Viết chương trình tìm số hoàn thiện (giới hạn 2 chữ số) và in nó ra màn hình.

THÍ DỤ

```
.DATA
EXTRN    MemVar : WORD, Array1 : BYTE , ArrLength :ABS
…
.CODE
EXTRN NearProc : NEAR , FarProc : FAR
….
MOV AX,MemVar
MOV BX, OFFSET Array1
MOV CX, ArrLength
…
CALL NearProc
….
CALL FarProc
…..
```

CƠ CHẾ LÀM VIỆC CỦA CTC

- **Cơ chế gọi và thực hiện CTC trong ASM cũng giống như ngôn ngữ cấp cao.**

➔ Khi gặp lệnh gọi CTC thì :
. Địa chỉ của lệnh ngay sau lệnh gọi CTC sẽ được đưa vào STACK.
. Địa chỉ của CTC được gọi sẽ được nạp vào thanh ghi IP.
. Quyền điều khiển của CT sẽ được chuyển giao cho CTC.
. CTC sẽ thực hiện các lệnh của nó và khi gặp RET, nó sẽ lấy địa chỉ cất trên STACK ra và nạp lại thanh ghi IP để thực thi lệnh kế tiếp.

PUBLIC EXTRN GLOBAL

Để thuận lợi trong việc dịch, liên kết chương trình đa file, Assembler cung cấp các điều khiển Public, Extrn và Global.

PUBLIC

Chỉ cho Assembler biết nhãn (label) nào nằm trong module này được phép sử dụng ở các module khác.

Cú pháp : PUBLIC tên nhãn

khai báo nhãn

TÊN CTC

TÊN BIẾN

TÊN ĐI TRƯỚC NHÃN

EXTRN

Báo cho Assembler biết những nhãn đã được khai báo PUBLIC ở các module khác được sử dụng trong module này mà không cần phải khai báo lại.

Cú pháp : EXTRN Tên nhãn : Kiểu

- BYTE
- DATAPTR
- WORD
- NEAR
- DWORD
- FAR
- PROC

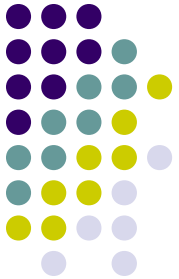
THAY THẾ PUBLIC VÀ EXTRN.

Viết chương trình nằm trên 2 file (2 module) với sự phân công như sau :
Module của chương trình chính (Main.ASM) có nhiệm vụ xác định Offset của 2 chuổi ký tự và gọi CTC nối 2 chuổi này và cho hiện kết quả ra màn hình.
Module CTC (Sub.ASM) làm nhiệm vụ nối 2 chuổi và đưa vào bộ nhớ.

Ví dụ minh hoạ về STACK, CALL/RET : chương trình in một số nguyên (16 bit) ra màn hình

```
PrintNum10    PROC
; số nguyên N nằm trong AX
              PUSH   BX     CX    DX
              MOV    CX, 0          ; so lan push (so ky tu)
laysodu:
          XOR     DX, DX       ;  cho DX = 0 trước khi chia
                  MOV    BX, 10
                  DIV     BX             ; số dư trong DX, phần nguyên
trong AX
                  PUSH  DX              ; lưu  phần dư vào stack
                  INC     CX
                 CMP    AX, 0          ; đã hết chưa?
                 JNZ     laysodu         ; chưa hết, lấy số dư tiếp
                  MOV    AH, 2
        INSO:
              POP     DX
          ADD    DL, '0'
          INT      21H
        LOOP   inso
           POP     DX      CX     BX
                   RET
           ENDP        PrintNum10
```

CHƯƠNG TRÌNH ĐA FILE



- Cho phép nhiều user cùng tham gia giải quyết 1 chương trình lớn.
- Sửa module nào thì chỉ cần dịch lại module đó.
- Mỗi module chỉ giải quyết 1 vấn đề ➔ dễ tìm sai sót.

VẤN ĐỀ TRUYỀN THAM SỐ

- **CHUYỂN GIÁ TRỊ CỦA THAM SỐ TỪ CT GỌI ➔ CT ĐƯỢC GỌI**

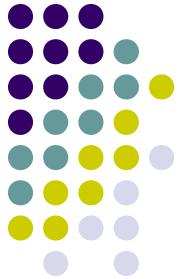
Có 3 cách truyền tham số

Thông qua thanh ghi

Thông qua biến toàn cục

Thông qua STACK

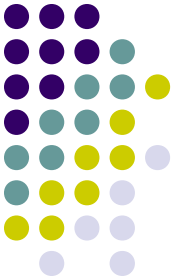
TRUYỀN THAM SỐ THÔNG QUA THANH GHI



- **DỄ**
- **ĐƠN GiẢN**
- **THƯỜNG ĐƯỢC SỬ DỤNG ĐỐI VỚI NHỮNG CT THUẦN TÚY ASM**

ĐẶT 1 GIÁ TRỊ NÀO ĐÓ VÀO THANH GHI Ở CTCHÍNH VÀ SAU ĐÓ CTC SẼ SỬ DỤNG GIÁ TRỊ NÀY TRONG THANH GHI.

TRUYỀN THAM SỐ THÔNG QUA BiẾN GLOBAL



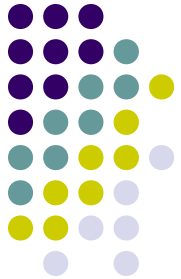
- KHAI BÁO BiẾN TOÀN CỤC.
- DÙNG NÓ ĐỂ CHUYỂN CÁC GIÁ TRỊ GiỮA CT GỌI VÀ CT ĐƯỢC GỌI.

CÁCH NÀY THƯỜNG ĐƯỢC DÙNG :

TRONG 1 CT ViẾT THUẦN TÚY BẰNG ASM

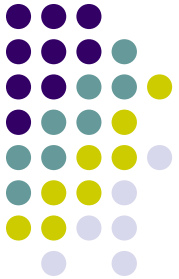
ViẾT HỖN HỢP GiỮA ASM VÀ 1 NGÔN NGỮ CẤP CAO

TRUYỀN THAM SỐ QUA STACK



- **PHỨC TẠP HƠN.**
- **DÙNG RẤT NHIỀU KHI ViẾT CHƯƠNG TRÌNH HỖN HỢP GiỮA ASM VÀ NGÔN NGỮ CẤP CAO.**

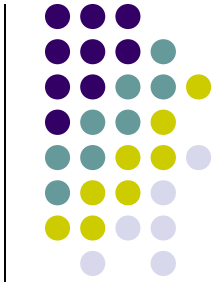
CHUYỂN GIÁ TRỊ TỪ CTCON LÊN CT CHÍNH.



- CŨNG THÔNG QUA CÁC THANH GHI,BỘ NHỚ VÀ STACK.

NẾU GIÁ TRỊ TRẢ VỀ LÀ 8 BIT HOẶC 16 BIT (CHO KHAI BÁO CHAR, INT, CON TRỎ GẦN) THÌ GIÁ TRỊ ĐÓ PHẢI ĐƯỢC ĐẶT TRONG THANH GHI AX CỦA HÀM TRƯỚC KHI QUAY VỀ CTCHÍNH.

CHUYỂN GIÁ TRỊ TỪ CTCON LÊN CT CHÍNH.

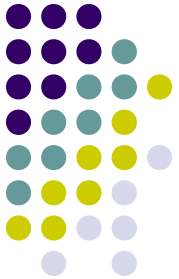


- **NẾU GIÁ TRỊ QUAY LẠI LÀ 32 BIT (CHO KHAI BÁO LONG, CON TRỎ XA) THÌ GIÁ TRỊ ĐÓ PHẢI ĐƯỢC ĐẶT TRONG THANH GHI DX,AX CỦA HÀM TRƯỚC KHI QUAY VỀ CT CHÍNH.**

NEAR | FAR báo cho lệnh RET lấy địa chỉ quay về chương trình gọi nó trong STACK.

- **NEAR : lấy địa chỉ OFFSET (16BIT) trong STACK và gán vào thanh ghi IP.**
- **FAR : lấy địa chỉ OFFSET và SEGMENT trong STACK nạp vào thanh ghi CS:IP.**

VẤN ĐỀ BẢO VỆ CÁC THANH GHI



- **CẦN ĐƯỢC QUAN TÂM TRONG QUÁ TRÌNH LẬP TRÌNH ASM.**
- **RẤT DỄ XẢY RA CÁC TRƯỜNG HỢP LÀM MẤT GIÁ TRỊ CỦA MÀ CT CHÍNH ĐÃ ĐẶT VÀO THANH GHI ĐỂ SỬ DỤNG SAU NAY KHI TA GỌI CTCON.**

CÁC VÍ DỤ MINH HỌA

NHẬP VÀO 1 SỐ HỆ HEX. IN RA SỐ ĐÃ NHẬP VỚI YÊU CẦU SAU :

ViẾT CTCON NHẬP SỐ

ViẾT CTCON XUẤT SỐ

CTCHÍNH GỌI 2 CTCON TRÊN.

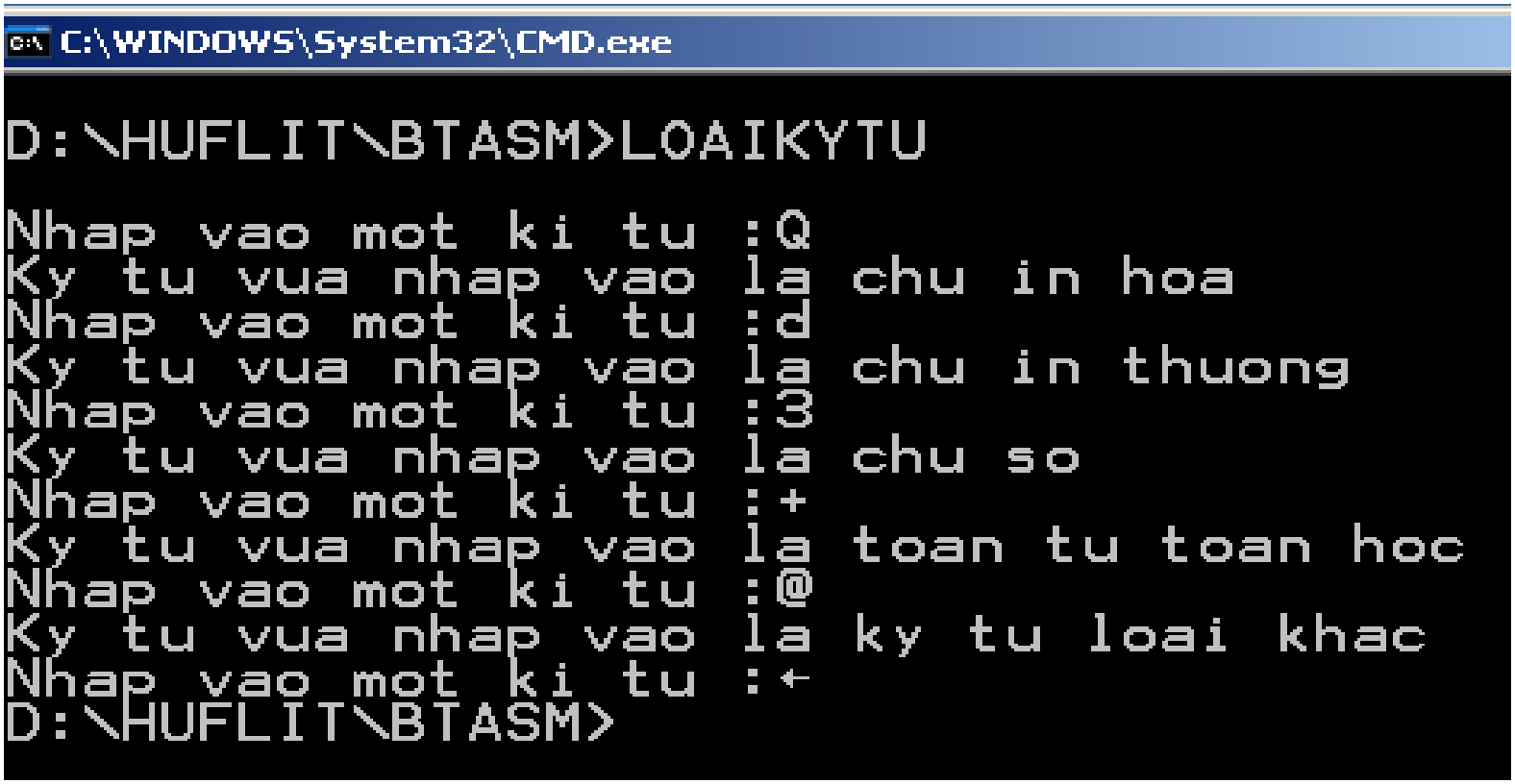
LUYỆN TẬP LẬP TRÌNH C10

Bài 1 : Viết chương trình nhập 1 số nguyên n (n<9). Tính giai thừa của n và xuất kết quả ra màn hình dưới dạng số Hex (giới hạn 16 bits).

Bài 2 :Viết chương trình nhap vao 1 chuổi ky tu. Hay in ra man hinh chuổi ky tu vua nhap theo thứ tự đảo (trong mỗi từ đảo từng ký tự).

Bài 3 :Viết chương trình kiểm tra một biểu thức đại số có chứa các dấu ngoặc (như (), [] và {}) là hợp lệ hay không hợp lệ .
Ví dụ : (a + [b – { c * (d – e) }] + f) là hợp lệ nhưng (a + [b – { c * (d – e)] } + f) không hợp lệ.
HD : dùng ngăn xếp để PUSH các dấu ngoặc trái (‘(‘, ’{‘, ‘[‘) vào Stack

Bài 4 : Viết chương trình nhập vào 1 ký tự, cho biết ký tự vừa nhập thuộc loại gi ? – ký tự, ký số ,toán tử toán học hay ký tự khác. Nếu ký tự là phím Escape thì thoát chương trình.



Bài 6 :Viết chương trình nhập 1 chuỗi ký tự. Xuất ký tự dưới dạng viết hoa ký tự đầu của từng từ, các ký tự còn lại là chữ thường

Ex :
Nhập : ngo phuoc nguyen
Xuất : Ngo Phuoc Nguyen
Nhập : VU tHanh hIEn
Xuất : Vu Thanh Hien

Bài 7 : Viết chương trình tìm số hoàn thiện (giới hạn 2 chữ số). Xuất các số hoàn thiện từ số lớn nhất đến số nhỏ.