```
; Written by Jared Dyreson
; C-Code
; if(quitReturnCode != 0){ quitFileDescriptor = quitReturnCode; }
; else { quitFileDescriptor = 0; }

%define RAISE_SYSTEM_EXIT 60
%define EXIT_STATUS 0

section .text
global _start

file_assignment:
mov r11, r12
jmp continuity

zero_assignment:
xor r12, r12 ; clear/set quitFileDescriptor equal to zero
jmp continuity

_start:
xor r11,r11 ; quitReturnCode
xor r12,r12 ; quitFileDescriptor
mov r11, 10 ; some arbitrary number for quitReturnCode which will satisfy the if block
cmp r11, 0 ; check if the value is set to zero
jne file_assignment
je zero_assignment
continuity:
mov rax, RAISE_SYSTEM_EXIT ; we want to exit
mov rdi, EXIT_STATUS ; return 0
syscall ; invoke
```

```
; Written by Jared Dyreson
; C-Code
; while(quitReturnCode != 0) { quitFileDescriptor = quitReturnCode; }

%define RAISE_SYSTEM_EXIT 60
%define EXIT_STATUS 0

global _start
section .text
_start:
mov r11, 10 ;  this is not to result in an infinite loop (quitReturnCode)
xor r12, r12 ; quitReturnCode
while_loop_entry:
cmp r11, 0 ; if quitReturnCode is zero, then break
mov r12, r11 ; assignment
dec r11 ; decrease counter
jne while_loop_entry ; then proceed to leave (NOT do while where the condition is checked befo
re proceeding)
je break
break:
mov rax, RAISE_SYSTEM_EXIT
mov rdi, EXIT_STATUS
syscall
```

```
; Written by Jared Dyreson
; C-Code
; do { quitTotalCost = quitTotalCost + quitPrice; } while (quitTotalCost < 1000)

global _start
section .text
_start:
xor r11, r11 ; quitTotalCost
mov r12, 999 ; quitPrice
; this loop is intended to only twice for speed
do_while_block:
add r11, r12 ; add <dst> <src>
cmp r11, 1000 ; our conditional statement
jl do_while_block ; continue doing the statement if the number is less than 1000
jnl break ; break if the opposite is true
break:
mov rax, 60
mov rdi, r12 ; just to see what the ouput would have been <./ouput ; echo $?> turns out i can’
t display 1000 but its there
syscall
```

```
; Written by Jared Dyreson
; C-Code
; for(int i = 0; i < 10; ++i){ if(i%2 == 0) { quitSum+=i; } }
; output should be 25; the program increments every other time

global _start
section .text

assign:
add r13, r11 ; quitSum+=i
inc r11 ; ++i
jmp for_loop_block ; return to for loop

_start:
xor r11,r11 ; this is is our indexer (int i = 0 )
mov r12, 10 ; our terminating condition
xor r13, r13 ; quitSum

; for loop block begins
for_loop_block:
cmp r11, r12 ; i < 10
jl continuity ; execute for loop body
jnl break ; leave if we are i == 10

continuity:
xor rdx, rdx ; remainder
mov rbx, 2 ; divisor
mov rax, r11 ; dividend
div rbx
cmp rdx, 0 ;if i %2 == 0 (no remainder)
je assign ; ^ jump to adding instruction
; else increment counter and return to head of for loop
inc r11
jmp for_loop_block

break:
mov rax, 60
mov rdi, r13
syscall
```