Intel x86 Linux ASM - Eljáráshívás

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_invoke.c -o method_invoke.o`
- Fordítás (+1 pont): `gcc -m32 -c -static <feladat>.S -o feladat_s.o`
- Linkelés (+1 pont): `gcc -m32 -static method_invoke.o feladat_s.o -o program`

Egyéb pontok:

- (+1 pont) Az assembly fájlban megtalálható az elvárt címke és globális.

Feladat leírás (+5 pont)

Írjunk egy eljárást assembly-ben amely a kaptott bemeneti tömböt bejárja, megvizsgálja hogy egy kapott egész értékkel osztható-e és ha igen, akkor az adott szám négyzetét egy eredmény tömbbe másolja. Az eljárás térjen vissza a kimeneti tömbbe másol elemek számával.

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)

```
int filterDivisibleNumbersAndSquare(
    int input[],
    int length,
    int divisor,
    int output[]);
```

1. **input** a bemeneti tömb, mely 32 bites előjeles egész értékeket tartalmaz.
2. **length** a bemeneti tömb hossza (hány elem van a tömbben).
3. **divisor** az érték amivel oszthatónak kell lennie a bemeneti tömbben található számoknak.
4. **output** a kimeneti tömb.

A visszatérési értéke az eljárásnak megadja hány elem van a **output** tömbben.

Példa

Ha a bemenet: `[ -1, -2, -3, -4 -5 ]` és a **divisor** `2` akkor az eredmény: `[ 4, 16 ]`. A visszatérési érték pedig `2`.