Turingov stroj

Pekić, Stjepan

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:632898

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-07-27**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

TURINGOV STROJ

Završni rad

Stjepan Pekić

Osijek, 2024. godina.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Stjepan Pekić
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R 4412, 11.10.2021.
JMBAG:	0165082344
Mentor:	prof. dr. sc. Josip Job
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Turingov stroj
Znanstvena grana završnog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rada:	Potrebno je istražiti i opisati mogućnosti primjene Turingovog stroja. Kao praktičan dio rada potrebno je simulirati rad Turingovog stroja. Tema rezervirana za: Stjepan Pekić

Datum prijedloga ocjene završnog rada od strane mentora:	17.06.2024.
Prijedlog ocjene završnog rada od strane mentora:	Dobar (3)
Datum potvrde ocjene završnog rada od strane Odbora:	26.06.2024.
Ocjena završnog rada nakon obrane:	Dobar (3)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	24.07.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 24.07.2024.

Ime i prezime Pristupnika:	Stjepan Pekić
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4412, 11.10.2021.
Turnitin podudaranje [%]:	7

Ovom izjavom izjavljujem da je rad pod nazivom**: Turingov stroj**

izrađen pod vodstvom mentora prof. dr. sc. Josip Job

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. PREGLED PODRUČJA TEME 2

2.1. Princip Rada Turingovog Stroja 2

2.2. Područje primjene Turingovog stroja 4

2.3. Primjeri primjene Turingovog stroja 7

3. PRIMJENA TURINGOVOG STROJA KAO BINARNOG KALKULATORA . 10

4. IMPLEMENTACIJA RJEŠENJA 15

4.1. Zbrajanje 15

4.2. Oduzimanje 19

4.3. Množenje 20

4.4. Dijeljenje 22

4.5. Poziv funkcija 22

5. ZAKLJUČAK 26

6. SAŽETAK 27

7. ABSTRACT 28

8. LITERATURA 29

9. ŽIVOTOPIS 30

1. UVOD

Turingov stroj, kojeg je 1936. Godine osmislio Alan Turing, predstavlja temeljni koncept u teorijskoj računalnoj znanosti i umjetnoj inteligenciji. Ovaj apstraktni model računala može izvesti bilo koji algoritam kroz jednostavne mehaničke operacije, što ga čini ključnim za razumijevanje modernih računala i računskih procesa. Iako Turingov stroj nije praktičan za stvarnu upotrebu zbog svojih fizičkih ograničenja, njegova teorijska važnost je neupitna.

Primjenu Turingovog stroja pronalazimo u različitim područjima računalne znanosti, uključujući teoriju algoritama, formalne jezike, teoriju složenosti i umjetnu inteligenciju. Njegov koncept omogućava analizu i razumijevanje osnovnih principa obrade podataka i računalnih operacija.

Ovaj se rad bavi realizacijom binarnog kalkulatora po principu rada Turingovog stroja. Problem kojeg rješava uključuje teorijsku pozadinu implementacije kalkulatora, način rada Turingovog stroja, primjere njegove primjene i povezane aplikacije. U prvom dijelu rada koncentracija je na teorijskim aspektima, dok drugi dio rada temeljito objašnjava funkcioniranje svih dijelova implementiranog koda binarnog kalkulatora.

1.1. Zadatak završnog rada

U teorijskom dijelu rada potrebno je istražiti i opisati mogućnosti primjene Turingovog stroja. Dok je kao praktični dio rada potrebno kodom simulirati rad Turingovog stroja.

2. PREGLED PODRUČJA TEME

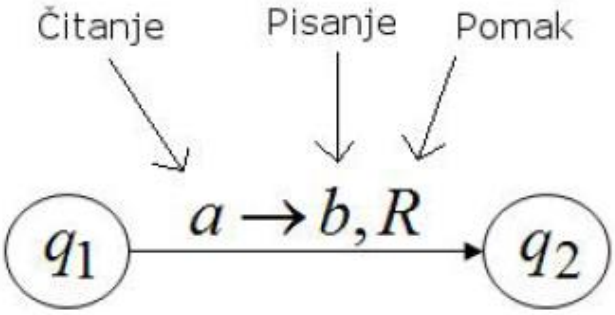
U ovom poglavlju analizirani su temeljni aspekti Turingovog stroja, uključujući njegov princip rada, područja primjene i konkretne primjere njegove uporabe. Kroz ova potpoglavlja pruža se sveobuhvatan pregled značaja i utjecaja Turingovog stroja u različitim granama računalne znanosti.

2.1. Princip Rada Turingovog Stroja

Turingov stroj je konceptualiziran kao uređaj koji se sastoji od dva glavna dijela, beskonačne vrpce koja služi kao memorija i 'glave' uređaja koja obavlja zadatke na podatcima zapisanima na vrpcu.

Beskonačnu vrpcu u stvarnosti može predstavljati papirnata vrpca koja je podijeljena na jednake kvadratiće u kojima su, na primjer, zapisani simboli '1', '0' ili '/' koji bi predstavljao prazno mjesto. Simbola može biti i više, ovisno o potrebi. 'Glava' uređaja djeluje na vrpci na način da pročita simbol koji je zapisan na vrpci, na temelju njega i trenutnog unutrašnjeg stanja u kojemu se glava nalazi zapisuje drugi simbol na to mjesto, to može biti isti simbol koji je bio i prije, zatim po potrebi mijenja svoje unutrašnje stanje te se pomakne po vrpci lijevo ili desno za točno jedno mjesto.

Ovaj opis principa rada Turingovog stroja prilagođen je prema konceptima predstavljenim na znanstvenoj web stranici Hrvatski matematički elektronički časopis.



Sl. 2.1. Način zapisa promjene stanja u Turingovom stroju

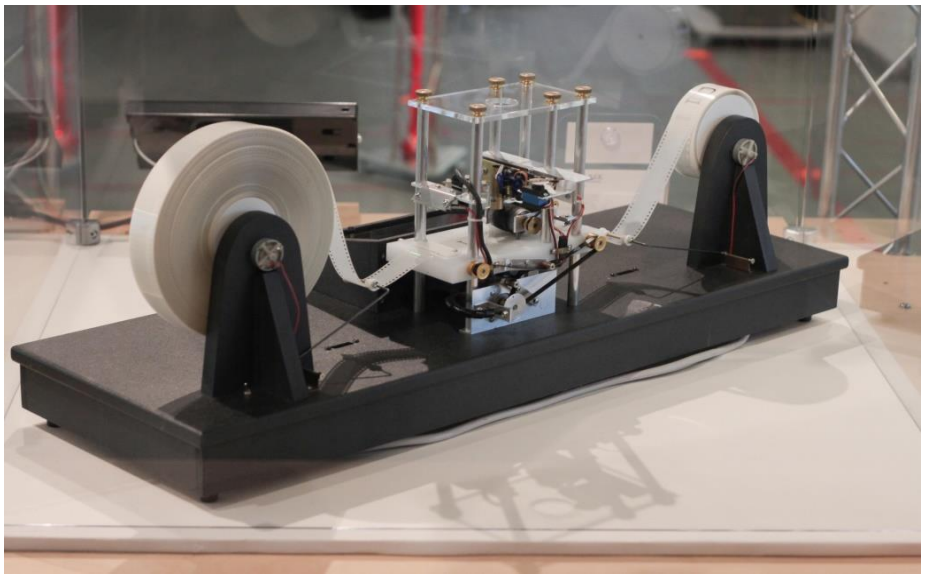
Na temelju ovog prilično jednostavnog postupka pomoću Turingovog stroja moguće je izvesti bilo koji algoritam koji izvodi računalo, odnosno sva moderna računala mogu raditi samo ono što se može opisati pomoću principa rada Turingovog stroja. Princip Turingovog stroja ustvari je toliko jak da se koristi za procjenu moći računalnih programa. Kada program može napraviti sve ono što Turingov stroj može, onda taj program označavamo kao **Turing kompletan** što je najveća razina po snazi računalnih programa.

Formalna definicija Turingovog stroja predstavljena je kao uređena sedmorka $(Q, \Sigma, \Gamma, _, s, F, \delta)$, gdje je:

- Q konačan skup čije elemente zovemo stanjima,
- Σ konačna ulazna abeceda,
- Γ konačna abeceda trake, takav da je $\Sigma \subset \Gamma$,
- $_ \in \Gamma \setminus \Sigma$ posebno odabran simbol blank, koji predstavlja "prazninu",
- $s \in Q$ istaknuto početno stanje,
- $F \subseteq Q$ skup završnih stanja,
- δ funkcija prijelaza

Također uz regularni Turingov stroj, (Slika 2.2.), imamo i njegove varijacije:

- **Turingov stroj sa više beskonačnih vrpci**
- nedeterministički Turingov stroj
- Turingov stroj sa više glava
- Turingov stroj sa dodatnom linijom sa koje stroj samo čita
- višedimenzionalni Turingov stroj (za zapisivanje koristi 2D tablicu)



Sl. 2.2. Model Turingovog stroja

2.2. Područje primjene Turingovog stroja

U ovom poglavlju cilj je navesti i opisati različite primjere primjene Turingovog stroja u stvarnom svijetu. Gdje, zašto i kao se koristi.

Turingov se stroj u praksi uglavnom koristi u apstraktnom obliku pomoću čijeg se principa pokušavaju riješiti problemi u misaonim eksperimentima i zadacima. Ne koristi se kao stvarni fizički stroj jer su računala puno efikasnija u rješavanju problema od mehaničkog uređaja. Funkcionalan mehanički stroj moguće je napraviti samo kao primjer za vizualno prikazivanje kako stroj radi na temelju nekog vrlo jednostavnog algoritma.

2.2.1. Problem izračunljivosti

Kada Turingov stroj radi na nekom algoritmu to može završiti na tri moguća načina. U dva slučaja stroj odradi određene korake i zaustavlja se te na temelju odrađenih koraka daje pozitivan ili negativan odgovor. Također postoji i treća opcija gdje se Turingov stroj nikada ne zaustavlja nego zapne u beskonačnoj petlji i stalno ponavlja isto, što se zna dogoditi i računalima, odnosno kada program ne završava, nego se izvodi u beskonačnost.

Izračunljivim problemom smatra se svaki problem za koji Turingov stroj izvrši algoritam i zaustavi se. Neovisno o dobivenom izlazu da li je pozitivan ili negativan bitno je da stroj dođe u finalno stanje mirovanja.

U području teorije izračunljivosti, problem zaustavljanja je problem odlučivanja koji se na neformalan način može opisati ovako: Za dani opis programa i konačnog ulaza, odlučuje zaustavlja li se program ili se izvršava unedogled.

Poznati problem neizračunljivosti je The Busy Beaver Problem koji je nastao 1962. godine. U ovom problemu se traži odgovor na pitanje koliki je maksimalan broj jedinica koje možemo zapisati na traku pomoću danog Turingovog stroja i kada se zaustavlja. Vrlo je važno da se stroj zaustavi jer se traži konačan broj jedinica kao odgovor, a ne prihvaća se odgovor da ide u beskonačnost jer onda se nikada ne zaustavlja.

Za ideju Turingovog stroja sa samo jednim unutrašnjim stanjem, za ovaj problem može se napraviti 64 različita ponašanja, svako ponašanje (stanje) predstavljeno je sa jednostavnim uputama o tome što stroj treba napraviti u slučaju određenog ulaznog podatka, koja Turingov stroj može izvesti u jednom koraku, odnosno 64 različita Turingova stroja. Iako postoji 64

različita stroja sa jednom karticom ponašanja ustvari to su samo 4 različite opcije ponašanja stroja,a 3 od 4 završe u stanju mirovanja.

Ponašanja koja dobijemo su:

- Ispisivanje jedinica ili nula u jednom smjeru unedogled
- Zapisivanje jedinice pomicanje u lijevo i zaustavljanje
- Zapisivanje jedinice pomicanje u desno i zaustavljanje
- Zapisivanje nule na mjesto nule pomicanje u jednu stranu i zaustavljanje

Tim ponašanjima dolazi se do zaključka da je sa jednim unutrašnjim stanjem moguće zapisati najviše jednu jedinicu prije zaustavljanja.

Povećanjem broja unutrašnjih stanja broj mogućih Turingovih strojeva eksponencijalno raste. Sa dva moguća stanja broj mogućih strojeva raste na 20,736, a rezultat su maksimalno 4 jedinice. Za tri stanja postoji preko 16 miliona strojeva i najbolji rezultat je 6 jedinica, a za 4 stanja najbolji rezultat je 13 te je mogućih strojeva preko 25.6 milijardi.

Za stroj sa pet stanja trenutni najbolji rezultat je ostvaren prije više desetljeća a iznosi 4098 jedinica. Istraživanje nije završeno jer neki strojevi još uvijek rade i još se ne može odrediti jesu li zapeli u petlji ili ne, ali ako nisu moguće je da će rezultat biti još veći.

Daljnjim povećanjem broja stanja dobivamo toliki broj kombinacija da ih je nemoguće proučiti sa trenutnim računalima i tu za sada zapinje daljnje istraživanje koje i dalje traje. Značajniji napredak bi mogao donijeti razvoj tehnologije i kvantnog računala.

2.2.2. Prepoznavanje uzoraka

Još za života Alan Turing postavio je mehaničko-matematički model koji je objašnjavao svojstva formiranja uzoraka u prirodi te je došao prilično blizu do definiranja prvog staničnog automata. Samo godinu dana nakon objavljivanja jednog od Turingovih radova došlo je do revolucionarnog otkrića u biologiji, a to je dvolančana struktura deoksiribonukleinske kiseline (DNA).

Struktura DNA, a još više RNA, izgledom jako podsjeća na vrpcu koju koristi Turingov stroj. Također i jedna i druga stvar služe za pohranjivanje podataka.

U ovom slučaju pomoću Turingovog stroja može se opisati proces replikacije DNA u kojoj se od jedne molekule stvaraju dvije. Molekula DNA se razdvaja na dva lanca koji svaki sebi treba napraviti odgovarajuću drugu stranu koju je i do sada imao. Svaka poveznica u lancu DNA sastoji se od dvije polovice i svaka strana je načinjena od jedne od četiri moguće vrste koje označavamo slovima A,C,G,T, gdje su odgovarajuće strane A i T te C i G. Kodiranjem tih jednostavnih pravila u Turingov stroj mogli bi napraviti simulator koji bi za zadan niz koji predstavlja polovicu DNA molekule generirao niz koji opisuje izgled druge polovice.

U kriptografiji Turingov stroj se primjenjuje za analizu i razvoj sigurnosnih algoritama. Pomoću njega može se oponašati modele napada na kriptografske sustave i olakšati testiranje sigurnosti tih sustava.

Kriptografija se često koristi teškim problemima, za koje treba jako puno računalne snage da bi se riješili, u svrhu čuvanja tajnih podatka. Tu do izražaja dolazi teorija kompleksnosti koja proučava koliko je potrebno snage i resursa da bi se određeni problem riješio u zadanom vremenu. Turingov stroj tu pomaže u analizi vremenske kompleksnosti kriptografskog algoritma, a time dobivamo informacije o kvaliteti i učinkovitosti algoritma

U kriptoanalizi Turingov se stroj može koristiti kao teorijski model za stvaranje algoritama za napade sirovom snagom gdje se redom isprobavaju kombinacije ključeva dok se ne odredi onaj pravi za dešifriranje.

2.2.3. Edukacija

Turingovi strojevi često se koriste kao pedagoški alati u informatičkom obrazovanju za podučavanje temeljnih pojmova računanja, algoritama i teorije automata. Temelji rada Turingovog stroja se također uče na FERIT-u na kolegiju Automati i formalni jezici te na kolegiju Matematičke osnove računarstva.

2.2.4. Simulacije i stvaranje modela

U različitim znanstvenim područjima simulacije složenih sustava izvode se pomoću računalnih modela temeljenih na Turingovim strojevima. Ove simulacije pomažu znanstvenicima da bolje i lakše razumiju i predvide ponašanje prirodnih pojava.

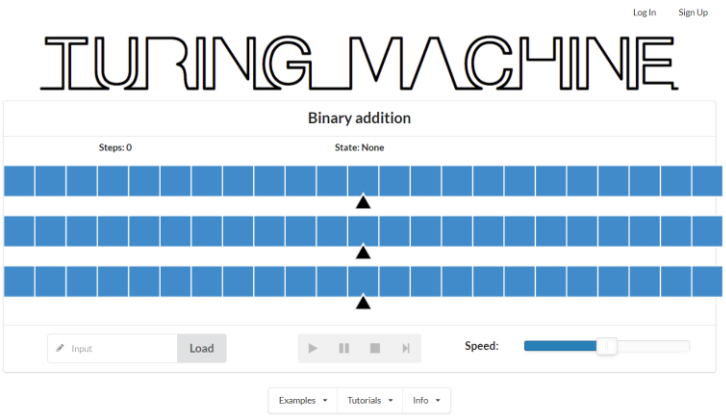
2.2.5. Programski jezici te razvoj i optimizacija kompajlera

Razumijevanje teoretskih temelja računanja, uključujući Turingove strojeve, ključno je za dizajniranje programskih jezika i kompajlera. Ovi se alati koriste za razvoj softverskih aplikacija za različite namjene.

2.3. Primjeri primjene Turingovog stroja

U današnje vrijeme primjećuje se značajno veća ponuda web aplikacija u usporedbi s Android aplikacijama koje se temelje na Turingovom stroju ili izvode binarno računanje. Ova razlika proizlazi iz nekoliko ključnih čimbenika. Web aplikacije su dostupne na bilo kojem uređaju s internet preglednikom, što ih čini pristupačnijima i jednostavnijima za korištenje. Razvoj i održavanje web aplikacija često su brži i manje zahtjevni u smislu resursa, budući da nije potrebno prilagođavanje za različite mobilne platforme. Osim toga, web aplikacije su omiljene u obrazovnom sektoru zbog svoje lakoće pristupa, što omogućuje studentima i istraživačima da bez instaliranja posebnog softvera koriste alate za simulaciju Turingovih strojeva i binarnog računanja. Trenutno u ponudi nema aplikacije koja isključivo odrađuje binarnu aritmetiku po principu Turingovog stroja nego su to uglavnom simulatori koji uz binarno zbrajanje rade još cijeli niz drugih operacija kao što su prepoznavanje palindroma ili utvrđivanja parnosti/neparnosti broja nula/jedinica u binarnim brojevima. Također uz simulatore postoje i programi koji se koriste za edukaciju u obrazovnim ustanovama te razne mobilne igrice koje oponašaju Turingov stroj.

2.3.1. Online Turing Machine Simulator - Online simulator Turingovog stroja

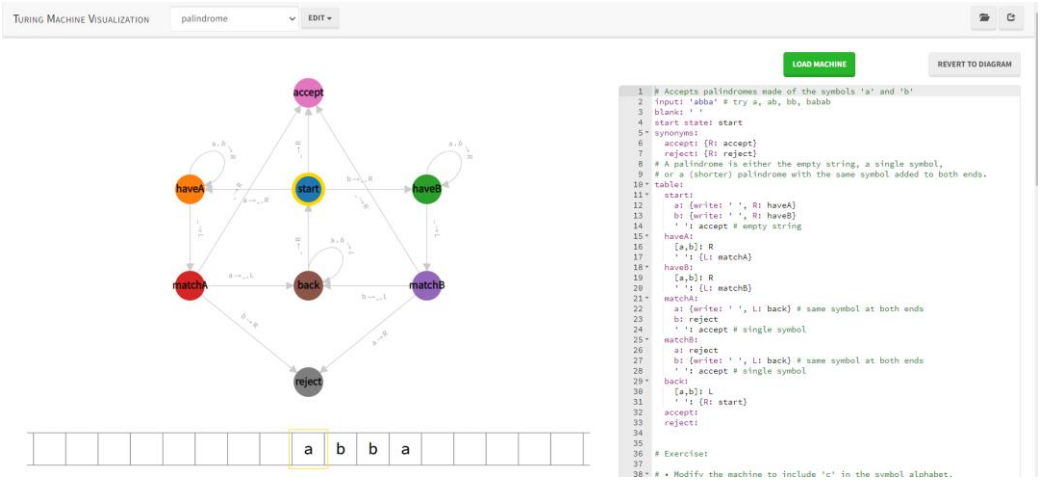


Slika 2.3. Online Turing Machine Simulator

Turing Machine Simulator, dostupan na web stranici turingmachinesimulator.com, (Slika 2.3.), pruža korisnicima intuitivno sučelje za istraživanje i simuliranje Turingovih strojeva. Ova aplikacija omogućuje korisnicima da programiraju strojeve definirajući stanja, simbole i prijelaze, te ih zatim testiraju s različitim ulaznim nizovima. Sučelje omogućuje vizualno praćenje izvršavanja stroja, olakšavajući analizu rezultata simulacije. Edukacijska vrijednost aplikacije leži u njenom interaktivnom pristupu, koji je koristan za studente i istraživače koji žele bolje razumjeti koncepte Turingovih strojeva, formalnih jezika i automata.

2.3.2. Turing machine visualization - Vizualizacija Turingovog stroja

Turing machine visualisation je također dostupna web stranica, (Slika 2.4.), koja korisnicima, uz pisanje koda, stvaranje stanja i prijelaza, omogućuje prikaz koda pomoću interaktivnog dijagrama konačnih automata. Preko tog dijagrama također pratimo izvršavanje programa i u svakom trenutku možemo vidjeti u kojem stanju se Turingov stroj nalazi.



Sl. 2.4. Turing machine visualization

2.3.3. Turing machine 2D – Turingov stroj 2D

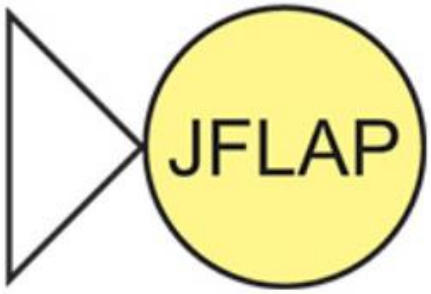
Turing Machine 2D je Android aplikacija koja omogućuje korisnicima istraživanje i simuliranje Turingovih strojeva u dvodimenzionalnom prostoru, (Slika 2.5.). Ova aplikacija nudi sučelje za programiranje stanja, simbola i prijelaza, te pruža vizualnu reprezentaciju izvršavanja stroja na dvodimenzionalnoj traci odnosno u tablici. Korisnici mogu definirati složene algoritme i pratiti njihovo izvršavanje u realnom vremenu, što ovu aplikaciju čini korisnom za edukaciju i istraživanje u području teorije računarstva.

Sl. 2.5. Turing machine 2D

2.3.4. JFLAP

JFLAP (Java Formal Languages and Automata Package), (Slika 2.6.), je sveobuhvatan obrazovni alat dizajniran za učenje formalnih jezika, automata i teorije računarstva. Omogućava stvaranje i

simulaciju raznih tipova automata, uključujući Turingove strojeve, regularne izraze, beskontekstne gramatike i druge formalne modele.



Sl. 2.6. JFLAP logo

2.3.5. CalckProfi – binarni online kalkulator

CalcProfi je web aplikacija koja pruža binarni kalkulator omogućujući korisnicima obavljanje raznih aritmetičkih operacija u binarnom sustavu, ali ne pomoću Turingovog stroja. Ova aplikacija nudi intuitivno sučelje za jednostavno zbrajanje, oduzimanje, množenje i dijeljenje binarnih brojeva, kao i konverzije između binarnih, decimalnih, heksadecimalnih i oktalnih sustava. CalcProfi je koristan alat za studente, programere i sve one koji rade s binarnim brojevima, pružajući brzo i točno rješenje za svakodnevne potrebe binarnog računanja, (Slika 2.7.).

Sl. 2.7. CalkProfi logo

Cilj ovog rada je program koji kombinira svojstva simulatora i vizualizatora Turingovog stroja te binarnog kalkulatora, na primjer kombinacija aplikacije Online simulator Turingovog stroja i CalckProfi. Program će raditi na principu turingovog stroja i nudit će operacije za potpuno funkcionalan binarni kalkulator, ali izmjena koda neće biti moguća.

3. PRIMJENA TURINGOVOG STROJA KAO BINARNOG KALKULATORA

U razmatranju kreiranja binarnog kalkulatora putem koncepta Turingovog stroja, ključno je prvo razumjeti osnove Turingovog stroja. Turingov stroj je apstraktni model računarstva koji se sastoji od konačnog skupa stanja, beskonačne vrpce podijeljene u ćelije koje mogu sadržavati simbole, te kretanja po tim vrpcama na temelju trenutnog stanja i simbola koji se nalazi pod glavom čitača. Inspiriran ljudskom interakcijom s papirom i olovkom prilikom izvođenja algoritama, Turingov stroj pruža konceptualni okvir odnosno temelje za bolje razumijevanje rada složenijih računalnih sustava i algoritama.

U ovom radu, fokus je na implementaciji funkcionalnog binarnog kalkulatora koristeći Python programski jezik kao alat za simulaciju rada Turingovog stroja. Kroz programski kod, cilj je stvoriti sustav koji može obavljati osnovne aritmetičke operacije poput zbrajanja, oduzimanja, množenja i dijeljenja na binarnim brojevima. Bitno je istaknuti da se binarni kalkulator modelira kao Turingov stroj s više beskonačnih vrpci, što omogućuje rukovanje binarnim brojevima i izvođenje osnovnih aritmetičkih operacija.

Prva aritmetička operacija je zbrajanje. Za izvođenje zbrajanje korištene su tri „beskonačne“ vrpce, tri unutarnja stanja te 27 tranzicija određenih unutarnjim stanjem i simbolima zapisanim na vrpci. Prve dvije vrpce služe za zapis ulaznih brojeva dok se treća vrpca koristi za zapisivanje rezultata. Prvo od tri stanja se koristi za namještanje glava stroja na početne položaje prije izvršavanja. Drugo i treće stanje služe za obavljanje zbrajanja jedinicu po jedinicu pribrojnika i izmjenjuju se ovisno o pojavi prijenosa.

Na primjeru zbrajanja dva binarna broja '1110' i '101' objašnjen je proces rada koda Turingovog stroja za zbrajanje, korak po korak. Na samome početku kada učitamo brojeve, vrpce izgledaju ovako:

3	⬇-označava položaj čitača na vrpci								
4			⬇						
5	vrpca 1	#	1	1	1	0	#		
6			⬇						
7	vrpca 2	#	1	0	1	#			
8		⬇							
9	vrpca 3	#	#	#	#	#	#	#	#

Sl. 3.1. Korak 1

Pozicije čitača označene su strelicama iznad svake vrpce i trenutno se nalazimo u stanju q0, (Slika 3.1.).

Tijekom stanje q0 pomiču se čitači vrpce 1 i 2 sve dok ne dođu do kraja zapisanih brojeva, odnosno dok ne dođu do simbola '#', (Slika 3.2 – Slika 3.5.).

3	⬇-označava položaj čitača na vrpci								
4				⬇					
5	vrpca 1	#	1	1	1	0	#		
6				⬇					
7	vrpca 2	#	1	0	1	#			
8		⬇							
9	vrpca 3	#	#	#	#	#	#	#	#

Sl. 3.2. Korak 2

3	⬇-označava položaj čitača na vrpci								
4					⬇				
5	vrpca 1	#	1	1	1	0	#		
6					⬇				
7	vrpca 2	#	1	0	1	#			
8		⬇							
9	vrpca 3	#	#	#	#	#	#	#	#

Sl. 3.3. Korak 3

3	⬇-označava položaj čitača na vrpci								
4						⬇			
5	vrpca 1	#	1	1	1	0	#		
6						⬇			
7	vrpca 2	#	1	0	1	#			
8		⬇							
9	vrpca 3	#	#	#	#	#	#	#	#

Sl. 3.4. Korak 4

Na slici 3.4. se vidi da čitač vrpce 2 očitava simbol '#' te se nakon toga zaustavlja i čeka da i čitač trake 1 također dođe do svoje željene pozicije.

3	⬇-označava položaj čitača na vrpci								
4							⬇		
5	vrpca 1	#	1	1	1	0	#		
6						⬇			
7	vrpca 2	#	1	0	1	#			
8		⬇							
9	vrpca 3	#	#	#	#	#	#	#	#

Sl. 3.5. Korak 5

U koraku 5 vidi se da su oba čitača došla do simbola '#'. Sada Turingov stroj prelazi u stanje q1 tijekom kojeg se čitači 1 i 2 kreću ulijevo.

3	⬇-označava položaj čitača na vrpci								
4						⬇			
5	vrpca 1	#	1	1	1	0	#		
6					⬇				
7	vrpca 2	#	1	0	1	#			
8		⬇							
9	vrpca 3	1	#	#	#	#	#	#	#

Sl. 3.6. Korak 6

Sa slikom 3.6. i korakom 6 stvarno počinje zbrajanje. Zbrajaju se simboli koje čitači vide na vrpcama 1 i 2 te se rezultat zapisuje na vrpcu 3. Ostaje se u stanju q1, po vrpcama 1 i 2 se pomiče se ulijevo, a po vrpci 3 udesno.

3	⬇-označava položaj čitača na vrpci								
4					⬇				
5	vrpca 1	#	1	1	1	0	#		
6				⬇					
7	vrpca 2	#	1	0	1	#			
8			⬇						
9	vrpca 3	1	1	#	#	#	#	#	#

Sl. 3.7. Korak 7

U ovom slučaju su isti simboli na vrpcama 1 i 2 samo su se zamijenili što rezultira istim rezultatom, (Slika 3.7.).

.

3	⬇-označava položaj čitača na vrpci								
4				⬇					
5	vrpca 1	#	1	1	1	0	#		
6			⬇						
7	vrpca 2	#	1	0	1	#			
8				⬇					
9	vrpca 3	1	1	0	#	#	#	#	#

Sl. 3.8. Korak 8

Na slici 3.8. dolazi do slučaja kada oba čitača na vrpcama 1 i 2 očitavaju simbol '1'. Kada se zbroje dvije jedinice u binarnom sustavu rezultat iznosi $10_{(2)}$, a njega se ne može zapisati na jedno mjesto. U tom slučaju jedinica se pamti, odnosno prenosi se dalje te se zbraja sa idućim

simbolima sa vrpci 1 i 2, a zapisuje se nula. Pošto je prisutna jedinica prijenosa vrijede drugačija pravila stoga se prelazi u stanje q2 i nastavlja se kretati u istim smjerovima po vrpcama.

3	⬇-označava položaj čitača na vrpci								
4			⬇						
5	vrpca 1	#	1	1	1	0	#		
6		⬇							
7	vrpca 2	#	1	0	1	#			
8					⬇				
9	vrpca 3	1	1	0	0	#	#	#	#

Sl. 3.9. Korak 9

Čitač na vrpci 2 stigao je do simbola '#' što znači da je prošao kroz sve znamenke drugog pribrojnika, tu će se zaustaviti i kod zbrajanja se uzima da on vidi nulu jer nula ne utječe na zbrajanje. Na vrpci 1 čitač vidi jedinicu, koja kad se zbroji sa nulom sa vrpce 2 daje rezultat 1, ali se stroj nalazi u stanju q2 i prisutna je jedinica prijenosa pa je rezultat opet $10_{(2)}$. Opet se zapisuje nula i nastavlja dalje u stanju q2, (Slika 3.9.)

3	⬇-označava položaj čitača na vrpci								
4		⬇							
5	vrpca 1	#	1	1	1	0	#		
6		⬇							
7	vrpca 2	#	1	0	1	#			
8						⬇			
9	vrpca 3	1	1	0	0	1	#	#	#

Sl. 3.10. Korak 10

Oba čitača su prošla preko svih znamenki pribrojnika što znači da je proces zbrajanja završio. Pošto se i dalje stroj nalazi u stanju q2 i postoji jedinica prijenosa, ona se mora zapisati na vrpcu 3 iako nema više znamenki sa vrpci 1 i 2 koje bi se zbrajale. Čitači na vrpcama 1 i 2 očitavaju simbol '#' što znači da se prelazi u finalno stanje q3 sa kojim se završava zbrajanje i stroj ulazi u stanje mirovanje, (Slika 3.10.).

U nastavku se programa obrađuje rezultat odnosno vrpca 3 i pripremama se za ispis tako što se uklone svi nepotrebni simboli '#' te se ispiše obrnutim redoslijedom, jer je obrnuto na nju i zapisano. Ovim postupkom dobije se rezultat zbrajanja brojeva $1110_{(2)}$ i $101_{(2)}$ koji iznosi $10011_{(2)}$, što bi u dekadskom sustavu bilo 14+5=19.

Na vrpcama 1 i 2 ništa se ne mijenja, odnosno onaj simbol koji se pročita taj simbol se i zapiše iz razloga da se u svakom trenutku može vidjeti sa čim se radilo u prethodnim koracima i kako se stiglo do trenutnog stanja i rezultata.

Oduzimanje je realizirano vrlo slično zbrajanju. Također su korištene tri vrpce, tri stanja i 23 tranzicije. Prve dvije vrpce služe za zapis ulaznih brojeva dok se treća vrpca koristi za zapisivanje rezultata. Kao i kod zbrajanja prvo stanje služi za pripremu dok se drugo i treće izmjenjuju u ovisnosti o tome da li postoji „posuđena“ jedinica, kao što bi se dogodilo u primjeru $100_{(2)}$ -$1_{(2)}$ gdje se od nule ne može oduzeti jedinica. Tijekom procesa oduzimanja u slučaju kada stroj ustanovi da je prvi upisani broj (umanjenik) manji od drugog (umanjitelja) onda se funkcija oduzimanja poziva ponovno ali sa zamijenjenim brojevima te se u ispis rezultata dodaje minus.

Množenje i dijeljenje ostvareno je kombinacijom zbrajanja i oduzimanja. Za množenje se dobije rezultat kao i kod zbrajanja odnosno oduzimanja dok se kod dijeljenja dobije rezultat cjelobrojnog dijeljenja i ostatak.

Kroz implementaciju ovog koda, osim razvijanja praktičnih vještina u programiranju, upoznaje se sa osnovnim konceptima teorijske računalne znanosti i binarne matematike. Analiza algoritama za osnovne aritmetičke operacije na binarnim brojevima pruža uvid u složenost računalnih problema te potiče razmišljanje o efikasnijim strategijama rješavanja problema. Također, kroz proces implementacije Turingovog stroja za binarni kalkulator, razvija se razumijevanje principa rada računalnih sustava te se susreće s izazovima i rješenjima koji su ključni u praksi računalne znanosti.

4. IMPLEMENTACIJA RJEŠENJA

U ovom poglavlju detaljno je opisana implementacija binarnog kalkulatora koji omogućava osnovne aritmetičke operacije: zbrajanje, oduzimanje, množenje i dijeljenje. Koristeći principe rada Turingovog stroja, razvijen je sustav koji simulira ove operacije na binarnim brojevima. Prikazani su ključni koraci u razvoju kalkulatora, uključujući algoritme za svaku operaciju, strukturu koda i način testiranja.

4.1. Zbrajanje

Na početku svake operacije pa tako i zbrajanja prvo se unose dva binarna broja koja služe kao argumenti u pozivanju funkcije i koriste se za inicijalizaciju listi koje koristimo kao beskonačne vrpce Turingovog stroja. Prije unošenja na vrpce također uklanjaju se nule s početka brojeva ako ih ima, (Programski kod 4.1.).

```
def addition(num1, num2, withsteps):
    while num1.startswith('0'): num1 = num1[1:]
    while num2.startswith('0'): num2 = num2[1:]
    tape1 = list('#' + num1 + '#')
    tape2 = list('#' + num2 + '#')
    tape3 = list('#'+ '#'+ '#'+ '#'+ '#'+ '#'+ '#'+ '#'+ '#'+ '#'+
'#'+ '#'+ '#'+ '#'+ '#'+ '#'+ '#'+ '#'+ '#'+ '#'+ '#')
```

Programski kod 4.1. Inicijalizacija vrpci

Programski kod 4.2. prikazuje početno postavljanje trenutnog stanja i položaja čitača na beskonačnoj vrpci, odnosno na listama gdje su zapisani brojevi koji se zbrajaju i brojača koraka koji služi za ispis.

```
    state = 'q0'
    head1 = 1  # Head position on tape 1
    head2 = 1  # Head position on tape 2
    head3 = 0  # Head position on tape 3
    step = 0
```

Programski kod 4.2. Početna stanja

Zatim slijedi popis tranzicija pomoću kojih se prolazi kroz znamenke zapisanih brojeva i na temelju njih mijenjaju stanja te se izvršava zbrajanje, čiji se rezultat zapisuje na treću vrpcu, (Programski kod 4.4.).

```
    ('q2', '0', '0'): ('q1', '0', '0', 'L', 'L', '1'),
```

Programski kod 4.3. Zapis tranzicije

Svaka tranzicija sastoji se od dva dijela. Prvi se dio sastoji od trenutnog stanja, simbola kojeg glava čitača vidi na prvoj vrpci i simbola sa druge vrpce. Na temelju te trojke pronalaze se podatci, za daljnji rad, koji su zapisani u drugom dijelu tranzicije. Drugi dio sastoji se redom od budućeg stanja, simbola koji se zapisuju na prvu i drugu vrpcu, zatim smjera kretanja u kojem se pomiče za jedno mjesto za prvu i drugu vrpcu te zadnje je simbol koji se zapisuje na treću vrpcu za rezultat, (Programski kod 4.3.).

Stanja kroz koja prolazi Turingov stroj označena su sa slovom 'q' i brojem, na primjer 'q1'. Za zbrajanje postoje tri radna stanja q0, q1 i q2 te stanje q3 koje označava završetak rada, odnosno finalno stanje.

```
    transitions = {
        ('q0', '0', '0'): ('q0', '0', '0', 'R', 'R'),
        ('q0', '0', '1'): ('q0', '0', '1', 'R', 'R'),
        ('q0', '1', '0'): ('q0', '1', '0', 'R', 'R'),
        ('q0', '1', '1'): ('q0', '1', '1', 'R', 'R'),
        ('q0', '#', '0'): ('q0', '#', '0', 'S', 'R'),
        ('q0', '#', '1'): ('q0', '#', '1', 'S', 'R'),
        ('q0', '0', '#'): ('q0', '0', '#', 'R', 'S'),
        ('q0', '1', '#'): ('q0', '1', '#', 'R', 'S'),
        ('q0', '#', '#'): ('q1', '#', '#', 'L', 'L'),

        ('q1', '0', '0'): ('q1', '0', '0', 'L', 'L', '0'),
        ('q1', '0', '1'): ('q1', '0', '1', 'L', 'L', '1'),
        ('q1', '1', '0'): ('q1', '1', '0', 'L', 'L', '1'),
        ('q1', '1', '1'): ('q2', '1', '1', 'L', 'L', '0'),
        ('q1', '#', '0'): ('q1', '#', '0', 'S', 'L', '0'),
        ('q1', '#', '1'): ('q1', '#', '1', 'S', 'L', '1'),
        ('q1', '0', '#'): ('q1', '0', '#', 'L', 'S', '0'),
        ('q1', '1', '#'): ('q1', '1', '#', 'L', 'S', '1'),
        ('q1', '#', '#'): ('q3', '#', '#', 'S', 'S', '#'),

        ('q2', '0', '0'): ('q1', '0', '0', 'L', 'L', '1'),
```

```
        ('q2', '0', '1'): ('q2', '0', '1', 'L', 'L', '0'),
        ('q2', '1', '0'): ('q2', '1', '0', 'L', 'L', '0'),
        ('q2', '1', '1'): ('q2', '1', '1', 'L', 'L', '1'),
        ('q2', '#', '0'): ('q1', '#', '0', 'S', 'L', '1'),
        ('q2', '#', '1'): ('q2', '#', '1', 'S', 'L', '0'),
        ('q2', '0', '#'): ('q1', '0', '#', 'L', 'S', '1'),
        ('q2', '1', '#'): ('q2', '1', '#', 'L', 'S', '0'),
        ('q2', '#', '#'): ('q3', '#', '#', 'S', 'S', '1') }
```

Programski kod 4.4. Popis tranzicija

Tijekom stanja q0 provodi se postavljanje glava čitača na određeni položaj nakon kojeg se počinje provoditi zbrajanje koje se odrađuje izmjenom stanja q1 i q2.

Stanje q1 je zaduženo za zbrajanje kada nema jedinice prijenosa, dok se u stanje q2 prelazi kada se pojavi prijenos i u njemu se ostaje sve dok se i ta jedinica prijenosa ne zapiše.

Glavni dio koda koji obavlja posao počinje sa provjerom finalnog stanja, očitavanjem simbola sa vrpci i postavljanjem trenutnog stanja, (Programski kod 4.5.).

```
    while state != 'q3':
        symbol1 = tape1[head1]
        symbol2 = tape2[head2]
        current_state = (state, symbol1, symbol2)
```

Programski kod 4.5. Očitavanje simbola

Nakon očitavanja simbola, na temelju tranzicija, obavlja se zapisivanje novih simbola na vrpce, pomiče se u zadanim smjerovima i novo stanje iz tranzicije postaje trenutno stanje, što možemo vidjeti u programskom kodu 4.6.

```
        if current_state in transitions:
            if state == 'q0':
                new_state, write_symbol1, write_symbol2, move1, move2
= transitions[current_state]
                tape1[head1] = write_symbol1
                tape2[head2] = write_symbol2
            else :
                new_state, write_symbol1, write_symbol2, move1, move2,
write_symbol3 = transitions[current_state]
                tape1[head1] = write_symbol1
```

```
                tape2[head2] = write_symbol2
                tape3[head3] = write_symbol3
                head3 = head3 + 1


            if move1 == 'R': head1 = head1 + 1
            elif move1 == 'L': head1 = head1 - 1
            if move2 == 'R': head2 = head2 + 1
            elif move2 == 'L':head2 = head2 - 1


            symbol1 = tape1[head1]
            symbol2 = tape2[head2]
            state = new_state


            if withSteps == 1:
                step =step + 1
                print("stanje vrpce na koraku ", step)
                print(tape3)
                print("\n")


            if write_symbol1 == '#' and (write_symbol2 == '0' or
write_symbol2 == '1'):
                state = 'q4'
                print("Prvi broj je manji od drugog.")
                print("Oduzimanje kreće ispočetka i rezultat će biti
negativan")
                break
        else:
            print("Nema definirane tranzicije za trenutno stanje:",
current_state)
            break
```

Programski kod 4.6. Glavni dio koda

Komad glavnog dijela koda, prikazan programskim kodom 4.7., omogućuje dodatnu opcionalnu funkcionalnost, a to je ispisivanje rezultatne vrpce tijekom svakog koraka rada.

```
            if withSteps == 1:
                step =step + 1
                print("stanje vrpce na koraku ", step)
                print(tape3)
                print("\n")
```

Programski kod 4.7. Ispis vrpce u svakom koraku

Kada se dođe u finalno stanje tada još ostaje priprema rezultata za ispis, odnosno povratak odgovora na poziv funkcije što nam je zapisano kao programski kod 4.8.

```
    tape3.reverse()
     result = ''.join(tape3).strip('#')
     return result
```

Programski kod 4.8. Obrada rezultata

4.2. Oduzimanje

Kod oduzimanja kod je gotovo isti kao i kod zbrajanja. Razlika se nalazi u tranzicijama koje ustvari i određuju cijelo ponašanje koda. Tranzicije za stanja q1 i q2 možemo vidjeti na slici ispod dok su tranzicije za stanje q0 iste, (Programski kod 4.9.).

```
        ('q1', '0', '0'): ('q1', '0', '0', 'L', 'L', '0'),
        ('q1', '0', '1'): ('q2', '0', '1', 'L', 'L', '1'),
        ('q1', '1', '0'): ('q1', '1', '0', 'L', 'L', '1'),
        ('q1', '1', '1'): ('q1', '1', '1', 'L', 'L', '0'),
        ('q1', '0', '#'): ('q1', '0', '#', 'L', 'S', '0'),
        ('q1', '1', '#'): ('q1', '1', '#', 'L', 'S', '1'),
        ('q1', '#', '#'): ('q3', '#', '#', 'S', 'S', '#'),

        ('q2', '0', '0'): ('q2', '0', '0', 'L', 'L', '1'),
        ('q2', '0', '1'): ('q2', '0', '1', 'L', 'L', '0'),
        ('q2', '1', '0'): ('q1', '1', '0', 'L', 'L', '0'),
        ('q2', '1', '1'): ('q2', '1', '1', 'L', 'L', '1'),
        ('q2', '0', '#'): ('q2', '0', '#', 'L', 'S', '1'),
        ('q2', '1', '#'): ('q1', '1', '#', 'L', 'S', '0'),
        ('q2', '#', '#'): ('q4', '#', '#', 'S', 'S', '#')
```

Programski kod 4.9. Tranzicije oduzimanja

Specifična situacija kod oduzimanja je da se može dobiti negativan rezultat iako su brojevi s kojima radimo pozitivni što nije slučaj kod zbrajanja, množenja i dijeljenja, (Programski kod 4.10.). U slučaju da je prvi upisani broj manji od drugog, stroj će to primijetiti tijekom izvođenja, prestati sa radom i opet pozvati funkciju oduzimanja, ali sa zamijenjenim brojevima te će dodati minus u ispisu rezultata.

```
            if write_symbol1 == '#' and (write_symbol2 == '0' or
write_symbol2 == '1'):
                state = 'q4' # Pomoću ovog stanja pokrećemo novi poziv
funkcije
                print("Prvi broj je manji od drugog.")
                print("Oduzimanje kreće ispočetka i rezultat će biti
negativan")
                break
```

Programski kod 4.10. Stanje q4 - ponovni poziv

U pripremi ispisa nalazi se dodatna provjera za izvanredno stanje q4 i ponovni poziv funkcije, (Programski kod 4.11.).

```
    tape3.reverse()
    result = ''.join(tape3).strip('#')
    if state == 'q4':
        #Prvi broj je manji od drugog pa pozivamo subtraction sa
(num2, num1)
        result ='-'+ subtraction(num2, num1, withSteps)
    return result
```

Programski kod 4.11. Ponovni poziv i priprema rezultata

4.3. Množenje

Množenje se sastoji iz dva dijela. U prvom dijelu sa vrpce 1 izvlači se informacija koliki je prvi broj, a zatim se u drugom dijelu izvršava zbrajanje drugog broja sa samim sobom onoliko puta koliki je prvi broj. Na početku se nalazi inicijalizacija vrpce prvog broja i stanja stroja te tranzicije za određivanje broja ponavljanja zbrajanja, (Programski kod 4.12.).

```
def multiplication(num1, num2):
    tape1 = list('#'+num1+'#')  # Tape for the first number
    transitions = {
        ('q0', '0'): ('q0', '0', 'R'),
        ('q0', '1'): ('q0', '1', 'R'),
        ('q0', '#'): ('q1', '#', 'L'),
        ('q1', '0'): ('q1', '0', 'L'),
        ('q1', '1'): ('q1', '1', 'L'),
```

```
        ('q1', '#'): ('q2', '#', 'S'), }
    state = 'q0'
    head1 = 1  # Head position on tape 1
    counter=0
    i=0
```

Programski kod 4.12. Inicijalizacija množenja

Zatim slijedi središnji dio, odnosno programski kod 4.13., gdje se utvrđuje koliko će se puta zvati zbrajanje drugog broja i prekidi za specifične situacije kada se množi nula ili sa jedinicom.

```
    while state != 'q2':
        symbol1 = tape1[head1]
        current_state = (state, symbol1)

        if current_state in transitions:
            new_state, write_symbol1, move1 =
transitions[current_state]
            tape1[head1] = write_symbol1

            if move1 == 'R': head1 = head1 + 1
            elif move1 == 'L': head1 = head1 - 1
            symbol1 = tape1[head1]

            if state == 'q1':
                if write_symbol1 == '1' : counter = counter + 2**i
                i=i+1
            state = new_state
    if counter == 0:
        result=0
        return result
    if counter == 1:
        result=num2
        return result
    if num2=='0':
        result=0
        return result
```

Programski kod 4.13. Utvrđivanje broja ponavljanja

Zadnji dio množenja je iterativni poziv za zbrajanje i vraćanje rezultata, (Programski kod 4.14.).

```
    result=num2
    while counter!= 1:
        result = addition(result, num2, 0)
        counter= counter-1
    return result
```

Programski kod 4.14. Završni dio množenja

4.4. Dijeljenje

Dijeljenje je implementirano kombinacijom zbrajanja i oduzimanja. Tijekom svake iteracije od prvog broja (djeljenik), pomoću funkcije oduzimanja, oduzima se drugi broj (djelitelj) i pomoću funkcije zbrajanja za jedan se povećava brojač (količnik) za svako uspješno oduzimanje. Kada se više ne može oduzimati, a da se ne ode u minus, tada se brojač ispisuje kao rezultat cjelobrojnog dijeljenja, a ono što je ostalo oduzimanjem od djeljenika se ispisuje kao ostatak cjelobrojnog dijeljenja, (Programski kod 4.15.).

```
def division (num1, num2):
    result = 0
    i=0
    brojac='0'
    while i != 1:
        result = subtraction(num1, num2,0)
        if  result.startswith('-'):
            i=1
            result='rezultat dijeljenja iznosi', brojac, 'sa
ostatkom', num1
        else:
            num1 =result
            brojac=addition(brojac,'1',0)
    return result
```

Programski kod 4.15. Kod dijeljenja

4.5. Poziv funkcija

Svi pozivi funkcija objedinjeni su u funkciji TuringMachineCalculator. Funkcija prima četiri argumenta, a to su dva binarna broja kao num1 i num2, simbol željene operacije kao operation i argument withSteps za koji se šalje nula ili jedinica. Na temelju simbola operacije funkcija TuringMachineCalculator odabire traženu operaciju i na kraju ispisuje dobiveni rezultat, (Programski kod 4.16.).

```
# Argument 'withSteps' nam nudi mogućnost ispisa rezultatne
# vrpce kroz svaki korak za operacije zbrajanja i oduzimanja.

# Ako želimo tu opciju, u pozivu funkcije na mjesto argumenta
# 'withSteps' trebamo poslati '1'.
def TuringMachineCalculator(num1, num2, operation, withSteps):
    if operation == '+':
```

```
        result = addition(num1, num2, withSteps)
    elif operation == '-':
        result = subtraction(num1, num2, withSteps)
    elif operation == '*':
        result = multiplication(num1, num2)
    elif operation == '/':
        result = division(num1, num2)
    else:
        print("Nepoznata operacija")
    print("Rezultat operacije:", result)
```

Programski kod 4.16. TuringMachineCalculator funkcija

Na posljetku, sve što treba napraviti je pokrenuti program, nakon čega će program zatražiti unos redom: prvog binarnog broja, drugog binarnog broja, jednog od četiri ponuđena simbola željene operacije (+, -, *, /) i unos jedinice za ispis koraka ili nule u suprotnom. Nakon unosa sva četiri podatka, program dalje sam odrađuje posao i ispisuje rezultat, odnosno korake i rezultat, (Programski kod 4.17.).

```
# Test the Turing machine calculator
num1 = input("Unesite prvi binarni broj: ")
num2 = input("Unesite drugi binarni broj: ")
operation = input("Unesite željenu operaciju (+, -, *, /): ")
withSteps = int(input("Želite li prikaz koraka? (1 za da, 0 za ne): "))
TuringMachineCalculator(num1, num2, operation, withSteps)
```

Programski kod 4.17. Poziv glavne funkcije programa

Nakon uspješnog pokretanja i izvršavanja programa, terminal izgleda ovako:

```
Unesite prvi broj u binarnom obliku: 11
Unesite drugi broj u binarnom obliku: 10
Unesite željenu operaciju (+, -, *, /): +
Želite li prikaz koraka? (1 za da, 0 za ne): 0
Rezultat operacije: 101
PS C:\Users\pekic>
```

Programski kod 4.18. Ispis rezultata

Prva četiri reda služe za unos podataka. Ispisuju se jedan po jedan, za unos svakog podatka posebno i na kraju se ispisuje rezultat. Potvrda unosa svakog podatka odrađuje se pritiskom tipke Enter na tipkovnici, (Programski kod 4.18.).

Ukoliko se unese jedinica za prikaz koraka, dobiva se nešto duži ispis sa ispisanom rezultatnom vrpcom kroz korake, (Programski kod 4.19.).

```
Unesite prvi binarni broj: 11
Unesite drugi binarni broj: 10
Unesite željenu operaciju (+, -, *, /): +
Želite li prikaz koraka? (1 za da, 0 za ne): 1
Stanje vrpce na koraku  1
['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#']
Stanje vrpce na koraku  2
['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#']
Stanje vrpce na koraku  3
['#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#']
Stanje vrpce na koraku  4
['1', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#']
Stanje vrpce na koraku  5
['1', '0', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#']
Stanje vrpce na koraku  6
['1', '0', '1', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#', '#',
'#', '#', '#', '#', '#', '#', '#']
Rezultat operacije: 101
PS C:\Users\pekic>
```

Programski kod 4.19. Ispis rezultata sa koracima rada

Korisničko sučelje za ovaj binarni kalkulator bi se moglo sastojati od dva ulazna polja za unos binarnih brojeva, padajućeg izbornika za odabir operacije (zbrajanje, oduzimanje, množenje i dijeljenje ili +, -, *, /), opcije za ispis koraka izračuna, gumba za pokretanje izračuna te mjesta za ispis rezultata i, ako je aktivirana opcija, koraka izračuna. Pri korištenju korisnik unosi binarne brojeve u odgovarajuća polja te bira željenu operaciju iz izbornika. Opcija za ispis koraka

omogućuje detaljan pregled procesa izračuna. Nakon pritiska na gumb za izračun, rezultat se prikazuje ispod ulaznih polja, a koraci izračuna, ako su odabrani, pojavljuju se ispod rezultata. Ovakvo intuitivno sučelje bi olakšalo korisnicima korištenje binarnog kalkulatora i omogućilo praćenje izračuna na jednostavan način.

5. ZAKLJUČAK

Turingov stroj predstavlja temeljni koncept u teorijskoj računalnoj znanosti, čiji je utjecaj na današnje razumijevanje računala i algoritama nemjerljiv. Alan Turing, svojim inovativnim radom iz 1936. godine, postavio je osnove za razvoj modernih računalnih sustava kroz jednostavni, ali moćni apstraktni model – Turingov stroj. Ovaj rad detaljno analizira i prikazuje kako Turingov stroj, iako teoretski i apstraktan, može biti ključan alat za razumijevanje i razvoj kompleksnih računalnih algoritama.

Kroz istraživanje i implementaciju binarnog kalkulatora po principu rada Turingovog stroja, demonstrirano je kako osnovni koncepti mogu biti primijenjeni za stvaranje funkcionalnih računalnih programa. Turingov stroj koristi beskonačnu vrpcu i glavu za čitanje i zapisivanje simbola, simulirajući na taj način osnovne operacije modernih računala. Unatoč svojoj apstraktnosti, ovaj model omogućuje izvedbu bilo kojeg algoritma koji suvremena računala mogu obraditi, što pokazuje njegovu snagu i svestranost.

Primjena Turingovog stroja u stvaranju binarnog kalkulatora za zbrajanje, oduzimanje, množenje i dijeljenje pruža konkretan primjer njegove praktične primjene. Kroz rad se pokazalo kako su osnovni principi rada Turingovog stroja dovoljno moćni da omoguće izvođenje složenih aritmetičkih operacija. Implementacija ovih operacija zahtijevala je pažljivo planiranje stanja, prijelaza i simbola, što dodatno potvrđuje teorijsku moć i praktičnu primjenjivost Turingovog stroja.

Osim tehničkih aspekata, rad naglašava i obrazovnu vrijednost Turingovog stroja. Kroz simulaciju i vizualizaciju njegovog rada, studenti i istraživači mogu bolje razumjeti temeljne koncepte računalne znanosti, uključujući teoriju algoritama, formalne jezike i teoriju složenosti. Turingov stroj služi kao idealan pedagoški alat koji omogućuje duboko razumijevanje osnovnih načela rada računala.

Zaključno, Turingov stroj, iako jednostavan u svojoj konstrukciji, predstavlja nevjerojatno moćan alat za razumijevanje i razvoj računalnih tehnologija. Njegov utjecaj proteže se od teorijskih osnova računalne znanosti do praktičnih aplikacija u obliku programskih jezika, algoritama i softverskih alata. Ovaj rad ne samo da prikazuje kako Turingov stroj može biti primijenjen u stvaranju binarnog kalkulatora, već i potvrđuje njegovu neprocjenjivu vrijednost kao obrazovnog i istraživačkog alata u računalnoj znanosti. Alan Turingov doprinos ostaje temelj na kojem se i dalje gradi napredak u računalnim tehnologijama, potvrđujući bezvremensku relevantnost i značaj njegovih ideja.

6. SAŽETAK

Britanski znanstvenik Alan Turing je 1936. godine opisao mehanički uređaj koji bi imao sve sposobnosti kao i moderno računalo, koji je po njemu i nazvan Turingov stroj. Turingov stroj vrlo je koristan kao ideja u misaonim eksperimentima, odnosno kao princip njegova rada

Turingov stroj sastoji se od „glave“ uređaja i „beskonačne“ vrpce sa koje glava čita i na koju zapisuje simbole. „Glava“ uređaja čita simbol sa vrpce, zatim na temelju tog simbola i unutrašnjeg stanja zapisuje određeni simbol na to mjesto te se zatim pomiče za najviše jedno mjesto desno ili lijevo po vrpci. Kada stroj odradi posao zaustavlja se i ostaje u stanju mirovanja.

Cilj ovog rada bio je ostvariti binarni kalkulator za zbrajanje, oduzimanje, množenje i dijeljenje primjenom načela rada Turingovog stroja. Osim pružanja koda za funkcionalan binarni kalkulator, rad sadrži i temeljito objašnjenje načina na koji funkcionira Turingov stroj, što je prikazano kroz konkretan primjer rada kalkulatora oponašajući Turingov stroj.

Ključne riječi: algoritmi, binarni kalkulator, osnove računarstva, stroj, Turing

7. ABSTRACT

British scientist Alan Turing described a mechanical device in 1936 that would have all the capabilities of a modern computer, which was named the Turing machine after him. The Turing machine is particularly useful as a concept in thought experiments and as a principle of operation.

The Turing machine consists of a "head" and an "infinite" tape from which the head reads and on which it writes symbols. The device's "head" reads a symbol from the tape, and based on that symbol and its internal state, writes a specific symbol at that location, usually moving one position to the right or left on the tape. When the machine completes its task, it stops and remains in a state of rest.

The aim of this paper was to develop a binary calculator for addition, subtraction, multiplication, and division using the principles of the Turing machine. In addition to providing the code for a functional binary calculator, the work includes a thorough explanation of how the Turing machine operates, illustrated through a concrete example of the calculator's operation emulating the Turing machine.

Keywords: algorithms, binary calculator, fundamentals of computing, machine, Turing

8. LITERATURA

[1] L. De Mol, Stanford Encyclopedia of Philosophy : Turing Machines, 24.09.2018. dostupno na: https://plato.stanford.edu/entries/turing-machine/ [24.4.2024]

[2] M. Doko i V. Novaković, Hrvatski matematički elektronički časopis: Izračunljivost i apstraktni strojevi, listopad 2006., dostupno na: http://e.math.hr/old/izracunljivost/index.html [23.4.2024.]

[3] M. Ugarte, Turing Machine, Martin Ugarte , 2017. godina, dostupno na : https://turingmachinesimulator.com [3.5.2024.]

[4] Brojni autori, konstantno uređivanje od kojih je zadnje bilo u ožujku 2024., dostupno na: https://en.wikipedia.org/wiki/Turing_machine [20.4.2024.]

[5] Turing Machine Visualisation, dostupno na: https://turingmachine.io/ [07.05.2024]

[6] Turingov stroj: opis i primjeri Turingovih strojeva, Punto Marinero, 20. 04. 2019., dostupno na: https://hr.puntomarinero.com/turing-machine-description-and-examples [23.4.2024.]

Slike

[1] Sl. 2.2. dostupno na: Turing Machine Model Davey 2012 - Turing machine - Wikipedia

[2] Sl. 2.3. izrađeno na: https://www.turingmachinesimulator.com

[3] Sl. 2.4. izrađeno na: https://turingmachine.io/

[4] Sl. 2.5. izrađeno na: https://play.google.com/store/apps/details?id=ru.hse.tm2d&hl=en_US

[5] Sl. 2.6. dostupno na: https://www.jflap.org/

[6] Sl. 2.7. dostupno na: https://www.calcprofi.com/

9. ŽIVOTOPIS

Autor ovog završnog rada, Stjepan Pekić rođen 16.07.2000. u Bielefeldu, Saveznoj Republici Njemačkoj, student je preddiplomskog studija programskog inženjerstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. U dosadašnjem školovanju završio je Osnovnu školu Bratoljuba Klaića u Bizovcu i III. gimnaziju u Osijeku.