

LẬP TRÌNH HỢP NGỮ X86

Mục đích

- Làm quen với ngôn ngữ lập trình Assembly trên kiến trúc x86
- Biết cách viết, dịch, chạy và chẩn lỗi (debug) một vài chương trình đơn giản

Tóm tắt lý thuyết

Hợp ngữ (assembly) là ngôn ngữ bậc thấp, giúp cho người lập trình không phải ghi nhớ mã máy (opcode) mà sử dụng các từ ngữ gọi nhớ (pseudo-code) gần với ngôn ngữ tự nhiên để miêu tả công việc cần thực hiện. Tuy vậy, assembly rất gần với ngôn ngữ máy, đòi hỏi người lập trình phải hiểu biết tương đối đầy đủ về cấu trúc phần cứng máy tính.

Với mỗi kiểu kiến trúc của bộ vi xử lý, có một bộ lệnh riêng, do đó, có một ngôn ngữ assembly riêng cho nó. Ở đây, chúng ta nghiên cứu assembly cho các bộ vi xử lý Intel thuộc họ x86. Các chương trình sẽ được viết cho chế độ thực (real mode) trong DOS và được biên dịch bằng Netwide Assembler.

Cấu trúc của một chương trình hợp ngữ

```
section .data
    <Khai báo dữ liệu (kiểu tĩnh)>
section .bss
    <Khai báo dữ liệu (kiểu động)>
section .code
_mainCRTStartup:    ; Nhận bắt đầu chương trình, có thể thay đổi
                   ; tùy thuộc vào loại project tạo trên Visual C
                   <Các lệnh thực thi>
```

Ví dụ: Chương trình sau in ra màn hình dòng chữ “Hello World”

```
global _mainCRTStartup    ; Hàm chính của chương trình
extern _ExitProcess@4      ; Gọi hàm thoát của Windows API
extern _GetStdHandle@4     ; Gọi hàm xử lý của Windows API
extern _WriteFile@20       ; Gọi hàm xuất của Windows API

section .data
hello_world      db    'Hello World', 10, 0    ; Khai báo chuỗi hello_world
bytes_written    dd    0                      ; Trả về chuỗi word 32-bit từ WriteFile
output_handle    dd    0                      ; The standard output handle

section .code
_mainCRTStartup:
    push        -11                      ; Mã thực hiện xuất
    call        _GetStdHandle@4          ; Gọi Hàm GetStdHandle của Windows API
    mov         [output_handle], eax     ; Lưu handle để xuất màn hình

    push        0
    push        dword bytes_written
    push        13                      ; Chiều dài của chuỗi
    push        dword hello_world       ; Lấy địa chỉ của chuỗi
    push        dword [output_handle]   ; Gọi Handle để xuất màn hình
    call        _WriteFile@20           ; Xuất chuỗi ra màn hình
```

```
push    0
call    _ExitProcess@4      ; Gọi hàm ExitProcess thoát chương trình
```

Lưu ý:

- Mọi chương trình đều phải có đoạn code thoát khỏi chương trình, nếu không chương trình sẽ không dừng khi hết chương trình của mình.

Khai báo biến trong hợp ngữ

Cú pháp trong .data:

<tên biến> d<Kiểu DL> <giá trị khởi tạo>

Cú pháp trong .bss:

<tên biến> res<Kiểu DL> <giá trị khởi tạo>

Các kiểu dữ liệu:

b (1 byte), w (2 bytes), d (4 bytes), q (8 bytes), t (10 bytes)

Ví dụ:

Khai báo trong .data

```
message    db 'Hello world!'
msglength  db 12
bufferize  dw 1024      ; Khai báo một word có kích thước là 1024
```

Khai báo trong .bss

```
filename    resb 255
number      resb 1
bignum      resw 1
realarray   resq 10
```

Các thanh ghi trong NASM

Dữ liệu của thanh ghi:

16-bit của thanh ghi

ah	ax	al
----	----	----

8-bit 16-bit

32-bit mở rộng

--

Các thanh ghi:

eax		ah	ax	al	Accumulator
ebx		bh	bx	bl	Base Index
ecx		ch	cx	cl	Count
edx		dh	dx	dl	Data
esp			sp		Stack Pointer
ebp			bp		Base Pointer
edi			di		Destination Index
esi			si		Source Index

Một số lệnh NASM cơ bản

Lệnh	Ghi chú	Ví dụ
Nhóm lệnh số học		
inc destination		inc ebx inc byte [edi] ;Adds 1 to any reg/mem except seg
dec destination		dec dl dec edi
add destination, source	$destination = destination + source$	add al, [ARRAY + esi] adc ecx, ebx ;Adds registers + Carry flag. ;Used for adding 64 bit nums. xadd ecx, ebx ;ecx=ecx+ebx, ebx=original ecx.
sub destination, source	$destination = destination - source$	sub eax, ebx ; eax = eax - ebx sbb ecx, ebx ; Subs registers - Carry flag.
imul/div mul/div	Nhân/Chia có dấu của số nguyên. Không dấu. al luôn là số hạng của phép nhân (hoặc ax hoặc eax). Kết quả được đưa vào ax (hoặc dx và ax hoặc edx hoặc eax).	mul bl ; ax=ax*bl (unsigned) imul bx ; dx ax=ax*bx (signed) imul cx, dx, 12H ; Special, cx=dx*12H (signed only)) mul ecx ; edx eax=eax*ecx div cl ; ah al=ax/cl, unsigned quotient in al, remainder in ah idiv cx ; dx ax=(dx ax)/cx
Nhóm lệnh logic		
and		and al, bl ;al=ax AND bl
or		or eax, 10 ;eax=eax OR 0000000AH
xor		xor ah, ch ;ah=ax XOR ch
test		test al, 4 ;Tests bit 2 in al – 00000100 jz LABEL ;Jump to LABEL if bit 2 is zero.
not		not ebx
neg		neg TEMP
shift		shl eax, 1 ;eax is logically shifted left 1 bit pos. sar esi, cl ;esi is arithmetically shifted right
rotate		rol si, 14 ;si rotated left by 14 places. rcr bl, cl ;bl rotated right cl places through carry.
Nhóm lệnh dịch chuyển		
mov	Lệnh dịch chuyển dữ liệu (kiểu	mov eax, [bar] ;Refers to the contents

	bytes, words and doublewords) giữa các thanh ghi và giữa thanh ghi và vùng nhớ.	<i>of bar</i> <code>mov eax, bar ;Refers to the address of bar</code> <code>mov eax,table[ebx] ;ERROR</code> <code>mov eax,[table+ebx] ;O.K.</code> <code>mov eax,[es:edi] ;O.K.</code> <code>data dw 0 ;Data type defined as double word.</code> ... <code>mov [data], 2 ;Doesn't work.</code> <code>mov word [data], 2 ;O.K.</code>
push, pop	<p>6 dạng của phương thức push và pop: Thanh ghi, bộ nhớ (từ bộ nhớ đến bộ nhớ), hằng số, thanh ghi đoạn, cờ hiệu, và tất cả các thanh ghi</p> <p>push: Nguồn của dữ liệu có thể là: Bất kỳ thanh ghi 16 hoặc 32 bit, hằng số, thanh ghi đoạn, word hoặc doubleword của bộ nhớ</p> <p>pop: Nguồn của dữ liệu có thể là: Bất kỳ thanh ghi 16 hoặc 32 bit, thanh ghi đoạn (ngoại trừ CS), word hoặc doubleword của bộ nhớ</p>	<pre>push dword input_filename_ptr push eax push 1</pre>
lea	Lấy địa chỉ của dữ liệu vào bất kỳ thanh ghi 32 bit nào.	<code>lea eax, [esi+edi]</code>

Các cờ hiệu:

Cờ	Ý nghĩa
Z	Result zero
C	Carry out – Cờ nhớ
A	Half carry out – Cờ nhớ phụ
S	Result positive
P	Result has even parity
O	Overflow occurred – Cờ tràn

Lệnh so sánh: cmp Rs1, Rs2

Ví dụ: cmp AL, DL

Một số ví dụ khi so sánh 2 thanh ghi AL và DL:

AL	DL	CF	ZF	SF	OF	PF	AF
56	57	1	0	1	0	1	1
200	101	0	0	0	1	1	0
101	200	1	0	1	1	0	1
200	200	0	1	0	0	1	0
-105	-105	0	1	0	0	1	0
-125	-124	1	0	1	0	1	1
-124	-125	0	0	0	0	0	0

Nhóm lệnh nhảy:

Lệnh nhảy ở các cờ đơn		
Lệnh	Ý nghĩa	Nhảy khi
Kiểm tra cờ 0: jz je jnz jne jecxz	jump if zero jump if equal jump if not zero jump if not equal jump if ECX = 0	ZF = 1 ZF = 0 ECX = 0 (Không có cờ để kiểm tra)
Kiểm tra cờ nhớ: jc jnc	jump if carry jump if no carry	CF = 1 CF = 0
Kiểm tra tràn: jo jno	jump if overflow jump if no overflow	OF = 1 OF = 0
Kiểm tra dấu: js jns	jump if (negative) sign jump if no (negative) sign	SF = 1 SF = 0
Kiểm tra parity(tính chẵn): jp jpe jnp jpo	jump if parity jump if parity is even jump if not parity jump if parity is odd	PF = 1 PF = 0
Nhảy khi so sánh 2 số không dấu		
Lệnh	Ý nghĩa	Điều kiện kiểm tra
je jz	jump if equal jump if zero	ZF = 1
jne jnz	jump if not equal jump if not zero	ZF = 0
ja jnbe	jump if above jump if not below or equal	CF = 0 và ZF = 0
jae jnb	jump if above or equal jump if not below	CF = 0
jb	jump if below	CF = 1

jnae	jump if not above or equal	
jbe	jump if below or equal	CF = 1 hoặc ZF = 1
jna	jump if not above	
Nhảy khi so sánh 2 số có dấu		
Lệnh	Ý nghĩa	Điều kiện kiểm tra
je	jump if equal	ZF = 1
jz	jump if zero	
jne	jump if not equal	ZF = 0
jnz	jump if not zero	
jg	jump if greater	ZF = 0 và SF = OF
jnle	jump if not less or equal	
jge	jump if greater or equal	SF = OF
jnl	jump if not less	
jl	jump if less	SF ≠ OF
jnge	jump if not greater or equal	
jle	jump if less or equal	ZF = 1 hoặc SF ≠ OF
jng	jump if not greater	

System Call:

Lệnh syscall làm treo sự thực thi của chương trình và chuyển quyền điều khiển cho HĐH (bằng cách gọi các hàm của Windows API).

Bảng các system call

Lệnh hợp ngữ	Ý nghĩa	Hàm trên C++
push -11 call _GetStdHandle@4 mov [output_handle], eax	Bắt đầu chương trình xuất.	hFile = GetStdHandle(-11);
push -10 call _GetStdHandle@4 mov [input_handle], eax	Bắt đầu chương trình nhập.	hFile = GetStdHandle(-10);
push 0 call _ExitProcess@4	Thoát chương trình.	return 0;
push 0 push dword bytes_written push length push dword output_string push dword [output_handle] call _WriteFile@20	Xuất. bytes_written: kích thước của chuỗi xuất ra. length: chiều dài của chuỗi output_string: địa chỉ của chuỗi xuất ra output_handle: lời gọi hàm đến GetStdHandle	WriteFile(hFile, output_string, length, &bytes_written, 0);
push 0 push dword bytes_read push length push dword input_string push dword [input_handle] call _ReadFile@20	Nhập. bytes_read: kích thước của chuỗi xuất nhập vào. length: chiều dài của chuỗi input_string: địa chỉ của chuỗi nhập vào output_handle: lời gọi hàm đến GetStdHandle	ReadFile(hFile, &input_string, length, bytes_read)

Stack**Tài liệu tham khảo**

[1]. <http://www.nasm.us/doc/nasmdoc0.html> The Netwide Assembler: NASM.

Bài tập

Hãy viết chương trình hợp ngữ NASM trên Window (không dùng lệnh giả) để giải quyết các bài toán sau:

1. Nhập vào một chuỗi, xuất lại chuỗi đó ra màn hình (echo).
Ví dụ:
Nhập một chuỗi: Hello
Chuỗi đã nhập: Hello
2. Nhập vào một ký tự, xuất ra ký tự liền trước và liền sau.
Ví dụ:
Nhập một ký tự: b
Ký tự liền trước: a
Ký tự liền sau: c
3. Nhập vào một ký tự hoa, in ra ký tự thường.
Ví dụ:
Nhập một ký tự: A
Ký tự thường: a
4. Nhập từ bàn phím 2 số nguyên, tính tổng, hiệu, tích, thương của 2 số.
Ví dụ:
Nhập số thứ nhất: 7
Nhập số thứ hai: 4
Tổng: 11
Hiệu: 3
Tích: 28
Thương: 1 dư 3
5. Nhập vào 2 số nguyên, xuất ra phép so sánh giữa 2 số.
Ví dụ:
Nhập số thứ nhất: 6
Nhập số thứ hai: 9
Số lớn hơn là: 9
6. Nhập một ký tự từ bàn phím. Nếu ký tự vừa nhập thuộc [0-9], [a-z], [A-Z] thì xuất ra màn hình ký tự đó và loại của ký tự đó (số, chữ thường, chữ hoa).
Ví dụ:
Nhập vào một ký tự: 5
Ký tự vừa nhập: 5 là số
Nhập vào một ký tự: f
Ký tự vừa nhập: f là chữ thường

Nhập vào một ký tự : D

Ký tự vừa nhập : D là chữ hoa

7. Nhập một mảng các số nguyên n phần tử, xuất mảng đó ra màn hình.

Ví dụ:

Nhap mang cac so nguyen: 1 2 3 4 5

Mang vua nhap: 1 2 3 4 5

8. Nhập vào một số nguyên n, tính tổng từ 1 đến n.

Ví dụ:

Nhap mot so: 4

Tong tu 1 den 4 la: 10

9. Nhập vào một chuỗi, xuất ra chuỗi ngược.

Ví dụ:

Nhap vao mot chuoi: hello

Chuoi nguoc la: olleh