William Montgomery
Jeffrey Grandt
CS 366 - Homework 11
Spring 2013

Take the C code attached below and compile the code for IA-32 and x86-64.
Then compare and contrast the code. Specifically consider and discuss in your writeup:

- **Are there differences in the way the stack is handled?**

Yes. Especially in the fact that all functions that take under 6 parameters only have arguments in registers, rather than on the stack. Now the stack mostly consists of local variables and not much else.

In addition, both %rsp and %rbp both point to the same value in x86-64.

Of course, register values prefixed with an ‘r’ correspond to 64 bit registers in x86-64, rather than the 32 bit values in IA32.

The x86-64 code uses less of the stack than the IA-32 code, as more temporary values can be stored in the registers.

Also, many of the registers have different naming conventions... e.g. the base/frame pointer (from IA32) was %ebp and is now %rbp in x86-64 (same goes with %esp and %rsp).

- **Are there differences in the way functions calls are handled?**

In the IA-32 code, the parameters must be pushed on the stack before the call. In the x86-64 code, the parameters are given in the registers. In the IA-32 code, the stack pointer must be incremented in code, while in the x86-64 code, the stack is managed by hardware. So, in the IA-32 code, the pointer to the array is passed to the _bubble function at %ebp + 8 and the number of elements is at %ebp + 12. In the x86-64 code, the pointer to the array is in %rdi (it needs all 64 bits as it is a pointer) and the number of elements is in %esi (the lowest 32 bits of %rsi as it is an int).

- **Is the same amount of memory required for the code? (In particular, how many bytes are required for the instructions for each of the implementations?)**

No. The x86-64 version is 50 bytes larger.

Submit your report along with both assembly files (IA-32 and x86-64 versions) generated by gcc.