Gabriel Groover (gtg3vv)

Inlab8.pdf

Passing Parameters

INT – **When passing by value, the caller uses the following commands.**

Mov DWORD PTR [ebp-4], 1

Mov eax, DWORD PTR [ebp-4]

Mov DWORD PTR [esp], eax

Call intpvi

This loads the value of the variable x into the register. The callee moves the value to eax and returns it.

Mov ebp, esp

Mov eax, DWORD PTR [ebp+8]

Ret

Passing by reference is almost the same for the caller. The only exception is moving the address into eax.

Lea eax, [ebp – 8]

Mov DWORD PTR [ebp-4], eax

Mov eax, DWORD PTR [esp], eax

The callee requires an extra mov to account for the variable now being an address. It must get the value at the address in eax.

Mov eax, DWORD PTR [ebp+8]

Mov eax, DWORD PTR [eax]

Char – Char will behave identically to ints for passing by reference and value. The only difference being that chars only require a single byte of space.

The caller uses the following to load the single byte acii value into the register:

Mov BYTE PTR [ebp-1], 99

Movsx eax, BYTE PTR [ebp-1]

Mov DWORD PTR [esp], eax

The callee is as expected with BYTE values.

```
Mov eax, DWORD PTR [ebp+8]

Mov BYTE PTR [ebp-4], al

Movzx eax, BYTE PTR [ebp-4]
```

Passing a pointer by value will behave identically to passing an int or char by reference. These are essentially the same thing as you are just passing an address.

Float – The storage of a float by value is the same as an int. The caller uses the following to load the value of the float into eax, and then esp:

```
Mov eax, DWORD PTR .LC1

Mov DWORD PTR [ebp-4] ,eax

Mov eax, DWORD PTR [ebp-4]

Mov DWORD PTR [esp], eax
```

The callee for float by value uses two extra lines of code from int to allow it to put the float value on the stack:

```
Mov DWORD PTR [ebp-4], eax

Fld DWORD PTD [ebp-4]
```

The caller for float by reference is exactly the same as for int by reference. The callee behaves exactly the same as int by reference (getting the value at the address in eax) with the added fld command to load the float.

Objects seem to behave the same as whatever values it contains. The lea command is used to handle pointers, but otherwise passing the object is the same as passing each of its values.

Arrays – The callee function for an array is the same as the one for whatever the array holds. The Caller however, stores each of the array values relative to ebp. For an array of 4 integers:

```
Mov DWORD PTR [ebp-16], 1

Mov DWORD PTR [ebp-12] , 2

Mov DWORD PTR [ebp-8],3

Mov DWORD PTR [ebp-4],4

Lea eax, [ebp-16]
```

This is exactly as we know arrays to behave. Each item is placed at an address relative to the starting point. (ebp – x) The base address is then placed in eax so that any element can be accessed.