Mark Yang

HW 0: Review of Systems Programming and Architecture Concepts/Tools

Problem 1

When the caller (alice) calls the callee (bob), the caller function pushes the three arguments in order from right to left onto the stack. After the arguments are pushed onto the stack, the caller calls the callee function. The EIP is pointing to the next instruction within the caller which means that the return address points to the top of the stack right before the call to the callee. The caller will also push the EAX, ECX, and EDX register contents onto the stack.

The callee now sets up its own stack frame and pushes EBP onto the stack and move contents of ESP onto EBP. This will allow the arguments passed by the caller to be referenced as an offset from EBP. Then, the two local variables (of type int) within the callee will be allocated space on the callee's stack frame with respect to the ESP by subtracting 4 bytes for each variable. Then the local variables can be referenced as an offset from EBP. ESP will now point to the end of the stack while EBP will point to the top/start of the stack (of the callee). If there is additional temporary storage required, there will be allocated space for that in the stack and ESP will just point to wherever the end of the stack would be.

While the EBP pointer stays at the top of the stack, the instructions will be executed, shifting the ESP around until the point where the function is complete and ready to return. The callee will store the return value in the EAX register (because return type is of type integer) and then the function will restore all the EBX, ESI, and EDI registers. The caller will then save the return value from the EAX register into wherever it needs to be and then will also pop EAX, ECX, and EDX registers which then the top of the stack will be at the same position as it was before the call to the callee.

Problem 2

```
        .file   “callbob.c”
        .text
        .globl  main
        .type   main, @function
main:
        pushl   %ebp                First you push the EBP onto the stack
        movl    %esp, %ebp          Then you transfer the contents of ESP to EBP which allows arguments of
                                    main to be referenced as an offset from EBP
        subl    $12, %esp           Next it reserves 12 bytes of memory on the stack
        movl    $5, 4(%esp)         Now it stores the second parameter (5) into 4 bytes of memory on the
                                    stack
        movl    $2, (%esp)          Next it stores the first parameter (2) into the stack
        call    bob                 This calls the function bob after it finished pushing all the parameters
                                    onto the stack (jump to bob)

            Return here

        movl    %eax, -4(%ebp)      Now you store the value at the EAX register into 4 bytes of memory on
                                    the stack
        leave                       Set the stack pointer back to the base frame address
        ret                         Return back to where caller first called the main function
        .size   main, .-main
        .globl  bob
        .type   bob, @function
bob:
        pushl   %ebp                Same thing as main, push the EBP onto the stack
        movl    %esp, %ebp          Transfer the contents of ESP to EBP
        subl    $4, %esp            Reserve 4 bytes of memory on the stack
        movl    $3, -4(%ebp)        Store the local variable’s (int z) value onto the stack (reserved from
                                    previous instruction)
        movl    12(%ebp), %eax      Get the value of y passed into bob and put it in EAX register
        addl    %eax, -4(%ebp)      Add the value from the EAX register to the value of local variable (int z)
        movl    -4(%ebp), %eax      Now you move the value to the EAX register
        leave                       Set the stack pointer back to the base frame address
        ret                         Return back to where caller first called bob (go to “Return here”)
        .size   bob, .-bob
        .ident  "GCC: (Gentoo 4.8.4 p1.5, pie-0.6.1) 4.8.4"
        .section        .note.GNU-stack,"",@progbits
```