Week 2

Overview

Tập lệnh assembly nhìn chung có thể chia làm 8 loại:

1. Data Movements Instructions:

- mov, lea,les,push, pop, pushf,popf

2. Conversions

- cbw,cwd,xlat

3. Arithmetic instructions:

-add,inc,sub,dec,cmp,neg,mul,imul,div,idiv

4. Logical, shift,rotate, and bit instructions:

- and,or,xor,not,shl,shr,rcl,rcr

5. I/O instructions:

-in/out

6. String instructions

- movs,stos,lods

7. Program flow control instructions

- jmp,call,ret, conditional jump

8. Miscellaneus intructions

- clc,stc,cmc

Processor Status Register (Flags)

Đưa ra thông tin về chế độ hiện tại của CPU và thông tin về tình trạng của 1 số instruction.

- Overflow : Ảnh hưởng bởi loại 3,4,8 của tập lệnh Assembly. Ví dụ nếu ta cộng 0x7FFF và 0x0001 thì sẽ vượt quá khả năng tính toán của thanh ghi. Lúc đó cờ Overflow sẽ đc set = 1

- Direction: Ảnh hưởng bởi String instruction. Khi Direction flag = 0 thì string sẽ được lưu theo thứ tự từ địa chỉ thấp → cao. Và =1 thì chiều ngược lại.

-Interupt : kiểm soát khả năng phản hồi việc yêu cầu gián đoạn. 1 số chương trình bắt buột CPU không bị gián đoạn.

- Trace: Được sử dụng trong các chương trình debug. khi set =1 thì CPU sẽ làm gián đoạn từng câu lệnh 1 và gởi tới các debugger. Trace bắt buộc phải phải dưa vào stack để hoạt động.

- Sign: dùng để tính toán số âm.

- Zero Flag: dùng khi tính toán sinh ra số 0. Thường được dùng trong các biểu thức so sánh và logical.

- Auxiliary Carry Flag : dùng để tính toán theo hệ số BCD. Vì phần lớn các chương trình không hề dùng hệ số BCD nên hiếm khi nó được dùng.

- Parity: Xác định bit chẵn lẽ.

- Carry:chỉ rõ unsigned overflow. (Tương tự với overflow flag)

1 số lệnh cơ bản:

Data movement instructions:

1. Mov

Dạng

mov reg , reg1

mov mem , reg

mov reg , mem

mov mem , immediate data

mov reg , immediate data

mov ax/al , mem

mov mem , ax/al

mov segreg , mem(16)

mov segreg , reg(16)

mov mem(16), segreg

mov reg(16) , segreg

Lưu ý: Không tồn tại “mov mem , mem” và “mov segreg , immediate data”

2 hạng tử phải có cùng độ lớn.

Lệnh tương đương:

operand1: = operand2

2.xchg

Lệnh dùng để swap 2 giá trị của 2 hạng tử.

Dạng

xchg operand1, operand2

3. lds, les, les, lgs, lss

load địa chỉ vào thanh ghi

Dạng

LxS dest, source

Tương đương:

lds reg(16), mem(32)

reg(16):= [mem(32)]

ds:=[mem(32)+2]

les reg(16), mem(32)

reg(16):= [mem(32)]

es:=[mem(32)+2]

lfs reg(16), mem(32)

reg(16):= [mem(32)]

fs:=[mem(32)+2]

lgs reg(16), mem(32)

reg(16):= [mem(32)]

gs:=[mem(32)+2]

lss reg(16), mem(32)

reg(16):= [mem(32)]

ss:=[mem(32)+2]

4. Lea

Tương tự như các lệnh trên. Nhưng địa chỉ load là địa chỉ cuối cùng sau khi được tính toán xong.  
  
Ví dụ:

lea ax, [bx]

Chỉ việc chép địa chỉ [bx] vào ax, thực tế dùng mov nhanh hơn nhiều.

lea ax, 3[bx]

Chép địa chỉ của 3[bx] vào ax. Thực hiện liên tiếp 2 thứ: Cộng 3 vào giá trị của bx và lưu vào ax. Lần này lea làm tốt hơn mov nhiểu.  
  
5. Push và pop

Push : Chuyển data vào Hardware Stack  
Pop: Chuyển data ở đỉnh ra khỏi stack.  
  
Miêu tả:(32 bits)  
  
push:  
 sp:=sp-4

[ss:sp] := 32 bit operand

pop:

[ss:sp] := 32 bit operand

sp:=sp+4  
  
Lưu ý:   
- Stack luôn nằm trong stack segment( ở vị trí của ss)

- Stack đi ngược xuống trong bộ nhớ. Ví thế nếu push giá trị vào stack, CPU sẽ lưu nó ở giá trị bộ nhớ thấp.  
- Stackpointer luôn luôn lưu vị trí đỉnh của stack.(Giá trị cuối cùng đc push)  
  
Stack cũng có thể dùng để lưu trữ tạm thời thanh ghi và biến.  
  
6. LAHF và SAHF  
- lahf (Load ah from flag) và sahf(store) là 2 lệnh tương thích với 1 con chip cũ của Intel.

Conversions  
  
1. movzx, movsx, cbw, cwd, cwde , cdq  
cbw - chép 7 bit của al và nối vào ax.

cwd - chép 16 bit của ax vào dx:ax

cwde - chép 16 bit của ax vào eax

cdq - chép 32 bit của eax vào eax:eax  
  
movsx cũng làm công việc tương tự. Chỉ khác là nó có thêm các toán tử.

movzs chuyển các giá trị unsign.

2. bswap

Chuyển từ 32 bit little edian và big edian.