Dokumentacija

Napomena* Ulazna datoteka mora biti formatirana na način da između instrukcija stoji jedan razmak (*whitespace),* te da su instrukcije međusobno radzvojene jednim znakom \n – odnosno svaka u posebnom redu.

Također data implementacija podržava samo sljedeće instrukcije:

```
public static ArrayList<String> rType = new ArrayList<>() {
    {
        add("add");
        add("addu");
        add("and");
        add("nor");
        add("or");
        add("sll");
        add("sllv");
        add("slt");
        add("sltu");
        add("srav");
        add("srlv");
        add("sub");
        add("subu");
        add("xor");
        add("mul");
    }
};

public static ArrayList<String> iTypeNoMemory = new ArrayList<>() {
    {
        add("beq");
        add("bne");
        add("addi");
        add("addiu");
        add("slti");
        add("andi");
        add("ori");
        add("xori");
        add("lui");
    }
};

public static ArrayList<String> iTypeMemory = new ArrayList<>() {
    {
        add("lb");
        add("lh");
        add("lw");
        add("sb");
        add("sh");
        add("sw");
    }
};
```

```
public static ArrayList<String> jType = new ArrayList<>() {
    {
        add("jal");
        add("j");
        add("jr");
    }
};
```

One su sačuvane u ArrayList-ama radi lakšeg dododavanja / modifikovanja.

Opis klasa

- **Instruction**

  Klasa *Instruction* predstavlja apstraktnu klasu koja modelira zajedničke atribute za sve instrukcije MIPS procesora.

- **RInstruction**

  Modelira instrukcije R tipa MIPS procesora. Izvedena je iz klase *Instruction*.

- **IInstruction**

  Modelira instrukcije I tipa koje ne rade sa memorijom. Nasljeđuje klasu *Instruction.*

- **IMemInstruction**

  Modelira instrukcije I tipa koje rade sa memorijom. Također nasljeđuje klasu *Instruction*

- **Jinstruction**

  Modelira instrukcije J tipa, također izvedena iz klase *Instruction*.

  Prethodno navedene klase su uglavnom *JavaBean*, pa nema neke pretjerane potrebe za detaljnom analizom. One se sastoje uglavnom on *gettera i settera* za atribute. Jedina metoda vrijedna napomene jeste dajOdredisni() koja na osnovu polimorfizma vraća odredišni registar date instrukcije nad kojom je pozvana.

- **MIPSDelayedBranchDetector**

  Predstavlja glavnu klasu u kojoj je smještena sva logika za odabir instrukcija zadrške. Instrukcija zadrške je birana na način opisan u vježbama tj. birana je instrukcija prije instrukcije grananja ukoliko je ona nezavisna. Ukoliko to nije slučaj birana je instrukcija poslije instrukcija grananja ukoliko na odredištu grananja ni jedan izvorišni registar nije jednak odredišnom registru instrukcije zadrške. Ukoliko ni ona nije ispunila uslove, provjeravana je instrukcija na odredištu grananja – po sličnim uslovima.

Metode klase MIPSDelayedBranchDetector

```
public static void main(String[] args)
```

U main klasi se vrši logika dobavljanja i obrade instrukcija, na način da se pozivaju određene metode.

```
private static boolean validateTarget(IInstruction branch, Instruction target,
ArrayList<Instruction> instructions)
```

Provjera da li instrukcija na odredištu grananja zadovoljava uslove da bude instrukcija zadrške.

```
private static boolean validateNext(IInstruction branch, Instruction next,
ArrayList<Instruction> instructions)
```

Provjera da li instrukcija iza instrukcije grananja zadovoljava uslove da bude instrukcija zadrške.

```
private static boolean validateIndependent(Instruction ins1, Instruction ins2)
```

Provjera da li instrukcija prije instrukcije grananja zadovoljava uslove da bude instrukcija zadrške.

```
public static ArrayList<Instruction> loadInstructions(File file) throws Exception
```

Učitava instrukcije iz .txt datoteke, na način da parsira svaku liniju u instrukciju koju dodaje u listu istrukcija. Na kraju parsiraranja, lista istrukcija se šalje na mjesto poziva metode te se vrši dalja obrada. Ukoliko dođe to bacanja izuzetka u ovoj metodi to je indikacija da format ulazne datoteke je neispravan.

```
public static void writeInscructions(ArrayList<Instruction> instructionList, String
filePath)
```

Kreira izlaznu datoteku output.txt u kojoj upisuje pronađene instrukcije zadrške ili obavještava da je nemoguće / nepotrebno pronaći takve za datu sekvencu instrukcija.

Testiranje

Za testiranje su uglavnom iskorištene instrukcije generisanje uz pomoć generatora sa c2.

Sekvence koje su testirane se nalaze u folderu.

Za sekevencu:

SUB R0, R5, R7

label1: BEQ R4, R4, label2

AND R2, R5, R8

ADD R0, R7, R5

BEQ R6, R0, label1

label2: LW R3, 0(R8)

Program je generisao sljedeći sadržaj output.txt datoteke:

```
Instrukcije zadrške su ispisane u sljedećem formatu
 branch instrukcija - njena instrukcija zadrške:

label1:  BEQ R4, R4, label2- SUB R0, R5, R7
BEQ R6, R0, label1- label2:  LW R3, 0(R8)
```

Zatim za sekvencu:

SUB R0, R4, R1

label1: BEQ R6, R0, label2

OR R4, R9, R9

ADD R0, R4, R3

BEQ R6, R0, label1

label2: LW R2, 0(R3)

Program je generisao sljedeći sadržaj output.txt datoteke:

```
Instrukcije zadrške su ispisane u sljedećem formatu
 branch instrukcija - njena instrukcija zadrške:

label1:  BEQ R6, R0, label2- OR R4, R9, R9
BEQ R6, R0, label1- label2:  LW R2, 0(R3)
```

Za sekvencu:

LW R3, 9(R8)

label1: BEQ R6, R4, label2

SUB R6, R8, R6

ADD R0, R7, R4

BEQ R1, R0, label1

label2: LW R3, 0(R7)

```
Instrukcije zadrške su ispisane u sljedećem formatu
 branch instrukcija - njena instrukcija zadrške:

label1:  BEQ R6, R4, label2       - LW R3, 9(R8)
BEQ R1, R0, label1    - label2:  LW R3, 0(R7)
```

Za sekvencu:

SUB R2, R3, R5

OR R8, R2, R3

ADDI R3, R3, 5

Program je generisao sljedeći sadržaj output.txt datoteke:

```
Za datu sekvencu instrukcija nije moguce pronaci instrukciju zadrske ili ona nije
potrebna
```

Ukoliko ulazna datoteka ne zadovoljava format koji je naznačen na početku dokumenta program će ispisati da ulazna datoteka nije ispravnog formata.

Autori

Autori datog programskog rješenja su:

- Allen Kovačević (18314)
- Robert Tomas (18524)
- Mirza Učanbarlić (18483)

Program je pisan u *Javi*, kao što je ranije navedeno, te razvijen je uz pomoć IntelliJ okruženja. Kao VSC je korišten GitHub. Sam rad na projektu najviše se odvijao uz pomoć *online meeting*-a tima gdje se najviše vremena iskoristilo za postizanje dogovora kako će se određeni dio realizirati i na koji način. Na taj način je obezbjeđena fer raspodjela zadataka po pojedincu.