Answers to Questions 1-5:

ASSEMBLY CODE GENERATED ON A WINDOWS MACHINE

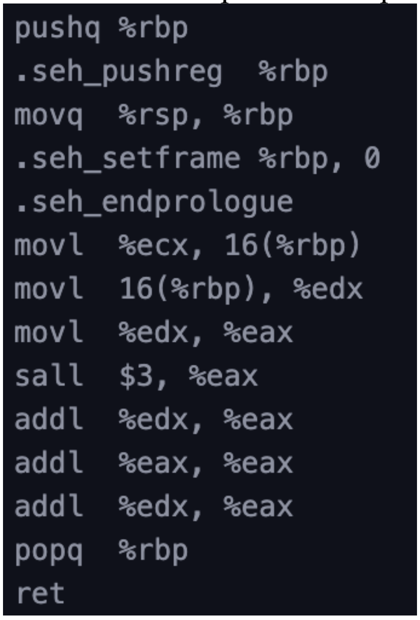
(Kenjie)

1. Yes, *x multiply by 19* can be implemented by using shifts and adds only. Notice how x * 19 is equivalent to (x * 16) + (x * 2) + x. Subsequently, multiplying by 16 is equivalent to shifting to the left by 4 and multiplying x by 2 is equivalent to shifting to the left by 1. From this, the code implementation can be as follows:

```
int dummy(int x){
  int ret = (x << 4) + (x << 1) + x;
  return ret;
}
```

(Czy)

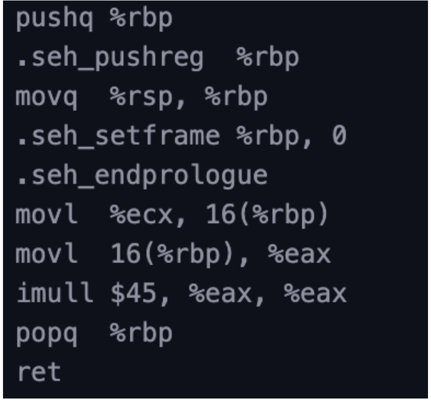
2. For the case of x * 19, x is shifted to the left by three (equivalent to x * 8). Subsequently, the output from that is added to itself—(x * 8) + (x * 8)—equivalent to x * 18. Lastly, x is added to the previous output, so now we have x * 19.



```
pushq %rbp
.seh_pushreg  %rbp
movq  %rsp, %rbp
.seh_setframe %rbp, 0
.seh_endprologue
movl  %ecx, 16(%rbp)
movl  16(%rbp), %edx
movl  %edx, %eax
sall  $3, %eax
addl  %edx, %eax
addl  %eax, %eax
addl  %edx, %eax
popq  %rbp
ret
```

(Gab)

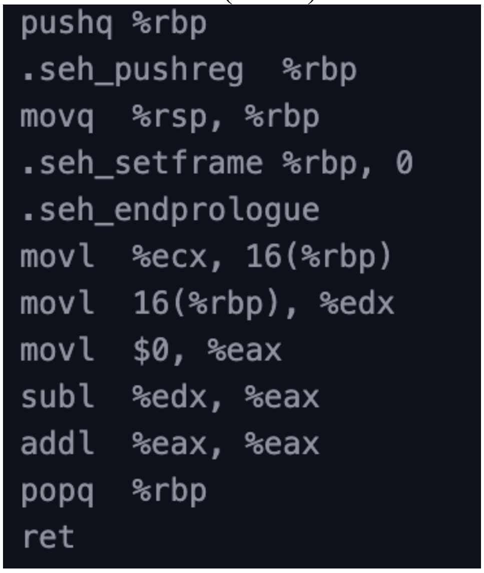
3. For the case of x * 45, a single multiplication is performed (x * 45).



```
pushq %rbp
.seh_pushreg  %rbp
movq  %rsp, %rbp
.seh_setframe %rbp, 0
.seh_endprologue
movl  %ecx, 16(%rbp)
movl  16(%rbp), %eax
imull $45, %eax, %eax
popq  %rbp
ret
```

(Kenjie)

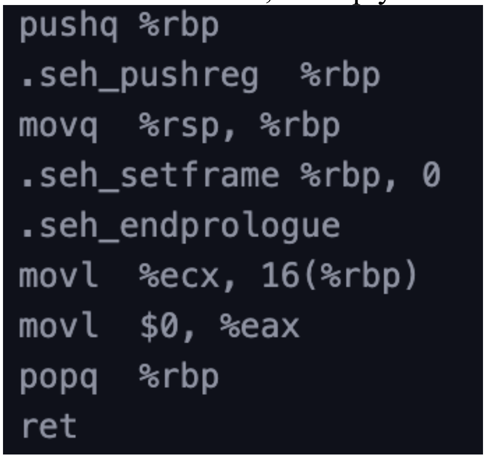
4. For the case of x * (-2), it negates the x by subtracting it from 0 (x → -x). Afterward, it is added to itself (-x * 2).



```
pushq %rbp
.seh_pushreg  %rbp
movq  %rsp, %rbp
.seh_setframe %rbp, 0
.seh_endprologue
movl  %ecx, 16(%rbp)
movl  16(%rbp), %edx
movl  $0, %eax
subl  %edx, %eax
addl  %eax, %eax
popq  %rbp
ret
```

(Czy)

5. For the case of x * 0, it simply returns 0.



```
pushq %rbp
.seh_pushreg  %rbp
movq  %rsp, %rbp
.seh_setframe %rbp, 0
.seh_endprologue
movl  %ecx, 16(%rbp)
movl  $0, %eax
popq  %rbp
ret
```