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 gợi 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:        ; Nhãn 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

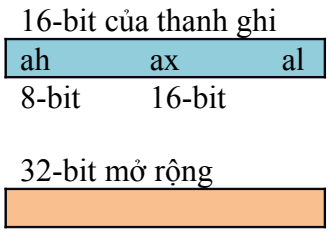
```
message     db 'Hello world!'
msglength   db 12
buffersize  dw 1024         ; Khai báo một word có kích thước là 1024
```

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:



Các thanh ghi:

eax		ah	ax	al	Accumulator
ebx		bh	bx	bl	Base Index
ecx		ch	cx	bl	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/idiv 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=al*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=al AND bl*
or		or eax, 10 *;eax=eax OR 0000000AH*
xor		xor ah, ch *;ah=ah 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 dự liệu (kiểu	mov eax, [bar] *;Refers to the contents*

	bytes, words and doublewords) giữa các thanh ghi và giữa thanh ghi và vùng nhớ.	*of bar* mov eax, bar *;Refers to the address of bar* mov eax,table[ebx] *;ERROR* mov eax,[table+ebx] *;O.K.* mov eax,[es:edi] *;O.K.* data dw 0 *;Data type defined as double word.* ... mov [data], 2 *;Doesn't work.* mov word [data], 2 *;O.K.*
push, pop	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 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 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	push dword input_filename_ptr push eax push 1
lea	Lấy địa chỉ của dữ liệu vào bất kỳ thanh ghi 32 bit nào.	lea eax, [esi+edi]

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
Kiểm tra cờ 0:	
jz	jump if zero
je	jump if equal
jnz	jump if not zero
jne	jump if not equal
jecxz	jump if ECX = 0
Kiểm tra cờ nhớ:	
jc	jump if carry
jnc	jump if no carry
Kiểm tra tràn:	
jo	jump if overflow
jno	jump if no overflow
Kiểm tra dấu:	
js	jump if (negative) sign
jns	jump if no (negative) sign
Kiểm tra parity(tính chẵn):	
jp	jump if parity
jpe	jump if parity is even
jnp	jump if not parity
jpo	jump if parity is odd

Nhảy khi so sánh 2 số không dấu

Lệnh	Ý nghĩa
je jz	jump if equal jump if zero
jne jnz	jump if not equal jump if not zero
ja jnbe	jump if above jump if not below or equal
jae jnb	jump if above or equal jump if not below
jb jnae	jump if below jump if not above or equal

jbe jna	jump if below or equal jump if not above
Nhảy khi so sánh 2 số có dấu	
Lệnh	**Ý nghĩa**
je jz	jump if equal jump if zero
jne jnz	jump if not equal jump if not zero
jg jnle	jump if greater jump if not less or equal
jge jnl	jump if greater or equal jump if not less
jl jnge	jump if less jump if not greater or equal
jle jng	jump if less or equal jump if not greater

System Call:

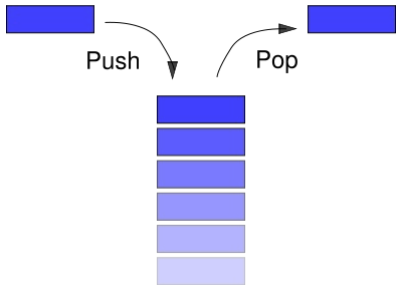
Lệnh syscall làm treo sự thực thi của chương trình và chuyển quyền điều khiển cho HĐH (bằng cách gọi các hàm của Windows API).

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 và Gọi hàm

Stack (ngăn xếp) là vùng nhớ đặc biệt được truy cập theo cơ chế "vào trước ra sau" (LIFO – Last In First Out), nghĩa là dữ liệu nào đưa vào sau sẽ được lấy ra trước. Ngăn xếp gồm nhiều phần tử, mỗi phần tử là một từ (4 byte).



Thanh ghi ESP đóng vai trò là con trỏ ngăn xếp (stack pointer), luôn chỉ đến đỉnh của stack. Stack phát triển theo chiều giảm của địa chỉ vùng nhớ (đỉnh của stack luôn có địa chỉ thấp).

Hai thao tác cơ bản trong stack là push (đưa một phần tử vào stack) và pop (lấy một phần tử ra khỏi stack).

- push: giảm ESP đi 4, lưu giá trị vào ô nhớ mà ESP chỉ đến.
- pop: copy giá trị trong vùng nhớ được chỉ đến bởi ESP, cộng 4 vào ESP.

Lệnh CALL và RET trong việc gọi chương trình con

- CALL: push địa chỉ lệnh phía sau CALL vào stack và nhảy đến chương trình con.
- RET: pop địa chỉ từ stack ra và jump đến địa chỉ đó.

Ví dụ: đoạn chương trình xuất chuỗi ngược, chuỗi nhập vào là "Hello"

```
start_loop:         push        1
                    lea         eax, [esi+edi]
                    push        eax
                    call        FileWrite
                    dec         edi
                    mov         al, [esi+edi]
                    or          al, al
                    jne         start_loop

                    push        2
                    push        new_lines
                    call        FileWrite

FileWrite:          mov         ebp, esp
                    add         ebp, 4
                    push        0
                    push        dword bytes_written
                    push        dword [ebp+4]
                    push        dword [ebp]
                    push        dword [output_handle]
                    call        _WriteFile@20
                    mov         eax, [bytes_written]
                    ret         8
```

Trạng thái bộ nhớ khi vào thủ tục FileWrite:

High address							
0x00406020	o					ESI	0x00406019
0x0040601C	H	e	l	l		EDI	7
						EAX	0x00406020
						ESP	0x0012FFB8
	1						
	EAX						
0x0012FFB8	ret						
Low address							

Trước khi thực hiện call _WriteFile@20

High address							
0x00406020	o					ESI	0x00406019
0x0040601C	H	e	l	l		EDI	7
						EAX	0x00406020
						ESP	0x0012FFA4
						EBP	0x0012FFBC
	1						
0x0012FFBC	EAX						
0x0012FFB8	ret						
0x0012FFB4	0						
0x0012FFB0	bytes_written						
0x0012FFAC	[ebp+4]						
0x0012FFA8	[ebp]						
0x0012FFA4	[output_handle]						
Low address							

Sau khi thực hiện call _WriteFile@20

High address							
0x00406020	o					ESI	0x00406019
0x0040601C	H	e	l	l		EDI	7
						EAX	0x00406020
						ESP	0x0012FFB8
						EBP	0x0012FFBC
	1						
0x0012FFBC	EAX						
0x0012FFB8	ret						
0x0012FFB4	0						
0x0012FFB0	bytes_written						
0x0012FFAC	[ebp+4]						
0x0012FFA8	[ebp]						
0x0012FFA4	[output_handle]						
Low address							

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:
Nhap mot chuoi: Hello
Chuoi da nhap: Hello

2. Nhập vào một ký tự, xuất ra ký tự liền trước và liền sau.

Ví dụ:
Nhap mot ky tu: b
Ky tu lien truoc: a
Ky tu lien sau: c

3. Nhập vào một ký tự hoa, in ra ký tự thường.

Ví dụ:
Nhap mot ky tu: A
Ky tu thuong: a

4. Nhập từ bàn phím 2 số nguyên, tính tổng, hiệu, tích, thương của 2 số.

Ví dụ:
Nhap so thu nhat: 7
Nhap so thu hai: 4
Tong: 11
Hieu: 3
Tich: 28
Thuong: 1 du 3

5. Nhập vào 2 số nguyên, xuất ra phép so sánh giữa 2 số.

Ví dụ:
Nhap so thu nhat: 6
Nhap so thu hai: 9
So lon hon la: 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