國立屏東大學 資訊工程學系 C語言程式設計

關於函式的傳回值

某次考試有位同學寫了以下的maxBall.c程式碼：

```
#include <stdio.h>

struct ball
{
    int  value;
    char label[10];
};

typedef struct ball Ball;

void showABall(Ball b)
{
    printf("%s(value=%d)\n", b.label, b.value);
}

Ball *maxBall(Ball *a, Ball *b)
{
    if(a->value>b->value)
    {
        a->value=200;
   //正確答案應寫做return a;
    }
    else
    {
        b->value=b->value;
   //正確答案應寫做return b;
    }
}

int main()
{
    Ball b1 ={ 200, "ball 1"};
    Ball b2 ={ 0, "ball 2" };
    Ball *max;
    scanf(" %d", &b2.value);
    max=maxBall(&b1, &b2);
    printf("max's value=%p\n", max);
    showABall(*max);
}
```

此程式會讓指標max指向b1與b2其value數值較大者，但這位同學在第20及第25行將原本應寫為□return a;□與「return b;□的程式碼，寫錯為□a→value=200;□與「b→value=b→value;□□有趣的是、神奇的是、不可思議的是…. 這個「錯誤」的程式的執行結果竟然是「正確」的！！請參考以下的執行結果：

```
[23:06 junwu@ws ~]$ cc test.c
[23:06 junwu@ws ~]$ ./a.out
20
ball 1(value=200)
[23:06 junwu@ws ~]$ ./a.out
300
ball 2(value=300)
[23:06 junwu@ws ~]$
```

其實這位同學所犯的錯誤是在maxBall()函式裡，少寫了return敘述，但卻「幸運地」寫出了「仍然會產生正確結果的不正確」程式。讓我們將上述程式中的maxBall()函式轉譯為組合語言，就可以看出原因了[1] – 由於在x86_64架構下，當函式的呼叫完成時會將rax暫存器的值傳回[2]（如果該函式有傳回值的話）：

```
maxBall:
        push    rbp
        mov     rbp, rsp
        mov     QWORD PTR [rbp-8], rdi   // 將第一個參數值(指標a的值)，放入stack
        mov     QWORD PTR [rbp-16], rsi  // 將第二個參數值(指標b的值)，也放
入stack
        mov     rax, QWORD PTR [rbp-8]   // 把第一個參數值(指標a的值)，放入rax暫存
器
        mov     edx, DWORD PTR [rax]     // 把rax所指向的地方的值(Ball結構體中
的value)放入edx暫存器
              // 也就是a->value
        mov     rax, QWORD PTR [rbp-16]  // 把第二個參數值(指標b的值)，放入rax暫存
器
        mov     eax, DWORD PTR [rax]     // 把rax所指向的地方的值(Ball結構體中
的value)放入eax暫存器
              // 也就是b->value
        cmp     edx, eax                 // 比較a->value與b->value
        jle     .L2                      // if (a->value <=b->value> goto L2
        mov     rax, QWORD PTR [rbp-8]   // ==> else 將第一個參數值(指標a的值),
放入rax暫存器
        mov     DWORD PTR [rax], 200                 將rax所指向的地方裡的值設定
為200
        jmp     .L4                      // 強制跳躍到L4
.L2:
        mov     rax, QWORD PTR [rbp-16]  // 將第二個參數值(指標b的值)，放入rax暫存
器
        mov     edx, DWORD PTR [rax]     // 將rax所指向的地方裡的值放入到edx暫存
器
        mov     rax, QWORD PTR [rbp-16]  // 將第二個參數值(指標b的值)，放入rax暫存
器
        mov     DWORD PTR [rax], edx     // 將edx暫存器的值放入到rax暫存器所指向
的地方暫存器
```

```
.L4:
        pop     rbp
        ret
```

[1)]
你可以使用https://godbolt.org/來進行轉譯。
[2)]
可參考Wikipedia https://wiki.cdot.senecacollege.ca/wiki/X86_64_Register_and_Instruction_Quick_Start