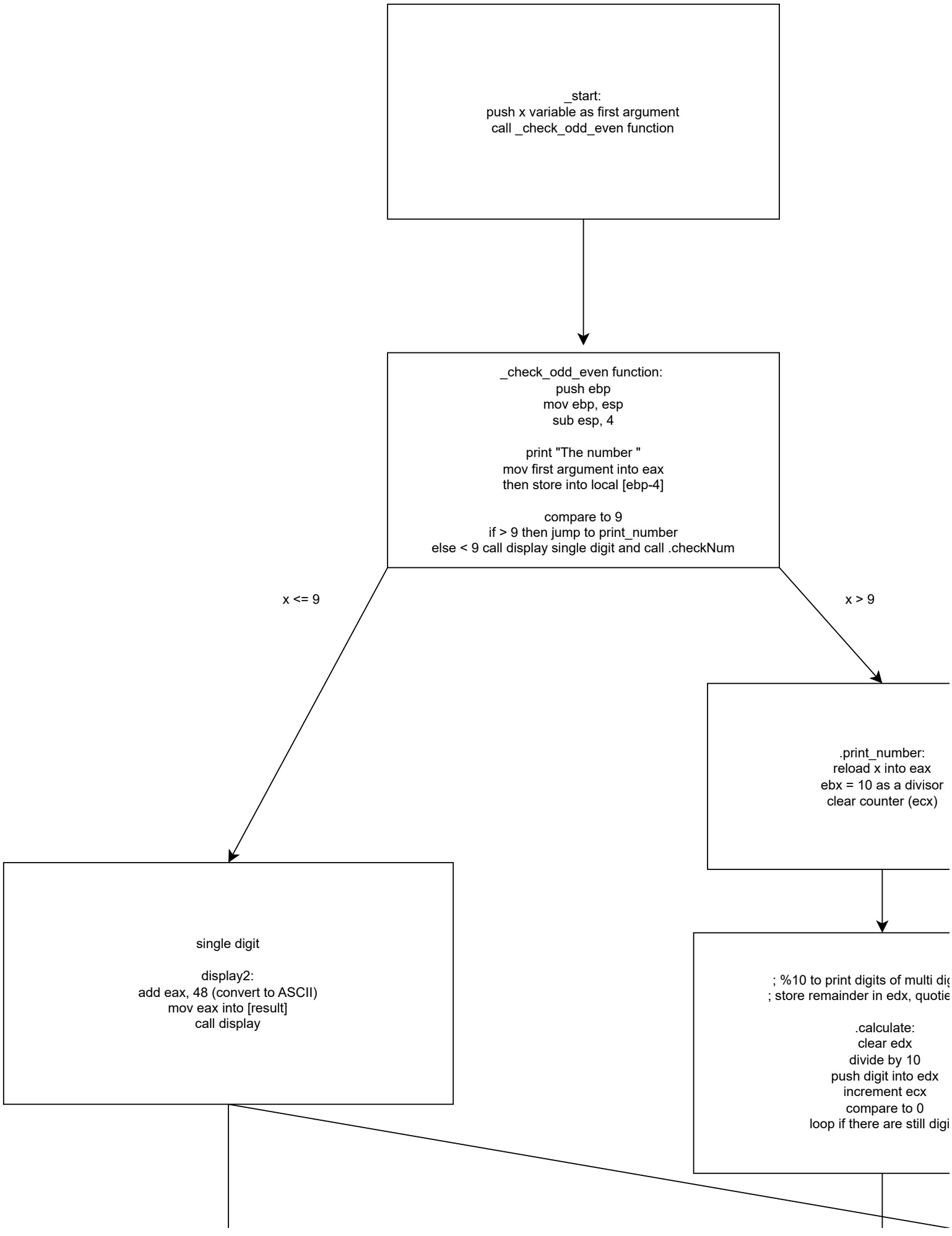
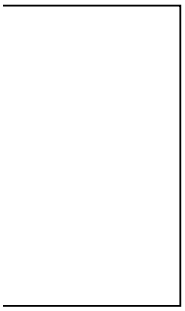
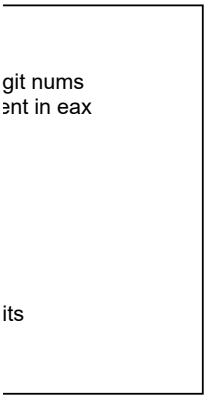
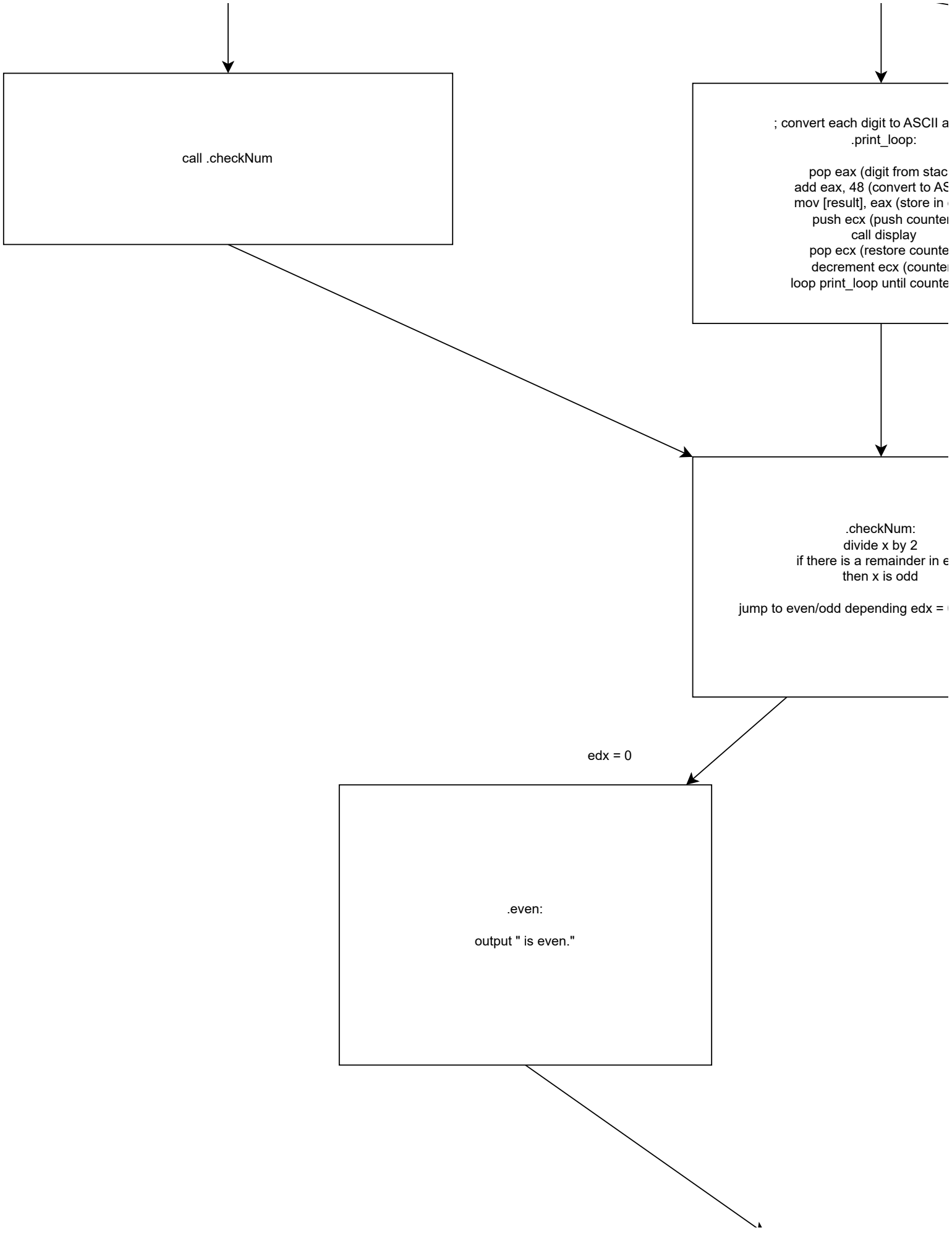
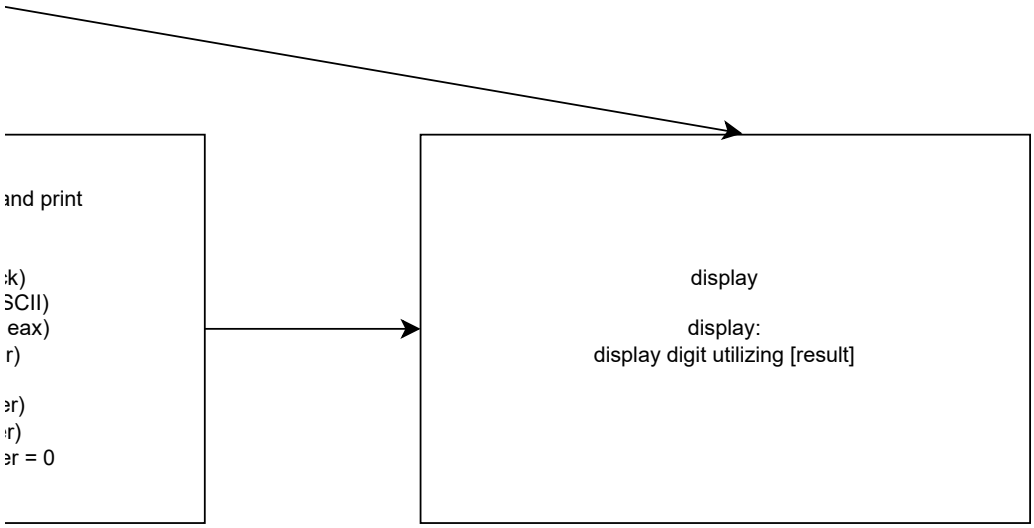
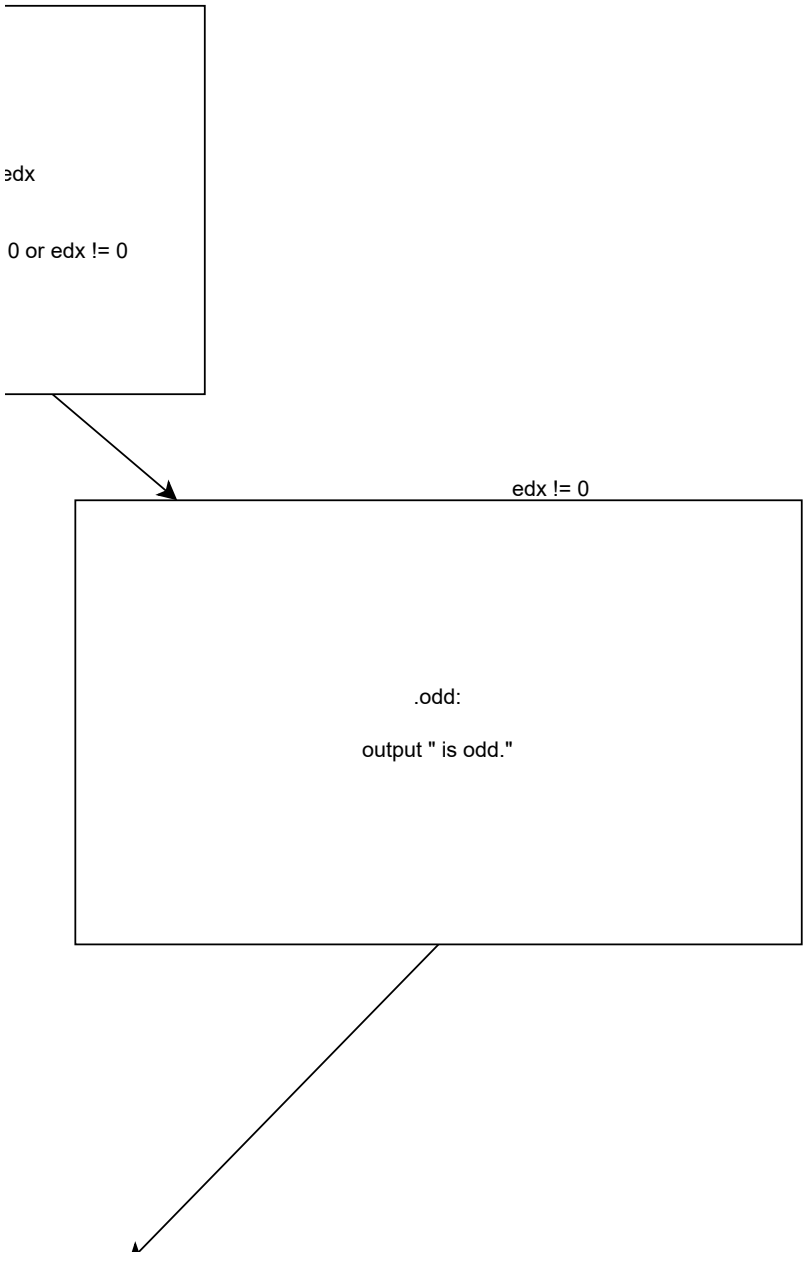

_start:
push x variable as first argument
call _check_odd_even function
_check_odd_even function:
push ebp
mov ebp, esp
sub esp, 4
print "The number "
mov first argument into eax
then store into local [ebp-4]
compare to 9
if > 9 then jump to print_number
else < 9 call display single digit and call .checkNum
x <= 9
x > 9
single digit
display2:
add eax, 48 (convert to ASCII)
mov eax into [result]
call display
.print_number:
reload x into eax
ebx = 10 as a divisor
clear counter (ecx)
; %10 to print digits of multi di
; store remainder in edx, quotie
.calculate:
clear edx
divide by 10
push digit into edx
increment ecx
compare to 0
loop if there are still digi

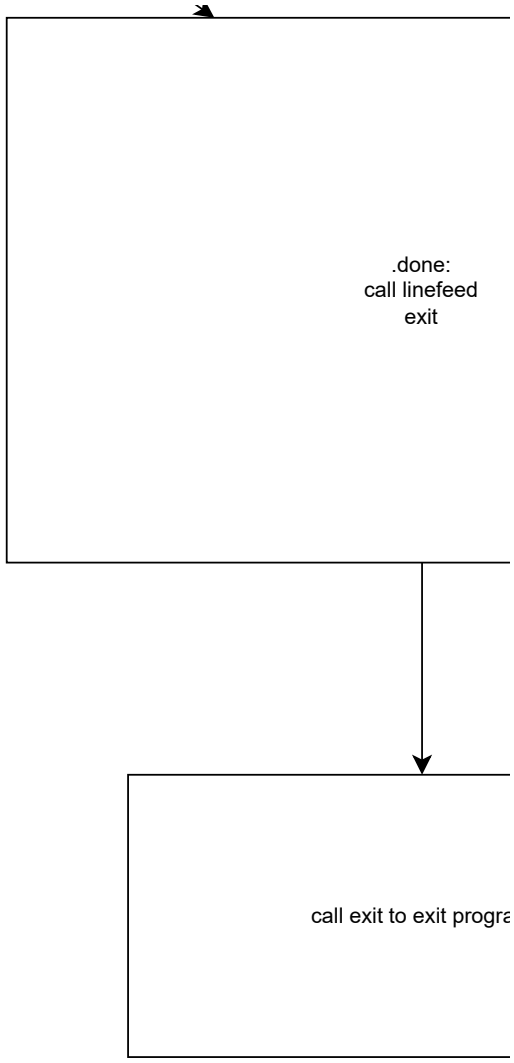


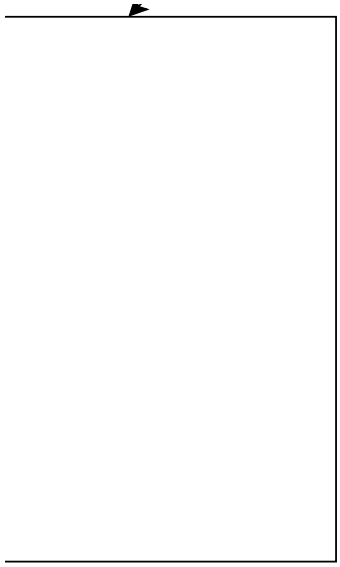

git nums
ent in eax
its

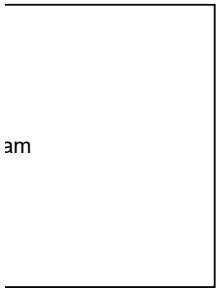

call .checkNum
; convert each digit to ASCII a
.print_loop:
pop eax (digit from stac
add eax, 48 (convert to AS
mov [result], eax (store in
push ecx (push counte
call display
pop ecx (restore counte
decrement ecx (counte
loop print_loop until counte
.checkNum:
divide x by 2
if there is a remainder in e
then x is odd
jump to even/odd depending edx =
edx = 0
.even:
output " is even."


and print
k)
SCII)
eax)
r)
er)
r)
er = 0
display
display:
display digit utilizing [result]


edx
0 or edx != 0
edx != 0
.odd:
output " is odd."


.done:
call linefeed
exit
call exit to exit progra




am