EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2022/421000078424

TROJKOVÁ SÚSTAVA A MOŽNOSTI JEJ VYUŽITIA

Diplomová práca

2022 **Bc. Daniel Ševčík**

EKONOMICKÁ UNIVERZITA V BRATISLAVE

FAKULTA HOSPODÁRSKEJ INFORMATIKY

Evidenčné číslo: 103004/I/2022/421000078424

TROJKOVÁ SÚSTAVA A MOŽNOSTI JEJ VYUŽITIA

Diplomová práca

Študijný program: Informačný manažment

Študijný odbor: Informačný manažment

Školiace pracovisko: Katedra aplikovanej informatiky

Vedúci záverečnej práce: doc. Ing. Jaroslav Kultan, PhD.

Bratislava 2022 **Bc. Daniel Ševčík**

Ekonomická univerzita v Bratislave
Fakulta hospodárskej informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Bc. Daniel Ševčík
Študijný program:	informačný manažment (Jednoodborové štúdium, inžiniersky II. st., denná forma)
Študijný odbor:	ekonómia a manažment
Typ záverečnej práce:	Inžinierska záverečná práca
Jazyk záverečnej práce:	slovenský
Sekundárny jazyk:	anglický

Názov: Trojková sústava a možnosti jej využitia

Anotácia: V bežnom živote sa mnohokrát nevieme rozhodnúť pri riešení úloh či daný výrok má hodnotu áno/nie 1/0. Častokrát vzniká problém, ktorého riešenieje založené na neurčitosti rozhodovania nie-možno-áno. Podobné problémy vznikajú aj v oblasti matematiky, kde boli zavedené kvázihodnoty.

Už Aristoteles v svojich dielach uvažuje pri riešení problémov s pojmom možno. Neskoršie tieto jeho úvahy boli zjednodušené na známe áno/nie. Pojemtrojkovej sústavy sa objavil až omnoho rokov neskôr. V druhej polovici 20. storočia sa objavili prvé elektronické počítače pracujúce v trojkovej sústave.

Vzhľadom na rýchlosť potreby spracovávania čoraz viac údajov, znovu vznikámyšlienka o možnosti tvorby počítača pracujúceho v trojkovej sústave.

Práca bude mať veľký význam aj v oblasti skvalitnenia vyučovacieho procesuv oblasti vyučovania riešenia optimalizačných úloh.

Vedúci:	doc. Ing. Jaroslav Kultan, PhD.
Katedra:	KAI FHI - Katedra aplikovanej informatiky FHI
Vedúci katedry:	Ing. Mgr. Peter Schmidt, PhD.
Dátum zadania:	29.10.2020

Dátum schválenia: 30.10.2020

Ing. Mgr. Peter Schmidt, PhD.
vedúci katedry

Čestné vyhlásenie

Čestne vyhlasujem, že túto diplomovú prácu som vypracoval samostatne s použitím uvedenej literatúry.

V Bratislave dňa .....

Bc. Daniel Ševčík

Poďakovanie

Touto cestou by som sa chcel poďakovať môjmu vedúcemu práce doc. Ing. Jaroslavovi Kultanovi, PhD. za podnetné návrhy, usmerňovacie konzultácie, čas, nápady a rady.

ABSTRAKT

ŠEVČÍK, Daniel: Trojková sústava a možnosti jej využitia. – Ekonomická univerzita v Bratislave. Fakulta hospodárskej informatiky; Katedra aplikovanej informatiky. – Vedúci záverečnej práce: doc. Ing. Jaroslav Kultan, PhD. – Bratislava: FHI EU, 2022, 68 strán.

Cieľom záverečnej práce je analyzovať možnosti trojkovej sústavy pri riešení úloh z rôznych oblastí. Práca je rozdelená do 5 kapitol. Obsahuje 30 obrázkov a 19 tabuliek. Prvá kapitola je venovaná súčasnému stavu riešenej problematiky doma a v zahraničí, histórie trojkovej sústavy, jej porovnania s binárnou sústavou a definovanie využitia trojkovej sústavy vo fuzzy logike. V druhej kapitole je uvedený cieľ práce a čiastkové úlohy. Tretia kapitola pozostáva z dôkladnej analýzy a možnosťami riešenia úloh v trojkovej sústave. Štvrtá kapitola sa týka samotného riešenia úloh pomocou trojkovej sústavy, jej výhody a možnosti riešenia úloh spojených s fuzzy logikou. Piata kapitola sa prostredníctvom diskusie venuje, ako by mohla trojková sústava pomôcť v rozhodovaní.

Kľúčové slová:

Trit, binárnosť, ternárnosť, fuzzy,

ABSTRACT

ŠEVČÍK, Daniel: Triple system and possibilities of its use. – University of Economics in Bratislava. Faculty of Economic Informatics; Department of Applied Informatics. – Supervisor of the thesis : doc. Ing. Jaroslav Kultan, PhD. – Bratislava: FHI EU, 2022, 68 pages.

The aim of the thesis is to analyze the possibilities of the triple system in solving problems from different areas. The work is divided into 5 chapters. It contains 30 pictures and 19 tables. The first chapter is devoted to current state of the issue at home and abroad, the history of the triple system, its comparison with the binary system and the definition of the use of the triple system in fuzzy logic. The second chapter presents the aim of the thesis and subtasks. The third chapter consists of the main tasks and possible solutions in the triple system. The fourth chapter deals with the solution of problems using the triple system, its advantages and possibilities of solving problems related to fuzzy logic. The fifth chapter discusses how the triple system could help in decision-making.

Key Words:

Trit, binary, ternary, fuzzy

Obsah

Zoznam obrázkov a tabuliek ... 9
Úvod ... 11
1. Súčasný stav riešenej problematiky doma a v zahraničí ... 12
1.1 Teoretické základy trojkovej sústavy ... 12
1.1.1 História trojkovej sústavy ... 13
1.1.2 Trojková sústava a jej možnosti ... 17
1.2 Štandardná ternárna logika ... 19
1.2.1 Základné operácie ... 21
1.2.2 Logické operácie pri rozhodovaní ... 23
1.3. Fuzzy logika a trojková sústava ... 26
1.3.1 Využitie v automobile ... 29
1.3.2 ExPro Master ... 29
2. Cieľ ... 32
3. Metódy a nástroje riešenia ... 33
3.1 Historické míľniky počítačov na báze ternárnej sústavy ... 33
3.2 Súčasné trendy vývoja ternárneho počítača ... 34
3.3 Dôkladná analýza trojkovej sústavy a porovnanie s dvojkovou sústavou ... 36
3.3.1 Operácie binárnych čísel ... 37
3.3.2 Operácie ternárnych čísel ... 41
3.3.3 Porovnanie dvojkovej a trojkovej sústavy ... 44
3.4 Možnosti riešenia problémov dvojkovou a trojkovou sústavou zo života ... 46
3.5 Riešenie niektorých logických úloh s využitím ternárnej sústavy ... 49
3.5.1 Fungovanie počítačových obvodov ... 50
3.5.2 Pixely a obrázky ... 51
3.5.3 Aplikácia v medicíne s fuzzy logikou ... 52
3.5.4 Aplikácia pri cenách s fuzzy logikou ... 54
4. Riešenie úlohy ... 56
4.1 SWOT analýza trojkovej sústavy ... 56
4.2 Prototyp ternárneho počítača ... 58
4.3 Ternárne kódovanie údajov ... 59
4.4 Porovnanie riešených logických úloh ... 60
4.4.1 Rozhodovací strom ... 60
4.4.2 Rozhodovanie v medicíne ... 61

4.4.3 Ternárna logika a fuzzy prístup 62
4.4.4 Ternárna logika v digitálnych obvodoch 63
5. Diskusia 65
Záver 66
Zoznam použitej literatúry 67

Zoznam obrázkov a tabuliek

Obrázok 1: Fowlerov stroj 14

Obrázok 2: Setun 15

Obrázok 3: Setun 70 17

Obrázok 4: Porovnanie obvodov 22

Obrázok 5: Rýchlosť vozidla 28

Obrázok 6: Schematický diagram ternárneho procesora od spoločnosti Intel 35

Obrázok 7: Binárne sčítavanie 38

Obrázok 8: Binárne odčítavanie 38

Obrázok 9: Binárne odčítavanie 39

Obrázok 10: Binárne násobenie 39

Obrázok 11: Binárne delenie 40

Obrázok 12: Ternárne nezáporné čísla 41

Obrázok 13: Ternárne záporné čísla 42

Obrázok 14: Sčítavanie ternárnych číslic 42

Obrázok 15: Násobenie ternárnych číslic 43

Obrázok 16: Násobenie ternárnych číslic 43

Obrázok 17: Program na delenie ternárnych číslic 44

Obrázok 18: Binárny a ternárny rozhodovací strom 45

Obrázok 19: Rozhodovanie pri parametrickom výbere 47

Obrázok 20: Binárna interpretácia stavov žiaroviek 50

Obrázok 21: Rozhodovanie v binárnej a ternárnej logike 51

Obrázok 22: Práca s RGB kódom 52

Obrázok 23: Funkcia na porovnávanie cien 54

Obrázok 24: Pravdepodobnostná hodnota trojstavového prvku 55

Obrázok 25: SWOT analýza 56

Obrázok 26: Ternárny procesor 58

Obrázok 27: Rozhodovanie v binárnej sústave 60

Obrázok 28: Rozhodovanie v ternárnej sústave 61

Obrázok 29: Hodnoty vstupného a výstupného napätia 64

Obrázok 30: Hodnoty napätia v ternárnej sústave 64

Tabuľka 1: Unárne a základné binárne operácie 18

Tabuľka 2: Ternárne hodnoty 19

Tabuľka 3: Počítanie v ternárnej sústave 20

Tabuľka 4: Porovnanie sústav 22

Tabuľka 5: Logické operácie 23

Tabuľka 6: Čísla v symetrickej, nesymetrickej a doplnkovej forme 36

Tabuľka 7: Sčítavanie ternárnych číslic 42

Tabuľka 8: Efektívnosť kódovania v určitých číselných základniach 45

Tabuľka 9: Porovnanie kódovania symbolov rôznej dĺžky 46

Tabuľka 10: Porovnanie využitia pamäte pre symboly rôznej dĺžky 46

Tabuľka 11: Trojstavové rozhodovanie v medicíne 47

Tabuľka 12: Trojstavové rozhodovanie v automobilizme 48

Tabuľka 13: Trojstavové rozhodovanie vo vzdelávaní 48

Tabuľka 14: Trojstavové rozhodovanie vo vzdelávaní 49

Tabuľka 15: Možná kombinácia liekov pri chrípke 53

Tabuľka 16: Zisk pamäte v ternárnej sústave oproti binárnej sústave 59

Tabuľka 17: Porovnanie riešenia v oblasti medicíny 61

Tabuľka 18: Riešenie príkladu vo fuzzy logike 62

Tabuľka 19: Riešenie príkladu v ternárnej logike 62

Úvod

Najčastejšie používaným číselným systémom v počítačoch a ďalších spínacích obvodoch je dobre známy binárny systém. Je to tak kvôli skutočnosti, že väčšina bežne používaných fyzických zariadení vykazuje dva ľahko rozlíšiteľné stavy. Myšlienku použitia trojkového systému na výpočty prvýkrát vyslovil v 13. storočí taliansky matematik Fibonacci [1].

Dvojhodnotová matematická logika, ktorá vládne všade vo svete počítačovej a inej inteligentnej techniky, podľa tvorcu ternárneho počítača Nikolaja Brusentsova, nezodpovedá zdravému rozumu, pretože vylúči ostatné závery, okrem pravdy a nepravdy. A predsa sa proces ľudského poznávania reality v žiadnom prípade neredukuje na áno alebo nie. Preto Brusentsov tvrdí, že na to, aby sa počítač stal inteligentným, musí byť trojstavový [1].

Trojhodnotová logika sa od dvojhodnotovej odlišuje tým, že okrem významov pravda a nepravda existuje ešte tretia, ktorá sa chápe ako nedefinovaná, neutrálna alebo možná. Zároveň je zachovaná kompatibilita s dvojhodnotovou logikou, čo znamená, že logické operácie so známymi hodnotami dávajú rovnaké výsledky .

Logike pracujúcej s tromi významami prirodzene zodpovedá trojčlenná symetrická číselná sústava. V tejto sústave sa používajú čísla -1, 0, 1. Taktiež sa môžeme stretnúť s označením -, 0, + [2].

Zo všetkých pozičných číselných sústav je trojčlenná najhospodárnejšia a to tým, že dokáže zapísať viac čísel, ako v akomkoľvek inom systéme, pričom sa použije rovnaký počet znakov. Napríklad v desiatkovej sústave na vyjadrenie čísel od 0 do 999 budeme potrebovať vyjadrenie pomocou tridsiatich číslic, v binárnom systéme možno rovnakých tridsať číslic použiť na kódovanie čísel v rozsahu od 0 do 32 767 a v ternárnom systéme od 0 do 59 048. Najekonomickejší číselný systém by bol základ rovný Eulerovmu číslu a číslo 3 je k nemu najbližšie celé číslo [2].

Nedá sa však povedať, že ternárny princíp v počítačovom inžinierstve je beznádejným anachronizmom. V poslednom desaťročí vznikla potreba hľadania nových počítačových technológií a niektoré z týchto technológií sa nachádzajú v oblasti trojice [2].

1. Súčasný stav riešenej problematiky doma a v zahraničí

Súčasne počítačové systémy pracujú s dvojhodnotovou základnou jednotkou bit (1/0). V živote sa veľmi často stáva, že nevieme presne povedať, či je daný výrok pravdivý alebo nepravdivý, či danú prácu urobíme alebo nie, či môžeme použiť ten alebo iný nástroj. Veľmi často odpovedáme slovom možno, zavoláme si ešte, ešte neviem a podobne. A práve táto možnosť pri využívaní dvojstavovej logiky chýba. Mnohé otázky týkajúce sa nejednoznačnosti sa mnohí matematici, alebo pracovníci iných odvetví, snažia riešiť inými spôsobmi. Jedným z nich je aj fuzzy matematika. Práve vyjadrovanie hodnôt so zadanou pravdepodobnosťou je cesta k zavedeniu trojstavovej logiky. Ak si povieme, že odpoveď „nie“ má pravdepodobnosť viac ako 0.9, tak predpokladáme, že sa jedná o hodnotu -1. Ak si povieme, že odpoveď „áno“ má pravdepodobnosť viac ako 0.9, tak predpokladáme, že sa jedná o hodnotu +1. V ostatných prípadoch si nie sme istí so svojou odpoveďou a preto môžeme zvoliť hodnotu 0 [3].

Dodatočná hodnota – nejednoznačnosť logiky sa vyskytuje už v prvej ukončenej teórii logiky Aristotela, ktorý medzi potvrdením a odmietnutím vložil tretiu možnosť – možno áno, možno nie. V nasledujúcom období bola logika zjednodušená na dva stavy. Táto forma logiky sa rýchlo rozvíjala a udomácnila. V rôznom čase sa pokúšali rozšíriť túto logiku viacerí vedci, ako napríklad Okkam, Leibnitz, Hegel a niektorí iní vedci, avšak trojkovú logiku rozpracoval až na začiatku dvadsiateho storočia poľský vedec Jan Lukasevič [3].

1.1 Teoretické základy trojkovej sústavy

Každý, kto pozná digitálnu výpočtovú techniku, vie o nulách a jednotkách, ktoré sú stavebnými kameňmi binárneho jazyka. Ale nie všetky počítače sú digitálne a nič nehovorí, že digitálne počítače musia byť binárne. V minulosti sa niekoľko ľudí pokúsilo o ternárnu alternatívu. Louis Howell navrhol programovací jazyk TriINTERCAL pomocou systému číslovania base-3 v roku 1991. Ruskí inovátori postavili niekoľko desiatok strojov base-3 pred viac ako päťdesiatimi rokmi, ale tento číselný systém sa neuchytil v širšom počítačovom svete [4].

V zásade ternárna číselná sústava nemala o nič menšie šance, ako binárna. Ktovie, ktorou cestou vývoja by sa uberal technický pokrok, keby pokusy zvíťazili nad bajtami, ako

by vyzerali moderné smartfóny či GPS navigácie a ako by ternárna číselná sústava ovplyvnila ich výkon.

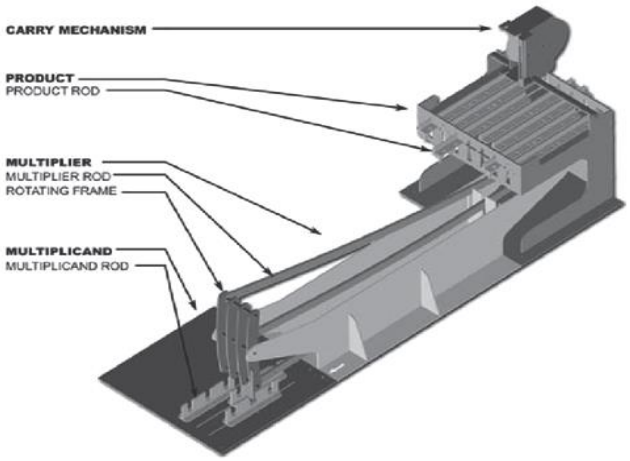
1.1.1 História trojkovej sústavy

Slovo ternárny predstavuje číslo tri, čiže všeobecne, niečo čo je ternárne je zložené z troch častí, respektíve divízií. Ternárna forma v hudbe je pesnička, ktorá pozostáva z troch sekcií. V matematike ternárnosť sú čísla so základňou čísla tri. Niekto preferuje slovo trinárny, pretože sa to rýmuje so slovom binárny. V tejto časti sa pozrieme na históriu ternárnosti v strojoch.

Fowlerov stroj

Prvý počítačový stroj s ternárnym číselným systémom bol zostrojený, dávno pred sovietskymi dizajnérmi, anglickým vynálezcom Thomasom Fowlerom v roku 1840. Pracoval ako bankový úradník a bol nútený vykonávať zložité výpočty. Na uľahčenie a urýchlenie práce zhotovil tabuľky na počítanie v mocninách dva a tri a neskôr tieto tabuľky vydal vo forme brožúry. Potom išiel ďalej a rozhodol sa úplne zautomatizovať výpočty v tabuľkách, preto zostrojil počítací stroj. Anglicky patentovaný systém v tej dobe bol nedokonalý. Fowlerov predchádzajúci vynález bol s minimálnymi zmenami skopírovaný a patentovaný mnohými bezohľadnými vynálezcami, preto sa v obave, že jeho nápad môže opäť niekto ukradnúť, rozhodol vyrobiť stroj v jedinom exemplári a vyrobený z dreva. Keďže drevo je nespoľahlivý materiál, Fowler musel vyrobiť tento stroj veľmi objemný, asi dva metre na dĺžku, aby bola zabezpečená dostatočná presnosť výpočtov. Ako však sám vynálezca napísal v sprievodnej poznámke, ak by tento stroj mohol byť vyrobený z kovu, nebol by väčší, ako písací stroj. Tento stroj bol jednoduchý, efektívny a využíval inovatívny prístup. Namiesto desiatkovej číselnej sústavy fungoval s triádami, teda s mocninami troch.

Žiaľ pozoruhodný vynález zostal nepovšimnutý, originál stroja sa nezachoval a jeho štruktúra je známa len z diela jeho syna, ktorý napísal životopis svojho otca. [5]



Obrázok 1: Fowlerov stroj [5]

Prvé sovietske skúsenosti

Na praktické využitie trojčlennej číselnej sústavy sa na viac ako sto rokov zabudlo. Ďalší, ktorí sa k tejto myšlienke vrátili, boli inžinieri z Katedry výpočtovej matematiky Fakulty mechaniky a matematiky Moskovskej štátnej univerzity. Všetko sa to začalo v roku 1954, keď katedra mala dostať elektronický počítač M-2, ale nevyšlo to, tak sa rozhodli postaviť si vlastný počítač. Nikolaj Petrovič Brusentsov bol vymenovaný za vedúceho skupiny, ktorá navrhla a vyrobila stroj. Najprv sa chystali vyrobiť binárny počítač, ale neskôr, práve z dôvodu hospodárnosti a jednoduchosti architektúry, dospeli k rozhodnutiu, že bude ternárny, využívajúc prirodzený ternárny symetrický kód, najjednoduchší zo symetrických kódov.

Do konca roka 1958 bola dokončená prvá kópia stroja, ktorá dostala názov Setun, podľa názvu rieky v Moskve. Tento stroj bol relatívne malý pre počítače tejto generácie a zaberal plochu 25-30 metrov štvorcových. Vďaka svojej architektúre bol schopný vykonať dvetisíc až štyritisíc päťsto operácií za sekundu. Na testoch v roku 1960 bol stroj uznaný ako vhodný na masové použitie v dizajnérskych kanceláriách, laboratóriách a univerzitách, po čom nasledovala objednávka na sériovú výrobu stroja Setun v závode matematických strojov v Kazani. Od roku 1961 do roku 1965 bolo vyrobených a prevádzkovaných 50 exemplárov po celej krajine, ale potom bola výroba obmedzená. Uvádza sa, že jedným z možných dôvodov je, že počítač sa ukázal byť príliš lacný na výrobu, a teda pre fabriku nerentabilný. Následne Nikolaj Brusentsov a Jevgenih Zhogolev vyvinuli modernejšiu verziu stroja, ktorá rovnako využívala princípy trojkovej číselnej logiky, Setun-70. Tento

stroj sa nikdy nedostal do sériovej výroby a jediný prototyp pracoval na Moskovskej štátnej univerzite až do roku 1987.

Obrázok 2: Setun [1]

Setun

Návrh tohto malého digitálneho stroja bol iniciovaný členom Akadémie vied, menom Sergei Lvovich Sobolev, v roku 1956. Predpokladalo sa, že vytvorí malý, lacný počítač, ktorý bude jednoduchý na použitie pre školy, výskumné laboratóriá, dizajnérske kancelárie a na kontrolu výroby. Pre tento cieľ bola v počítačovom centre univerzity vytvorená skupinka mladých inžinierov. Zorganizoval sa spoločný seminár pre programátorov a inžinierov, ktorého stálymi účastníkmi boli S. L. Sobolev, K. A. Semendjev, M. R. Shura-Bura a I. S. Berezin. Skúmali sa problémy optimalizácie počítačovej architektúry a technickej realizácie a diskutovalo sa o variantoch budúceho počítača. Vzhľadom na nízku spoľahlivosť počítačových prvkov na vákuových trubiciach a nedostupnosť tranzistorov boli navrhnuté rýchle prvky na miniatúrnych feritových jadrách a polovodičových diódach. Tieto prvky fungujú ako transformátor prúdu a boli účinným základom pre implementáciu prahovej logiky a najmä jej ternárnej verzie. Ternárne logické prvky, v porovnaní s binárnymi, poskytujú väčšiu rýchlosť a spoľahlivosť, vyžadujú menej vybavenia a výkonu a to boli dôvody, prečo sa začali zaujímať o návrh ternárneho počítača [1].

Jednoduchosť, hospodárnosť a elegancia počítačovej architektúry sú priamym a prakticky veľmi dôležitým dôsledkom ternárnej architektúry, presnejšie reprezentácie údajov symetrickým kódom (-1, 0, 1). Skúsenosti s tvorbou, programovaním a aplikáciou počítača Setun jednoznačne potvrdili významné preferencie ternárnosti, na rozdiel od binárnosti. Napriek tomu, že dizajnéri boli mladí a ich skupina bola malá, Setun bol

pripravený na použitie už v decembri 1958, to bolo za dva roky od začiatku vývoja. Setun fungoval správne a dokázal spúšťať už existujúce aplikácie. V roku 1960 to bolo dostatočné množstvo aplikácií a bolo možné začať oficiálne testovať Setun. Toto testovanie prebehlo v apríli 1960 veľmi úspešne. Počítač ukázal, že je nezvyčajne spoľahlivý, stabilný pri prevádzke a pomerne jednoduchý na výrobu, vhodný pre širokú škálu aplikácií a odporúčaný na výrobu [1].

Bohužiaľ, úradníci počítačovej výroby v ZSSR mali negatívny postoj k neplánovanému a nezvyčajnému plodu univerzitnej fantázie. Namiesto podpory inovácií a získania možného zisku sa ho pokúsili natrvalo zničiť. Na Setun bolo vytvorených veľa objednávok, avšak kvôli minimálnej podpore bola výrobná kapacita obmedzená maximálne na 15 počítačov. Zrušila sa aj plánovaná výroba v Československu. V roku 1965 bola výroba zastavená napriek neuspokojeným žiadostiam, pretože bol nahradený binárnym počítačom s rovnakým výkonom, ale viac ako 2,5-krát drahším [1].

Celkovo bolo vyrobených 50 počítačov Setun (vrátane vzoriek), z toho tridsať bolo nainštalovaných na univerzitách a vysokých školách a zvyšok vo výskumných laboratóriách po celej krajine [1].

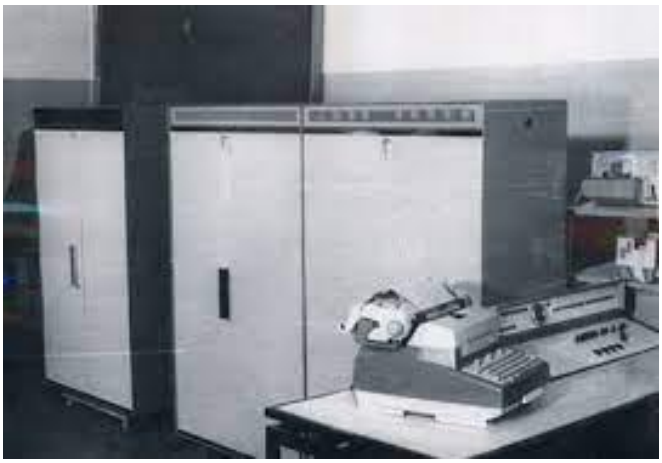
Setun 70

Na základe pozitívnych skúsenosti so Setunom bola navrhnutá, v programovacom jazyku podobnom Angol X, architektúra ternárneho počítača. Tento počítač, s názvom Setun 70, bol predstavený v roku 1970. V tomto počítači boli unikátnosti ternárnosti navrhnuté s väčším porozumením a úplnosťou, čo znamená, že bol stanovený ternárny formát na kódovanie symbolov, s názvom tryte, pozostávajúci zo šiestich tritov (čo predstavuje približne 9,5 bitov), čo umožňovalo väčšiu variáciu dĺžky operandu -1, 2 a 3 trytes a to znamenalo, že dĺžka výsledku mohla byť až šesť tritov. Setun 70 bol však posledným zástupcom počítačov s ternárnou logikou, pretože po ňom bol výskum zastavený [1].

Po Setune bolo vykonaných niekoľko experimentálnych projektov nadšencov, ale boli to buď veľmi nedokonalé stroje oveľa horšej výkonnosti, ako u binárnych strojov, alebo to boli dokonca len softvérové emulácie na binárnom hardvéri. Hlavným dôvodom je, že použitie ternárnych prvkov v počítačoch zatiaľ neposkytuje žiadne výrazné výhody oproti binárnym prvkom, ktoré sú sériovo vyrábané, jednoduchšie a lacnejšie z hľadiska nákladov. Aj keby bol teraz ternárny počítač zostavený, lacný a mal by porovnateľné vlastnosti s binárnymi počítačmi, musel by byť s nimi plne kompatibilný, čo predstavuje veľkú

prekážku. Už vývojári Setun-70 čelili potrebe zabezpečiť kompatibilitu. Aby si mohli vymieňať informácie s inými univerzitnými strojmi, museli pridať možnosť čítať binárne dáta a konvertovať ich [1].

Jednou z aktuálnych oblastí výskumu je hľadanie alternatívnych spôsobov zvýšenia výkonu procesora. Každých dvadsaťštyri mesiacov sa počet tranzistorov v procesore približne zdvojnásobí. Tento trend je známy ako Moorov zákon a nemôže trvať večne. Rozsah prvkov a spojení sa dá merať v nanometroch a čoskoro budú vývojári čeliť mnohým technickým ťažkostiam a od istého momentu bude lacnejšie hľadať alternatívne spôsoby, ako urobiť procesory výkonnejšími, než pokračovať v pretekoch o nanometre [1].



Obrázok 3: Setun 70 [1]

1.1.2 Trojková sústava a jej možnosti

Trojková, respektíve ternárna sústava, je omnoho bližšie k základnému ľudskému mysleniu, ako dvojhodnotová sústava. Len málokedy si vieme predstaviť jednoznačnú odpoveď na mnohé otázky. Práve sústava, ktorá umožní odpovedať formou blízkou reálnemu životu, môže napomôcť aj k lepšiemu a efektívnejšiemu riešeniu určitých problémov. Pri podrobnom porovnávaní trojkového systému s dvojkovým systémom je vidieť, že trojkový systém má oveľa širšie možnosti. Pri deväť znakovom dvojkovom kóde máme k dispozícií čísel 2^9, čo predstavuje 512 čísel, pričom trojkový systém ponúka 3^9, čo predstavuje 19 863 čísel a to je vyše tridsaťosem krát viac [3].

Okrem rozšírenia množstva čísel, ktoré môžeme vyjadriť pomocou rovnakého počtu znakov, využívanie daného systému má aj iné výhody. Veľmi výhodné je napríklad používanie symetrického trojkového kódu v tvare -1, 0, 1. Takáto forma zápisu nepožaduje

prevod čísel na záporné čísla pri operáciách odčítania a tým sa veľmi zrýchľuje aj matematické spracovanie čísel [3].

Základné operácie s tritom

V literatúre existuje veľký počet operácií s tritom, avšak mnohí autori sa odlišujú formou zápisu a hodnotou výsledku danej operácie. Jedným zo základných problémov je spôsob označovania hodnoty tritu. Jedná sa o takzvaný symetrický a nesymetrický kód, kde jednotlivé hodnoty stavov sú -1, 0, 1 alebo 0, 1, 2. Pri riešení mnohých ekonomických úloh je potrebné brať do úvahy nejednoznačnosť operácií s tritom. Pre lepšiu názornosť a jednoznačnosť pri riešení vybraných úloh je potrebné definovať základné operácie s tritom:

1. ***Unárne operácie***

   Predstavujú hodnotu daného tritu po realizácií určitej operácie. Rozdielom, oproti štandardnému binárnemu systému, je existencia kladného alebo záporného posunu, kde hodnota sa môže zmeniť smerom k vyššej alebo nižšej hodnote. Taktiež predpokladáme, že negáciou kladného stavu získame záporný stav a opačne. A negáciou neurčitého stavu získame stav neurčitý.

   Tabuľka 1: Unárne a základné binárne operácie [Vlastné spracovanie]

Hodnota veličiny	Negácia	Kladný posun	Záporný posun	Poznámka
-1	1	0	-1	
0	0	1	-1	
1	-1	1	0	

A	B	AvB	A^B
-1	-1	-1	-1
-1	0	-1	0
-1	1	0	0
0	-1	-1	0
0	0	0	0
0	1	1	0
1	-1	0	0
1	0	1	0
1	1	1	1

2. **Binárne operácie**

   Tento systém operácií rieši otázku výsledku vzájomného pôsobenia výrokov s využitím trojhodnotovej logiky. Pri riešení operácií medzi dvoma výrokmi vzniká problém s využitím stavu možno. Pri klasickom stave -1 a 1 platia isté pravidlá ako v dvojkovej sústave. Pri riešení vzťahov so stavom "možno" niekedy vzniká problém správnej interpretácie nejednoznačnosti výroku. Na základe mnohých skúseností je možné jednotlivé tabuľky modifikovať [3].

Trojstavový systém a kvantové počítače

Kvantový počítač je počítačové zariadenie, ktoré pracuje na základe kvantovej mechaniky. Kvôli obrovskej rýchlosti systémov umožňuje kvantový počítač dešifrovať správy pomocou populárneho asymetrického kryptografického algoritmu RSA. Doteraz je tento algoritmus považovaný za relatívne spoľahlivý, pretože iný efektívny spôsob rozkladu čísel na prvotné faktory pre klasický počítač je v súčasnosti neznámy. Ak by ste napríklad chceli získať prístup k nejakej kreditnej karte, ku ktorej oficiálne prístup nemáte a nemáte k nej potrebné údaje, museli by ste rozdeliť čísla na dva jednoduché multiplikátory, ktoré majú niekoľko stoviek čísel. Dokonca aj najrýchlejším moderným počítačom by toto trvalo stokrát viac, ako vek vesmíru. Vďaka Shoreho algoritmu sa táto úloha stáva pre kvantový počítač uskutočniteľnou.

Je zrejmé, že ternárny symetrický systém je v niektorých ukazovateľoch lepší ako binárny systém. Počítače môžu byť stavané na takzvaných coutrites, čo sú vlastne trity s tromi možnými stavmi použité v kvantovom počítači. S príchodom kvantových počítačov trity získali nový záujem spoločností, ale zatiaľ je to len vo fáze testovania a vyvíjania [3].

1.2 Štandardná ternárna logika

Logika amerického matematika a informatika, menom Stephen Cole Kleene, rozširuje konvenčnú logiku pravda a nepravda o tretiu hodnotu, neznáma. Rôzni autori navrhli priradenie rôznych číselných ekvivalentov k týmto hodnotám, ako napríklad:

Tabuľka 2: Ternárne hodnoty [Vlastné spracovanie]

Pravdivostné hodnoty	Číselné	Krátka forma
pravda	1	+
neznáma	0	0
nepravda	-1	-

Hlavnou motiváciou pre výber týchto číselných hodnôt je, že spĺňa požiadavku, že pravda je väčšia ako nepravda a umožňuje rovnocennosť medzi logickou negáciou a celočíselnou negáciou.

Ternárna logika sa zaoberá tromi diskrétnymi stavmi, ale samotné ternárne číslice, podľa Jeffa Connellyho, môžu byť definované rôznymi spôsobmi:

- Nevyvážená (nesymetrická) ternárnosť – {0, 1, 2}
- Frakčná nevyvážená ternárnosť – {0, ½, 1}
- Vyvážená (symetrická) ternárnosť – {-1, 0, 1}
- Logika neznámeho stavu – {F, ?, T}
- Binárne kódovaná ternárnosť – {T, F, T}

Vyvážená ternárnosť

Ternárna výpočtová technika bola vo väčšine prípadov aplikovaná pomocou vyváženej ternárnosti, -1, 0 , 1. Zápornú hodnotu každej vyváženej ternárnej číslice je možné nahradiť znamienkom plus, mínus a naopak. Vyvážená ternárnosť dokáže vyjadriť záporné hodnoty rovnako ľahko ako kladné.

Nevyvážená ternárnosť

Ternárna výpočtová technika môže byť taktiež navrhnutá na báze nevyváženej ternárnosti pomocou čísel 0, 1 a 2. Pôvodné 0 a 1 sú opísané, ako bežné binárne hodnoty a číslo 2 sa používa ako takzvaný únikový prúd v prípade nejasnosti logiky.

Na to, aby sme pochopili, ako sa počíta v ternárnej sústave, nám pomôže nasledujúca tabuľka:

Tabuľka 3: Počítanie v ternárnej sústave [Vlastné spracovanie]

Číslo	Vyjadrenie v trojkovej sústave
0	0
1	1
2	2
3	10
4	11
5	12
6	**20**
7	21
8	22
9	**100**
10	101

Princíp spočíva v tom, že prvé tri čísla sa rátajú na pravej strane od nuly po dvojku, ako je vidno v tabuľke. Keďže ternárna sústava obsahuje trity, tak namiesto toho, aby sme použili nasledujúce číslo po dvojke, pridáme na ľavú stranu +1, ako môžeme vidieť v treťom riadku. Pri čísle osem vidíme, že na ľavú stranu nemôžeme pridať +1, lebo by z toho vzniklo tri. V tom prípade na ľavú stranu pridáme tretiu cifru 1 a z dvojok sa stanú opäť nuly [6].

1.2.1 Základné operácie

Ternárna logika a ternárna číselná sústava môžu byť aplikovateľné napríklad v strojárstve a informatike, pretože je možné pomocou nej vytvoriť digitálne systémy pomocou pamäťových jednotiek s tromi odlišnými stavmi (flip-flap-flop) a pomocou logických obvodov s tromi odlišnými vstupnými a výstupnými úrovňami, ako je kladné, záporné a nulové napätie [7].

Ternárna logika môže byť vnímaná ako zovšeobecnenie binárnej logiky, pričom operátory pracujú presne ako známy boolean operátor v oblasti pravdivých a falošných hodnôt, zatiaľ čo dúfajme, že stále dávajú nejaký zmysel v prítomnosti nulových hodnôt. Rôzni autori použili rozličné hodnoty pravdy pre rôzne koncepty [7].

V Kleeneho ternárnej logike je nula hodnota, ktorá môže byť pravdivá alebo nepravdivá. Ak sa nulové hodnoty podieľajú na logickom výraze, výsledok sa môže vypočítať pomocou boolean logiky nahradením všetkých výskytov nuly nezávisle pravdou a nepravdou. Ak je výsledok vždy rovnaký, táto hodnota je tiež hodnotou logického výrazu ternárnosti. V opačnom prípade je jeho hodnota nulová [7].

Binárna logika má $2^2 = 4$ unárne operátory, pridanie tretej hodnoty v ternárnej logike vedie k celkom $3^3 = 27$ rôznych operátorov pri jednej vstupnej hodnote. Podobne, keď binárna logika má 16 rôznych binárnych operátorov (operátory s dvoma vstupmi), ternárna logika má 19 683 takýchto operátorov. Tam, kde môžeme ľahko pomenovať významnú časť binárnych operátorov (not, and, or, nand, nor, exclusive or, equivalence, implication), je nerozumné pokúšať sa pomenovať všetky, okrem malého zlomku možných ternárnych operátorov [7].

Ternárna logika umožňuje kompaktnejšie a efektívnejšie kódovanie informácií, ako ekvivalentná binárna logika. Argument je taký, že ak predpokladáme, že digitálny obvod má

N možných vstupných kombinácií, potom binárny obvod potrebuje $\log_2 N$ vstupov a ternárny obvod potrebuje $\log_3 N$ vstupov [7].

$$\frac{\text{Počet vstupov pre ternárny obvod}}{\text{Počet vstupov pre binárny obvod}} = \frac{\log_3 N}{\log_2 N} = \frac{\log_2 N / \log_2 3}{\log_2 N} = \frac{1}{\log_2 3} = 0{,}63$$

Obrázok 4: Porovnanie obvodov [Vlastné spracovanie]

Preto ternárna realizácia danej binárnej logickej funkcie by mala vyžadovať 0,63 násobok vstupov, ako zodpovedajúca binárna realizácia.

Hoci ternárne logické obvody by mali vyžadovať menej vstupných riadkov ako ekvivalentné binárne obvody, ternárne logické obvody v súčasnosti nie sú praktickou voľbou, pretože:

- Technologická implementácia hardvéru vytvoreného na ternárnej logike je stále v teoretickej, simulačnej a laboratórnej testovacej úrovni;
- Predstavujúce tri úrovne ternárnej logiky (0, 1, 2 alebo -1, 0, 1) s použitím úrovní napätia existujúcej technológie ešte nie je účinne definované;
- Nevyvíja sa žiadny výpočtový model a programovací jazyk, ale simulácia výsledkov implementácie ternárneho obvodu pomocou doplnkového polovodiča oxidu kovu, rezonančnej tunelovacej diódy a uhlíkových nanotrubkových technológií dokazuje, že ternárna logika môže by voľbou pre budúcu výpočtovú techniku [7].

Porovnanie ternárnej sústavy s desatinnou sústavou

Jedná ternárna číslica sa rovná jednému tritu, ktorý v tomto porovnávaní môže mať hodnoty 0, 1, 2. V nasledujúcej tabuľke sú niektoré spoločné hodnoty v desatinnej (decimal), deviatkovej (nonary) a ternárnej sústave (ternary):

Tabuľka 4: Porovnanie sústav [8]

Decimal	Ternary	Nonary	Decimal	Ternary	Nonary
0 =	0 =	0	1 =	1 =	1
3 =	10 =	3	2 =	2 =	2
9 =	100 =	10	4 =	11 =	4
27 =	1000 =	30	8 =	22 =	8
81 =	10000 =	100	16 =	121 =	17
243 =	100000 =	300	32 =	1012 =	35
729 =	1000000 =	1000	64 =	2101 =	71
Decimal	**Ternary**	**Nonary**	128 =	11202 =	152
1 =	1 =	1	256 =	100111 =	314
10 =	101 =	11	512 =	200222 =	628
100 =	10201 =	121	1024 =	1101221 =	1412
1000 =	1101001 =	1331			

Rovnako, ako je pre ľudí jednoduchšie používať osmičkovú (base-8) alebo šestnástkovú (base-16) sústavu namiesto binárnej, je ľahšie používať nonárnu (base-9) namiesto ternárnej (base-3) [8].

Päť tritov dokáže vyjadriť slovo len o trochu menšie, ako 8 bitov. To znamená, že počítať s desiatimi tritmi by bol porovnateľný so 16 bitovým mini počítačom a dvadsať tritový počítač by bol porovnateľný s 32 bitovým počítačom. Na druhej strane, väčší zmysel, oproti tritovému počítaču (base 3), by dával počítač založený na 9-tritových alebo 27-tritových slovách. 27-tritové slovo by dokázalo reprezentovať 7 625 597 484 987 hodnôt, čo je približne podobne, ako by vedelo reprezentovať 44 bitové slovo. Ternárny ekvivalent byte sa nazýva tryte [8].

1.2.2 Logické operácie pri rozhodovaní

Logické operácie pri rozhodovaní nám môžu priniesť tri rôzne výsledky, ako pravda (true), nepravda (false) alebo neznámy stav (unknown).

Tabuľka 5: Logické operácie [Vlastné spracovanie]

A	*B*	*A* OR *B*	*A* AND *B*	NOT *A*
True	True	True	True	False
True	Unknown	True	Unknown	
True	False	True	False	
Unknown	True	True	Unknown	Unknown
Unknown	Unknown	Unknown	Unknown	
Unknown	False	Unknown	False	
False	True	True	False	True
False	Unknown	Unknown	False	
False	False	False	False	

Táto tabuľka znázorňuje výsledky niektorých logických operácií pre systém s pravdou, nepravdou a neznámou. V tejto tabuľke kľudne môžeme stav neznámy metaforicky považovať za zapečatenú krabicu obsahujúcu buď jednoznačnú pravdu alebo jednoznačnú nepravdu. Znalosť, či neznámy stav tajne reprezentuje pravdu alebo nepravdu je ale zatiaľ neznáma. Niektoré logické operácie však môžu priniesť jednoznačný výsledok aj keď zahŕňajú neznámy stav. Napríklad ak vieme, že TRUE alebo TRUE sa rovná TRUE,

TRUE alebo FALSE sa rovná tiež TRUE, tak možno vyvodiť, že TRUE alebo UNKNOWN sa rovná tiež TRUE [9].

Ternárna logika v databázových aplikáciách

SQL implementuje ternárnu logiku ako prostriedok na spracovanie obsahu poľa NULL. SQL používa hodnotu NULL na reprezentáciu chýbajúcich údajov v databáze. Ak pole neobsahuje definovanú hodnotu, SQL predpokladá, že to znamená, že skutočná hodnota existuje, ale táto hodnota sa v databáze momentálne nezaznamenáva. Chýbajúca hodnota nie je rovnaká, ako číselná hodnota nula alebo hodnota reťazca nulovej dĺžky. Porovnanie čohokoľvek s NULL, dokonca aj s inou hodnotou NULL, má za následok neznámy stav. Napríklad výraz *"City = ´Paris´"* sa vyhodnotí ako FALSE, ak predpokladáme že tam očakával hodnotu Chicago. Inými slovami v SQL, nedefinované pole môže reprezentovať akúkoľvek možnú hodnotu, chýbajúce mesto môže a nemusí reprezentovať Paris [9].

Pomocou ternárnej logiky, SQL môže vyriešiť stav neznámej pomocou vyhodnotenia výrazov boolean. Výraz *"City = ´Paris´ OR Balance < 0.0"* sa vyhodnotí ako TRUE pre každý záznam, ktorý obsahuje záporné číslo alebo slovo Paris. Výraz sa vyhodnotí ako FALSE v prípade, ak záznam neobsahuje slovo Paris, ani nezáporné číslo. V ostatných prípadoch sa záznam zmení na neznámy, pretože neobsahuje ani hodnotu Paris a ani hodnotu čísla [9].

V SQL Data Manipulation Language, pravdivý stav TRUE pre nejaký výraz vyvolá nasledujúcu akciu v riadku, zatiaľ čo stav UNKNOWN alebo FALSE nevyvolá žiadnu akciu. Takto je implementovaná v jazyku SQL ternárna logika, aj keď pre používateľa sa správa ako binárna [9].

Ako sme si spomenuli už predtým, SQL používa logiku s tromi hodnotami. Okrem true a false môže byť výsledok logických výrazov neznámy. Trojhodnotová logika v SQL je dôsledkom podpory null na označenie chýbajúcich údajov. Ak hodnota ovplyvňuje výsledok logického výrazu, výsledok nie je pravdivý ani nepravdivý, ale neznámy, respektíve null. Trojhodnotová logika je neoddeliteľnou súčasťou jadra SQL a obsahuje ju takmer každá databáza. Hodnota v SQL v podstate môže byť akákoľvek. Preto nie je možné povedať, či je porovnanie pravdivé, alebo nepravdivé a to je dôvod, prečo prichádza tretia logická hodnota, neznáma. Neznáma môže byť pravdivá alebo nepravdivá, v závislosti od hodnôt. To je dôvod, prečo SQL má predikát na testovanie, či hodnota je alebo nie je odlišná od predikátu pri porovnaní a spracovaní dvoch hodnôt [9].

Ternárna logika v elektronike

Teória digitálnej elektroniky podporuje štyri odlišné logické hodnoty:

- **1, H alebo vysoká** – zvyčajne predstavuje hodnotu TRUE
- **0, L alebo nízka –** zvyčajne predstavuje hodnotu FALSE
- **X, neznáma alebo neviem**
- **Z alebo odpojený vstup**

Hodnota X neexistuje v reálnych obchodoch, je to len zástupný symbol používaný v simulátoroch a na konštrukčné účely. Niektoré simulátory podporujú reprezentáciu hodnoty Z, iné nie. Hodnota Z existuje v reálnych obvodoch, ale len ako výstupný jav [10].

Použitie hodnoty X v simuláciách – mnoho simulačných nástrojov na popis hardvéru, ako je verilog (jazyk pre popis harware v programovacom jazyku C) a VHDL (jazyk pre popis harware v programovacom jazyku Ada a Pascal), podporujú neznámu hodnotu. Neznáma hodnota môže byť výsledkom chyby návrhu, ktorú môže projektant opraviť pred syntézou do skutočného obvodu. Neznáma hodnota tiež predstavuje neinicializované hodnoty pamäte a vstupy obvodov predtým, ako simulácia potvrdí, aká by mala byť skutočná vstupná hodnota [10].

Použitie hodnoty X v digitálnom dizajne – pri navrhovaní digitálneho obvodu môžu byť niektoré podmienky mimo rozsah účelu, ktorý bude obvod vykonávať. Preto sa dizajnér nestará o to, čo sa deje v týchto podmienkach. Okrem toho nastane situácia, že vstupy do obvodu sú maskované inými signálmi, takže hodnota tohto vstupu nemá žiadny vplyv na správanie obvodu. V týchto situáciách je tradičné použiť hodnotu X ako zástupný symbol na označenie neznámej. Akonáhle je konštrukcia obvodu dokončená a skutočný obvod je postavený, hodnoty X už nebudú existovať. Stanú sa určitou hmatateľnou hodnotou 0 alebo 1 [10].

Použitie hodnoty Z – niektoré digitálne zariadenia podporujú formu trojfázovej logiky na svojich výstupoch. Tieto tri stavy sú 0, 1 a Z. Bežne označovaná tristate logika (ochranná známka patri National Semiconductor) zahŕňa obvykle TRUE, FALSE a tretí transparentný stav s vysokou impedanciou, ktorý účinne odpojí logický výstup. To poskytuje efektívny spôsob, ako pripojiť niekoľko logických výstupov k jednému vstupu, kde sú všetky okrem jedného vložené do stavu s vysokou impedanciou, čo umožňuje zostávajúcemu výstupu pracovať v normálnom binárnom systéme. Toto sa bežne používa na pripojenie bánk

počítačovej pamäte a iných podobných zariadení k spoločnej dátovej zbernici. Veľké množstvo zariadení môže komunikovať cez ten istý kanál jednoducho tak, že zabezpečí, že naraz je povolené len jedno [10].

Je dôležité poznamenať, že zatiaľ čo výstupy môžu mať jeden z troch stavov, vstupy môžu rozoznať iba dva. Hoci by sa dalo tvrdiť, že stav s vysokou impedanciou je v skutočnosti neznámy stav, v prevažnej väčšine normálnej elektroniky neexistuje absolútne žiadne ustanovenie na interpretáciu stavu s vysokou impedanciu ako stavu samého o sebe. Vstupy môžu byť detegované iba 0 a 1 [10].

Ternárna logika teda môže byť implementovaná v elektronike, hoci zložitosť návrhu doteraz spôsobila, že je nehospodárna a záujem o výskum bol obmedzený, pretože normálna binárna logika je oveľa lacnejšia na implementáciu a vo väčšine prípadov môže byť ľahko nakonfigurovaná tak, aby napodobňovala ternárne systémy. Existujú však užitočné aplikácie vo fuzzy logike a oprave chýb a bolo vyrobených niekoľko takýchto skutočných logických zariadení [10].

1.3. Fuzzy logika a trojková sústava

Všetky údaje, ktoré počítače prijímajú, spracúvajú, uchovávajú a aj produkujú, sú v konečnom dôsledku iba čísla. Človek sa však málokedy vyjadruje v presných pojmoch – jeho vyjadrovanie je často nepresné. Ale ako povedať stroju, aby vybral faktúru s nie príliš veľkou čiastkou asi v strede februára? Ako naučiť stroj rozpoznať aj pojmy nie príliš veľká a asi v strede [11]?

Niektoré spoločné znaky fuzzy logiky a využitia trojkovej sústavy

Fuzzy logika je súbor nie veľmi prísnych pravidiel, v ktorých na dosiahnutie cieľa môžu byť použité radikálne myšlienky, intuície a skúsenosti odborníkov získané v danej oblasti. Najčastejšie sa používa v expertných systémoch, neurónových sieťach a systémoch umelej inteligencie. Namiesto tradičných hodnôt pravda a nepravda je vo fuzzy logike používaný širší rozsah hodnôt ako je to možné, niekedy, nepamätám a iné. Fuzzy logika je nevyhnutná, keď na nejakú otázku nie je jasná odpoveď alebo vopred všetky možné situácie sú neznáme. Umelá inteligencia a neurónové siete skúšajú modelovať ľudské správanie na

počítači. A keďže ľudia zriedka vidia vonkajší svet iba čierny a biely, je potrebné použiť fuzzy logiku. Práve určitou formou takéhoto modelovania môže byť použitie trojkovej logiky [11].

Využitie tritov v týchto úlohách môže zjednodušiť spôsob rozhodovania, zrýchliť vyhľadávanie optimálnej ceny nejakého výrobku pri výbere lacnejších materiálov, z ktorých je vyrobený alebo sa dá taktiež využiť pri výbere optimálnej cesty.

Jedným z najzrejmejších argumentov v prospech trojstavového systému sú logické úlohy váženia dvoch nákladov. Zvážime dva objekty, A a B, na bežných váhach, ktoré umožnia určiť dva protiklady. Hmotnosť A>B alebo hmotnosť B>A. Ale tiež je možné, že A=B, čiže problém hmotnosti má 3 riešenia a neexistuje žiadne označenie pre túto situáciu v dvojstavovej logike. Podobné tretie rozhodnutie priniesol remízový výsledok futbalového zápasu alebo napríklad stav štátov Švajčiarsko a Fínsko počas konfrontácie medzi NATO a Varšavskou zmluvou.

Ďalším príkladom môže byť pohyb Slnka na oblohe, kde prítomnosť Slnka na oblohe označíme číslom jeden a absenciu slnka číslom 0. Ako označiť stav, keď horizont je už osvetlený jasnými lúčmi, ale solárny disk sa ešte neukázal?

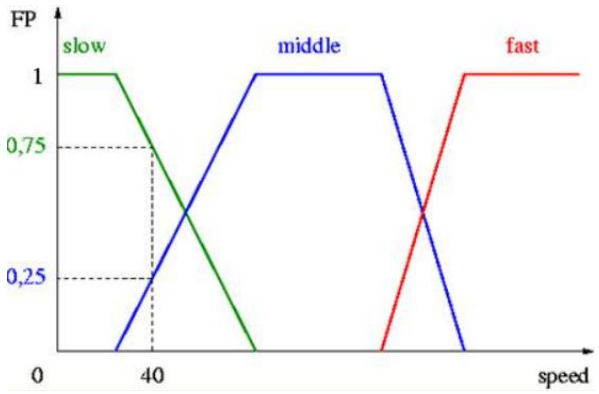
Pre produktívnu prácu ternárneho počítača sú potrebné kvalitné ternárne softvéry, ktoré zatiaľ neexistujú. K dnešnému dňu sa na vývoj binárnych softvérových projektov vynaložilo milióny hodín a miliardy dolárov a znovu ich vytvoriť od začiatku v trojkovej logike by bolo príliš drahé. V opačnom prípade je ternárny počítač matematicky najlepším riešením, čo sa týka logiky a práce so zápornými číslami [11].

Základy fuzzy logiky

Ukážme si príklad, v ktorom máme program, ktorý simuluje správanie motorového vozidla na vozovke. Predstavme si situáciu, ktorú by pozorovateľ popísal takto: „Vozovka pod automobilom stúpa a výkon vozidla je príliš malý, tak postupuje pomaly.“ Počítač by takúto situáciu ťažko vyhodnotil a bolo by potrebné zadať nejaké konkrétne vstupné údaje, ako napríklad strmosť vozovky a výkon automobilu a z výsledku simulácie by počítač odvodil rýchlosť auta, ale ťažko ohodnotiť, či je to veľa alebo málo. Nejednoznačnosť ľudskej reči sa prenáša do bežného uvažovania. Napríklad, ak sa pri šoférovaní vozidla približujeme k zákrute, v mysli sa rozhodujeme, či je zákruta príliš ostrá a treba pribrzdiť, alebo či je zákruta mierna a stačí trochu ubrať z plynu. Pre takúto situáciu naša myseľ vyhodnotí tieto pravidlá a človek sa podľa toho správa. A práve na takomto princípe je založená podstata fuzzy logiky [12].

Fuzzy množiny sú vlastne formálnym zovšeobecnením klasických množín. V matematike ide o dobre preskúmanú a uzavretú oblasť. Ak uvažujeme o klasických množinách, môžeme pre každý prvok x rozhodnúť, že do množiny A buď patrí (1) alebo nepatrí (0). Príslušnosť prvku x do fuzzy množiny A udáva takzvaná hodnota funkcia príslušnosti, ktorá môže nadobúdať hodnoty z intervalu (0;1). Takisto ale vieme vytvoriť množiny, ktorá by mohla nadobúdať hodnoty 0, 1, 2 alebo -1, 0, 1. Čiže by sa jednoducho do fuzzy logiky dala implementovať ternárna sústava. Je potrebné rozlíšiť, že nejde o pravdepodobnosť, s ktorou prvok patrí do fuzzy množiny, ale skôr o silu, s ktooru do nej patrí [12].

Majme jeden pojem (takzvanú lingvistickú premennú) rýchlosť, ktorú chceme popísať tromi kvalitatívnymi pojmami (takzvanými lingvistickými hodnotami): slow, middle a fast. Teda prvky z množiny rýchlostí vieme zaradiť do troch úrovní, pričom hranice medzi týmito množinami nie sú ostré a preto sa nazývajú fuzzy množinami. Vedeli by sme si teda určiť, že keď auto ide pomaly nadobúda hodnotu -1, keď ide stredne rýchlo tak hodnotu 0 a keď ide rýchlo tak hodnotu 1 [17].



Obrázok 5: Rýchlosť vozidla [17]

Čiže premenná rýchlosť má definované tri hodnoty, ktoré predstavujú tri rôzne fuzzy množiny definované tromi rôznymi lichobežníkovými funkciami príslušnosti, ktorých hodnoty sú na zvislej osi. Funkcie príslušnosti pre jednotlivé fuzzy premenné sú navzájom farebne odlíšené. Toto by teda bol názorný príklad, ako sa vo fuzzy logike dá rozlíšiť určitá udalosť na tri stavy. V ďalších dvoch podkapitolách si ukážeme dva jednoduché príklady spojenia fuzzy logiky a ternárnej číselnej sústavy [17].

1.3.1 Využitie v automobile

Systém ABS je klasickou časťou dnešných automobilov. Aj v tomto prípade sa dá hovoriť o spojení fuzzy logiky a ternárnej logiky. Brzdný systém a dynamika sú silne nelineárne a komplexné charakteristiky, čo spôsobuje veľké prekážky pri návrhu regulátorov ABS, aj v prípade zjednodušenia matematického modelu. Využitie fuzzy logiky, ako inteligentného, znalostného systému je vhodné vtedy, keď je systém nelineárny, zložitý a často matematicky neopísateľný. Systém ABS slúži na zabráneniu blokovaniu kolies kvôli čomu by mohol vzniknúť šmyk a tým dovoľuje riadiť auto aj počas intenzívneho brzdenia bez ohľadu na kvalitu povrchu vozovky [17].

Hardvérová implementácia fuzzy logiky zabezpečila, že dĺžka jedného kontrolného cyklu fuzzy logiky klesla iba na 0.5 ms. Takéto riešenie viedlo k výborným výsledkom správania sa brzdových systémov testovaných vozidiel.

Funkcia fuzzy logiky prispôsobuje radenie automatickej prevodovky charakteru jazdy vodiča. Fuzzy logiky pozná kedy vodič ide výkonným športovým spôsobom a dovolí vytáčať motor do vyšších otáčok. naopak pri menej temperamentnej ekonomickejšej jazde radenie prebieha automaticky v nižších otáčkach, aby sa znížila spotreba i hlučnosť. Tento riadiaci program, ktorý pracuje v závislosti na jazdnom odpore dokáže prispôsobiť radenie i jazde do kopca, proti vetru alebo s prívesom. Využitím trojhodnotovej logiky sa docielilo rozdelenie, na základe snímačov, kedy sa dá hovoriť, že ekonomická jazda nadobúda hodnotu -1, športová jazda má hodnotu 1 a všetko medzi má hodnotu 0, čím systém dokáže rozlíšiť, ako prispôsobiť radenie [12].

1.3.2 ExPro Master

ExPro Master je program využívajúci fuzzy logiku, ktorý implementuje intuitívne zrejmú logiku riešenia analytických problémov hodnotenia, prognózovania a klasifikácie osobou, ktorá je v dobrom súlade so všeobecne uznávanými princípmi štúdia zložitých systémov, a preto ju možno považovať za konštruktívne riešenie pre širokú škálu systémových problémov. Štruktúra riešenia samostatného expertno-analytického problému zahŕňa tieto hlavné informačné zložky:

- koncepčný model predmetnej oblasti expertno-analytickej úlohy alebo systému preferencií, ktorý je formalizovaným vyjadrením odborníka o úlohe, jej prvkoch

a súvislostiach, hodnotenia predmetov z predmetnej oblasti alebo jednoducho predmetov reálneho sveta, ktoré sa analyzujú pri riešení problému;
- vonkajšie faktory dynamiky, prezentované vo forme štatistických údajov (ktoré popisujú stav konceptuálneho modelu a objektov v minulosti) a faktory budúcnosti (ktoré popisujú možné zmeny konceptuálneho modelu a objektov v budúcnosti);
- korekcie alebo vnútorné faktory dynamiky, ktoré generuje samotný konceptuálny model podľa stanovených pravidiel.

Systém preferencií je najdôležitejšou súčasťou expertno-analytickej úlohy a je určený na formalizovanú prezentáciu vedomých znalostí odborníka o štruktúre, súvislostiach a charakteristikách prvkov predmetnej oblasti riešeného problému. Systém preferencií je reprezentovaný ako množina vrcholov a riadených spojení medzi nimi. Vrcholy preferenčného systému popisujú koncepty, ktoré nastavuje odborník a nesú špecifické sémantické zaťaženie v závislosti od úlohy. Tieto pojmy sú zas definované prostredníctvom iných pojmov pomocou odkazov. Na vzťahy sa môžeme pozerať ako na vzťahy, ktoré určujú vplyv niektorých pojmov na iné [13].

Na formalizáciu prepojení pojmov preferenčného systému v softvérovom balíku sa používa konštruktívna fuzzy miera Sugeno, ktorá je pre každý kontext každého konceptu špecifikovaná na množine jeho konkrétnych pojmov. Inými slovami, ku každému vrcholu je priradených niekoľko fuzzy mier podľa počtu jeho kontextov. Každý pojem môže mať v rôznych kontextoch rôzne významy.

Fuzzy miery majú aj jednu úžasnú vlastnosť. Podporujú pojem modality expertných úsudkov a môžu formalizovať nielen preferencie na vrcholoch preferenčného systému, ale aj naznačovať sémantickú konotáciu (významový odtieň) týchto preferencií (možno, veľmi pravdepodobne, pravdepodobne nevyhnutnú atď.). Vplyv sémantickej konotácie je taký veľký, že v niektorých prípadoch môže viesť k opačným výsledkom, čo v plnej miere potvrdzuje doterajšia prax. Použitie fuzzy mier na reprezentáciu odborných znalostí je charakteristickou črtou a výhodou softvérového balíka [13].

Vonkajšie faktory dynamiky sú jednou z hlavných súčastí softvérového balíka, ktorá určuje variabilitu v čase tak systému preferencií, ako aj hodnotenia objektov. Vonkajšie faktory dynamiky môžu mať rôznu fyzikálnu povahu. Ako jednu z možností možno uvažovať o pôsobení niektorých vonkajších udalostí vo vzťahu k skúmanému systému. Vonkajšie faktory dynamiky sú teda zložkou softvérového balíka, ktorý zabezpečuje

dynamiku riešenia expertno-analytických úloh v závislosti od zmien vonkajších podmienok fungovania systému [13].

Korekcie alebo vnútorné faktory dynamiky sú tiež jednou z hlavných súčastí softvérového balíka, ktorá určuje variabilitu v čase tak systému preferencií, ako aj hodnotenia objektov. Na rozdiel od vonkajších faktorov sú vnútorné faktory generované samotným systémom preferencií na základe hodnotenia stavu jedného z objektov v daných časových bodoch. Pôsobenie opráv je tiež zamerané na kontext vrcholu alebo na charakteristiku objektu. Môžu byť ovplyvnené aj niekoľkými korektúrami, z ktorých každá má svoj vlastný význam. Korekcie spolu s vonkajšími faktormi tvoria jediné pole vplyvu [13].

2. Cieľ

Primárnym cieľom tejto diplomovej práce je analyzovať vlastnosti trojkovej sústavy, uviesť rozdiely medzi dvojkovou a trojkovou číselnou sústavou a poukázať na kladné vlastnosti trojkovej sústavy. V tejto práci si definujeme viacero príkladov na efektívne využívanie trojkovej sústavy v oblasti medicíny, techniky, vzdelávania a podobne a následne si ukážeme riešenia niektorých praktických úloh, v ktorých by mohla byť trojková sústava efektívne využiteľná. Ďalším cieľom diplomovej práce je poukázať na možnosti ternárneho počítača, ktorý by mohol fungovať na báze trojkovej sústavy.

Aby bolo možné zrealizovať tieto ciele, bude treba vyriešiť v tejto diplomovej práci viacero úloh:

- urobiť dôkladnú analýzu trojkovej sústavy a porovnať ju s dvojkou sústavou
- vybrať viaceré oblasti zo života človeka a možnosti riešenia problémov bežnou dvojkovou sústavou a trojkovou sústavou
- poukázať na historické míľniky počítačov na báze trojkovej sústavy a nájsť súčasné trendy v oblasti vývoja trojkového počítača
- vyriešiť niektoré logické úlohy s využitím trojkovej sústavy

3. Metódy a nástroje riešenia

V logike je trojhodnotová logika (tiež trinárna logika, trivalentná, ternárna alebo trojčlenná, niekedy skrátene 3VL) akýkoľvek z niekoľkých mnohohodnotových logických systémov, v ktorých sú tri pravdivostné hodnoty označujúce pravdivú, nepravdivú a nejakú neurčitú tretiu hodnotu. Táto logika je v kontraste s bežnejšie známymi bivalentnými logikami (ako je klasická vetná alebo booleanovská logika), ktoré poskytujú iba pravdivé a nepravdivé hodnoty.

3.1 Historické míľniky počítačov na báze ternárnej sústavy

V tejto kapitole si stručne zhrnieme poznané počítače na báze ternárnej sústavy, ktoré sú reálne, simulované alebo len imaginárne.

TERNAC

TERNAC je softvérovo emulovaný stroj vytvorený jednou z univerzít v New Yorku vývojárom menom Gideon Frieder v roku 1973. Jeho cieľom bolo vytvoriť ternárny počítať s vyhodnocovaním pomocou ternárnej logiky a jej aritmetiky oproti binárnej logike. TERNAC nebol sám o sebe počítačom, ale emulácia, ktorá bežala na inom stroj menom Borroughs B1700.

Dytrax 1000

Dytrax 1000 je imaginárny počítač, ktorý bol spomenutý v science fiction seriály The Fall of Binary Symbolism. V jeden časti bol spomenutý imaginárny ternárny počítač, the Dytrax 1000. Bohužiaľ, webová stránka aj so seriálom zmizli z internetu, čiže viac informácii nie je možno dohľadať.

TRIPS Processor

COMP203, spoločnosť na univerzite vo Wellingtone, zmieňovala imaginárny TRIPS procesor, ktorý podporoval ternárnu aritmetiku pomocou štvoritých tritových operandov s fixnou šírkou.

Team R2D2

Team R2D2, zložený z vývojárov Daniel Chillet, Ekue Kinvi-Boh a Olivier Sentieys, opísali VHDL modely so základnou ternárnou logikou a aritmetikou a pridali k tomu zopár

základných ternárnych aritmetických prvkov, ako napríklad adder, multiplier alebo shifter v projekte menom Multiple-Valued Logic architectures and circuits [14].

Setun

Ako sme si už spomenuli, asi najdôležitejším a najznámejším strojom, ktorý bol plne funkčný a postavený na ternárnej logike, bol SETUN. Setun bol reálny, fyzický ternárny počítač, postavený so sčítacou a násobiacou funkciu na Moskovskej univerzite. Jeho trojkové obvody menom flip-flap-flops neboli originálne, preto boli zapojené dvojkové obvody tak, aby mali tri stabilné stavy. Dokumenty z minulosti písali o tom, že voľba ternárnej sústavy v počítači bola vyvíjaná preto, lebo sa dokázalo, že v určitých zmysloch base-3 poskytuje najefektívnejšie využitie. Avšak tento stoj bol zostrojený na base-4 s použitím len troch stavov, keďže technológia na báze base-3 nebola v tej dobe dostupná. Setun mal štyritisíc magnetických jadier, štyritisíc germániových diód, sto tranzistorov, štyridsať vákuových trubíc, 20 kHz hodiny a 150 mW tranzistorový rozptyl. Ternárna logika bola implementovaná kombináciou dvoch feritových prvkov a ich zapojením tak, aby mohli predstavovať tri stavy. Tento prístup bol úspešný, ale neurobil nič, aby znížil počet požadovaných prvkov, pretože v skutočnosti by tieto dve feritové jadrá mohli potenciálne predstavovať dva binárne bity, čo predstavuje viac informácií ako jeden ternárny trit [14].

3.2 Súčasné trendy vývoja ternárneho počítača

Po mnoho desaťročí sa zvýšenie výkonu procesora poháňanom Moorovým zákonom dosahovalo prostredníctvom vylepšenej výrobnej technológie. Vyzerá to tak, že koniec týchto pretekov je však za rohom, pretože sa približujeme k limitu, ktorý určujú zákony fyziky. Stratégovia z firmy Intel pred pár rokmi dospeli k záveru, že jedným z riešení tohto problému by mohol byť prechod z binárnej číselnej sústavy na ternárnu číselnú sústavu. V tomto smere už bolo urobených niekoľko krokov a môžeme povedať, že nejaká z najbližších generácií procesorov Intel Core by mala byť ternárna [15].

Binárny číselný systém a binárna logika, ktorá sa používa v počítačoch, sa zdá byť optimálna, v skutočnosti je to len jedna z možností, ktorá má aj svoje nevýhody. Systém číslovania 0/1 je jednoduchý, ale binárna číslica je malá a logika áno/nie je v niektorých prípadoch hrubá a nepružná. Od čias Aristotela vedci identifikovali ternárnu logiku, kde

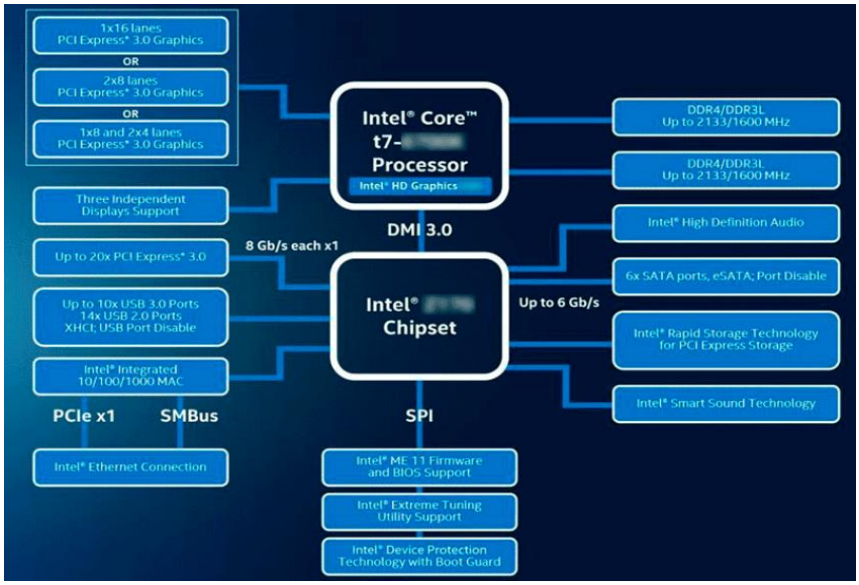
okrem áno a nie existuje tretí neurčitý význam, ako najprirodzenejší pre proces poznania. Čo sa týka aritmetiky, ternárny počet má tiež výhody oproti binárnemu [15].

Intel sa rozhodol v roku 2015 spustiť program, ktorý nazvali Dallas. V tomto programe pripravili:

- Dvojkrokový cyklus uvoľnenia procesora, menom tick-tock, bol nahradený trojkrokovým, ternárnym;
- Boli vykonané zmeny v architektúre procesora, ktorých podrobné informácie zatiaľ predstavené neboli;
- Špecifikácie hardvérových komponentov boli pripravené pre výrobcov počítačových komponentov;
- Boli optimalizované základné softvérové nástroje a knižnice spoločnosti Intel, najmä kompilátory, pre prácu s ternárnou sústavou.

Ternárna sústava je natívne kompatibilná s binárnou, pretože tá je jej podmnožinou. Pre dosiahnutie maximálneho výkonu však bude potrebná optimalizácia všetkého softvéru, ktorá bude postupne implementovaná. Schematický diagram ternárneho procesora sa takmer nelíši od toho binárneho.

Prvé modely ternárnych procesorov Intel Core by sa mali čoskoro objaviť, ale zatiaľ Intel neprezrádza žiadne ďalšie informácie, pokým nebudú mať plnofunkčný procesor pripravený na použitie. Táto novinka pôsobí senzačným dojmom, pretože ide o nevyhnutný a včasný krok. Nepochybuje sa o tom, že ternárne procesory majú veľkú budúcnosť [15].



Obrázok 6: Schematický diagram ternárneho procesora od spoločnosti Intel [15]

3.3 Dôkladná analýza trojkovej sústavy a porovnanie s dvojkovou sústavou

Ternárna číselná sústava používa base-3, na rozdiel od binárnej base-2. Zvyčajne tieto sústavy spájame s digitálnymi počítačmi alebo desatinnou aritmetickou sústavou. Každá ternárna číslica, alebo inak povedané trit, nesie približne 1,58 bitov, v porovnaní s desatinnou číslicou, ktorá nosí 3,32 bitov. N-bitové binárne číslo môže vo všeobecnosti obsahovať približne n/1,58 tritov, alebo ak použijeme trit ako jednotku informácií, jeden bit reprezentuje približne 0,63 tritu [16].

Podľa Kleenovej logiky hovoríme o logických premenných, ktoré majú hodnoty pravda, nepravda a neznáma. Pri kódovaní čísel pomocou ternárnej logiky ignorujeme konvenčné významy týchto troch symbolov a jednoducho priraďujeme číselné hodnoty, ktoré zodpovedajú implicitnému poradiu. Najčastejšie používané kódovania ternárnej sústavy je 0, 1, 2 a -1, 0, +1 [16].

Prvé kódovanie sa používa pri reprezentovaní nesymetrickej base-3, ktoré úzko zodpovedá známemu kódovaniu používanému pre desatinné a binárne čísla. Počítanie v tejto číselnej sústave začína číslami 0, 1, 2, 10, 11, 12, 20, 21, 22 atď., čo zodpovedá číslam od nuly do osem [16].

Druhé kódovanie sa používa pri reprezentácii symetrickej base-3, ktoré zapisujeme buď -1, 0, +1 alebo sa môžeme stretnúť aj s označením -, 0, +. V tomto systéme sa počítanie začína nasledovne: 0, +, +-, +0, ++, +-, -, +-0, +-+, +0-, atď., čo zodpovedá číslam od nula po osem. V nasledujúcej tabuľke sú uvedené všetky čísla v symetrickej, nesymetrickej a doplnkovej forme:

Bal	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	---	--0	--+	-0-	-00	-0+	-+-	-+0	-++	0--	0-0	0-+	60k	000	60m	0+-	0+0	0++	+--	+-0	+-+	+0-	+00	+0+	++-	++0	+++
Hept	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H	K	M	N	P	R	T	V	X	Z
Uns	000	001	002	010	011	012	020	021	022	100	101	102	110	111	112	120	121	122	200	201	202	210	211	212	220	221	222
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Tabuľka 6: Čísla v symetrickej, nesymetrickej a doplnkovej forme [8]

Vo vyššie uvedenej tabuľke riadok označený ako Hept poskytuje heptavintimálne (base-27) kódovanie tritu. Rovnako, ako u šestnástkovej sústavy, sa heptavintimál používa na reprezentáciu samotného vzoru tritu bez ohľadu na jeho interpretáciu v symetrickej alebo

nesymetrickej trojkovej sústave. Heptavintimál A teda predstavuje nesymetrickú hodnotu 10 a symetrickú -3 [16].

Výpočet pomocou binárneho systému je jednoduchší, pretože má len dva stavy, zapnutý a vypnutý. Až na zopár výnimiek, ktoré aj tak nie sú všeobecne rozšírené, neexistuje žiadne elektronické zariadenie, ktoré by fungovalo na inej ako binárnej logike. Desatinná sústava používa desať symbolov (od čísla nula po číslo deväť) a je prirodzená voľba pre počítanie. Znázornenie veľkých čísel sa vďaka nej dá jednoducho uskutočniť, pretože ak by mal byť pre každé číslo používaný iný znak, znamenalo by to teoreticky nekonečné množstvo symbolov. Preto sa vyvinuli pozičné zápisy, ktoré nám umožňujú reprezentovať ľubovoľné alebo všetky čísla zreteľne s konečnou množinou symbolov [16].

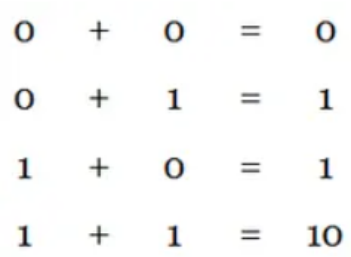
Ak je nejaký systém založený na tom, aby vyžadoval čo najmenej pamäte, bolo by logické použiť číselný systém s čo najväčším základom. Napríklad pre číselný systém, ktorý má 10 000 ako základ, môže reprezentovať číslo medzi 0 a 9 999 iba jedna číslica. Na druhej strane, pre unárny číselný systém sa na jeden symbol použije jeden znak. Optimálny číselný systém by bol taký, ktorý sa približuje k Eulerovmu číslo e. Najbližšie celé číslo k Eulerovmu číslu by bolo číslo 3 [16].

Vlastnosťou vo vyváženej, respektíve symetrickej číselnej sústave sú, že sa záporné číslo získa zámenou čísla 1 a -1, znamienko čísla je dané jeho najvýznamnejším nenulovým tritom a operácia zaokrúhľovania na najbližšie celé číslo je identická so skrátením [16].

3.3.1 Operácie binárnych čísel

Binárnosť je základný dvojčíselný systém, ktorý používa dve čísla, 0 a 1, na reprezentáciu stavov pravdy a nepravdy. Binárna aritmetika je nevyhnutnou súčasťou rôznych digitálnych systémov. Binárne čísla môžeme sčítať, odpočítať, násobiť a deliť rôznymi metódami. Vďaka skutočnosti, že binárna sústava obsahuje len dve čísla, sú aritmetické operácie oveľa jednoduchšie.

Binárne sčítavanie sa skladá zo štyroch možných elementárnych operácií:



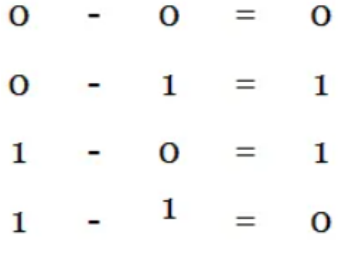
Obrázok 7: Binárne sčítavanie [Vlastné spracovanie]

Na základe obrázku vyššie je zrejmé, že prvé tri operácie produkujú výsledok, ktorého dĺžka je jedna číslica, ale pre poslednú operáciu sú výsledok dve číslice, čo nazývame prenos. Keď je súčet dvoch bitov väčší ako jedna, musíme posunúť stĺpec vľavo. V desiatkovej sústave by bolo 1+1 rovné dvom. V binárnom sa to ráta tak, že $1*2^1 + 0*2^0$, preto nulu necháme vpravo a jednotku posunieme naľavo od nuly do druhého stĺpca.

Ako príklad si ukážeme v binárnej sústave pripočítať číslo jedenásť k číslu trinásť. Číslo jedenásť zapísane v binárnej sústave je 1011 a číslo trinásť 1101, čiže 1011 + 1101 = ?. Pri výpočte postupujeme sprava doľava. Postup výpočtu bude:

1. Prvý stĺpec je 1 + 1 = 10. Necháme v prvom stĺpci nulu a jednotku prenesieme do druhého stĺpca.
2. Druhý stĺpec je 1 + 0 + 1(prenesená) = 10. Zas zapíšeme nulu a jednotku prenesieme do tretieho stĺpca.
3. Tretí stĺpec je 0 + 1 + 1(prenesená) = 10. Napíšeme nulu a jednotku prenesieme do posledného stĺpca.
4. Štvrtý stĺpec je 1 + 1 + 1(prenesená) = 11. Napíšeme jednotku a jednotku prenesieme vľavo do nového stĺpca, ktorý doteraz neexistoval, pretože neuchovával žiadnu hodnotu. Výsledok potom bude 11000.

Binárne odčítanie sa skladá zo štyroch možných elementárnych operácií:



Obrázok 8: Binárne odčítavanie [Vlastné spracovanie]

Písomný algoritmus spočíva v tom, že si menšenec a menšiteľ zapíšeme pod seba tak, aby jednotky toho istého rádu boli v rovnakom stĺpci. Potom postupne odčítame číslice nultého rádu, číslice prvého rádu, číslice druhého rádu atď. Ak cifra i-teho rádu menšenca je menšia ako cifra i-teho rádu menšiteľa, potom formálne zväčšíme počet jednotiek i-teho rádu v menšenci o 10_2. Aby sa však rozdiel nezmenil, musíme menšiteľa zväčšiť o jednu jednotku i+1-ého rádu.

11001_2	111011_2	1001_2	1100_2
$-\ 11010_2$	$-\ 111_2$	$-\ 1101_2$	$-\ 110_2$
1001_2	110100_2	110_2	1010_2

Obrázok 9: Binárne odčítavanie [Vlastné spracovanie]

Binárne násobenie sa skladá zo štyroch možných elementárnych operácií:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Obrázok 10: Binárne násobenie [Vlastné spracovanie]

Binárne násobenie je presne také isté, ako násobenie v desatinných číslach, to znamená, že násobíme čísla sprava doľava, násobíme každú číslicu jedného čísla každým číslom druhého čísla a potom ich zrátame. Ako príklad si v binárnej sústave ukážeme násobenie čísel 11 a 13, čo sa znázorňuje ako 1011*1101. Postup výpočtu bude:

1. Prvým krokom by bolo vynásobenie čísla 1011 posledným číslom druhého čísla, a to je 1. Výsledok bude 1011.
2. Druhým krokom bude vynásobenie čísla 1011 druhým číslom sprava druhého čísla, číslom 0. Výsledok bude 00000, číslic je päť, pretože pripájame na koniec jednu nulu.
3. Ďalším krokom bude vynásobenie toho istého čísla tretím číslom sprava a výsledok bude 101100. Tu už sme pripájali na koniec dve nuly
4. Posledným násobením vyjde výsledok, po pripojení troch núl nakoniec, číslo 1011000.

5. Teraz spočítame všetky výsledky z predošlých krokov dokopy. 1011 + 0000 + 101100 + 1011000 = ((1011 + 00000) + 101100) + 1011000 = (01011 + 101100) + 1011000 = 110111 + 1011000 = 10001111. Takže výsledok je $1*2^7 + 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 128 + 8 + 4 + 2 + 1 = 143 = 11 * 13$.

Binárne delenie je podobné desatinnému deleniu, ale tu sa používa len jednotka a nula, pričom delenie nulou nemá žiadny význam (no meaning):

0	÷	0	=	No meaning
0	÷	1	=	1
1	÷	0	=	No meaning
1	÷	1	=	1

Obrázok 11: Binárne delenie [Vlastné spracovanie]

Skúsime teda najprv vydeliť číslo 6 číslom 3. V binárnom kóde by to bolo 110 / 11. Ideme zľava doprava až pokým ľavá strana nebude väčšia alebo rovná, ako deliteľ. Potom odpočítame násobok deliteľa, ktorý je menší alebo rovný ako delenec. Násobiteľ(1) sa pripojí ku kvocientu a výsledok odčítania je zvyšok. Znížime o jeden bit (zľava doprava) pre zostávajúcu časť delenca a skontrolujeme, či je väčší ako deliteľ. Ak nie, pripojíme 0 ku kvocientu. Ak áno, opakujeme postup.

Kroky ku vydeleniu našich dvoch čísel v binárnej sústave by teda boli:

1. Prvé číslo delenca, jeden, nie je väčšie ani rovné číslu 11. Nemusíme pridať 0 do výsledku, pretože 0 na ľavej strane je nepodstatná. Nič teda nerobíme a pokračujeme ďalej.
2. Číslo jedenásť je väčšie alebo rovné, ako deliteľ, čiže pridáme do výsledku číslo jeden a odpočítame 11 od 11, čo nám dá zvyšok 0. Odpočítavame jeden-jeden a nie jedenásť od jedenásť.
3. Zostáva nám už len posledné číslo, 0, ktoré nie je väčšie ani rovné 11, čiže pridáme do výsledku 0.

Keďže už sme všetko vydelili a neostal nám žiadny zvyšok, tak si to už len skontrolujeme. Výsledok nám vyšiel 10 bez zvyšku a vieme, že v binárnej sústave je číslo 10 rovné dvom.

3.3.2 *Operácie ternárnych čísel*

Rovnako ako pri binárnych číslach, začneme aritmetické operácie ternárnych čísel sčítavaním. Ak chceme pripočítať číslo v ternárnej sústave, pripočítame ho k najmenej významnej číslici bez ohľadu na číselnú základňu. Ak je pridanie k najmenej významnej číslici nenulové, pridáme ju na ďalšiu číslicu. Pripočítavanie sa vykonáva prenosom z najmenej významnej číslice až po najviac. Ukážeme si pripočítanie čísla 5:

V desatinnej sústave: V nepodpísanej ternárnej: Vo vyváženej ternárnej:

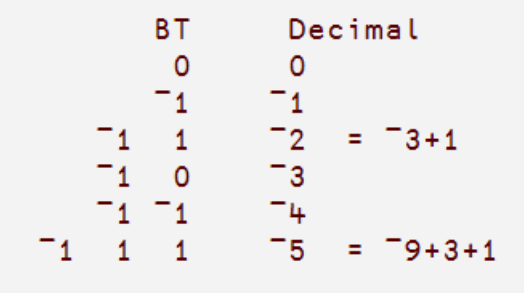
```
  5        1        0 0
 +1       12        +--
---      + 1      +   +
  6      ---      ----
         20        +-0
```

Vyvážená ternárna sústava (BT) reprezentuje silu base-3: 1, 9, 27, 81, 243..., rozlišuje sa od štandardnej ternárnej sústavy tak, že namiesto len kladných čísel (0, 1, 2) používa záporné (+1, 0, -1). V tabuľke nižšie je zopár nezáporných čísel spolu s ich decimálnou ekvivalenciou [18]:

```
          BT     Decimal
           0     0
           1     1
         1 -1    2   = 3 + -1          "one, bar-one"
         1  0    3
         1  1    4
      1 -1 -1    5   = 9 + -3 + -1     "one, bar-one, bar-one"
      1 -1  0    6
      1 -1  1    7   = 9 + -3 +  1
      1  0 -1    8
      1  0  0    9
      1  0  1   10
      1  1 -1   11
      1  1  0   12
      1  1  1   13
   1 -1 -1 -1   14   = 27 + -9 + -3 + -1
```

Obrázok 12: Ternárne nezáporné čísla [18]

Vyvážená ternárne číslice sú symetrické vzhľadom na ich znamienko, čiže negatívne číslo je záporné číslo každej z jeho číslic. Tu je niekoľko negatívnych symetrických čísel:



Obrázok 13: Ternárne záporné čísla [18]

Sčítanie ternárnych čísel

Jednoduchá tabuľka, na základe ktorej sa môžeme riadiť pri sčítavaní symetrických ternárnych číslic je:

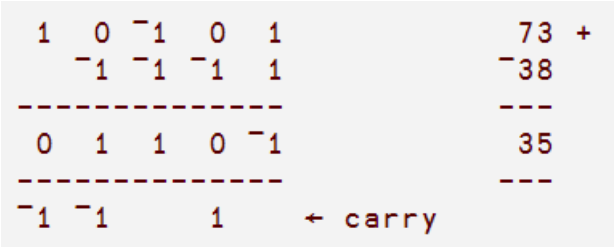
Tabuľka 7: Sčítavanie ternárnych číslic [Vlastné spracovanie]

+	**-1**	**0**	**1**
-1	1-	-1	0
0	-1	0	1
1	0	1	-1+

1- Znamená 1, kde pri pripočítavaní vložíme do ďalšieho stĺpca číslo -1

-1+ Znamená -1, kde pri pripočítavaní vložíme do ďalšieho stĺpca číslo 1

Ako príklad si zrátame čísla 73 a -38:



Obrázok 14: Sčítanie ternárnych číslic [18]

Odčítanie by vyzeralo presne tak isto, ako pripočítavanie s pridaním negácie do menšiteľa.

Násobenie ternárnych čísel

Na násobenie ternárnych čísel využívame jednoduché stredoškolské násobenie, na základe nasledujúcej tabuľky:

×	$\bar{1}$	0	1
$\bar{1}$	1	0	$\bar{1}$
0	0	0	0
1	$\bar{1}$	0	1

Obrázok 15: Násobenie ternárnych číslic [19]

Ako príklad si vynásobíme čísla 25 a -5:

```
        1  0 -1  1          25 ×
           -1  1  1          -5
        -----------       ----
-1  0  1 -1        +       -225 +
    1  0 -1  1              75
       1  0 -1  1           25
-----------------         ----
-1  1  1  1  0  1         -125
-----------------         ----
      -1
```

Obrázok 16: Násobenie ternárnych číslic [19]

Delenie ternárnej sústavy je veľmi komplexné a v praxi sa stretneme so zložitými programami, ktoré to dokážu vypočítať. Ako príklad si jeden uvedieme, ale nebudeme si ho bližšie rozoberať.

```
balanced int rem, quo; /* the remainder and quotient, return values */
void div( balanced int ividend, balanced int divisor ) {
    balanced int one = 1; /* determines whether to negate bits of quotient */
    if (divisor < 0) { /* take absolute value of divisor */
        divisor = -divisor;
        one = -one;
    }
    quo = dividend;
    rem = 0;
    for (i = 0; i < trits_per_word; i++) {
        /* first shift rem-quo double register 1 trit left */
        (rem,quo) = (rem,quo) <<3 1;

        /* second, compute one trit of quotient */
        if (rem > 0) {
            balanced int low = rem - divisor;
            if ( (-low < rem)
            ||   ((-low == rem) && (quo > 0)) ) {
                quo = quo + one;
                rem = low;
            }
        } else if (rem < 0) {
            balanced int high = rem + divisor;
            if ( (-high > rem)
            ||   ((-high == rem) && (quo < 0)) ) {
                quo = quo - one;
                rem = high;
            }
        }
    }
}
```

Obrázok 17: Program na delenie ternárnych číslic [19]

3.3.3 Porovnanie dvojkovej a trojkovej sústavy

Matematicky je ternárne kódovanie efektívnejšie ako binárne kódovanie. Je málo používané vo výpočtoch, pretože technológia pre binárne spracovanie je už zavedená a implementácia ternárneho kódovania je komplikovanejšia, ale zostáva relevantná pre algoritmy, ktoré využívajú rozhodovacie stromy a v komunikácií. Okrem úvah o efektívnosti, ternárne kódovanie sa javí vhodnejšie ako binárne kódovanie pri klasifikácií mnohých problémov reálneho sveta ako umelá inteligencia alebo medicína [20].

Problémom optimálneho kódovania čísel sa zaoberali mnohí vedci a je známe, že ternárne kódovanie je efektívnejšie ako binárne. Dôvodom, prečo sa v hardvéri veľmi nepoužíva, je technológia. Implementácia ternárneho kódovania je zložitejšia vo výpočtoch, hoci sa používa v digitálnej komunikácii na opravu chýb. Ternárne kódovanie momentálne zostáva relevantné v algoritmoch, kde sa používajú rozhodovacie stromy [20].

Uvažujme o kódovaní čísel pri ľubovoľnom základe, ktorý si označíme písmenom b. Vzhľadom k tomu, že pravdepodobnosť použitia symbolov b možno považovať za rovné

1/b, informácie spojené s každým symbolom sú *log b*. Efektívnosť kódovania má vzorec E(b) = *ln b*/b. Na základe tohto vzorca vieme zistiť efektívnosť kódovania v určitých base, ktoré je znázornené v nasledujúcej tabuľke:

Tabuľka 8: Efektívnosť kódovania v určitých číselných základniach [20]

b	2	e	3	4	5	8	10	12	20	100
E(b) nats	0.347	0.368	0.366	0.347	0.322	0.260	0.230	0.207	0.150	0.046
E(b) bits	0.500	0.531	0.528	0.500	0.465	0.375	0.331	0.298	0.216	0.066

Na základe tejto tabuľky vieme určiť, že najefektívnejšie je b=3, keďže je to najbližšie k písmenu *e* (porovnanie hodnoty 0,366 s optimálnou hodnotou e = 0,368).

Keďže kódovanie má aspekty, ktoré presahujú rámec reprezentácie údajov, problém optimálneho kódovania vedie k mnohým filozofickým otázkam. Keďže binárne kódovanie nie je optimálne, mala by existovať logika, ktorá má viac ako dve triedy, ale nie je jasné, ako sa táto logika prejaví v prirodzených procesoch alebo v poznaní. Absolútna optimálnosť sa zdá byť nedosiahnuteľná, pretože zodpovedá necelému číslu, čo je iracionálne. Môžeme konštatovať, že matematika nie je optimálna ani úplná, spôsob reprezentácie reality a tento nedostatok možno k známym obmedzeniam matematiky [20].

Rozhodovacie stromy sú prítomne vo väčšine dátových štruktúr. Každý uzol (okrem koreňa) v strome je spojený usmernenou hranou práve z jedného uzla. Aj keď sa binárne stromy používajú najčastejšie, existujú situácie, kedy ternárne stromy ponúkajú výhody. Rozhodovacie stromy pre hodnoty údajov spojené s rôznymi pravdepodobnosťami sú dané aritmetickým kódovaním alebo Huffmanovím kódovaním. Huffmanovo kódovanie je optimálne, ak sú všetky pravdepodobnostné symboly integrálne mocniny prevrátenej hodnoty. Tým, že sa upustí od obmedzenia, že každý symbol sa musí previesť na integrál, počet bitov v aritmetickom kódovaní dosahuje vyššiu efektivitu. Ale keďže implementácia je zložitejšia, obmedzíme toto riešenie len na Huffmanovo kódovanie [20].

Obrázok 18: Binárny a trinárny rozhodovací strom [20]

Na obrázku vidieť, že priemerná dĺžka symbolu v binárnej sústave je 29/9 = 3,22 bitov, pričom v ternárnej sústave vpravo sú to 2 trity. Keďže jeden trit je rovný $\log_2 3 = 1{,}585$ bitov, ternárne kódovanie je efektívnejšie.

Tabuľka 9: Porovnanie kódovania symbolov rôznej dĺžky [20]

Binary	Ternary
3: 0, 10, 11	3: 0, 1, 2
4: 00, 01, 10, 11	4: 0, 1, 20, 21
5: 00, 01, 10, 110, 111	5: 0, 1, 20, 21, 22
6: 00, 01, 100, 101, 110, 111	6: 0, 10, 11, 20, 21, 22
7: 00, 010, 011, 100, 101, 110, 111	7: 0, 10, 11, 12, 20, 21, 22
8:000, 001, 010, 011, 100, 101, 110, 111	8: 00, 01, 10, 11, 12, 20, 21, 22
9: 000, 001, 010, 011, 100, 101, 110, 1110, 1111	9: 00, 01, 02, 10, 11, 12, 20, 21, 22

Tabuľka 10: Porovnanie využitia pamäte pre symboly rôznej dĺžky [20]

Symbols	3	4	5	6	7	8	9	10
Binary (bits)	1.66	2	2.4	2.7	2.86	3	3.2	3.4
Ternary (trits)	1	1.5	1.6	1.83	1.86	2	2	2.2
Ternary (bits)	1.59	2.38	2.54	2.91	2.93	3.17	3.17	3.48

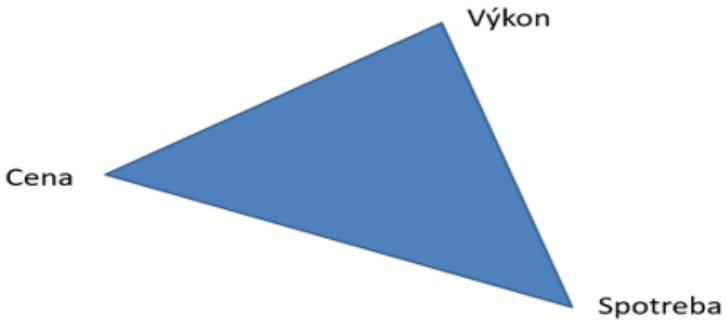
Z tejto základnej kombinatoriky je celkom jasné, že keďže ide o väčšiu množinu, ternárne kódovanie je efektívnejšie než binárne kódovanie pre špecifické hodnoty. Pri zvážení nerovnomerných pravdepodobností, menšia abeceda binárneho kódu zvyšuje výhody ternárneho kódovania ešte viac [20].

3.4 Možnosti riešenia problémov dvojkovou a trojkovou sústavou zo života

Riešenie úloh z bežného života, ekonomiky, techniky, filozofie a pod. si vyžaduje neštandardný prístup, ktorý nie je možné vyjadriť klasickou logikou využívajúcou dvojstavovú logiku. Mnohokrát sa nevieme presne rozhodnúť áno alebo nie, ale často využívame možnosti možno áno, respektíve možno nie. Pre porozumenie si uvedieme niekoľko príkladov [11].

Napríklad pri nákupe motorového vozidla sa rozhodujeme medzi viacerými parametrami, ako napríklad výkon, cena a spotreba, avšak aj keď budeme vedieť hodnoty daných parametrov, nie vždy sa hneď rozhodneme áno alebo nie. Veľmi často je možné počuť odpoveď možno.

Druhý prípad je napríklad z oblasti rozhodovania v oblasti riadenia výroby na základe poznatkov z oblasti trhu. Niektoré výrobky majú dobrý odbyt na trhu, iné slabší, a niektoré takmer žiadny. Preto často vzniká otázka, či budeme vyrábať ten daný výrobok a ako klasickú odpoveď na danú otázku môžeme dostať áno, nie alebo že sa ešte rozhodneme.



Obrázok 19**:** Rozhodovanie pri parametrickom výbere [11]

Iným prípadom je problém z oblasti medicíny. Na liečenie choroby existuje viacero liekov. Mnohé z nich majú aj vedľajšie účinky. Nasledujúca tabuľka obsahuje trojstavové rozhodovanie možnosti použitia liekov:

Tabuľka 11: Trojstavové rozhodovanie v medicíne [11]

	Liek1	**Liek2**	**Liek3**	**Liek4**	**Liek5**	**Liek6**
Choroba 1	1	1	0	0	-1	-1
Choroba 2	1	0	-1	0	-1	1
Choroba 3	-1	-1	-1	0	0	1
Choroba 4	-1	-	1	1	0	0

Hodnota v tabuľke sa týka možnosti používania daného lieku na liečenie niektorých chorôb, kde 1 znamená áno, -1 znamená nie a 0 znamená, že v istých prípadoch je ho možno použiť ako náhradný liek.

Podobné problémy vznikajú napríklad aj pri výmene oleja. Veľmi často v servise navrhnú, že daný olej je práve pre vaše auto (1), iný olej nesmiete použiť (-1) a niektorý je možné použiť, pretože to nie je originál, ale vozidlu neuškodí (0).

Tabuľka 12: Trojstavové rozhodovanie v automobilizme [11]

	Olej 1	Olej 2	Olej 3	Olej 4
Auto 1	1	1	0	0
Auto 2	1	0	-1	0
Auto 3	-1	-1	-1	0
Auto 4	-1	-	1	1

Taktiež v oblasti vzdelávania je možné použiť trojstavové veličiny, ako napríklad v organizačnej časti je možné zobraziť úspešnosť študentov pri možnostiach prechodu do vyššieho ročníka [11].

Tabuľka 13: Trojstavové rozhodovanie vo vzdelávaní [11]

č.št	meno	priezvisko	stupeň	ročník	postup
2560	Ján	Kováč	I	2	1
3251	Peter	Novotný	I	1	1
2562	Karol	Opatrný	I	2	-1
4100	Ján	Rybárik	I	1	0
3551	Miloš	Slavík	I	2	0
	poznámka				
	1	prevedný			
	-1	neprevedený / opakuje ročník			
	0	podmienečne prevedený / opakuje predmet			

V prípade nedosiahnutia plného počtu kreditov, napríklad 55 zo 60, študent bude podmienečne prevedený do vyššieho ročníka. V tradičnej logike nie je možné jednoznačne povedať áno alebo nie.

Takisto je možné ďalšie využívanie napríklad v oblasti hodnoteniach riešenia nejakého zadania. Niektorý študent síce správne vyriešil úlohu, ale riešenie nie je optimálne, alebo nie sú uvedené všetky možnosti. Pre tieto účely, okrem možnosti správne (1) a nesprávne (-1), je možno zaviesť parameter neoptimálne (0).

Podobne je možné vytvoriť zoznam študentov na určité akcie, napríklad zaradený (1), nezaradený, ale nemáme signál pre určenie náhradníka (-1) alebo náhradník (0).

Zaujímavou oblasťou využívania trojstavovej logiky je vytvorenie databázy predmetov, ktoré si študent volí v priebehu štúdia. V danom prípade, niektoré predmety sú povinné, iné voliteľné a iné v danom ročníku nie je vhodné si voliť alebo dokonca nie je možné, nakoľko študent ešte neabsolvoval predmet, ktorý musí byť absolvovaný prd začatím daného predmetu.

Tabuľka 14: Trojstavové rozhodovanie vo vzdelávaní [11]

program	stupeň	ročník	Inf 1	Inf 2	DBS	PocS	Prog1	Prog2	MIS	...
HI	I	1	1	1	0	0	1	-1	-1	
HI	I	2	-1	0	1	1	-1	0	-1	
HI	I	3	-1	0	0	1	0	1	-1	
HI	II	1								
HI	II	2								
MRIT	I	1								
MRIT	I	2								
MRIT	I	3								
MRIT	II	1								
MRIT	II	2								

Jeden a ten istý predmet, pre určitý študijný program a ročník, môže nadobudnúť hodnotu -1, 0, 1 v závislosti od toho, aké predmety už študent absolvoval. Veľkú úlohu má zavedenie trojstavovej logiky v oblasti riešenia optimalizačných úloh s využívaním pravdepodobnostných veličín [11].

3.5 Riešenie niektorých logických úloh s využitím ternárnej sústavy

Zatiaľ čo mať rozsiahle znalosti matematiky nie je potrebné na učenie sa počítačového programovania, určite pomáha mať základné pochopenie niektorých základných matematík, ktoré umožňujú fungovanie výpočtovej techniky.

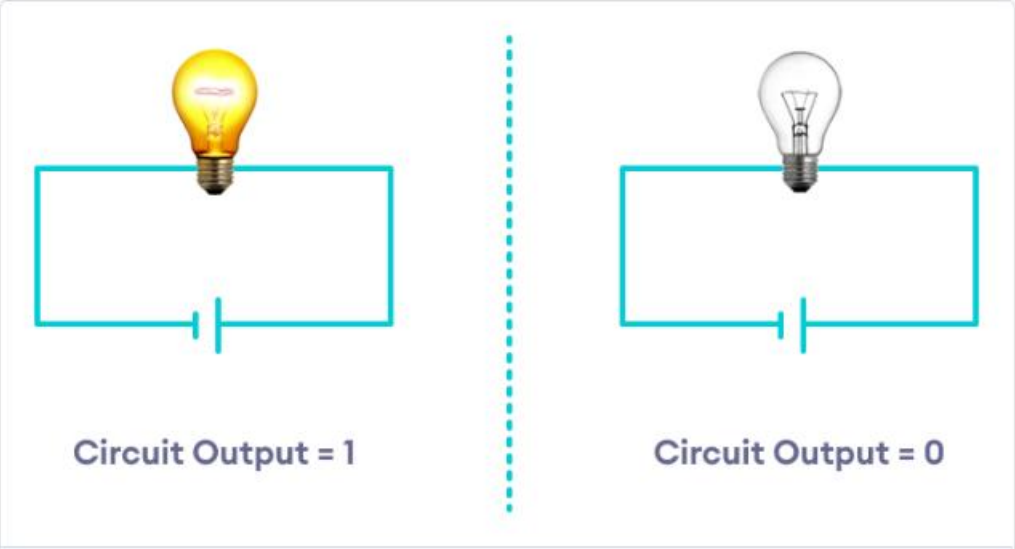
Termín binárny znamená niečo, čo má iba dva možné objekty alebo stavy. V binárnom číselnom systéme sú tieto dva objekty čísla 0 a 1. Tieto dve čísla môžu predstavovať rôzne veci. Existuje nespočetné množstvo úloh, ktoré sú ľahko riešiteľné pomocou binárnej sústavy, ale predsa len by bolo lepšie a logickejšie použiť v niektorých prípadoch ternárnu sústavu.

3.5.1 Fungovanie počítačových obvodov

Počítače pracujú na elektrických signáloch generovaných obvodmi. Aby sme mohli navrhnúť počítač, ktorý beží efektívne, potrebujeme systém, ktorý dokáže interpretovať elektrické signály jednoduchým a efektívnym spôsobom. Dobrým spôsobom, ako to urobiť, je interpretovať elektrické signály ako binárne hodnoty:

- 0 pre nízku hodnotu napätia
- 1 pre vysokú hodnotu napätia

Jednoduchší spôsob, ako o tom premýšľať, by mohol byť predstaviť si žiarovku. Ak je žiarovka vypnutá, tento stav sa interpretuje tak, že má hodnotu 0. Ak je zapnutá, interpretuje sa tak, že má hodnotu 1.



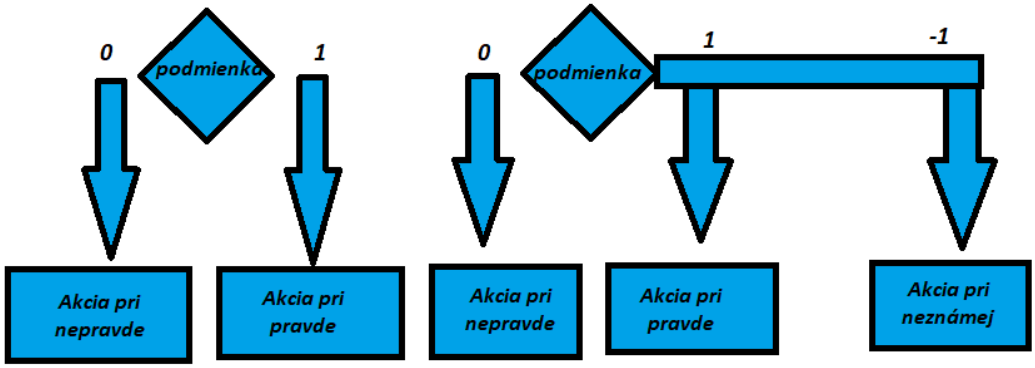
Obrázok 20: Binárna interpretácia stavov žiaroviek [21]

Počítače na báze ternárnej sústavy by ale vedeli spraviť oveľa viac, uchovávať oveľa viac informácii v rovnako veľkej pamäti a tým byť výkonnejšie. Keďže binárna sústava rozlišuje pri jednom rozhodovacom procese dve možnosti, už z čisto logického hľadiska je zrejmé, že keby uchovávala tri, dokáže omnoho viac

Namiesto možností zapnuté a vypnuté, by sa do obvodu dalo integrovať aj napríklad polovičný prúd, ktorý by vytváral tretiu možnosť. Bolo by potrebné na to implementovať modernú technológiu na určovanie úrovne napätia, ale určite by to otvorilo veľa ďalších možností. Binárna sústava sa používa v počítačoch úplne vo všetkom, či už v textoch, číslach, rozhodovacích operáciách, obrázkoch alebo zvukoch. Napríklad namiesto takzvaných flip-flop, keby sa použili flip-flap-flop, tak by sme do prvej hodnoty vedeli

uložiť 0, do druhej -1 a do tretej +1. Tieto tri vstupy by sa vedeli nakombinovať len na jednom trite a vytvoriť vstup, ktorý by si vyžadoval dva bity. Akú pamäť zaberá jeden bit a jeden trit sme si už porovnali a preto vieme, že pamäť by sa ušetrila. A to hovoríme len o jednom trite. Predstavte si, že počítače fungujú na nespočetne veľa bitoch, kde by mohla byť ušetrená pamäť a tým zvýšený absolútny výkon celého počítačového systému [21].

Pochopenie ternárnych čísel nám tak môže pomôcť pochopiť teoretické fungovanie počítačových operácií na abstraktnej úrovni, aj keď náš slabý ľudský intelekt nám neumožní pochopiť úplnú zložitosť počítačových operácií. Moderné počítače používajú binárnu logiku na to, aby sa neustále rozhodovali. Tieto rozhodnutia vedú k tomu, že naše počítače podniknú určitý postup namiesto iného.



Obrázok 21: Rozhodovanie v binárnej a ternárnej logike [Vlastné spracovanie]

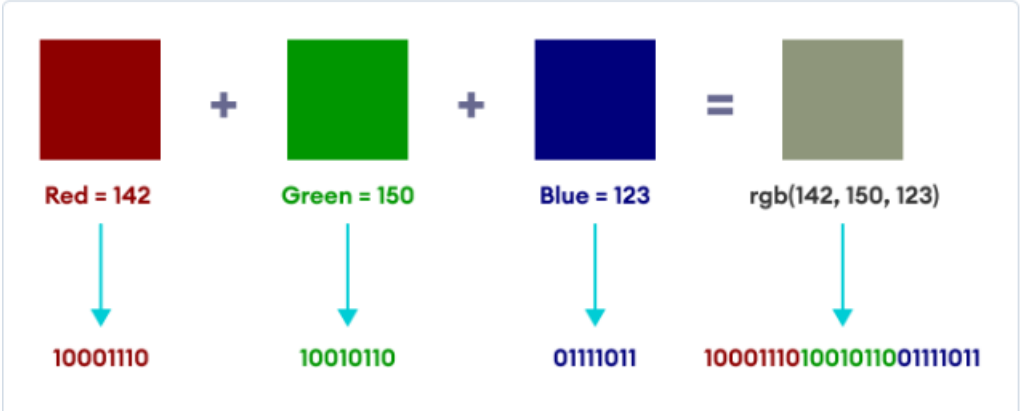
3.5.2 *Pixely a obrázky*

Nie je prekvapením, že obrázky sú tiež často reprezentované číslami. V počítačoch sa obrázky najčastejšie vytvárajú pomocou malých farebných štvorcov, nazývaných pixely. Premýšľajte o mozaike v reálnom živote, ktorá predstavuje obraz alebo vzor vytvorený spojením mnohých malých farebných kúskov. Dá sa to porovnať k skladačke puzzle, kde kombinujeme menšie kúsky na to, aby sme vytvorili väčší, úplný obraz [21].

Pixely fungujú podobným spôsobom. Tisíce malých farebných štvorcov tvoria obrázky, ktorú sú zobrazené na našich obrazovkách. Existuje mnoho spôsobov, ako sú farby v pixeloch kódované, ale najčastejšie používaným kódom je RGB kód (červená, zelená a modrá). RGB kódy fungujú kombináciou týchto farieb, aby vytvorili všetky odtiene, ktoré

vidíme na moderných zariadeniach. Každá z troch farebných zložiek je kodifikovaná číslom, ktorého hodnoty sa pohybujú od 0 do 255.

Ako príklad uvedieme farbu reprezentovanú RGB kódom. Tento farebný kód má tri komponenty: červená (142), zelená (150) a modrá (123). Vo vnútri našich počítačov je každá z týchto farebných komponentov reprezentovaná ich binárnymi ekvivalentmi pomocou ôsmich bitov a potom sú kombinované dohromady. Binárny kód pre červenú farbu (142) je 1000 1110, binárny kód pre zelenú farbu (150) je 1001 0110 a binárny kód pre modrú (123) je 111 1011. Počítač kombinuje tieto čísla zľava doprava, aby uložil RGB kód do pamäte. Kompletná RGB kombinácia týchto dvoch farieb by bola 100011101001011001111011 [21].



Obrázok 22: Práca s RGB kódom [21]

Keďže sa stále bavíme o kódovaní v počítačovom systéme, takisto vieme povedať, že v prípade, že by sme RGB farby zakódovali do ternárneho systému, bolo by to rýchlejšie, efektívnejšie a jednoduchšie na zaťaženie pamäte. Ternárny kód pre červenú farbu v nesymetrickej ternárnej sústave by bolo 12 021, pre zelenú farbu by ternárny kód bol 12 120 a pre modrú farbu by ternárny kód bol 11 120. Jednoduchšia práca s touto sústavou by zapríčinila oveľa širšie možnosti manipulácie s obrazom, skrytie jedného obrázka vo vnútri druhého a veľa iných komplexných úloh bez výrazných obmedzení.

3.5.3 Aplikácia v medicíne s fuzzy logikou

V tejto oblasti je vhodné spojiť fuzzy logiku s binárnou a ternárnou logikou, pretože zdravie človeka zahŕňa viaceré úrovne neistoty a nepresnosti a je neodmysliteľnou súčasťou medicíny. Jedno ochorenie sa môže prejaviť úplne odlišne v závislosti od pacienta a s rôznou intenzitou. Jeden príznak môže zodpovedať rôznym chorobám. Na druhej strane, niekoľko chorôb prítomných u pacienta môže interagovať a interferovať so zvyčajným popisom

ktorejkoľvek z chorôb. Najlepší a najpresnejší popis entít choroby používa lingvistické výrazy, ktoré sú tiež nepresné a vágne. Navyše, klasické koncepty zdravia a choroby sa navzájom vylučujú a sú opačné [22].

Na zvládnutie nepresnosti a neistoty máme k dispozícii fuzzy logiku. Fuzzy logika zavádza čiastočné pravdivostné hodnoty, medzi pravdou a nepravdou. My sa pozrieme na prípady, kedy si situáciu, medzi pravdou a nepravdou, označíme len jedným stredným bodom, aby sme ukázali využitie a pochopenie ternárnej sústavy v praxi.

Podľa Aristotelovej logiky máme pre daný výrok alebo stav iba dve logické hodnoty, pravda-nepravda, čierna-biela alebo 1-0. V skutočnom živote veci nie sú čierne ani biele, ale väčšinou sú sivé. Preto je v mnohých praktických situáciách vhodné zvážiť medziľahlú logickú hodnotu. Zvážme si teda zdravie v súvislosti s logikou. Môžeme si určiť, že výrok si zdravý, má hodnotu 1 a výrok si chorý má hodnotu -1. Ale logicky môžu existovať hodnoty medzi tým, ako napríklad nie som chorý, ale ľudovo povedané „niečo na mňa lezie“. Tento výrok by mohol mať hodnotu 0. Ak teda človek je zdravý, nemusí brať žiadne lieky, ak je chorý, musí brať lieky na konkrétnu chorobu, ktorou je nakazený, ale ak je človek v hodnote 0, môže urobiť preventívne riešenia, ako napríklad brať viac vitamínov alebo nevystavovať sa zime [22].

Tu si ukazujeme jednoduchú fuzzy logiku spojenú s ternárnou logikou, ktorú však v nejakej konkrétnej praxi ťažko využijeme, ale chceli sme poukázať na to, že ternárne rozhodovanie je prítomné aj v bežnom ľudskom živote.

Takisto si môžeme dať príklad s konkrétnym užívaním liekov. Predstavme si nejaký konkrétny obraz človeka a jeho choroby, na ktorú existujú rôzne lieky, avšak nie všetky sú medzi sebou kombinovateľné. Pri tomto príklade budeme brať do úvahy len obyčajnú chrípku. Na chrípku sa využíva pomerne veľa druhov liekov, ale nie všetky sú možné medzi sebou kombinovať. Skutočnosti, na ktoré sa budem snažiť poukázať, nemusia byť z medicínskeho hľadiska pravdivé, ide len o ukázanie ternárnej logiky v databáze liekov v praxi [22].

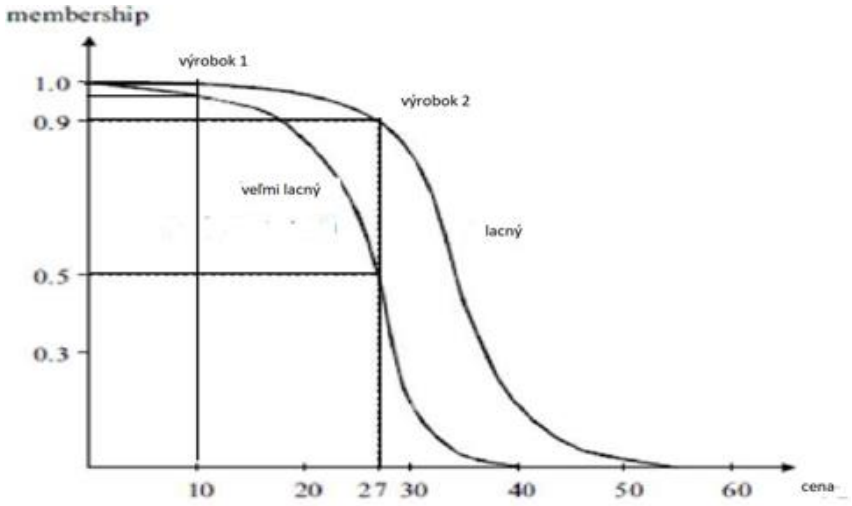
Tabuľka 15: Možná kombinácia liekov pri chrípke [Vlastné spracovanie]

Lieky	Antipyretiká	Analgetiká	Lokálne dekongestíva	Antiseptiká a dezinficienciá	Prípravky na zvýšenie imunity
Antipyretiká	0	-1	-1	1	1
Analgetiká	-1	0	0	0	-1
Lokálne dekongestíva	-1	0	0	1	-1
Antiseptiká a dezinficienciá	1	0	1	0	0
Prípravky na zvýšenie imunity	1	-1	-1	0	0

Na trhu je nespočetne množstvo liekov a nemožno si pri všetkých presne pamätať, ktoré spolu nemožno kombinovať, ktoré nemajú spolu žiadny výrazný efekt, alebo ktoré spolu možno a je odporúčané kombinovať. V tejto tabuľke sú označené číslom 1 lieky, ktoré možno spolu kombinovať a aj sa to odporúča, číslom -1 lieky, ktoré spolu nemožno kombinovať a číslom 0 lieky, ktoré spolu možno kombinovať, ale spolu nemajú nejaký výrazný efekt. Toto je len ilustračný príklad fungovania jednej z mnoho oblastí v reálnom živote, avšak táto téma ešte nie je natoľko rozvinutá [22].

3.5.4 Aplikácia pri cenách s fuzzy logikou

Fuzzy logika je novým impulzom pri zavádzaní trojstavovej logiky a jej využívaní pri riešení mnohých úloh. Pre jednoduchosť si predstavíme príklad, keď o niektorom výrobku hovoríme, že je drahý, ale nie príliš drahý, respektíve že je lacný, ale nie príliš lacný. Uvažujme, že výrobok je lacný. Pojem lacný je nejasný. Aby sme vedeli presne určiť daný pojem, je potrebné si definovať funkciu. Budeme uvažovať o cene v rozmedzí 0-60 Eur [11].

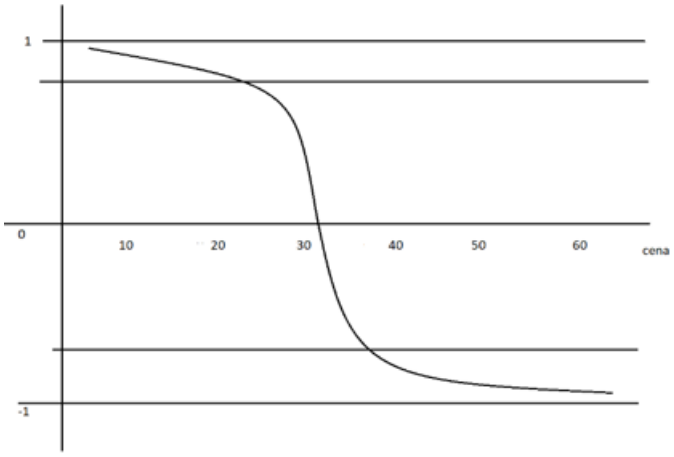


Obrázok 23: Funkcia na porovnávanie cien

Horizontálna os ukazuje cenu a na vertikálnej osi sú hodnoty funkcie. Krivka lacný ukazuje šance, ktoré musia byť vo fuzzy množine lacná. Ak budeme definovať funkciu lacná pomocou danej funkcie, tak ak výrobok má cenu 10 eur, tak ho zaradíme do množiny lacný a hodnota funkcie bude 1. Podobne, ak cena bude napríklad 27 eur, tak tiež ho môžeme zaradiť do množiny lacný a hodnota funkcie bude 0.9. Naopak, výrobok s cenou okolo 60 nemôžeme zaradiť medzi lacné výrobky a hodnota funkcie je blízka 0 [11].

Vytvorme funkciu veľmi lacný, potom hodnoty danej funkcie pre cenu 10 eur bude 0.98 a pre cenu 27 to bude 0.5. Potom výrobok s cenou 40 a viac už vôbec nemôžeme priradiť do skupiny veľmi lacných.

Ako je vidieť z príkladu, využívanie fuzzy logiky umožňuje určiť netriviálne rozhodovanie o priradení výrobku do nejakej skupiny. Uvedený postup je však pre mnohé praktické príklady veľmi zložitý. Omnoho jednoduchšie je rozhodovanie pomocou trojstavovej logiky, kde -1 by znamenalo, že výrobok je drahý, 0 by znamenalo, že výrobok nie je drahý, ani lacný a 1 by znamenalo, že je lacný. V prípade, že vytvoríme pravdepodobnostnú krivku nad množinou ceny, potom interval logickej hodnoty 0 označuje výrobok, ktorý nie je drahý, ale nie je ani lacný [11].



Obrázok 24: Pravdepodobnostná hodnota trojstavového prvku [11]

Využívanie tritov v takýchto a podobných úlohách môže zjednodušiť spôsob rozhodovania, zrýchliť vyhľadávanie optimálne ceny výrobku pri výbere nedrahých materiálov, z ktorých je vyrobený. Okrem výberu materiálu je možné vedený, zjednodušený výpočet použiť aj pri výstavbe, vyhľadávaní kvázi optimálnej cesty a podobne [11].

4. Riešenie úlohy

V tejto kapitole si ukážeme reálne riešenia problémov pomocou trojkovej sústavy a priblížime si jej výhody. Ternárna sústava je čoraz rozšírenejšia a preto sa ňou okrem matematikov začínajú zaoberať aj firmy, ktoré pracujú s výpočtovou technikou.

4.1 SWOT analýza trojkovej sústavy

SWOT analýza je bežne používaná a dôležitá technika na analýzu interného a externého prostredia s cieľom poskytnúť systematický prístup a podporu rozhodovania. Tento prístup zahŕňa systematické myslenie a komplexnú diagnostiku faktorov súvisiacich s novým produktom, technológiou, manažmentom alebo plánovaním. SWOT, všeobecne známy v oblasti strategického riadenia, analyzuje príležitosti a hrozby identifikované hodnotením vnútorného prostredia, ako aj silné a slabé stránky hodnotením vonkajšieho prostredia. SWOT maximalizuje silné stránky a príležitosti a minimalizuje slabé stránky a hrozby. Inými slovami, premieňa slabé stránky na silné stránky a hrozby na príležitosti. SWOT analýza neposkytuje analytické prostriedky na určenie relevantnej dôležitosti alebo schopnosť posúdiť vhodnosť alternatívny rozhodovania založenú na rôznych faktoroch [23].

Aj keď táto analýza sa tvorí najmä v podnikoch, vieme ju použiť rovnako aj na analyzovanie určitej technológie. My si túto analýzu skúsime použiť na ternárnej sústave a aspoň čiastočne si priblížiť jej vnímanie.

Obrázok 25: SWOT analýza [23]

Pri SWOT analýze môžu byť určité neistoty, s ktorými sa stretávame pri hodnotení vnútorného, či vonkajšieho prostredia. Dá sa povedať, že je nemožné, aby príležitosti a hrozby vyplývajúceho z externého prostredia firmy boli vždy určité v akomkoľvek stave. V dôsledku toho je tiež nemožné presne zmerať číselné hodnoty, ale vieme povedať, že keď rozšírime možnosti oproti binárnej sústave o jednu ďalšiu možnosť, vedelo by to priniesť isté výhody.

Výhody

- Ternárny počítač by vedel držať rovnaké množstvo informácií v menšej pamäti;
- Vedel by rozšíriť počet možnosti o jednu, oproti binárnemu systému ;
- Väčší objem dát cez jeden rozhodovací strom;
- Tento numerický systém by vedel posunúť moderné technológie oveľa ďalej s výkonnosťou, čím by sa otvorilo veľa nových možností.

Nevýhody

- Nekompatibilita s binárnymi systémami, čiže zložitá implementácia;
- nemuselo by to byť také jednoduché na chápanie ako binárny systém, keďže lepšie ľudia rozumejú áno/nie, aj keď ľudské vnímanie sa neuzatvára len na dve možnosti;
- v prípade, že by sa technologický priemysel rozhodol smerovať k ternárnemu systému v počítačoch, bolo by to neskutočne drahé na prechod, respektíve vývoj všetkých systémov na túto sústavu.

Príležitosti

- eliminácia nadbytočného využívania pamäte a rozšírenie možností práce s pamäťou;
- výkonnejšie technológie a technologický pokrok v modernom svete.

Hrozby

- nekompatibilita s modernými technológiami;
- veľmi náročná technologická implementácia ternárnej sústavy do digitálnych zariadení [23].

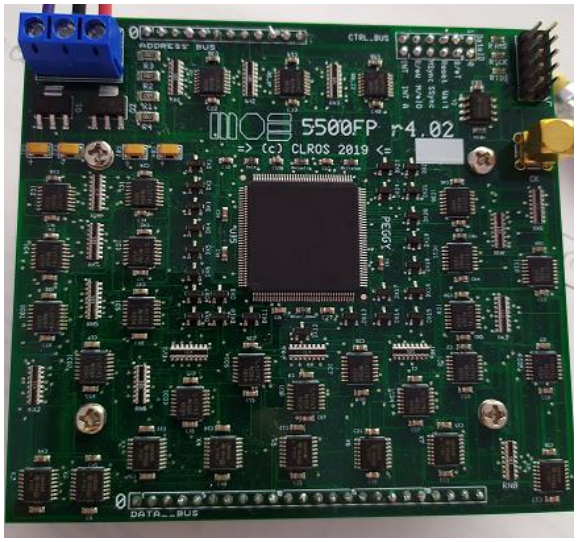
4.2 Prototyp ternárneho počítača

Na prekonanie Moorovho zákona je potrebné používať inovatívne technológie. Tento problém riešime zvýšením hustoty informácií. Z tohto dôvodu by bolo vhodné použiť ternárny systém namiesto binárneho. V istom prototype tohto systému dosiahli výsledok, kde kapacita tohto prototypu bola viac ako sedemdesiat miliárd krát väčšia, ako komerčný 32-bitový procesor. Tento prototyp má ternárnu architektúru RISC s nasledujúcimi vlastnosťami:

- 24 tritová zbernica údajov
- 12 tritová zbernica adries
- 10x10 centimetrová doska procesora
- Ternárne signály na externej zbernici údajov
- RISC ternárnu architektúru

Okrem toho, že je tento systém zaujímavý pre obrovské množstvo informácií a ich zberanie v ternárnych zberniciach nižších rozmerov, môže byť aplikovaný aj na vznikajúce oblasti informatiky vďaka ternárnym údajom, ako napríklad v:

- Umelej inteligencii a deep learningu (aplikácia pre učenie úloh umelých neurónových sietí), pretože ľudský mozog skôr rozmýšľa ternárne, ako binárne;
- Pokročilé a inovatívne výpočty;
- Priemyselná a antropomorfná (prenášanie ľudských vlastností) robotika;
- Automobilový priemysel;
- Výskum algoritmov [24].



Obrázok 26: Ternárny procesor [24]

4.3 Ternárne kódovanie údajov

Vyvážená ternárna číslica, nazývaná aj trit, má hodnoty -1, 0, 1. Tieto čísla môžu byť kódované binárne ako 11, 00 a 01 na priame použitie v digitálnych obvodoch. Preto sa v tejto časti pozrieme na dekompresiu sekvencie bitov do sekvencie binárne kódovaných vyvážených ternárnych jednotiek číslic.

Ternárne kódovanie údajov sa ukázalo ako užitočné napríklad vo výpočtovej technike na všeobecné použitie, na bezdrôtový prenos, reprezentáciu textúr v obrázkoch, v kvantových výpočtoch, optických superpočítačoch a umelých neurónových sieťach [25].

V mnohých aplikáciách binárna hodnota predstavuje postupnosť (-1, 0, 1), ktorú je potrebné uložiť do pamäte. Preto je potrebné nájsť kódovanie, ktoré minimalizuje kompresiu a dekompresiu. Aby sme to dosiahli, budeme implementovať VLSI (Very Large-Scale Integration) neurónových sietí využívajúci ternárne váhy, v ktorých hodnoty hmotnosti sa do pamäte zapisujú len raz a čítajú sa takmer nepretržite. V tomto prípade je potrebné vytvoriť sekvenciu binárne zakódovaných vyvážených ternárnych hodnôt zo zakódovanej binárnej sústavy. V tejto časti si ukážeme, že nie je užitočné komprimovať viac ako 5 tritov na 8 bitoch [25].

Ternárna číslica obsahuje $\log_2(3) = 1{,}586$ bitov informácií. Vypočítame maximálny teoretický zisk pamäte, ktorý možno dosiahnuť kompresiou tritov v binárnom systéme.

Tabuľka 16: Zisk pamäte v ternárnej sústave oproti binárnej sústave [25]

trits	bits	gain	free values $(2^{bits} - 3^{trits})$
1	2	0.00%	1
2	4	0.00%	7
3	5	16.67%	5
4	7	12.50%	47
5	8	20.00%	13
6	10	16.67%	295
7	12	14.29%	1909
8	13	18.75%	1631
9	15	16.67%	13085
10	16	20.00%	6487
17	27	20.58%	5077565

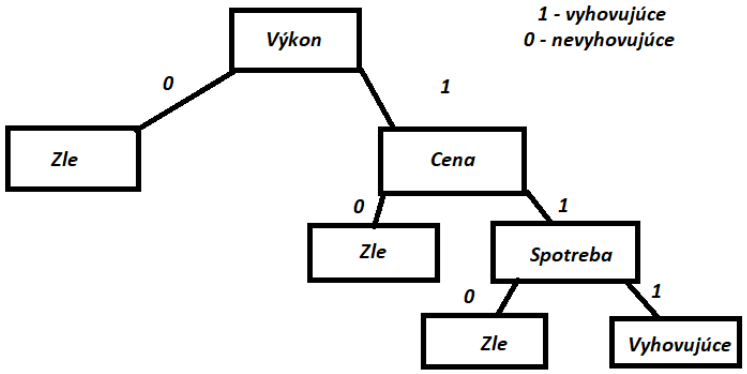
Tabuľka uvádza skutočný zisk (gain) pre malé hodnoty n. Ako možno vidieť, nie je veľký záujem o kódovanie viac sekvencií, ako 5 tritov na 8 bitoch. Najbližší vyšší zisk, oproti 5 tritom, vidíme na 17 tritoch, ale pracovanie s takouto vysokou pamäťou by už bolo veľmi náročné a úplne nepraktické, preto by sme si zvolili 5 tritov [25].

4.4 Porovnanie riešených logických úloh

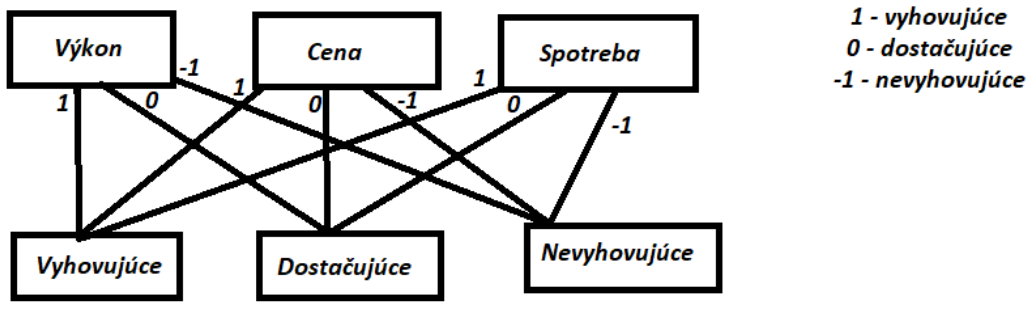
V tejto kapitole si porovnáme riešené úlohy z kapitoly 3.4, ako by vyzerali, keby využívali binárnu a ternárnu logiku a aké výhody by mohla priniesť ternárna sústava v oblasti rozhodovania.

4.4.1 Rozhodovací strom

Prvou úlohou, ktorú sme riešili, bolo rozhodovanie pri parametrickom výbere. Predstavme si teda, že sa rozhodujeme medzi viacerými parametrami, ako sú výkon, cena a spotreba. V binárnej sústave by sme museli mať rozhodovací strom s viacerými uzlami.



Obrázok 27: Rozhodovanie v binárnej sústave [Vlastné spracovanie]



Obrázok 28: Rozhodovanie v ternárnej sústave [Vlastné spracovanie]

Ako vidno, v binárnej sústave, len na posúdenie, ktoré auto by bolo vyhovujúce, bez možnosti dostačujúceho vozidla, musí mať rozhodujúci strom o dve úrovne viac. Keby sme sem chceli zakomponovať dostačujúce vozidlo, strom by sa rozšíril, čo by malo za následok väčšie množstvo využitia pamäte v prípade, že by sme takýto, respektíve podobný princíp preniesli do výpočtovej techniky. V podstate ide o to, že pomocou tej jednej jednotky navyše, na základe ktorej sa môžeme rozhodovať, si vieme ušetriť pamäť, aj rozhodovacie cykly.

4.4.2 Rozhodovanie v medicíne

Rozoberali sme si problém v oblasti medicíny, ktorý sa týkal liečenia choroby, na ktorú existuje viacero liekov s mnohými vedľajšími účinkami. V binárnej sústave by sme si mohli akurát tak zadefinovať, ktorý liek na nejakú chorobu je vhodný a ktorý nie je. Ale nemusíme sa obmedzovať len na dva stavy. Teoreticky by bolo možné tento problém rozvetviť aj v binárnej sústave, ale oveľa jednoduchšie bude použiť v tomto prípade ternárnu sústavu. Vieme si potom zadefinovať, či na nejakú konkrétnu chorobu je vhodný liek (1), či vhodný nie je (-1) alebo či je možné ho nahradiť iným liekom. Rovnako by sme to vedeli používať aj v prípade, či konkrétne lieky je možné spolu kombinovať, alebo nie je.

Tabuľka 17: Porovnanie riešenia v oblasti medicíny [Vlastné spracovanie]

Riešenie v binárnej sústave				Riešenie v ternárnej sústave		
	Liek A	Liek B	Liek C	Liek A	Liek B	Liek C
Liek A	0	1	1	0	-1	1
Liek B	1	0	1	-1	0	-1
Liek C	1	1	0	1	-1	0

V tabuľke vidno, že je praktickejšie použiť ternárnu sústavu, pretože nám to rozšíri možnosti a v praxi by nám to ušetrilo pamäť vo výpočtovej technike. Samozrejme, logicky by sme mohli uvažovať o tom, že kebyže toto rozšírime o ďalšie možnosti, bolo by to ešte oveľa praktickejšie. Pre ozrejmenie, o tomto neuvažujeme práve z dôvodu, že keď sme porovnávali využívanie jednotlivých číselných sústav tak sme zistili, že najvýhodnejší je číselný systém, ktorý sa približuje k Eulerovmu číslu e = 2,718281828459, a to je práve ternárny číselný systém.

4.4.3 Ternárna logika a fuzzy prístup

Táto časť poukazuje na využitie trojhodnotovej logiky spoločne s fuzzy logikou. Predstavíme si príklad, kde máme tri atribúty, cenu, výkon a váhu. Naše požiadavky sú, aby to malo nízku cenu, vysoký výkon a váhu približne tridsať kilogramov:

- Cena pod 200 (vrátane) bude jednoznačne spĺňať požiadavku (1), cena nad 250 (vrátane) jednoznačne nespĺňa požiadavku (-1) a cena medzi 200 a 250 teoreticky spĺňa (0)
- Vysoký výkon znamená, že zariadenia s hodnotou výkonu nad 200 (vrátane), jednoznačne spĺňajú požiadavku (1), výkon pod 150 (vrátane) jednoznačne nespĺňa požiadavku (-1) a medzi 150 a 200 teoreticky spĺňa požiadavku (0)
- Požiadavka, že váha má byť približne tridsať kilogramov znamená, že váha menšia ako 25 kilogramov a väčšia ako 35 kilogramov nespĺňa požiadavku (-1) a váha medzi 25 a 35 kilogramov spĺňa požiadavku (1).

Tabuľka 18: Riešenie príkladu vo fuzzy logike [Vlastné spracovanie]

Zariadenie	Cena [EUR]	Výkon [kW]	Váha [kg]	Výsledok
Z1	300	205	29	0
Z2	200	195	27,5	0,9
Z3	125	190	31	0,8
Z4	240	190	30	0,2
Z5	225	210	32,5	0,5

V tejto tabuľke vidíme, že zariadenie Z1 je nevhodné a ostatné zariadenia sú viac menej vhodné, ale ani jedno z nich nie je úplne vyhovujúce. Zriadenia Z2 až Z4 sú v oblasti možno spĺňa požiadavku a zariadenie Z2 sa najviac približuje k vyhovujúcemu.

Tabuľka 19: Riešenie príkladu v ternárnej logike [Vlastné spracovanie]

Zariadenie	Cena [EUR]	Výkon [kW]	Váha [kg]	Výsledok
Z1	-1	1	1	-1
Z2	1	0	1	0
Z3	1	0	1	0
Z4	0	0	1	0
Z5	0	1	1	0

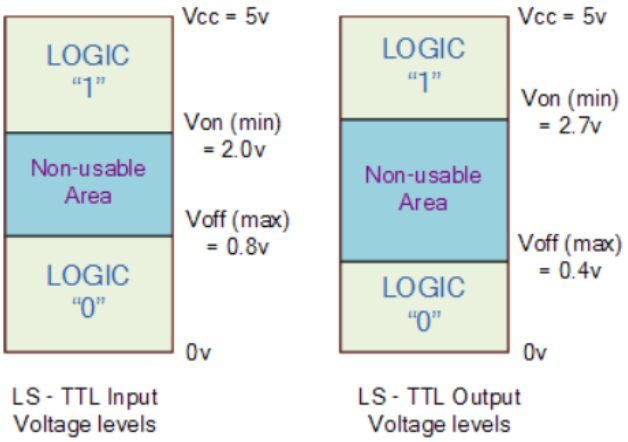
V tejto tabuľke zas vidíme, že zariadenie Z1, rovnako ako v predchádzajúcej tabuľke, je nevyhovujúce a ostatné zariadenie sú rovnako v oblasti možno spĺňa požiadavku.

V podstate sa výsledok pri použití ternárnej logiky nezmenil, ale z tejto tabuľky nie je zrejmé, ktorý z výsledok by bol najviac vyhovujúci, ale z tabuľky ešte vieme vyčítať, že zariadenie Z4, oproti zariadeniam Z2, Z3 a Z5, spĺňa len jednu požiadavku, takže z týchto piatich možností by bolo druhé najmenej vyhovujúce. Ďalej by sme si mohli určiť priority jednotlivých atribútov, či sme ochotný si priplatiť za vyšší výkon, alebo či sme ochotný akceptovať nižší výkon za vyhovujúcu cenu.

Toto je, samozrejme, len ilustračný príklad, ako by sa dala ternárna logika využiť v spojitosti s fuzzy logikou. Možností je veľmi veľa a vo veľa prípadoch by použitie tejto logiky bolo zmysluplnejšie, keďže rátame s tým, že táto logika by bola využívaná v ternárnej počítačovej architektúre. V výpočtovej technike by bolo oveľa zložitejšie pracovať s fuzzy logikou a tak zmena logických fuzzy príkladov na príklady s použitím ternárnej logiky by mohlo byť prínosné.

4.4.4 Ternárna logika v digitálnych obvodoch

Digitálne úrovne momentálne spracúvajú len dva signály, ktoré obsahujú úrovne napätia, respektíve stavy. Tieto dva stavy sú pomenované ako logická nula a logická jednotka. Vo všeobecnosti logická jednotka predstavuje vyššie napätie, ako je napríklad päť voltov, ktoré sa bežne označuje ako vysoká hodnota, zatiaľ čo logická nula predstavuje nízke napätie, ako napríklad 0 voltov alebo uzemnenie a bežne sa označuje ako nízka hodnota. Tieto dve diskrétne úrovne napätia predstavujú digitálne hodnoty, jednotky a nuly, ktoré sa bežne nazývajú binárne čísla. V digitálnych a výpočtových obvodoch a aplikáciách sa bežne označujú ako bity.

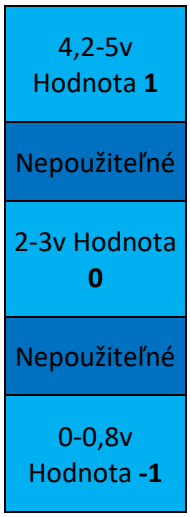


Obrázok 29: Hodnoty vstupného a výstupného napätia [21]

Obrázok naľavo znázorňuje vstupné napätie a rozsah, v ktorom platí logická jednotka (1), ktoré napätie je nepoužiteľné a v ktorom rozsahu napätia platí logická nula(0). Obrázok napravo znázorňuje to isté, ale v prípade výstupného napätia.

Pri použití päť voltového zdroja sa akýkoľvek napäťový vstup medzi dvoma a piatimi voltami rozpozná ako logická jednotka a každý napäťový vstup pod 0,8 sa rozpozná ako logická nula.

Ale vieme si, zatiaľ len na teoretickej úrovni, predstaviť, že by snímače vedeli fungovať aj na ternárnej logike. V prípade, že by úroveň napätie bola nízka, vyhodnotilo by sa to v ternárnej logike ako hodnota -1. V prípade, že by dosahovala nad 2 volty ale pod 3,5 voltov, tak by sa vyhodnotila táto hodnota ako 0 a v prípade, že by dosahovala úroveň nad 3,5 voltov, tak by nadobudla hodnotu 1.



Obrázok 30: Hodnoty napätia v ternárnej sústave [Vlastné spracovanie]

Keby sa ternárna logika použila v digitálnych obvodoch, vedela by sa rozšíriť použiteľnosť na základe vstupného, respektíve výstupného napätia a rozšíriť o jednu možnosť, ako by obvod mohol reagovať, čo by exponenciálne zvýšilo prenášaný počet údajov a informácií v porovnaní s binárnou logikou pre rovnaký počet logických bitov. Ternárny logický systém je jednou z foriem viachodnotovej logiky (MVL).

5. Diskusia

V predchádzajúcich kapitolách sme riešili trojkovú číselnú sústavu, jej porovnanie s binárnou sústavou a možnosti využitia trojkovej sústavy v jednoduchých príkladoch z bežného života.

Pre nasadenie trojkovej sústavy vo výpočtovej technike by bolo potrebné urobiť veľa zložitých krokov pre jej úspešnú a funkčnú implementáciu. V prípade, že by sa dnešný svet rozhodol smerovať k využívaniu trojkovej sústavy v praxi, vedelo by to priniesť veľmi veľa výhod. Tými najpodstatnejšími by boli nárast pamäte, s ktorou by technika vedela pracovať a rozšírenie možností o jednu dodatočnú hodnotu. Keďže ľudské uvažovanie sa skôr približuje k trojhodnotovej ako dvojhodnotovej logike, aj nasadenie tohto číselného systému by bolo zreteľnejšie pre pochopenie ľudskou mysľou.

Momentálne sa v praxi trojková sústava veľmi nepoužíva, preto je náročné riešiť reálne úlohy s jej využitím a museli by prísť rôzne výskumy, ako by táto sústava vedela pomôcť v praxi. Pre približnú predstavu sme si jednotlivé prípady využitia opísali hlavne z hľadiska, ako ľudská myseľ premýšľa v trojkovej sústave a ako by toto vedelo byť implementované v rozhodovacích procesoch výpočtovej techniky.

Tento číselný systém nemusí byť striktne použitý len v rozhodovacích cykloch výpočtovej techniky, ale takisto ho vieme využiť v rôznych iných oblastiach, ako napríklad fuzzy logika alebo digitálne obvody, čomu sme venovali samostatné podkapitoly.

Keď sa pozrieme na hardvérové periférie a zákony fyziky, tak v počítačoch sa používa binárny systém, lebo v minulosti bolo oveľa ťažšie merať a ovládať elektrické signály. V počítači je každé číslo len elektrickým signálom, kde sa rozlišuje stav 1 definovaný záporným nábojom a stav 0 definovaný kladným nábojom. V modernej dobe ale už meranie a ovládanie elektrických signálov nepredstavuje žiadny problém. Preto by sa už mohlo začať viac uvažovať o použití trojkovej sústavy v počítačovej technológii a investovať viac času výskumom, ako by to mohlo ovplyvniť budúcnosť výpočtovej techniky.

Záver

V diplomovej práci bola opísaná trojková sústava, jej história, porovnanie s dvojkovou sústavou a možnosti jej využitia v praxi. Binárna sústava prináša množstvo nedokonalostí a nepresností, ktoré by sa vedeli ľahko odstrániť pomocou trojkovej sústavy, ktorá by bežnú logiku rozšírila o jednu ďalšiu možnosť.

Daný problém je natoľko veľký, že ho začínajú riešiť firmy, ktoré sa zaoberajú výrobou a vývojom počítačov. Tretia možnosť v oblasti rozhodovania prináša nespočetne veľa výhod a táto dodatočná tretia kategória by dokázala odstrániť neurčitosť a v rozhodovacích častiach rozšíriť cykly o jednu jednotku. Práve preto veľké firmy, ako napríklad Intel alebo IBM, sa postupne túto trojkovú sústavu snažia dostať to povedomia ľudí, ktorí sa zaoberajú budúcimi možnosťami počítačov.

Existuje veľmi veľa možností jej využitia. Trojková sústava môže byť aplikovaná v medicíne, vzdelávaní, počítačových systémoch a kdekoľvek, kde úloha a jej riešenie nie je obmedzené len na dva možné výsledky. Prakticky ju vo svojej mysli používame rovnako bežne, ako by sme teoreticky mohli aj v procese rozhodovania. Vedela by nám zjednodušiť chápanie komplexnejších problémov a zrozumiteľní ich riešenia.

Prínos napísania a vytvorenia tejto záverečnej práce spočíval v poskytnutí informácií o trojkovej sústave, jej možnému využitiu a dostať jej výhody do povedomia. Momentálne sa v bežnom modernom svete málokedy stretneme s trojkovou sústavou vo výpočtovej technike. Pritom bežné ľudské vnímanie nie je obmedzené len na dve možnosti, s ktorými sa bežne stretávame. Už len zvýšenie možností rozhodovania a operácií o jednu číslicu, respektíve proces na tej istej rozhodovacej úrovni by nám mohlo pomôcť posunúť výkonnosť výpočtovej techniky vpred.

Zoznam použitej literatúry

[1] BROUSENTSTOV, N. P.; MASLOV, S.P.; RAMIL, ALVAREZ, J.; ZHOGOLEV, E. A.. Development of ternary computers at Moscow State University. [online]. 2011. [cit. 2022.01.15]. Dostupné na internete: https://www.computer-museum.ru/english/setun.htm

[2] JONES, Douglas. The Ternary Manifesto. [online]. 2012. [cit. 2022.01.15]. Dostupné na internete: https://homepage.cs.uiowa.edu/~dwjones/ternary/

[3] KULTAN, Jaroslav. Three-state logic-trit. Open Online Journal for Research and Education. 2018. s. 1-7. [cit. 2022.01.17]. ISSN 2313-1640.

[4] BROWN, David. Why not Ternary Computers? [online]. 2021. [cit. 2022.01.17]. Dostupné na internete: https://www.techopedia.com/why-not-ternary-computers/2/32427

[5] GLUSKER, Mark. The ternary calculating machine of Thomas Fowler. [online] 2005. [cit. 2022.01.20]. Dostupné na internete: https://www.mortati.com/glusker/fowler/about.htm

[6] HAYES, Brian. Third base. American Scientist. Sigma Xi, the Scientific Research Society. [online]. 2001. [cit. 2022.01.20] Dostupné na internete: https://web.williams.edu/Mathematics/sjmiller/public_html/105Sp10/addcomments/Hayes_ThirdBase.htm

[7] JONES, Douglas. Standart Ternary Logic. [online]. 2012. [cit. 2022.01.20]. Dostupné na internete: https://homepage.cs.uiowa.edu/~dwjones/ternary/logic.shtml

[8] JONES, Douglas. Heptavintimal Encoding of Ternary Values. [online]. 2012. [cit. 2022.02.02]. Dostupné na internete: https://homepage.cs.uiowa.edu/~dwjones/ternary/hept.shtml

[9] WINAND, Markus. The Three-Valued Logic of SQL. [online]. 2016. [cit. 2022.02.02]. Dostupné na internete: https://modern-sql.com/concept/three-valued-logic#logical-operations

[10] KUMAR, A. Sathish et. al.. International Journal on Computer Science and Engineering. [online]. 2010. [cit. 2022.02.15]. Dostupné na internete: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.302.1808&rep=rep1&type=pdf

[11] KULTAN, Jaroslav. Logické systémy pri riešení ekonomických úloh. [online]. 2018. [cit. 2022.02.15]. Dostupné na internete: https://fhi.euba.sk/www_write/files/veda-vyskum/konferencie/aiesa/AIESA_2018/Zbornik_AIESA2018.pdf

[12] MARCEL, M. Fuzzy logika. [online]. 2012.[cit. 2022.02.20] Dostupné na internete: http://marcelm.szm.com/

[13] MUSFIGHT. Čo je fuzzy logika: princíp činnosti, príklady, aplikácia. Prehľad softvéru fuzzy logiky. [online]. 2019. [cit. 2022.03.10]. Dostupné na internete: https://musfight.ru/sk/audio-device/chto-takoe-nechetkaya-logika-fuzzy-logic-princip-raboty-primery-primenenie-obzor/

[14] CONNELLY, J. Ternary Computing Testbed 3-Trit Computer Architecture, Cal. Poly., San Luis Obispo, [online]. 2008. [cit. 2022.03.10]. Dostupné na internete: https://www.scribd.com/document/78370674/Ternary-Computing-Testbed-3-Trit-Computer-Architecture

[15] AKULOV, Arkady. Intel processors will become ternary. [online]. 2017. [cit. 2022.03.15]. Dostupné na internete: https://sudonull.com/post/71965-Intel-processors-will-become-ternary-Intel-Blog

[16] JONES, Douglas. Standart Ternary Logic. [online]. 2012. [cit. 2022.03.20]. Dostupné na internete: https://homepage.cs.uiowa.edu/~dwjones/ternary/arith.shtml

[17] BABJAK, Jozef. Ako pracujú fuzzy systémy. [online]. 2003. [cit. 2022.03.20]. Dostupné na internete: https://www.root.cz/clanky/ako-pracuju-fuzzy-systemy/

[18] Knuth, Donald. The Art of Computer Programming - Volume 2: Seminumerical Algorithms. Addison-Wesley, 3rd ed., 1998. ISBN 0-201-89684-2. Dostupné na: http://jeff.tk/wiki/Image:KnuthTaoCPVol2-pg207%2C8.pdf

[19] JONES, Douglas. Fast Ternary Multiplication and Division. [online]. 2012. [cit. 2022.04.01]. Dostupné na internete: https://homepage.cs.uiowa.edu/~dwjones/ternary/multiply.shtml

[20] KAK, Subhash. On Ternary Coding and Three-Valued Logic. [online]. 2019. [cit. 2022.04.01]. Dostupné na internete: https://arxiv.org/ftp/arxiv/papers/1807/1807.06419.pdf

[21] SHAKYA, Udayan. Binary Numbers and the Working of Computers. [online]. 2020. [cit. 2022.04.02]. Dostupné na internete: https://www.programiz.com/blog/working-of-binary-numbers-in-computers/

[22] Phuong, Nguyen Hoang; Kreinovich, Vladik. Fuzzy Logic and its Applications in Medicine. [online]. 2000. [cit. 2022.04.05]. Dostupné na internete: http://digitalcommons.utep.edu/cs_techrep/498

[23] SÁKAL,P. a kol. Strategický manažment v praxi manažéra, Trnava : SP

SYNERGIA, 2007, 703 s. [cit. 2022.04.10]. ISBN 978-80-89291-04-5

[24] LA ROSA, Claudio. Ternary Computer System. [online]. 2019. [cit. 2022.04.10]. Dostupné na internete: https://www.ternary-computing.com/

[25] Olivier Muller, Adrien Prost-Boucle, Alban Bourge, Frédéric Pétrot. Efficient Decompression of Binary Encoded Balanced Ternary Sequences. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. [online]. 2019. [cit. 2022.04.10]. Dostupné na internete: https://hal.archives-ouvertes.fr/hal-02103214