(19)

SUOMI - FINLAND

(FI)

PATENTTI- JA REKISTERIHALLITUS

PATENT- OCH REGISTERSTYRELSEN

FINNISH PATENT AND REGISTRATION OFFICE

(10) **FI 925057 A7**

(12) **JULKISEKSI TULLUT PATENTTIHAKEMUS**
PATENTANSÖKAN SOM BLIVIT OFFENTLIG
PATENT APPLICATION MADE AVAILABLE TO THE PUBLIC

(21) Patenttihakemus - Patentansökan - Patent application 925057

(51) Kansainvälinen patenttiluokitus - Internationell patentklassifikation - International patent classification
G06F 9/455
G06F 9/44

(22) Tekemispäivä - Ingivningsdag - Filing date **03.03.1992**

(23) Saapumispäivä - Ankomstdag - Reception date **06.11.1992**

(41) Tullut julkiseksi - Blivit offentlig - Available to the public **06.11.1992**

(43) Julkaisupäivä - Publiceringsdag - Publication date **13.06.2019**

(86) Kansainvälinen hakemus - Internationell ansökan - International application **03.03.1992 PCT/US1992/001715**

(32) (33) (31) Etuoikeus - Prioritet - Priority

07.03.1991 US 666071

(71) Hakija - Sökande - Applicant

1 • **Digital Equipment Corporation,** 146 Main Street, Maynard, Mass. 01754, AMERIKAN YHDYSVALLAT, (US)

(72) Keksijä - Uppfinnare - Inventor

1 • **Robinson, Scott G.,** USA, AMERIKAN YHDYSVALLAT, (US)
2 • **Sites, Richard L.,** USA, AMERIKAN YHDYSVALLAT, (US)
3 • **Witek, Richard T.,** USA, AMERIKAN YHDYSVALLAT, (US)

(74) Asiamies - Ombud - Agent

Kolster Oy Ab, Salmisaarenaukio 1, 00180 Helsinki

(54) Keksinnön nimitys - Uppfinningens benämning - Title of the invention

Järjestelmä ja menetelmä, joka säilyttää lähdekäskyjen atomisuuden kää nnetyssä ohjelmakoodissa

System och förfarande för att bevara atomiciteten hos källinstruktione r i en översatt programkod

Järjestelmä ja menetelmä, joka säilyttää lähdekäskyjen atomisuuden käännetyssä ohjelmakoodissa

Keksinnön tausta

Esillä oleva keksintö liittyy järjestelmiin ja menetelmiin ohjelmakoodien sovittamiseksi suoritettavaksi erilaisissa tietokonejärjestelmissä sekä erityisesti järjestelmiin ja menetelmiin alkuperäisessä ohjelmassa olevista virheistä ilmoittamiseksi alkuperäisen ohjelman käännöksen suorittamisen aikana.

Tietokoneohjelmoinnin varhaisvuosina tietokoneohjelmien käskyt kehitettiin mikrokoodin tasolla. Ohjelmistotekniikan kehittymisen ja laajenemisen myötä yhdistettiin enemmän tehtäviä yksittäisiin monimutkaisiin käskyihin, jotka olivat suoritettavissa tällaista käskyjen mutkikkuutta varten suunnitellulla laitteistoarkkitehtuurilla.

Käskyjen kasvanut monimutkaisuus tuotti yleensä suurentuneita hinta/suorituskykyetuja laitekustannusten ja suoritusominaisuuksien kehittyvässä ympäristössä. Tästä johtuen mutkikkaat käskykantakoodit (CISC, complex instruction set code) tulivat laajalti hyväksytyiksi.

Käskyjen mutkikkuuden lisääntyessä järjestelmän laitteiston suunnittelu suuremmille suoritusnopeuksille on kuitenkin tullut vaikeammaksi. Sen asemesta supistettu käskykantakoodi (RISC, reduced instruction set code) yhdistettynä sitä vastaavaan tietokonelaitteiston RISC-arkkitehtuuriin on saavuttanut hyväksymisen mekanismina, joka johtaa merkittävästi parantuneeseen järjestelmän hinnan ja suorituskyvyn suhteeseen.

RISC-järjestelmä käyttää yleensä yksinkertaisempia peruskäskyjä haluttujen toimintojen ohjaamiseksi. Yksi RISC-käsky normaalisti määrittelee yhden operaation käyttäen enintään yhtä muistiosoitetta. Lisäksi RISC-järjestelmä normaalisti käyttää rekisteriä kullakin peruskäskyl-

lä. RISC-käskykannan käskyt ovat silti korkeammalla tasolla kuin mikrokoodi.

Tyypillisessä CISC-järjestelmässä yksi käsky voi määritellä mutkikkaan operaatiosarjan ja se voi suorittaa monta suoraa muistiosoitusta. Siten CISC-käskyn suorittamat operaatiot voivat tarvita useita RISC-käskyjä.

RISC-järjestelmä suunnitellaan yleensä käyttäen optimoituja laitteisto- ja ohjelmistosovituksia, jotka aikaansaavat järjestelmän nopeamman toiminnan, paremman kokonaissuorituskyvyn ja pienemmät järjestelmäkustannukset verrattuna käytettävissä olevien laitteistojen kustannuksiin ja suorituskykyyn.

Eräs este vaihtamiselle CISC-järjestelmistä RISC-järjestelmiin on suurten ohjelmakirjastojen olemassaolo, jotka on kehitetty CISC-järjestelmille ja joita ei ole yleisesti saatavilla RISC-järjestelmille. Kun tietokonejärjestelmän käyttäjä aikoo hankkia uuden tietokonejärjestelmän, eräs käyttäjän suurimpia huolenaiheita on se, voidaanko käyttäjän sovellusohjelmakirjastoa käyttää tai voidaanko se muuntaa käytettäväksi uudessa tietokonejärjestelmässä ja mitkä tämän kirjaston vaihtamiskustannukset olisivat. Siten sellaisille tietokojejärjestelmän käyttäjille, jotka haluavat saada paremman hinta/suorituskykysuhteen RISC-tietokonejärjestelmien avulla, on erittäin tärkeätä, että on käytettävissä taloudellinen ja tehokas mekanismi käyttäjän sovellusohjelmakirjaston sovittamiseksi eli "siirtämiseksi" suoritettavaksi RISC-tietokonejärjestelmässä.

Käyttäjällä on ohjelmien siirtämiseksi käytettävissä useita vaihtoehtoja. Voidaan käyttää uudelleen kääntämistä tai uudelleen koodaamista, mutta näitä tekniikoita käytetään tyypillisesti sellaisella korkean tason kielellä kuten FORTRANilla kirjoitettujen ohjelmien siirtämiseen, joilla joko ei ole yksityiskohtaisia koneriippuvuuksia tai joista mahdolliset koneriippuvuudet poistetaan käsin suo-

ritettavilla ohjelman muutoksilla. Lisäksi uudelleen kääntämisessä tai uudelleen koodaamisessa käyttäjällä on tyypillisesti kaikki vastuu ohjelman muuttamisesta ja ohjelman käyttäytymisen varmistamisista.

Vaihtoehtoisesti voidaan käyttää tulkintaproseduureja, mutta tämän lähestymistavan haittana tyypillisesti on ohjelman olennaisesti alentunut suorituskyky.

Tulkintaproseduurit ovat tarkemmin sanoen ohjelmia, jotka toimivat tietyssä tietokoneessa ja lukevat lähdekäskyjen (jotka voivat hyvin olla erityyppiselle tietokoneelle tarkoitettuja käskyjä) virtaa datana ja kullakin lähdekäskyllä suorittavat annetun operaation. Tällaiset proseduurit suorittavat tyypillisesti 10 - 100 konekielistä käskyä mainitussa tietyssä tietokoneessa yhden ainoan lähdekäskyn tulkitsemiseksi. Siten tulkintaproseduurit aikaansaavat ohjelman olennaisesti alentuneen suorituskyvyn verrattuna toiminnallisesti ekvivalenttisen koodin suoraan suorittamiseen mainitussa tietyssä tietokoneessa.

Vaikuttavin ja tehokkain siirto kuitenkin käsittää koodin kääntämisen. Koodin muuttamisessa olemassaolevan ohjelman kukin käsky käännetään yhdeksi tai useammaksi kohdekoneen kielellä ilmaistuksi käskyksi. Näin ollen CISC-ohjelmien kääntäminen RISC-ohjelmiksi tai yleisemmin sellainen ohjelman kääntäminen, jossa käännetyllä koodilla on suhteellisesti supistettu käskykanta, vaatii "monta" tai "useita" käskyjä käännetyssä koodissa kutakin käännettävän koodin käskyä kohti. Suoritettaessa koodinkäännöksiä "yhdestä moneksi" eli CISC:stä RISC:ksi on yleensä kuitenkin vaikeata säilyttää monia niistä käskyjen käyttäytymistä koskevista takuista, jotka alunperin on annettu CISC-koodille tai muulle suhteellisen mutkikkaalle koodille.

Eräs normaali CISC-takuu, joka aiheuttaa jonkin verran vaikeuksia kääntämisessä, on se vaatimus, että mitään muuta CISC-käskyä tai sen osaa ei voi suorittaa yhden CISC-käskyn alkamisen ja päättymisen välillä. Näin ollen

CISC:tä RISC:ksi käännettäessä on olennaista, että tätä tyyppiä oleva käskyjen eheys eli käskyjaotus säilytetään. Käskyjaotuksen säilyttäminen edellyttää, että tilan tai muistin atomisuus säilytetään. Siten joko kaikkien käskyssä olevien muistiin osoittamisten täytyy näyttää tapahtuvan tai mitään ei saa näyttää tapahtuvan käännetyssä koodissa.

Käskyjaotuksen säilyttämiseksi käännösprosessissa täytyy olla olemassa varmistus sille, että käännettyjen käskyjen kukin ryhmä tai "jaoke", joka vastaa kutakin mutkikkaampaa käskyä, suoritetaan siten että se tuottaa saman tuloksen kuin minkä vastaava mutkikkaampi käsky olisi tuottanut. Tämän täytyy päteä, vaikka asynkronisia tapahtumia voi esiintyä minkä tahansa yksinkertaisempien käännettyjen käskyjen "jaokkeiden" suorituksen aikaan.

Patenttihakemus (1870-0410), johon edellä on viitattu ja joka on jätetty samanaikaisesti, koskee keksintöä, joka aikaansaa käskyjaotuksen säilymisen koodin kääntämisessä ja käännetyn koodin suorittamisessa. Lisäksi patenttihakemus 1870-0410 yleisesti aikaansaa tila-atomisuuden ja erityisesti paljastaa mekanismin ja menetelmän tila-atomisuuden takaamiseksi yhden kirjoittamisen käskyjen tapauksessa käännettävässä CISC- tai muussa koodissa.

Yhden kirjoittamisen käsky sisältää enintään yhden tilakirjoituksen, joka voi mahdollisesti kohdata poikkeuksen, mutta se voi sisältää rajoittamattoman määrän poikkeuksettomia tilakirjoituksia. Näiden kahden tilakirjoituksen lajin välillä ei ole limittäisyyttä.

Termin "poikkeus" tarkoituksena tässä on viitata mihin tahansa tilaan, joka estää käskyn suorituksen jatkamisen. Tyypillisiin poikkeuksiin kuuluvat:

a. muistipoikkeukset kuten sivuvirhe tai osoitusvirhe;

b. aritmeettiset poikkeukset kuten liukuluvun ylivuoto tai nollalla jakaminen; sekä

c. käskypoikkeukset kuten luvaton operaatiokoodi tai keskeytyskäsky.

Tila-atomisuus on sama kuin käskyjaotus yhden kirjoittamisen käskyn tapauksessa, joka käännetään ja suoritetaan patenttihakemuksen 1870-0410 mukaan. Kuitenkin muut käskylajit, jotka täytyy kääntää, sisältävät sekvenssejä, jotka aiheuttavat erityisiä ongelmia tila-atomisuuden ja käskyjaotuksen säilymisen aikaansaamisessa. Tällaisiin käskyihin kuuluvat sellaiset, joissa on:

a. luku-muutos-kirjoitus-sekvenssi, joka on "lukittu" tai luku-muutos-kirjoitus, joka vaatii osittaisen muistisanan kirjoittamisen ja joka täytyy suorittaa monisuoritinjärjestelmässä ilman toisen prosessorin välillä suorittamaa kirjoittamista; sekä

b. moninkertaiset tilakirjoitukset, jotka voivat mahdollisesti kohdata poikkeuksen ja joiden kaikkien täytyy näyttää joko tapahtuvan tai ei tapahtuvan.

Näissä erikoistapauksissa tarvitaan aivan erityiset mekanismit ja menetelmät eteen tulevien erikoistilanteiden huomioonottamiseksi yritettäessä aikaansaada tila-atomisuus ja käskyjaotus käännetyssä koodissa. Moninkertaisia tilakirjoituksia omaavien käskyjen tapauksessa tila-atomisuusongelma ilmenee asynkronisen tapahtuman esiintyessä käskysekvenssin suorituksen aikana ainakin yhden jälkeen, ennenkuin kaikki tilakirjoitukset (muistiin tai rekisteriin) on suoritettu. Siten poikkeus voisi esiintyä sekvenssissä jäljellä olevissa suoritettavissa käännetyn koodin käskyissä siten, että jos käskyjaotuksen poikkeus joko keskeytetään tai sitä jatketaan, niin tilavirhe voi syntyä, koska palautumaton tilanmuutos on jo voinut esiintyä jo suoritetun yhden tilakirjoituksen yhteydessä.

Sellaisessa tapauksessa, jossa käännetty koodi suoritetaan järjestelmässä, jossa on useita yhteisellä muistilla varustettuja suorittimia, voi esiintyä tila-atomisuusongelma, koska ensimmäinen suoritin, jossa käännettyä

koodia ollaan suorittamassa, voi osittain suorittaa luku-muutos-kirjoitus-sekvenssin käskyjaokkeessa, ja sen jälkeen, mutta ennenkuin luku-muutos-kirjoitus-sekvenssi on suoritettu loppuun, toinen suoritin voi kirjoittaa tämän luku-muutos-kirjoitus-sekvenssin osoittamaan tilaosoitteeseen. Tällöinkin jos käskyjaokkeen suoritus joko keskeytetään tai sitä jatketaan toisen suorittimen suorittaman ristiriidassa olevan tilaosoituksen jälkeen, voi syntyä tilavirhe, koska palautumaton muutos on jo voinut esiintyä.

Näin ollen esillä oleva keksintö koskee rakennetta ja menetelmiä sellaisten käännettyjen koodien tuottamiseksi ja suorittamiseksi, joilla on suhteellisesti supistetut käskykannat, olemassaolevista koodeista, joilla on mutkikkaammat käskykannat, säilyttäen samalla käskyjaotus ja tila-atomisuus, missä käskyt sisältävät käskyjä, joihin liittyy erityistilanteita, kuten moninkertaisia tai osittaisia kirjoittamisia, lukituskäskyjä sekä monisuoritinpohjainen suoritusympäristö. Esillä oleva keksintö tekee siten mahdolliseksi tietokonejärjestelmän hinta/suorituskyky-suhteen parannusten toteuttamisen samalla kun se säilyttää sovelluskoodi-investoinnit myös tapauksissa, joissa tällaisia erikoistilanteita esiintyy.

Keksinnön yhteenveto

On aikaansaatu järjestelmä tai menetelmä ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi ja toisen ohjelmakoodin suorittamiseksi säilyttäen samalla ensimmäisen koodin käskyjen tila-atomisuus. Ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti- ja rekisteritilan sekä toiselle käskykannalle, joka on supistettu suhteessa ensimmäiseen käskykantaan, sovitetun toisen arkkitehtuurin omaavassa tietokoneessa suoritettava.

Ensimmäinen tietokone kääntää ensimmäisen koodin käskyt vastaaviksi toisen koodin käskyiksi rakennekoodin mukaan, joka määrittelee ensimmäisen koodin käskyt toisen koodin käskyjen avulla. Toisen koodin käskyt kutakin ensimmäisen koodin käskyä kohti organisoidaan jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä vähintään kaksi ryhmää, missä ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilapäivityksen ja jotka voidaan keskeyttää suorituksen jälkeen ilman tilavirheen vaaraa, ja toisessa ryhmässä ovat kaikki muisti- ja rekisteritilan päivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävän ensimmäisen koodin käskyn toteuttamiseksi.

Ensimmäinen erityinen kirjoituskäsky strukturoidaan sisältämään ensimmäinen alisekvenssi yhden ensimmäiseen muistipaikkaan kirjoittamisen käsittelemiseksi sen vaatimuksen mukaan, että mainittu ensimmäinen alisekvenssi täytyy suorittaa ilman mitään keskeytystä ja ilman minkään muun mainittuun muistitilaan ehkä liitetyn suorittimen väliin tulevia ristiriidassa olevia kirjoittamisia mainittuun ensimmäiseen muistipaikkaan. Lisäksi strukturoidaan toinen erityinen kirjoituskäsky sisältämään toinen alisekvenssi moninkertaisten kirjoittamisten käsittelemiseksi, jotka täytyy suorittaa ilman keskeytystä.

Toinen tietokonejärjestelmä on sovitettu toiselle arkkitehtuurille toisen ohjelmakoodin suorittamiseksi. Toisen koodin suorituksen aikana käytetään elintä, joka määrittää kunkin asynkronisen tapahtuman esiintymisen toisen koodin suorituksen aikana sekä kunkin toisen suorittimen suorittaman, ensimmäiseen muistipaikkaan kohdistuvan ristiriidassa olevan kirjoittamisen esiintymisen, jos mainittu toinen suoritin on liitetty muistitilaan.

Mikä tahansa jaokkeinen toisen koodin käskysekvenssi keskeytetään uudelleenyritystä varten ensimmäisen koodin käskyjen tila-atomisuuden ja ensimmäisen koodin käs-

kyjaotuksen säilyttämiseksi, jos asynkroninen tapahtumakeskeytys esiintyy tämän sekvenssin suorituksen aikana, ennenkuin kaikki ensimmäisen ryhmän käskyt on suoritettu tai, jos ensimmäisen ryhmän käskyt on suoritettu, ennen minkä tahansa sellaisen toisen ryhmän käskyn suoritusta, joka on alttiina mahdolliselle poikkeukselle, ja tällä tavoin mahdollistetaan seuraava asynkronisen tapahtuman käsittely.

Ensimmäinen erityinen käskyjen alisekvenssi missä tahansa jaokkeisessa toisen koodin käskysekvenssissä, joka sisältää ensimmäisen alisekvenssin, keskeytetään uudelleenyritystä varten, kunnes suoritus on onnistuneesti päättynyt, jos mainittu toinen suoritin suorittaa ristiriidassa olevan kirjoituksen ennen mainitun ensimmäisen alisekvenssin suorituksen päättymistä. Mikä tahansa jaokkeinen toisen koodin käskysekvenssi, joka sisältää mainitun ensimmäisen alisekvenssin, keskeytetään uudelleenyrittämistä varten, jos asynkroninen tapahtumakeskeytys esiintyy mainitun ensimmäisen alisekvenssin suoritusyrityksen aikana.

Asynkronisen tapahtumakeskeytyksen käsittelyä viivästetään ja mikä tahansa suoritettavana oleva toisen koodin käskysekvenssi suoritetaan loppuun A) jos mainittu toinen alisekvenssi sisältyy jaokkeiseen käskysekvenssiin ja jos tämä asynkroninen tapahtumakeskeytys esiintyy aikaisintaan ensimmäisen kirjoittamisen jälkeen mainitun toisen käskyn alisekvenssin suorituksen aikana tai B) jos tämä asynkroninen tapahtumakeskeytys esiintyy kaikkien mainitussa toisessa ryhmässä olevien, mahdolliselle keskeytykselle alttiiden tilapäivityskäskyjen suorituksen jälkeen.

Piirustusten lyhyt selitys

Oheiset piirustukset, jotka sisältyvät tähän patenttiselitykseen ja muodostavat sen osan, esittävät tämän keksinnön yhtä suoritusmuotoa ja muodostavat yhdessä seli-

tyksen kanssa tämän keksinnön tavoitteiden, etujen ja periaatteiden selityksen. Piirustuksissa:

Kuvio 1 esittää toiminnallista yleislohkokaaviota, joka osoittaa sen yleisen tavan, jolla sovellusohjelmat (a.) luodaan ensimmäisessä tietokonejärjestelmässä (jolla on ensimmäinen käskykanta ja joka on osoitettu X:llä) suoritettavaksi ja (b.) käännetään eri tietokonejärjestelmässä (joka on osoitettu Y:llä) suoritettavaksi, jolla on suhteellisesti supistettu käskykanta;

Kuvio 2 esittää parhaana pidetyn X-Y-käännösohjelman yleistettyä toiminnallista lohkokaaviota ja yleistietokonejärjestelmää, jossa X-Y-käännösohjelma suoritetaan Y:ssä suoritettavan sovelluskoodin kehittämiseksi, X:n käskyjaotusta käyttäen, syötetystä X:n sovelluskoodista;

Kuvio 3 esittää tämän keksinnön mukaisen X-Y-käännösohjelman yleistä vuokaaviota;

Kuvio 4 esittää Y-tietokonejärjestelmän toiminnallista lohkokaaviota, johon tietokonejärjestelmään Y:n sovelluskoodi ladataan, sekä käskyjaotuksen ohjausohjelmaa (IGC, instruction granularity control), etuoikeutetun arkkitehtuurin kirjaston (PAL, privileged architecture library) koodirutiinia ja latauslukittua ehdollista sekvenssiä, joka on tallennettu ohjaamaan Y-koodin suoritusta, missä käskyjaotus ja tila-atomisuus säilytetään esillä olevan keksinnön mukaan;

Kuviot 5A ja 5B esittävät vuokaavioita, jotka esittävät IGC-ohjelman suorittamia toimintavaiheita;

Kuvio 6 esittää kaaviota, joka esittää X-Y-koodinkääntämistä, joka käsittää useita kirjoittamisia, sekä asynkronisten tapahtumien suhdetta niihin;

Kuvio 7 esittää kuvion 4 PAL-koodirutiinia yksityiskohtaisesti ja osoittaa tavan, jolla sitä käytetään esillä olevan keksinnön mukaan tila-atomisuuden ja käskyjaotuksen säilyttämiseksi useiden muistiin kirjoittamisten tapauksessa; ja

Kuvio 8 esittää vuokaaviota, joka edustaa kuvion 4 latauslukittua/ehdollista tallennussekvenssiä yksityiskohtaisesti ja osoittaa tavan, jolla sitä käytetään esillä olevan keksinnön mukaan tila-atomisuuden ja käskyjaotuksen säilyttämiseksi osittaisen kirjoittamisen ja lukittujen päivityskäskyjen tapauksissa.

Parhaana pidetyn suoritusmuodon selitys

Kuten kuviossa 1 on esitetty, lähdekoodilla kirjoitettu sovellusohjelma 10 on eräs joukosta sovellusohjelmia, joita pidetään käyttäjän ohjelmakirjastossa X-tietokonejärjestelmässä 12 suorittamista varten. X-tietokonejärjestelmän 12 laitteistoarkkitehtuuri on sovitettu toimimaan X-käskykannalla, jota käytetään tuotettaessa ohjelman 10 tai muun käyttäjän kirjastossa olevan sovellusohjelman suoritettava muoto.

Ohjelman 10 sovittamiseksi käytettäväksi Y-tietokonejärjestelmässä 10 on välttämätöntä, että ohjelman 10 suoritettava muoto tuotetaan Y:ssä suoritettavissa olevana koodina 22 käyttäen Y:n käskykantaa, jolle Y-tietokonejärjestelmän 20 laitteistoarkkitehtuuri on sovitettu.

Y-käskykanta käyttää yleensä vähemmän peruskäskyjä kuin mitä X-käskykanta käyttää, ja X-koodin muunnos Y-koodiksi vaatii käskyjen muunnoksen "yhdestä moneksi". X-käskykanta voi olla CISC-käskykanta ja Y-käskykanta voi olla RISC-käskykanta. Esimerkiksi, kuten kuviossa 1 erityisesti on kuvaustarkoituksia varten esitetty, X-järjestelmä voi käyttää VAX?-arkkitehtuuria ja Y-järjestelmä voi käyttää supistetun käskykannan arkkitehtuuria, jota kutsutaan VAX? RISC-arkkitehtuuriksi Digital Equipment Corporationin piirissä. Digital Equipment Corporationin, esillä olevan patenttihakemuksen haltijan, valmistamassa laitteistossa käytetään kumpaakin arkkitehtuuria.

Kuten kuviossa 1 on esitetty, sovellusohjelma 10 voidaan siirtää Y:ssä suoritettavaksi koodiksi 22 joko epäsuoraa tietä 24 tai suoraa tietä 26. Suora siirto ai-

kaansaadaan käyttämällä Y-kääntäjää 28 ja Y-linkittäjää 30. Tulokseksi saatava Y:ssä suoritettava koodi on osoitettu viitenumerolla 22B.

Jos Y-kääntäjää 28 ja Y-linkittäjää 30 ei ole ollenkaan kehitetty tai jos ne muutoin eivät ole käytettävissä tai jos käyttäjä päättää olla käyttämättä suoraa siirtotietä 26 siihen liittyvien huonojen puolien takia, niin voidaan käyttää epäsuoraa tietä 24 X-sovellusohjelman siirtämiseksi Y-järjestelmään sekä ohjelmainvestointien säästöjen että järjestelmän paremman suorituskyvyn saavuttamiseksi.

Epäsuoralla tiellä ohjelma 10 muunnetaan suoritettavaksi koodiksi 14 Y-tietokonejärjestelmälle 12 X-kääntäjän 16 ja X-linkittäjän 18 avulla. Tulos on X:ssä suoritettava koodi 14, joka voi toimia X-tietokonejärjestelmässä 12. X-Y-kääntäjä 32 kääntää X:ssä suoritettavan koodin, joka on merkitty viitenumerolla 22A. Koodin kääntäminen aikaansaadaan siten, että Y-koodin käsittely X-koodin aikaansaamiseksi tapahtuu tarkasti käskyjaotuksen mukaan, vaikka Y-koodi perustuukin supistettuun käskykantaan.

X-Y-kääntäjää 32 käytetään parhaana suoritusmuotona X:ssä suoritettavan koodin 14 kääntämiseksi vastaavaksi Y:ssä suoritettavaksi sovelluskoodiksi, joka on osoitettu viitenumerolla 22A. Koodinkäännös aikaansaadaan siten, että Y-koodin suorittaminen antaa tarkasti X-koodin tulokset samalla käskyjaotuksella ja tila-atomisuudella, vaikka Y-koodi perustuu supistettuun käskykantaan.

X-sovelluskoodin kääntäminen Y-sovelluskoodiksi

Koodinkäännösjärjestelmää 40 (kuvio 2) käytetään kuviossa 1 ilmenevän kääntäjän 32 toteuttamiseksi. Käännösjärjestelmä 40 käsittää tavanomaisen yleistietokoneen, jossa on suoritin 42, muistijärjestelmä 44 ja erilaisia syöttö-/tulostuslaitteita, joiden välityksellä X-sovelluskoodi 43 syötetään käännettäväksi.

Käännöksen tulokset kehitetään esillä olevan keksinnön mukaan Y-koodiksi 45, joka on järjestetty ja muutoin strukturoitu käännettävän koodin laitetakuiden määrittelemällä tavalla. Y-koodi 45 on erityisesti strukturoitu helpottamaan X-käskyjaotuksen ja tila-atomisuuden taattua säilyttämistä, kun Y-koodi todellisuudessa suoritetaan. Esimerkkinä CISC-laitetakuista VAX?-arkkitehtuuri sisältää seuraavat takuut ja standardit:

1. Yhden käskyn täytyy joko mennä loppuun asti tai näyttää siltä, ettei sitä ole aloitettukaan - ei sallita käskyn osittaista suorittamista, sen keskeyttämistä, toisten käskyjen suorittamista eikä keskeytetyn käskyn mahdollista uudelleenaloitusta keskeltä.

2. Muisti on virtuaalinen, joten mikä tahansa muistiosoitus voi kohdata sivuvirhe- tai osoitussuojauspoikkeuksen, joka aiheuttaa sen, että käskyä ei suoriteta loppuun.

3. Yksi käsky voi kirjoittaa useisiin muistipaikkoihin; joko kaikkien kirjoittamisten täytyy tapahtua tai yhtäkään ei saa tapahtua. Jos yhtäkään ei tapahdu, käsky aloitetaan uudelleen alusta eikä siitä kohdasta, jossa kirjoittaminen epäonnistui.

4. Muistioperandit voivat olla (osittain) limittäin siten, että yhden monista kirjoittamisista suorittaminen ja pysähtyminen sen jälkeen voi kirjoittaa lähdeoperandien päälle ja tehdä käskyn uudelleenaloittamisen mahdottomaksi.

5. Yksi käsky voi suorittaa luku-muutos-kirjoitussekvenssin.

6. Käskyjen operandien sallitaan olla mielivaltaisen tavupituuden omaavia mielivaltaisissa tavuosoitteissa, kun taas toteutetut muistilaitteet tyypillisesti voivat lukea tai kirjoittaa vain kokonaislukuna ilmaistavan määrän kohdakkain olevia muistisanoja, jota käsittävät tyypillisesti 4 tai 8 tavua. Yksi operandi voi siten kattaa

2 tai useampia muistisanoja, ja yhden operandin osoittaminen voi käsittää ylimääräisten tavujen osoittamisen ensimmäisessä ja viimeisessä muistisanassa.

7. Monisuoritinjärjestelmässä eri suorittimien suorittamien osoitusten vierekkäisiin tavuihin täytyy aina näyttää olevan itsenäisiä - ts. tavun 5 kirjoittaminen yhdellä suorittimella ei saa häiritä tavun 6 kirjoittamista toisella suorittimella, vaikka kumpikin kirjoittaminen käsittäisi samaan muistisanaan osoittavan luku-muutos-kirjoitus-sekvenssin.

8. Monisuoritinjärjestelmässä lukittujen käskyjen avulla tapahtuvien osoitusten täytyy näyttää olevan atomisia - ts. lukittu luku-muutos-kirjoitus tavuun 4 yhdellä suorittimella ei saa koskaan häiritä samaan paikkaan kohdistuvaa lukittua luku-muutos-kirjoitus-toimintaa toisella suorittimella.

Muistijärjestelmä 44 käsittää muiden osien lisäksi tavanomaisen datamuistiosan 46 ja osan 48, johon tietokoneen käyttöjärjestelmä on tallennettu. X-Y-koodinkääntämisessä käytettävä peruselementti on käännösohjelma 50, joka on tallennettu toiseen muistinosaan. Syötetty X-koodi 43 tallennetaan X-koodilistaksi 62. Lisäksi Y-käskyjen sekvensoinnin ohjaamiseksi on tallennettu Y-käskyjen järjestämiskriteerit 52 ja X-Y-käskykoodirakenteet 54 sekä käskyjen operaatiotarkentimien että käskyjen operandien osoitetarkentimien kääntämisen mahdollistamiseksi.

Käännösohjelman 50 yleinen vuokaavio on esitetty parhaana pidetyssä muodossaan kuviossa 3. Lohkossa 60 peräkkäiset käskyt syötetään peräkkäisessä järjestyksessä tallennetusta X-koodilistasta 62 ohjelmasilmukassa 64 käsiteltäväksi.

Silmukassa 64 toimintolohko 66 kehittää Y-käskyn operaatio- ja operanditarkentimet, jotka vastaavat kulloinkin käsiteltävää X-käskyä, tallennettujen X-Y-koodirakenteiden 54 (kuvio 2) mukaan. Seuraavaksi, kuten toimin-

tolohko 68 esittää, tulokseksi saatu Y-koodi järjestetään ennalta määrättyjen kriteerien mukaan, jotka helpottavat X:n käskyjaotuksen säilyttämistä Y-koodin myöhemmin tapahtuvan varsinaisen suorittamisen aikana.

Graafinen esitys X-Y-käskyjen kääntämisestä on esitetty kuviossa 6.

Jokainen X-käsky yleisesti suorittaa sellaisia alkeistehtäviä kuten syötteiden ottaminen, syötteiden muuntaminen, tulosten asettaminen väliaikaiseen muistiin ja tilan päivityksen suorittaminen muisti- ja rekisteripaikoilla. Kun X-käsky käännetään "moneksi" Y-käskyksi, Y-käskyjen organisoimiseksi käytettävät järjestämiskriteerit 52 (kuvio 2) ovat mieluimmin sellaiset, jotka ryhmittävät ja järjestävät X-käskyt Y-koodissa kulloinkin käännettävän X-käskyn (jaokkeen) osalta seuraavasti:

1. Ensimmäisenä käskyryhmänä G1 Y-koodissa ovat ne käskyt, jotka ottavat syötteitä ja asettavat nämä syötteet väliaikaiseen muistiin.

2. Toisena käskyryhmänä G2 Y-koodissa ovat ne käskyt, jotka operoivat syötteillä, kehittävät muunnettuja tuloksia ja tallentavat nämä tulokset väliaikaiseen muistiin.

3. Kolmantena käskyryhmänä G3 Y-koodissa ovat ne käskyt, jotka päivittävät X:n tilan (muistin tai rekisterin) ja jotka ovat alttiina mahdollisille poikkeuksille (kuten jäljempänä on määritelty).

4. Neljäntenä ja viimeisenä käskyryhmänä G4 Y-koodissa ovat ne, jotka päivittävät X:n tilan (muistin tai rekisterin) ja jotka eivät ole alttiina mahdollisille poikkeuksille.

X:n muistitila, jota edustaa viitenumero 95 kuviossa 4, ja X:n rekisteritila, jota edustaa viitenumero 97 kuviossa 4, viittaavat siihen Y-koneessa olevaan muisti- ja rekisterirakenteeseen, jossa on varattu sisältämään X-koodin määrittelemät muistipaikat. X:n muisti- ja rekis-

teritilojen voidaan myös sanoa olevan se muistitila ja rekisteritila, jotka ovat X-arkkitehtuurille näkyviä.

Lisäinformaatiota käännetyn koodin käskyjen selitetyllä tavalla tapahtuvan organisoinnin eduista, erityisesti sovellettaessa yksinkertaiseen yhden kirjoittamisen Y-käskyihin, on esitetty viitteenä mainitussa patenttihakemuksessa 1870-0410.

Tila-atomisuus olennaisesti vaatii, että kaikki X-tilaosoitukset näyttävät tapahtuvan ilman väliintuloa tai ilman että yhtäkään näyttää tapahtuvan. Tämä ehto tarvitaan X-käskyjaotuksen aikaansaamiseksi. Alempana selitetyssä erikoistapauksessa X:n tila-atomisuus aikaansaadaan esillä olevan keksinnön toiminnan avulla, jolloin mahdollistetaan X-käskyjaotuksen aikaansaaminen.

Kuvion 3 mukaan, kun toimintolohko 68 järjestää Y-käskykoodin kuten selitetty, lohko 70 ratkaisee, onko kulloinenkin Y-käsky rajamerkki "X-jaokkeen" merkki sillä käskyllä, josta se on johdettu. Peräkkäin käsitellyillä Y-käskyillä ilmaistut kyllä- ja ei-bitit toimintolohko 72 tallentaa X-rajakäskyjen bittikarttaan.

Seuraavaksi suoritetaan joukko testejä sen ratkaisemiseksi, mitä useista käännöksen käsittelyhaaroista on seurattava. Kukin haara vastaa yleisesti ennalta määriteltyä käännöstapausta (ts. X-käskyn lajia) niillä ennalta määritellyillä tapauksilla, jotka on luokiteltu erityisiksi ja jotka vaativat erityiskäännöksen suorittamista muistin atomisuuden säilyttämiseksi. Tässä suoritusmuodossa voidaan seurata mitä tahansa kolmesta haarasta 65, 67 ja 69 kulloinkin suoritettavan X-käskyn rakenteellisen luonteen mukaan.

Yleisesti ottaen haarat 65, 67 ja 69 käsittelevät kukin X-käskyn ja tuottavat käännetyn koodin, joka yleisesti säilyttää laitteistotakuut ja joka erityisesti säilyttää tila-atomisuudet.

Huomattakoon vielä, että käännöksen tarkoituksia varten oletetaan, että yhden kohdennetun pitkän sanan (longword, 4 tavua) tai kvadisanan (quadword, 8 tavua) yksi RISC-tallennuskäsky on atominen RISC-koneessa siinä mielessä, että kaikki tavut muutetaan samanaikaisesti, ja tallennus ei vaikuta mihinkään muihin tavuihin. Lisäksi oletetaan, että koko tila pidetään joko muistissa tai rekistereissä, että muistiosoitukset voivat aikaansaada virtuaalimuistipoikkeuksia ja että yksinkertaiset rekisterisiirrot eivät koskaan aikaansaa poikkeuksia. Lisäksi oletetaan, että ulkoiset tapahtumat voivat keskeyttää RISC-käskyjen sekvenssin mielivaltaisen RISC-käskyn kohdalla.

Kuviossa 3 käännöshaaraa 65 seurataan, jos testilohko 120 ilmaisee, että kulloinenkin X-käsky on yksinkertainen yhden kirjoittamisen käsky. Tässä tapauksessa käsittely suoritetaan kuten viitteenä mainitussa patenttihakemuksessa 1870-0410 on esitetty. Erityisesti kuten toimintolohko 122 osoittaa, mitään erityistä käännöstyötä ei tarvita, ja lohko 74 ratkaisee, onko käännettäviä X-käskyjä vielä jäljellä. Jos on, ohjelmasilmukan 64 suoritus toistuu.

Tarkastellaan uudelleen käännöksen vuokaaviota. Jos kulloinenkin X-käsky ei ole yksinkertainen yhden kirjoittamisen käsky, niin testilohkossa 124 ratkaistaan, onko X-käsky yksi useista ennalta määritellyistä erityisistä yhden kirjoittamisen tapauksista. Esillä olevassa suoritusmuodossa on kaksi ennalta määriteltyä erityistä yhden kirjoittamisen tapausta, ts. osittainen kirjoituskäsky ja lukittu päivityskäsky. Jos jompikumpi erityisistä yhden kirjoittamisen tapauksista esiintyy, niin toimintolohko 126 toimii käännöshaarassa 67 tila-atomisuuden säilyttämiseksi sijoittamalla tila-atomisuussekvenssin käännettyyn koodiin. Tila-atomisuussekvenssi takaa (ajon aikana) joko 1) osittaisen kirjoittamisen tai lukitun päivityksen loppuunsuorittamisen, jos keskeytystä ei tapahdu luku-muutos-

kirjoitus-sekvenssin aikana, tai 2) osittaisen kirjoittamisen tai lukitun päivityksen keskeytyksen uudelleenyritystä varten, jos keskeytys tapahtuu luku-muutos-kirjoitus-sekvenssin aikana.

Lohkossa 126 käännettyyn koodiin sijoitettu tila-atomisuussekvenssi on mieluimmin sellainen, jota kutsutaan latauslukituksi/ehdolliseksi tallennussekvenssiksi. Sopiva laitteistomekanismi tämän sekvenssin toteuttamiseksi Y-tietokonejärjestelmässä ajon aikana on paljastettu viitteenä mainitussa Digital Equipment Corporationin patenttihakemuksessa PD90-0259.

Kuviossa 8 on esitetty yleistetty vuokaavio viitenumerolla 126A osoitetun latauslukitun/ehdollisen tallennussekvenssin suorittaman ajonaikaisen logiikan kuvaamiseksi. Siten kun sekvenssiä 126A kutsutaan, toimintolohko 128 lataa sen muistisanan, jolla luku-muutos-kirjoitus- (RMW, read-modify-write) operaatio on määrä suorittaa.

Lohkot 130, 132 ja 134 suorittavat muutostehtävät. Kuvaavassa tapauksessa, jossa määriteltyyn muistipaikkaan lisätään tavu, lohko 130 suorittaa siirto-operaation tavukohdennuksen aikaansaamiseksi muistisanassa. Seuraavaksi lohko 132 peittää muutettavan tavun nollilla. Lopuksi lohko 134 panee peitetyn tavun paikalleen muistisanaan.

Jos RMW-sekvenssin suorituksen kuluessa ajon aikana toinen suoritin kirjoittaa samaan muistipaikkaan, niin RMW-sekvenssi ei onnistu estämään näiden kahden itsenäisen muistiin kirjoittamisen välistä häiriötä. Lohko 136 suorittaa ehdollisen tallennuksen sen ilmaisemiseksi, onko toinen muistiin kirjoittaminen samaan muistipaikkaan tapahtunut RMW-sekvenssin luku- ja muutososien aikana. Jos toista kirjoittamista ei ole tapahtunut, ehdollinen tallennus toteutetaan siten, että muutettu muistisana kirjoitetaan tila- tai muistiatomisuus säilyttäen, koska ei esiintynyt toisen kirjoittamisen aiheuttamaa ristiriitaa. Testilohko 138 päättää sekvenssin. Kulloisenkin Y-käskyn

jatkokäsittelyä ajon aikana ohjataan silloin käskyjaotuksen ohjausohjelmoinnin mukaan, joka on täydellisemmin selitetty jäljempänä.

Jos lohkossa 136 toinen suoritin on suorittanut kirjoittamisen RMW-sekvenssin luku- ja muistiosien aikana muutettavan sanan muistipaikkaan, niin ehdollinen tallennus peruutetaan ja siten säilytetään muistin atomisuus, ja RMW-sekvenssiä yritetään myöhemmin uudelleen.

Tarkastellaan jälleen kuvion 3 käännösprosessia. Sen jälkeen kun sekvenssi 126A on sijoitettu kulloiseenkin Y-käskyyn, lohko 74 testaa on lisää X-käskyjä. Jos niitä vielä on, lohko 67 palaa käskyyn, joka suorittaa siirtymisen lohkoon 60 ohjelmasilmukan toistamiseksi.

Yhteenvetona haaran 67 suorittaman yhden kirjoittamisen CISC-RISC-käännöksen käsittelyn tapauksesta todetaan, että CISC-käskyllä on yksi osittainen (1 tai 2 tavun) kirjoittaminen tilaan, ja käännetty koodi suoritetaan monisuoritinjärjestelmässä eli tarvitaan lukittu osoitus. Tässä tapauksessa itsenäinen tavuosoitus ja luku-muutos-kirjoitus-sekvenssi täytyy käsitellä sopivasti, jos tila-atomisuus on määrä säilyttää.

Haaran 67 suorittaman yhden kirjoittamisen tapauksessa käännös on rajoitettu siten, että ryhmän 1 ja 2 käskyt tekevät kaiken CISC-käskyjen työn lukuunottamatta muisti- ja/tai rekisteritilan päivitystä, mukaanlukien päivitettävän operandin sisältävän pitkän sanan/kvadisanan latauslukittu päivitys. Ryhmän 3 käskyt sisältävät ehdollisen tallennuskäskyn samaan pitkään sanaan/kvadisanaan, ja ryhmän 4 käskyt sisältävät käskyt, jotka epäonnistuessaan haarautuvat sekvenssin alkuun, mitä seuraa yksinkertaisia rekisterisiirtoja.

Käännetty sekvenssi, joka keskeytetään ennenkuin käskyryhmä 3 on suoritettu loppuun, epäonnistuu ehdollisessa tallennuskäskyssä kun se aloitetaan uudelleen, ja haarautuu siksi takaisin sekvenssin alkuun. Lisäksi

jos toinen suoritin kirjoittaa määriteltyyn pitkään sanaan/kvadisanaan latauslukitun käskyn jälkeen, mutta ennen ehdollista tallennuskäskyä, niin ehdollinen tallennus epäonnistuu ja haarautuu siksi takaisin sekvenssin alkuun.

Käännetty sekvenssi, joka keskeytetään sen jälkeen kun käskyryhmä 3 on suoritettu loppuun, mutta ennenkuin käskyryhmä 4 on suoritettu loppuun, pakotetaan suorittamaan käskyryhmä 4 loppuun mekanismilla, joka tulkitsee eteenpäin yksinkertaisilla rekisterisiirroilla, kuten viitteenä mainitussa patenttihakemuksessa 1870-0410 on täydellisemmin esitetty. Nettovaikutuksena on, että kukin käännetty sekvenssi joko suoritetaan alusta loppuun ilman mitään keskellä olevaa käännettyä sekvenssiä ja ilman muuta kirjoittamista kohteena olevaan pitkään sanaan/kvadisanaan tai tämän sekvenssin suoritus keskeytetään ennen ryhmän 3 loppuun suorittamista ja suoritusta yritetään seuraavaksi uudelleen alusta.

Tarkastelleen jälleen kuviota 3. Jos testilohko 124 havaitsee, että käännettävässä X-käskyssä ei ole mitään erityistä yhden kirjoittamisen käskyä, niin mennään käännöshaaraan 69 ja lohko 128 panee merkille sen seikan, että käännettävä X-käsky on toinen erikoistapaus, ts. moninkertaisen kirjoittamisen X-käsky. Toimintolohko 130 sijoittaa käännettyyn käskykoodiin tällöin PAL_CALL-rutiinin 132 (kuvio 7) tila-atomisuuden aikaansaamiseksi usean kirjoittamisen käskyillä tavalla, joka on täydellisemmin selitetty jäljempänä.

PAL_CALL-rutiini 132 kutsutaan suoritettavaksi ajon aikana etuoikeutetun arkkitehtuurin kirjastosta (Privileged Architecture Library), joka sisältyy tietokonejärjestelmään 20 ja on tyypillisesti käytettävissä monissa tietokonearkkitehtuureissa sellaisen mekanismin aikaansaamiseksi, joka suorittaa siitä kutsutut rutiinit korkeimmalla käyttöjärjestelmäprioriteetilla. Yleisesti ottaen PAL_CALL-rutiini 132 suorittaa kaikki tilakirjoi-

tukset tila-atomisuuden säilyttäen, jos mitään asynkronisia tapahtumakeskeytyksiä ei ole esiintynyt ennen rutiinin kutsumista ja jos mitään mahdollisia poikkeuksia ei ole havaittu jäljellä olevassa sekvenssissä kulloisessakin Y-koodin käskyjaokkeessa. Muussa tapauksessa PAL_CALL-rutiini 132 epäonnistuu ennenkuin se suoritetaan, ja muistiatomisuus säilyy seuraavaa uudelleenyritystä varten.

PAL_CALL-rutiini 132 toteutetaan mieluimmin viitteenä mainitussa patenttihakemuksessa PD86-0114 paljastetulla ja selitetyllä laitteistorakenteella.

Tarkastelleen kuviota 6, jossa on kaaviollinen esitys asynkronisten tapahtumien suhteesta Y-koodisiin käskyihin useiden kirjoittamisten tapauksessa. Tässä erikoistapauksessa nuoli 75 osoittaa asynkronista tapahtumaa, joka esiintyy sen jälkeen kun ensimmäinen kirjoittaminen on käsitelty (PAL_CALL-rutiinilla 132), mutta ennenkuin kaikki useat kirjoittamiset on suoritettu PAL_CALL-rutiinilla 132. Yleisesti ottaen tila-atomisuus voidaan säilyttää tässä tapauksessa viivästämällä keskeytyksen suorittamista, kunnes PAL_CALL-rutiini 132 ja loppuosa käskyjaokkeesta on suoritettu.

Viitataan erityisesti PAL_CALL-rutiinin 132 kuviossa 7 esitettyyn vuokaavioon. Sen jälkeen kun käännetyn koodin ajon aikana suoritettavana oleva Y-koodinen jaoke on kutsunut PAL_CALL-rutiinia 132, mennään tähän rutiiniin kuten toimintolohko 152 osoittaa. Seuraavaksi lohko 154 testaa, onko keskeytys tapahtunut RS:n jälkeen ja ennen PAL_CALL-rutiinin 132 kutsumista. Jos on, lohko 156 palaa käskysekvenssin epäonnistumisesta ilmoittavaan sanomaan tila-atomisuuden säilyttämiseksi, ohjaus palautetaan kutsuvalle proseduurille poistumislohkon 157 kautta kulloisenkin Y-koodisen jaokkeen uutta suoritusyritystä varten.

Jos keskeytyksiä ei ole tapahtunut PAL_CALL-rutiinia edeltävän kriittisen ajan kuluessa, niin testilohko 158 ratkaisee, voidaanko kaikki jäljellä olevat tilaosoi-

tukset suorittaa loppuun ilman poikkeuksia. Ellei, lohko 156 palaa jälleen suoritetun käskysekvenssin epäonnistumisesta ilmoittavaan sanomaan tila-atomisuuden säilyttämiseksi, ja rutiinista poistutaan lohkon 157 kautta Y-koodisekvenssin uudelleenyrityksen sallimiseksi.

Jos jäljellä olevat tilaosoitukset voidaan suorittaa loppuun ilman poikkeuksia, PAL_CALL-rutiinin 132 suoritus aloitetaan ja toimintolohko 159 suorittaa ensimmäisen kirjoittamisen latauslukituksen ja muutoksen, jos osittainen kirjoittaminen on määritelty. Lohko 160 suorittaa tällöin ehdollisen tallennuksen osittaisella kirjoittamisella tai tallennuksen täydellisellä kirjoittamisella.

Testilohko 162 ratkaisee, onko ehdollinen tallennus epäonnistunut osittaisen kirjoittamisen tapauksessa. Jos on, rutiini palautetaan mieluimmin lohkoon 159 osittaisen kirjoituksen uudelleenyrittämistä varten kuten esitetty. Haluttaessa voidaan tässä kohdassa palauttaa epäonnistumisesta ilmoittava sanoma.

Jos täydellinen kirjoittaminen on tallennettu tai kun osittainen kirjoittaminen on onnistuneesti tallennettu, niin ensimmäinen kirjoittaminen usean kirjoittamisen sekvenssissä on tapahtunut, ja kaikki kirjoittamiset Y-koodisekvenssissä täytyy suorittaa loppuun tila-atomisuuden säilyttämiseksi, kuten toimintolohko 164 osoittaa.

Seuraavaksi esikirjoituslohko 166 käsittelee toisen kirjoittamisen Y-koodisekvenssissä, tallennus- tai ehdollinen tallennus -lohko 168 toteutetaan ja sen jälkeen suoritetaan testilohko 170, joka testaa, onko ehdollinen tallennus epäonnistunut. Tämä toiminta on samanlainen kuin se tapa, joka on selitetty lohkojen 158, 160 ja 162 suorittaman ensimmäisen kirjoittamisen käsittelyn osalta. Sama toimintojen osajoukko (ts. ne toiminnot, jotka sisältyvät lohkoihin 158, 160 ja 162) suoritetaan kullakin seuraavalla kirjoittamisella usean kirjoittamisen sekvenssissä, kuten yksi ainoa toimintolohko 172 esittää, kun osittai-

nen tai täydellinen kirjoittaminen on onnistuneesti suorittanut loppuun kunkin edeltävän kirjoittamisen. Kun kaikki kirjoittamiset on tallennettu, lohko 174 ilmaisee PAL_CALL-rutiinin 132 suorittamisen onnistuneen, ja poistuminen tapahtuu lohkon 157 kautta. Kulloinenkin Y-käskyjaoke voidaan silloin suorittaa loppuun tila-atomisuus säilyttäen.

Kuten käännöshaarojen 65 ja 67 tapauksessa kuviossa 3, haara 69 lopuksi tarkistaa, tarvitseeko kääntää lisää X-käskyjä, ja jos näin on, lohko 60 syöttää seuraavan X-käskyn käännettäväksi.

Yhteenvetona haaran 69 suorittamasta CISC-RISC-käännöksen käsittelystä todettakoon, että CISC-käskyllä on enemmän kuin yksi kirjoittaminen tilaan (johtuen useista kohteista tai yhdestä kohdentamattomasta kohteesta). Mitkään lukitut tilaosoitukset eivät liity tähän tapaukseen, mutta itsenäiset tavuosoitukset täytyy asianmukaisesti käsitellä, ja kaikki tai ei yhtään määritellyistä kirjoittamisista täytyy suorittaa, jos tila-atomisuus on määrä säilyttää.

Käännös on rajoitettu siten, että käskyryhmät 1 ja 2 alkavat lue ja aseta (read-and-set) -käskyllä. Näitä ryhmiä seuraavat käskyt, jotka suorittavat kaiken CISC-käskyn työn lukuunottamatta muisti- ja/tai rekisteritilan päivitystä, mukaanlukien mahdollisesti kunkin päivitettävän pitkän sanan/kvadisanan lataukset. Käskyryhmä 3 sisältää PAL_CALL-rutiinin 132, joka määrittelee kaikki tallennukset, ja käskyryhmä 4 sisältää haarautumisen epäonnistumisen tapauksessa (branch-on-fail) sekvenssin alkuun, mitä seuraavat yksinkertaiset rekisterisiirrot.

Käännetty sekvenssi, joka keskeytetään ennenkuin käskyryhmä 2 on suoritettu loppuun, nollaa RISC-tilabitin, jonka lue ja aseta on asettanut, mikä aiheuttaa PAL_CALL-rutiinin 132 palaamisen epäonnistumisen ilmoittavaan sanomaan ja siksi haarautumaan takaisin sekvenssin alkuun.

Sellaista laitteistorakennetta käytettäessä, joka on selitetty viitteenä mainitussa patenttihakemuksessa PD86-0114, PAL_CALL-rutiini 132 menee ei-keskeytettävän RISC-koodin etuoikeutettuun sekvenssiin.

PAL_CALL-rutiini 132 ei tee tallennuksia ja palauttaa epäonnistumisesta ilmoittavan sanoman, kun väliin tulevan keskeytys on esiintynyt, ts. jos lue ja aseta -käskyn asettama RISC-tilabitti on nollattu. Muussa tapauksessa suoritetaan kaikkien mahdollisten tallennuspaikkojen tutkiminen tarkistamalla, onko mitään mahdollisia virtuaalimuistipoikkeuksia. Jos jokin sellainen tavataan, PAL_CALL-rutiinia 132 ei suoriteta loppuun, poikkeukset käsitellään, ja lue ja aseta -käskyn asettama tilabitti nollataan. Tämä aiheuttaa seuraavaksi PAL_CALL-rutiinin 132 uudelleensuorittamisen epäonnistumisesta ilmoittavan sanoman palauttamiseksi ja siksi haarautumisen takaisin sekvenssin alkuun. Muussa tapauksessa PAL_CALL-rutiini 132 suorittaa kaikki annetut tallennukset. Sen tehdessään se käyttää edellisessä tutkimisessa käytettyä virtuaalimuisti-informaatiota, vaikka toinen suoritin olisikin samanaikaisesti päivittämässä yhteisen muistin sivutauluja. Tallennukset eivät siten kehitä mitään virtuaalimuistipoikkeuksia.

Kullakin osittaisella muistisanan tallennuksella (partial-memory-word store) etuoikeutettu koodi käyttää latauslukittu/muutos/ehdollinen tallennus -sekvenssiä. Mitään aikaisempia tallennuksia ei ole suoritettu ja ensimmäinen tällainen ehdollinen tallennus epäonnistuu (koska toinen suoritin tallentaa samaan muistisanaan muutoksen aikana), ja toteutus voi joko palauttaa "epäonnistui"-ilmoituksen PAL_CALL-rutiinista 132 tai se voi toistaa vain latauslukittu/muutos/ehdollinen tallennus -sekvenssin. Sen jälkeen kun yksi tallennus on tapahtunut, seuraavat latauslukittu/muutos/ehdollinen tallennus -sekvenssit täytyy toistaa etuoikeutetun koodin sisällä, kunnes ne onnistu-

vat. Kaikkien määriteltyjen tallennusten tultua suoritetuksi etuoikeutettu koodi palaa onnistuneesti, ja käskyryhmä 3 on suoritettu loppuun.

Käännetty koodi, joka keskeytetään sen jälkeen kun käskyryhmä 3 on suoritettu loppuun, mutta ennenkuin käskyryhmä 4 on suoritettu loppuun, pakotetaan suorittamaan käskyryhmä 4 loppuun mekanismilla, joka tulkitsee eteenpäin yksinkertaisten rekisterisiirtojen avulla, kuten viitteenä mainitussa patenttihakemuksessa 1870-0410 on täydellisemmin esitetty. Nettovaikutuksena on, että sekvenssi joko suoritetaan alusta loppuun ilman että keskellä on mitään muuta käännettyä sekvenssiä ja ilman mitään väliin tulevaa kirjoittamista kohteena oleviin muistisanoihin tai se keskeytetään ennen käskyryhmän 3 suoritusta ja sitä yritetään seuraavaksi uudelleen alusta.

Kun kaikki X-käskyt on käännetty haarojen 65, 67 ja 69 avulla, ohjelmasilmukan 64 syklinen suoritus päättyy ja kertynyt Y-koodi asetetaan käytettäväksi tulosteena kuten toimintolohko 76 osoittaa.

Y-tuloskoodin suoritus X:n tila-atomisuuden ja käskyjaotuksen takaavalla tavalla

Kuten kuviossa 4 on esitetty, Y-suoritinta 80, joka vastaa Y-tietokonejärjestelmää 20 (kuvio 1) käytetään Y-tuloskoodin suorittamiseen, missä X:n tila-atomisuuden ja käskyjaotuksen säilyminen on taattu. Tavanomaiset datansyöttö-/tulostuslaitteet 82 ja ajastuskello 84 on liitetty Y-suorittimeen 80, ja aika ajoin nämä laitteet kehittävät keskeytyksiä, jotka muodostavat asynkronisia tapahtumia, jotka vaativat suorittimen toiminnan tilapäistä poikkeamista Y-koodin suorittamisesta. Ilman esillä olevan keksinnön aikaansaamaa suojaustakuuta näiden tai muiden keskeytysten aiheuttama suorittimen poikkeaminen pystyy aiheuttamaan X:n tila-atomisuuden ja jaokkeisuuden rikkoutumisen erityistä lajia olevia käskyjä suoritettaessa kuten edellä on selitetty.

Kuten kuviossa 4 on esitetty, lohko 86 edustaa kehitetyn Y-koodin syöttämistä syöttölaitteelta Y-suorittimeen 80 liitetyn muistijärjestelmän 90 osaan 88, ja lohko 87 edustaa Y-koodin suorituksen seurauksena tulostuslaitteille kehitettyjä datatulosteita. Muistijärjestelmä 90 käsittää myös tavanomaisen dataosan 92, tavanomaisen käyttöjärjestelmäosan 94 sekä edellä mainitun X:n muistitilan 95. Y-suoritin 80 sisältää edellä mainitun X:n rekisteritilan 97.

Käskyjaotuksen ohjausohjelma (IGC, instruction granularity control program) 96 muistijärjestelmässä 90 on strukturoitu valvomaan Y-koodin suoritusta käskyjaotuksen säilyttämiseksi. Kuvioissa 5A ja 5B esitetty vuokaavio esittää täydellisemmin IGC-ohjelman 96 toimintaa Y-koodia suoritettaessa.

Yleisesti ottaen IGC-ohjelma 96 ohjaa Y-koodisiksi käskyjaokkeiksi käännettyjä yhden kirjoittamisen ja useiden kirjoittamisten X-käskyjä Y-koodin suorittamisen aikana. Yksinkertaisten yhden kirjoittamisen käskyjen tapauksessa kaikki tila-atomisuuden ja käskyjaotuksen ohjaus suoritetaan kuten tässä patenttiselityksessä että edellä viitteenä mainitussa patenttihakemuksessa 1870-0410 on paljastettu. Erityisten yhden kirjoittamisen käskyjen ja useiden kirjoittamisten käskyjen tapauksessa tila-atomisuuden ohjaus suoritetaan näiden käskyjen käännetyn koodin suorittamisen yhteydessä kuten tässä on kuvioiden 3, 6 ja 8 yhteydessä selitetty ja muussa tapauksessa IGC-ohjelman 96 avulla suoritetulla käsittelyllä.

Tarkemmin sanoen IGC-ohjelma 96 (kuvio 5A) alkaa kuten lohkossa 98 on osoitettu asynkronisen tapahtuman kehittämisellä. Yleisesti ottaen asynkroninen tapahtuma on määritelty Y-käskyvirran poikkeamisena, joka aiheutuu keskeytyksistä, jotka voisivat mahdollisesti kehittää X:n tilanmuutoksia, jotka ovat käännetylle X-koodille näkyviä. Viitataan jälleen kuvioon 6, jossa on kuvaava esitys asyn-

kronisten tapahtumien suhteesta sisältävien Y-käskyjen X-jaokkeeseen, joka sisältää useita kirjoittamisia.

Tarkastellaan edelleen kuviossa 5A olevaa vuokaaviota. Lohko 100 asettaa väliaikaisen pidon asynkronisen tapahtuman käsittelylle, ja toimintolohko 102 merkitsee muistiin asynkronisen tapahtuman aikana käsiteltävän Y-käskyn muistiosoitteen (merkitty PC-AE).

Seuraavaksi lohko 104 tarkistaa rajakäskyjen bittikartan ja ratkaisee, onko Y-käsky PC-AE X-käskyraja. Jos se on, testilohko 106 ohjaa IGC-ohjelman haaran 107 kautta lohkoon 108, joka Y-koodin suorituksen keskeyttämisen asynkronisen tapahtuman käsittelemiseksi rikkomatta X-koodin käskyjaotusta.

Jos Y-käsky PC-AE ei ole X-käskyraja, niin toimintolohko 110 kohdentaa Y:n käskylaskurin PC seuraavaan tallennettuun tai myötäsuunnassa olevaan Y-käskyyn, joka on X-käskyraja. IGC-ohjelma 96 seuraa sen jälkeen ohjelmahaaraa 111 lohkoon 108, joka suorittaa asynkronisen tapahtuman käsittelyn kuten edellä on selitetty, nytkin rikkomatta X-käskyjaotusta. Tässä tapauksessa asynkroninen tapahtuma on esiintynyt sellaisella ajanhetkellä, jolloin vain yksi tai useampi X-käskyjaokkeessa oleva käsky on suoritettu, mutta kaikkia Y-käskyjä ei ole suoritettu, ja X-käskyjaotuksen säilyminen aikaansaadaan ohjelmalohkon 110 toiminnan avulla.

Tarkemmin sanoen, kuten kuviossa 5B olevalla lohkolla 110 on esitetty, ohjelmasilmukassa 113 oleva toimintolohko 112 suorittaa Y-käskyjen myötäsuuntaisen selauksen seuraavan Y-käskyn löytämiseksi, joka on X-käskyrajalla. Testilohko 114 tarkistaa kunkin myötäsuunnassa selatun Y-käskyn ja ratkaisee, voisiko keskeytyksen käsittely ennen jäljellä olevien Y-käskyjen suorittamista tuottaa sellaisen Y-käskyjen suoritustuloksen, joka eriää siitä tuloksesta, joka tuotettaisiin, jos vastaava X-koodi suoritettaisiin saman asynkronisen tapahtuman vaikuttaessa.

Suorittaessaan kunkin myötäsuunnassa olevan Y-käskyn testauksen testilohko 114 mieluimmin ratkaisee, voisiko Y-käskyn suoritusyritys tuottaa poikkeustilanteen, jos asynkronisen tapahtuman käsittely sallittaisiin ja Y-koodisekvenssin suoritus sitten aloitettaisiin uudelleen. Yleisesti ottaen käskyllä on poikkeus, jos sitä mahdollisesti ei voi suorittaa loppuun. Seuraavat ovat ne yleiset poikkeusluokat, jotka myötäsuuntaiseen Y-käskyyn samaistuessaan, kehittävät Y-koodin keskeytyksen seuraavalle varmennetulle Y-käskylle, joka on X-raja:

1) Muistinhallintapoikkeukset kuten osoituksen ohjausvirheet tai sivuvirheet.

2) Aritmeettiset poikkeukset kuten liukuluvun ylivuotovirheet tai nollalla jako -virheet.

3) Käskypoikkeukset kuten luvattomat operaatiokoodit tai keskeytysoperaatiokoodit.

Tämän keksinnön parhaana pidetyssä suoritusmuodossa luettelo sovellettavissa olevista poikkeuksista käännettävällä koodilla sijoitetaan IGC-ohjelman 96 suorituksen aikana lukemaan muistiin. Poikkeuksien määritykset suoritetaan siis tarkistamalla kukin myötäsuunnassa selattu Y-käsky tätä tallennettua poikkeusluetteloa vastaan.

Peräkkäiset Y-käskyt testataan myötäsuuntaisessa selauksessa ja jos millään selatuista Y-käskyistä ei ole poikkeusta, jäljellä olevat Y-käskyt suoritetaan ennenkuin lohkon 108 (kuvio 5A) suorittama asynkronisen tapahtuman käsittely sallitaan ilman että rikotaan X-käskyjaotusta kuten edellä on selitetty.

Jos toisaalta myötäsuunnassa selatulla Y-käskyllä ilmenee poikkeus lohkon 114 suorittamassa testissä, toimintolohko 118 välittömästi varmentaa Y-ohjelmalaskurin osoittamaan seuraavaan varmennettuun Y-käskyyn, joka on X-käskyraja, ja lohkon 108 (kuvio 5A) suorittama asynkronisen tapahtuman käsittely sallitaan jälleen ilman että

rikotaan X-käskyjaotusta. Tällä tavoin vältetään vieläpä mahdollisuus X-käskyjaotuksen rikkomiseen.

Kokonaisuutena katsoen esillä oleva keksintö tarjoaa tehokkaan mekanismin "yhdestä moneksi" -sovelluskoodin käännösten suorittamiseksi. Kehitetty koodi vastaa tarkasti alkuperäistä koodia suoritustulosten sekä tila-atomisuuden ja käskyjaotuksen suhteen. Atomisuus ja jaokkeisuus on taattu yksinkertaisilla yhden kirjoittamisen käskyillä sekä erityiskäskyillä mukaanlukien usean kirjoittamisen ja luku-muutos-kirjoitus-tyyppiä olevat. Näin ollen investoinnit alkuperäiseen CISC- tai muuhun koodiin voidaan säästää ja samalla voidaan saada hinta/suorituskyky -hyötyjä käytettäessä sovelluskoodin käännöksiä RISC- tai muissa paremman hinta/suorituskyky-suhteen omaavissa tietokonejärjestelmissä, joilla on suhteellisesti supistettu käskykanta.

Kysymyksessä olevan tekniikan asiantuntijat voivat tehdä erilaisia muunnoksia ja muunnelmia esillä olevan keksinnön mukaiseen parannettuun järjestelmään ja menetelmään käskyjen tila-atomisuuden säilyttämiseksi käännetyllä ohjelmakoodilla poikkeamatta tämän keksinnön piiristä ja hengestä. Tarkoituksena siis on, että esillä oleva keksintö käsittää tällaiset muunnokset ja muunnelmat siinä laajuudessa kuin ne ovat oheisten patenttivaatimusten ja niiden ekvivalenttien suojapiirissä.

Patenttivaatimukset:

1. Menetelmä ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi ja toisen ohjelmakoodin suorittamiseksi samalla kun säilytetään ensimmäisen koodin käskyjen tila-atomisuus, missä ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti- ja rekisteritilan sekä toiselle käskykannalle, joka on supistettu suhteessa ensimmäiseen käskykantaan, sovitetun toisen arkkitehtuurin omaavassa tietokoneessa suoritettava, t u n n e t t u siitä, että menetelmä käsittää seuraavat vaiheet:

käytetään ensimmäistä tietokonetta ensimmäisen koodin käskyjen kääntämiseksi vastaaviksi toisen koodin käskyiksi;

organisoidaan toisen koodin käskyt kutakin ensimmäisen koodin käskyä kohti jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä vähintään kaksi ryhmää, missä ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilan päivityksen ja jotka voidaan keskeyttää suorituksen jälkeen ilman tilavirheen vaaraa, ja toisessa ryhmässä ovat kaikki muisti- ja rekisteritilan päivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävän ensimmäisen koodin käskyn toteuttamiseksi;

sisällytetään mainittuun toiseen käskyryhmään ennalta määrättyjä yksittäisiä kirjoituskäskyjä varten ensimmäinen erityinen kirjoituskäsky, jolla on ensimmäinen alisekvenssi yhden ensimmäiseen muistipaikkaan kirjoittamisen käsittelemiseksi sen vaatimuksen mukaan, että mainittu ensimmäinen alisekvenssi täytyy suorittaa ilman mitään keskeytystä ja ilman minkään muun mainittuun muistitilaan ehkä liitetyn suorittimen väliin tulevia ristirii-

dassa olevia kirjoittamisia mainittuun ensimmäiseen muistipaikkaan;

sisällytetään mainittuun toiseen käskyryhmään moninkertaisia kirjoituskäskyjä varten toinen erityinen kirjoituskäsky toisen alisekvenssin sisällyttämiseksi moninkertaisten kirjoittamisten käsittelemistä varten, jotka täytyy suorittaa ilman mitään keskeytystä ja ilman minkään muun mainittuun muistitilaan ehkä liitetyn suorittimen ristiriidassa olevia kirjoittamisia;

käytetään mainittua toista tietokonejärjestelmää toisen ohjelmakoodin suorittamiseksi;

määritetään kunkin asynkronisen tapahtuman esiintyminen toisen koodin suorituksen aikana;

määritetään toisen koodin suorituksen aikana mainitun toisen suorittimen, jos se on kytketty mainittuun muistitilaan, kunkin ristiriidassa olevan kirjoittamisen mainittuun ensimmäiseen muistipaikkaan esiintyminen;

keskeytetään uudelleenyrittämistä varten mikä tahansa jaokkeinen toisen koodin käskysekvenssi ensimmäisen koodin käskyn tila-atomisuuden ja ensimmäisen koodin käskyjaotuksen säilyttämiseksi, jos asynkroninen tapahtumakeskeytys esiintyy tämän sekvenssin suorituksen aikana, ennenkuin kaikki ensimmäisen ryhmän käskyt on suoritettu tai, jos ensimmäisen ryhmän käskyt on suoritettu, ennen minkä tahansa sellaisen toisen ryhmän käskyn suoritusta, joka on alttiina mahdolliselle poikkeukselle, ja tällä tavoin mahdollistetaan seuraava asynkronisen tapahtuman käsittely;

keskeytetään ja yritetään uudelleen, kunnes suoritus onnistuu, minkä tahansa jaokkeisen toisen koodin käskysekvenssin, joka sisältää mainitun ensimmäisen alisekvenssin, mainittua ensimmäistä erityistä käskyjen alisekvenssiä, jos mainittu toinen suoritin suorittaa ristiriidassa olevan kirjoituksen ennen mainitun ensimmäisen alisekvenssin suorituksen päättymistä;

keskeytetään uudelleenyritystä varten minkä tahansa jaokkeisen toisen koodin käskysekvenssi, joka sisältää mainitun ensimmäisen alisekvenssin, jos asynkroninen tapahtumakeskeytys esiintyy mainitun ensimmäisen alisekvenssin suoritusyrityksen aikana; sekä

viivästetään asynkronisen tapahtumakeskeytyksen käsittelyä ja suoritetaan loppuun mikä tahansa suoritettavana oleva toisen koodin käskysekvenssi A) jos mainittu toinen alisekvenssi sisältyy jaokkeiseen käskysekvenssiin ja jos tämä asynkroninen tapahtumakeskeytys esiintyy aikaisintaan ensimmäisen kirjoittamisen jälkeen mainitun toisen käskyn alisekvenssin suorituksen aikana tai B) jos tämä asynkroninen tapahtumakeskeytys esiintyy kaikkien mainitussa toisessa ryhmässä olevien, mahdolliselle keskeytykselle alttiiden tilapäivityskäskyjen suorituksen jälkeen.

2. Patenttivaatimuksen 1 mukainen menetelmä, t u n n e t t u siitä, että mainittu ensimmäinen alisekvenssi on luku-muutos-kirjoitus-alisekvenssi,

ja siitä, että mainittua ensimmäistä alisekvenssiä käytetään osittaisen kirjoituskäskyn tai lukitun päivityskäskyn toteuttamiseksi, joka mainittu ensimmäinen alisekvenssi sisältää latauslukitun ehdollisen tallennussekvenssin, joka lukee ja muuttaa syöttödatan, tallentaa tulosdatan ehdollisesti ja jättää mainitun ensimmäisen alisekvenssin suorittamatta, jos ristiriidassa oleva kirjoitus on tapahtunut sen suorituksen aikana, tai suorittaa loppuun mainitun ensimmäisen alisekvenssin, jos ristiriidassa olevaa kirjoitusta ei ole tapahtunut sen suorituksen aikana.

3. Patenttivaatimuksen 1 mukainen menetelmä, t u n n e t t u siitä, että mainittu toinen alisekvenssi sisältää yhden etuoikeutetun arkkitehtuurin kirjaston (PAL, privileged architecture library) kutsurutiinin, joka suorittaa mainitun toisen alisekvenssin ja kaikki siihen si-

sältyvät kirjoittamiset, kun mainittu PAL-kutsurutiini on aloitettu, ja että mainitun PAL-kutsurutiinin aloittaminen on sallittu, jos aikaisemmin ei on esiintynyt keskeytystä kulloisenkin käskysekvenssin suorituksessa ja jos kaikki jäljellä olevat tilaosoitukset voidaan suorittaa loppuun ilman poikkeusta,

ja siitä, että mainitussa PAL-kutsurutiinissa on ensimmäinen alirutiini, joka testaa ensimmäistä suoritettavaa kirjoittamista sen ratkaisemiseksi, onko se osittainen kirjoitus, suorittaa latauslukitun ehdollisen tallennuksen mainitun ensimmäisen kirjoittamisen suorittamiseksi, jos se on osittainen kirjoittaminen, yrittää selektiivisesti uudelleen latauslukittua ehdollista kirjoittamista, kunnes se saadaan suoritetuksi loppuun, jos ehdollinen kirjoitus epäonnistuu mainitun toisen suorittimen ristiriidassa olevan tilakirjoituksen johdosta mainitun ensimmäisen alirutiinin suoritusyrityksen aikana, suorittaa loppuun ehdollisen kirjoittamisen kirjoitustoiminnon, jos ristiriidassa olevaa tilakirjoitusta ei ole esiintynyt ensimmäisen yrityksen aikana, jos mitään uudelleenyritystä ei ole valittu tai seuraavan uudelleenyrityksen aikana, jos uudelleenyritykset on valittu, ja että mainittu PAL-kutsurutiini lukitaan loppuunsuorittamista varten mainitun ensimmäisen alirutiinin tultua suoritetuksi loppuun, ja että toinen alirutiini seuraavaksi testaa toisen suoritettavan kirjoittamisen sen ratkaisemiseksi, onko se osittainen kirjoittaminen, suorittaa latauslukitun ehdollisen kirjoittamisen mainitun toisen kirjoittamisen suorittamiseksi, jos se on osittainen kirjoittaminen, yrittää uudelleen latauslukittua ehdollista kirjoittamista, kunnes se saadaan suoritetuksi, jos ehdollinen tallennus epäonnistuu mainitun toisen suorittimen suorittaman ristiriidassa olevan tilakirjoituksen johdosta mainitun toisen alirutiinin suoritusyrityksen aikana, ja että kunkin seuraavaksi suoritettavan kirjoittamisen käsittelee alirutiini, joka on

olennaisesti samanlainen kuin mainittu toinen alirutiini, kunnes kaikki kirjoittamiset on suoritettu mainitun PAL-kutsun loppuunsuorittamiseksi,

ja siitä, että ensimmäistä aliohjelmaa yritetään uudelleen, kunnes sen loppuunsuorittaminen onnistuu.

4. Menetelmä ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi ja toisen ohjelmakoodin suorittamiseksi samalla kun säilytetään ensimmäisen koodin käskyjen tila-atomisuus, missä ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti- ja rekisteritilan sekä toiselle käskykannalle, joka on ensimmäisen käskykannan suhteen supistettu, sovitetun tietokannan omaavassa tietokoneessa suoritettava, t u n n e t t u siitä, että menetelmän käsittää seuraavat vaiheet:

käytetään ensimmäistä tietokonetta ensimmäisen koodin käskyjen kääntämiseksi vastaaviksi toisen koodin käskyiksi rakennekoodin mukaan, joka määrittelee ensimmäisen koodin käskyt toisen koodin käskyjen avulla;

organisoidaan toisen koodin käskyt kutakin ensimmäisen koodin käskyä kohti jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä ainakin kaksi ryhmää, missä ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilapäivityksen ja jotka voidaan keskeyttää suorittamisen jälkeen ilman tilavirheen vaaraa, sekä toisen ryhmän, jossa on kaikki muisti- ja rekisteritilan päivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävän ensimmäisen koodin käskyn toteuttamiseksi;

strukturoidaan erityinen kirjoituskäsky sisältämään alisekvenssi yhden ensimmäiseen muistipaikkaan kirjoittamisen käsittelemiseksi sen vaatimuksen mukaan, että mainittu yksi kirjoittaminen täytyy suorittaa ilman keskeytystä ja ilman että mahdollinen mainittuun muistitilaan

ehkä kytketty toinen suoritin välillä suorittaa ristiriidassa olevia kirjoittamisia mainittuun ensimmäiseen muistipaikkaan;

käytetään toiseen arkkitehtuuriin sovitettua toista tietokonetta toisen ohjelmakoodin suorittamiseksi;

määritetään kunkin asynkronisen tapahtuman esiintyminen toisen koodin suorituksen aikana;

määritetään toisen koodin suorituksen aikana jokaisen mainittuun ensimmäiseen muistipaikkaan kohdistuvan ristiriidassa olevan kirjoittamisen esiintyminen, jonka mainittu toinen suoritin suorittaa, jos se on kytketty mainittuun muistitilaan;

keskeytetään uudelleenyritystä varten jaokkeinen toinen koodisekvenssi ensimmäisen koodin käskyn tila-atomisuuden ensimmäisen koodin käskyjaotuksen säilyttämiseksi, jos asynkroninen tapahtumakeskeytys tapahtuu sekvenssin suorituksen aikana ennenkuin kaikki ensimmäisen ryhmän käskyt on suoritettu tai, jos ensimmäisen ryhmän käskyt on suoritettu, ennen minkä tahansa toisen ryhmän käskyn suorittamista, joka an alttiina mahdolliselle poikkeukselle, jolla tavoin mahdollistetaan seuraavaksi suoritettava asynkronisen tapahtuman käsittely;

keskeytetään uudelleenyritystä varten, kunnes suoritus on onnistuneesti loppuunsuoritettu, mainittu erityinen käskysekvenssi missä tahansa jaokkeisessa toisen koodin käskysekvenssissä, joka sisältää mainitun alisekvenssin, jos mainittu toinen suoritin suorittaa ristiriidassa olevan kirjoittamisen ennen mainitun alisekvenssin suorituksen päättymistä;

keskeytetään uudelleenyritystä varten mikä tahansa jaokkeinen toisen koodin käskysekvenssi, joka sisältää mainitun alisekvenssin, jos asynkroninen tapahtumakeskeytys esiintyy mainitun alisekvenssin suoritusyrityksen aikana; ja

viivästetään asynkronisen tapahtuman käsittelyä ja suoritetaan loppuun mikä tahansa suoritettavana oleva jaokkeinen toisen koodin käskysekvenssi, jos asynkroninen tapahtumakeskeytys esiintyy kaikkien mainitussa toisessa ryhmässä olevien mahdolliselle keskeytykselle alttiiden tilapäivityskäskyjen jälkeen.

5. Patenttivaatimuksen 4 mukainen menetelmä, t u n n e t t u siitä, että ensimmäinen alisekvenssi on luku-muutos-kirjoitus-alisekvenssi.

6. Patenttivaatimuksen 5 mukainen menetelmä, t u n n e t t u siitä, että mainittua ensimmäistä alisekvenssiä käytetään toteuttamaan osittainen kirjoituskäsky tai lukittu päivityskäsky, että mainittu ensimmäinen alisekvenssi sisältää latauslukitun ehdollisen kirjoituskäskyn, joka lukee ja muuttaa syöttödatan, ehdollisesti tallentaa tulosdatan ja epäonnistuu mainitun ensimmäisen alisekvenssin suorittamisessa, jos sen suorituksen aikana on esiintynyt ristiriidassa oleva kirjoittaminen, tai suorittaa loppuun mainitun ensimmäisen alisekvenssin, jos sen suorittamisen aikana ei ole esiintynyt ristiriidassa olevaa kirjoittamista.

7. Menetelmä ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi ja toisen ohjelmakoodin suorittamiseksi samalla kun säilytetään ensimmäisen koodin käskyjen tila-atomisuus, missä ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti- ja rekisteritilan sekä toiselle käskykannalle, joka on ensimmäisen käskykannan suhteen supistettu, sovitetun toisen arkkitehtuurin omaavassa tietokoneessa suoritettava, t u n n e t t u siitä, että menetelmä käsittää:

ensimmäisen tietokoneen käyttämisen ensimmäisen koodin käskyjen kääntämiseksi vastaaviksi toisen koodin

käskyiksi rakennekoodin mukaan, joka määrittelee ensimmäisen koodin käskyt toisen koodin käskyjen avulla;

toisen koodin käskyjen organisoimisen kutakin ensimmäisen koodin käskyä kohti jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä ainakin kaksi ryhmää, missä ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilapäivityksen ja jotka voidaan keskeyttää suorituksen jälkeen ilman tilavirheen vaaraa, ja missä toisessa ryhmässä on kaikki muisti- ja rekisteritilan päivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävänä olevan ensimmäisen koodin käskyn toteuttamiseksi;

erityisen kirjoituskäskyn strukturoimisen sisältämään alisekvenssi useiden kirjoitusten käsittelemiseksi, jotka kaikki täytyy suorittaa ilman keskeytystä;

toiselle arkkitehtuurille sovitetun toisen tietokoneen käyttämisen toisen ohjelmakoodin suorittamiseksi;

kunkin asynkronisen tapahtuman esiintymisen määrittämisen toisen koodin suorituksen aikana;

minkä tahansa jaokkeisen toisen koodin käskysekvenssin keskeyttämisen uudelleenyritystä varten ensimmäisen koodin käskyjen tila-atomisuuden ja ensimmäisen koodin käskyjaotuksen säilyttämiseksi, jos asynkroninen tapahtumakeskeytys tapahtuu sekvenssin suorituksen aikana ennenkuin kaikki ensimmäisen ryhmän käskyt on suoritettu tai, jos ensimmäisen ryhmän käskyt on suoritettu, ennen minkä tahansa toisen ryhmän sellaisen käskyn suorittamista, joka on alttina mahdolliselle poikkeukselle, jolla tavoin tehdään mahdolliseksi seuraava asynkronisen tapahtuman käsittely;

asynkronisen tapahtumakeskeytyksen käsittelyn viivästämisen ja minkä tahansa suoritettavana olevan jaokkeisen toisen koodin käskysekvenssin loppuunsuorittamisen A) jos mainittu alisekvenssi sisältyy jaokkeiseen käskysekvenssiin ja jos asynkroninen tapahtumakeskeytys esiin-

tyy aikaisintaan ensimmäisen kirjoittamisen jälkeen mainitun käskysekvenssin suorituksen aikana tai B) jos asynkroninen tapahtumakeskeytys esiintyy kaikkien mainitussa toisessa ryhmässä olevien tilapäivityskäskyjen jälkeen, jotka ovat alttiita mahdolliselle poikkeukselle.

8. Patenttivaatimuksen 7 mukainen menetelmä, t u n n e t t u siitä, että mainittu toinen alisekvenssi sisältää yhden etuoikeutetun arkkitehtuurin kirjaston (PAL, privileged architecture library) kutsurutiinin, joka suorittaa mainitun toisen alisekvenssin ja kaikki siihen sisältyvät kirjoittamiset, kun mainittu PAL-kutsurutiini on aloitettu, ja että mainitun PAL-kutsurutiinin aloittaminen on sallittu, jos aikaisemmin ei on esiintynyt keskeytystä kulloisenkin käskysekvenssin suorituksessa ja jos kaikki jäljellä olevat tilaosoitukset voidaan suorittaa loppuun ilman poikkeusta,

ja siitä, että mainitussa PAL-kutsurutiinissa on ensimmäinen alirutiini, joka testaa ensimmäistä suoritettavaa kirjoittamista sen ratkaisemiseksi, onko se osittainen kirjoitus, suorittaa latauslukitun ehdollisen tallennuksen mainitun ensimmäisen kirjoittamisen suorittamiseksi, jos se on osittainen kirjoittaminen, yrittää selektiivisesti uudelleen latauslukittua ehdollista kirjoittamista, kunnes se saadaan suoritetuksi loppuun, jos ehdollinen kirjoitus epäonnistuu mainitun toisen suorittimen ristiriidassa olevan tilakirjoituksen johdosta mainitun ensimmäisen alirutiinin suoritusyrityksen aikana, suorittaa loppuun ehdollisen kirjoittamisen kirjoitustoiminnon, jos ristiriidassa olevaa tilakirjoitusta ei ole esiintynyt ensimmäisen yrityksen aikana, jos mitään uudelleenyritystä ei ole valittu tai seuraavan uudelleenyrityksen aikana, jos uudelleenyritykset on valittu, ja että mainittu PAL-kutsurutiini lukitaan loppuunsuorittamista varten mainitun ensimmäisen alirutiinin tultua suoritetuksi loppuun, ja että toinen alirutiini seuraavaksi testaa toisen suoritet-

tavan kirjoittamisen sen ratkaisemiseksi, onko se osittainen kirjoittaminen, suorittaa latauslukitun ehdollisen kirjoittamisen mainitun toisen kirjoittamisen suorittamiseksi, jos se on osittainen kirjoittaminen, yrittää uudelleen latauslukittua ehdollista kirjoittamista, kunnes se saadaan suoritetuksi, jos ehdollinen tallennus epäonnistuu mainitun toisen suorittimen suorittaman ristiriidassa olevan tilakirjoituksen johdosta mainitun toisen alirutiinin suoritusyrityksen aikana, ja että kunkin seuraavaksi suoritettavan kirjoittamisen käsittelee alirutiini, joka on olennaisesti samanlainen kuin mainittu toinen alirutiini, kunnes kaikki kirjoittamiset on suoritettu mainitun PAL-kutsun loppuunsuorittamiseksi.

9. Menetelmä ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi ensimmäisen koodin tila-atomisuuden säilyttämisen helpottamiseksi, kun toinen koodi suoritetaan, missä ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti- ja rekisteritilan sekä toiselle käskykannalle, joka on supistettu suhteessa ensimmäisen käskykantaan, sovitetun toisen arkkitehtuurin omaavassa tietokoneessa suoritettava, t u n n e t t u siitä, että menetelmä käsittää:

ensimmäisen tietokoneen käyttämisen ensimmäisen koodin käskyjen kääntämiseksi vastaaviksi toisen koodin käskyiksi rakennekoodin mukaan, joka määrittelee ensimmäisen koodin käskyt toisen koodin käskyjen avulla;

toisen koodin käskyjen organisoimisen kutakin ensimmäisen koodin käskyä kohti jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä ainakin kaksi ryhmää, missä ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilapäivityksen ja jotka voidaan keskeyttää suorittamisen jälkeen ilman tilavirheen vaaraa, sekä toisen ryhmän, jossa on kaikki muis-

ti- ja rekisteritilan päivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävän ensimmäisen koodin käskyn toteuttamiseksi;

erityisen kirjoituskäskyn strukturoimisen sisältämään alisekvenssi yhden ensimmäiseen muistipaikkaan kirjoittamisen käsittelemiseksi sen vaatimuksen mukaan, että mainittu yksi kirjoittaminen täytyy suorittaa ilman keskeytystä ja ilman että mahdollinen mainittuun muistitilaan ehkä kytketty toinen suoritin välillä suorittaa ristiriidassa olevia kirjoittamisia mainittuun ensimmäiseen muistitilaan;

toisen erityisen kirjoituskäskyn strukturoimisen sisältämään toisen alisekvenssi useiden kirjoitusten käsittelemiseksi, jotka kaikki täytyy suorittaa ilman keskeytystä;

mainitun ensimmäisen alisekvenssin strukturoimisen keskeytymään uudelleenyritystä varten, kunnes suoritus on onnistuneesti päättynyt, jos mainittu toinen suoritin suorittaa ristiriidassa olevan kirjoittamisen ennen mainitun ensimmäisen alisekvenssin loppuunsuorittamista;

mainitun ensimmäisen alisekvenssin struktruoimisen epäonnistumaan, jos asynkroninen tapahtumakeskeytys tapahtuu mainitun ensimmäisen alisekvenssin uudelleenyrityksen aikana, jolloin mahdollistetaan minkä tahansa jaokkeisen toisen koodin sellaisen käskyn uudelleenyrittäminen, joka sisältää mainitun ensimmäisen alisekvenssin; sekä

mainitun toisen alisekvenssin strukturoimisen etuoikeutettua, ei keskeytettävissä olevaa suoritusta varten, ja tällöin minkä tahansa asynkronisen tapahtumakeskeytyksen suorituksen viivästämisen toisen koodin suorituksen aikana siihen asti kunnes mainitun toisen alisekvenssin suoritus on päättynyt.

10. Menetelmä toisen ohjelmakoodin suorittamiseksi samalla kun säilytetään sen ensimmäisen ohjelmakoodin tila-atomisuus, josta toinen ohjelmakoodi on käännetty, mis-

sä ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti- ja rekisteritilan sekä toiselle käskykannalle, joka on supistettu suhteessa ensimmäiseen käskykantaan, sovitetun toisen arkkitehtuurin omaavassa tietokoneessa suoritettava, missä toisen koodin käskyt kutakin ensimmäisen koodin käskyä kohti on organisoitu jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä ainakin kaksi ryhmää, joista ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilapäivityksen ja jotka voidaan keskeyttää suorituksen jälkeen ilman tilavirheen vaaraa, ja joista toisessa ryhmässä on kaikki muisti- ja tilapäivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävän ensimmäisen koodin käskyn toteuttamiseksi, missä ensimmäinen erityinen kirjoituskäsky on strukturoitu sisältämään ensimmäinen alisekvenssi yhden ensimmäiseen muistipaikkaan kohdistuvan kirjoittamisen käsittelemiseksi sen vaatimuksen mukaan, että mainittu yksi kirjoittaminen täytyy suorittaa ilman keskeytystä ja ilman mainittuun muistitilaan ehkä liitetyn minkä tahansa toisen suorittimen välillä suorittamia mainittuun ensimmäiseen muistipaikkaan kohdistuvia ristiriidassa olevia kirjoittamisia, ja missä toinen erityinen kirjoituskäsky on strukturoitu sisältämään toinen alisekvenssi useiden kirjoittamisten käsittelemiseksi, jotka kaikki täytyy suorittaa ilman keskeytystä, t u n n e t t u siitä, että menetelmä käsittää seuraavat vaiheet:

käytetään toiselle arkkitehtuurille sovitettua toista tietokonetta toisen ohjelmakoodin suorittamiseksi;

määritetään kunkin asynkronisen tapahtuman esiintyminen toisen koodin suorituksen aikana;

määritetään toisen koodin suorituksen aikana mainitun toisen suorittimen kunkin mainittuun ensimmäiseen

muistipaikkaan kohdistuvan ristiriidassa olevan kirjoittamisen esiintyminen, jos mainittu toinen suoritin on liitetty mainittuun muistitilaan;

keskeytetään uudelleenyritystä varten mikä tahansa jaokkeinen toisen koodin käskysekvenssi ensimmäisen koodin käskyn tila-atomisuuuden ja ensimmäisen koodin käskyjaotuksen säilyttämiseksi, jos asynkroninen tapahtumakeskeytys esiintyy sekvenssin suorituksen aikana ennenkuin kaikki ensimmäisen ryhmän käskyt on suoritettu tai, jos ensimmäisen ryhmän käskyt on suoritettu, ennen minkä tahansa toisen ryhmän sellaisen käskyn suoritusta, joka on alttiina mahdolliselle poikkeukselle, jolla tavoin mahdollistetaan seuraava asynkronisen tapahtuman käsittely;

keskeytetään uudelleenyritystä varten, kunnes suoritus on onnistuneesti päättynyt, mainittu ensimmäinen erityinen käskysekvenssi missä tahansa sellaisessa jaokkeisessa toisessa käskysekvenssissä, joka sisältää mainitun ensimmäisen alisekvenssin, jos mainittu toinen suoritin on suorittanut ristiriidassa olevan kirjoittamisen ennen mainitun ensimmäisen alisekvenssin suorituksen päättymistä;

keskeytetään mikä tahansa sellainen jaokkeinen toisen koodin käskysekvenssi, joka sisältää mainitun ensimmäisen alisekvenssin, uudelleenyritystä varten, jos asynkroninen tapahtumakeskeytys esiintyy mainitun ensimmäisen alisekvenssin suoritusyrityksen aikana; sekä

viivästetään asynkronisen tapahtumakeskeytyksen käsittelyä ja viedään loppuun mikä tahansa suoritettavana oleva jaokkeinen toisen koodin käskysekvenssi A) jos mainittu alisekvenssi sisältyy jaokkeiseen käskysekvenssiin ja jos asynkroninen tapahtumakeskeytys esiintyy aikaisintaan ensimmäisen kirjoittamisen jälkeen mainitun käskysekvenssin suorituksen aikana tai B) jos asynkroninen tapahtumakeskeytys esiintyy kaikkien mainitussa toisessa ryh-

mässä olevien tilapäivityskäskyjen jälkeen, jotka ovat alttiita mahdolliselle poikkeukselle.

11. Järjestelmä ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi ja toisen ohjelmakoodin suorittamiseksi tavalla, joka säilyttää ensimmäisen koodin käskyjen tila-atomisuuden, missä ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti- ja rekisteritilan sekä toiselle käskykannalle, joka on supistettu suhteessa ensimmäiseen käskykantaan, sovitetun toisen arkkitehtuurin omaavassa tietokoneessa suoritettava, t u n n e t t u siitä, että järjestelmä käsittää:

ensimmäisen tietokonejärjestelmän, jossa on ensimmäinen suoritin ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi, ja ensimmäisen muistijärjestelmän, joka on liitetty mainittuun ensimmäiseen suorittimeen;

elimen kunkin ensimmäisessä koodissa peräkkäin olevan käskyn kääntämiseksi toisen koodin käskyiksi mainitun rakennekoodin mukaan tallennettavaksi toisena ohjelmakoodina mainittuun ensimmäiseen muistijärjestelmään;

elimen toisen koodin käskyjen organisoimiseksi kutakin ensimmäisen koodin käskyä kohti jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä ainakin kaksi ryhmää, joista ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilapäivityksen ja jotka voidaan keskeyttää suorituksen jälkeen ilman tilavirheen vaaraa, ja joista toisessa ryhmässä on kaikki ja muisti- ja rekisteritilan päivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävän ensimmäisen koodin käskyn toteuttamiseksi;

elimen ensimmäisen erityisen kirjoituskäskyn strukturoimiseksi sisältämään ensimmäinen alisekvenssi ensim-

mäiseen muistipaikkaan kohdistuvan yhden kirjoittamisen käsittelemiseksi sen vaatimuksen mukaan, että mainittu yksi kirjoittaminen täytyy suorittaa ilman keskeytyksiä ja ilman mainittuun muistitilaan ehkä liitetyn mahdollisen toisen suorittimen välillä suorittamia mainittuun ensimmäiseen muistipaikkaan kohdistuvia ristiriidassa olevia kirjoittamisia;

elimen toisen erityisen kirjoituskäskyn struktruroimiseksi sisältämään toinen alisekvenssi useiden kirjoittamisten käsittelemiseksi, jotka kaikki täytyy suorittaa ilman keskeytystä;

toisen tietokonejärjestelmän toisen koodin suorittamiseksi, joka on kehitetty mainitun ensimmäisen tietokonejärjestelmän tulosteena, jolla mainitulla toisella tietokonejärjestelmällä on mainittu toinen arkkitehtuuri ja toinen suoritin ja muisti- ja rekisteritila, joka sisältää mainittuun toiseen suorittimeen liitetyn toisen muistijärjestelmän;

elimen kunkin asynkronisen tapahtuman esiintymisen määrittämiseksi toisen koodin suorituksen aikana ja mainitun toisen suorittimen suorittaman mainittuun ensimmäiseen muistipaikkaan kohdistuvan ristiriidassa olevan kirjoittamisen esiintymisen määrittämiseksi, jos mainittu toinen suoritin on liitetty mainittuun muistitilaan;

elimen minkä tahansa jaokkeisen toisen koodin käskysekvenssin keskeyttämiseksi ensimmäisen koodin käskyjen tila-atomisuuden ja ensimmäisen koodin käskyjaotuksen säilyttämiseksi, jos asynkroninen tapahtumakeskeytys esiintyy sekvenssin suorituksen aikana ennenkuin kaikki ensimmäisen ryhmän käskyt on suoritettu tai, jos ensimmäisen ryhmän käskyt on suoritettu, ennen toisen ryhmän minkä tahansa sellaisen käskyn suoritusta, joka on alttiina mahdolliselle keskeytykselle, jolla tavoin mahdollistetaan seuraava asynkronisen tapahtuman käsittely;

elimen mainitun ensimmäisen erityisen käskysekvenssin keskeyttämiseksi uudelleenyritystä varten missä tahansa jaokkeisessa toisen koodin käskysekvenssissä, joka sisältää mainitun ensimmäisen alisekvenssin, jos mainittu toinen suoritin on suorittanut ristiriidassa olevan kirjoittamisen mainitun ensimmäisen alisekvenssin aikana;

elimen minkä tahansa sellaisen jaokkeisen toisen koodin käskysekvenssin keskeyttämiseksi uudelleenyritystä varten, joka sisältää mainitun ensimmäisen alisekvenssin, jos asynkroninen tapahtumakeskeytys esiintyy mainitun ensimmäisen alisekvenssin suoritusyrityksen aikana; sekä

elimen asynkronisen tapahtumakeskeytyksen käsittelyn viivästämiseksi ja minkä tahansa suoritettavana olevan jaokkeisen toisen koodin käskysekvenssin loppuunsuorittamiseksi A) jos mainittu alisekvenssi sisältyy jaokkeiseen käskysekvenssiin ja jos asynkroninen tapahtumakeskeytys esiintyy aikaisintaan ensimmäisen kirjoittamisen jälkeen mainitun käskysekvenssin suorituksen aikana tai B) jos asynkroninen tapahtumakeskeytys esiintyy kaikkien mainitussa toisessa ryhmässä olevien tilapäivityskäskyjen jälkeen, jotka ovat alttiita mahdolliselle poikkeukselle.

12. Patenttivaatimuksen 11 mukainen järjestelmä, t u n n e t t u siitä, että ensimmäinen alisekvenssi on luku-muutos-kirjoitus-alisekvenssi,

ja siitä, että mainittua ensimmäistä alisekvenssiä käytetään osittaisen kirjoituskäskyn tai lukitun päivityskäskyn toteuttamiseksi, missä mainittu ensimmäinen alisekvenssi sisältää latauslukitun ehdollisen kirjoitussekvenssin, joka lukee ja muuttaa tulodatan, ehdollisesti tallentaa tulosdatan ja epäonnistuu mainitun alisekvenssin suorittamisessa, jos sen suorituksen aikana on esiintynyt ristiriidassa olevan kirjoittaminen, tai suorittaa loppuun mainitun ensimmäisen alisekvenssin, jos sen suorittamisen aikana ei ole esiintynyt ristiriidassa olevaa kirjoittamista,

ja siitä, että mainittu toinen alisekvenssi sisältää yhden etuoikeutetun arkkitehtuurin kirjaston (PAL, privileged architecture library) kutsurutiinin, joka suorittaa mainitun toisen alisekvenssin ja kaikki siihen sisältyvät kirjoittamiset, kun mainittu PAL-kutsurutiini on aloitettu, ja että mainitun PAL-kutsurutiinin aloittaminen on sallittu, jos aikaisemmin ei ole esiintynyt keskeytystä kulloisenkin käskysekvenssin suorituksessa ja jos kaikki jäljellä olevat tilaosoitukset voidaan suorittaa loppuun ilman poikkeusta.

13. Patenttivaatimuksen 12 mukainen järjestelmä, t u n n e t t u siitä, että mainitussa PAL-kutsurutiinissa on ensimmäinen alirutiini, joka testaa ensimmäistä suoritettavaa kirjoittamista sen ratkaisemiseksi, onko se osittainen kirjoitus, suorittaa latauslukitun ehdollisen tallennuksen mainitun ensimmäisen kirjoittamisen suorittamiseksi, jos se on osittainen kirjoittaminen, yrittää selektiivisesti uudelleen latauslukittua ehdollista kirjoittamista, kunnes se saadaan suoritetuksi loppuun, jos ehdollinen kirjoitus epäonnistuu mainitun toisen suorittimen ristiriidassa olevan tilakirjoituksen johdosta mainitun ensimmäisen alirutiinin suoritusyrityksen aikana, suorittaa loppuun ehdollisen kirjoittamisen kirjoitustoiminnon, jos ristiriidassa olevaa tilakirjoitusta ei ole esiintynyt ensimmäisen yrityksen aikana, jos mitään uudelleenyritystä ei ole valittu tai seuraavan uudelleenyrityksen aikana, jos uudelleenyritykset on valittu, ja että mainittu PAL-kutsurutiini lukitaan loppuunsuorittamista varten mainitun ensimmäisen alirutiinin tultua suoritetuksi loppuun, ja että toinen alirutiini seuraavaksi testaa toisen suoritettavan kirjoittamisen sen ratkaisemiseksi, onko se osittainen kirjoittaminen, suorittaa latauslukitun ehdollisen kirjoittamisen mainitun toisen kirjoittamisen suorittamiseksi, jos se on osittainen kirjoittaminen, yrittää uudelleen latauslukittua ehdollista kirjoittamista, kunnes se

saadaan suoritetuksi, jos ehdollinen tallennus epäonnistuu mainitun toisen suorittimen suorittaman ristiriidassa olevan tilakirjoituksen johdosta mainitun toisen alirutiinin suoritusyrityksen aikana, ja että kunkin seuraavaksi suoritettavan kirjoittamisen käsittelee alirutiini, joka on olennaisesti samanlainen kuin mainittu toinen alirutiini, kunnes kaikki kirjoittamiset on suoritettu mainitun PAL-kutsun loppuunsuorittamiseksi.

14. Järjestelmä ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi ja toisen ohjelmakoodin suorittamiseksi tavalla, joka säilyttää ensimmäisen koodin käskyjen tila-atomisuuden, missä ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti- ja rekisteritilan sekä toiselle käskykannalle, joka on supistettu suhteessa ensimmäiseen käskykantaan, sovitetun toisen arkkitehtuurin omaavassa tietokoneessa suoritettava, t u n n e t t u siitä, että järjestelmä käsittää:

ensimmäisen tietokonejärjestelmän, jossa on ensimmäinen suoritin ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi, ja ensimmäisen muistijärjestelmän, joka on liitetty mainittuun ensimmäiseen suorittimeen;

elimen kunkin ensimmäisessä koodissa peräkkäin olevan käskyn kääntämiseksi toisen koodin käskyiksi mainitun rakennekoodin mukaan tallennettavaksi toisena ohjelmakoodina mainittuun ensimmäiseen muistijärjestelmään;

elimen toisen koodin käskyjen organisoimiseksi kutakin ensimmäisen koodin käskyä kohti jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä ainakin kaksi ryhmää, joista ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilapäivityksen ja jotka voidaan keskeyttää suorituksen jälkeen ilman tilavirheen vaaraa, ja joista toisessa ryhmässä

on kaikki ja muisti- ja rekisteritilan päivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävän ensimmäisen koodin käskyn toteuttamiseksi;

elimen erityisen kirjoituskäskyn strukturoimiseksi sisältämään ensimmäinen alisekvenssi ensimmäiseen muistipaikkaan kohdistuvan yhden kirjoittamisen käsittelemiseksi sen vaatimuksen mukaan, että mainittu yksi kirjoittaminen täytyy suorittaa ilman keskeytyksiä ja ilman mainittuun muistitilaan ehkä liitetyn mahdollisen toisen suorittimen välillä suorittamia mainittuun ensimmäiseen muistipaikkaan kohdistuvia ristiriidassa olevia kirjoittamisia;

toisen tietokonejärjestelmän toisen koodin suorittamiseksi, joka on kehitetty mainitun ensimmäisen tietokonejärjestelmän tulosteena, jolla mainitulla toisella tietokonejärjestelmällä on mainittu toinen arkkitehtuuri ja toinen suoritin ja muisti- ja rekisteritila, joka sisältää mainittuun toiseen suorittimeen liitetyn toisen muistijärjestelmän;

elimen kunkin asynkronisen tapahtuman esiintymisen määrittämiseksi toisen koodin suorituksen aikana ja mainitun toisen suorittimen suorittaman mainittuun ensimmäiseen muistipaikkaan kohdistuvan ristiriidassa olevan kirjoittamisen esiintymisen määrittämiseksi, jos mainittu toinen suoritin on liitetty mainittuun muistitilaan;

elimen minkä tahansa jaokkeisen toisen koodin käskysekvenssin keskeyttämiseksi ensimmäisen koodin käskyjen tila-atomisuuden ja ensimmäisen koodin käskyjaotuksen säilyttämiseksi, jos asynkroninen tapahtumakeskeytys esiintyy sekvenssin suorituksen aikana ennenkuin kaikki ensimmäisen ryhmän käskyt on suoritettu tai, jos ensimmäisen ryhmän käskyt on suoritettu, ennen toisen ryhmän minkä tahansa sellaisen käskyn suoritusta, joka on alttiina mahdolliselle keskeytykselle, jolla tavoin mahdollistetaan seuraava asynkronisen tapahtuman käsittely;

elimen mainitun ensimmäisen erityisen käskysekvenssin keskeyttämiseksi uudelleenyritystä varten missä tahansa jaokkeisessa toisen koodin käskysekvenssissä, joka sisältää mainitun ensimmäisen alisekvenssin, jos mainittu toinen suoritin on suorittanut ristiriidassa olevan kirjoittamisen mainitun ensimmäisen alisekvenssin aikana;

elimen minkä tahansa sellaisen jaokkeisen toisen koodin käskysekvenssin keskeyttämiseksi uudelleenyritystä varten, joka sisältää mainitun ensimmäisen alisekvenssin, jos asynkroninen tapahtumakeskeytys esiintyy mainitun ensimmäisen alisekvenssin suoritusyrityksen aikana; sekä

elimen, joka sisältää mainitun toisen suorittimen ja mainitun toisen muistijärjestelmän, asynkronisen tapahtumakeskeytyksen käsittelyn viivästämiseksi ja minkä tahansa suoritettavana olevan jaokkeisen toisen koodin käskysekvenssin loppuunsuorittamiseksi, jos asynkroninen tapahtuma esiintyy kaikkien sellaisten mainitussa toisessa ryhmässä olevien tilapäivityskäskyjen jälkeen, jotka ovat alttiina mahdolliselle poikkeukselle.

15. Patenttivaatimuksen 22 mukainen järjestelmä, t u n n e t t u siitä, että ensimmäinen alisekvenssi on luku-muutos-kirjoitus-alisekvenssi,

ja siitä että mainittua ensimmäistä alisekvenssiä käytetään osittaisen kirjoituskäskyn tai lukitun päivityskäskyn toteuttamiseksi, missä mainittu ensimmäinen alisekvenssi sisältää latauslukitun ehdollisen tallennuskäskyn, joka lukee ja muuttaa tulodatan, ehdollisesti tallentaa tulosdatan ja epäonnistuu mainitussa ensimmäisessä alisekvenssissä, jos sen suorittamisen aikana on esiintynyt ristiriidassa oleva kirjoittaminen, tai suorittaa mainitun ensimmäisen alisekvenssin loppuun, jos sen suorituksen aikana ei ole tapahtunut ristiriidassa olevaa kirjoittamista.

16. Järjestelmä ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi ja toisen ohjelmakoodin

suorittamiseksi tavalla, joka säilyttää ensimmäisen koodin käskyjen tila-atomisuuden, missä ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti- ja rekisteritilan sekä toiselle käskykannalle, joka on supistettu suhteessa ensimmäiseen käskykantaan, sovitetun toisen arkkitehtuurin omaavassa tietokoneessa suoritettava, t u n n e t t u siitä, että mainittu järjestelmä käsittää:

ensimmäisen tietokonejärjestelmän, jossa on ensimmäinen suoritin ensimmäisen ohjelmakoodin kääntämiseksi toiseksi ohjelmakoodiksi, ja ensimmäisen muistijärjestelmän, joka on liitetty mainittuun ensimmäiseen suorittimeen;

elimen kunkin ensimmäisessä koodissa peräkkäin olevan käskyn kääntämiseksi toisen koodin käskyiksi mainitun rakennekoodin mukaan tallennettavaksi toisena ohjelmakoodina mainittuun ensimmäiseen muistijärjestelmään;

elimen toisen koodin käskyjen organisoimiseksi kutakin ensimmäisen koodin käskyä kohti jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä ainakin kaksi ryhmää, joista ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilapäivityksen ja jotka voidaan keskeyttää suorituksen jälkeen ilman tilavirheen vaaraa, ja joista toisessa ryhmässä on kaikki ja muisti- ja rekisteritilan päivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävän ensimmäisen koodin käskyn toteuttamiseksi;

elimen erityisen kirjoituskäskyn strukturoimiseksi sisältämään alisekvenssi useiden kirjoituskäskyjen käsittelemiseksi, jotka kaikki täytyy suorittaa ilman keskeytystä;

toisen tietokonejärjestelmän toisen koodin suorittamiseksi, joka on kehitetty mainitun ensimmäisen tieto-

konejärjestelmän tulosteena, jolla mainitulla toisella tietokonejärjestelmällä on mainittu toinen arkkitehtuuri ja toinen suoritin ja muisti- ja rekisteritila, joka sisältää mainittuun toiseen suorittimeen liitetyn toisen muistijärjestelmän;

elimen kunkin asynkronisen tapahtuman esiintymisen määrittämiseksi toisen koodin suorituksen aikana;

elimen minkä tahansa jaokkeisen toisen koodin käskysekvenssin keskeyttämiseksi ensimmäisen koodin käskyjen tila-atomisuuden ja ensimmäisen koodin käskyjaotuksen säilyttämiseksi, jos asynkroninen tapahtumakeskeytys esiintyy sekvenssin suorituksen aikana ennenkuin kaikki ensimmäisen ryhmän käskyt on suoritettu tai, jos ensimmäisen ryhmän käskyt on suoritettu, ennen toisen ryhmän minkä tahansa sellaisen käskyn suoritusta, joka on alttiina mahdolliselle keskeytykselle, jolla tavoin mahdollistetaan seuraava asynkronisen tapahtuman käsittely;

elimen, joka sisältää mainitun toisen suorittimen ja mainitun toisen muistijärjestelmän, asynkronisen tapahtumakeskeytyksen käsittelyn viivästämiseksi ja minkä tahansa suoritettavana olevan jaokkeisen toisen koodin käskysekvenssin loppuunsuorittamiseksi A) jos mainittu alisekvenssi sisältyy jaokkeiseen käskysekvenssiin ja jos asynkroninen tapahtumakeskeytys esiintyy aikaisintaan ensimmäisen kirjoittamisen jälkeen mainitun käskysekvenssin suorituksen aikana tai B) jos asynkroninen tapahtumakeskeytys esiintyy kaikkien mainitussa toisessa ryhmässä olevien tilapäivityskäskyjen jälkeen, jotka ovat alttiita mahdolliselle poikkeukselle.

17. Järjestelmä toisen ohjelmakoodin suorittamiseksi samalla kun säilytetään sen ensimmäisen ohjelmakoodin tila-atomisuus, josta toinen ohjelmakoodi on käännetty, missä ensimmäinen ohjelmakoodi on ensimmäiselle käskykannalle sovitetun ensimmäisen arkkitehtuurin omaavassa tietokoneessa suoritettava ja toinen ohjelmakoodi on muisti-

ja rekisteritilan sekä toiselle käskykannalle, joka on supistettu suhteessa ensimmäiseen käskykantaan, sovitetun toisen arkkitehtuurin omaavassa tietokoneessa suoritettava, missä toisen koodin käskyt kutakin ensimmäisen koodin käskyä kohti on organisoitu jaokkeiseksi käskysekvenssiksi, jossa on järjestyksessä ainakin kaksi ryhmää, joista ensimmäinen ryhmä sisältää ne toisen koodin käskyt, jotka suorittavat muun käskytoiminnon kuin tilapäivityksen ja jotka voidaan keskeyttää suorituksen jälkeen ilman tilavirheen vaaraa, ja joista toisessa ryhmässä on kaikki muisti- ja tilapäivityskäskyt mukaanlukien mahdollinen erityinen kirjoituskäsky, joka tarvitaan käännettävän ensimmäisen koodin käskyn toteuttamiseksi, missä ensimmäinen erityinen kirjoituskäsky on strukturoitu sisältämään ensimmäinen alisekvenssi yhden ensimmäiseen muistipaikkaan kohdistuvan kirjoittamisen käsittelemiseksi sen vaatimuksen mukaan, että mainittu yksi kirjoittaminen täytyy suorittaa ilman keskeytystä ja ilman mainittuun muistitilaan ehkä liitetyn minkä tahansa toisen suorittimen välillä suorittamia mainittuun ensimmäiseen muistipaikkaan kohdistuvia ristiriidassa olevia kirjoittamisia, ja missä toinen erityinen kirjoituskäsky on strukturoitu sisältämään toinen alisekvenssi useiden kirjoittamisten käsittelemiseksi, jotka kaikki täytyy suorittaa ilman keskeytystä, t u n n e t t u siitä, että järjestelmä käsittää:

elimen toiselle arkkitehtuurille sovitetun toisen tietokoneen käyttämiseksi toisen ohjelmakoodin suorittamiseen;

elimen, joka määrittää kunkin asynkronisen tapahtuman esiintyminen toisen koodin suorituksen aikana;

elimen, joka määrittää toisen koodin suorituksen aikana mainitun toisen suorittimen kunkin mainittuun ensimmäiseen muistipaikkaan kohdistuvan ristiriidassa olevan kirjoittamisen esiintymisen, jos mainittu toinen suoritin on liitetty mainittuun muistitilaan;

elimen, joka keskeyttää uudelleenyritystä varten minkä tahansa jaokkeisen toisen koodin käskysekvenssin ensimmäisen koodin käskyn tila-atomisuuden ja ensimmäisen koodin käskyjaotuksen säilyttämiseksi, jos asynkroninen tapahtumakeskeytys esiintyy sekvenssin suorituksen aikana ennenkuin kaikki ensimmäisen ryhmän käskyt on suoritettu tai, jos ensimmäisen ryhmän käskyt on suoritettu, ennen minkä tahansa toisen ryhmän sellaisen käskyn suoritusta, joka on alttiina mahdolliselle poikkeukselle, ja mahdollistaa tällä tavoin seuraavan asynkronisen tapahtuman käsittelyn;

elimen, joka keskeyttää uudelleenyritystä varten, kunnes suoritus on onnistuneesti päättynyt, mainitun ensimmäisen erityisen käskysekvenssin missä tahansa sellaisessa jaokkeisessa toisessa käskysekvenssissä, joka sisältää mainitun ensimmäisen alisekvenssin, jos mainittu toinen suoritin on suorittanut ristiriidassa olevan kirjoittamisen ennen mainitun ensimmäisen alisekvenssin suorituksen päättymistä;

elimen, joka keskeyttää minkä tahansa sellaisen jaokkeisen toisen koodin käskysekvenssin, joka sisältää mainitun ensimmäisen alisekvenssin, uudelleenyritystä varten, jos asynkroninen tapahtumakeskeytys esiintyy mainitun ensimmäisen alisekvenssin suoritusyrityksen aikana; sekä

elimen, joka viivästää asynkronisen tapahtumakeskeytyksen käsittelyä ja suorittaa loppuun minkä tahansa suoritettavana olevan jaokkeisen toisen koodin käskysekvenssin A) jos mainittu alisekvenssi sisältyy jaokkeiseen käskysekvenssiin ja jos asynkroninen tapahtumakeskeytys esiintyy aikaisintaan ensimmäisen kirjoittamisen jälkeen mainitun käskysekvenssin suorituksen aikana tai B) jos asynkroninen tapahtumakeskeytys esiintyy kaikkien mainitussa toisessa ryhmässä olevien tilapäivityskäskyjen jälkeen, jotka ovat alttiita mahdolliselle poikkeukselle.

Patentkrav:

1. Förfarande för att konvertera en första programkod till en andra programkod och för att utföra den andra programkoden medan man bibehåller status-atomicitet för den första kodens instruktioner, där den första programkoden är utförbar på en dator som har en första arkitektur adapterat till en första instruktionsbas och den andra programkoden är utförbar på en andra dator som har minnes- och registerstatus och en andra arkitektur adapterad till en andra instruktionsbas som är reducerad i relation till den första instruktionsbasen, k ä n n e t e c k n a d därav, att förfarandet omfattar följande skeden:

man använder en första dator för att konvertera den första kodens instruktioner till motsvarande andra kodens instruktioner;

man organiserar den andra kodens instruktioner för varje första kods instruktion till en fraktionerad instruktionssekvens som har i följd minst två grupper, en första grupp som inkluderar de andra kodens instruktioner som utför andra instruktioner än statusuppdatering och som kan avbrytas efter utförning utan risk för statusfel, och en andra grupp som har alla minnes- och registerstatusuppdateringsinstruktioner inklusive speciella skrivinstruktioner som krävs för att implementera den konverterade första kodens instruktion;

man inkluderar till nämnda andra instruktionsgrupp, för förutbestämda enskilda skrivinstruktioner en första speciell skrivinstruktion, som har en första undersekvens för att behandla en enskild skrivning till en första minnesplats i enlighet med ett krav om att nämnda första undersekvens bör utföras utan något avbrott och utan några i mellankommande motstridiga skrivningar till nämnda första minnesplats av någon annan prosessor som kan vara kopplad till nämnda minnesplats;

man inkluderar till nämnda andra instruktionsgrupp för multipla skrivinstruktioner en andra speciell skrivinstruktion för att inkludera en andra undersekvens för behandling av multipla skrivningar som bör utföras utan något avbrott och utan några i mellan kommande motstridiga skrivningar av någon annan prosessor som kan vara kopplad till nämnda minnesstatus;

man använder nämnda andra datorsystem för att utföra den andra programkoden;

man avgör förekomsten av varje asynkronisk händelse under utförningen av den andra koden;

man avgör under utförningen av den andra koden förekomsten av varje motstridiga skrivning till nämnda första minnesställe av nämnda andra prosessor ifall den är kopplad till nämnda minnesstatus;

man avbryter för ett nytt försök vilken som helst fraktionerad andra kods instruktionssekvens för att bevara första kodens instruktions status-atomicitet och första kods instruktionsfraktionering ifall ett asynkroniskt händelseavbrott sker under sekvensens utförning före alla av den första gruppens instruktioner har utförts eller, ifall den första gruppens instruktioner har utförts, före utförandet av vilken som helst andra grupps instruktion som är föremål för ett möjligt undantag, härmed möjliggörandes följande asynkroniska händelses behandling;

man avbryter och försöker på nytt tills utförningen lyckats, vilken som helst fraktionerad andra kods instruktionssekvens, som innehåller nämnda första undersekvens, nämnda första särskilda instruktionssekvens, ifall nämnda andra prosessor utför en motstridig skrivning före avslutandet av nämnda första undersekvens utförning;

man avbryter för ett nytt försök vilken som helst fraktionerad andra kods instruktionssekvens, som innehåller nämnda första undersekvens, ifall ett asynkroniskt

händelseavbrott förekommer under ett utförningsförsök av nämnda första undersekvens; samt

man fördröjer behandlingen av det asynkroniska händelseavbrottet och slutför vilken som helst utförbar andra kods instruktionssekvens A) ifall nämnda andra undersekvens ingår i den fraktionerade instruktionssekvensen och ifall denna asynkroniska händelsesekvens förekommer tidigast efter den första skrivningen under utförningen av nämnda andra instruktions undersekvens eller B) ifall denna asynkroniska händelsesekvens förekommer efter utförningen av alla i nämnda andra grupp varande, för avbrott känsliga statusuppdateringsinstruktioner.

2. Förfarande enligt patentkravet 1, k ä n n e t e c k n a d därav, att nämnda första undersekvens är en läs-modifiera-skriv-undersekvens,

och av, att nämnda första undersekvens används för utförandet av en partiell skrivinstruktion eller en låst uppdateringsinstruktion, vilken nämnda första undersekvens innehåller en belastningslåst villkorlig lagringssekvens som läser och modifierar inmatningsdata, lagrar resultatdata villkorligt och låter bli att utföra nämnda första undersekvens ifall en motstridig skrivning har skett under dess utförning, eller slutför nämnda första undersekvens ifall en motstridig skrivning ej har skett under dess utförning.

3. Förfarande enligt patentkravet 1, k ä n n e t e c k n a d därav, att nämnda andra undersekvens innehåller en priviligierad arkitektur bibliotek (PAL, privileged architecture library) anropsrutin, som utför nämnda andra undersekvens och alla därtill hörande skrivningar, då nämnda PAL-anropsrutin har påbörjats, och att påbörjandet av nämnda PAL-anropsrutin är tillåtet, ifall det tidigare ej har förekommit avbrott i utförandet av ifrågavarande instruktionssekvens och ifall alla kvarvarande statusaccesser kan slutföras utan undantag,

och av, att i nämnda PAL-anropsrutin finns en första underrutin som testar den första utförda skrivningen för att avgöra, ifall den är en partiell skrivning, utför en belastningslåst villkorlig lagring för att utföra nämnda första skrivning, ifall den är en partiell skrivning, försöker selektivt på nytt utföra det belastningslåsta villkorliga skrivandet tills det kan slutföras, ifall den vilkorliga skrivningen misslyckas på grund av nämnda andra prosessors motstridiga statusskrivning under utförningsförsök av nämnda första underrutin, slutför den villkorliga skrivningens skrivningsfunktion, ifall en motstridig statusskrivning ej har förekommit under det första försöket, ifall inget nytt försök har valts eller under följande nya försök, ifall nya försök har valts, och att nämnda PAL-anropsrutin låses för slutförning efter att nämnda första underrutin har blivit slutförd, och att den andra underrutinen sedan testar en andra utförbar skrivning för att avgöra om den är en partiell skrivning, utför en belastningslåst villkorlig skrivning för att utföra nämnda andra skrivning, ifall den är en partiell skrivning, försöker på nytt med det belastningslåsta villkorliga skrivandet tills det kan slutföras, ifall den villkorliga lagringen misslyckas på grund av nämnda andra prosessor utförda motstridiga statusskrivning under utförningsförsök av nämnda andra underrutin, och att varje följande utförda skrivning behandlas av en underrutin, som är väsentligt likadan som nämnda andra underrutin, ända tills alla skrivningar har utförts för att slutföra nämnda PAL-anrop,

och av, att det första underprogrammet utförs på nytt tills dess slutförning lyckas.

4. Förfarande för att konvertera en första programkod till en andra programkod och för att utföra den andra programkoden medan man bibehåller status-atomicitet för den första kodens instruktioner, där den första programkoden är utförbar på en dator som har en första arki-

tektur adapterat till en första instruktionsbas och den andra programkoden är utförbar på en andra dator som har minnes- och registerstatus och en andra arkitektur adapterad till en andra instruktionsbas som är reducerad i relation till den första instruktionsbasen, k ä n n e t e c k n a d därav, att förfarandet omfattar följande skeden:

man använder en första dator för att konvertera den första kodens instruktioner till motsvarande andra kodens instruktioner;

man organiserar den andra kodens instruktioner för varje första kods instruktion till en fraktionerad instruktionssekvens som har i följd åtminstone två grupper, en första grupp som inkluderar de andra kodens instruktioner som utför andra instruktioner än statusuppdatering och som kan avbrytas efter utförning utan risk för statusfel, och en andra grupp som har alla minnes- och registerstatus uppdateringsinstruktioner inklusive speciella skrivinstruktioner som krävs för att implementera den konverterade första kodens instruktion;

man strukturerar en särskild skrivinstruktion för att innehålla en undersekvens för att behandla en skrivning till ett första minnesställe i enlighet med kravet att nämnda ena skrivning bör utföras utan något avbrott och utan några i mellan kommande motstridiga skrivningar till nämnda första minnesställe av någon annan prosessor som kan vara kopplad till nämnda minnesställe;

man använder en andra dator som är anpassad till den andra arkitekturen för att utföra den andra programkoden;

man avgör förekomsten av varje asynkronisk händelse under utförningen av den andra koden;

man avgör under utförningen av den andra koden förekomsten av varje motstridiga skrivning till nämnda första minnesställe av nämnda andra prosessor ifall den är kopplad till nämnda minnesstatus;

man avbryter för ett nytt försök vilken som helst fraktionerad andra kods instruktionssekvens för att bevara första kodens instruktions status-atomicitet och första kods instruktionsfraktionering ifall ett asynkroniskt händelseavbrott sker under sekvensutförningen före alla första gruppens instruktioner har utförts eller, ifall den första gruppens instruktioner har utförts, före utförandet av vilken som helst andra grupps instruktion som är föremål för ett möjligt undantag, härmed möjliggörandes följande asynkroniska händelsers behandling;

man avbryter för ett nytt försök vilken som helst fraktionerad andra kods instruktionssekvens, som innehåller nämnda undersekvens, ifall ett asynkroniskt händelseavbrott förekommer under utförningsförsöket av nämnda undersekvens; och

man fördröjer behandlingen av den asynkroniska händelsen och slutför vilken som helst utförbar fraktionerad andra kods instruktionssekvens ifall det asynkroniska händelseavbrottet förekommer efter alla i nämnda andra grupp varande för möjliga avbrott känsliga statusuppdateringsinstruktioner.

5. Förfarande enligt patentkravet 4, k ä n n e t e c k n a d därav, att den första undersekvensen är en läs-modifiera-skriv-undersekvens.

6. Förfarande enligt patentkravet 5, k ä n n e t e c k n a d därav, att nämnda första undersekvens används för utförandet av en partiell skrivinstruktion eller en låst uppdateringsinstruktion, att nämnda första undersekvens innehåller en belastningslåst villkorlig lagringssekvens som läser och modifierar inmatningsdata, lagrar resultatdata villkorligt och misslyckas med att utföra nämnda första undersekvens ifall en motstridig skrivning har skett under dess utförning, eller slutför nämnda första undersekvens ifall en motstridig skrivning ej har skett under dess utförning.

7. Förfarande för att konvertera en första programkod till en andra programkod och för att utföra den andra programkoden medan man bibehåller status-atomicitet för den första kodens instruktioner, där den första programkoden är utförbar på en dator som har en första arkitektur adapterat till en första instruktionsbas och den andra programkoden är utförbar på en andra dator som har minnes- och registerstatus och en andra arkitektur adapterad till en andra instruktionsbas som är reducerad i relation till den första instruktionsbasen, k ä n n e t e c k n a d därav, att förfarandet omfattar:

användandet av en första dator för att konvertera den första kodens instruktioner till motsvarande andra kodens instruktioner enligt strukturkoden som definierar den första kodens instruktioner med hjälp av den andra kodens instruktioner;

organiserandet av den andra kodens instruktioner för varje första kods instruktion till en fraktionerad instruktionssekvens som har i följd minst två grupper, där den första gruppen inkluderar de andra kodens instruktioner som utför andra instruktioner än statusuppdatering och som kan avbrytas efter utförning utan risk för statusfel, och i vilken andra grupp finns alla minnes- och registerstatus uppdateringsinstruktioner inklusive en särskild skrivinstruktion som krävs för att implementera den konverterade första kodens instruktion;

en särskild skrivinstruktions strukturering för att innehålla en undersekvens för behandling av flera skrivningar som alla bör utföras utan avbrott;

användandet av en andra dator som är anpassad till den andra arkitekturen för att utföra den andra programkoden;

avgörandet av förekomsten av varje asynkronisk händelse under utförningen av den andra koden;

ett avbrytande för ett nytt försök av vilken som helst fraktionerad andra kods instruktionssekvens för att bevara status-atomicitet för den första kodens instruktioner och den första kodens instruktionsfraktionering ifall ett asynkroniskt händelseavbrott sker under sekvensutförningen före alla första gruppens instruktioner har utförts, eller ifall den första gruppens instruktioner har utförts före utförandet av vilken som helst sådan instruktion av den andra gruppen som kan utgöra ett möjligt undantag, härmed möjliggörandes behandlingen av följande asynkroniska händelse;

fördröjandet av behandlingen av det asynkroniska händelseavbrottet och slutförningen av vilken som helst utförbar fraktionerad andra kods instruktionssekvens A) ifall nämnda undersekvens ingår i den fraktionerade instruktionssekvensen och ifall den asynkroniska händelsesekvensen förekommer tidigast efter den första skrivningen under utförningen av nämnda instruktionssekvens eller B) ifall den asynkroniska händelsesekvensen förekommer efter alla i nämnda andra grupp varande statusuppdateringsinstruktioner som kan utgöra ett möjligt undantag.

8. Förfarande enligt patentkravet 1, k ä n n e t e c k n a d därav, att nämnda andra undersekvens innehåller en priviligierad arkitektur bibliotek (PAL, privileged architecture library) anropsrutin, som utför nämnda andra undersekvens och alla därtill hörande skrivningar, då nämnda PAL-anropsrutin har påbörjats, och att påbörjandet av nämnda PAL-anropsrutin är tillåtet, ifall det tidigare ej har förekommit avbrott i utförandet av ifrågavarande instruktionssekvens och ifall alla kvarvarande statusaccesser kan slutföras utan undantag,

och av, att i nämnda PAL-anropsrutin finns en första underrutin som testar den första utförda skrivningen för att avgöra, ifall den är en partiell skrivning, utför en belastningslåst villkorlig lagring för att utföra nämn-

da första skrivning, ifall den är en partiell skrivning, försöker selektivt på nytt utföra det belastningslåsta villkorliga skrivandet tills det kan slutföras, ifall den villkorliga skrivningen misslyckas på grund av nämnda andra prosessors motstridiga statusskrivning under utförningsförsök av nämnda första underrutin, slutför den villkorliga skrivningens skrivningsfunktion, ifall en motstridig statusskrivning ej har förekommit under det första försöket, ifall inget nytt försök har valts eller under följande nya försök, ifall nya försök har valts, och att nämnda PAL-anropsrutin låses för slutförning efter att nämnda första underrutin har blivit slutförd, och att den andra underrutinen sedan testar en andra utförbar skrivning för att avgöra om den är en partiell skrivning, utför en belastningslåst villkorlig skrivning för att utföra nämnda andra skrivning, ifall den är en partiell skrivning, försöker på nytt med det belstningslåsta villkorliga skrivandet tills det kan slutföras, ifall den villkorliga lagringen misslyckas på grund av nämnda andra prosessor utförda motstridiga statusskrivning under utförningsförsök av nämnda andra underrutin, och att varje följande utförda skrivning behandlas av en underrutin, som är väsentligt likadan som nämnda andra underrutin, ända tills alla skrivningar har utförts för att slutföra nämnda PAL-anrop.

9. Förfarande för att konvertera en första programkod till en andra programkod för att underlätta bibehållandet av den första kodens status-atomicitet medan den andra koden utförs, där den första programkoden är utförbar på en dator som har en första arkitektur adapterat till en första instruktionsbas och den andra programkoden är utförbar på en dator som har minnes- och registerstatus och en andra arkitektur adapterad till en andra instruktionsbas som är reducerad i relation till den första instruktionsbasen, k ä n n e t e c k n a d därav, att förfarandet omfattar:

användandet av en första dator för att konvertera den första kodens instruktioner till motsvarande andra kodens instruktioner enligt strukturkoden som definierar den första kodens instruktioner med hjälp av den andra kodens instruktioner;

organiserandet av den andra kodens instruktioner för varje första kods instruktion till en fraktionerad instruktionssekvens som har i följd minst två grupper, där den första gruppen inkluderar de andra kodens instruktioner som utför andra instruktioner än statusuppdatering och som kan avbrytas efter utförning utan risk för statusfel, och en andra grupp i vilken finns alla minnes- och registerstatus uppdateringsinstruktioner inklusive en särskild skrivinstruktion som krävs för att implementera den konverterade första kodens instruktion;

en särskild skrivinstruktions strukturering för att innehålla en undersekvens för att behandla en skrivning till ett första minnesställe i enlighet med kravet att nämnda ena skrivning bör utföras utan något avbrott och utan några i mellan kommande motstridiga skrivningar till nämnda första minnesställe av någon annan prosessor som kan vara kopplad till nämnda minnesställe;

struktureringen av en annan särskild skrivinstruktion för att innehålla en annan undersekvens för behandling av flere skrivningar som alla bör utföras utan avbrott

struktureringen av nämnda första undersekvens att avbrytas för ett nytt försök tills utföringen har slutförts lyckat, ifall nämnda andra prosessor utför en motstridig skrivning före slutförningen av nämnda första undersekvens;

struktureringen av nämnda första undersekvens att misslyckas ifall det asynkroniska händelseavbrottet sker under ett nytt försök av nämnda första undersekvens, varvid man möjliggör ett nytt försök av vilken som helst så-

dan fraktionerad andra kods instruktion, som innehåller nämnda första undersekvens; samt

struktureringen av nämnda första undersekvens för en prioriterad, icke avbrytbar utförning, och härvid fördröjningen av utförningen av vilket som helst asynkroniskt händelseavbrott under utförningen av den andra koden, ända tills nämnda första undersekvens utförning har avslutats.

10. Förfarande för utförning av en andra programkod medan man bibehåller statusatomicitet för en första programkods från vilken den andra koden är konverterad, där den första programkoden är utförbar på en dator som har en första arkitektur adapterat till en första instruktionsbas och den andra programkoden är utförbar på en andra dator som har minnes-och registerstatus och en andra arkitektur adapterad till en andra instruktionsbas som är reducerad i relation till den första instruktionsbasen, där den andra kodens instruktioner för varje första kods instruktion har organiserats till en fraktionerad instruktionssekvens som har i följd minst två grupper, en första grupp som inkluder de andra kodens instruktioner som utför andra instruktioner än statusuppdatering och som kan avbrytas efter utförning utan risk för statusfel, och en andra grupp som har alla minnes- och registerstatus uppdateringsinstruktioner inklusive en speciell skrivinstruktion som krävs för att implementera den konverterade första kodens instruktion, där en första särskild skrivinstruktion har strukturerats för att innehålla en första undersekvens för att behandla en skrivning till ett första minnesställe i enlighet med kravet att nämnda ena skrivning bör utföras utan något avbrott och utan några i mellan kommande motstridiga skrivningar till nämnda första minnesställe av vilken som helst annan prosessor som kan vara kopplad till nämnda minnesstatus, och där en annan särskild skrivinstruktion har strukturerats för att innehålla en annan undersekvens för behandling av flere skriv-

ningar som alla bör utföras utan avbrott, k ä n n e t e c k n a d därav, att förfarandet omfattar följande skeden:

man använder en andra dator anpassad för den andra arkitekturen för att utföra den andra programkoden;

man avgör förekomsten av varje asynkronisk händelse under utförningen av den andra koden;

man avgör under utförningen av den andra koden förekomsten av varje motstridiga skrivning till nämnda första minnesställe av nämnda andra prosessor ifall den är kopplad till nämnda minnesstatus;

man avbryter för ett nytt försök vilken som helst fraktionerad andra kods instruktionssekvens för att bevara status-atomicitet för den första kodens instruktioner och den första kodens instruktionsfraktionering ifall ett asynkroniskt händelseavbrott sker under sekvensutförningen före alla första gruppens instruktioner har utförts eller, ifall den första gruppens instruktioner har utförts före utförandet av vilken som helst andra grupps instruktion som kan utgöra ett möjligt undantag, varvid man möjliggör behandlingen av följande asynkroniska händelse;

man avbryter, och försöker på nytt tills utförningen lyckats, nämnda första särskilda instruktionssekvens, i vilken som helst sådan fraktionerad andra instruktionssekvens som innehåller nämnda första undersekvens, ifall nämnda andra prosessor har utfört en motstridig skrivning före avslutandet av nämnda första undersekvens utförning;

man avbryter för ett nytt försök vilken som helst fraktionerad andra kods instruktionssekvens, som innehåller nämnda första undersekvens, ifall ett asynkroniskt händelseavbrott förekommer under ett utförningsförsök av nämnda första undersekvens; samt

man fördröjer behandlingen av det asynkroniska händelseavbrottet och slutför vilken som helst utförbar andra kods instruktionssekvens A) ifall nämnda undersek-

vens ingår i den fraktionerade instruktionssekvensen och ifall den asynkroniska händelsesekvensen förekommer tidigast efter den första skrivningen under utförningen av nämnda instruktionssekvens eller B) ifall det asynkroniska händelseavbrottet förekommer efter utförningen av alla i nämnda andra grupp varande, för avbrott känsliga statusuppdateringsinstruktioner.

11. System för att konvertera en första programkod till en andra programkod och för att utföra den andra programkoden på ett sätt som bibehåller status-atomicitet för den första kodens instruktioner, där den första programkoden är utförbar på en dator som har en första arkitektur adapterat till en första instruktionsbas och den andra programkoden är utförbar på en andra dator som har minnes- och registerstatus och en andra arkitektur adapterad till en andra instruktionsbas som är reducerad i relation till den första instruktionsbasen, k ä n n e t e c k n a d därav, att systemet omfattar:

ett första datorsystem som har en första prosessor för att konvertera den första programkoden till en andra programkod, och ett första minnessystem som är anslutet till nämnda första prosessor;

ett organ för konvertering av varje på varandra följande instruktion i den första koden till den andra kodens instruktioner, enligt nämnda strukturkod, för lagring som den andra programkoden i nämnda första minnessystem;

ett organ för att organisera den andra kodens instruktioner för varje första kods instruktion till en fraktionerad instruktionssekvens som har i följd minst två grupper, en första grupp som inkluderar de andra kodens instruktioner som utför andra instruktioner än statusuppdatering och som kan avbrytas efter utförning utan risk för statusfel, och en andra grupp som har alla minnes- och registerstatus uppdateringsinstruktioner inklusive speci-

ella skrivinstruktioner som krävs för att implementera den konverterade första kodens instruktion;

ett organ för strukturering av en första särskild skrivinstruktion för att innehålla en första undersekvens för att behandla en skrivning till ett första minnesställe i enlighet med kravet att nämnda ena skrivning bör utföras utan något avbrott och utan några i mellan kommande motstridiga skrivningar till nämnda första minnesställe av en annan prosessor som möjligen kan vara kopplad till nämnda minnesstatus;

ett organ för en annan särskild skrivinstruktions strukturering för att innehålla en andra undersekvens för behandling av flera skrivningar som alla bör utföras utan avbrott;

ett andra datorsystem för utförning av en andra kod som har utvecklats som nämnda första datorsystems utskrift, vilket nämnda andra datorsystem har nämnda andra arkitektur och andra prosessor och minnes- och registerstatus, som innehåller ett till nämnda andra prosessor anslutet andra minnessystem,

ett organ för definiering av förekomsten av varje asynkronisk händelse under utförningen av den andra koden och definieringen av förekomsten av en motstridig skrivning till nämnda första minnesställe av nämnda andra prosessor ifall nämnda andra prosessor är kopplad till nämnda minnesställe;

ett organ för avbrytandet av vilken som helst fraktionerad andra kods instruktionssekvens för att bevara status-atomicitet för den första kodens instruktioner och den första kodens instruktionsfraktionering ifall ett asynkroniskt händelseavbrott sker under sekvensutförningen före alla första gruppens instruktioner har utförts eller, ifall den första gruppens instruktioner har utförts före utförandet av vilken som helst sådan andra grupps instruktion som är utsatt för ett möjligt avbrott, på vilket sätt

man möjliggör behandlingen av följande asynkroniska händelse;

ett organ för nämnda första särskilda instruktionssekvens avbrytande för ett nytt försök i vilken som helst fraktionerad andra kods instruktionssekvens som innehåller nämnda första undersekvens, ifall nämnda andra prosessor har utfört en motstridig skrivning under nämnda första undersekvens;

ett organ för att avbryta för ett nytt försök vilken som helst sådan fraktionerad andra kods instruktionssekvens, som innehåller nämnda första undersekvens, ifall ett asynkroniskt händelseavbrott förekommer under ett utförningsförsök av nämnda första undersekvens; samt

ett organ för att fördröja behandlingen av det asynkroniska händelseavbrottet och för att slutföra vilken som helst utförbar andra kods instruktionssekvens A) ifall nämnda undersekvens ingår i den fraktionerade instruktionssekvensen och ifall den asynkroniska händelsesekvensen förekommer tidigast efter den första skrivningen under utförningen av nämnda instruktionssekvens eller B) ifall det asynkroniska händelseavbrottet förekommer efter utförningen av alla i nämnda andra grupp varande, för avbrott känsliga statusuppdateringsinstruktioner.

12. System enligt patentkravet 11, k ä n n e t e c k n a t därav, att den första undersekvensen är en läs-modifiera-skriv-undersekvens,

och av, att nämnda första undersekvens används för utförandet av en partiell skrivinstruktion eller en låst uppdateringsinstruktion, där nämnda första undersekvens innehåller en belastningslåst villkorlig lagringssekvens som läser och modifierar inmatningsdata, lagrar resultatdata villkorligt och misslyckas med att utföra nämnda undersekvens ifall en motstridig skrivning har skett under dess utförning, eller slutför nämnda första undersekvens

ifall en motstridig skrivning ej har skett under dess utförning,

och av att nämnda andra undersekvens innehåller en priviligierad arkitektur bibliotek (PAL, privileged architecture library) anropsrutin, som utför nämnda andra undersekvens och alla därtill hörande skrivningar, då nämnda PAL-anropsrutin har påbörjats, och att påbörjandet av nämnda PAL-anropsrutin är tillåtet, ifall det tidigare ej har förekommit avbrott i utförandet av ifrågavarande instruktionssekvens och ifall alla kvarvarande statusanvisningar kan slutföras utan undantag.

13. System enligt patentkravet 12, k ä n n e t e c k n a t därav, att i nämnda PAL- anropsrutin finns en första underrutin som testar den första utförda skrivningen för att avgöra, ifall den är en partiell skrivning, utför en belastningslåst villkorlig lagring för att utföra nämnda första skrivning, ifall den är en partiell skrivning, försöker selektivt på nytt utföra det belastningslåsta villkorliga skrivandet tills det kan slutföras, ifall den villkorliga skrivningen misslyckas på grund av nämnda andra prosessors motstridiga statusskrivning under utförningsförsök av nämnda första underrutin, slutför den villkorliga skrivningens skrivningsfunktion, ifall en motstridig statusskrivning ej har förekommit under det första försöket, ifall inget nytt försök har valts eller under följande nya försök, ifall nya försök har valts, och att nämnda PAL-anropsrutin låses för slutförning efter att nämnda första underrutin har blivit slutförd, och att en annan underrutin sedan testar en andra utförbar skrivning för att avgöra om den är en partiell skrivning, utför en belastningslåst villkorlig skrivning för att utföra nämnda andra skrivning, ifall den är en partiell skrivning, försöker på nytt med det belastningslåsta villkorliga skrivandet tills det kan slutföras, ifall den villkorliga lagringen misslyckas på grund av en av nämnda andra prosessor

utförd motstridig statusskrivning under utförningsförsök av nämnda andra underrutin, och att varje följande utförda skrivning behandlas av en underrutin, som är väsentligt likadan som nämnda andra underrutin, ända tills alla skrivningar har utförts för att slutföra nämnda PAL-anrop.

14. System för att konvertera en första programkod till en andra programkod och för att utföra den andra programkoden på ett sätt som bibehåller status-atomicitet för den första kodens instruktioner, där den första programkoden är utförbar på en dator som har en första arkitektur adapterat till en första instruktionsbas och den andra programkoden är utförbar på en andra dator som har minnes- och registerstatus och en andra arkitektur adapterad till en andra instruktionsbas som är reducerad i relation till den första instruktionsbasen, k ä n n e t e c k n a d därav, att systemet omfattar:

ett första datorsystem som har en första prosessor för att konvertera den första programkoden till en andra programkod, och ett första minnessystem som är anslutet till nämnda första prosessor;

ett organ för konvertering av varje på varandra följande instruktion i den första koden till den andra kodens instruktioner enligt nämnda strukturkod för lagring som den andra programkoden i nämnda första minnessystem;

ett organ för att organisera den andra kodens instruktioner för varje första kods instruktion till en fraktionerad instruktionssekvens som har i följd minst två grupper, en första grupp som inkluderar de andra kodens instruktioner som utför andra instruktioner än statusuppdatering och som kan avbrytas efter utförning utan risk för statusfel, och en andra grupp som har alla minnes- och registerstatus uppdateringsinstruktioner inklusive speciella skrivinstruktioner som krävs för att implementera den konverterade första kodens instruktion;

ett organ för strukturering av en särskild skrivinstruktion för att innehålla en första undersekvens för att behandla en skrivning till ett första minnesställe i enlighet med kravet att nämnda ena skrivning bör utföras utan något avbrott och utan några i mellan kommande motstridiga skrivningar till nämnda första minnesställe av en annan möjlig prosessor som kan vara kopplad till nämnda minnesstatus;

ett andra datorsystem för utförning av en andra kod som har utvecklats som nämnda första datorsystems utskrift, vilket nämnda andra datorsystem har nämnda andra arkitektur och andra prosessor och minnes- och registerstatus, som innehåller ett till nämnda andra prosessor anslutet andra minnessystem,

ett organ för definiering av förekomsten av varje asynkronisk händelse under utförningen av den andra koden och definieringen av förekomsten av en motstridig skrivning till nämnda första minnesställe av nämnda andra prosessor ifall nämnda andra prosessor är kopplad till nämnda minnesställe;

ett organ för avbrytandet av vilken som helst fraktionerad andra kods instruktionssekvens för att bevara status-atomicitet för den första kodens instruktioner och den första kodens instruktionsfraktionering ifall ett asynkroniskt händelseavbrott sker under sekvensutförningen före alla första gruppens instruktioner har utförts eller, ifall den första gruppens instruktioner har utförts före utförandet av vilken som helst sådan andra grupps instruktion som är utsatt för ett möjligt avbrott, på vilket sätt man möjliggör behandlingen av följande asynkroniska händelse;

ett organ för nämnda första särskilda instruktionssekvens avbrytande för ett nytt försök i vilken som helst fraktionerad andra kods instruktionssekvens som innehåller nämnda första undersekvens, ifall nämnda andra prosessor

har utfört en motstridig skrivning under nämnda första undersekvens;

ett organ för att avbryta för ett nytt försök vilken som helst sådan fraktionerad andra kods instruktionssekvens, som innehåller nämnda första undersekvens, ifall ett asynkroniskt händelseavbrott förekommer under ett utförningsförsök av nämnda första undersekvens; samt

ett organ som innehåller nämnda andra prosessor och nämnda andra minnessystem, för att fördröja behandlingen av det asynkroniska händelseavbrottet och för att slutföra vilken som helst utförbar andra kods instruktionssekvens, ifall den asynkroniska händelsen förekommer efter utförningen av alla sådana i nämnda andra grupp varande statusuppdateringsinstruktioner som kan utgöra ett möjligt undantag.

15. System enligt patentkravet 14, k ä n n e t e c k n a t därav, att den första undersekvensen är en läs-modifiera-skriv-undersekvens,

och av, att nämnda första undersekvens används för utförandet av en partiell skrivinstruktion eller en låst uppdateringsinstruktion, där nämnda första undersekvens innehåller en belastningslåst villkorlig lagringssekvens som läser och modifierar inmatningsdata, lagrar resultatdata villkorligt och misslyckas i nämnda första undersekvens ifall en motstridig skrivning har skett under dess utförning, eller slutför nämnda första undersekvens ifall en motstridig skrivning ej har skett under dess utförning.

16. System för att konvertera en första programkod till en andra programkod och för att utföra den andra programkoden på ett sätt som bibehåller status-atomicitet för den första kodens instruktioner, där den första programkoden är utförbar på en dator som har en första arkitektur adapterat till en första instruktionsbas och den andra programkoden är utförbar på en andra dator som har minnes- och registerstatus och en andra arkitektur adapterad till

en andra instruktionsbas som är reducerad i relation till den första instruktionsbasen, k ä n n e t e c k n a d därav, att systemet omfattar:

ett första datorsystem som har en första prosessor för att konvertera den första programkoden till en andra programkod, och ett första minnessystem som är anslutet till nämnda första prosessor;

ett organ för konvertering av varje på varandra följande instruktion i den första koden till den andra kodens instruktioner enligt nämnda strukturkod för lagring som den andra programkoden i nämnda första minnessystem;

ett organ för att organisera den andra kodens instruktioner för varje första kods instruktion till en fraktionerad instruktionssekvens som har i följd minst två grupper, en första grupp som inkluderar de andra kodens instruktioner som utför andra instruktioner än statusuppdatering och som kan avbrytas efter utförning utan risk för statusfel, och en andra grupp som har alla minnes- och registerstatus uppdateringsinstruktioner inklusive speciella skrivinstruktioner som krävs för att implementera den konverterade första kodens instruktion;

ett organ för strukturering av en särskild skrivinstruktion för att innehålla en undersekvens för att behandla flera skrivinstruktioner som alla bör utföras utan något avbrott;

ett andra datorsystem för utförning av en andra kod som har utvecklats som det nämnda första datorsystemets utskrift, vilket nämnda andra datorsystem har nämnda andra arkitektur och andra prosessor och minnes- och registerstatus, som innehåller ett till nämnda andra prosessor anslutet andra minnessystem,

ett organ för definiering av förekomsten av varje asynkronisk händelse under utförningen av den andra koden;

ett organ för avbrytandet av vilken som helst fraktionerad andra kods instruktionssekvens för att bevara

status-atomicitet för den första kodens instruktioner och den första kodens instruktionsfraktionering ifall ett asynkroniskt händelseavbrott sker under sekvensutförningen före alla första gruppens instruktioner har utförts eller, ifall den första gruppens instruktioner har utförts före utförandet av vilken som helst sådan andra grupps instruktion som är utsatt för ett möjligt avbrott, på vilket sätt man möjliggör behandlingen av följande asynkroniska händelse;

ett organ, som innehåller nämnda andra prosessor och nämnda andra minnessystem, för att fördröja behandlingen av det asynkroniska händelseavbrottet och för att slutföra vilken som helst utförbar andra kods instruktionssekvens A) ifall nämnda undersekvens ingår i den fraktionerade instruktionssekvensen och ifall den asynkroniska händelsesekvensen förekommer tidigast efter den första skrivningen under utförningen av nämnda instruktionssekvens eller B) ifall det asynkroniska händelseavbrottet förekommer efter utförningen av alla i nämnda andra grupp varande, för avbrott känsliga statusuppdateringsinstruktioner.

17. System för att utföra en andra programkod samtidigt som man bibehåller den första programkodens statusatomicitet från vilken den andra programkoden är konverterad, där den första programkoden är utförbar på en dator som har en första arkitektur adapterat till en första instruktionsbas och den andra programkoden är utförbar på en dator som har minnes- och registerstatus och en andra arkitektur adapterad till en andra instruktionsbas som är reducerad i relation till den första instruktionsbasen, där den andra kodens instruktioner för varje första kods instruktion är organiserad till en fraktionerad instruktionssekvens som har i följd minst två grupper, där den första gruppen inkluderar de andra kodens instruktioner som utför andra instruktioner än statusuppdatering och som

kan avbrytas efter utförning utan risk för statusfel, och där den andra gruppen har alla minnes- och registerstatus uppdateringsinstruktioner inklusive speciella skrivinstruktioner som krävs för att implementera den konverterade första kodens instruktion, där en första särskild skrivinstruktion har strukturerats för att innehålla en första undersekvens för att behandla en skrivning till ett första minnesställe i enlighet med kravet att nämnda ena skrivning bör utföras utan något avbrott och utan några i mellan kommande motstridiga skrivningar till nämnda första minnesställe av vilken som helst annan prosessor som möjligen kan vara kopplad till nämnda minnesstatus, och där en annan särskild skrivinstruktion har strukturerats att innehålla en andra undersekvens för behandling av flera skrivningar som alla bör utföras utan avbrott, k ä n n e t e c k n a t därav, att systemet omfattar:

ett organ för användning av en andra dator anpassad till en andra arkitektur för utförning av en andra programkod;

ett organ som definierar förekomsten av varje asynkroniska händelse under utförningen av den andra koden;

ett organ som under utförningen av den andra koden definierar förekomsten av varje motstridig skrivning till nämnda första minnesställe av nämnda andra prosessor ifall nämnda andra prosessor är kopplad till nämnda minnesställe;

ett organ som avbryter för ett nytt försök vilken som helst fraktionerad andra kods instruktionssekvens för att bevara status-atomicitet för den första kodens instruktioner och den första kodens instruktionsfraktionering ifall ett asynkroniskt händelseavbrott sker under sekvensutförningen före alla första gruppens instruktioner har utförts eller, ifall den första gruppens instruktioner har utförts före utförandet av vilken som helst sådan andra grupps instruktion som kan utgöra ett möjligt undantag,

och möjliggör på detta sätt behandlingen av följande asynkroniska händelse;

ett organ för nämnda första särskilda instruktionssekvens avbrytande för ett nytt försök tills utförningen lyckats i vilken som helst sådan fraktionerad andra kods instruktionssekvens som innehåller nämnda första undersekvens, ifall nämnda andra prosessor har utfört en motstridig skrivning före slutförningen av nämnda första undersekvens;

ett organ som avbryter för ett nytt försök vilken som helst sådan fraktionerad andra kods instruktionssekvens, som innehåller nämnda första undersekvens, ifall ett asynkroniskt händelseavbrott förekommer under ett utförningsförsök av nämnda första undersekvens; samt

ett organ som fördröjer behandlingen av det asynkroniska händelseavbrottet och som slutför vilken som helst utförbar andra kods instruktionssekvens A) ifall nämnda undersekvens ingår i den fraktionerade instruktionssekvensen och ifall det asynkroniska händelseavbrottet förekommer tidigast efter den första skrivningen under utförningen av nämnda instruktionssekvens eller B) ifall det asynkroniska händelseavbrottet förekommer efter utförningen av alla sådana i nämnda andra grupp varande statusuppdateringsinstruktioner som kan utgöra ett möjligt undantag.

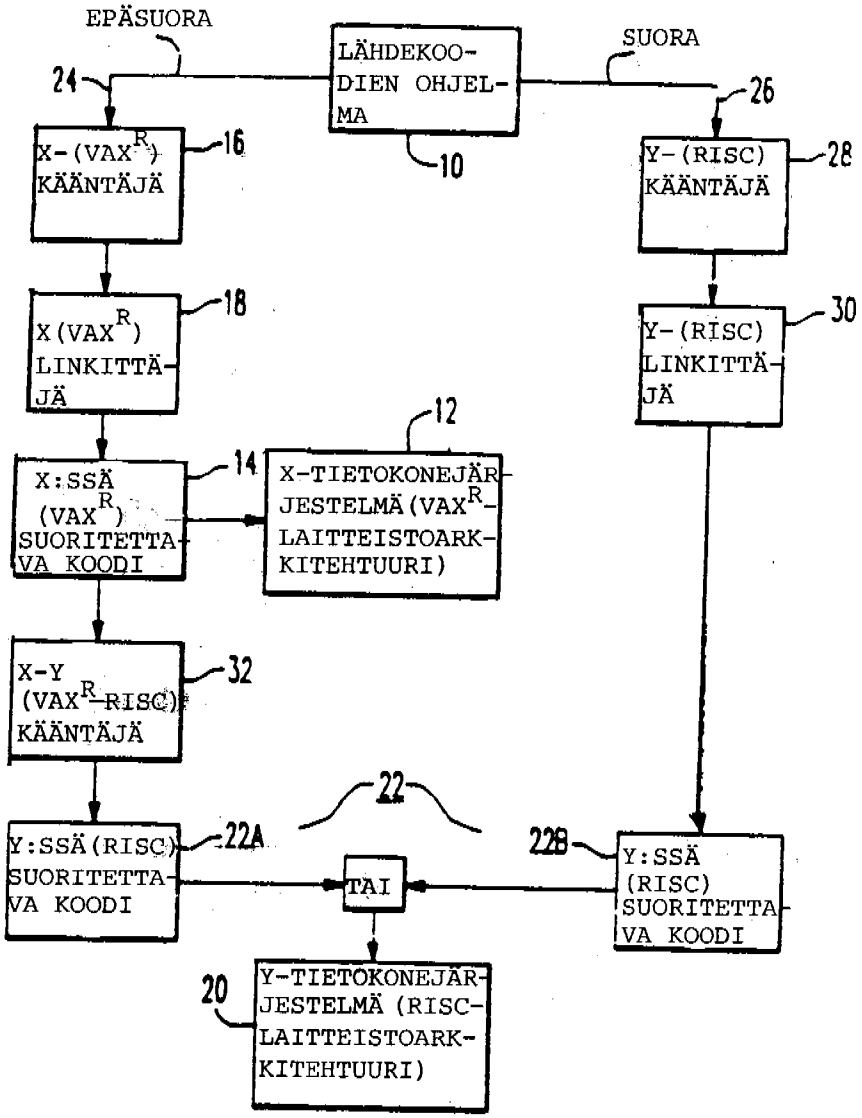
EPÄSUORA
LÄHDEKOO-
DIEN OHJEL-
MA
SUORA
24
26
10
X-(VAX^R)
KÄÄNTÄJÄ
16
Y-(RISC)
KÄÄNTÄJÄ
28
X(VAX^R)
LINKITTÄ-
JÄ
18
Y-(RISC)
LINKITTÄ-
JÄ
30
12
X:SSÄ
(VAX^R)
SUORITETTA-
VA KOODI
14
X-TIETOKONEJÄR-
JESTELMÄ(VAX^R-
LAITTEISTOARK-
KITEHTUURI)
X-Y
(VAX^R-RISC)
KÄÄNTÄJÄ
32
22
Y:SSÄ(RISC)
SUORITETTA-
VA KOODI
22A
TAI
22B
Y:SSÄ
(RISC)
SUORITETTA-
VA KOODI
20
Y-TIETOKONEJÄR-
JESTELMÄ (RISC-
LAITTEISTOARK-
KITEHTUURI)

Fig. 1

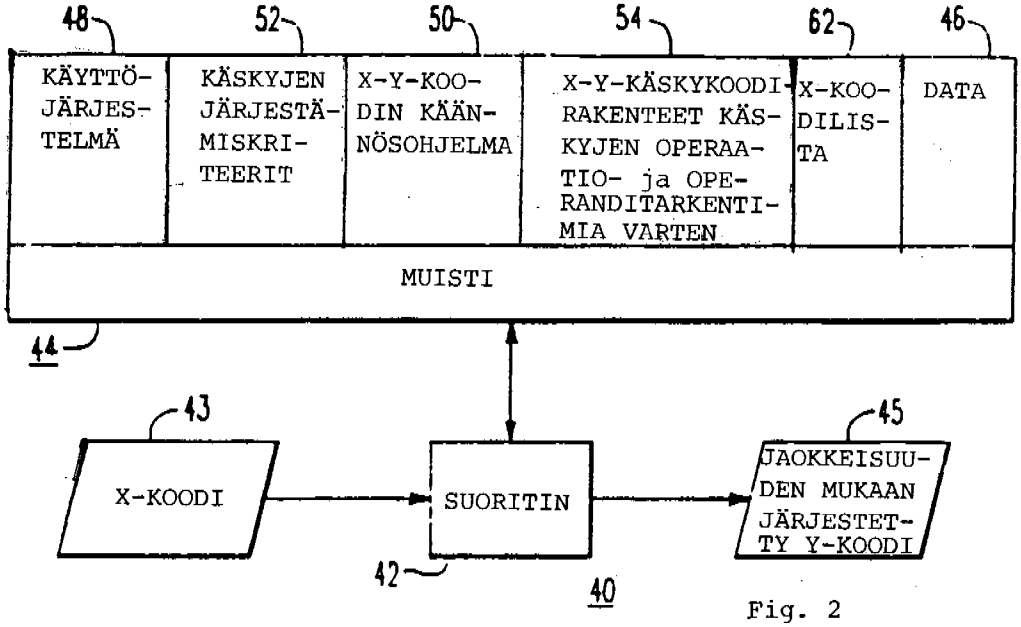
48
52
50
54
62
46
KÄYTTÖ-JÄRJES-TELMÄ
KÄSKYJEN JÄRJESTÄ-MISKRI-TEERIT
X-Y-KOO-DIN KÄÄN-NÖSOHJELMA
X-Y-KÄSKYKOODI-RAKENTEET KÄS-KYJEN OPERAA-TIO- ja OPE-RANDITARKENTI-MIA VARTEN
X-KOO-DILIS-TA
DATA
MUISTI
44
43
X-KOODI
SUORITIN
45
JAOKKEISUU-DEN MUKAAN JÄRJESTET-TY Y-KOODI
42
40

Fig. 2

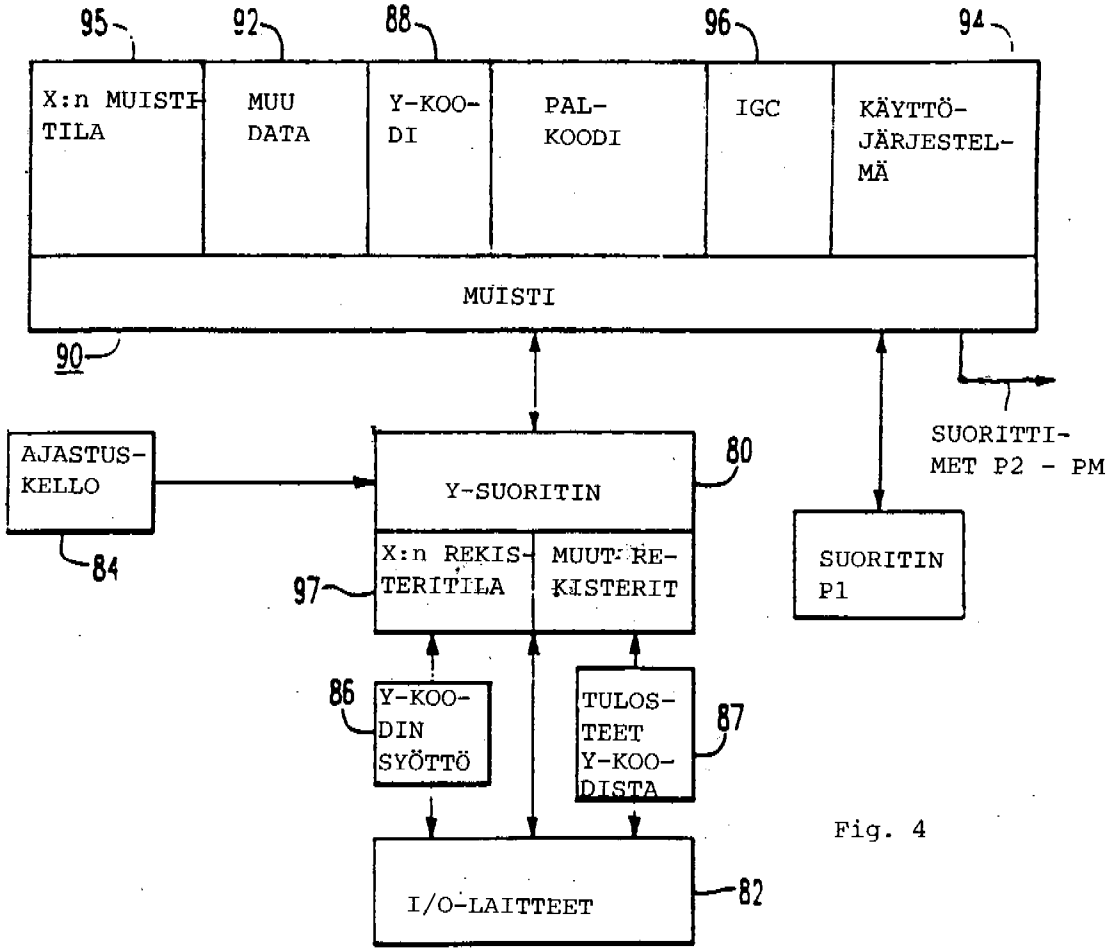
95
92
88
96
94
X:n MUISTI-TILA
MUU DATA
Y-KOO-DI
PAL-KOODI
IGC
KÄYTTÖ-JÄRJESTEL-MÄ
MUISTI
90
AJASTUS-KELLO
84
Y-SUORITIN
80
X:n REKIS-TERITILA
97
MUUT RE-KISTERIT
SUORITTI-MET P2 - PM
SUORITIN P1
86
Y-KOO-DIN SYÖTTÖ
TULOS-TEET Y-KOO-DISTA
87
I/O-LAITTEET
82

Fig. 4

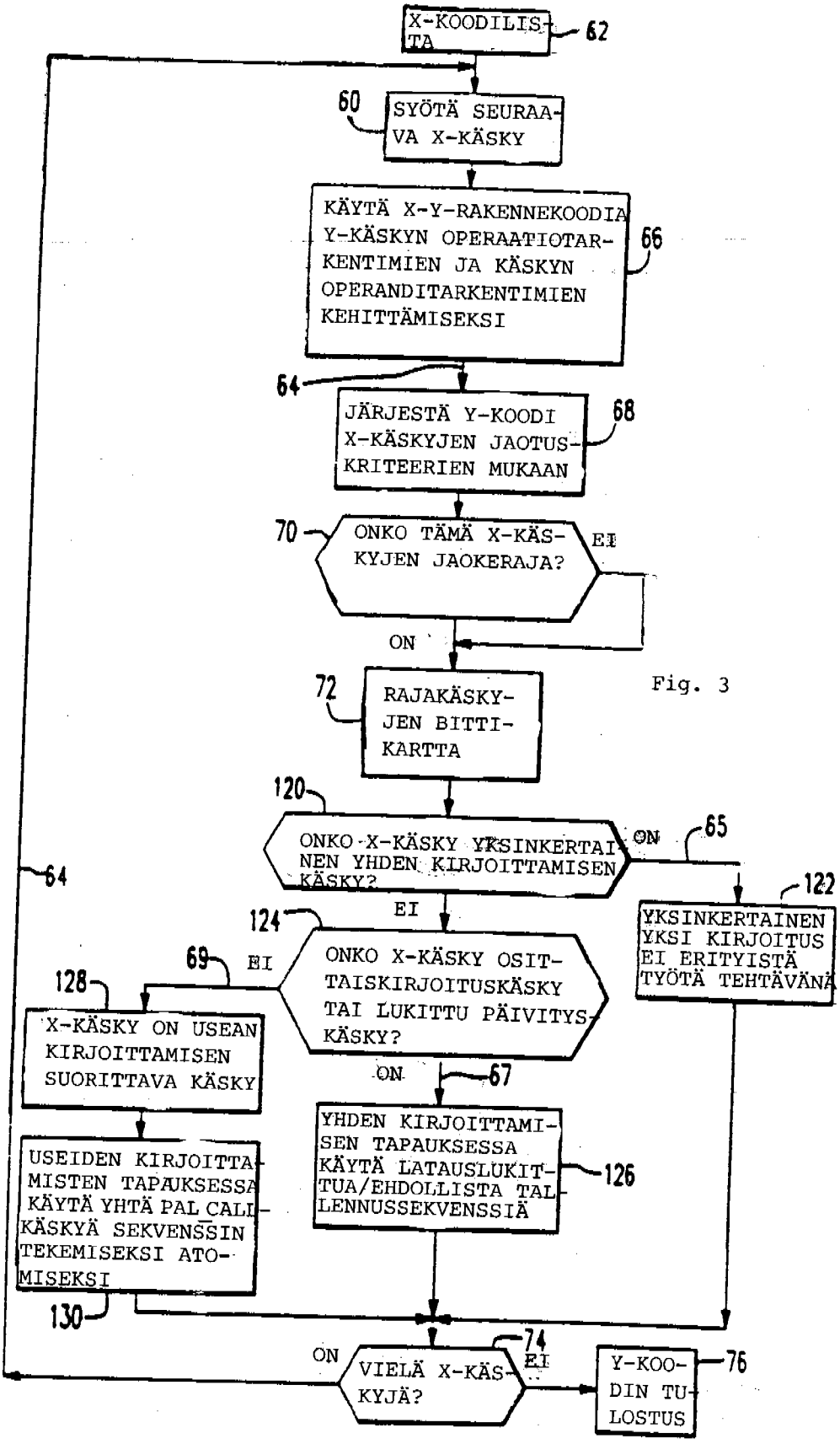
X-KOODILISTA
62
60
SYÖTÄ SEURAAVA X-KÄSKY
KÄYTÄ X-Y-RAKENNEKOODIA Y-KÄSKYN OPERAATIOTARKENTIMIEN JA KÄSKYN OPERANDITARKENTIMIEN KEHITTÄMISEKSI
66
64
JÄRJESTÄ Y-KOODI X-KÄSKYJEN JAOTUSKRITEERIEN MUKAAN
68
70
ONKO TÄMÄ X-KÄSKYJEN JAOKERAJA?
EI
ON
72
RAJAKÄSKYJEN BITTIKARTTA
120
ONKO X-KÄSKY YKSINKERTAINEN YHDEN KIRJOITTAMISEN KÄSKY?
ON
65
122
YKSINKERTAINEN YKSI KIRJOITUS EI ERITYISTÄ TYÖTÄ TEHTÄVÄNÄ
64
EI
124
69
EI
ONKO X-KÄSKY OSITTAISKIRJOITUSKÄSKY TAI LUKITTU PÄIVITYSKÄSKY?
128
X-KÄSKY ON USEAN KIRJOITTAMISEN SUORITTAVA KÄSKY
ON
67
YHDEN KIRJOITTAMISEN TAPAUKSESSA KÄYTÄ LATAUSLUKITTUA/EHDOLLISTA TALLENNUSSEKVENSSIÄ
126
USEIDEN KIRJOITTAMISTEN TAPAUKSESSA KÄYTÄ YHTÄ PAL CALL-KÄSKYÄ SEKVENSSIN TEKEMISEKSI ATOMISEKSI
130
ON
74
EI
VIELÄ X-KÄSKYJÄ?
Y-KOODIN TULOSTUS
76

Fig. 3

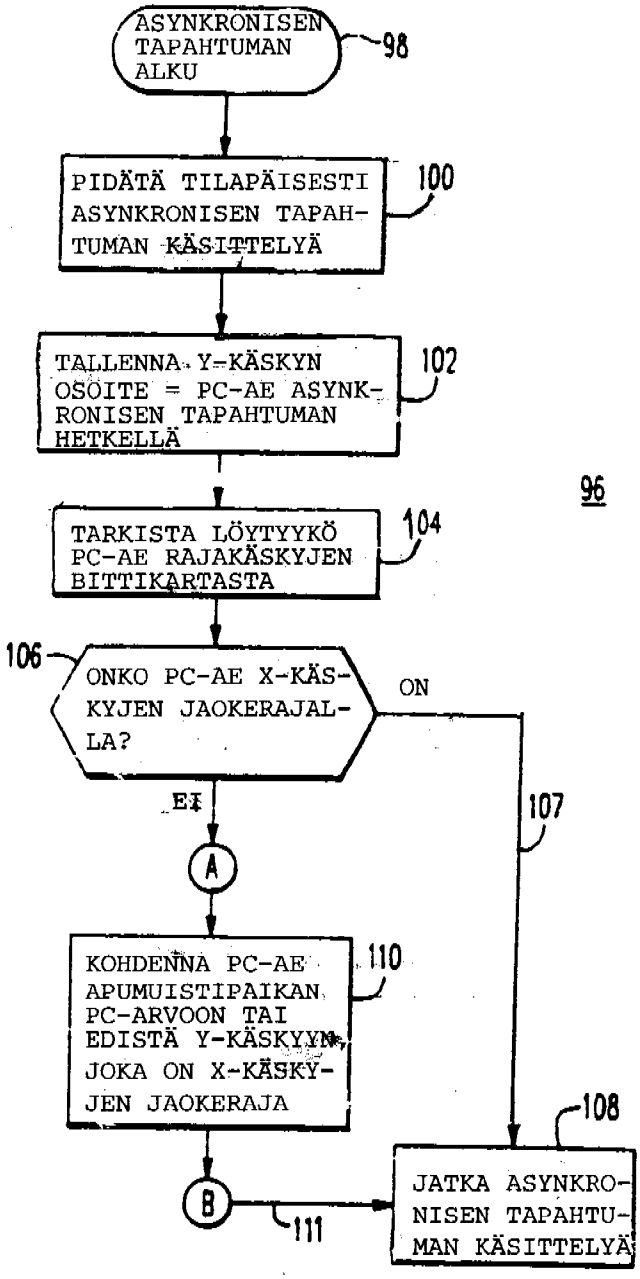
ASYNKRONISEN TAPAHTUMAN ALKU
98
PIDÄTÄ TILAPÄISESTI ASYNKRONISEN TAPAHTUMAN KÄSITTELYÄ
100
TALLENNA Y-KÄSKYN OSOITE = PC-AE ASYNKRONISEN TAPAHTUMAN HETKELLÄ
102
96
TARKISTA LÖYTYYKÖ PC-AE RAJAKÄSKYJEN BITTIKARTASTA
104
106
ONKO PC-AE X-KÄSKYJEN JAOKERAJALLA?
ON
EI
107
A
KOHDENNA PC-AE APUMUISTIPAIKAN PC-ARVOON TAI EDISTÄ Y-KÄSKYYN, JOKA ON X-KÄSKYJEN JAOKERAJA
110
108
B
111
JATKA ASYNKRONISEN TAPAHTUMAN KÄSITTELYÄ

Fig. 5A

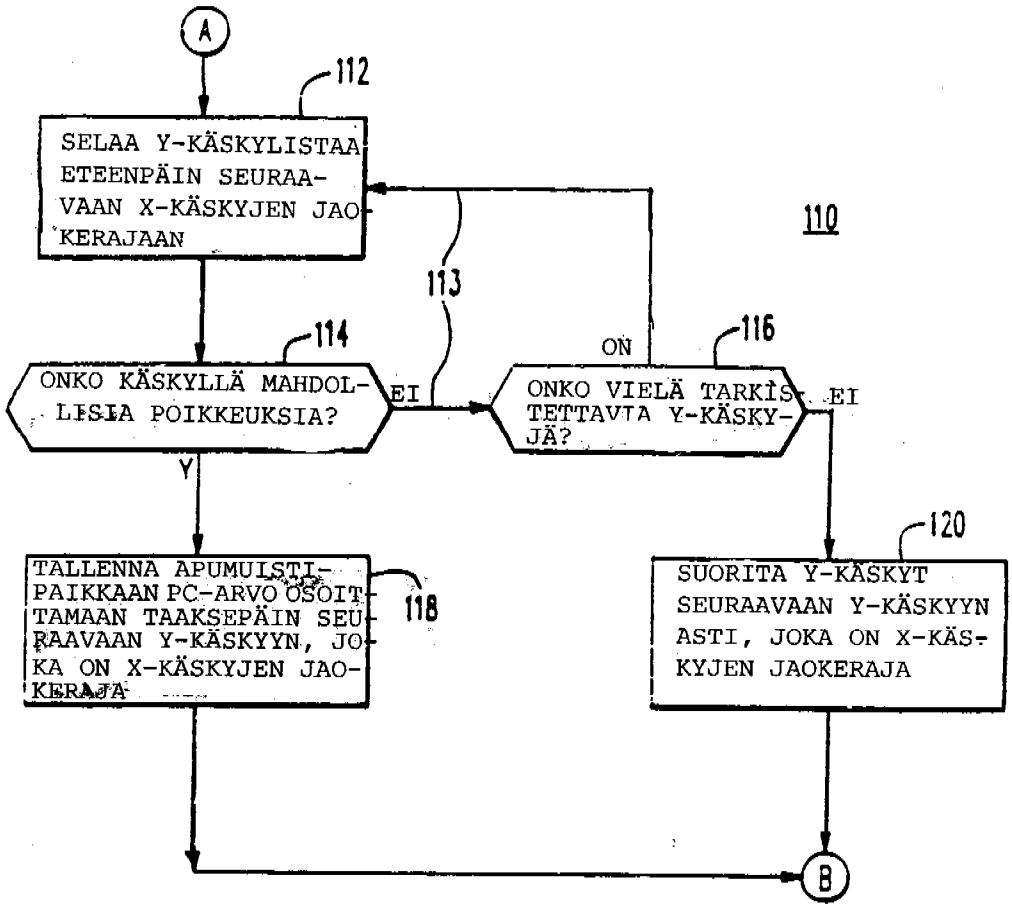
A
112
SELAA Y-KÄSKYLISTAA ETEENPÄIN SEURAAVAAN X-KÄSKYJEN JAOKERAJAAN
110
113
114
ONKO KÄSKYLLÄ MAHDOLLISIA POIKKEUKSIA?
EI
ON
116
ONKO VIELÄ TARKISTETTAVIA Y-KÄSKYJÄ?
EI
Y
120
TALLENNA APUMUISTIPAIKKAAN PC-ARVO OSOITTAMAAN TAAKSEPÄIN SEURAAVAAN Y-KÄSKYYN, JOKA ON X-KÄSKYJEN JAOKERAJA
118
SUORITA Y-KÄSKYT SEURAAVAAN Y-KÄSKYYN ASTI, JOKA ON X-KÄSKYJEN JAOKERAJA
B

Fig. 5B

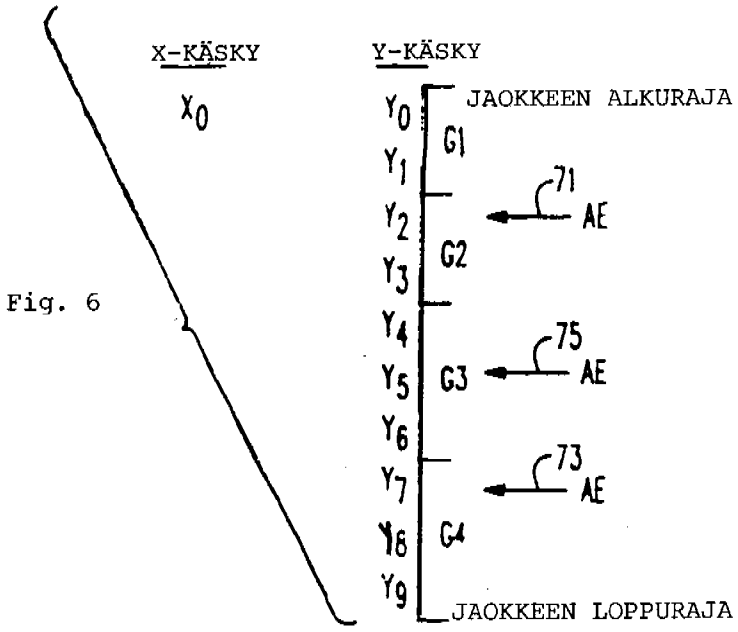
X-KÄSKY
Y-KÄSKY
X_0
Y_0
JAOKKEEN ALKURAJA
G1
Y_1
Y_2
71
AE
G2
Y_3
Y_4
75
Y_5
G3
AE
Y_6
Y_7
73
AE
Y_8
G4
Y_9
JAOKKEEN LOPPURAJA

Fig. 6

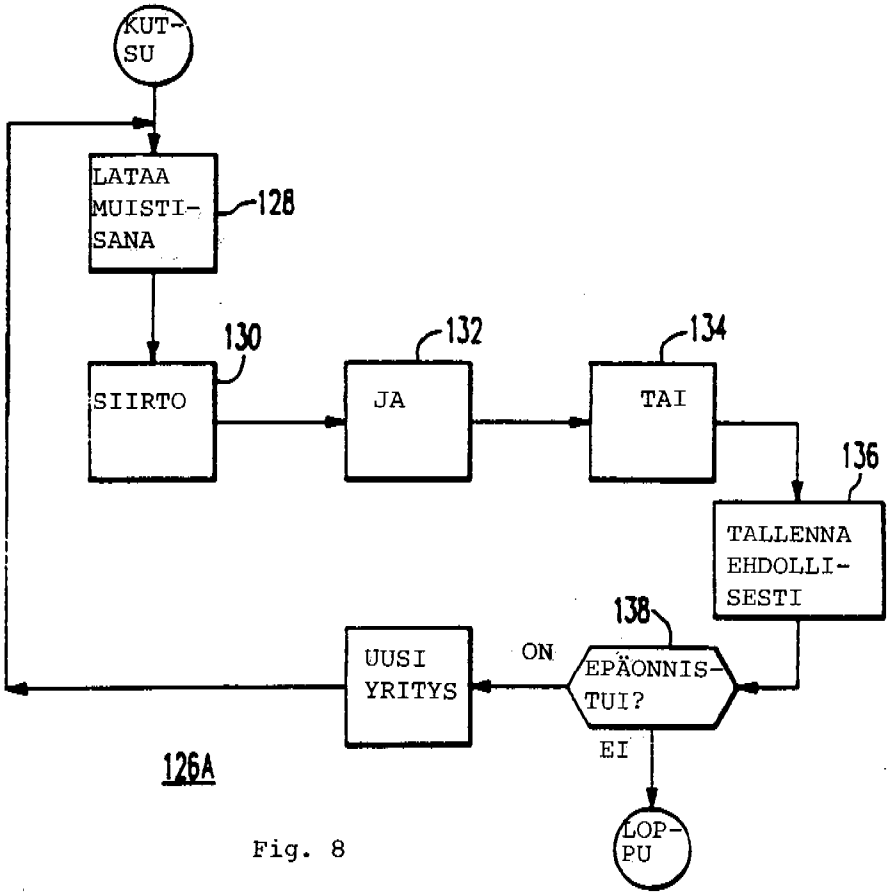
KUT-
SU
LATAA
MUISTI-
SANA
128
130
SIIRTO
132
JA
134
TAI
136
TALLENNA
EHDOLLI-
SESTI
138
UUSI
YRITYS
ON
EPÄONNIS-
TUI?
EI
LOP-
PU
126A

Fig. 8

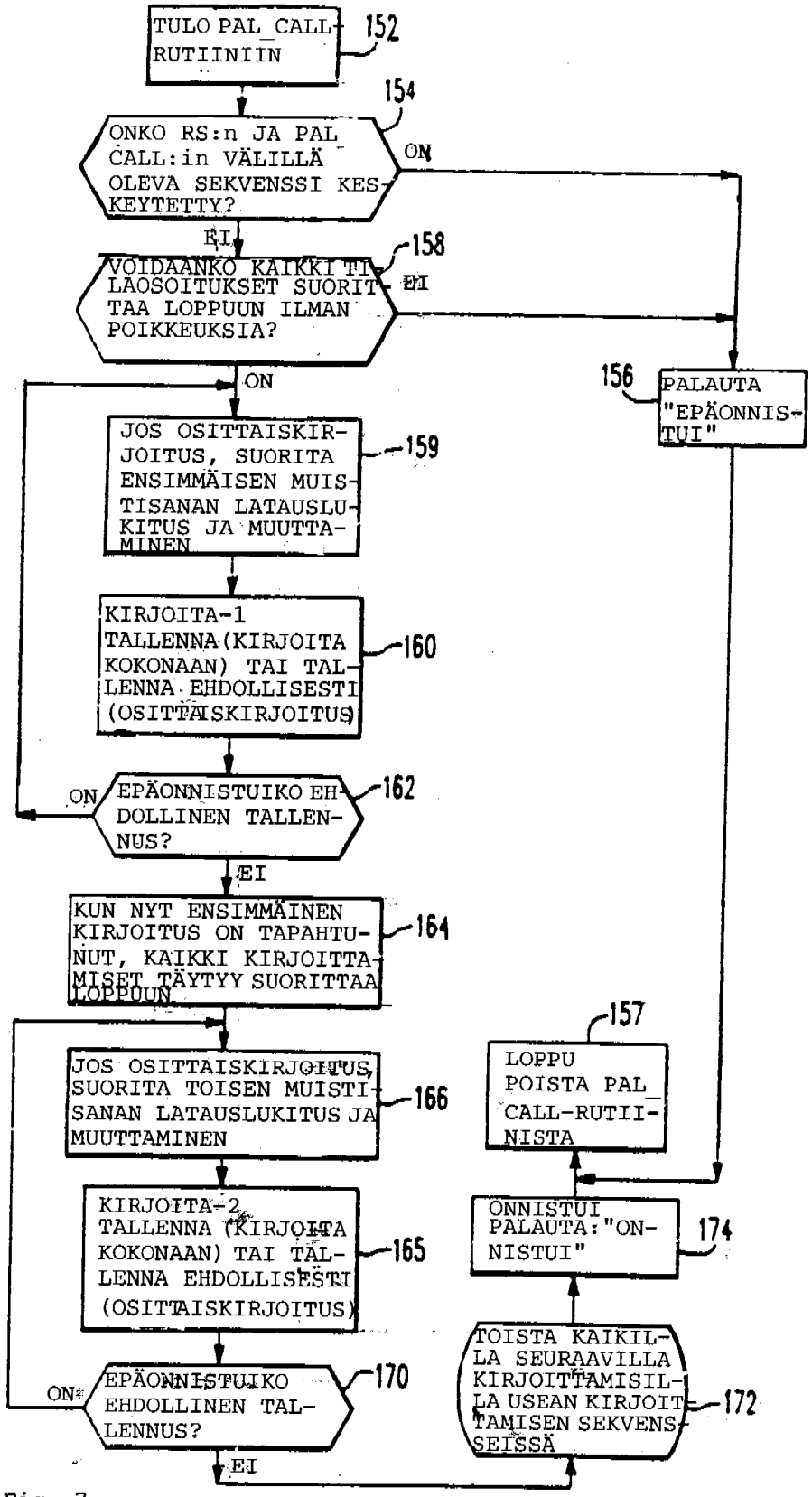
TULO PAL_CALL-RUTIINIIN
152
ONKO RS:n JA PAL_CALL:in VÄLILLÄ OLEVA SEKVENSSI KESKEYTETTY?
154
ON
EI
VOIDAANKO KAIKKI TILAOSOITUKSET SUORITTAA LOPPUUN ILMAN POIKKEUKSIA?
158
EI
ON
156
PALAUTA "EPÄONNISTUI"
JOS OSITTAISKIRJOITUS, SUORITA ENSIMMÄISEN MUISTISANAN LATAUSLUKITUS JA MUUTTAMINEN
159
KIRJOITA-1 TALLENNA (KIRJOITA KOKONAAN) TAI TALLENNA EHDOLLISESTI (OSITTAISKIRJOITUS)
160
ON
EPÄONNISTUIKO EHDOLLINEN TALLENNUS?
162
EI
KUN NYT ENSIMMÄINEN KIRJOITUS ON TAPAHTUNUT, KAIKKI KIRJOITTAMISET TÄYTYY SUORITTAA LOPPUUN
164
JOS OSITTAISKIRJOITUS, SUORITA TOISEN MUISTISANAN LATAUSLUKITUS JA MUUTTAMINEN
166
KIRJOITA-2 TALLENNA (KIRJOITA KOKONAAN) TAI TALLENNA EHDOLLISESTI (OSITTAISKIRJOITUS)
165
ON
EPÄONNISTUIKO EHDOLLINEN TALLENNUS?
170
EI
157
LOPPU POISTA PAL_CALL-RUTIINISTA
ONNISTUI PALAUTA:"ONNISTUI"
174
TOISTA KAIKILLA SEURAAVILLA KIRJOITTAMISILLA USEAN KIRJOITTAMISEN SEKVENSSISSÄ
172

Fig. 7