5. Assembler

Pri tej nalogi bomo namesto običajnih programskih jezikov, kot so C/C++, pascal, java in podobni, uporabljali zelo preprost zbirni jezik (assembler) za nek izmišljen, zelo preprost procesor.

Program je zaporedje ukazov (vsi možni ukazi so opisani spodaj); pred vsakim ukazom lahko stoji tudi oznaka (labela), na katero se lahko sklicujemo pri pogojnih skokih. Razen v primeru pogojnih skokov pa se ukazi izvajajo po vrsti.

Program lahko uporablja poljubno število celoštevilskih spremenljivk (kot tip **int** oz. integer).

Poleg tega obstajata še dve logični spremenljivki, E in L, ki ju ne moremo spreminjati ali brati neposredno, pač pa z njima delajo nekatere inštrukcije (CMP nastavi njuno vrednost, JE in JL pa njuno vrednost bereta).

Poleg tega obstaja tudi sklad, ki lahko hrani poljubno dolgo zaporedje (seznam) celoštevilskih vrednosti. Do vsebine sklada ne moremo dostopati drugače kot prek ukazov PUSH (ki doda nov element na vrh sklada) in POP (ki pobere element z vrha sklada).

Oglejmo si zdaj seznam vseh ukazov, ki jih podpira naš namišljeni procesor:

- ADD x, y — prišteje vrednost spremenljivke y spremenljivki x (rezultat shrani v x);
- SUB x, y — odšteje vrednost spremenljivke y od spremenljivke x (rezultat shrani v x);
- CMP x, y — primerja vrednosti spremenljivk x in y, ter nastavi logični vrednosti (zastavici) E in L: E dobi vrednost **true**, če je $x = y$, sicer dobi vrednost **false**; L dobi vrednost **true**, če je $x < y$, sicer dobi vrednost **false**;
- SET x, c — nastavi spremenljivko x na vrednost c (ki mora biti neka celoštevilska konstanta);
- JE lbl — skoči na labelo lbl, če ima E vrednost **true**;
- JL lbl — skoči na labelo lbl, če ima L vrednost **true**;
- PUSH x — doda vrednost spremenljivke x na vrh sklada;
- POP x — pobere vrednost z vrha sklada in jo vpiše v spremenljivko x; če je bil sklad prazen, se procesor sesuje.

V tem zbirnem jeziku **napiši zaporedje ukazov**, ki izračuna zmnožek števil v spremenljivkah x in y; ob koncu izvajanja mora biti ta zmnožek shranjen v spremenljivki x. Predpostavi, da sta na začetku izvajanja vrednosti spremenljivk x in y večji ali enaki 0 in da sta dovolj majhni, da pri množenju ne bo prišlo do težav zaradi prekoračitve obsega celih števil ali česa podobnega. Poleg spremenljivk x in y lahko uporabiš še poljubno mnogo svojih pomožnih spremenljivk, poleg tega pa lahko uporabljaš tudi sklad, vendar mora biti sklad ob koncu izvajanja tvojega zaporedja ukazov vedno v enakem stanju kot na začetku izvajanja.

Zaželeno je, da je tvoja rešitev čim bolj učinkovita, torej da ob računanju zmnožka $x \cdot y$ izvede čim manj ukazov.

Za primer si oglejmo naslednje zaporedje ukazov, ki rešuje malo drugačen problem — s sklada pobere najprej n in nato še n števil ter na koncu v spremenljivki vsota izračuna vsoto tistih n števil.

```
  POP n
  SET vsota, 0
zacetek:
  SET temp, 0
  CMP n, temp
  JE konec
  POP x
  ADD vsota, x
  SET temp, 1
```

```
    SUB n, temp
    CMP temp, temp
    JE zacetek
konec:
```