組合語言 高階語言介面

- C++ 使用組合語言 Inline assembly
- GNU C 使用組合語言

Inline assembly

- 語法

```
__asm{
        組合語言
}
或是
__asm 組合語言
__asm 組合語言
```

Inline assembly

- 注意內部變數使用堆疊所以無法參考

```
static int k;     //有需要則靜態變數
main() {
    __asm{
          mov  k, eax
    }
```

C++ 使用組合語言 Inline assembly

- 進入 Visual studio 2005, 新增主控台應用程式

```
#include <iostream>
using namespace std;
static int k;
int _tmain( int argc, _TCHAR*argv[] ) {
    __asm{
            mov  eax, 12345h
            mov  k, eax
    }
    cout << hex << k << endl;
    system("pause");
}
```

C++ 使用組合語言

- 可以使用以下語法

```
#include <iostream>
using namespace std;
static int k;
int _tmain( int argc, _TCHAR*argv[] ) {
        __asm mov  eax, 12345h
        __asm mov  k, eax

    cout << hex << k << endl;
    system("pause");
}
```

GNU C 使用組合語言

- 用於 UNIX 系統與 Dev-C++
- 須使用 GNU as 語法

GNU as 語法

1) 暫存器前面加 %
   %eax %ebx %ecx %edx
2) 移動方向相反
3) 記憶體大小指定放在指令後面

```
mov  k, eax
```

事實上爲 mov DWORD PTR k, eax
所以改爲

```
movl   %eax, k
```

GNU as 語法

4) 數值前面加上 $

```
mov  eax, 123h
```

改爲

```
movl   $0x123, %eax
```

GNU as 語法

5) 位址前面加上 $

```
mov  OFFSET k, eax
```

改爲

```
movl   %eax, $k
```

GNU C Inline assembly

- 語法為 asm(" 組合語言 ");

```
asm("movl $0x12345,%eax");
asm("movl %eax,_k");
```

GNU C++ 使用組合語言 (Dev C++)

```
#include <iostream>
using namespace std;
static int k;
int main() {

    asm("movl $0x12345,%eax");
    asm("movl %eax,_k");

    cout << hex << k << endl;
    system("pause");
}
```

變數 k
Linux gcc 符號爲 k
Dev C++符號爲 _k