power:

pushl %ebp (1)

movl %esp, %ebp (2)

movl 12(%ebp), %ebx (3)

movl 8(%ebp), %edx (4)

movl $1, %eax (5)

L1:

test %ebx, %ebx (6)

je done (7)

imull %edx, %eax (8)

decl %ebx (9)

jmp L1

done:

popl %ebp

ret

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The Power function: after the initial pushl (1) and movl (2) of variables into workable registers, register ebx is set to i(3) and edx to x(4) while register eax to 1(5). At L1: ebx is tested to check if equal to 0(6), and if it is jump to done: (7). Otherwise multiply edx with eax(8) initially and decrement ebx until 0 while looping through L1: (9).

The Power Function: takes in 2 variables as arguments x and i, and sets a third variable to 1. L1: is a loop that continues until i is equal to 0, in which x is initially multiplied to the variable set to 1, and then within the loop multiplied by itself. After each multiplication, the variable i is decremented until the loop completes at 0.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fillarray:

pushl %ebp (1)

movl %esp, %ebp

xor %ecx, %ecx (2)

movl 8(%ebp), %edx (3)

movl 12(%ebp), %edi (4)

movl 16(%ebp), %ebx (5)

FL1:

cmpl %ebx, %ecx (6)

je F1End

push %ebx (7)

pushl %ecx

pushl %edx

call power (8)

popl %edx (9)

popl %ecx

popl %ebx

movl %eax, (%edi) (10)

incl %ecx (11)

add $4, %edi (12)

jmp FL1

F1End:

popl %ebp

ret

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The fillarray function: after the initial pushl and movl of variables into workable registers (1), the register ecx is set to 0 by using the operation xor (2). Moving the respective function arguments into registers, x to edx (3), a[] to edi (4), and n to ebx (5), the code proceeds to FL1:, where n, ebx, is compared to 0, ecx. Ecx being initially set to 0, the length of the array n in register edx, is compared to ecx in a loop while ecx is being incremented (11), and should it be equal we jump to F1End (6).

Next, the registers ecx and edx are pushed (7), while n in register ebx is saved before the power function is called (8) with registers ecx and edx, the value of x, as the arguments. Once the function call ends, those same registers are restored by the popl operation (9) before moving eax into an empty index of edi (10) which is an array. A new empty index is then set to edi (12) for the next value of x to the power to i is set into.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fillarray2:

pushl %ebp (1)

movl %esp, %ebp

xor %ecx, %ecx (2)

movl 8(%ebp), %edx (3)

movl 12(%ebp), %edi

movl 16(%ebp), %ebx

movl $1, %eax

FL2:

cmpl %ebx, %ecx (4)

je F2End

movl %eax, (%edi) (5)

imull %edx, %eax (6)

incl %ecx (7)

add $4, %edi (8)

jmp FL2

F2End:

popl %ebp

ret

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The fillarray2 function: after the initial pushl (1) and movl of variables into workable registers, ecx is set to 0 (2). The registers edx, edi, ebx and eax are set to x, a[], n and 1 respectively (3). The index of the array is compared to the length of the array, which is initially set to 0, in a loop until equal, in which case the loop jumps to F2End: (4).

Register eax, which has the value of 1 initially, is then moved into an index of the array, edi(5), before being multiplied by x, edx (6). This signifies x to the power of the current index. Ecx, which is initially 0, is then incremented (7) while the next index of the array, edi (8), is retrieved before FL2 loops.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

compare:

pushl %ebp (1)

movl %esp, %ebp

xor %eax, %eax (2)

movl 8(%ebp), %edx (3)

movl 12(%ebp), %ebx

movl 16(%ebp), %ecx

CLoop:

test %ecx, %ecx (4)

je CLT

movl (%ebx), %esi (5)

cmpl (%edx), %esi (6)

jne CEnd

addl $4, %edx (7)

addl $4, %ebx (8)

decl %ecx (9)

jmp CLoop

CLT:

movl $1, %eax (10)

CEnd:

popl %ebp

ret

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The compare function: after the initial pushl (1) and movl of variables into workable registers, eax is set to 0 by the xor operation (2). The registers edx is set to point to the first value of array1, ebx is set to point to the first value of array2, and ecx is set to n (3).

In CLoop, ecx is tested if equal to 0(4) as the loop condition which ends at CLT. Register ebx, the current value of array2, is stored in register esi (5) and then compared to edx, the first value of array1 (6). Should they not be equal, the loop terminates and jumps to CEnd. Otherwise, the next values of edx and ebx are set by incrementing the position of the pointer by 4, which is the bit value of an int (7&8). Register ecx is then decremented and the loop restarts (9). In CLT: eax is set to 1 should ecx be equal to 0(10).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_