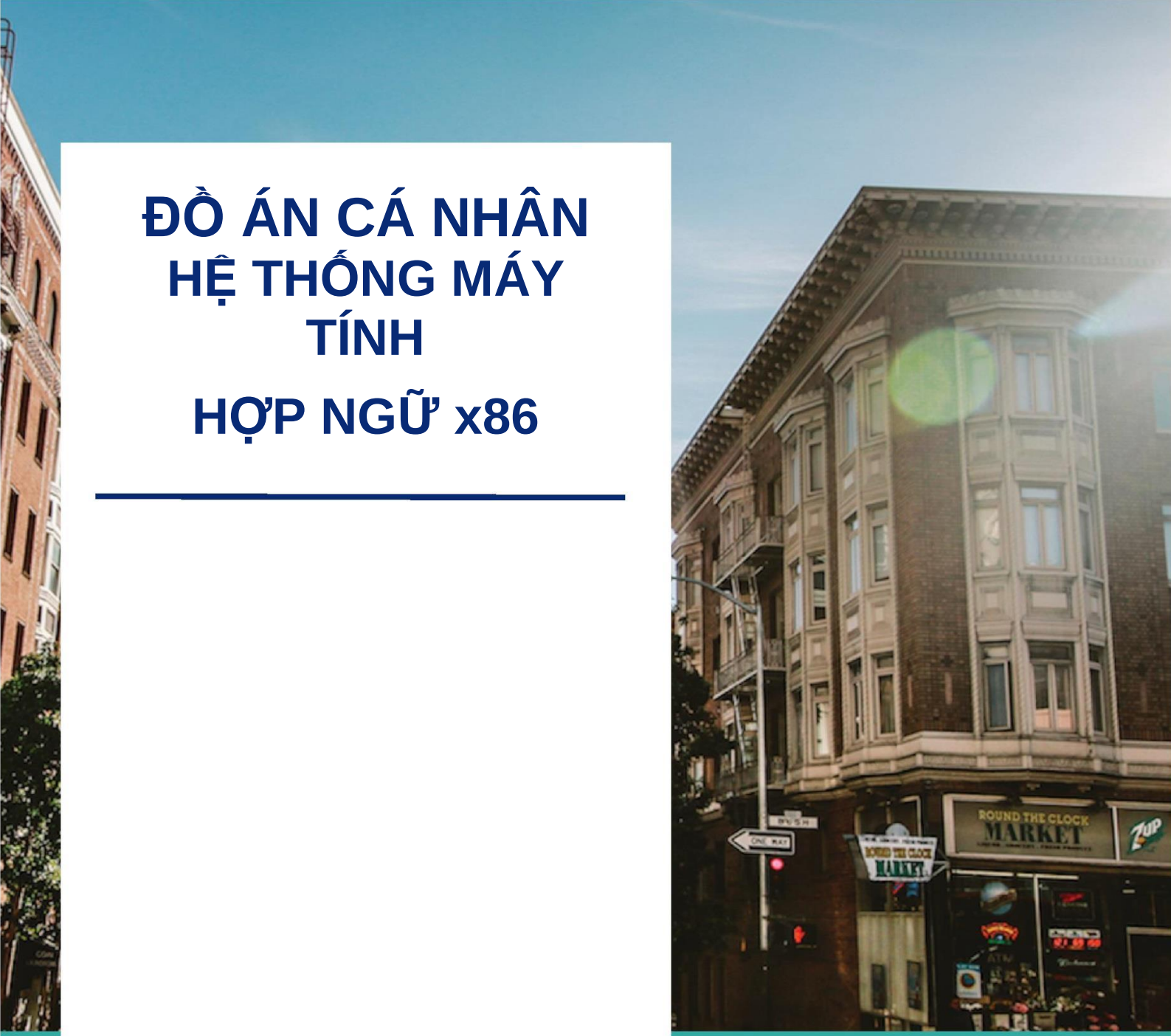
ĐỒ ÁN CÁ NHÂN
HỆ THỐNG MÁY TÍNH
HỢP NGỮ x86



30 THÁNG MƯỜI HAI

VNUHCM-US
20CLC08
GV HƯỚNG DẪN: THẦY LÊ QUỐC HÒA
Tác giả: LÊ ĐỨC ĐẠT - 20127674

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHTN
ĐHQG-HCM

1. GIỚI THIỆU:

a. Hợp ngữ là gì?

Hợp ngữ là ngôn ngữ trung gian giữa ngôn ngữ lập trình bậc cao và ngôn ngữ máy. Nó là một cấp trên ngôn ngữ máy. Hợp ngữ dễ hiểu hơn ngôn ngữ máy nhưng khó hơn các ngôn ngữ lập trình bậc cao. Ngôn ngữ này còn được gọi là ngôn ngữ cấp thấp vì nó gần với cấp độ phần cứng. Để HTTL viết chương trình hiệu quả bằng Assembly, người lập trình cần hiểu rõ về kiến trúc máy tính và cấu trúc thanh ghi. Một trình biên dịch đặc biệt được gọi là trình hợp dịch được sử dụng để chuyển đổi các lệnh của hợp ngữ sang mã máy hoặc mã đối tượng.

b. x86 là gì?

Thuật ngữ x86 dùng để chỉ tới kiến trúc tập lệnh của dòng vi xử lý 8086 của Intel. 8086 được Intel đưa ra năm 1978.

Intel xem dòng phát triển 8086 là IA-32. Kiến trúc x86 này rất phổ biến cho các thế hệ máy tính cá nhân đang hiện hữu trong nhiều gia đình. Kiến trúc x86 gần như chiếm toàn bộ thị phần máy tính cá nhân,máy workstation và cả server thậm chí siêu máy tính. Vì tính phổ biến của nó và hỗ trợ tài liệu rất tốt từ Intel nên x86 được rất nhiều lập trình phần mềm viết chương trình chạy trên nó. Phần mềm được viết cho x86 bao gồm các nền OS: MS DOS, Window, Linux, BSD và các biến thể Unix.

Kiến trúc x86 không phổ biến hoặc phù hợp lắm với hệ thống nhúng. Nếu kiến trúc x86 được Intel gọi là IA-32 thì Intel còn có thế hệ không cùng kiến trúc là IA-64 hay Itanium. Itanium có sự tiến bộ hơn so với x86 với thiết kế ban đầu là 64 bit. Ngoài Intel sản xuất chip kiến trúc x86 còn có: AMD, VIA.

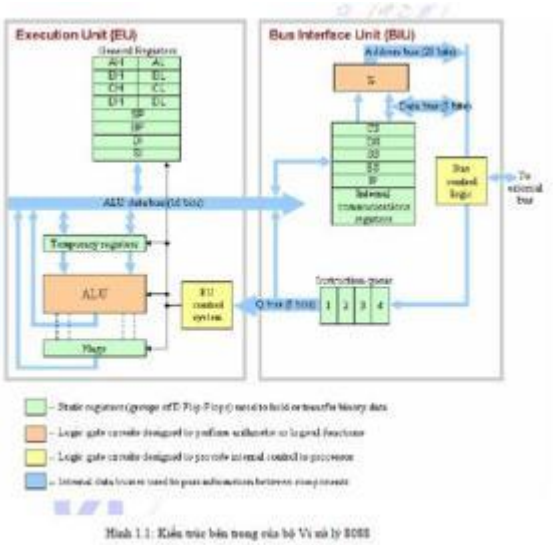
c. Vì sao ta lại phải học hợp ngữ?

Có - lý do chính để học lắp ráp cho các nhà phát triển C và C ++ là nó giúp hiểu những gì đang xảy ra dưới lớp mã C và C ++. Không phải là bạn sẽ thực sự viết mã trong assembly, nhưng bạn sẽ có thể nhìn vào quá trình tháo gỡ mã để đánh giá hiệu quả của nó và bạn sẽ hiểu các tính năng C và C ++ khác nhau hoạt động tốt hơn nhiều như thế nào.

2.NỀN TẢNG LÍ THUYẾT – MÔ HÌNH HOẠT ĐỘNG:

Kiến trúc bộ vi xử lý 8086/80386 (X86 – 32bit):

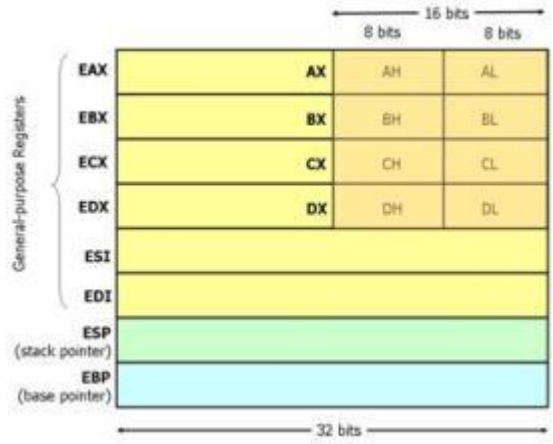
Cơ bản là giống hình dưới đây, chỉ có thay đổi là các thanh ghi chuyển từ 16 bit thành dạng 32 bit nên sẽ hơi khác một chút! (xem Hình 2: cấu trúc các thanh ghi chung ở phần 3.a.)



1. **Các thanh ghi (register) cần nhớ:**

Đối với cấu trúc Intel 32 bit, chúng ta có các nhóm thanh ghi chính được liệt kê bên dưới (mỗi thanh ghi có độ dài 32 bit)

a. **Các thanh ghi dùng chung: Những thanh ghi được CPU sử dụng như bộ nhớ siêu tốc trong các công việc tính toán, đặt biến tạm, hay giữ giá trị tham số. Chúng ta hay gặp bốn bốn thanh ghi chính là: EAX, EBX, ECX, EDX.Các thanh ghi này cũng có chức năng tương tự các thanh ghi AX BX CX DX (trong hệ thống cũ 16 bit).Tuy nhiên trong vì đây là các thanh ghi 32bit nên ta cũng có thể sử dụng các thanh ghi trong đó như là các thanh ghi 8bit 16bit 32bit tùy ý như EAX(32bit) AX(16bit) AH(8bit) AL(8bit)**



Hình 2: cấu trúc các thanh ghi chung.

Công dụng:

- **EAX : Dùng cho các phép toán,logic,chuyển dữ liệu**
- **EBX : Dùng để lưu giữ địa chỉ**
- **ECX : Dùng làm bộ đếm các vòng lặp**
- **EDX : Dùng phối hợp EAX trong phép nhân chia …**

Ngoài ra còn có thanh ghi địa chỉ và con trỏ bộ nhớ:

- **Thanh ghi EIP : thanh ghi con trỏ lệnh.đây là thanh ghi rất quan trọng.nó sẽ trỏ đến vị trí của lệnh kế tiếp khi 1 lệnh được thực thi.Vì thế nó là 1 thanh ghi đặc biệt mà nó sẽ không tham gia vào các lệnh như là 1 toán hạng được.**
- **Thanh ghi ESI và EDI : các thanh ghi chỉ mục(index).Hay sử dụng trong chuỗi hoặc mảng**
- **Thanh ghi ESP : trỏ đến đầu stack.stack là 1 vùng trong bộ nhớ chứa các biến tạm thời,các dữ liệu và địa chỉ Stack làm việc theo nguyên tắc LIFO(last in first out).**
- **Thanh ghi EBP : là 1 thanh ghi dùng trong việc truy xuất dữ liệu trong stack.Công dụng EBP còn trong việc lưu trữ vị trí lệnh tiếp theo khi ứng dụng thực hiện lệnh CALL để gọi hàm(chương trình con) để return lại lệnh kế tiếp.**

1. **Các thanh ghi cờ:**

Gồm thanh ghi cờ trạng thái và thanh ghi cờ điều khiển.Các cờ hay dùng trong các phép tính,điều kiện thực hiện các phép nhảy

CF : ON khi cộng có nhớ,trừ có mượn ở bit cao
AF : ON khi có mượn hay có nhớ ở bit3
SF : ON khi phép tính có bit cao nhất âm
OF : ON khi phép tính có dấu sai.nêu cộng 2 số cùng dấu mà kết quả có dấu ngược
thì có tràn xảy ra.Khi ta cộng hai sô khác dấu thì không bao giờ có tràn
PF : ON khi 8bit thấp là số chẵn
ZF : ON khi phép toán trả về giá trị 0

Sau đây là bảng các lệnh nhảy có điều kiện :

Jxx - Jump Instructions Table

Mnemonic	Meaning	Jump Condition
JA	Jump if Above	CF=0 and ZF=0
JAE	Jump if Above or Equal	CF=0
JB	Jump if Below	CF=1
JBE	Jump if Below or Equal	CF=1 or ZF=1
JC	Jump if Carry	CF=1
JCXZ	Jump if CX Zero	CX=0
JE	Jump if Equal	ZF=1
JG	Jump if Greater (signed)	ZF=0 and SF=OF
JGE	Jump if Greater or Equal (signed)	SF=OF
JL	Jump if Less (signed)	SF != OF
JLE	Jump if Less or Equal (signed)	ZF=1 or SF != OF
JMP	Unconditional Jump	unconditional
JNA	Jump if Not Above	CF=1 or ZF=1
JNAE	Jump if Not Above or Equal	CF=1
JNB	Jump if Not Below	CF=0
JNBE	Jump if Not Below or Equal	CF=0 and ZF=0
JNC	Jump if Not Carry	CF=0
JNE	Jump if Not Equal	ZF=0
JNG	Jump if Not Greater (signed)	ZF=1 or SF != OF
JNGE	Jump if Not Greater or Equal (signed)	SF != OF
JNL	Jump if Not Less (signed)	SF=OF
JNLE	Jump if Not Less or Equal (signed)	ZF=0 and SF=OF
JNO	Jump if Not Overflow (signed)	OF=0
JNP	Jump if No Parity	PF=0
JNS	Jump if Not Signed (signed)	SF=0
JNZ	Jump if Not Zero	ZF=0
JO	Jump if Overflow (signed)	OF=1
JP	Jump if Parity	PF=1
JPE	Jump if Parity Even	PF=1
JPO	Jump if Parity Odd	PF=0
JS	Jump if Signed (signed)	SF=1
JZ	Jump if Zero	ZF=1

- **Các hàm/khai báo/chức năng:**

+ .model small: Khai báo kích thước.
+ .stack 100h: phục vụ vào việc sử dụng ngăn xếp(stack – DSA…).
+ .data: Cho dòng muốn in ra để hiển thị và chú thích them.
+ .code: Nơi để thao tác với hợp ngữ.

a. DEMO 1: (Xuất dòng msg).

```
.model small
.stack 100h
.data
    msg db "LE DUC DAT - 20127674 $"

.code
    mov ax,@data
    mov ds,ax

    lea dx, msg
    mov ah,9
    int 21h

    mov ah,4ch
    int 21h

end
```

- **Cách thức vận hành:**

+ Đầu tiên, in ra dãy kí tự và dung kiểu dữ liệu db (move ax, @data).

+ Ở data, nạp dữ liệu data đã được định nghĩa ở trên vào thanh ghi ax (mov ds, ax).

+ Sau đó, dữ liệu được chuyển đến thanh ghi ds (data segment).

+ Dùng hàm 9 của ngắt 21 để in chuỗi kí tự ra màn hình

(mov ah, 9

int 21h).

+ Dùng hàm ngắt 4ch để dừng chương trình.

b. DEMO 2: (Tính tổng 2 số nhập từ bàn phím).

```
.model small
.stack 100h
.data
    msg1 db 10,13, 'nhap so thu nhat: $'
    msg2 db 10,13, 'nhap so thu hai: $'
    msg3 db 10,13, 'tong cua hai so la: $'

    num1 db ?
    num2 db ?
    num3 db ?

.code
    ;lay du lieu tu data vao ds
    mov ax,@data
    mov ds,ax
    ;xuat msg1
    mov ah,9h
    lea dx,msg1
    int 21h
    ;nhap 1 ki tu tu ban phim
    mov ah, 1h
    int 21h

    sub al, 30h
    mov num1, al
    ;xuat msg2
    mov ah, 9h
    lea dx, msg2
    int 21h
    ;nhap 1 ki tu tu ban phim
    mov ah, 1h
    int 21h

    sub al, 30h
    mov num2, al
    ;xuat msg3
    mov ah, 9h
    mov dx, offset msg3
    int 21h
    ;tinh tong
    mov al, num1
    mov bl, num2
    add al, bl

    add al, 30h
    mov num3, al
    ;xuat tong
    mov ah, 2h
    mov dl, num3
    int 21h

    ;thoat huong trinh
    mov ah, 4ch
    int 21h
end
```

+++ Ở đây, ta khai báo thêm 3 biến num1, num2, num3 nhằm mục đích tính toán.

+++ Giải thích "10,13,": Trong ASCII, 10 có nghĩa là dòng mới, 13 có nghĩa là chuyển dòng/xuống dòng nên "10,13," có nghĩa tương tự như endline trong c++ hay \n trong C, MARS.

- **Cách thức vận hành**

+ B1: lấy dữ liệu từ data vào ds(Tương tự cách làm ở demo 1).

+ B2: Xuất msg1 "Nhap so thu nhat: " vào dx.

+ B3: Ta nhập 1 kí tự từ bàn phím và trừ đi 30h.

+ B4, B5: Tương tự B2, B3.

+ B6: Xuất msg3(dĩ nhiên phải đi kèm với kết quả phép tính +).

+ B7: Tính tổng bằng cách dung hàm add al, bl(al từ biến num1, bl từ biến num2).

+ B8: Cộng lại cho 30h để ra kết quả đúng nhất(Trong demo có nói rõ).

+ B9: Xuất tổng ra màn hình qua biến num3.

+ B10: Thoát chương trình.

3. DEMO(LINK YOUTUBE):

https://www.youtube.com/watch?v=4VQMBBRYVRw.

4. TƯ LIỆU THAM KHẢO:

- https://www.youtube.com/watch?v=hweIx0SFnuQ.
- https://www.youtube.com/watch?v=3GwfTVOYQuA.
- https://tailieuhoctap123blog.files.wordpress.com/2016/06/bai-giang-asm-cho-tdh-24-02-2014.pdf.
- http://flatassembler.net/docs.php?article=manual.
- https://wikimaytinh.com/bang-ma-ascii-la-gi.html.
- https://www.jundat95.com/2015/10/tap-lenh-vong-lap-loop-lea-trong-asm.html.
- https://www.youtube.com/watch?v=_fThJ6wvHfQ.
- https://darkknightjk.wordpress.com/2013/02/16/hoc-lap-trinh-hop-ngu-assembly-cung-op-op-ly-thuyet-part-1/.