Tietokoneen toiminnan jatkokurssi, kesä 2020
Harri Kähkönen

Viikkotehtävä 3

Palauta vastauksesi tähän viikkotehtävään kurssin Moodle-oppimisympäristössä tiistaina 18.8.2020 kello 13.59 mennessä pdf-tiedostona. Huomioithan, että sinun on oltava ilmoittautunut viimeistään edellisenä päivänä, jotta pääset Moodleen. Tehtävien ratkaisut käydään läpi ohjatusti Zoomissa tiistaina 18.8.2020 kello 14.15 – 16.00 ja 16.15 – 18.00. Voit osallistua niistä kumpaankin halutessasi. Zoom-linkki on saatavana tietoturvasyistä vain Moodlessa. Muistattehan vastatessa, että yliopistossa niin kokeissa kuin harjoitustehtävissä on plagiointi kielletty, eli esimerkiksi kurssimateriaalista tai muualta suoraan kopioitu tai vähäisesti muunneltu vastaus ei ole sallittu. Opiskeluun yhdessä kannustetaan, mutta kunkin tulee palauttaa itsenäinen tuotos, ei kopiota toisen vastauksista.

Viikon 3 tehtävien sisällöstä

Tässä viikkotehtävässä on tehtäviä koskien ttk-91-ohjelmointia ja tiedon muuttumattomuutta. Tehtävissä keskitytään saavuttamaan tälle kurssille asetetut ttk-91-ohjelmoinnin oppimistavoitteet. Ttk-91 -ohjelmoinnin oppimistavoitteet on esitelty mooc-materiaalissa kohdassa https://tietokoneen-toiminnan-jatkokurssi.mooc.fi/luku-5/5-titokone-titotrainer. Mooc-materiaalissa on myös ttk-91-ohjelmointiin liittyviä tehtäviä. Tavoitteiden saavuttamista varten on hyödyllistä tehdä myös Titotrainer-tehtävät vaikeusasteeltaan 100 - 660 tasoilla a – e. Voit tehdä niitä halutessasi omassa tahdissasi. Ohjelmointiosaaminen tentitään kokeessa erityisesti perustuen mooc-materiaalin tehtäviin ja viikkotehtäviin. Olen kirjoittanut ttk-91-oppaan tänne: https://github.com/epicharri/tito/blob/master/learning_materials/ttk-91ohjeita.md#opas-ttk-91-ohjelmointiin. Tämän viikkotehtävän ttk-91-tehtävät ovat pääosin tehtävissä tuon oppaan avulla.

Yleisohjeita Titokoneen käyttöön

Käytä Titokoneen versiota 1.203. Kurssimateriaalissa on tarkemmat tiedot. Kirjoita ohjelmakoodi jollakin editorilla, jolla voi tallentaa päättää itse tiedostopäätteen ja käytä tiedostopäätteenä .k91. Esimerkiksi voit käyttää Nanoa, Geditiä tai VS Codea. Word ja muut vastaavat tekstinkäsittelyohjelmat eivät sovellu tarkoitukseen.

Ohjelmaa kirjoittaessasi muista aina laittaa myös ohjelman lopetuskäsky. Käännä ja aja ohjelma Titokoneella. Titokoneen näkymän oikeassa alalaidassa on näkymä, jossa näkyy tietoa käännösvaiheesta ja suoritusaikana suoritetuista käskyistä. Kun käyttäjältä pyydetään syötettä, siitä huomautetaan myös kyseisessä näkymässä. Pyydetty syöte laitetaan Titokoneessa kenttään KBD. Tuloste näkyy kentässä CRT. Viimeisin tulostettu luku näkyy siinä kentässä aina ylimpänä.

Koskien tämän viikon ohjelmointitehtäviä

Kun tehtävässä pyydetään tekemään (kirjoittamaan, antamaan, …) ohjelma, vastauksena tulee aina antaa kyseisen ohjelman lähdekoodi. Lähdekoodi on se ohjelmakoodi, jonka olet kirjoittanut, eli ei siis käännetty koodi. Kokeile ohjelmaa aina Titokoneessa. Laita vastaukseen koodi copypastettuna siten, että vertaisarvioija saa sen myös helposti copypastettua Titokoneeseen, jos tulee tarve myös testata ohjelman toimivuus. Osassa tehtäviä pyydetään myös kirjoittamaan vastaukseen symbolitaulun sisältö tai muuta tietoa, jotka näet Titokoneessa tai jotka voit selvittää toki koodistakin.

Tehtäviä on 50 (alakohdat a, b, …). Osa tehtäviä on nopeita tehdä (katsot materiaalista tai ttk-91-oppaasta vastauksen), osa vaatii vähän enemmän miettimistä ja osa voi olla hyvinkin työläitä, taitotasosta riippuen. Arvioinnissa jokainen tehtävä on saman arvoinen. Tältä viikolta saa täydet viikkotehtäväpisteet, kun tehtävistä saa vähintään 40 pistettä 50:stä ja sen vähäisemmästä pistemäärästä lineaarisesti.

1. **Ttk-91-ohjelmoinnin perusteita**

(a) Luvun lukeminen käyttäjältä rekisteriin ja luvun tulostus
Tee ttk-91-ohjelma, joka lukee käyttäjältä luvun rekisteriin 1 ja tulostaa sen näytölle. Aja se Titokoneessa. Anna vastauksena ohjelman lähdekoodi.

(b) Pienin ja suurin ttk-91:n hyväksymä kokonaisluku
Mikä on pienin ja mikä on suurin ttk-91:ssä käytettävä kokonaisluku? Miksi?
Entä mitkä ne ovat silloin, kun luku on ilmaistu suoraan konekäskyssä? Mistä ero johtuu?

(c) Muuttujan tilanvaraus ja alustus
Miten globaalille muuttujalle varataan tilaa ja annetaan alkuarvo? Voidaanko se tehdä ohjelman suoritusaikana? Mikä on pienin ja suurin arvo, joka muuttujalle voidaan antaa? Anna esimerkkinä ttk-91-ohjelma, jossa määritellään muuttujat minInt ja maxInt alkuarvoinaan pienin ja suurin luku mitä muuttujalle voi antaa ja joka tulostaa nuo luvut. *(Globaali muuttuja tarkoittaa muuttujaa, joka on viitattavissa koko ohjelmassa. Tämä kysymys ei siis koske paikallisia muuttujia, joista kerrotaan sivulla https://tietokoneen-toiminnan-jatkokurssi.mooc.fi/luku-6/1-aliohjelmat.)*

(d) Arvon tallettaminen muuttujaan
Tee ttk-91-ohjelma, joka ensin kysyy käyttäjältä luvun rekisteriin R1 ja tallettaa sen muuttujaan *Anna*. Aja koodi Titokoneella. Anna vastaukseksi ohjelmakoodi. Kirjoita vastaukseen myös ohjelman symbolitaulu. Mikä on muuttujan *Anna* osoite?

(e) Osoitinmuuttujan käyttö
Tee ttk-91-ohjelma, jossa määrittelet muuttujat *x* ja *px*. Anna muuttujalle *x* aloitusarvoksi 0 ja *px*:lle 0. Aseta konekäskyjen avulla muuttujan *px* arvoksi muuttujan *x* osoite. Px on nyt muuttujan x osoitinmuuttuja. Tämän jälkeen ilman muuttujan *x* käyttöä suoraan luet käyttäjältä luvun, talletat sen luvun muuttujaan *x* ja toteutat $x = x + 1$. Ohjelman päättyessä siis muuttujan *x* arvon on oltava yhden suurempi kuin käyttäjän antama luku.

(f) Vakiot
Miten määrittelet ttk-91-ohjelmassa suoraan konekäskyn osoite-/vakiokenttään sopivan vakion? Minkä suuruinen se saa olla? Anna esimerkkinä ohjelma, jossa on määritelty vakiot minEqu ja maxEqu ja jossa ladataan vakio pienin rekisteriin r1 ja vakio suurin rekisteriin r2 ja tulostetaan sen jälkeen nuo vakiot. Testaa Titokoneella, että ohjelma toimii. Anna vastauksessasi myös symbolitaulun sisältö.

(g) Suuret vakiot
Entä jos haluat käyttää suurempaa (tai negatiivisista pienempää) vakiota kuin osoitekenttään mahtuu? Miten tällöin alustat vakion nimeltä miljardi? Anna esimerkkinä ohjelma, jossa on määritelty vakio *miljoona*, jonka arvo on 1000000, ja joka lataa vakion *miljoona* arvon rekisteriin r1 sekä tulostaa sen. Testaa Titokoneella, että ohjelma toimii. Mitä yhteistä on näillä suurilla vakioilla ja muuttujilla? Mitä eroa?

(h) Tilanvaraus taulukolle
Anna ohjauskäsky, jolla varataan tilaa 100 alkioiselle taulukolle nimeltä tauluA.

(i) Tilanvaraus 2-ulotteiselle taulukolle
Palauta mieleesi ensimmäisestä viikkotehtävästä 2-ulotteiset taulukot. Anna käskyt (kääntäjän ohjauskäskyt / pseudokäskyt), jolla varataan tilaa taulukolle A, jossa on 3 riviä ja 7 saraketta ja taulukolle B, jossa on 7 riviä ja 3 saraketta.

(j) Tilanvaraus taulukolle, jonka alkioina on tietueita
Anna kääntäjän ohjauskäsky, joilla varaat tilaa taulukolle, jossa on 30 tietuetta. Kussakin tietueessa on 7 kenttää.

2. Laskutoimitukset ja bittitason logiikkaoperaatiot

(a) Jakolasku
Kirjoita ohjelma, joka toteuttaa laskutoimituksen a = 9 / 2 ja tulostaa sen jälkeen muuttujan a arvon. Aja se Titokoneella. Anna vastauksessasi ohjelmasi lähdekoodi. Huomaathan, että jakolaskun tulos pyöristyy alaspäin kokonaisluvuksi.

(b) Yhteenlasku, vähennyslasku ja kertolasku
Kirjoita ohjelma, joka toteuttaa laskutoimituksen d = (a+b) * (a-c). Ohjelma ei siis kysy käyttäjältä mitään eikä tulosta mitään. Alustat nuo muuttujat siis haluamillasi sallitun suuruisilla kokonaisluvuilla. Huomioithan, että laskutoimituksen tulos ei saa missään vaiheessa mennä yli sallitun rajan. Aja koodi Titokoneella. Anna vastauksessasi ohjelman lisäksi myös ohjelman symbolitaulu sekä muuttujien a, b, c ja d osoitteet ja arvot.

(c) Jakojäännös
Kirjoita ohjelma, joka kysyy käyttäjältä jaettavan ja jakajan ja sen jälkeen tulostaa näiden osamäärän ja jakojäännöksen. Aja ohjelma Titokoneella. Anna vastaukseksi ohjelman lähdekoodi.

(d) Bittien siirtokäsky SHL
Oletetaan, että muuttujan *opMaski* arvo on 255 eli 0x 00 00 00 FF. Anna konekäskyt, joilla saat muutettua muuttujan *opMaski* arvoksi 0x FF 00 00 00. Yhden käskyistä on oltava SHL.

(e) Bittien siirtokäsky SHR
Anna konekäskyt, joilla saat eristettyä muuttujan *HAA* vasemmanpuoleiset 8 bittiä siten, että ne ovat muuttujan *opKoodi* arvona. Jos esimerkiksi muuttujan *HAA* arvo on 0x1120007B = 287309947, niin muuttujan *opKoodi* arvon tulee olla 0x11.

(f) Bittien siirtokäsky SHL
Oletetaan, että muuttujan *vRekMask* arvo on 7 eli
0b 0000 0000 0000 0000 0000 0000 0000 0111. Anna konekäskyt, joilla saat SHL-käskyä hyödyntäen muutettua muuttujan *vRekMask* arvoksi
0b 0000 0000 1110 0000 0000 0000 0000 0000.

(g) Maski ja AND
Anna konekäskyt, joilla saat eristettyä muuttujasta *HAA* kolme bittiä, jotka ovat vasemmalta lukien 9., 10. ja 11. bitti ja talletettua niiden arvon muuttujaan *vasenRek*.

Käytä bittimanipulointiin edellä määriteltyä muuttujaa vRekMask, AND-käskyä ja SHR-käskyä.

(h) Bittimanipulointia: bittien siirtokäskyt SHR ja SHL, maskit ja looginen operaatio AND

Kirjoita ohjelma, jossa on jossakin kohdassa ohjelmaa käskyrivi

```
ElsaFM Load R1, =27
```

ja joka tallettaa muuttujaan *kasky* osoitteessa *ElsaFM* olevan konekäskyn sekä käyttäen bittimaskeja sekä AND, SHR ja SHL -operaatioita eristää osoitteessa *ElsaFM* olevasta konekäskystä operaatiokoodin, vasemman puoleisen rekisterin numeron (0,…,7), osoitusmoodin, oikean puoleisen rekisterin numeron ja vakio-osan muuttujiin *opKoodi, vasenRek*, *moodi*, *oikeaRek* ja *vakioOsa*. Aja ohjelma Titokoneessa. Anna vastauksena koko ohjelman lähdekoodin lisäksi symbolitaulut ja jokaisen muuttujan arvo ohjelman suorituksen jälkeen.

(i) Bittitason loogiset operaatiot: OR

Jatka edellisen kohdan ohjelmaa siten, että muutat sopivan maskin, AND-operaation ja OR-operaation avulla osoitteessa *ElsaFM* olevan konekäskyn vakio-osan arvoksi 28. (Huom! Älä tee tätä yhteenlaskuoperaation avulla, koska tässä on tarkoitus harjoitella näitä loogisia operaatioita.) Anna vastauksena ohjelman lähdekoodi.

(j) Bittitason loogiset operaatiot: NOT ja kahden komplementti

Kirjoita ohjelma, joka kysyy käyttäjältä luvun, muuntaa luvun vastaluvuksi ottamalla luvusta kahden komplementin käyttäen operaatiota NOT ja yhteenlaskuoperaatiota ADD sekä lopuksi tulostaa näin muutetun vastaluvun. (Ttk-91:ssä luvut on tallennettu Big-Endian -kahden komplementtimuodossa.) Anna vastauksena ohjelman lähdekoodi.

(k) Bittitason loogiset operaatiot: XOR

Miten voidaan XOR-operaation avulla saada selville, onko kahdessa eri rekisterissä olevat arvot samat?

(l) Kahden rekisterin arvojen vaihtaminen (swap) ilman apurekisteriä: XOR

Tee ohjelma, jossa aluksi rekisteri R1 sisältää arvon 10 ja rekisteri R2 arvon 20. Vaihda rekisterien R1 ja R2 arvot keskenään käyttämällä pelkästään XOR-käskyjä. (Katso algoritmi täältä: https://en.wikipedia.org/wiki/XOR_swap_algorithm.) Aja ohjelma Titokoneessa ja tarkista Registers -kohdasta, että arvot ovat muuttuneet. Anna vastauksena ohjelman lähdekoodi.

3. IF-THEN-ELSE, toistorakenteet ja taulukot

(a) IF-THEN-ELSE

Tee ohjelma, joka kysyy käyttäjältä syötteenä kokonaisluvun. Jos luku on ei-negatiivinen, luku tulostetaan. Muuten luku muunnetaan vastaluvukseen ottamalla kahden komplementti käyttäen NOT ja ADD -operaatioita ja tulostetaan luku. Tässä on siis myös tarkoitus uudelleen harjoitella kahden komplementti -asioita. Anna vastauksena ohjelman lähdekoodi.

(b) Toistorakenne

Tee ohjelma, joka kysyy käyttäjältä lukuja. Jos luku on 0, lopetetaan. Jos luku on negatiivinen, pyydetään uusi luku. Jos luku on positiivinen, tulostetaan luku ja pyydetään uusi luku.

(c) Toistorakenne

Tee ohjelma, joka toistorakennetta hyödyntäen pyytää käyttäjältä 10 lukua ja tulostaa lopuksi niiden summan. Kokeile koodin toimivuus Titokoneessa. Anna vastauksena ohjelman lähdekoodi.

(d) Taulukon alustus

Tee ohjelma, joka alustaa toistorakenteen avulla 10-alkioisen taulukon T siten, että T[i]=i*i. Anna vastauksena ohjelman lähdekoodi.

(e) Taulukon alustus: kertoma

Tee toistorakennetta hyödyntäen ohjelma, joka alustaa 13-alkioisen taulukon nimeltä kertoma siten, että kohdassa kertoma[k] on luvun k kertoma. Siis kohdassa kertoma[0] on luvun 0 kertoma eli 0!=1, kohdassa kertoma[1] luvun 1 kertoma eli 1!=1, kohdassa kertoma[2] luvun 2 kertoma eli 2! = 2*1! ja kohdassa kertoma[k] on luvun k kertoma on k*(k-1)!. Aja koodi Titokoneessa. Anna vastauksena ohjelman lähdekoodi, symbolitaulu ja kertomien arvot taulukolle kertoma varatuista muistipaikoista. *(Vinkki: tallenna ensin taulukon indeksiin 0 luvun 0 kertoma 1. Tee sen jälkeen toistorakenteen avulla taulukon alustus alkaen indeksistä 1 siten, että kertoma[k] = k*kertoma[k-1].)*

4. Taulukot joissa on tietueita sekä kaksiulotteiset taulukot

(a) Taulukko, jonka alkioina on tietueita

Tee seuraavanlainen ohjelma:
- Ohjelmassa varataan tila taulukolle, jossa on 100 tietuetta. Kussakin tietueessa on kentät opiskNro, pisteet ja koepisteet.
- Käyttäjältä kysytään tietueen indeksiä (0...99) taulukossa
- Jos käyttäjä antaa liian suuren indeksin, palataan takaisin kysymään indeksiä.
- Ohjelma lopetetaan antamalla indeksiksi negatiivinen luku
- Seuraavaksi käyttäjältä luetaan opiskelijanumero (opiskNro), pisteet ja koepisteet sekä talletetaan nuo tiedot indeksin mukaiseen tietueeseen.
- Tietueen indeksin kysymistä ja tietojen tallentamista kysytään toistuvasti, kunnes käyttäjä antaa indeksiksi negatiivisen luvun.

(b) Kaksiulotteisen taulukon alustus

Tee seuraavanlainen ohjelma:
- Ohjelmassa varataan tila 2-ulotteiselle riveittäin tallennetulle taulukolle S, jossa on 3 riviä ja 7 saraketta.
- Varataan tila muuttujalle X alustaen se arvolla 10.
- Määritellään vakiot K ja L, missä K:n arvo on tuon taulukon korkeus ja L tuon taulukon leveys.
- Alustetaan toistorakenteen avulla ja vakioita K ja L käyttäen taulukon S arvot siten, että jokaisen alkion arvona on muuttujan X arvo.

Anna vastauksena ohjelman lähdekoodi.

(c) Kaksiulotteisen taulukon alkioiden käyttö

Tee ohjelma, jossa on määritetty 5-rivinen ja 6-sarakkeinen riveittäin tallennettu taulukko T ja jossa kysytään käyttäjältä *rivi*, *sarake* ja *arvo.* Jos rivi tai sarake ei ole sallitun suuruinen, tulostetaan ”virheilmoituksena” 9999 ja pyydetään rivi, sarake ja arvo uudelleen. Lopuksi toteutetaan T[*rivi*, *sarake*] = *arvo.* Huomioithan, että taulukon

indeksöinnit ovat kuten yleensäkin ohjelmointikielissä. Anna vastauksena ohjelman lähdekoodi.

(d) Kaksiulotteisen sarakkeittain tallennetun taulukon alustus

Tee ohjelma, joka alustaa toistorakenteen avulla 5-rivisen ja 6-sarakkeisen sarakkeittain tallennetun taulukon arvot siten, että T[rivi, sarake] = 0, jos rivin indeksi on parillinen, muutoin 1. Anna vastauksena ohjelman lähdekoodi.

(e) Matriisien yhteenlasku osa 1

Tee ohjelma, jossa on määritelty 8*4 (rivit*sarakkeet) kokoiset matriisit A, B ja C. Matriisi on siis 2-ulotteinen taulukko.Taulukot on tallennettu riveittäin. Matriisin rivien määrä pitää (tässä tehtävässä, ei yleisesti) määrittää EQU-vakiolla M ja sarakkeiden määrä EQU-vakiolla N. Alusta toistorakennetta käyttäen matriisin A alkioiden arvoksi 1, matriisin B alkioiden arvoksi 2 ja matriisin C alkioiden arvoksi 0. Anna vastauksessa ohjelman lähdekoodi.

(f) Matriisien yhteenlasku osa 2

Jatka edellisen kohdan ohjelmaa toteuttamalla C=A+B. Tämä tarkoittaa sitä jokaista matriisien A, B ja C indeksiä rivi, sarake kohden tulee päteä C[rivi, sarake]=A[rivi,sarake]+B[rivi,sarake]. Muista kokeilla tätäkin Titokoneessa. Anna vastauksessa ohjelman lähdekoodi.

5. Aliohjelmat

(a) **Yksinkertainen aliohjelma**

Tee ohjelma, jossa muuttuja x on alustettu pääohjelmatasolla ja toteuta ohjelmaan seuraavat:

Aliohjelman kutsu siten, että ohjelman päättyessä x=f(2,3,4) ja aliohjelmana funktio f(a,b,c), joka palauttaa arvon a*b+c. Aliohjelmalle f annetaan parametreina kolme kokonaislukua a, b ja c arvoparametreina, ja se palauttaa paluuarvona laskutoimituksen a*b+c tuloksen. Noudata suosituksen mukaista aliohjelman kutsumis- ja toteutustapaa. Testaa ohjelmaa, että se toimii. Anna vastauksena ohjelman lähdekoodi.

(b) Viite- ja arvoparametrit, paluuarvo ja ulostuloparametri aliohjelmassa

Tässä tehtävässä kutsuttava aliohjelma ASETA(T, K, L, Y, X, A) toteuttaa sijoituksen T[Y,X]=A eli asettaa taulukon T riville Y sarakkeeseen X muuttujan A arvon ja palauttaa arvon 1 jos sijoittaminen onnistuu ja 0 jos sijoittaminen ei onnistu. Onnistuminen tarkoittaa, että Y ja X ovat sallituissa rajoissa. Taulukon korkeus (eli rivien määrä) on K, leveys (eli sarakkeiden määrä) on L, rivin indeksi on Y ja sarakkeen indeksi on X. Parametreista T on viiteparametri ja samalla ulostuloparametri, K, L, Y ja X ovat arvoparametreja ja A on viiteparametri.

Tee ohjelma, jossa määrittelet pääohjelmatasolla 5-rivisen ja 7-sarakkeisen riveittäin talletetun taulukon nimeltään *Pelilauta*, muuttujan *HAA* jonka arvo on 42 ja joka funktiota ASETA(Pelilauta, 5, 7, 4, 3, HAA) kutsumalla asettaa Pelilauta[4,3]=HAA ja tulostaa lopuksi luvun 1 jos sijoitus onnistui ja luvun 0 jos se ei onnistunut. Anna vastauksena ohjelmakoodi. *HUOM! Ohjelmaa ei voi testata, ennen kuin funktio ASETA on toteutettu. Ohjelmaa laajennetaan siis seuraavassa tehtävässä.)*

(c) Jatkoa edelliseen.
Toteuta edellisen tehtävän funktio ASETA(T, K, L, Y, X, A). Noudata suosituksen mukaista aliohjelman kutsumis- ja toteutustapaa. Muista kokeilla Titokoneella. Anna vastauksena ohjelmakoodi. *HUOM! Tässä edellisen kohdan jatkotehtävässä kokeillaan isoa osaa aliohjelmiin liittyviä parametrien välitystapoja. Tämän tekemiseen saattaa mennä paljon aikaa, mutta toisaalta tämä voi olla joillekin nopeakin tehdä. Tehtävä on tarkoituksellisesti ”All or nothing at all” niin kuin laulussa sanotaan. Älä siis huolestu, jos tämä on vaikea. Pyydä Telegramissa apua tarvittaessa. Ja jos osaat, niin neuvo muita: saat palkkioksi hyvän mielen toiselle ja neuvoessa omakin osaaminen syventyy!*

(d) Aktivointitietue ja aktivointitietuepino
Selitä, lyhyesti ja ytimekkäästi esimerkiksi ranskalaisia viivoja käyttäen mitä ovat aktivointitietue ja aktivointitietuepino sekä kerro mitkä aliohjelmien toteuttamisen ongelmat niiden käyttö ratkaisee.

6. Tiedon suojaaminen: Pariteetti, Hamming-koodi ja CRC

(a) Pariteettibitti
Oletetaan, että yksinkertaisen laitteen eräässä väylässä halutaan siirtää 4 bittiä tietoa kerralla eli siinä on 4 johdinta dataa varten. Laitteen kehittäjät ovat päätyneet suojaamaan datan pariteettibitin avulla, joten väylään on lisätty yksi ylimääräinen johdin.

Oletetaan parillinen pariteetti. Mikä pariteettibitin kuuluu olla, jos väylää pitkin kuljetetaan luku 12? Perustele.

(b) Pariteettibitti jatkuu
Jatkoa edelliseen. Oletetaan pariton pariteetti. Mikä pariteettibitin kuuluu olla, jos väylää pitkin kuljetetaan luku 10? Perustele.

(c) Pariteettibitti jatkuu
Jos väylä on suojattu yhden pariteettibitin avulla, montako virhettä sillä voidaan havaita? Voidaanko virheitä paikallistaa ja korjata?

(d) Hamming-koodin perusteita
Entä jos tuo 4-bittinen väylä halutaan suojata Hamming-koodilla joka havaitsee osan kahden bitin virheistä ja kaikki yhden bitin virheet sekä osaa korjata yhden bitin virheen? Montako ylimääräistä johdinta Hamming-koodin bittejä varten tällöin tarvitaan? Perustele.

(e) Hamming-koodin perusteita jatkuu
Yllä mainittua 4-bittistä väylää pitkin lähetetään luku 12 suojattuna. Suojauksen suorittaa tiedon lähettävä logiikkapiiri. Mitkä arvot pitää olla kullakin Hamming-koodin pariteettibiteistä, kun oletamme parillisen pariteetin mukaisen Hamming-koodin suojauksen? Perustele.

(f) Hamming-koodin perusteita jatkuu
Yllä mainitun mukaisesti Hamming-koodilla suojattu data saapuu väylää pitkin määränpäähänsä, jossa on logiikkapiiri joka tarkistaa tiedon oikeellisuuden. Databiteistä yksi on muuttunut ja saapunut data onkin luku 4. Oletetaan, että pariteettibitit ovat

säilyneet ennallaan. Monenko bitin virhe on kyseessä? Voidaanko virhe paikallistaa ja korjata? Miten tuo logiikkapiiri tunnistaa missä bitissä virhe on? Miten virhe korjataan? Miten logiikkapiiri tarkistaa, että korjaus onnistui?

(g) Hamming-koodin perusteita jatkuu

Jatkoa edelliseen. Oletetaan nyt, että samaisella väylällä lähetetään luku 12 ja perille saapuukin luku 15. Oletetaan myös, että Hamming-koodin pariteettibitit ovat säilyneet ennallaan. Monenko bitin virhe on kyseessä? Logiikkapiiri tekee nyt samat toiminnot kuin aiemminkin virheen paikallistamiseksi ja korjaamiseksi. Voidaanko virhe paikallistaa ja korjata? Miten logiikkapiiri tunnistaa missä bitissä virhe on? Miten se korjaa sen? Miten logiikkapiiri tarkistaa, että korjaus onnistui? Onnistuiko korjaus?

(h) Hamming-koodin perusteita jatkuu: virhe pariteettibitissä

Entä jos toisella kertaa väylää pitkin saapunutta tietoa tarkistaessa logiikkapiiri huomaa virheen olevan jossakin pariteettibitissä: Miten piiri tietää, että virhe onkin pariteettibitissä? Mitä piiri silloin tekee?

(i) Hamming-koodin perusteita jatkuu: koodi joka löytää kaikki kahden bitin virheet

Jatkoa edelliseen. Mitä muutoksia väylään pitää tehdä, jotta voidaan havaita kaikki kahden bitin virheet ja korjata kaikki yhden bitin virheet? Miten logiikkapiirin pitää tällöin tehdä tarkistukset ja korjaukset?

(j) Hamming-koodi 32-bittiseen dataväylään

Oletetaan, että tietokoneen dataväylässä halutaan saada kulkemaan 32 bittiä dataa kerrallaan ja se halutaan suojata Hamming-koodilla, joka havaitsee kaikki yhden bitin virheet ja osaa korjata kaikki yhden bitin virheet. Montako ylimääräistä johdinta tarvitaan? Perustele.

(k) Hamming-koodi 32-bittiseen dataväylään jatkuu

Jatkoa edelliseen kohtaan. Entä jos halutaan havaita kaikki kahden bitin virheet ja korjata kaikki yhden bitin virheet? Montako ylimääräistä johdinta tällöin tarvitaan? Perustele. Miten tällöin tarkistetaan virheet?

(l) Hamming-koodi 32-bittiseen dataväylään jatkuu

Oletetaan, että 32-bittistä dataväylää pitkin on lähetetty muistista suorittimen MMU:lle sana 0xC1A24015. Dataväylässä on lisäksi johtimet pariteettibiteille. Kyseessä on Hamming-koodi, joka tunnistaa kaikki yhden bitin virheet ja osaa korjata kaikki yhden bitin virheet. Muistipiirin puolella väylää oleva logiikkapiiri määrittelee pariteettibittien arvot noille johtimille muistista noudetun datan perusteella. Oletetaan, että pariteettibitit on numeroitu oikealta vasemmalle, eli pariteettibitit suojaavat tuon datan bittejä oikealta vasemmalle. Oletetaan, että väylällä yksi biteistä vaihtuu ja MMU:n saatua datan ja pariteettibitit, se havaitsee virheen olevan bitissä 37.

Mitkä pariteettibitit ilmaisivat virheen sijainnin? Mikä on tuo MMU:lle saapunut virheellinen 32-bittinen data? Mitkä ovat pariteettibittien arvot?

(m) CRC

Miksi tietoliikenteessä suositaan ennemmin CRC-suojausta kuin Hamming-koodia? Kerro lyhyesti mihin CRC-suojaus perustuu. (Huom! CRC-suojaukseen palataan tarkemmin Tietoliikenteen perusteet -kurssilla. Tässä on tarkoitus vastata asiaan vain pääpiirteittäin.)