//Yujian Li (yl7kd) 03/29/16 Section 102

InLab08

In this lab we look into a different aspect of computer science: The Assembly Code. During the inlab process, we use the assembly code generated by a test C++ program in order to understand how data was passed in the system.

Because there are different types of data, storing in the assembly code will treat each code. When integer was passed by value, it was directly used “mov” command to store value into the designated register “exa” and then call the function. While it was passed by reference, it used “lea” command and also store the actual address into the “exa” register. In the callee section there is also an extra line of “mov eax, DWORD PTR [eax]” comparing to pass by value. When char type was passed by value and reference the assembly code does the similar operation except it uses “BYTE PTR” instead of “DWORD PTR” when storing the values due to the size of the char is less than int. Since passing pointers is already a reference of certain value therefore there is no case of passing by reference for pointers. When passing by value, “mov eax, DWORD PTR [esp+28] , mov DWORD PTR [eax], mov eax, DWORD PTR [esp+28], mov DWORD PTR [esp], eax” were the operations it performed, though it did not use the “lea” operation, it did do the similar operation as the process of passing by reference for int or char. Float operation is also similar with int, especially within the callee section. In the main section, it used “mov eax, DWORD PTR .LC0” to deal with floating numbers. Because there are multiple elements within the object, the elements are stored orderly in the main section with 4 byte away as the pointer address has shown. Because of the stack structure nauture, when storing into the stack, the callee will move the last elements into the stack first. For array in assembly code. It works very similar to the object. As the main code has shown, it was passed by reference and inside the callee, the callee will first access its base address then try to access each elements inside array by the operation “add 4”. As we have studied before, the process of passing by reference is very similar to passing by pointers, and also shown in the code int (reference and pointer).

Reference Code:

Int (by value)

Main:

```
push    ebp
mov     ebp, esp
and     esp, -16
sub     esp, 32
mov     DWORD PTR [esp+28], 1
mov     eax, DWORD PTR [esp+28]
mov     DWORD PTR [esp], eax
call    _Z4testi
mov     eax, 0
```

Callee:

```
push    ebp
mov     ebp, esp
sub     esp, 24
mov     eax, DWORD PTR [ebp+8]
```

```
mov    DWORD PTR [esp+4], eax
mov    DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
```

Int (by reference):

main:

```
push   ebp
mov    ebp, esp
and    esp, -16
sub    esp, 32
mov    DWORD PTR [esp+28], 1
lea    eax, [esp+28]
mov    DWORD PTR [esp], eax
call   _Z4testRi
mov    eax, 0
```

callee:

```
push   ebp
mov    ebp, esp
sub    esp, 24
mov    eax, DWORD PTR [ebp+8]
mov    eax, DWORD PTR [eax]
mov    DWORD PTR [esp+4], eax
mov    DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
```

Char:

By value:

main:

```
push   ebp
mov    ebp, esp
and    esp, -16
sub    esp, 32
mov    BYTE PTR [esp+31], 97
movsx  eax, BYTE PTR [esp+31]
mov    DWORD PTR [esp], eax
call   _Z4testc
mov    eax, 0
```

callee:

```
push   ebp
mov    ebp, esp
sub    esp, 24
mov    eax, DWORD PTR [ebp+8]
mov    BYTE PTR [ebp-12], al
movsx  eax, BYTE PTR [ebp-12]
mov    DWORD PTR [esp+4], eax
mov    DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
```

By reference:

callee:

```
push    ebp
mov     ebp, esp
sub     esp, 24
mov     eax, DWORD PTR [ebp+8]
movzx   eax, BYTE PTR [eax]
movsx   eax, al
mov     DWORD PTR [esp+4], eax
mov     DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
```

main:

```
push    ebp
mov     ebp, esp
and     esp, -16
sub     esp, 32
mov     BYTE PTR [esp+31], 97
lea     eax, [esp+31]
mov     DWORD PTR [esp], eax
call    _Z4testRc
mov     eax, 0
```

Pointer:

by value:

main:

```
push    ebp
mov     ebp, esp
and     esp, -16
sub     esp, 32
mov     eax, DWORD PTR [esp+28]
mov     DWORD PTR [eax], 1
mov     eax, DWORD PTR [esp+28]
mov     DWORD PTR [esp], eax
call    _Z4testPi
mov     eax, 0
```

callee:

```
push    ebp
mov     ebp, esp
sub     esp, 24
mov     eax, DWORD PTR [ebp+8]
mov     DWORD PTR [esp+4], eax
mov     DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
```

Float:

By Value:

callee:

```
push    ebp
mov     ebp, esp
```

```
        sub     esp, 24
        mov     eax, DWORD PTR [ebp+8]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
```

main:

```
        push    ebp
        mov     ebp, esp
        and     esp, -16
        sub     esp, 32
        mov     eax, DWORD PTR .LC0
        mov     DWORD PTR [esp+28], eax
        mov     eax, DWORD PTR [esp+28]
        mov     DWORD PTR [esp], eax
        call    _Z4testf
        mov     eax, 0
```

By reference:

callee:

```
        push    ebp
        mov     ebp, esp
        sub     esp, 24
        mov     eax, DWORD PTR [ebp+8]
        mov     eax, DWORD PTR [eax]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
```

main:

```
        push    ebp
        mov     ebp, esp
        and     esp, -16
        sub     esp, 32
        mov     eax, DWORD PTR .LC0
        mov     DWORD PTR [esp+28], eax
        lea     eax, [esp+28]
        mov     DWORD PTR [esp], eax
        call    _Z4testRf
        mov     eax, 0
```

Object:

By value:

main:

```
        push    ebp
        mov     ebp, esp
        and     esp, -16
        sub     esp, 32
        mov     DWORD PTR [esp+24], 1
        mov     DWORD PTR [esp+28], 2
        mov     eax, DWORD PTR [esp+24]
```

```
        mov     edx, DWORD PTR [esp+28]
        mov     DWORD PTR [esp], eax
        mov     DWORD PTR [esp+4], edx
        call    _Z4test1a
        mov     eax, 0
```

callee:

```
        push    ebp
        mov     ebp, esp
        push    ebx
        sub     esp, 20
        mov     ebx, DWORD PTR [ebp+12]
        mov     eax, DWORD PTR [ebp+8]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
```

By reference:

callee:

```
        push    ebp
        mov     ebp, esp
        push    ebx
        sub     esp, 20
        mov     eax, DWORD PTR [ebp+8]
        mov     ebx, DWORD PTR [eax+4]
        mov     eax, DWORD PTR [ebp+8]
        mov     eax, DWORD PTR [eax]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:_ZSt4cout
```

main:

```
        push    ebp
        mov     ebp, esp
        and     esp, -16
        sub     esp, 32
        mov     DWORD PTR [esp+24], 1
        mov     DWORD PTR [esp+28], 2
        lea     eax, [esp+24]
        mov     DWORD PTR [esp], eax
        call    _Z4testR1a
        mov     eax, 0
```

Array:

by value:

callee:

```
        push    ebp
        mov     ebp, esp
        push    ebx
        sub     esp, 20
```

```
        mov     eax, DWORD PTR [ebp+8]
        add     eax, 4
        mov     ebx, DWORD PTR [eax]
        mov     eax, DWORD PTR [ebp+8]
        mov     eax, DWORD PTR [eax]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:_ZSt4cout

main:
        push    ebp
        mov     ebp, esp
        and     esp, -16
        sub     esp, 32
        mov     DWORD PTR [esp+28], 1
        lea     eax, [esp+28]
        mov     DWORD PTR [esp], eax
        call    _Z4testPi
        mov     eax, 0
```