IT4778
LẬP TRÌNH HỆ THỐNG

TS. Đỗ Quốc Huy
huydq@soict.hut.edu.vn

Lập trình hợp ngữ

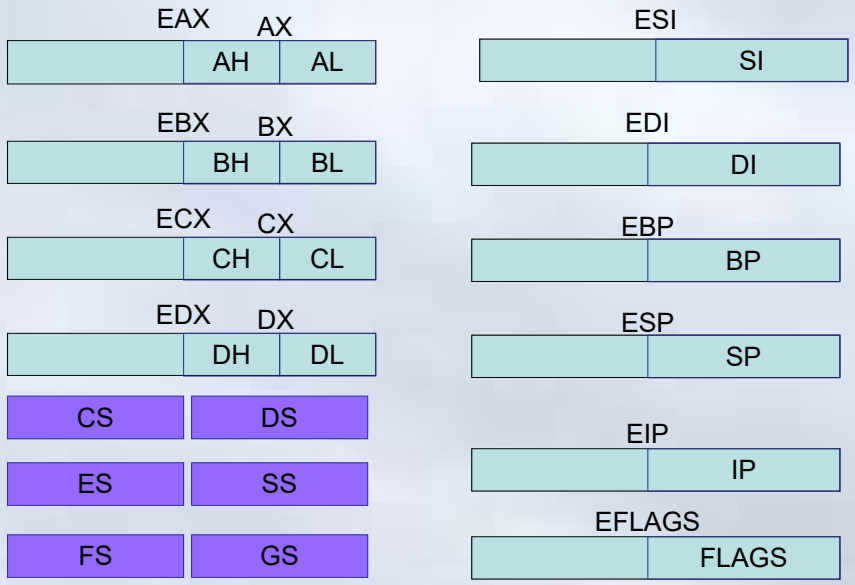
LẬP TRÌNH HỢP NGỮ
Hợp ngữ >< Ngôn ngữ bậc cao

Loại ứng dụng	Các ngôn ngữ bậc cao	Hợp ngữ
Các phần mềm doanh nghiệp, đơn nền tảng, kích cỡ tầm trung tới lớn.	Các cấu trúc có quy chuẩn, Dễ đọc, bảo trì các đoạn mã lệnh lớn	Các cấu trúc ít quy chuẩn, Khó bảo trì
Trình điều khiển thiết bị phần cứng (device driver)	Có thể không cho truy nhập trực tiếp đến phần cứng. Khó bảo trì	Truy nhập phần cứng trực tiếp và đơn giản. Dễ bảo trì khi chương trình ngắn và lập tài liệu cẩn thận.
Các ứng dụng doanh nghiệp đa nền	Thường khả chuyển, mã nguồn có thể được dịch trên các môi trường đích không thay đổi nhiều.	Thường phải viết lại cho từng nền tảng, khó bảo trì
Hệ thống nhúng và ứng dụng game yêu cầu truy xuất phần cứng trực tiếp	Tạo ra quá nhiều mã thực thi và có thể không hiệu quả	Mã thực thi nhỏ và chạy nhanh

LẬP TRÌNH HỢP NGỮ
Nội dung chính

- Thành phần cơ bản của hợp ngữ
- Lập trình hợp ngữ căn bản
- Cấu trúc tập lệnh của x86
- Thủ tục và Macro
- Dữ liệu có cấu trúc
- Hợp ngữ và ngôn ngữ bậc cao

Các thanh ghi của 80x86



Các thanh ghi của 80x86

1. Các thanh ghi 8 bit thông dụng:
   AL, BL, CL, DL, AH, BH, CH, DH
2. Các thanh ghi 16 bit thông dụng:
   AX, BX, CX, DX, SP, BP, SI, DI
3. Các thanh ghi 32 bit thông dụng:
   EAX, EBX, ECX, EDX, ESP, EBP, ESI,EDI
   (*Accumulator, Base, Counter, Data, Stack pointer, Base pointer, Source index, Destination Index*)
4. Các thanh ghi đoạn: 16 bit
   CS ,SS, DS, (Code, Stack, Data Segment)
   ES, FS, GS (Additional segments)
5. Thanh ghi con trỏ lệnh 16/32 bit: IP/EIP

Hằng số nguyên

- Số nhị phân: 10110**b**
- Số bát phân: 123**q**
- Số thập phân: 1234**d**, 1234
- Số thập lục phân: 12AB**h**
  - Phải bắt đầu bởi một chữ số: 05Ah

Toán tử dấu (**+, -**)

- Tùy chọn

Hằng ký tự và hằng chuỗi

- Hằng ký tự
  - Được đặt trong cặp nháy đơn hoặc kép: 'A'; «A»
  - Mỗi ký tự ASCII 1 byte
  - Chuyển thành mã ASCII bởi trình dịch hợp ngữ
    - Khai báo 'A' và 41h là tương đương
- Hằng chuỗi
  - Chuỗi ký tự đặt trong cặp nháy đơn hoặc kép
    - « Hello » ; 'Hello'
  - Mỗi ký tự chiếm 1 byte
  - Chấp nhận chuỗi nhúng: '*Say «Hello» to him* '

Từ dành riêng và tên tự định nghĩa

- Từ dành riêng
  - Được sử dụng với các mục đích xác định
  - Gồm lệnh, chỉ thị, toán tử, ký hiệu..
  - Khác nhau giữa các bộ dịch
- Tên tự định nghĩa (identifier)
  - Dãy bao gồm: chữ cái, chứ số, _, $, ?, @, .
  - Bắt đầu bởi: chữ cái, _, ?, $, @
    - Bắt đầu bởi chấm (.): Thường mang ý nghĩa đặt biệt
  - Không phân biệt chữ hoa, chữ thường

Chỉ thị (directives)

- Các lệnh được nhận dạng và được xử lý bởi trình dịch hợp ngữ
  - Không phải là các lệnh của bộ VXL (Intel)
  - Được sử dụng để khai báo vùng dữ liệu, mã lệnh, kiểu bộ nhớ, khai báo thủ tục,..
  - Không phân biệt chữ hoa, chữ thường
- Khác nhau giữa các bộ dịch hợp ngữ
  - TASM, NASM, MASM,..

Câu lệnh

- Được dịch ra ngôn ngữ máy bởi trình dịch hợp ngữ và được thực hiện bởi CPU
  - Sử dụng kiến trúc tập lệnh 32bit (IA-32)
- Mỗi dòng lệnh có cú pháp

  label**:** Instruction operands **;**comment
  - **Label:** → Nhãn, Tùy chọn
  - **[Prefix] Instruction** → Chỉ thị; Bắt buộc
  - **Operands** → Toán hạng; phụ thuộc vào chỉ thị
  - **; Comment** → Chú thích dòng lệnh; Tùy chọn

Nhãn

- Tuân theo quy tắc của tên tự định nghĩa
- Đóng vai trò đánh dấu vị trí
  - Đánh dấu địa chỉ (*Offset*) của phần dữ liệu/ lệnh
- Nhãn dữ liệu
  - Phải duy nhất
  - Không có dấu : theo sau
  - Ví dụ: **Count** DW 100
- Nhãn lệnh
  - Đích của câu lệnh nhảy, lệnh lặp
  - Có dấu : theo sau
  - Ví dụ: **Done:**

Câu lệnh ([Prefix] Instruction)

- Phần [Prefix] sử dụng trong các hàm chuỗi
- Được viết dưới dạng dễ nhớ
- Có thể được phân thành nhiều lớp
  - Lệnh toán học logic: ADD, IDIV, AND, OR,..
  - Lệnh gán dữ liệu (data transfer): MOV, PUSH,..
  - Lệnh điều khiển: CMP, JMP, JZ, JNZ, JC, JNC,..
  - Lệnh liên quan tới bit: CLD, STD, SAR, SAL,..
- Có nhiều dạng lệnh
  - Lệnh không cần toán hạng: NOP, STD,..
  - Lệnh cần 1 toán hạng: INC EAX, Jmp Dest
  - Lệnh cần 2 toán hạng: MOV AH,100; ADD AX,10

Toán hạng

- Là các tham số của câu lệnh
- Có thể là
  - Các thanh ghi: MOV ES, AX
  - Hằng số: MOV AX, 100
  - Biểu thức hằng: MOV AX, 100+200*12
  - Tham chiếu tới vùng nhớ
    - Bộ nhớ (Nhãn dữ liệu): MOV AX, Counter
    - Con trỏ MOV AX, [BX]
- Kích thước vùng nhớ tham chiếu:
  - BYTE PTR, WORD PTR, DWORD PTR,
  - Ví dụ: MOV EAX, DWORD PTR [BX]

Chú thích

- Chấp nhận chú thích dòng lệnh
- Bắt đầu bởi dấu chấm phẩy (;)
- Chú thích trên nhiều dòng
  - Cho phép trong một số bộ dich hợp ngữ
  - Bắt đâu bởi chỉ thị COMMENT và ký tự tùy chọn
  - Kết thúc bởi ký tự đã chọn trước đó

LẬP TRÌNH HỢP NGỮ

Nội dung chính

- Thành phần cơ bản của hợp ngữ
- Lập trình hợp ngữ căn bản (TASM)
- Cấu trúc tập lệnh của x86
- Thủ tục và Macro
- Dữ liệu có cấu trúc
- Hợp ngữ và ngôn ngữ bậc cao

Định nghĩa dữ liệu

- Mục đích
  - Dành vùng nhớ để lưu giá trị các biến
- Cú pháp

  [*name*] *directive initializer* [,*initializer*] . . .

  - Ví dụ

    Count DB 100

    DW100

Định nghĩa dữ liệu

- Các chỉ thị sử dụng
  - **DB** BYTE
  - **DW** WORD
  - **DD** DWORD
  - **DQ** QUADWORD
  - **DT** Ten BYTE
- Các giá trị khởi tạo (initializers)
  - Trở thành giá trị nhị phân khi lưu trong bộ nhớ
  - Giá trị khởi tạo là **?** → Không quan tâm

Khai báo các biến đơn

- Biến kiểu **char/byte**
  - Cnt DB -40 ;Cnt là vị trí (địa chỉ) của 1 byte
  - Ch DB 'A' ;Ch là vị trí của 1 byte có giá trị 41h
- Biến kiểu **int**
  - Cnt DW 1234h; 2byte chứ giá trị 1234h
    ;Lưu trữ ngược: Cnt:34h, Cnt+1:12h
  - Max DW 'AB'; 2byte chứa giá trị 42h,41h??
- Biến kiểu **long**
  - Cnt DD ? ; 4bytes, có giá trị ngẫu nhiên

Khai báo các biến mảng

- Mảng là một dãy byte nhớ liên tiếp nhau
- Biến mảng là dãy byte nhớ được xác định địa chỉ bởi một tên
- Ví dụ: **Arr DB 10h, 20h ,30h, 40h**
  - Mảng 4 phần tử kiểu byte,
  - Các phần tử được khởi tạo 10h, 20h, 30h,40h
  - Arr: là phần tử đầu (Byte đầu)
  - Arr+1 phần tử thứ 2 (20h), Arr+2: Phần tử thứ 3,..

MOV AX, WORD PTR Arr; AX: 2010h

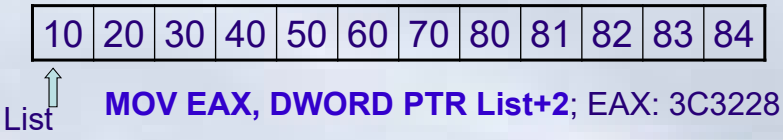
Khai báo các biến mảng → Ví dụ

- Khai báo với các biểu diễn dữ liệu khác nhau

```
Buf DB 'A', 65, 41h, 01000001b,101q
        ;5 byte liên tiếp đều chứa giá trị 65
```

- Khai báo trên nhiều dòng

```
List DB 10, 20, 30, 40   ;12 bytes liên tiếp
     DB 50, 60, 70, 80   ;sử dụng như mảng các
     DB 81, 82, 83, 84   ;bytes
```

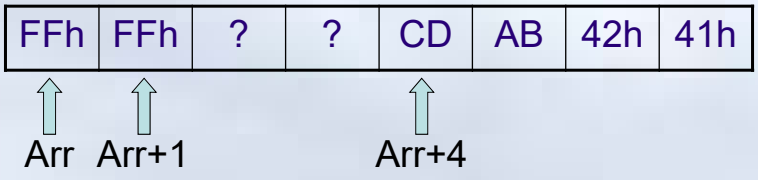


MOV EAX, DWORD PTR List+2; EAX: 3C32281Eh

Khai báo các biến mảng

Arr DW -1,?, 0ABCDh, 'AB'

- Mảng 4 phần tử kiểu Word,



- Mov AX, Arr ;AX:FFFF
- Mov AX, Arr+4 ;AX: ABCDh
- Mov AX, Arr+5 ;AX; 42ABh
- Mov AH, Arr+5 Error: Operand type not match
- Mov AH, BYTE PTR Arr+5 ;AH: ABh

Khai báo các biến chuỗi

- Chuỗi là một mảng của các ký tự
- Theo quy ước
  - Các ký tự của chuỗi đặt trong cặp nháy đơn/kép
  - Thường kết thúc bởi null (*0*) hoặc ký tự '$' (24h)
- Ví dụ
  - St1 DB "Hello$" ;Các xâu St1 St2, St3 là
  - St2 DB 'H','e','l','l ','o','$' ; tương đương
  - St3 DB 48h, 65h,6Ch,6Ch,6Fh,24h ;
  - St4 DB "This is a very long string in the "
    DB "assembly language $",

Khai báo các biến chuỗi

- Cho phép dùng ký tự điều khiển trông xâu
  - 0Dh = carriage return
  - 0Ah = line feed
- Ví dụ

```
menu DB "Checking Account",0dh,0ah,0dh,0ah
     DB "1. Create a new account",0dh,0ah
     DB "2. Open an existing account",0dh,0ah
     DB "3. Credit the account",0dh,0ah
     DB "4. Debit the account",0dh,0ah
     DB "5. Exit",0dh,0ah
     DB "Choice > $"
```

Toán tử DUP

- Sử dụng để tạo không gian cho mảng, chuỗi
- Cú pháp: *Counter* **DUP** (*Argument*)
  - **Counter** và ***Argument*** là hằng, biểu thức hằng
- Ví dụ
  - Arr1 DB 20 DUP(0) ; 20 bytes, khởi tạo zero
  - Arr2 DB 20 DUP(?) ; 20 bytes, không khởi tạo
  - Arr3 DB 2 DUP("ABCDE") ; 10 bytes: "ABCDEABCDE"
  - Arr4 DB 10,3 DUP(0),20 ; 5 bytes: 10, 0, 0, 0, 20
  - Arr5 DW 4 DUP(10) ; 8bytes: 0A,00,0A,00,0A,00,0A,00
  - Arr6 DD 2 DUP('ABCD'); 8bytes: 68,67,66,65,68,67,66,65

Các hằng số

- Sử dụng chỉ dẫn **=** (dấu bằng)
- Cú pháp: **name = expression**
  - Expresion trả về số nguyên 32 bit
  - Giá trị hằng có thể được định nghĩa lại
- Ví dụ

```
Max= 70+50
…….
Mov Ah, Max
Max = 50
```

Tính toán kích thước của mảng

- Sử dụng toán tử trả về vị trí hiện thời:
  - Ký tự **$**
  - Trừ đi địa chỉ của mảng/chuỗi ⇒ số bytes
  - Chia cho kích thước một phần tử ⇒ số p/tử
- Ví dụ

```
Arr DB 10,20,30,40      ; Mảng các BYTE
ArrSize = ($ - Arr)
Arr DW 10, 20, 30, 40   ; Mảng các WORD
ArrSize = ($ - Arr)/2
Arr DD 10,20,30,40      ; Mảng các DWORD
ArrSize = ($ - Arr) /4
```


Chỉ thị **EQU**

- Dùng định nghĩa một tên hằng
  - Có thể là hằng số, hằng chuỗi
  - Không cho phep định nghĩa lại
- Cú pháp Name **EQU** Constant
- Ví dụ
  - PI EQU 3.1416 ;Đinh nghĩa hằng số
  - DOSInt **EQU** 21h
  - Str **EQU** 'Press eny key to continue..$'; hằng xâu
  - Msg **DB** Str ;Định nghĩa biến xâu
  - Mov BX, **Offset** Msg ;Sử dụng
  - Int DOSInt ;Sử dụng

Một số lệnh đơn giản

- Lệnh **MOV**
  - Chuyển DL giữa các thanh ghi và vị trí nhớ
  - Cú pháp: **MOV** Dest, Source
  - Ví dụ: MOV AX, BX; Mov DS, [Count]
- Lệnh **XCHG**
  - Hoán đổi nội dung 2 thanh ghi hoặc một thanh ghi và một biến nhớ
  - Ví dụ: XCHG AX, BX

Một số lệnh đơn giản→Giới hạn của lệnh

- Các lệnh có thể đòi hỏi tham số
  - Tham số là các thanh ghi, biến nhớ, hằng
- Các lệnh bị giới hạn khả năng kết hợp
- Ví dụ lệnh MOV

Destination	Source			
	Gen. Reg.	Seg. Reg.	Mem. Loc.	Const
Gen. Reg.	Y	Y	Y	Y
Seg. Reg.	Y	N	Y	N
Mem. Loc.	Y	Y	N	Y
Const	N	N	N	N

Một số lệnh đơn giản

Lệnh **ADD** & **SUB**

- Cộng trừ nội dung 2 thanh ghi, thanh ghi và biến nhớ, thanh ghi với hằng số
- Cú pháp: **ADD** Dest, Source
  SUB Dest, Source
- Ví dụ:

```
ADD Count, BX
ADD AX, 10
SUB AX, Count
SUB AX, BX
```


Một số lệnh đơn giản

- Lệnh **INC** & **DEC**
  - Tăng/ giảm đi 1 đơn vị của thanh ghi biếnnhớ
  - Cú pháp: **INC** Dest
    DEC Dest
- Lệnh **NEG**
  - Lấy mã bù 2 của thanh ghi, biến nhớ
  - Cú pháp: **NEG** Destination
- Lệnh **INT**
  - Gọi dịch vụ của BIOS / Hệ điều hành (DOS)
  - Cú pháp: **INT** IntNo

Cấu trúc chương trình hợp ngữ

- Bao gồm các phần: Mã lệnh (code), Dữ liệu (Data), Ngăn xếp (Stack)
  - Mỗi phần là một đoạn chương trình và được trình dịch hợp ngữ chuyển thành đoạn bộ nhớ
  - Các phần có thể chung nhau đoạn bộ nhớ
- Có các loại chương trình
  - *.COM: Chỉ một đoạn bộ nhớ duy nhất
  - *.EXE: Cho phép nhiều đoạn

  Phụ thuộc mô hình bộ nhớ

Mô hình bộ nhớ

- Cú pháp: **.Model** Model_Type

Model	Mô tả
Tiny:	Code, Data,Stack trong cùng mọt đoạn bộ nhớ. Thường dùng cho file *. com
Small	Code, Data trong cùng đoạn
Medium	Code nhiều đoạn, Data một đoạn
Compact	Code một đoạn, Data nhiều đoạn
Large	Nhiều đoạn Code, Data, mảng nhỏ hơn 64k
Huge	Nhiều đoạn Code, Data, mảng lớn hơn 64k

Đoạn mã lệnh

- Chứa các câu lệnh của chương trình
- Bắt đầu bằng chỉ dẫn **.Code**
  - Các lệnh được tổ chức thành thủ tục

```
SubProc PROC
;Thân chương trình con SubProc
SubProc EndP
```

  - Có thể dùng lệnh **Jmp** nhảy qua phần DL cục bộ
  - Thường bắt đầu bởi thiết lập thanh ghi DS để trỏ tới đoạn dữ liệu

Đoạn dữ liệu

- Chứa các khai báo biến, khai báo hằng
- Bắt đầu bằng chỉ dẫn **.Data**
- Ví dụ

```
.Data
    Max  EQU 100
    N    DW  100
    Msg  DB 'Hello world!$'
```

Đoạn ngăn xếp

- Dùng lưu trữ thông tin dạng Stack
- Được khai báo **.Stack size**
  - Size: Kích thước của ngăn xếp
- Ví dụ

  .Stack 100h; Đặt ngăn xếp kích thước 256bytes

File *. EXE template, Version 1

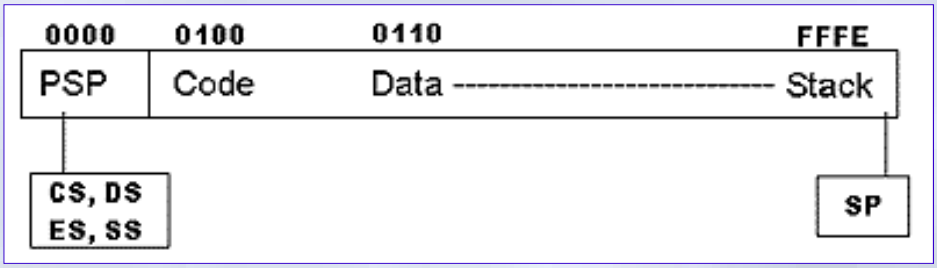
```
.MODEL Large                ;Chương trình lớn, gồm nhiều đoạn
.386                        ; Intel 386
.STACK 100h                 ; Đoạn ngăn xếp 256byte
.DATA                ; Đoạn dữ liệu
   ;Khai báo các biến, hằng
.CODE                       ; Đoạn lệnh
mainProc PROC
   mov ax,@data     ; initialize DS
   mov ds,ax
                ;Các câu lệnh của chương trình con main
   .exit        ;Câu lệnh kết thúc chương trình (AH:4Ch, Int21h)
mainProc ENDP
        ;Các thủ tục khác nếu cần thiết
END mainProc ;mainProc là tên của chương trình  chính
```

File *. EXE template, Version 2

```
.386
ASSUME CS: CSEG, DS: DSEG
DSeg SEGMENT ; Tạo đoạn Dseg là đoạn dữ liệu
   ; Khai báo biên
DSeg ENDS
CSeg SEGMENT     ; Tạo đoạn CSeg là đoạn mã
begin:
        MOV AX, DSeg ; Khởi động địa chỉ đoạn dữ liệu
        MOV DS, AX
        ;Các câu lệnh khác
        MOV AH, 4Ch ; Thoát chương trình
        INT 21h
CSeg ENDS
   END begin         ;Điểm bắt đầu của chương trình
```


File * .COM template

- Là ảnh nhị phân của chương trình mã máy
- Được nạp vào đoạn có địa chỉ thấp
  - Đặt khối PSP chiêm 256bytes đầu
  - Tiếp đến Mã lệnh, Dữ liệu, Stack
  - Các thanh ghi đoạn bằng nhau; IP:100h



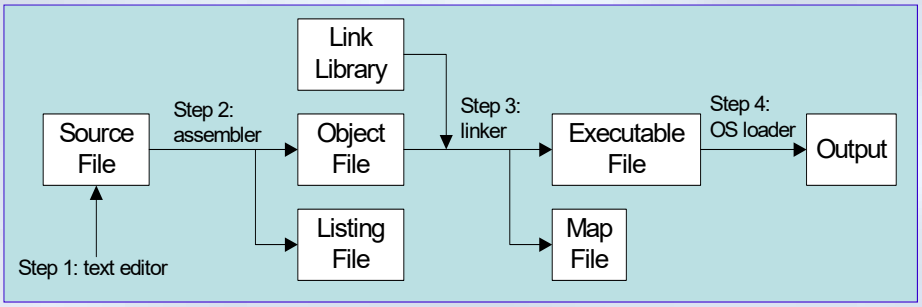
File .COM template, Version 1

```
.model tiny     ; Kiểu bộ nhớ
.386            ;Processor 32bit
.data
     ; Khai báo dữ liệu, hằng số
.code
  .startup      ;Điểm bắt đầu của chương trình
      ;Các câu lệnh của chương trình
  .exit         ;Điểm kết thúc,Trình dịch chèn
                ;tự động hàm 4C, ngắt 21h
end
```


File .COM template, Version 2

```
.386            ;Cho Intel 386, dùng với các thanh ghi 32bit
Prog  SEGMENT
   ASSUME CS: Prog,DS: Prog,SS: Prog,ES: Prog
   ORG 100h
Start:
   Jmp MainProg     ;Nhảy tới phần thân chương trình
        ; Khai báo dữ liệu, hằng, chương trình con..
   MainProg:
                ;Các câu lệnh của chương trình chính
   Int 20h      ;Kết thúc chương trình .COM
Prog  ENDS
END Start ;Start là đia chỉ bắt đầu
```

Dịch chương trình hợp ngữ



- Dịch hợp ngữ: TASM, MASM, NASM
  - Tạo ra file đối tượng (.OBJ)
  - Tạo ra danh sách (.LST): chứa thông tin về việc dịch
- Liên kết các file đối tượng, thư viện
  - Tạo file thực thi (.COM, .EXE)
  - Tạo ra file bản đồ (.MAP): Thông tin về mỗi đoạn

Dịch chương trình bởi TASM&TLINK

C\>Tasm test.asm → Tạo file .Obj

C:\>Tlink Test.obj /3 → Tạo file .Exe

C:\>Tlink /t/3 Test.obj → Tạo file .Com

Xử lý theo lô: Tạo File makeCom.bat
- Cls
- echo Compile and creat com file
- tasm %1.asm
- tlink /t /3 %1.obj
- del %1.obj

C:\> makeCom Test (Enter)

Ví dụ → Hello world ! (File .EXE)

```
.MODEL Large                ;Chương trình lớn, gồm nhiều đoạn
.386                        ; Intel 386
.STACK 100h                 ; Đoạn ngăn xếp 256byte
.DATA                       ; Đoạn dữ liệu
        Msg Db 'Hello world',0Dh,0Ah,'$'
.CODE                       ; Đoạn lệnh
mainProc PROC
   mov ax,@data     ; initialize DS
   mov ds,ax
   MOV AH, 9
   MOV DX, OFFSET Msg
   Int 21h
   .exit                ;Kết thúc chương trình (AH:4Ch, Int21h)
mainProc ENDP
END mainProc            ;mainProc là tên của chương trình chính
```


Ví dụ → Hello world ! (File .COM)

```
.model tiny ;
.386                    ;Processor 32bit
.data
    Msg Db 'Hello world',0Dh,0Ah,'$'
.code
   .startup
      mov dx, Offset Msg  ;printf(msg)
      mov Ah, 9
      int 21h
      mov ah, 0h              ;getch()
      int 16h
   .exit
end
```