Intel x86 Linux ASM - CaseInvert

Kiértékelés: Egy Intel 32 bit Linux assembly-ben írt **feladat.S** fájlt kell feltölteni. Ezt az assembly fájlt a biró fordítja, szimbólumokat ellenőriz és végül linkeli. Ezek után futtatja és ellenőrzi, hogy **a kimenet megegyezik-e karakterre pontosan az elvárt eredménnyel**.

A biró által végrehajtott fordítási és linkelési parancsok:

- Fordítás (c keret): `gcc -m32 -c -static method_test.c -o method_test.o`
- Fordítás: `gcc -m32 -c -static <feladat>.S -o feladat_s.o`
- Linkelés: `gcc -m32 -static method_test.o feladat_s.o -o program`

Feladat leírás (+2 pont)

Írjunk egy `caseInvert` nevű eljárást assembly-ben amely a kapott bemeneti stringet a nagybetűket kisbetűre és a kisbetűket nagybetűre alakítja. Az eredményt pedig egy kimeneti karaktertömbbe készíti el. Bármilyen nem abc karaktert módosítás nélkül másoljunk a kimeneti tömbbe. Figyeljünk arra, hogy egy string mivel van lezárva illtve mivel kell lezárni!

A `minta.zip` tartalmaz egy egyszerű C teszt kódrész amivel lehet tesztelni. Az ebben található C fájlt nem kell feltölteni.

Elvárt függvény prototípus (C-ben)

```
void caseInvert(const char input[], char* output);
```

1. **input** a bemeneti string. Ezen nem szabad módosítani! Módosítás esetén a program hibás lehet.
2. **output** a kimeneti tömb ahova az átalakítás eredményét kell másolni.

A visszatérési értéke az eljárásnak nincs.

Példa

Ha a bemenet: `"hello Bello!"` akkor az eredmény: `"HELLO bELLO!"`.

Egyszerűsített használat

- Fordítás egy lépésben: `gcc -m32 -static method_test.c megoldas.S -o program`
- Futtatás sikeres fordítás után: `./program`

Tipp (azoknak akik még olvassák ezt)

Jelen esetben karaktereken kell dolgozni és nem integereken. Integerek esetén 4 bájtos adatokkal dolgoztunk, karakterek esetén ez már nem igaz. Emlékezzünk vissza a korábban tanultakra! (Vagy nézzük meg a gyakorlati jegyzetet)

Pszeudo implementáció:

```
void caseInvert(const char input[], char* output) {
    idx = 0;
    for (; input[idx] != 0; idx++) {
        ch = input[idx];

        if ('a' <= ch && ch <= 'z') {
            output[idx] = 'A' + (ch - 'a');
        } else if ('A' <= ch && ch <= 'Z') {
            output[idx] = 'a' + (ch - 'A');
        } else {
            output[idx] = ch;
        }
    }

    output[idx] = '\0';

}
```