Test Cursul 2

1. Presupunand ca src este o adresa kernel valid iar dst este setat la NULL, care din urmtoarele apeluri vor genera un kernel oops: `memcpy(dst, src, 1)`, `copy_to_user(dst, src, 1)`, `copy_from_user(dst, src, 1)`.
   - Apelul `memcpy` va genera un oops, pentru că se încearcă scrierea la o adresă nevalidă. Argumentul "memcpy nu face un apel de sistem" nu ţine ? suntem deja în kernel space. Chiar dacă kernel-ul nu este link-editat cu biblioteca standard C, are propria implementare de `memcpy`; puteţi consulta sursele pentru definiţia generică sau cea specifică arhitecturii x86.
   - Apelul `copy_to_user` nu va genera un oops, explicaţia se găseşte în curs.
   - Apelul `copy_from_user` va genera un oops, pentru că doar pointerul din userspace (sursa, în acest caz) este verificat (sursa aici). Destinaţia este un pointer din kernel şi, dacă nu este validă, va provoca un oops.
2. De ce nu este recomandata folosirea de apeluri recursive in kernel?
   - Apelurile recursive creează multe cadre de stivă şi stiva are dimensiune fixă (şi mică) în kernel.
3. Care sunt dejavantajele pasarii parametrilor apelurilor de sistem prin registri (si nu pe stiva)? (de exemplu pe arhitectura x86)
   - Numărul mic de registre, dimensiunea fixă, nu se pot transmite structuri sau vectori sunt răspunsuri acceptate.
   - Accesul la registre **nu** este mai lent decât la stivă.
   - Nu riscăm să pierdem valorile registrelor, ele sunt salvate la context switch.