1. Date su sljedeće naredbe u programskom jeziku C:

   a) a = b–c–d;
   b) a = b;
   c) a = –a;
   d) a = 0.

   Napisati ove naredbe u MIPS asemblerskom jeziku pod pretpostavkom da su promjenljivima **a**, **b**, **c** i **d** dodijeljeni registri $16, $17, $18 i $19, respektivno.

Rješenje:

a) Izraz se ne može riješiti direktno, zbog toga što MIPS aritmetičke instrukcije imaju tačno 3 operanda. Stoga ćemo najprije oduzeti **c** od **b**, a onda od toga oduzeti **d**, tj.: a = (b–c)–d. Koristićemo privremeni registar $8 za smještanje međurezultata b-c.

```
sub $8, $17, $18      # u registar $8 smještamo b-c
sub $16, $8, $19      # u registar $16 (promjenljiva a) smještamo (b-c)-d.
```

Prva razlika u odnosu na više programske jezike je ta što su operandi MIPS instrukcija registri, kojih u našem slučaju ima **32** (u oznaci $0, $1, ..., $31). S druge strane, operandi viših jezika su promjenljive, kojih može biti proizvoljno mnogo. Dužina registara, kao i dužina jedne memorijske riječi, je **32** bita.

Druga razlika u odnosu na većinu viših programskih jezika je ta što MIPS kod poštuje pravilo programske linije, tj. svaka instrukcija MIPS koda se nalazi u zasebnom redu. Komentari, ukoliko ih ima, se završavaju na kraju reda u kojem su počeli.

b) **a=b** ćemo napisati kao **a=b+0**. Vrijednost 0 dobijamo kao sadržaj registra $0 (programer ne može promijeniti sadržaj registra $0)

```
add $16, $0, $17      # u registar $16 smještamo vrijednost b+0=b.
```

Na ovaj način smo došli do instrukcije **move $16, $17** koja kopira sadržaj jednog registra u drugi. Iako nije hardverski implementirana, MIPS asembler prihvata ovu instrukciju. Ovakve instrukcije se nazivaju *pseudoinstrukcijama*.

c) **–a** ćemo napisati kao **0-a**, pa imamo:

```
sub $16, $0, $16      # u registar $16 smještamo vrijednost –a=0-a.
```

d) **a=0** ćemo dobiti kao **a=0+0**:

```
add $16, $0, $0       # u registar $16 smještamo vrijednost a=0+0.
```

2. Dat je dio koda u programskom jeziku C:

$$C[i] = A[i] – B[i]$$

a) Napisati ovu naredbu u MIPS asemblerskom jeziku pod pretpostavkom da su početne adrese nizova A, B i C: 1000, 1400 i 1800, respektivno. Pretpostaviti da registar $18 sadrži vrijednost 4*i. Kao privremene registre koristiti $19 i $20.

b) Kako izgleda format instrukcija dobijenih pod a)?

Rješenje:

a)

```
lw $19, 1000($18)   # i-ti element niza A u registar $19
lw $20, 1400($18)   # i-ti element niza B u registar $20
sub $20, $19, $20   # oduzimanje tih brojeva i rezultat u registar $20
sw $20, 1800($18)   # registar $20 u i-ti element niza C.
```

b) Izgled polja kod aritmetičkih instrukcija (R-tip) i instrukcija za prenos podataka (I-tip):

op operacija instrukcije;
rs registar prvog operanda;
rt registar drugog operanda;
rd registar u koji se smješta rezultat operacije;
shamt vrijednost za koju se šiftuje vrijednost registra (ovdje se ne koristi);
funct funkcijsko polje i ono bira varijantu operacije iz *op* polja.

Kod *lw* i *sw* index niza ide u *rs*, a registar u koji se smešta podatak iz memorije je *rt*.

lw $19, 1000($18) **35 18 19 1000** (dekadno)
1000 1110 0101 0011 0000 0011 1110 1000 (binarno)

sub $20, $19, $20 **0 19 20 20 0 34** (32 za sabiranje umjesto 34)
0000 0010 0111 0100 1010 0000 0010 0010

sw $20, 1800($18) **43 18 20 1800**
1010 1110 0101 0100 0000 0111 0000 1000