.data

x: .space 4

//x: .long 5

doi: .long 2

trei: .long 3

nr: .long 0

formatScanf: .asciz "%d"

formatPrintf: .asciz "%d\n"

.text

.global main

f:

pushl %ebp

movl %esp, %ebp

pushl %ebx

movl 8(%ebp), %ebx

//%ebx=x

cmp $1, %ebx

je ret0

movl %ebx,%eax

xorl %edx, %edx

divl doi

cmp $0, %edx

je par

jmp impar

ret0:

popl %ebx

popl %ebp

ret

par:

addl $1, nr

//x=x/2

movl %ebx,%eax

xorl %edx, %edx

divl doi

pushl %eax

call f

popl %ebx

popl %ebx

popl %ebp

ret

impar:

addl $1, nr

//x=3x+1

movl %ebx,%eax

mull trei

incl %eax

pushl %eax

call f

popl %ebx

popl %ebx

popl %ebp

ret

main:

//citire x

pushl $x

pushl $formatScanf

call scanf

popl %ebx

popl %ebx

//apelare functie

pushl x

call f

popl %ebx

afisare\_rezultat:

pushl nr

pushl $formatPrintf

call printf

popl %ebx

popl %ebx

exit:

movl $1, %eax

xorl %ebx, %ebx

int $0x80

Eroarea segmentation fault se obtine atunci cand incerci sa accesezi o zona de memories si nu iti este permis acest lucru. Deoarece se face movl $100, %esp se distruge stack pointerul, care ar trebui sa retina adresa varfului stivei, apoi se face un pushl %eax, care da practice eroarea.

%edi este un registru calle saved, deci trebuie pus pe stiva nainte sa fie modificat in functia f, apoi scos de pe stiva inainte de instructiunea ret.

%ecx este un registru caller saved, deci nu ne este garantata restaurarea lui dupa functii. II salvam valoarea pe stiva inainte de functia f, apoi o restauram, la fel si in cazul functiei printf.

%edi retine adresa unui array de tip long, astfel trebuie sa inlocuim incl %edi cu addl $4,%edi, deoarece tipul long este pe 4 bytes.

Partea 0x02

Codul nu contine o bucla infinita, deoarece daca la adresa memorata in %ecx se incrementeaza repetat, se va intampla un stack overflow iar valoarea din ecx va ajunge 0, moment in care programul va ajunge cu executarea la et\_exit.

Procedura primeste 2 argumente (liniile 5,9,19,20)

Conditia de salt de la linia 8 este ca adresa din %eax sa fie diferita de 0. (linia 7)

Niciunul dintre argumente nu este un pointer.

Nu, deoarece movzbl efectueaza un mov de la byte la long (linia 22)

Tipul de date al argumentelor este byte. (linia 22)

Pentru un input x=0 nu s-ar respecta prima ramura (de iesire din recursivitate). 0 fiind numar par, va se va executa f(0/2) , adica f(0). Practic s-a format o bucla infinita, astfel se vor creea nivele de recursivitate pana cand memoria alocata se va termina, de aceea output-ul este eroarea segmentation fault.