Tomi Lundberg

Liukulukujen vaihtoehtoisia esitystapoja

Tietotekniikan kandidaatintutkielma

16. joulukuuta 2015

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Tomi Lundberg

Yhteystiedot: tomi.t.lundberg@student.jyu.fi

Ohjaaja: Sanna Mönkölä

Työn nimi: Liukulukujen vaihtoehtoisia esitystapoja

Title in English: Alternative representations for floating-point numbers

Työ: Kandidaatintutkielma

Sivumäärä: 19+0

Tiivistelmä: Tässä työssä tarkastellaan IEEE:n liukulukustandardista poikkeavia liukulukujen esitystapoja. Tarkastelun kohteina ovat yksinkertainen murtolukujärjestelmä ja erikoiskäyttöön hyvin soveltuva logaritmijärjestelmä. Edellä mainittuja lukujärjestelmiä verrataan standardiin ja lopuksi tarkastellaan erästä standardijärjestelmää sekä logaritmijärjestelmää hyödyntävää hybridilukujärjestelmää.

Avainsanat: Liukuluku, lukujärjestelmä, murtoluku, logaritmi

Abstract: This work inspects the floating-point representations which deviate from the IEEE standard for floating-point numbers. Subjects of investigation are a simple slash number system and a logarithmic number system designed for special aplications. The two systems are compared agains the standard system and finally a case of hybrid number system is inspected.

Keywords: Floating-point number, number system, fraction, logarithm

Sisältö

1 Johdanto

Tänä päivänä suurin osa liukulukulaskennasta tapahtuu sellaisilla laitteistoilla, ohjelmilla tai niiden yhdistelmillä, jotka noudattavat ainakin osittain vuonna 1985 julkaistua ja vuonna 2008 päivitettyä standardia IEEE-754 (IEEE 785 2008) liukuluvuista ja niiden aritmetiikasta. Ennen standardia olemassaoloa useat eri tahot, erityisesti laitevalmistajat, määrittelivät oman liukulukuaritmetiikkansa. Nämä erilaiset liukulukuaritmetiikat eivät ole olleet yhteensopivia, eivätkä aina edes välttämättä tarkkoja.

Reaalilukujen esitys tietokoneessa on ollut eräs tietotekniikan peruskysymyksistä jo vuosikymmeniä. Miten esittää mahdollisimman hyvin äärellisessä muistissa lukuja, joille ei ole olemassa päättyvää tai edes jaksollista desimaalikehitelmää? Mikä lasketaan riittäväksi tarkkuudeksi, kun tiedetään, että lukua ei voida mitenkään esittää tarkasti tietokoneessa? Entä miten toimitaan sellaisten aritmeettisten operaatioiden jälkeen, kun tulos päättyy kahden vierekkäisen luvun välille? Edellä mainitut kysymykset ovat vain osa liukulukuihin liittyvistä ongelmista.

Liukulukulaskenta on vaativaa niin teoreettisesti kuin käytännön kannalta, kuten edellisessä kappaleessa esitetyt kysymykset osoittavat. Teoreettisesti liukuluvut pohjautuvat reaalilukuihin ja monet tulokset johdetaankin reaalianalyysistä. Reaalilukujen joukko on kuitenkin äärettömän suuri ja tästä syystä tietokoneessa äärellisellä tallennustilalla ei ikinä kyetä esittämään kaikkia reaalilukuja. Monimutkaisemmaksi asian tekee vielä se, että reaalilukujen liitäntälait ja osittelulaki eivät päde yleisesti liukuluvuille. Laskujärjestyksellä saattaa siis olla huomattavaa merkitystä sovelluksissa, mikäli asiaa ei osata ottaa huomioon.

Standardi IEEE-754 määrittelee:

- viisi perustavaa liukulukuesitystä, kaksikantaisen 32-, 64- ja 128- bittisen esitystavan sekä kymmenkantaisen 64- ja 128-bittisen esitystavan.
- Aritmeettisia operaatioita näille esitystavoille, kuten yhteen-, vähennys-, kerto- ja jakolaskun, neliöjuuren, suuruuden vertailun ja muita.

- Muunnokset kokonaislukujen ja liukulukujen välillä, eri liukulukuesitysten välillä sekä liukulukujen ja merkkijonojen välillä.
- Liukulukujen poikkeukset.

Asioiden nimitykset hieman vaihtelevan englannin kielen ja suomen kielen välillä, joten tässä tutkielmassa käytetään seuraavia nimeämiskäytänteitä:

- desimaaliesitys, engl. *floating-point*
- murtolukuesitys, engl. *slash number system*
- logaritmiesitys, engl. *sign/logarithm number system*

Erityisesti termi *floating-point* tuottaa kääntämisessä hankaluuksia, sillä kun voidaan tarkoittaa niin liukulukuja reaalilukujen osajoukkona kuin IEEE:n standardin määrittelemää liukulukuesitystä. Termiä *liukuluku* käytetään ensimmäisessä tarkoituksessa ja termiä *desimaaliesitys* jälkimmäisessä tarkoituksessa. Lisäksi termiä *murtoluku* käytetään silloin, kun kyseessä on murtolukuesityksen mukaisesti esitettävä luku ja termiä *rationaaliluku* silloin, kun kyse on rationaaliluvusta matemaattisessa merkityksessä.

Edellä esitetyt esitystavat eivät ole ainoita mahdollisia liukulukuesityksiä. Tämän työn ulkopuolelle jäävät esimerkiksi kiintolukuesitys (engl. *fixed-point*) ja jäännöslukuesitys (engl. *residual number system*).

Tämän työn tarkoituksena on tutkia kirjallisuuskatsauksella kahta standardista poikkeavaa liukulukujen esitystapaa yleisluonteisesti. Lisäksi haetaan vastausta kysymykseen, miksi IEEE valitsi standardikseen juuri tämän nykyisen esitystavan. Tässä työssä keskitytään ensisijaisesti matalan tason (ns. bittitason) toteutuksiin.

Tämä tutkielma rakentuu seuraavasti. Luvussa 2 tarkastellaan murtolukuihin pohjautuvaa esitystä. Luvussa 3 tarkastellaan, miten liukulukuja voidaan esittää logaritmin avulla. Luvussa 4 vertaillaan yleisellä tasolla näitä kahta esitystä desimaaliesitykseen. Luvussa 5 käydään läpi maininnan tasolla muita esitysmuotoja. Lopuksi luku 6 tiivistää tutkielman löydökset.

2 Murtolukuesitys

Tässä luvussa esitellään murtolukuesityksen kaksi variaatiota sillä tavalla kuin ne on alunperin määritelty artikkelissa (Matula 1975, s. 90). Esitysten aiottuja toteutuksia käydään läpi yleisluontoisesti, mutta pääpaino on esitysten yleisissä ominaisuuksissa.

Mielivaltaista reaalilukua voidaan approksimoida rationaaliluvulla, koska niin rationaali- kuin reaaliluvuilla on tiheysominaisuus. Esimerkiksi kahden rationaaliluvun välissä on aina rationaaliluku. Tiheysominaisuus on reaalilukujen yksi keskeisimmistä ominaisuuksista ja matemaattisen analyysin kulmakiviä.

Murtolukuesitystä on määritelty kahdenlaista, staattista (engl. *fixed-slash*) ja dynaamista (engl. *floating-slash*) (Matula 1975; Matula, Kornerup 1978, s. 90; s. 30 ja 32). Molemmissa esityksissä liukuluku esitetään kahden kokonaisluvun avulla, yksi luku osoittajana ja yksi nimittäjänä. Esitystavat eroavat toisistaan sisäiseltä toteutukseltaan esim. lukualueiltaan, mutta niissä on samoja matemaattisia ominaisuuksia.

Staattisessa murtolukuesitysessä luku esitetään $2N + 2$ bitillä, missä $N \in \mathbb{N}$, $N \geq 2$ ja N ilmaisee osoittajaa tai nimittäjää esittävän kokonaisluvun bittien määrän (Matula, Kornerup 1978, s. 30). Yhtä bittiä käytetään etumerkille ja yhtä bittiä käytetään tarkistusbittinä. Yksi tarkistusbitin tehtävistä on se, että näin luku on esitettävissä parillisella määrällä bittejä riippumatta luvun N valinnasta (Matula, Kornerup 1978, s. 31).

Esitys on täysin yhteensopiva kokonaislukujen kanssa täsmälleen siinä tilanteessa, kun nimittäjässä on luku yksi (Matula, Kornerup 1978, s. 31 ja 32). Tällöin staattista murtolukuesitystä voidaan käyttää esittämään N bittistä kokonaislukua ja murtolukuesitystä voikin siten pitää kokonaislukujen laajennoksena (Matula, Kornerup 1978, s. 31 ja 32).

Staattisessa murtolukuesityksessä on mahdollista bittitasolla suorittaa oikealle siirtoa samanaikaisesti osoittajassa ja nimittäjässä niin kauan, kunnes vähintään osoit-

tajassa tai nimittäjässä on vähiten merkitsevän bitin kohdalla nollabitti (eli vähintään toinen luvuista on pariton) (Matula, Kornerup 1978, s. 30). Tämä oikealle siirto vastaa kakkosella supistamista. Tämä ei luonnollisestikaan riitä supistamisessa, mutta se on nopea tapa supistaa kaksikantaisessa lukujärjestelmässä luvun kaksi monikerrat.

Dynaaminen murtolukuesitys toimii staattisen murtolukuesityksen laajennoksena. Se on rakenteeltaan monimutkaisempi ja samalla ilmaisuvoimaisempi staattiseen murtolukuesitykseen verrattuna. Se käsittää ne murtoluvut, joille pätee $p+q = N-1$, kun p on osoittajan bittien lukumäärä, q on nimittäjän bittien lukumäärä ja esityksessä on käytetty yhteensä N bittiä, joista yksi on varattu etumerkille (Matula, Kornerup 1978, s. 32)

Etumerkin, osoittajan ja nimittäjän kenttien lisäksi sanaan on varattava tilaa myös $\lceil \log_2(N-1) \rceil$ bitin verran murtoviivan paikan esittämiseen (Matula, Kornerup 1978, s. 32). Merkintä $\lceil X \rceil$ tarkoittaa ns. kattofunktioa eli pienintä kokonaislukua, joka on suurempi kuin X. Vaikka dynaamiessa murtolukuesityksessä joudutaankin käyttämään bittejä murtoviivan esittämiseen, se tarjoaa mahdollisuuden esittää staattiseen esitykseen verrattuna itseisarvoltaan suurempia ja pienempiä lukuja. Esimerkiksi kun $N = 8$ jää merkille yksi bitti, ja murtoviivan osoittamiseen $\lceil \log_2(8-1) \rceil = 3)$ bittiä. Tällöin jää neljä bittiä itse luvulle, mikä riittää kaikkien neljäbittisten kokonaislukujen esittämiseen, jolloin kahdeksan bitin dynaamisella esityksellä voidaan esittää samat kokonaisluvut kuin yhteensä kymmenellä bittiä sisältävällä staattisella murtolukuesityksellä.

Esitetyt murtolukuesitykset eivät kuitenkaan ole täysin tasa-arvoisessa asemassa keskenään. Vaikka edellä huomattiinkin, että samalla bittimäärällä staattinen ja dynaaminen esitys kykenevät esittämään samat kokonaisluvut, ei vastaava tulos päde enää esitettäville murtoluvuille. Oletetaan esimerkiksi, että käytössä on kymmenen bittiä kummallakin esityksellä. Staattisessa esityksessä näistä kymmenestä bitistä yksi on etumerkille ja toinen tarkastusbitille, jättäen siten neljä bittiä osoittajalle ja neljä bittiä nimittäjälle. Siten staattisella esityksellä on mahdollista esittää esimerkiksi luku $\frac{1110_2}{1101_2} = \frac{14}{13}$. Dynaamisessa esityksessä on myös yksi bitti varattu etumer-

kille, mutta murtoviivalle kuuluu $\lceil \log_2(9) \rceil = 4$ bittiä. Koska tämän jälkeen on käytössä enää viisi bittiä (teoriassa kuusi, sillä nimittäjässä on implisiittisesti aina johtava ykkösbitti), ei ole mahdollista saada aikaiseksi sekä osoittajaan että nimittäjään yhtäaikaa neljäbittisiä lukuja.

Yksi murtolukuesityksen silmiinpistävä ominaisuus on mahdollisuus tuottaa mille tahansa nollasta poikkeavalle luvulle tarkka käänteisluku nopeasti. Staattisessa esityksessä riittää osoittajan ja nimittäjän lukujen vaihtaminen keskenään. Dynaamisessa esityksessä joudutaan lisäksi ottamaan huomioon murtoviivan paikka (Matula, Kornerup 1978, s. 30 ja s. 31).

Murtolukuesitysten aritmeettisille operaatioille (yhteen-, vähennys-, kerto- ja jakolaskulle) voidaan taata tarkat tulokset käytettäessä laskennassa kaksoistarkkuudellisia rekisterejä, eli rekisterejä, joissa sanan pituus on $2N$, kun osoittajan ja nimittäjän sanan pituus on N bittiä (Matula, Kornerup 1978, s. 29). Kaksoistarkkuus pitää huolen siitä, osoittajan ja nimittäjän kokonaislukuaritmetiikalla saavutetaan tarkkoja tuloksia, joista sitten saattaa olla mahdollista "toipua" sievennyksen myötä niin, ettei yli- tai alivuotoa tapahdu itse murtoluvulle.

Molemmissa murtolukuesityksissä on mahdollisuus esittää poikkeuksia, kuten NaN ja $\pm\infty$ (positiivinen ja negatiivinen äärettömyys). Esimerkiksi staattisessa murtolukuesityksessä äärettömyys esitetään asettamalla nimittäjän kaikki bitit nollaksi ja osoittajan vähiten merkitsevän bitti asetetaan päälle (Matula, Kornerup 1978, s.). Etumerkkibitillä saadaan selvitettyä, onko kyseessä negatiivinen vai positiivinen äärettömyys. Vastaavasti vaihtamalla osoittajan ja nimittäjän rooleja saadaan aikaan nolla kun halutaan ilmaista alivuotoa, eli laskutoimituksen tulos jää niin lähelle nollaa, että tarkkuus ei riitä erottamaan lukua nollasta poikkeavaksi (Matula, Kornerup 1983, s. 10). Poikkeus NaN saadaan vastaavalla tavalla, mutta vähiten merkitsevän bitin tilalle asetetaan nolla.

Sekä yli- että alivuodon äärettömyys ja nolla sekä NaN mahdollistavat lisäksi viestin kuljettamisen. Esimerkiksi alivuodon tapauksessa kun osoittajan kaikki bitit ovat nollia, niin nimittäjässä vähiten merkitsevää bittiä lukuunottamatta voidaan muilla

biteillä tuoda esille erilaisia tarpeelliseksi katsottuja (debug)viestejä (Matula, Kornerup 1983, s. 11). Nämä viestit kuitenkin ovat täysin toteutusriippuvaisia.

Yksi molempien murtolukuesityksien heikkouksista on turha toisto esitettävissä luvuissa. Rationaalilukujen luonteeseen kuuluu, että esimerkiksi

$$\frac{1}{2} = \frac{2}{4} = \frac{4}{8} = \ldots$$

Tällöin murtolukuesitys toistaa samoja lukuja useasti ja esityksen ilmaisuvoima kärsii. On kuitenkin osoitettu, että tämä lukujen toistaminen vie kummaltakin murtolukuesitykseltä arviolta yhden bitin ilmaisuvoimaa pois, tappion pienentyessä käytetyn tietokoneen sanan koon kasvaessa (Matula, Kornerup 1978, s. 31, teoreema 1 ja seuraus 1.1; s. 31, teoreema 2 ja seuraus 2.1).

Aritmeettisten operaatioiden jälkeen joudutaan sieventämään. Sieventäminen vaatii osoittajan ja nimittäjän yhteisten tekijöiden etsimistä ja vaikka tehtävään onkin kehitetty toimivia algoritmeja (esimerkiksi Eukleideen algoritmi kahden luvun suurimman yhteisen tekijän hankkimiseen), niin tekijöiden etsiminen on silti operaationa aikaa vievää erityisesti, kun kysessä olevat numerot ovat itseisarvoltaan suuria.

Valitettavasti lähdemateriaali ei käsittele monimutkaisempia laskutoimituksia, kuten neliöjuurta tai potenssiin korotuksia. Neliöjuuri ja potenssiin korotus tuottavan vastaavia ongelmia kuin muiden lukujärjestelmien kanssa, sillä esimerkiksi neliöjuuren ottaminen kokonaisluvusta ei useinkaan tuota kokonaislukua. Tällöin joudutaan turvautumaan erilaisiin likiarvoihin, jolloin perusoperaatioiden eli yhteen-, vähennys-, kerto- ja jakolaskun lupaamat tarkat tulokset kärsivät.

3 Logaritmiesitys

Tässä luvussa esitetään kaksi erilaista toteutusta logaritmiesitykselle. Vaikka lähteenä käytetyt artikkelit on kirjoitettu kolmenkymmenen vuoden erolla toisistaan, niissä on huomattavia samankaltaisuuksia. Lisäksi molemmissa esityksissä on sisäisestä toteutustavasta riippumattomia yleisiä ominaisuuksia, jotka ovat tämän kirjoitelman kannalta kiinnostavia.

Logaritmiesitykseen ei ole olemassa standardia. Eri esityksen toteutustavoilla on yhteisiä piirteitä, kuten logaritmifunktion ominaisuuksien hyödyntäminen nopean kerto- ja jakolaskun saavuttamiseksi. Logaritmiesitystä ei ole suunniteltu muunlaisten liukulukuesitysten korvaajaksi, vaan tiettyjen sovellusalueiden (kuten hahmontunnistuksen) vaatimaan laskentaan (Swartzlander, Alexopoulos 1975, s. 1238).

Logaritmillä on aina kantaluku. Yleisimmät logaritmifunktiot ovat niin sanottu luonnollinen logaritmi, jonka kantalukuna on irrationaalinen Neperin luku, ja kymmenkantainen (Gribbsin) logaritmi. Tässä työssä käsiteltävässä logaritmiesityksessä on kantalukuna luku kaksi. Kaksikantainen logaritmi on luonnollinen valinta kaksikantaisessa lukujärjestelmässä, jollaisella tietokone sisäisesti toimii.

Yleisen logaritmifunktion lähtöjoukko on positiivisten reaalilukujen joukko ja maalijoukona on reaalilukujen joukko. Logaritmifunktio on siis määritelty ainoastaan positiivisille reaaliluvuille, riippumatta kantaluvun valinnasta. Koska myös negatiiviset luvut ovat laskennan kannalta merkityksellisiä, on eri toteutuksessa otettu negatiiviset luvut huomioon pääsääntöisesti erillisen etumerkkiä esittävän bitin avulla. Artikkelissa (Swartzlander, Alexopoulos 1975) esitellään toteutus, missä etumerkin avulla määritetään, onko luku positiivinen vai negatiivinen. Lisäksi kyseisessä toteutuksessa käytetään toteutuskohtaista vakiota varmistamaan, ettei jouduta ongelmiin negatiivisten logaritmien kanssa. Artikkelissa (Lee, Lever 2003) esitelty toteutus käyttää myös etumerkkiä desimaaliesityksen tavoin, mutta siinä on luovuttu hankalan toteutuskohtaisen vakion käytöstä. Sen sijaan mahdolliset negatiiviset lukuarvot kuvataan arvoiksi, jotka ovat pienempiä kuin yksi.

Aiemmin mainittu logaritmifunktion kertolaskun yksinkertaisuus ja nopea suoritus perustuvat logaritmifunktion ominaisuuteen

$$log(XY) = log(X) + log(Y), \tag{3.1}$$

missä X ja Y ovat positiivisia kokonaislukuja.Tämän ominaisuuden vuoksi kertolasku palautuu yksinkertaiseksi yhteenlaskuksi. Tulon merkki saadaan kertojan ja kerrottavan merkkien biteistä bittitason XOR(poissulkeva looginen tai)-operaatiolla. Kaavan 3.1 yhteenlasku on tarkka operaatio, mutta siinä on mahdollisuus ylivuotoon(Lee, Lever 2003, kappale 3.1). Ylivuoto on kuitenkin mahdollista havaita ja asettaa tarvittavat liput äärettömyydelle, jota yleensä käytetään ylivuodon tapahtuessa.

Jakolasku on hyvin samankaltainen kertolaskun kanssa, sillä erolla että

$$log(\frac{X}{Y}) = log(X) - log(Y),$$

missä $X, Y \in \mathbb{R}_+$. Kuten kertolaskussa, myös jakolaskussa tuloksen merkki saadaan jaettavien lukujen XOR-operaatiosta. Muuten erona on, että ylivuodon sijasta voi tapahtua alivuoto (Lee, Lever 2003, kappale 3.2).

Logaritmifunktion suuri vahvuus kerto- ja jakolaskun lisäksi on potenssiin korotuksen yksinkertaisuus. Logaritmifunktion laskusäännöistä saadaan

$$log(X^a) = a \times log(X).$$

Tällä viattoman näköisellä kaavalla on suuri vaikutus potenssiin korotukseen ja siten juurenottoon (seurausta siitä, että juurenotto voidaan muuntaa murtopotenssiksi). Esimerkiksi n. juuren ottaminen logaritmista saadaan muotoon $log(\sqrt[n]{X}) = log(X^{\frac{1}{n}}) = \frac{1}{n} \times log(X)$.

Logaritmin luonteen vuoksi logaritmiesitys on aritmeettisista operaatioista nimenomaan kertolaskun suhteen tehokas. Valitettavasti näin ei ole yhteenlaskun suhteen. Yhteenlasku määritelläänkin logaritmiesityksessä kertolaskun avulla.

Määritellään lukujen A ja B olevan logaritmiesityksellä esitettäviä lukuja, jolloin nii-

den yhteenlasku on määritelty kaavalla:

$$A + B = A(1 + \frac{B}{A}),$$

joka vielä voidaan esittää muodossa

$$A\ \psi(\frac{B}{A}), \tag{3.2}$$

missä funktio ψ määritellään kaavalla $\psi(X) = 1 + X$ (Swartzlander, Alexopoulos 1975, s. 1240). Kahden logaritmiesityksellä esitettävän luvun summan laskemiseen kuuluu siis kaavan 3.2 nojalla jakolasku, funktion ψ arvon laskeminen ja vielä lopuksi kertolasku.

Yhteen- ja vähennyslaskussa käytettävä funktio ψ voidaan laskea usealla eri tavalla. Yksi vaihtoehto on käyttää ROM-muistia, mutta se on epäkäytännöllistä tietokoneen käyttämän sanan koon kasvaessa nopeasti kasvavan muistin tarpeen vuoksi (Swartzlander, Alexopoulos 1975, s. 1941). ROM-muistin lisäksi funktion arvo on myös mahdollista laskea interpoloimalla epälineaarista funktiota (Lee, Lever 2003, kappale 3.3). Vähennyslaskussa on myös huomioitava se, että laskun tulos voi olla nolla. Logaritmifunktio on määritely vain positiivisille reaaliluvuille, joten tarvitaan jonkinlainen erityisjärjestely asian hoitamiseksi. Nollan logaritmiksi voidaan sopia esimerkiksi $-\infty$, mutta kyseinen erikoistapaus on kyettävä huomaamaan laskennan aikana.

Logaritmiesitys ei sovellu hyvin pelkästään kokonaisluvuilla suoritettaviin laskutoimituksiin vakioidun suhteellisen virheen takia (Swartzlander, Alexopoulos 1975, s. 1941). Tästä syystä logaritmiesitystä ei ole tarkoitettu yleiskäyttöiseen (kokonaisluku)laskentaan, vaan se on parhaimmillaan laskutoimituksissa, joissa on runsaasti kertomista, jakamista, neliöjuuren ottamista sekä potenssiin korottamista.

4 Vertailua

Tässä luvussa tarkastellaan luvussa 2 esitettyä murtolukuesitystä ja luvussa 3 esitettyä logaritmiesitystä desimaaliesitykseen. Koska erityisesti logaritmiesityksestä ei ole yksikäsitteistä esitystapaa, niin vertailussa pysytään esitysten yleisissä ominaisuuksissa. Kuitenkin lähdeartikkelien toteutuskohtaiset analyysit osoittavat suuntaantavasti yleisiä piirteitä ja eroja esitysten välillä, joten joihinkin toteutuskohtaisiin yksityiskohtiin on pakko tutustua.

4.1 Desimaaliesitys vs. murtolukuesitys

IEEE:n standardi määrittää, että kantaluvun kaksi desimaaliesitys on yksikäsitteinen, eli jokainen desimaaliesityksen luku kuvautuu eri luvuksi. Näin ei kuitenkaan murtolukuesityksessä ole, vaan useat bittiyhdistelmät tuottavat tulkittuna (ja sievennettynä) saman luvun. Vaikka murtolukuesitysksessä kyetään osoittamaan tämän häviön lähenevän yhtä bittiä sanan koon kasvaessa, niin silti desimaaliesityksessä kyetään esittämään suurempaa lukualuetta, kun molemmilla esityksillä on käytössä saman verran bittejä.

Murtolukuesityksen peruslaskutoimitukset tarvitsevat laskennalliseti vähemmän resursseja kuin desimaaliesityksen peruslaskutoimitukset, johtuen murtolukuesitykseen liittyvien algoritmien olevan operaatioita kokonaisluvuilla. Kuitenkin lähdeartikkelista puuttuvat edistyneemmät laskutoimitukset kuten neliöjuuren ottaminen, joten näiden operaatioiden suhteen ei pystytä tekemään reilua vertailua.

Myös laskutoimitusten tarkkuuksissa on eroja. Desimaaliesityksessä saattaa käydä niin, että suureen luvun ja verrattain pienen luvun yhteenlaskussa voi pieni luku hävitä tuloksesta. Tämä johtuu laskentatarkkuudesta, koska desimaaliesityksessä lukujen välit kasvavat suuriin lukuihin mentäessä, joten pienet lisäykset saattavat pyöristää tuloksen toisena operandina olevaan lukuun eikä todelliseen tulokseen. Murtolukuesityksessä ei peruslaskutoimitusten kohdalla ole näin, kun oletetaan, ettei tulos johda poikkeuksiin, vaan pienen ja suuren luvun yhteenlasku tuottaa aina

oikean tuloksen.

Laskutoimitusten nopeudesta on sen sijaan hankalaa sanoa mitään yleispätevää. Desimaaliesityksestä tiedetään, että kaukana toisistaan olevien lukujen laskutoimitukset saattavat viedä pitempään, kuin jos luvut olisivat lähekkäin. Murtolukuesityksessä sen sijaan saattaa eteen tulla sievennys sekä sitä kautta osoittajan ja nimittäjän suurimman yhteisen tekijän laskeminen vastaan. Murtolukuesityksessäkään ei pystytä lupaamaan etukäteen operaatioiden kestoja juurikan sieventämisen takia, sillä hyvässä tapauksessa se ei vie pitkään, mutta todella huonossa tapauksessa kohdataan pitkiä sievennysoperaatioita useasti laskennan aikana.

Käänteislukujen kanssa toiminen on desimaaliesityksessä yllättävän hankalaa. Esimerkiksi luvun 10 käänteisluku $\frac{1}{10}$ ei ole esitettävissä desimaaliesityksessä tarkasti. Yleisesti aina kun kyseessä ei ole jokin luvun kaksi potenssi joudutaan desimaaliesityksessä turvautumaan likiarvioon. Kuitenkin murtolukuesityksessä luvun 10 käänteisluvun hankkiminen on hyvinkin yksinkertaista. Staattisessa murtolukuesityksessä riittää osoittajan ja nimittäjän vaihtaminen keskevään. Dynaamisessa murtolukuesityksessä joudutaan tekemään hieman enemmän työtä murtoviivan paikan laskemisen vuoksi, mutta kyseinen laskutoimitus on hyvin yksinkertainen. Murtolukuesityksen käänteislukuoperaatio ei ole pelkästään helppo ja nopea suorittaa, vaan lisäksi se tuottaa aina tarkan käänteisluvun.

4.2 Desimaaliesitys vs. logaritmiesitys

Desimaaliesitys ja logaritmiesitys voidaan muotoilla yllättävän samankaltaisella tavalla. Periaatteessa logaritmiesityksen voidaan ajatella olevan desimaaliesityksen erityistapaus, kun mantissa on aina 1 (Lee, Lever 2003, kappale 2.2). Tähän seikkaan verrattuna onkin yllättävää, miten erilaiset laskennalliset ominaisuudet esityksillä on.

Logaritmiesityksestä on laskettu operaatioiden aikavaatimukset verrattuna vastaavaan desimaaliesitykseen (Swartzlander, Alexopoulos 1975, s. 1241, taulukko 1). Vertailussa käy ilmi, että logaritmiesityksen kerto- ja jakolasku vie kaksinkertaisen

ajan vastaavan desimaaliesityksen yhteen- ja vähennyslaskuun verrattuna. Kuitenkin logaritmiesityksen yhteen- ja vähennyslasku vie nelinkertaisen ajan desimaaliesityksen yhteen- ja vähennyslaskuun verrattuna.

Kerto- ja jakolaskun tapauksessa tilanne kääntyykin toisinpäin. Desimaaliesityksen kertolasku vie sen omaan yhteenlaskuun verrattuna seitsemänkertaisen ajan. Siten logaritmiesitys on yli kolme kertaa nopeampi kertolaskussaan kuin vastaava desimaaliesitys. Kun vertailu siirretään jakolaskuun, niin logaritmiesitys onkin jo neljä kertaa vastaavaa desimaaliesitystä nopeampi. Vaikka lähdeartikkeleissa ei vertailtu tilannetta esimerkiksi neliöjuuren ottamisessa, logaritmiesitys saattaa olla desimaaliesitykseen verrattuna vielä jakolaskuakin nopeampi, kun otetaan huomioon miten yksinkertaisesti logaritmiesitys muuttaa neliöjuuren jakolaskuksi, kun desimaaliesityksessä joudutaan turvautumaan iteratiivisiin algoritmeihin.

Desimaaliesitys on yleiskäyttöinen, mitä logaritmiesitys sen sijaan ei ole. Tästä syystä logaritmiesitys ei tule saavuttamaan sitä käyttöastetta mitä desimaaliesitys on nyt varsinkin standardoituna saavuttanut. Kuitenkin näiden esitystapojen yhteisestä käytöstä voi tulevaisuudessa tulla suurikin sovellusala. Tätä mahdollisuutta tutkitaan luvussa 5.

5 Hybridiesitys

Tutkittaessa desimaali- ja logaritmiesityksiä huomattiin molempien esitysten toimivan tehokkaasti eri laskutoimituksille, desimaaliesityksen yhteen- ja vähennyslaskuille ja logaritmiesityksen muille aritmeettisille perusoperaatioille. Tästä syystä onkin kiinnostavaa tutkia, voisiko näitä esitystapoja soveltaa samaan laskentaan yhdessä, eikä erikseen kuten on totuttu. Tälläistä useampaa kuin yhtä esitystapaa laskennan aikana käyttävää järjestelmää kutsutaan hybridiesitykseksi.

Vaikka kahdella eri esitystavalla olisi molemmilla käytettävissä sama määrä bittejä, ne silti käyttävät bittejä eri tavalla, eri kohdista (esimerkiksi toinen järjestelmä voisi käyttää "oikeimmanpuolista" bittiä esittämään etumerkkiä, toinen järjestelmä taas "vasemmanpuoleisinta"), eri operaatioilla ja niin edelleen. Tästä johtuen eri esitystavat eivät kykene suoraan operoimaan eri esitystapojen lukuja, vaan luku on käännettävä toisen esitystavan ymmärtämään muotoon.

Artikkelissa (Lee, Lever 2003, kappale 3.5.1) esitetään eräs algoritmi luvun muutamiseen normalisoidusta desimaaliesityksestä logaritmiesitykseen ja päinvastoin. Kääntäminen desimaaliesityksestä logaritmiesitykseen vaatii kolme askelta: Ensin tarkastetaan onko kyseessä jokin desimaaliesityksen poikkeustapauksista. Toiseksi lasketaan käännös $\log_2(1.xxx \times 2^{exp}) = \log_2(1.xxx) + \log_2(2^{exp}) = \log_2(1.xxx) + exp$. Kolmanneksi lasketaan arvo $\log_2(1.xxx)$. Kolmas askel voidaan suorittaa eri tavoin. Esimerkiksi pienelle tietokoneen sanan koolle voidaan käyttää yksinkertaisesti taulukkoa, yksinkertaisen tarkkuuden (32-bittiä) sanan koolle voidaan käyttää likiarvon tuottamaa ROM-muistiin perustuvaa algoritmia ja kaksoistarkkuudelliseen (64-bittiä) sanan kokoon voidaan käyttää mantissaan tekijöihin jakamiseen perustuvaa iteratiivista algoritmia.

Muunnos logaritmiesityksestä desimaaliesitykseen tapahtuu myös kolmessa vaiheessa. Aluksi tarkastetaan, onko kyseessä jokin logaritmiesityksen poikkeustilanteista. Toiseksi otetaan logaritmiesityksen eksponentin kokonaislukuosa ja siirretään se desimaaliesityksen eksponentiksi. Siirtoon kuuluu ns. biaksen lisääminen,

koska logaritmin kokonaislukuosa on säilötty kahden komplementtina. Kolmanneksi kokonaisluku muunnetaan desimaaliesityksen mantissaksi. Tässä muunnoksessa käytetään taulukkoa arvojen etsimiseksi, joten sanan koon kasvaessa käytettävän taulukon koko kasvaa liian suureksi muistiin, joten ensin suoritetaan ensin luvun hajotelma käyttäen samankantaisten potenssien kertolaskun ominaisuutta $2^x \times 2^y = 2^{x+y}$. Näin käytetään useampaa, mutta pienempää taulukkoa. Hajotelman kääntöpuolena on, että suorittaessa hajotelma k kertaa, joudutaan suorittaman $k-1$ kertolaskua.

Edellen artikkelissa (Lee, Lever 2003, taulukot 2,3,4 ja 5) tarkastellaan, millaisissa olosuhteissa kannattaa valita käytettävä esitystapa. Tarkastelu suoritettiin vertaamalla käytettävän piirin kokoa, joka riippuu laskennan operaatioiden jakaumasta. Tarkastelu jakaantuu kahteen vaiheeseen riippuen siitä, onko mahdollista suorittaa kääntämistä lukujärjestelmästä toiseen kesken laskennan vai onko käytetäänkö ainoastaan yhtä esitystapaa koko laskennan aikana.

Artikkelin (Lee, Lever 2003) mukaan määritellyn logaritmiesityksen tapauksessa havaittiin, että jos ei suoriteta kääntämistä, niin kertolaskuja pitäisi olla yksinkertaisen tarkkuuden kanssa vähintään 71% ja kaksinkertaisen tarkkuuden kanssa vähintään 73%, jotta logaritmiesitys kannattaa valita (Lee, Lever 2003, kappale 5.1). Vastaavasti jakolaskuja pitäisi olla vähintään 49% yksinkertaisen tarkkuuden kanssa ja 44% kaksoistarkkuuden kanssa, jotta kannattaa valita logaritmiesitys (Lee, Lever 2003, kappale 5.1). Mikäli vaihtaminen lukujärjestelmästä toiseen on mahdollista laskennan aikana, yksinkertaisen tarkkuuden tapauksessa pitäisi olla vähintään $2,5$ ja kaksoistarkkuuden kanssa vähintään 8 logaritmiesitykseen sopivaa operaatioita, että kannattaa valita logaritmiesitys kääntämisen kanssa, muuten kannattaa pitäytyä pelkästään desimaaliesityksessä (Lee, Lever 2003, kappale 5.2).

6 Yhteenveto

Tässä tutkielmassa on esitetty kaksi erilaista tapaa esittää liukulukuja IEEE:n standardin desimaaliesityksen lisäksi. Murtolukuesitys on mukana yksinkertaisuutensa ja historiallisen kuriositeettinsa suhteen esimerkkinä siitä, miten yksinkertaista liukulukujen toteuttaminen niin laitteistoon kuin ohjelmointikieliin olla, vaikka ne eivät tarjoaisi tälläistä mahdollisuutta suoraan. Logaritmiesitys taas osoittaa tällä tutkimusalalla olevan vielä tarjottavaa johonkin erityistarpeeseen vastaavien esitystapojen myötä.

Vaikka desimaaliesitys onkin murtolukuesitystä monimutkaisempi, niin desimaaliesityksen ilmaisuvoimaa ei voi kiistä. Logaritmiesitykseen verrattuna desimaaliesitys on yleiskäyttöisempi, joten sen valitseminen on helpompaa ja varmempaa eri sovellusalueilla, mikäli ei olla täysin varmoja laskennan soveltumiselta juurikin logaritmiesitykselle. Desimaaliesitys on luultavasti valikoitunut standardiksi juurikin sen yleiskäyttöisyyden takia.

Laitteistotasolla päädytään todennäköisesti joskus siihen pisteeseen, ettei suorittimille enää mahdu lisää fyysisiä laskentaa suorittavia komponentteja. Silloin joudutaan laskentaa nopeuttamaan muilla keinoin. Kun muunnokset esitystapojen välille saadaan riittävän tehokkaiksi varsinaiseen käyttöön, laskentaa voidaan nopeuttaa kehittämällä sellaisia algoritmeja, jotka tunnistavat laskennasta esitystapojen vaihtoon liittyvät osat. Tästä syystä tutkimus alalla on edelleen oleellista.

Tämä tutkimus ei syventynyt muihin esitystapoihin mitenkään syvällisesti. Jatkotutkimukseen on aihetta nimenomaan edellä mainituissa muunnosalgoritmeissa, kuin myös uusien mahdollisten esitystapojen etsimisessä.

Kirjallisuutta

Matula, D 1975. *Fixed-slash and floating-slash rational arithmetic*. Computer Arithmetic

Matula, D ja Kornerup, P. 1978. *A Feasibility Analysis of Binary Fixed-slash and Floating-slash Number Systems*. Computer Arithmetic

Matula, D ja Kornerup, P. 1983. *Finite Precision Rational Arithmetic: Slash Number Systems*. Computer Arithmetic

Goldberg, D. 1991. *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. Computing Surveys

Swartzlander, Earl E. ja Alexopoulos, Aristides G. 1975 *The Sign/Logarithm Number System*. IEEE Transactions on Computers

Lee, B ja Lever, K. 2003. *Logarithmic Number System and Floating-point Implementations of a well-conditioned RLS Estimation Algorithm on FPGA*. Computer Arithmetic

IEEE-SA Standards Board 2008. *IEEE Standard for Floating-Point Arithmetic*. IEEE Std 754™-2008