NORGE

(12) **PATENT**

(19) NO (11) **304459** (13) B1

(51) Int Cl[6] G 06 F 11/00

Patentstyret

(21) Søknadsnr	19924259	(86) Int. inng. dag og søknadsnummer	03.03.1992, PCT/US92/01714
(22) Inng. dag	05.11.1992	(85) Videreføringsdag	05.11.1992
(24) Løpedag	03.03.1992	(30) Prioritet	07.03.1991, US, 666025
(41) Alm. tilgj.	06.01.1993		15.05.1991, US, 700770
(45) Meddelt dato	14.12.1998		

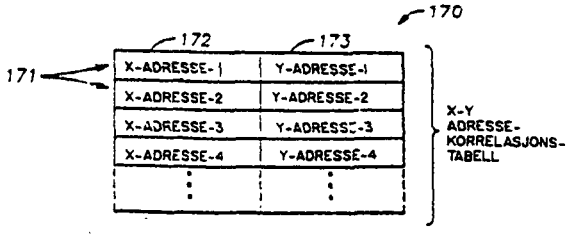
(73) Patenthaver Digital Equipment Corp, 146 Main Street, Maynard, MA 01754, US
(72) Oppfinner Scott G. Robinson, Tyngsboro, MA, US
Richard L. Sites, Boylston, MA, US
(74) Fullmektig Jens F. C. Langfeldt, Bryns Patentkontor AS, 0106 Oslo

(54) Benevnelse **Forbedret feilrapportering for utførelse av oversatt kode**

(56) Anførte publikasjoner Ingen

(57) Sammendrag

I en situasjon der et første dataprogram er oversatt for å frembringe et annet dataprogram blir en feil som oppstår under utførelsen av det annet dataprogram rapportert i konteksten for det første program. Dette blir gjort ved å avbryte utførelsen av det annet program når feilen oppstår, bestemmelse av en første adresse er adressen for den instruksjon i det annet dataprogram som frembragte feilen, bestemmelse på grunnlag av den første adresse av en andre adresse for en instruksjon i det første dataprogram hvorfra instruksjonen i det annet dataprogram ble oversatt og rapportering om at feilen har oppstått og bruk av den andre adresse for å angi at feilen er knyttet til instruksjonen i det første dataprogram. Den andre adresse benyttes fortrinnsvis for å vise til tilbakesporing og symbolsk navninformasjon som ble frembragt da det første dataprogram ble kompilert fra kildekoden. Tilbakesporingsinformasjonen gir linjetallet for den kildekode hvorfra instruksjonen i det første dataprogram ble kompilert og den symbolske navninformasjon gir navnet for en rutine som inneholder instruksjonen i det første program eller en variabel som blir benyttet av instruksjonen.



Foreliggende oppfinnelse vedrører en fremgangsmåte til drift av en digital datamaskin for rapportering av en feil som oppstår under utførelsen av et andre dataprogram som er en oversettelse av et første dataprogram, hvilket første dataprogram er blitt frembragt ved kompilering av et kildeprogram, der det første dataprogrammet innbefatter instruksjoner fra et første instruksjonssett som er basert på en første instruksjonsarkitektur, der hver av instruksjonene i det første dataprogrammet har en respektiv adresse, der det andre dataprogrammet innbefatter instruksjoner fra et andre instruksjonssett basert på en andre instruksjonsarkitektur, der hver av instruksjonene i det andre dataprogrammet har en respektiv adresse, der noen av instruksjonene i det første dataprogrammet hver er blitt oversatt til en respektiv flerhet av instruksjonene i det annet dataprogrammet, idet feilen oppstår ved utførelse av en av instruksjonene i det andre dataprogrammet.

Videre vedrører oppfinnelsen et digitalt datamaskinsystem digitalt datamaskinsystem for rapportering av opptreden av en feil under utførelsen av et andre dataprogram som er en oversettelse av et første dataprogram, der det første dataprogrammet er blitt generert ved kompilering av et kildeprogram, idet det første dataprogrammet innbefatter instruksjoner fra et første instruksjonssett basert på en første instruksjonsarkitektur, idet hver av instruksjonene i det første dataprogrammet har en respektiv adresse, idet det andre dataprogrammet innbefatter instruksjoner fra et andre instruksjonssett basert på en andre instruksjonsarkitektur, der hver av instruksjonene i det andre dataprogrammet har en respektiv adresse, idet noen av instruksjonene i det første dataprogrammet hver er blitt oversatt til en respektiv flerhet av hver av instruksjonene i det andre dataprogrammet, idet feilen bevirkes ved utførelsen av en av instruksjonene i det andre dataprogrammet.

I de tidligere år med programmering av datamaskiner ble instruksjoner for dataprogrammer frembragt på mikrokode-nivået. Med utvikling og vekst av utførelsen av programvare ble flere oppgaver kombinert i enkle komplekse instruksjoner som kunne utføres med datamaskiner som hadde en maskinvarearkitektur beregnet for den grad av kompleksitet instruksjonen hadde.

Økende kompleksitet når det gjelder instruksjonene førte i alminnelighet til økende foredeler når det gjelder pris/egenskaper ved utvikling av omgivelsene for omkostningene for datamaskiners maskinvare og muligheter når det gjelder egenskaper. Som et resultat ble komplekse instruksjonssettkoder (CISC) godt mottatt i vide kretser.

Med økende kompleksitet når det gjelder instruksjonene er det imidlertid blitt vanskeligere å konstruere systemmaskinvare for høyere utførelseshastigheter. Istedet er en redusert instruksjonssettkode (RISC) sammen med korrelert RISC datamaskinvarearkitektur fått anerkjennelse som en mekanisme som fører til et betydelig forbedret forhold mellom systemets pris/egenskaper.

Et RISC-system anvender i alminnelighet enklere grunninstruksjoner for å lede ønskede operasjoner. En enkel RISC-instruksjon vil normalt spesifisere en enkel operasjon med minst en enkel minnetilgang. Videre danner et RISC-system normalt et register for hver grunninstruksjon. Instruksjonene i et RISC-instruksjonssett er således fremdeles på et høyere enn mikrokode.

I det typiske CISC-system kan en enkel instruksjon spesifisere en kompleks sekvens av operasjoner og dette kan kreve mange direkte tilganger til minner. Dermed kan operasjoner som utføres av en CISC-instruksjon kreve flere RISC-instruksjoner.

Et RISC-system er i alminnelighet utført med optimalisert samspill mellom maskinvare og programvare som gir hurtigere systemoperasjoner, bedre samlede systemegenskaper og lavere systemomkostninger i forhold til omkostninger ved tilgjengelig maskinvare og dens yteevne.

En hindring ved omformning fra CISC-systemer til RISC-systemer er at det finnes store programvarebiblioteker som er blitt utviklet for CISC-systemer og som ikke er alminnelig tilgjengelige for RISC-systemer. Når brukeren av et datamaskinsystem velger å gå over til et nytt datamaskinsystem gjelder en av brukerens hovedoverveielser om brukerens bibliotek av anvendelsesprogrammer kan benyttes eller omdannes til bruk i den nye datasystem og hva omkostningene ved utskiftning av biblioteket ville være. For datasystembrukere som ønsker å få bedre pris-/ egenskapsforhold ved RISC-datasystemer er det således viktig at en økonomisk og effektiv mekanisme utvikles for tilpasning, eller "utflytting" av brukerens bibliotek med anvendelsesprogrammer til utførelse på RISC-datasystemet.

Flere valg står til rådighet for brukerens programutflytting. Rekompilering eller rekoding kan anvendes men disse teknikker blir vanligvis benyttet for utflyttingsprogrammer som er skrevet i et høynivåspråk som f.eks. FORTRAN som enten ikke har noen detaljerte maskinavhengigheter eller som har eventuelle eksisterende maskinavhengigheter fjernet ved manuelle programmeringsmodifiksjoner. Ved rekompilering eller rekoding har videre brukeren vanligvis alt ansvar for programmodifikasjonen og garantier for programmets opptreden.

Som et alternativ kan det anvendes oversettelsesprosedyrer, men ulempen ved denne løsning er vanligvis betydelige reduserte programegenskaper. Mer bestemt er tolkeprosedyrer programvare som kjøres på en datamaskin og leser en strøm av subjektinstruksjoner (som godt kan være instruksjoner for en annen type datamaskin) som data og for hver subjektinstruk-

sjon utfører den angitte operasjon. Slike prosedyrer utfører vanligvis 10 til 100 maskininstruksjoner på den ene datamaskinen for å tolke en enkel subjektinstruksjon. Dermed vil tolkeprosedyrer utgjøre en betydelig reduksjon i programegenskapene sammenlignet med direkte utførelse av funksjonelt ekvivalent kode på denne ene datamaskinen. Den mest effektive utflytting innebærer imidlertid kodeoversettelse. Ved kodeoversettelse blir hver instruksjon fra et eksisterende program oversatt til en eller flere instruksjoner i språket for mottakermaskinen. Således vil en oversettelse av CISC-programmer til RISC-programmer og mer generelt en programoversettelse der den oversatte kode har et forholdsvis redusert instruksjonssett kreve "flere" eller "mange" instruksjoner i den oversatte kode for hver instruksjon i den kode som blir oversatt.

Et problem som skyldes oversettelse fra CISC til RISC-kode er at når utførelsen av det oversatte program avbrytes på grunn av en feil er informasjonen om feilen vanligvis basert på tilstanden for RISC-maskinen. Feilsøkeprogrammer som vanligvis er knyttet til høynivå CISC-kildekode antar imidlertid at informasjonen om en feil er basert på tilstanden for en CISC-maskin. Et feilsøkeprogram benytter f.eks. feilsøkeinformasjonen som er frambragt når CISC-kildekoden blir kompilert i en CISC-objektkode. Feilsøkeinformasjonen innbefatter vanligvis informasjon som knytter symbolske (ASCII) navn med nummeriske CISC-program og dataadresser. Feilsøkeprogrammet benytter denne symbolske navneinformasjon til å sette programmet istand til å inspisere innholdene i minnet i CISC-maskinen ved adresser som er spesifisert med de valgte symbolske navn. Vanligvis innbefatter også feilsøkeinformasjonen tilbakesporingsinformasjon som knytter linjetallene i CISC-kildekoden til tilsvarende nummeriske adresser i CISC-objektkoden. Når en feil oppstår under utførelse av CISC-programmet i CISC-maskinen vil operativsystemet i CISC-maskinen lese et maskinvareregister i CISC-maskinen for å bestemme adressen

til den CISC-instruksjon som skaper feilen. Operativsystemet fører denne adresse til et feilrapporteringsprogram som henviser til tilbakesporingsinformasjonen for å gi rapport til programmereren om linjetallet for CISC-kildekoden hvorfra feilen oppsto.

Fordi feilsøkeprogrammene som er knyttet til kompileringen av CISC-kildekoden er basert på CISC-objektkodeadresser, CISC-instruksjoner og maskinvaren i CISC-maskinen frembringer de ikke feilsøking i CISC-kildekoden under utførelse av et oversatt program på en RISC-maskin. Feilsøkingen i CISC-kildekoden under utførelsen av et tilsvarende RISC-objektprogram på en RISC-maskin kunne gjøres med et feilsøkeprogram som har referanse til den informasjon som ble frembragt ved kompilering av CISC-kildeprogrammet til RISC-objektkode men denne teknikk ville kreve omkompilering av CISC-kildekoden. I noen tilfeller har videre CISC-kildekoden gått tapt eller blitt ødelagt slik at rekompilering ikke er mulig.

EP-A-0261247 (Fanuc) omhandler en fremgangsmåte for å feilsøke et høynivåspråk i en digital datamaskin. En første datamaskin lager en omdanningstabell for å avbilde samsvar mellom et høynivå (kompilert) programs linjetall, instruksjonene som befinner seg der, og en tilhørende lavnivås instruksjonsadresse. En andre datamaskin mottar omdanningstabellen og lavnivåspråk og utfører sistnevnte. Omdanningstabellen tillater feil i utførelsen av lavnivåspråket å bli formidlet tilbake i en gitt linje av kildeprogrammet.

Det skal for ordens skyld også krysshenvises til følgende patenter og patentsøknader som er beslektet med foreliggende patentsøknad:

US-patentsøknad 07/666070, med tittelen BRANCH RESOLUTION VIA BACKWARD SYMBOLIC ECECUTION av Richard L. Sites.

US-patent nr. 5287450, med tittelen LOCATING PROGRAM CODE BY SUCCESSIVE CODE EXECUTION AND INTERPRETATION av Richard L. Sites.

US-patentsøknad nr. 07/666210, med tittelen USE OF STACK DEPTH TO IDENTIFY MACHINE CODE MISTAKES av Richard L. Sites.

US-patent nr. 5336766, med tittelen CROSS-IMAGE REFERENCING OF PROGRAM CODE av Richard L. Sites.

US-patent nr. 5301325, med tittelen USE OF STACK DEPTH TO IDENTIFY ARCHITECTURE AND CALLING STANDARD DEPENDENCIES IN MACHINE CODE av Thomas R. Benson.

US-patent nr. 5339238, med tittelen REGISTER USAGE TRACKING TO SUPPORT COMPILED 32-BIT CODE IN 64-BIT ENVIRONMENT av Thomas R. Benson.

US-patent nr. 5307492, med tittelen MAPPING ASSEMBLY LANGUAGE ARGUMENT LIST REFERENCES IN TRANSLATING GODE FOR DIFFERENT MACHINE ARCHITECTURES av Thomas R. Benson.

US-patentsøknad nr. 07/666082, med tittelen TRACKING CONDITION CODES IN TRANSLATING CODE FOR DIFFERENT MACHINE ARCHITECTURES av Thomas R. Benson.

US-patent nr. 5297291, med tittelen LINKING OF PROGRAM UNITS AT PROGRAM ACTIVATION av Daniel L. Murphy.

US-patentsøknad nr. 07/666196, med tittelen AUTOMATIC FLOWGRAPH GENERATION FOR PROGRAM ANALYSIS AND TRANSLATION av Richard L. Sites.

US-patentsøknad nr. 07/666888, med tittelen IMPROVED SOFTWARE DEBUGGING SYSTEM AND METHOD ESPECIALLY ADAPTED FOR CODE DEBUGGING WITHIN A MULTI-ARCHITECTURE ENVIRONMENT av James A. Woolridge, Ronald F. Brender og Henry N. Grieb, III.

US patent nr. 5339423, med tittelen IMPROVED SYSTEM AND METHOD FOR JACKETING CROSS-DOMAIN CALLS IN A MULTI-CODE EXECUTION AND DEBUGGING SYSTEM WITHIN A MULTI-ARCHITECTURE ENVIRONMENT av Ronald F. Brender og Michael V. Iles.

Ifølge oppfinnelsen kjennetegnes den innledningsvis nevnte fremgangsmåten ved at instruksjonene fra et første instruksjonssett kan erstattes av én eller flere instruksjoner fra det andre instruksjonssettet, idet fremgangsmåten omfatter trinnene:

(a) å avbryte utførelsen av det andre dataprogrammet når feilen oppstår,

(b) å bestemme en første adresse i nevnte ene av instruksjonene i det andre dataprogrammet,

(c) å henvise til en adressekorreleringstabell for å bestemme fra den første adressen en andre adresse av en av instruksjonene i det første dataprogrammet fra hvilket den ene av instruksjonene i det andre dataprogrammet ble oversatt,

(d) å henvise til tilbakesporingsinformasjon for å identifisere en del av kildekode i kildeprogrammet fra hvilket den ene av instruksjonene i det første dataprogrammet ble kompilert, og

(e) å rapportere at feilen fant sted og å indikere en del av kildekode i kildeprogrammet hvorfra den ene av nevnte instruksjoner i det første dataprogrammet ble kompilert,

at adressekorreleringstabellen innbefatter innføringer som hver har en adresse til en respektiv instruksjon i det første dataprogrammet og en adresse til en første instruksjon i et sett av instruksjoner i det andre dataprogrammet, idet settet av instruksjoner i det andre dataprogrammet er blitt oversatt fra den respektive instruksjonen i det første dataprogrammet, der ikke alle av adressene i instruksjonene i det første dataprogrammet er blitt inkludert i innføringene i adressekorreleringstabellen,

(f) at en grenseinstruksjonsbitavbildning ble generert under oversettelse av det første dataprogrammet for å oppgi et

andre dataprogram, der grenseinstruksjonsbitavbildningen indikerer instruksjoner i det andre dataprogrammet som begynner sett med instruksjoner oversatt fra respektive instruksjoner i det første dataprogrammet, og

at trinnet (c) med henvisning til adressekorreleringstabellen dessuten innbefatter å avsøke grenseinstruksjonsbit-avbildningen mellom en første posisjon i bitavbildningen som tilsvarer den første adressen og en andre posisjon i bitavbildningen som tilsvarer en adresse i det andre dataprogrammet som er en innføring i adressekorrelerings-tabellen,

å telle indikasjoner i bitavbildningen mellom den første posisjonen og den andre posisjonen for å oppnå en telling av instruksjoner,

å lese fra innføringen i adressekorreleringstabellen en tredje adresse i det første dataprogrammet, og bestemme den andre adressen fra den tredje adressen og tellingen av instruksjoner. Videre kjennetegnes det innledningsvis nevnte datamaskinsystemet ved at det digitale datamaskinsystemet omfatter i kombinasjon:

(a) middel for å avbryte utførelsen av det andre data-programmet når feilen oppstår,

(b) middel for å bestemme en første adresse i den ene av instruksjoner i det andre dataprogrammet,

(c) middel for å henvise til en adressekorreleringstabell for å bestemme fra den første adressen en andre adresse for en av instruksjonene i det første dataprogrammet hvorfra den ene av instruksjonene i det andre dataprogrammet ble over-satt,

(d) middel for å henvise til tilbakesporingsinformasjon for å identifisere en del av en kildekode i kildeprogrammet hvorfra den ene av instruksjonene i det første datapro-grammet ble kompilert og

(e) middel for å rapportere at feilen opptrådte, og indikere nevnte del av kildekode i kildeprogrammet hvorfra en av instruksjonene i det første dataprogrammet ble kompilert,

at adressekorrelasjonstabellen innbefatter innføringer som hver har en adresse for en respektiv instruksjon i det første dataprogrammet og en adresse for en første instruksjon i et sett av instruksjoner i det andre dataprogrammet, idet settet av instruksjoner i det andre dataprogrammet er blitt oversatt fra respektive instruksjon i det første dataprogrammet, der ikke alle av adressene for instruksjonene i det første dataprogrammet er blitt inkludert i innføringene i adressekorreleringstabellen,

at en grenseinstruksjonsbitavbildning ble generert under oversettelsen av det første dataprogrammet for å oppnå et andre dataprogram, idet grenseinstruksjonsbitavbildning indikerer instruksjoner i det andre dataprogrammet som begynner sett av instruksjoner oversatt fra respektive instruksjoner i det første programmet, og

at nevnte middel for å henvise til adressekorrelasjonstabellen dessuten innbefatter middel for å avsøke nevnte grenseinstruksjonsbitavbildning mellom en første posisjon i bitavbildningen som tilsvarer nevnte første adresse og en andre posisjon i bitavbildningen som tilsvarer en adresse i det andre dataprogrammet som er i en innføring i adressekorrelasjonstabellen, middel for å telle indikasjoner i bitavbildningen mellom den første posisjon og den andre posisjon for å oppnå en telling av instruksjoner, middel for å lese fra nevnte innføring i adressekorrelasjonstabellen en tredje adresse i det første dataprogrammet, og middel for å bestemme den andre adresse fra den tredje adressen og tellingen av instruksjoner.

Ytterlligere utførelsesformer av fremgangsmåten og datamaskinsystemet vil fremgå av den etterfølgende beskrivelse og de vedlagte patentkrav.

I en situasjon der et første dataprogram er blitt oversatt for å danne et andre dataprogram tillater i korthet foreliggende oppfinnelse at en feil som oppstår under utførelse av det andre dataprogram blir rapportert i konteksten til det

første dataprogram. Dette gjøres ved å avbryte utførelsen av det andre program når feilen oppstår; ved bestemmelse av en første adresse som er adressen til den instruksjonen i det annet dataprogram som var årsak til feilen; ved på bakgrunn av den første adresse å bestemme en andre adresse for en instruksjon i det første dataprogram hvorfra instruksjonen i det andre dataprogram ble oversatt; og ved rapportering om at feilen har oppstått og ved bruk av den andre adresse for å angi at feilen er knyttet til instruksjonen i det første dataprogram.

Den andre adresse blir fortrinnsvis benyttet til å angi at feilen er knyttet til instruksjoner i det første dataprogram ved henvisning til tilbakesporing og den symbolske navninformasjonen som ble frembragt da det første dataprogram ble kompilert fra kildekoden. Tilbakesporingsinformasjonen blir det henvist til for å angi linjetallet for kildekoden hvorfra instruksjonen i det første dataprogram ble kompilert. Den symbolske navninformasjon blir det henvist til for å angi det symbolske navn som er knyttet til feilen som f. eks. navnet på en rutine som inneholdt instruksjonen i det første program eller en variabel som ble benyttet av instruksjonen.

I en foretrukket utførelsesform blir adressen for instruksjonen i det første program bestemt ut fra adressen til informasjonen i det annet program ved henvisning til en adressekorrelasjonstabell og en grenseinstruksjonsbitavbildning. Adressekorrelasjonstabellen innbefatter innganger som hver har en adresse for en respektiv instruksjon i det første program og en adresse for en første instruksjon i et sett med instruksjoner i det annet program som er blitt oversatt fra den respektive instruksjonen. Den første instruksjonen i et sett med instruksjoner i det annet program som er blitt oversatt fra en respektiv instruksjon i det første program blir betegnet som "grenseinstruksjon". Grenseinstruksjonene i det annet program skiller seg fra de andre instruksjoner i det andre program med flagg i grensein-

struksjonsbitavbildningen. For å finne en andre adresse for en instruksjon i det første program svarende til en første adresse i det annet program blir adressekorrelasjonstabellen avsøkt etter den første adresse for å finne den andre adresse direkte fra tabellen eller ellers for å finne den største adresse i det annet program som er mindre enn den første adresse. Hvis den annen adresse ikke finnes direkte fra tabellen blir grenseinstruksjonsbitavbildningen avsøkt samtidig med telling av antallet grenseinstruksjoner mellom den første adresse og den nevnte største adresse. Den andre adresse blir så funnet ved videreføring av en instruksjonsadresseposisjonsgiver samtidig med spalting av dette samme antall instruksjoner i det første program der det begynnes med instruksjonen ved den adresse i tabellinngangen som også innbefatter den nevnte største adresse.

Oppfinnelsen skal nå nærmere beskrives med hensyn til bl.a. hensikter og fordeler med oppfinnelsen, idet beskrivelsen er gitt med henvisning til de vedlagte tegninger.

Fig. 1 viser et samlet funksjonsblokkskjema som angir den generelle måte hvorpå anvendelsesprogrammene blir (a.) frembragt til utførelse av et første datasystem (med et første instruksjonssett og betegnet som X) og (b.) oversatt for utførelse på et forskjellig datasystem (angitt som Y) med et forholdsmessig redusert instruksjonssett.

Fig. 2 viser et generalisert funksjonsblokkskjema for det foretrukne X-Y oversettelsesprogram og et datasystem for flere formål der X-Y oversettelsesprogrammet utføres for å frembringe, med X-instruksjongranularitet en Y-utførbar anvendelseskode på grunnlag av en inngangs X- anvendelseskode.

Fig. 3 viser et generelt flytskjema for X-Y oversettelsesprogrammet som tidligere er vist på fig. 2,

Fig. 4 viser et funksjonsblokkskjema for Y-datasystemet med Y-anvendelseskoden som ble oppnådd på grunnlag av X-kodeoversettelsen og lastet i Y-datamaskinen for utførelse sammen med et instruksjonsgranularitetsstyreprogram (IGC) som overvåker utførelsen av Y-koden for å garantere at X-instruksjonens granularitet bevares.

Fig. 5A og FB viser flytskjemaer som representerer de foretrukne funksjonstrinn som utføres av IGC-programmet.

Fig. 6 viser et skjema som symboliserer X-Y kodeinstruksjonens oversettelse og forholdet til asynkrone begivenheter som er knyttet til dette.

Fig. 7 er et blokkskjema for maskinvaren i Y-datamaskinen som skal reagere på unntagelser.

Fig. 8 er et flytidagram for de operasjoner som utføres av maskinvaren på fig. 7 og et operativsystems unntagelsesrutine for håndtering som utføres av Y-datamaskinen.

Fig. 9 er et flytskjema for en prosedyre som utføres av en brukermodusunntagelsesrutine for Y-datamaskinen,

Fig. 10 er et skjema for en X-Y adressekorrelasjonstabell som blir dannet under oversettelsen av X-anvendelseskoden til Y-anvendelseskoden og blir benyttet til fremskaffelse av adressen for en X-kodeinstruksjon som har fått gitt adressen til en Y-kodeinstruksjon ved utførelse av feilrapporteringsmetoden ifølge oppfinnelsen.

Fig. 11 er et skjema som representerer overensstemmelsen mellom instruksjonene i X-anvendelseskoden og Y-anvendelseskoden og X-adressene og Y-adressene som er lagret i adressekorrelasjonstabellen på fig. 10.

Fig. 12 er et skjema for en grenseinstruksjonsbitavbildning som ble frembragt under oversettelsen av X-anvendelseskoden til Y-anvendelseskoden og benyttet til utførelse av den foretrukne fremgangsmåte ifølge oppfinnelsen.

Fig. 13 er et flytskjema for en prosedyre til anvendelse av X-Y adressekorrelasjonstabellen på fig. 10 og grenseinstruksjonsbitavbildningen på fig. 12 ved utførelse av den foretrukne fremgangsmåte ifølge oppfinnelsen.

Fig. 14 er et blokkskjema for feilsøkeinformasjon som frembringes under kompileringen av kildekoden for X-anvendelsesprogrammet og

Fig. 15 er et flytskjema for en prosedyre til anvendelse av feilsøkeinformasjonen på fig. 14 for å angi at en feil er knyttet til en av instruksjonene i X-anvendelseskoden.

Siden oppfinnelsen kan underkastes forskjellige modifikasjoner og få alternative former er en spesiell utførelsesform for denne vist som eksempel på tegningene og den vil her bli beskrevet i detalj. Det skal imidlertid påpekes at det ikke er hensikten å begrense oppfinnelsen til den spesielle form som er vist men tvert imot at oppfinnelsen skal dekke alle modifikasjoner, ekvivalenter og alternativer som faller innenfor oppfinnelsens ramme slik den er definert i kravene.

Som vist på fig. 1 er et anvendelsesprogram 10, som er skrevet i kildekode, et av et antall anvendelsesprogrammer som finnes i brukerens programbibliotek til utførelse på et X-datasystem 12. Maskinvarearkitekturen for X- datasystemet 12 er beregnet for å virke med et X- instruksjonssett som anvendes ved dannelsen av en utførbar form for programmet 10 eller andre utførelsesprogrammer i brukerens bibliotek.

For å tilpasse programmet 10 til bruk med et Y-datasystem 20 er det nødvendig at den utførbare form for programmet 10 gis

som en Y-utførbar kode 22 som anvender et Y-instruksjonssett som maskinvarearkitekturen for Y-datasystemet 20 er tilpasset til.

Y-instruksjonssettet benytter i alminnelighet færre grunninstruksjoner enn det X-instruksjonssettet gjør og omformningen av X-kode til Y-kode krever "en for mange" instruksjonsoversettelser. X-instruksjonssettet kan være et CISC-instruksjonssett og Y-instruksjonssettet kan være et RISC-instruksjonssett. F. eks. kan som vist for illustrasjonens skyld på fig. 1, X- systemet anvende VAX-® arkitektur og Y-systemet kan anvende en redusert instruksjonssettarkitektur som kalles RISC-arkitekturen i Digital Equipment Corporation. Begge arkitekturer er innbefattet i utstyr som fremstilles av Digital Equipment Corporation og som foreliggende ansøkning er overdratt til.

Som vist på fig. 1 kan anvendelsesprogrammet 10 utflyttes til den Y-utførbare kode 22 enten i en indirekte bane 24 eller en direkte bane 26. Direkte utflytting oppnås ved bruk av en Y-kompilator 28 og en Y-lenker 30. Den resulterende Y-utførbare kode er betegnet med henvisningstallet 22B.

Hvis en Y-kompilator 28 og en Y-lenker 30 aldri er blitt utviklet eller av andre årsaker ikke er tilgjengelige eller brukeren velger ikke å benytte den direkte utflyttingsbane 26 p.g.a. tilknyttede ulemper kan den indirekte bane 24 benyttes til utflytting av X-anvendelsesprogrammet til Y-systemet for å oppnå både en besparelse i programinvesteringen og i forbedring i systemets egenskaper.

I den indirekte bane blir programmet 10 omformet til utførbar kode 14 for X-datasystemet 12 ved hjelp av en X-kompilator 16 og en X-lenker 18. Resultatet er en X-utførbar kode 14 som kan kjøres på X-datasystemet 12. En X-Y-oversetter 32 oversetter den X-utførbare kode som er betegnet med henvisningstallet 22A. Kodeoversettelsen foregår slik at Y-

koden utføres for å danne X-koderesultatene nøyaktig med instruksjonsgranularitet selvom Y-koden er basert på et redusert instruksjonssett.

Et kodeoversettelsessystem 40 (fig. 2) blir fortrinnsvis anvendt for å implementere oversetteren 32 som er vist på fig. 1. Oversettelsessytemet 40 innbefatter en vanlig datamaskin for flere formål med en prosessor 42, et minnesystem 44, og forskjellige inngangs-/utgangsanordninger (ikke vist) ved hjelp av hvilke X-anvendelseskoden 43 kan føres som innmating for oversettelse.

Oversettelsesresultatene blir frembragt i overensstemmelse med foreliggende oppfinnelse som Y-kode 45 og denne blir ordnet for å bevare sikre garantier for at koden blir oversatt og særlig for å forenkle en garantert bevarelse av X-instruksjonens granularitet når Y-koden virkelig utføres.

Sikre CISC-garantier er eksemplifisert med sikre VAX®-garantier som er beskrevet mer fullstendig i PCT-publikasjon WO 92/15946.

Minnesystemet 44 innbefatter, sammen med andre seksjoner, en vanlig datalagerseksjon 46 og en seksjon 48 hvori datamaskinens operativsystem er lagret. Et grunnelement som anvendes i X-Y-kodeoversettelsen er et oversettelsesprogram 50 som er lagret i en annen minneseksjon. Inngangs- X-koden 43 blir lagret som en X-kodelister 62. For videre å styre sekvenseringen av Y-instruksjoner blir Y- instruksjonsordningskriterier 52 lagret og X-Y- instruksjonskodemønstre 54 blir lagret for å muliggjøre oversettelse av både instruksjonsoperasjonsspesifikatorer og instruksjonsoperandspesifikatorer.

Et generelt flytskjema for oversettelsesprogrammet 50 er vist i sin foretrukne utførelsesform på fig. 3. I blokken 60 blir påhverandrefølgende X-instruksjoner innført sekvensielt

fra den lagrede X-kodeliste 62 for behandling gjennom en programsløyfe 64.

I sløyfen 64 frembringer funksjonsblokken 66 Y- instruksjonsoperasjon og operandspesifikator svarende til den pågående behandlede X-instruksjonen og i overensstemmelse med de lagrede X-Y-kodemønstre 54 (fig. 2). Deretter blir som angitt med funksjonsblokken 68 den resulterende Y-kode ordnet i overensstemmelse med på forhånd bestemte kriterier som forenkler bevarelsen av X- instruksjonens granularitet under den påfølgende egentlige Y-kodeutførelse.

En grafisk gjengivelse av en X-Y-instruksjonsoversettelse er vist på fig. 6.

Hver X-instruksjon har i alminnelighet de elementære oppgaver med å skaffe innganger, modifisere innganger, anbringe resultatene i midlertidige lagre og fremkomme med tilstandsoppdatering for minne-/ og registerlagersteder. Når en X-instruksjon blir oversatt til "mange" Y-instruksjoner er de ordningskriteriene 52 (fig. 2) som anvendes for å organisere Y-instruksjonene fortrinnsvis de som grupperer og ordner Y-instruksjonene i Y-kode for den pågående oversatte X-instruksjon (granulat) som følger:

1. En første gruppe på G1 med instruksjoner i Y-koden er de som får innmatninger og plasserer disse innmatninger i et midlertidig lager.

2. En andre gruppe G2 med instruksjoner i Y-koden er de som arbeider med innmatninger, frembringer modifiserte resultater og lagrer disse resultater i et midlertidig lager.

3. En tredje gruppe G3 med instruksjoner i Y-koden er de som oppdaterer X-tilstanden (minne eller register) og er utsatt for mulige unntagelser (som angitt i det følgende).

4. En fjerde og siste gruppe G4 med instruksjoner i Y-koden er de som oppdaterer X-tilstanden (minne eller register) og er fri for mulige unntagelser.

X-tilstanden innbefatter X-minnetilstand og X- registertilstand. X-minnetilstanden som er representert med henvisningstallet 94 på fig. 4 og X-registertilstanden som er vist med henvisningstallet 97 gjelder henholdsvis minne-/ og registerstruktur i den Y-maskinen som er dedikert til å være X-kodedefinerte lagersteder. X-minne-/ og registertilstandene kan også sies å være minnetilstand og registertilstand som er synlige for X-arkitekturen.

Som et resultat av implementeringen av de foretrukne Y-instruksjonsorganiserende kriterier i X-Y-kodeoversettelsen blir bevarelse X-instruksjonens granularitet lettet av de følgende årsaker:

1. Hvis en asynkron hendelse opptrer under Y-kodeutførelsen på et hvilket som helst tidspunkt under utførelsen av de første to grupper G1 og G2 av Y-instruksjoner som vist med en pil 71 i skjemamet på fig. 6 blir X-instruksjonen granularitet opprettholdt ved å muliggjøre behandlingen av den asynkrone hendelse og understøtte Y-instruksjonens teller PC til neste støtte Y-instruksjon (Y_0 på fig. 6) som er en granulær grense for en X-instruksjon.

Ved å avbryte utførelsen av den pågående Y-kodesekvens for et gjenforsøk blir et mulig brudd i X-instruksjonens granularitet unngått fordi en mulig svikt i tilstandstilgangen blir unngått i enhver gruppe G3 instruksjon i den pågående Y-kodesekvens. Som et resultat av Y-kodeorganisasjonen blir imidlertid bare midlertidige lagersteder slettet og X-instruksjonens granularitet blir bevart siden utførelsen av det pågående Y-kodegranulat er blitt forsinket inntil den kan bli behandlet med instruksjonsgranularitet etter behandlingen av den asynkrone hendelse.

2. Hvis en asynkron hendelse opptrer etterat Y- instruksjonene i gruppene G1 og G2 er blitt utført og hvis det ikke finnes noen Y-instruksjoner i gruppe G3 eller om hendelsen opptrer etter utførelse av alle Y-instruksjoner som er innbefattet i gruppen G3 som angitt med en pil 73 på fig. 6 kan behandlingen av den asynkrone hendelse kort forsinkes mens gruppe G4 instruksjonene blir utført med forhåndskjennskap om at ingen tilstandsunntagelser er mulige. Igjen vil utførelsen ha X-instruksjonens granularitet.

I det tilfellet der flere Y-instruksjoner er innbefattet i gruppen G3 (ikke vist på fig. 6) og en asynkron hendelse opptrer etterat en Y-instruksjon er blitt utført men minst en Y-instruksjon i gruppe G3 er tilbake for å bli utført, kan det anordnes minneatomitet i den løpende Y-kodegranul for å bevare X- instruksjonens-granularitet. Minneatomitet krever hovedsaklig at alle minnetilganger for X-instruksjonen synes å oppstå eller synes ikke å oppstå og er nødvendig for X-instruksjonens granularitet. I den foretrukne utførelse der enkle én-skrive X-instruksjoner håndteres, blir X-minneatomiteten og X- instruksjonsgranulariteten oppnådd samtidig.

Med ny henvisning til fig. 3 vil blokken 70, så snart funksjonsblokken 68 beordrer Y-instruksjonskoden som beskrevet, bestemme om den pågående Y-instruksjon er en grense eller "X-granulat" markør for den instruksjon hvorfra den ble avledet. Ja- og neibiter som blir påvist for påhverandre følgende behandlede Y-instruksjoner blir registrert i en X- grenseinstruksjons bitavbildning med funksjonsblokken 72.

En utprøvningsblokk 74 kontrollerer om flere X-instruksjoner er listet opp for å bli oversettelsesbehandlet. Hvis så er tilfelle blir sløyfen 64 gjentatt. Når alle X-instruksjoner er blitt oversatt blir sløyferundgangen avsluttet og den samlede Y-kode gjort tilgjengelig som utgang, som angitt med

funksjonsblokken 76. De følgende to spesielle eksempler for CISC instruksjon fra VAX® arkitekturen viser oversettelses-funksjonen ifølge oppfinnelsen:

Eksempel 1:

```
INCL  (R1)+
```

Denne instruksjon legger til 1 til langordet (4-byter eller 32 biter) for minne hvis adresse befinner seg i R1. R1 blir deretter inkrementert for å peke mot det neste langord. Den følgende RISC (Y) kode ble frembragt av oversetteren som arbeider med de beskrevne organiseringskriterier:

```
ldl     r23, (r1)      ;  last innehodlet i minnet ved
                          adressen (R1) til R23
addl    r23,#1,r24     ;  inkrementer R23 med 1 og legg
                          inn R24
stl     r24, (r1)      ;  lagre R24 i minnet ved
                          adressen (R1)
lda     r1,4(r1)       ;  inkrementer R1 med fire for å
                          peke mot neste langord
```

Eksempel 2:

```
MOVL  (R1)+,  (R2)+
```

Denne instruksjon fører langordet fra minnet som er adressert med innholdet av R1 til det langord som er adressert med innholdet av R2. Både R1 og R2 blir inkrementert for å peke mot det neste langord etter detsom det opprinnelig ble pekt mot da instruksjonen begynte. Den følgende RISC (Y) kode ble frembragt av oversetteren som arbeider med de beskrevne organiseringskriterier:

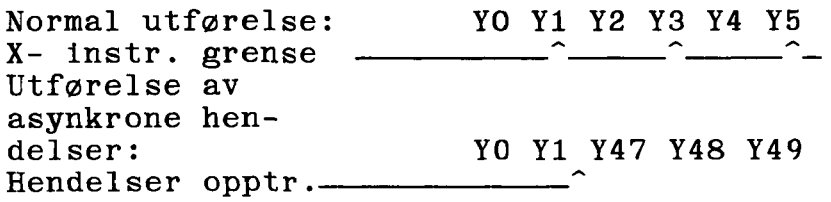
```
ldl     r23, (r1)      ; last inneholdet av minnet
                         ved (R1) til R23
stl     r23, (r2)      ; lagre inneholdet av R23 i
                         minnet ved lagersted (R2)
lda     r1,4(r1)       ; inkrementer R1 med fire for
                         å peke mot neste langord
lda     r2,4(r2)       ; inkrementer R2 med fire for
                         å peke mot neste langord
```

For utførelse av resultant Y-koden med garantert granularitet for X-instruksjonen er det anordnet en Y-prosessor 80 svarende til Y-datasystemet 20 (fig. 1) til utførelse av resultant Y-koden med X-instruksjonsgranularitet som garanteres å bli bevart. Vanlige datainngangs-/utgangsanordninger 82 og en intervallklokke 84 er koblet til Y-prosessoren 80 og fra tid til annen frembringer disse anordninger avbrytelser som danner asynkrone hendelser som krever en midlertidig avledning av prosessorens operasjon fra Y-kodeutførelsen. (Fig. 4) Uten garantert granularitet for X-instruksjonen vil prosessoravledningen som oppstår på grunn av disse eller andre avbrytelser være istand til å bevirke et sammenbrudd av X-instruksjonens granularitet ved Y kodeutførelsen. Dette finner vanligvis sted på grunn av mulige inngangsforandringer i Y-instruksjonen etter avbrytelsen men før minnetilstandsforandringene er blitt gjort av den pågående Y-instruksjon.

Som vist på fig. 4 representerer blokken 86 innmatningen av den frembragte Y-kode fra en inngangsanordning til en seksjon 88 i et minnesystem 90 som er koblet til Y-prosessoren 80 og en blokk 87 representerer datautmatninger som frembringes for utgangsanordningene som et resultat av Y-kodeutførelsen. Minnesystemet 90 innbefatter også en vanlig dataseksjon 92 og en valig operativsystemseksjon 94.

Et styreprogram (IGC) 96 for instruksjonsgranularitet er lagret i en annen minneseksjon er betegnet med samme henvisningstall. Operasjonen av IGC programmet 96 ved utførelsen av Y-koden er mer fullstendig gjengitt på flytskjemaene som er vist på fig. 5A og 5B.

IGC programmet 96 (fig. 5A) begynner som angitt ved 98 med frembringelsen av en asynkron hendelse. Generelt er en asynkron hendelse definert som en avledning fra Y-instruksjonsstrømmen p.g.a. avbrytelser som potensielt kunne frembrige X-tilstandsforandringer som er synlige for den oversatte X-kode. Det skal igjen vises til fig. 6 når det gjelder en skjematisk gjengivelse av forholdet mellom asynkrone hendelser og et X-granulat av Y-instruksjoner. Det følgende viser som eksempel interaksjoner mellom en asynkron hendelse og Y-instruksjonsstrømmen;



IGC programmet 96 må rette inn Y-instruksjonsadressen mot Y1 (hvis Y2 har en mulig unntagelse som definert i det følgende) eller mot Y3 (ved utførelse av Y2 hvis Y2 ikke har noen mulige unntagelser) før utførelse av eventuelle andre forandringer i X-tilstanden.

Med ny henvisning til flytskjemaet på fig. 5A er et midlertidig hold innført ved behandlingen av den asynkroen hendelse med funksjonsblokken 100 og minneadressen for Y- instruksjonen (betegnet som PC-AE) som blir behandlet ved tidspunktet for den asynkrone hendelse blir registrert i en funksjonsblokk 102.

Deretter blir den bitavbildning det tidligere er vist til kontrollert av blokken 104 for å bestemme om Y-instruksjonen

PC-AE er en X-instruksjonsgrense. Hvis den er det vil prøveblokken 106 lede IGC programmet over banen 107 til blokken 108 som tillater gjenopptagelse av behandlingen av den asynkrone hendelse uten å bryte X-kodeinstruksjonens granularitet.

Hvis Y instruksjonen PC-AE ikke er en X-instruksjonsgrense vil funksjonsblokken 110 rette inn Y-instruksjonens teller PC med den neste foregående eller neste oppkommende foroverrettede Y-instruksjon som er en X-instruksjonsgrense. En programbane 111 blir så fulgt av IGC programmet 96 til blokken 108 for behandling av den asynkrone hendelse som tidligere beskrevet, igjen uten å bryte X-kodeinstruksjonens granularitet. I dette tilfelle har den asynkrone hendelse oppstått på et tidspunkt da ikke alle Y-instruksjoner er blitt utført innenfor et X-instruksjonsgranulat og bevarelsen av X-instruksjonens granularitet blir oppnådd ved utførelse av programblokken 110 som forklart mer i detalj på fig. 5B. Markørene "A" og "B" på fig. 5A angir hvor detaljblokkene på fig. 5B passer inn i flytskjemamet på fig. 5A.

På fig. 5B utføres foroverrettet avsøkning av Y-instruksjonene av funksjonsblokken 112 i en programsløyfe 113 for å finne den neste Y-instruksjon som er en X- instruksjonsgrense. En prøveblokk 114 prøver hver foroverrettet avsøkende Y-instruksjon for å bestemme om behandlingen av avbruddet før utførelse av de gjenværende Y-instruksjoner kunne frembringe Y-kodeutførelsesresultat som er forskjellig fra det resultat som ville ha blitt frembragt om den tilsvarende X-kode hadde blitt utført med påtrykning av den samme asynkrone hendelse.

Ved utførelsen av hver foroverrettet Y-instruksjonsprøve bestemmer prøveblokken 114 fortrinnsvis om en unntagelsestilstand kan frembringes ved en forsøkt utførelse av Y-instruksjonen hvis den asynkrone hendelse ble tillatt behandlet og utførelsen av Y-kodesekvensen deretter blir tilbakekalt.

Generelt sett har en instruksjon en unntagelse hvis den eventuelt ikke kan fullføres. De følgende er de generelle klasser for unntagelser som, hvis de blir identifisert for en foroverrettet Y-instruksjon, frembringer et Y-kodeavbrudd for den neste tidligere Y-instruksjonen som er en X-grense:

1.) Minnestyrerunntagelser som f. eks. krenking av tilgangsstyring eller sidefeil.

2.) Aritmetiske unntagelser så som overflytfeil for flytpunkt, eller deling med null feil.

3.) Instruksjonsunntagelser så som illegale operasjonskoder eller bruddpunktsoperasjonskoder.

I den foretrukne utførelsesform for oppfinnelsen er en liste over anvendbare unntagelser for den kode som oversettes anbragt i et lager som gir tilgang for IGC programmet under utførelsen. Utførelsesbestemmelser gjøres således ved henvisning av hver foroveravsøkt Y-instruksjon mot den lagrede unntagelsesliste.

Det følgende er et eksempel på et unntagelsestilfelle:

Et delt minnelagersted blir inkrementert med en asynkron hendelse.

INCL @#33 inkrementerer lagerstedet 33.
RISC instruksjonene er:

```
LDL     R23,33(R31)    ;  last R23 fra lagerstedet 33
LDA     R23,1(R2)      ;  legg til 1 til innholdet av
                          R23
STL     R23,33(R31)    ;  lagre R23 resultatet tilbake
                          ved lagerstedet 33
```

Hvis den synkrone hendelse finner sted mellom LDA og STL-instruksjonene og behandlingen av den asynkrone hendelse innkremeterer lagerstedet 33 vil det inkrement som gjøres under hendelsen gå tapt når strømmen gjenopptas ved STL-instruksjonen.

Det følgende er et eksempel på et tilfelle uten unntagelse: Instruksjonen INCL R1 inkrementerer R1. RISC-instruksjonen er:

```
LDAR1,1(R1)                ; inkrement R1
```

Det er her ingen muligheter for en unntagelse.

Hvis under prøvingen av påhverandrefølgende Y-instruksjoner ved foroverrettet avsøkning og hvis alle avsøkte Y-instruksjoner ikke viser noen unntagelser (blokk 116) blir de gjenværende Y-instruksjoner utført før behandlingen av den asynkrone hendelse muliggjøres av blokken 108 (fig. 5A) uten å bryte X-instruksjonens granularitet som tidligere beskrevet. Hvis på den annen side en foroverrettet avsøkt Y-instruksjon viser en unntagelse under prøven med blokken 118, vil funksjonsblokken 118 støtte opp Y-programtelleren til neste oppstøttende Y-instruksjon som er en X- instruksjonsgrense og behandlingen av den asynkrone hendelse blir igjen startet av blokken 108 (fig. 5A) uten å bryte X- instruksjonens granularitet. Til og med muligheten for et brudd i X-instruksjonens granularitet blir på denne måte unngått og X-instruksjonens granularitet er dermed garantert.

Ved oppsummering av tilfellet med oversettelse av en instruksjon som ikke har noe skrevet om tilstand eller bare har et enkelt innrettet fullt langord eller kvadord skrevet finnes det ingen byte- eller uinnrettede- skriveproblemer og ingen stengeproblemer. Det eneste problem er det det som går ut på å ta enten hele sekvensen eller intet av den sekvens som synes å begynne utførelse. Oversettelsen er begrenset

slik at i tilfellet CISC-til-RISC oversettelse gjør de første og andre grupper av RISC-instruksjonene alt arbeidet for CISC-instruksjonen bortsett fra oppdatering av minnet og/eller registertilstand, den tredje gruppe RISC-instruksjoner innbefatter null eller en RISC-lagerinstruksjon til et fullt innrettet langord eller kvadord og den fjerde gruppe av RISC-instruksjoner innbefatter bare enkle registeroverføringer.

En oversatt sekvens som blir avbrutt før den andre instruksjonsgruppe er fullført blir tvunget til å starte på nytt ved begynnelsen av sekvensen ved å benytte en bittabell som markerer begynnelsen av koden for hver oversatt CISC-instruksjon. En oversatt sekvens som blir avbrutt etter den andre instruksjonsgruppe er fullført men før den tredje instruksjonsgruppe er fullført blir tvunget til å fullføre den tredje instruksjonsgruppe med enkle registeroverføringer.

Slutteffekten er at sekvensen enten utføres fra begynnelse til slutt uten noen annen oversatt sekvens i midten eller den bringes til opphør før fullførelse av gruppe 2 og prøves på nytt fra begynnelsen på et senere tidspunkt. Dette gjelder også en 1- eller 2- byte ikkesperret skriving på en enkeltprosessormaskin. Den oversatte sekvens innbefatter en tilstands les-modifiser-skrivesekvens og den enkle skriving er gruppe 3 instruksjonen.

Foreliggende oppfinnelse angår mer bestemt en fremgangsmåte til drift av Y-datasystemet 20 på fig. 1 under utførelsen av et Y-dataprogram som er en oversettelse av et X- dataprogram. Med andre ord gjelder foreliggende oppfinnelse særlig rapportering av en feil når et kildekodeprogram 10 utflyttes til den Y-utførebare kode langs den indirekte banen 24. I denne situasjonen vil feilsøkeinformasjonen som vanligvis frembringes av X-kompilatoren 16 innbefatte adressene til instruksjonene i den X-utførbare kode. Som beskrevet videre i det følgende er imidlertid unntagelsesmaskinvaren og den Y-

utførbare kode 22A organisert og arbeider på Y-kodeadresser. Den foreliggende oppfinnelse danner et ledd mellom en første adresse for en Y kodeinstruksjon som fører til en feil og en andre adresse for en X-kodeinstruksjon som Y-kodeinstruksjonen var oversatt fra. Dette ledd benyttes i en feilhåndteringsprosedyre som begynner med et unntagelsessignal i maskinvaren for Y-prosessoren (80 på fig. 4) og avslutter med rapport om feilen i tilknytning til en angivelse av X-kodeinstruksjonen.

Det skal nå vises til fig. 7 som gjengir den maskinvare som generelt er betegnet med 130 i Y-prosessoren (80 på fig. 4) som svarer på et respektivt unntagelsessignal på en hvilken som helst av en flerhet av unntagelsesinnganger 131. Unntagelsessignalene kan f. eks. innbefatte paritetsfeilsignaler som angir maskinvarefeil, minneunntagelser som er resultat av instruksjoner om tilgang til minner og aritmetiske unntagelser fra en artimetisk logikkenhet 132 som er resultater av aritmetiske instruksjoner.

Foreliggende oppfinnelse angår særlig rapporten av anvendelsesprogramfeil som skaper unntagelser så som aritmetiske unntagelser fra den artitmetiske logikkenhet 132. Aritmetiske unntagelser innbefatter f. eks. heltalls overflyt eller flytpunk overflyt der resultatet er for stort til å bli lagret i et på forhånd bestemt heltalls- eller flytpunkt (flytende komma) format; heltall eller flytpunkt delt med null der den tilførte divisor er null; og flytpunktet underflyt der resultatet er for lite til å bli uttrykt av et på forhånd bestemt flytpunkt-format.

Den aritmetiske logikkenhet 132 føler nærvær av de aritmetiske unntagelsestilstander når som helst en aritmetisk operasjon blir utført. Noen av Y-kodeinstruksjonene kobler ut eller hindrer unntagelsestilstanden i å frembringe et unntagelsessignal og andre Y-kodeinstruksjoner skaper en aritmetisk unntagelsestilstand for å frembringe et unntagel-

sessignal. Denne funksjon er vist på fig. 7 med en OG port 133.

Y-prosessoren er istand til å svare på bare en enkelt av unntagelsesinnmatningene på et hvilket som helst tidspunkt. For å kunne håndtere situasjonen med flere samtidige unntagelsessignaler blir unntagelsesinnmatningene 131 mottatt i en prioritetskoder 134 som velger ut et enkelt av signalene. Unntagelsessignalet som blir gitt prioritet er angitt med en kode som frembringes av prioritetskoderen 134.

Unntagelsesinnmatningene 131 blir satt sammen i en ELLER port 135 for å frembringe et unntagelsessignal som blir gjenkjent av styrelogikk 136. Styrelogikken 136 bringer midlertidig utførelsen av Y-kodeinstruksjonene til opphører og styrer samlingen av data som gjelder unntagelsen fra de respektive registere eller overføringsporter 137, 138, 139. Data som gjelder unntagelsen innbefatter "unntagelses PC" som er Y-kodeadressen (PC-AE) for den instruksjon som skaper unntagelsen. "Unntagelse PSL" som er programtilstandslangordet som angir prosessortilstanden på tidspunktet for utførelse av den instruksjon som dannet unntagelsen og den unntagelsestypeinformasjon som er definert av unntagelsestype. Når det eksempelvis gjelder en aritmetisk unntagelse, avgir den aritmetiske logikkenhet 132 en kode som angir den aritmetiske unntagelsestype. Som svar på koden for den unntagelse som har prioritet velger en multiplekser 140 unntagelsestypeinformasjonen for den unntagelse som ble valgt av prioritetskoderen 134. Styrelogikken 136 fører til at unntagelsesinformasjonen blir plassert sekvensielt på en databuss 141 og stakklageret i et kjernestakklager. Som svar på koden for den unntagelse som er valgt av prioritetskoderen 134 blir så en respektiv unntagelsesvektor lest fra en tidklareringstabell 142 for unntagelser og lastet i programtelleren 143. Styrelogikken 136 begynner så utførelsen av instruksjonene og starter med unntagelsesvektoren i programtelleren 143.

Det vises nå til fig. 8 der det er gjengitt et flytskjema som generelt er betegnet med 150 og som videre viser arbeidsmåten for maskinvaren på fig. 7 i forbindelse med en operativsystemrutine for håndtering av unntagelser. Som beskrevet ovenfor når det gjelder fig. 7 vil maskinvaren på fig. 7, når en unntagelse opptrer la utførelsen av instruksjonene vente som angitt i trinnet 151 på fig. 8 hvoretter unntagelses PC, unntagelses PSL og unntagelsestypeinformasjoner plasseres på kjernemodusstakklageret som angitt i trinn 152 hvoretter programtelleren lastes med den respektive unntagelsesvektor fra tidsklareringstabellen som angitt i trinnet 153. I dette tilfellet er unntagelsesvektoren en unntagelsesbehandler i "kjernemodusen" i operativsystemet.

Operativsystemets rutine for behandling av unntagelse begynner i trinn 154 med kontroll om unntagelsen er en "brukermodus" unntagelse som skal håndteres av et brukerprogram istedet for bare av operativsystemet. En maskinvarefeil som f.eks. oppstår i svaret på utførelse av en instruksjon vil normalt ikke være en brukermodusfeil og operativsystemet vil behandle unntagelsen i trinn 155. For en brukermodusunntagelse i trinn 156 blir utførelsesinformasjonen som ble anbragt på kjernemodusstakklageret og innholdet av flerbruksregisterene i Y prosessoren satt inn i brukermodusstakklageret. Sluttelig, i trinn 157 hopper utførelsen til brukermodusunntagelsesrutine som er definert for håndtering av unntagelsen.

Der skal nå vises til fig. 9 der det er gjengitt et flytskjema 160 for brukermodusunntagelsesrutinen. I det første trinn 161 blir X-maskintilstandens informasjoner samlet. Med andre ord blir en konsistent modell av X-maskinens tilstand rekonstruert i Y-maskinens registere og brukerminne. Som forklart nærmere i det følgende i forbindelse med et spesielt eksempel på Y-koden for INCB -(R1) X-kodeinstruksjonen blir X-kodeinstruksjonene oversatt til Y-kode på en slik måte at en konsistent modell av X- maskintilstanden kan rekonstrueres

i trinn 161. Deretter, i trinn 162 blir Y-adressen for Y-kodeinstruksjonen med unntagelsen tatt vare på i minnelagersted Y_ADDR slik at den er tilgjengelig senere for feilrapportering i overensstemmelse med fremgangsmåten ifølge oppfinnelsen.

I trinnene 163 og 164 blir posisjonsangiveren PC-AE rettet inn ved en X-instruksjons granulære grense som beskrevet ovenfor i forbindelse med trinnene 106 og 110 på fig. 5A. Dette kan muliggjøre gjenvinning og fortsatt programutførelse fra den innrettede Y-adresse posisjonsgiver PC-AE når det er et program til en tilstandshåndtering. Tilgjengeligheten for en tilstandshåndtering for unntagelsestypen blir kontrollert i trinn 165. Når en slik tilstandshåndtering er tilgjengelig vil utførelsen avgrenes til tilstandshåndtering i trinn 166. Ellers vil i trinn 167 brukermodusunntagelsesrutinen skaffe adressen til X-kodeinstruksjonen hvorfra Y-kodeinstruksjonen som fører til feilen ble oversatt. Den foretrukne fremgangsmåte til frembringelse av denne X- instruksjonsadresse svarende til Y_ADDR er beskrevet videre i det følgende under henvisning til fig. 13.

I trinn 168 blir feilen rapportert sammen med dens tilknytning til den X-instruksjon som har X-adresse svarende til Y-ADDR. Den foretrukne feilrapporteringsprosedyre er videre beskrevet nedenfor med henvisning til fig. 15. Stort sett kan et Y-kodeprogram til utførelse av trinnet 168 frembringes ved oversettelse av et kjent X-kodeprogram for rapportering av feil som opptrer når X-kode anvendelsesprogrammer blir utført på et X-kode datasystem og ved en mindre modifisering av det oversatte program slik at det refererer adressene til X-instruksjonen som adressen til den instruksjon som bevirket feilen og slik at enhver annen X-maskins tilstandsinformasjon som det refereres til er den X-maskinens tilstandsinformasjon som ble samlet i trinnet 161. Derfor er en viktig fordel ved foreliggende oppfinnelse at den muliggjør eksisterende X-kode feilsøkerrutiner å bli oversatt

og benyttet med små modifikasjoner til feilsøking i kjøring av oversatte anvendelsesprogrammer i Y-kode datasystemer.

Den foretrukne utførelsesform for oppfinnelsen benytter en X-Y-adressekorrelasjonstabell for å skaffe den X- instruksjonsadressen som tilsvarer Y_ADDR. Som vist på fig. 10 innbefatter X-Y-adressekorreleringstabellen 170 innmatninger 171 som hver har en adresse 172 for en respektiv instruksjon i X-kodeanvendelsesprogrammet og en adresse 173 for en tilsvarene instruksjon i Y-kodeanvendelsesprogrammet som er blitt oversatt fra X-kodeanvendelsesprogrammet. Hver adresse 172, 173 er f.eks. et 32 bit langord. Når en instruksjon for en respektiv X-adresse i tabellen blir oversatt til en serie av Y-kodeinstruksjoner i Y-kodeanvendelsesprogram er en tilsvarende Y-kodeinstruksjonsadresse i tabellinngangen adressen til den første instruksjon i serien med instruksjoner.

Det er å foretrekke at ikke alle adresser til instruksjonene i X-kodeanvendelsesprogrammet er innbefattet i X-Y-adressekorreleringstabellen 170. Tabellen ville ellers bli meget stor og ville oppta en god del av lagerplass og i for høy grad øke søketiden for tabellen. I tillegg til at den anvendes for rapportering av feil i overensstemmelse med foreliggende oppfinnelse kan tabellen 170 benyttes for tilbakeføring fra en X-kodetolker ved utførelse av en delvis oversettelse av et X-kodeanvendelsesprogram. Selvom en fullstendig oversettelse alltid er ønskelig er det ikke alltid mulig å foreta en fullstendig oversettelse på grunn av vanskeligheten med å finne alt om X-koden i et X-kodeanvendelsesprogram. I denne situasjonen kan X-kodetolkeren på en effektiv måte gå tilbake til utførelse av Y kode i oversettelsen sålenge X-kodeadressene i tabellen 170 er inngangspunktene for grunnblokker i X-koden i X-kodeanvendelsesprogrammet. Hver grunnblokk er en sekvens av tilstøtende instruksjoner som har et enkelt kjent innngangspunkt ved begynnelsen av grunnblokken. Utførelse overføres

bare til begynnelsene av grunnblokkene og utførelse overføres bare fra avslutningene av grunnblokkene. Delvise oversettelser og driften av en tolker i en slik situasjon er videre beskrevet i nevnte kryssrelaterte US-patentsøknad nnr. 07/666196 og som det vises til på ny som en referanse.

Det skal nå vises til fig. 11 der det er gjengitt et eksempel på endel av et X-kode anvendelsesprogram betegnet som "hoved" og som er blitt tegnet i form av et flytskjema 180 for å vise programmets grunnblokker. Den første grunnblokk består av X-INSTRUKSJON-1. Den andre grunnblokk består av X-INSTRUKSJON-2, X-INSTRUKSJON-3 og X-INSTRUKSJON-4. Den tredje grunnblokk består av X-INSTRUKSJON-6. På fig. 11 er overensstemmelsen mellom X- kodeinstruksjonene og deres tilsvarende Y-kodeinstruksjoner vist som Y-INSTRUKSJON-1 til Y-INSTRUKSJON-14 og X-kode-/ og Y-kodeadressene i X-Y-adressekorreleringstabellen 170 på fig. 11 er også vist.

Den foretrukne utførelsesform for oppfinnelsen benytter også grenseinstruksbitavbildning til frembringelse av de X-instruksjonsadresser som tilsvarer Y_ADDR i den typiske situasjonen der Y_ADDR ikke blir funnet i en innmatning til X-Y-adressekorreleringstabellen. Det skal nå vises til fig. 12 der det foretrukne format for grenseinstruksjonsbitavbildningen 185 er vist. Enbits flagg så som flagget 186 er pakket i langord som f. eks.. langordet 187. Hvert flagg svarer til en Y-kodeinstruksjon og flagget blir stilt av oversetteren (32 på fig. 1) når den tilsvarende Y-kodeinstruksjon er en grenseinstruksjon.

Flaggene i grenseinstruksjonsbitavbildningen 185 blir også satt for å angi "kunstig" granulære grenser i VAX® X-koden. Ønskeligheten om å angi kunstige granulære grenser oppstår ved oversettelse av VAX® X-kodeinstruksjoner med "nøyaktige" feller for RISC Y-kodeinstruksjoner som ikke har nøyaktige feller. En instruksjon har en nøyaktig felle når utførelsen av instruksjonen frembringer samme resultat når en unntagel-

sestilstand oppstår (annet enn frembringelse av et unntagelsessignal) uansett om unntagelsen er åpnet eller inne. I det foretrukne RISC-datasystem har aritmetiske instruksjoner ikke nøyaktige feller fordi mer kompliserte kretser og utførelsessykler er nødvendige for å fullføre resultatet av de aritmetiske instruksjoner etterat unntagelsestilstander er påvist, men hvis programmereren ønsker å emulere nøyaktige feller kan dette gjøres med ett par instruksjoner. Operasjonen blir først kodet med en instruksjon som har feller sperret og deretter blir operasjonen kodet en andre gang med en instruksjon som har feller åpnet. For å bevare X-instruksjonens granularitet for RISC Y-koden blir en "tøm felle" (DRAINT) instruksjon kodet etterat instruksjonen har fellene åpnet. P.g.a. at RISC-datasystemet ellers kunne utføre mer enn én instruksjon, samtidig innbefattende den instruksjon som har feller sperret sammen med en instruksjon som følger den instruksjon som har fellene åpnet, sikrer DRAINT-instruksjonen at når en felle oppstår kan X-maskinens tilstand lett samles i trinnet 161 på fig. 9 for å danne en konsistent X-maskintilstand svarende til utførelsen av den X-kodeinstruksjon som er blitt oversatt.

Et bestemt eksempel på oversettelsen av en X-kodeinstruksjon til en serie Y-kodeinstruksjoner innbefattende en Y-kodeinstruksjon med feller åpnet er gitt nedenfor:

Tittel: Inkrement Byte Eksempel Kode

```
;
; De følgende kodeduplikater:  INCB-(R1)
; med VAX® atomitet over bruk av en PAL rutine for å lagre ;
resultatene.
;
;======>Granulær VAX® grense
      BIS        R31,R31,R24   ;fjern interferens fra telling
topp: RS         R31           ;sett flagg som benyttes av PAL
                                rutinen til
prøve
                                ;for avbrudd eller unntagelser
      LDQU       R16,-1(R1)    ;mest sannsynlig uinnrettet
      LDA        R17,-1(R1)    ;adresse benyttet for
                               innretning av byter
      EXTBL      R16,R17,R18   ;byte til 0000000A
```

```
        SLL         R18,#56,R18  ;byte til A0000000
        BIS         R31,#1,R19   ;posisjonskonstant 1
        SLL         R19,#56,R19  ;10000000
        ADDQ         R18,R19,R16  ;inkrementert til B000000
                                  (ingen felle)
        SRL         R16,#56,R16  ;byte til 0000000B
        KALL_PAL TRN_STB         ;lagre byte i R16 ved adresse i
                                  R17
        BEQ     RO,INTERFERERE   ;feil, gjør denne til en
                                 foroverrettet gren
til
                                 ;narr grenforutsigelser
        SUBQ    R1,#1,R1          ;utfør auto-dekrement (ingen
                                  felle)
;======>Kunstig Granulær Grense
        ADDQ/V R18,R19,R16       ;omgjør operasjonen for
                                innstilling av overflytfelle
                                 ;hvis noen
        DRAINT                   ;vent på felle om noen
;======>Granular VAX® Grense

;     .
;     .
;     .

;   kontroller for å se om sløyfegjentagelsene er for mange

INTERFERER;        ADDQ     R24,#1,R24     ;;inkrement
interferens
                                           ;;telle
                   CMPULE   R24,#255,RO    ;;vært her for A
mens?
                   BNE      RO,TOP         ;;for, prøv igjen
                   FEIL_KONTROLL           ;;avbryt...

          .AVSLUTNING
```

I dette eksempel opptrer ADDQ-instruksjonen en første gang med feller sperret for å frembringe et fullstendig aritmetisk resultat og en andre gang nær enden av serien av Y-kodeinstruksjoner med feller åpnet, fulgt av en DRAINT-instruksjon. Adressen til instruksjonen med feller åpnet er angitt ovenfor som en "kunstig granulær grense" og dens tilhørende flagg blir stilt i grenseinstruksjonens bitavbildningen. Dette fører til en forbedret egenskap ved håndtering av avbrytelser fordi resultat av X- kodeinstruksjonen (andre enn en mulig felle) er fullstendig ved den kunstige granulære grense. Når en avbrytelse finner sted like foran fellen kan derfor PC_AE rettes inn mot den kunstige granulare grense i trinn 110 på

fig. 5A for å bevare X-kodens instruksjonsgranularitet uten fullstendig gjenopprettelse av grenseinstruksjonen ved begynnelse av den Y-kode som er oversatt fra X-kodeinstruksjonen.

Det skal nå vises til fig. 13 der det er gjengitt et flytskjema 190 for den foretrukne prosedyre til bruk av X-Y-adressekorrelasjonstabellen (170 på fig. 10) og grenseinstruksjonsbitavbildningen (185 på fig. 12) for å finne adressen til den X-kodeinstruksjon som tilsvarer Y-kodeadressen Y_ADDR. I det første trinn 191 blir X-Y-adressekorrelasjonstabellen avsøkt for å finne den største Y-kodeadresse i tabellen (Y_ADDR_MAX) som er mindre eller lik Y_ADDR. Deretter blir i trinn 192 innholdet i et minnelagersted X-adressen satt lik X-kodeadressen fra tabellinngangen innbefattende Y_ADDR_MAX. Deretter i trinn 193 blir prosedyren på fig. 13 avsluttet med den ønskede X- kodeadresse i X_ADRESS når Y_ADDR_MAX er lik Y_ADDR. I trinn 194 vil ellers innholdet i et minnelagersted COUNT bli slettet for å begynne telling av antallet av Y-kodegrenseinstruksjoner mellom Y_ADDR og Y_ADDR_MAX. Dette gjøres samtidig med avsøkning av grenseinstruksjonsbitavbildningen.

I trinn 195 blir flagget i bitavbildningen ved stillingen som tilsvarer (Y_ADDR) kontrollert. Når flagget er satt i trinn 196 vil oppkodingen av den følgende instruksjon (ved Y_ADDR + 4 fordi instruksjonene er 4 byter lange i Y-instruksjonsarkitekturen) bli inspisert for å finne ut om det er en DRAINT instruksjon. Når dette er tilfelle er instruksjonen ved Y_ADDR en instruksjon med feller åpnet og det tilsvarende flagg som er blitt satt ved 195 er et flagg for en kunstig granulær grense. Hvis flagget er satt, men ikke er for en kunstig granulær grense, blir det i trinn 197 COUNT inkrementert med én.

I trinn 198 blir Y_ADDR dekrementert med fire byter svarende til lengden av en RISC Y-kodeinstruksjon. For å prøve om

avsøkningen tilbake til Y_ADDR_MAX er avsluttet blir Y_ADDR sammenlignet med Y_ADDR_MAX og når Y_ADDR er større Y_ADDR_-MAX fortsetter avsøkning i trinn 195. I trinn 200 vil ellers verdien av COUNT bli sammenlignet med null. Når tellingen er null vil verdien av X-ADRESSENE være den ønskede adresse for X-kodeinstruksjonen og utførelsene fortsetter i trinn 168 på fig. 9. I trinn 201 vil X_ADRESSEN ellers bli benyttet som en dekodeposisjonsgiver for X-kodeinstruksjon og blir ført frem ved spalting av en X-instruksjon og i trinnet 202 COUNT blir den dekrementert med 1. Trinnene 200, 201 og 202 blir gjentatt inntil den tellede antall av X-instruksjoner blir spaltet og X-ADRESSEN blir ført frem til den ønskede adresse.

Det skal nå vises til fig. 14 der det er gjengitt et blokkskjema for feilsøkeinformasjon 220 som ble frembragt da X-kodeanvendelsesprogrammet ble kompilert. For å muliggjøre rapportering av en kildekodes linjetall svarende til en gitt X-kode med instruksjonsadresse innbefatter feilsøkeinformasjonen tilbakesporingsinformasjon 221. Tilbakesporingsinformasjonen innbefatter f. eks. en liste med registreringer av det X-kodeadresseområdet som er knyttet til hvert linjetall i kildekoden. X-kodeadresseområdet angir adresseområdet for de X-kodeinstruksjoner som ble kompilert fra linjen med kildekode. Noen kompilatorer kompilerer separate kildekodemoduler som blir lenket sammen av lenkeren (18) på fig. 1. I dette tilfellet frembringer kompilatoren en modultabell (ikke vist) som har registreringer innbefattende modulnavn og tilknyttede X- kodeadresseområder. I dette tilfellet frembringer kompilatoren også en separat tabell med tilbakesporingsinformasjon for hver modul.

Feilsøkeinformasjonen 220 innbefatter videre symbolske navninformasjoner 222 der navnene som oppstår i anvendelses-kildekoden knyttes til minneadresser og stakklagersteder det vises til i X-kodeanvendelsesprogrammet. Den symbolske navninformasjon 222 innbefatter f. eks. registreringer av det

X-kodeadresseområdet som er knyttet til hver globale variable i kildekoden og registreringer av X-kodeadresseområdet som er knyttet til hver rutine i kildekoden. I tillegg kan hver rutine være tilknyttet et antall registrering av stakklagerrammens forskyvning for hver lokale variable i rutinen. Registreringene av den symbolske navninformasjon 220 innbefatter posisjonsgivere mot strenger 223 som inneholder de symbolske navn ASCII kode.

Det skal nå vises til fig. 15 der den foretrukne prosedyre til bruk av feilsøkeinformasjonen på fig. 14 for rapportering av en feil og dens tilknytning til X-ADRESSEN for X-kodeinstruksjonen som frembringer feilen er vist. I trinn 231 blir informasjonen om registreringer av tilbakesporing (221 på fig. 14) avsøkt etter en registrering med et adresseområde som innbefatter X-ADRESSE. Når en slik registrering finnes blir i trinn 232 kildekodens linjenummer i registreringen rapportert til programmereren eller brukeren. I trinn 233 blir typen av feilen bestemt på grunnlag av unntagelsestypeinformasjonen og rapportert til programmereren eller brukeren. Deretter blir i trinn 234 registreringene av symbolsk navninformasjon avsøkt etter et symbolsk navn som er knyttet til feilen. Registreringene av rutinene blir f. eks. avsøkt etter en registrering med et adresseområde innbefattende X-ADRESSE. Hvis et tilsluttet rutinenavn finnes blir det rapport i trinn 235 sammen med sin type og sin ASCII streng. Når det symbolske navn er en delrutine fortsetter i trinn 236 feilrapporteringen til trinn 237. I trinn 237 blir stakklageret i X-maskinens tilstandsinformasjon (samlet i trinn 161 på fig. 9) gjennomgått for å frembringe en sporing av et kjede med kallrutiner som ender med delrutine som blir rapportert til programmereren eller brukeren.

I trinnene 234 og 235 kan globale og lokale variable som er knyttet til en feil også bli funnet og rapportert. Når det gjelder en overflyt eller underflyt aritmetisk feil f. eks. kan X-kodeinstruksjonen ved X-ADRESSE og noen påfølgende

instruksjoner undersøkes i et forsøk på å identifisere en resultatadresse for en global eller lokal variabel hvori resultatet av den aritmetiske operasjon er lagret. I tilfelle med en dele-med-null feil kan noen få forutgående X-kodeinstruksjoner undersøkes i et forsøk på å identifisere kildeadressen for en global eller lokal variabel der divisoren har sin opprinnelse.

Eksempler på feilmeldinger som kunne bli rapportert ved bruk av prosedyren på fig. 15 innbefatter følgende:

LINJE 200 HELTALLS OVERFLYT I DELRUTINE "DIVIDENDE" NÅR DEN KALLES MED RUTINE "HOVED" VED LINJE 50, RESULTAT LAGRET I LOKAL VARIABEL "TOTAL".

LINJE 500 FLYTENDE, DELE MED 0 I DELRUTINE "LAGER" NÅR DEN KALLES MED RUTINE "HOVED" VED LINJE 75, DIVISOR FRA GLOBAL VARIABEL "DELER".

Under henvisning til det ovenstående er det beskrevet en fremgangsmåte og et system til rapportering av en feil som oppstår under utførelsen av et Y-kodeanvendelsesprogram i konteksten for et X-kodeanvendelsesprogram hvorfra Y-kodeanvendelsesprogrammet ble oversatt. Oppfinnelsen frembringer adressen i X-koden for en X-kodeinstruksjon svarende til den Y-kodeinstruksjon som frembragte feilen sammen med konsistent informasjon om den tilstand som en X-kodemaskin vil ha etter fullførelse av utførelsen av X-kodeinstruksjonen. Derfor kan feilen rapporteres med feilsøkerrutiner som er oversatt fra X-kode til Y-kode og som benytter feilsøkeinformasjonen frembragt under kompilering av kildekoden for X-kodeanvendelsesprogrammet.

P a t e n t k r a v

1.

Fremgangsmåte til drift av en digital datamaskin for rapportering av en feil som oppstår under utførelsen av et andre dataprogram (22A) som er en oversettelse av et første dataprogram (14), hvilket første dataprogram (14) er blitt frembragt ved kompilering av et kildeprogram (10), der det første dataprogrammet (14) innbefatter instruksjoner fra et første instruksjonssett som er basert på en første instruksjonsarkitektur, der hver av instruksjonene i det første dataprogrammet har en respektiv adresse, der det andre dataprogrammet innbefatter instruksjoner fra et andre instruksjonssett basert på en andre instruksjonsarkitektur, der hver av instruksjonene i det andre dataprogrammet har en respektiv adresse, der noen av instruksjonene i det første dataprogrammet (14) hver er blitt oversatt til en respektiv flerhet av instruksjonene i det annet dataprogrammet (22A), idet feilen oppstår ved utførelse av en av instruksjonene i det andre dataprogrammet, k a r a k t e r i s e r t v e d at instruksjonene fra et første instruksjonssett kan erstattes av én eller flere instruksjoner fra det andre instruksjonssettet, idet fremgangsmåten omfatter trinnene:

(a) å avbryte utførelsen av det andre dataprogrammet (22A) når feilen oppstår,

(b) å bestemme en første adresse (173) i nevnte ene av instruksjonene i det andre dataprogrammet (22A),

(c) å henvise til en adressekorreleringstabell (170) for å bestemme fra den første adressen (173) en andre adresse (172) av en av instruksjonene (171) i det første dataprogrammet (14) fra hvilket den ene av instruksjonene (171) i det andre dataprogrammet ble oversatt,

(d) å henvise til tilbakesporingsinformasjon (221) for å identifisere en del av kildekode i kildeprogrammet (10) fra hvilket den ene av instruksjonene (171) i det første dataprogrammet (14) ble kompilert, og

(e) å rapportere at feilen fant sted og å indikere en del av kildekode i kildeprogrammet hvorfra den ene av nevnte instruksjoner (171) i det første dataprogrammet (14) ble kompilert,

at adressekorreleringstabellen (170) innbefatter innføringer som hver har en adresse til en respektiv instruksjon i det første dataprogrammet og en adresse til en første instruksjon i et sett av instruksjoner i det andre dataprogrammet, idet settet av instruksjoner i det andre dataprogrammet er blitt oversatt fra den respektive instruksjonen i det første dataprogrammet, der ikke alle av adressene i instruksjonene i det første dataprogrammet er blitt inkludert i innføringene i adressekorreleringstabellen,

(f) at en grenseinstruksjonsbitavbildning (185) ble generert under oversettelse av det første dataprogrammet (14) for å oppgi et andre dataprogram (22A), der grenseinstruksjonsbitavbildningen (185) indikerer instruksjoner i det andre dataprogrammet som begynner sett med instruksjoner oversatt fra respektive instruksjoner i det første dataprogrammet, og

at trinnet (c) med henvisning til adressekorreleringstabellen (170) dessuten innbefatter å avsøke grenseinstruksjonsbitavbildningen (185) mellom en første posisjon i bitavbildningen som tilsvarer den første adressen (173) og en andre posisjon i bitavbildningen som tilsvarer en adresse i det andre dataprogrammet som er en innføring i adressekorreleringstabellen,

å telle indikasjoner i bitavbildningen mellom den første posisjonen og den andre posisjonen for å oppnå en telling av instruksjoner,

å lese fra innføringen i adressekorreleringstabellen en tredje adresse i det første dataprogrammet, og bestemme den andre adressen fra den tredje adressen og tellingen av instruksjoner.

2.

Fremgangsmåte som angitt i krav 1, k a r a k t e r i s e r t v e d at symbolsk navninformasjon (222) ble

generert når det første dataprogrammet (14) ble kompilert, idet den symbolske navninformasjonen (222) knytter symbolske navn i kildeprogrammet (10) til adresser i det første dataprogrammet (14), og å henvise til den symbolske navninformasjonen for å identifisere et symbolsk navn som er knyttet til feilen og rapportere det symbolske navnet sammen med feilen.

3.

Fremgangsmåte som angitt i krav 2, karakterisert ved at det symbolske navnet er navnet på en rutine i kildeprogrammet (10) som innbefatter kildekode hvorfra den ene av instruksjonene i det første dataprogrammet (14) ble kompilert.

4.

Fremgangsmåte som angitt i krav 1, karakterisert ved at den andre adressen bestemmes fra den tredje adressen ved spalting av et valgt antall av instruksjoner som begynner fra den tredje adressen, idet det valgte antallet av instruksjoner er basert på tellingen av instruksjoner.

5.

Fremgangsmåte som angitt i krav 1, karakterisert ved at grenseinstruksjonsbitavbildningen (185) også innbefatter indikasjoner av instruksjoner i det andre dataprogrammet (22A) som har feller klargjort, og der indikasjonen av instruksjoner i det andre dataprogrammet (22) som har feller klargjort ikke telles under tellingsindikasjonene i bitavbildningen mellom den første og den andre posisjon for å oppnå tellingen av instruksjoner.

6.

Digitalt datamaskinsystem for rapportering av opptreden av en feil under utførelsen av et andre dataprogram (22A) som er en oversettelse av et første dataprogram (14), der det første

dataprogrammet (14) er blitt generert ved kompilering av et kildeprogram (10), idet det første dataprogrammet (14) innbefatter instruksjoner fra et første instruksjonssett basert på en første instruksjonsarkitektur, idet hver av instruksjonene i det første dataprogrammet har en respektiv adresse, idet det andre dataprogrammet innbefatter instruksjoner fra et andre instruksjonssett basert på en andre instruksjonsarkitektur, der hver av instruksjonene i det andre dataprogrammet (22) har en respektiv adresse, idet noen av instruksjonene i det første dataprogrammet (14) hver er blitt oversatt til en respektiv flerhet av hver av instruksjonene i det andre dataprogrammet, idet feilen bevirkes ved utførelsen av en av instruksjonene i det andre dataprogrammet, k a r a k t e r i s e r t v e d at det digitale datamaskinsystemet omfatter i kombinasjon:

(a) middel for å avbryte utførelsen av det andre dataprogrammet (22A) når feilen oppstår,

(b) middel for å bestemme en første adresse (173) i den ene av instruksjoner i det andre dataprogrammet (22A),

(c) middel for å henvise til en adressekorreleringstabell (170) for å bestemme fra den første adressen (173) en andre adresse (172) for en av instruksjonene (171) i det første dataprogrammet (14) hvorfra den ene av instruksjonene (171) i det andre dataprogrammet ble oversatt,

(d) middel for å henvise til tilbakesporingsinformasjon (221) for å identifisere en del av en kildekode i kildeprogrammet (10) hvorfra den ene av instruksjonene i det første dataprogrammet (14) ble kompilert og

(e) middel for å rapportere at feilen opptrådte, og indikere nevnte del av kildekode i kildeprogrammet hvorfra en av instruksjonene (171) i det første dataprogrammet (14) ble kompilert,

at adressekorrelasjonstabellen (170) innbefatter innføringer som hver har en adresse for en respektiv instruksjon i det første dataprogrammet og en adresse for en første instruksjon i et sett av instruksjoner i det andre dataprogrammet, idet settet av instruksjoner i det andre dataprogrammet er blitt

oversatt fra respektive instruksjon i det første dataprogrammet, der ikke alle av adressene for instruksjonene i det første dataprogrammet er blitt inkludert i innføringene i adressekorreleringstabellen,

at en grenseinstruksjonsbitavbildning (185) ble generert under oversettelsen av det første dataprogrammet (14) for å oppnå et andre dataprogram (22A), idet grenseinstruksjonsbitavbildning (185) indikerer instruksjoner i det andre dataprogrammet som begynner sett av instruksjoner oversatt fra respektive instruksjoner i det første programmet, og

at nevnte middel for å henvise til adressekorrelasjonstabellen (170) dessuten innbefatter middel for å avsøke nevnte grenseinstruksjonsbitavbildning (185) mellom en første posisjon i bitavbildningen som tilsvarer nevnte første adresse og en andre posisjon i bitavbildningen som tilsvarer en adresse i det andre dataprogrammet som er i en innføring i adressekorrelasjonstabellen, middel for å telle indikasjoner i bitavbildningen mellom den første posisjon og den andre posisjon for å oppnå en telling av instruksjoner, middel for å lese fra nevnte innføring i adressekorrelasjonstabellen en tredje adresse i det første dataprogrammet, og middel for å bestemme den andre adresse fra den tredje adressen og tellingen av instruksjoner.

7.

Digitalt datamaskinsystem som angitt i krav 6, k a r a k t e r i s e r t v e d dessuten å omfatte middel for å henvise til symbolsk navninformasjon (222) for å identifisere et symbolsk navn tilhørende feilen, og rapportere det symbolske navnet sammen med feilen.

8.

Digitalt datamaskinsystem som angitt i krav 6, k a r a k t e r i s e r t v e d at nevnte middel for å bestemme den andre adressen fra den tredje adressen inneholder middel for å spalte et valgt antall av instruksjoner som begynner fra

den tredje adresse, idet det valgte antall av instruksjoner velges basert på tellingen av instruksjoner.

9.

Digitalt datamaskinsystem som angitt i krav 6, k a r a k t e r i s e r t v e d at grenseinstruksjonsbitavbildningen (185) også innbefatter indikasjoner om instruksjoner i det andre dataprogrammet (22A) som har feller klargjort, og der midlet for telling ikke teller indikasjoner om instruksjoner i det andre dataprogrammet (22A) som har feller klargjort.

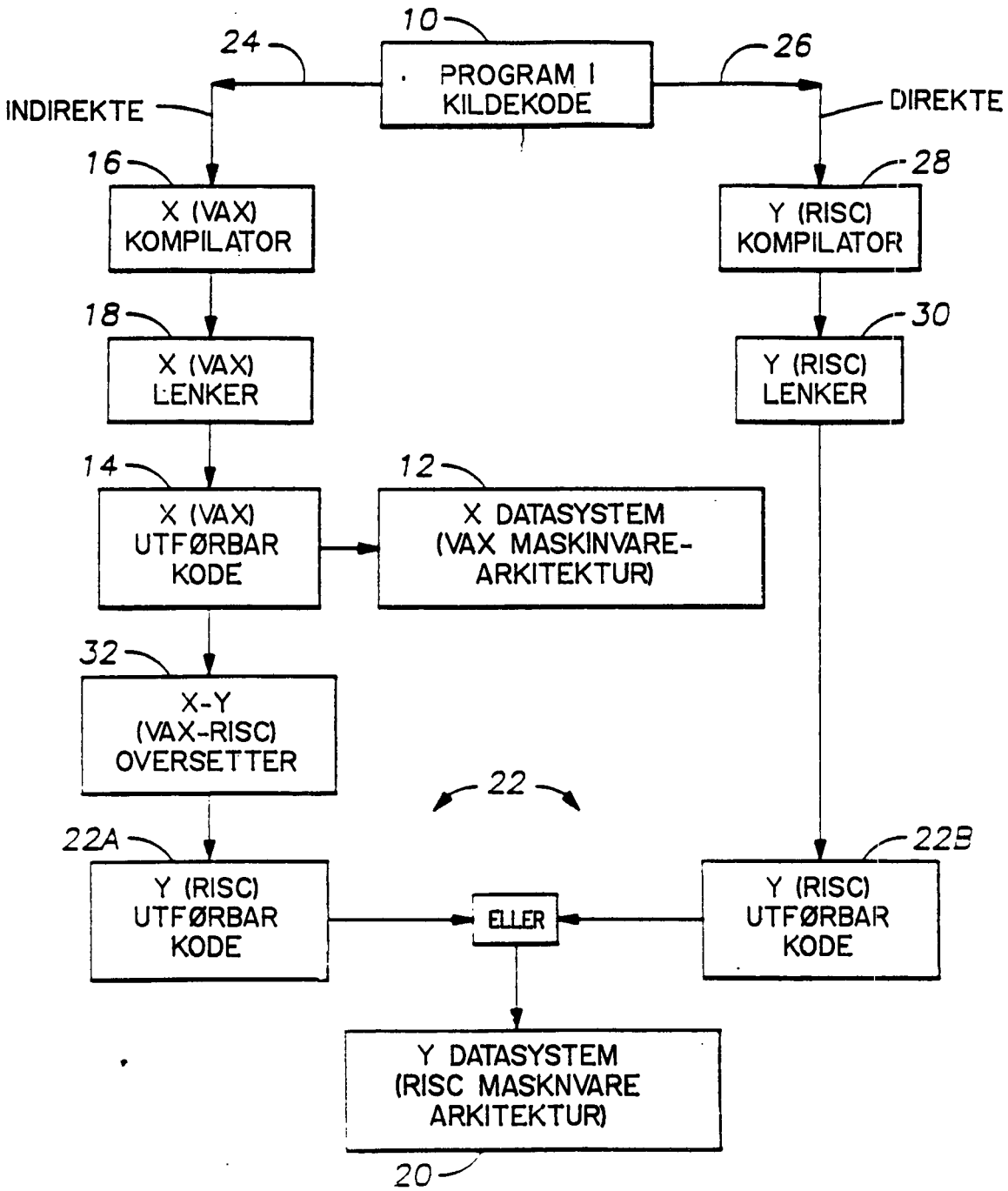

10
PROGRAM I KILDEKODE
24
INDIREKTE
26
DIREKTE
16
X (VAX) KOMPILATOR
28
Y (RISC) KOMPILATOR
18
X (VAX) LENKER
30
Y (RISC) LENKER
14
X (VAX) UTFØRBAR KODE
12
X DATASYSTEM (VAX MASKINVARE-ARKITEKTUR)
32
X-Y (VAX-RISC) OVERSETTER
22
22A
Y (RISC) UTFØRBAR KODE
ELLER
22B
Y (RISC) UTFØRBAR KODE
Y DATASYSTEM (RISC MASKNVARE ARKITEKTUR)
20

FIG. 1

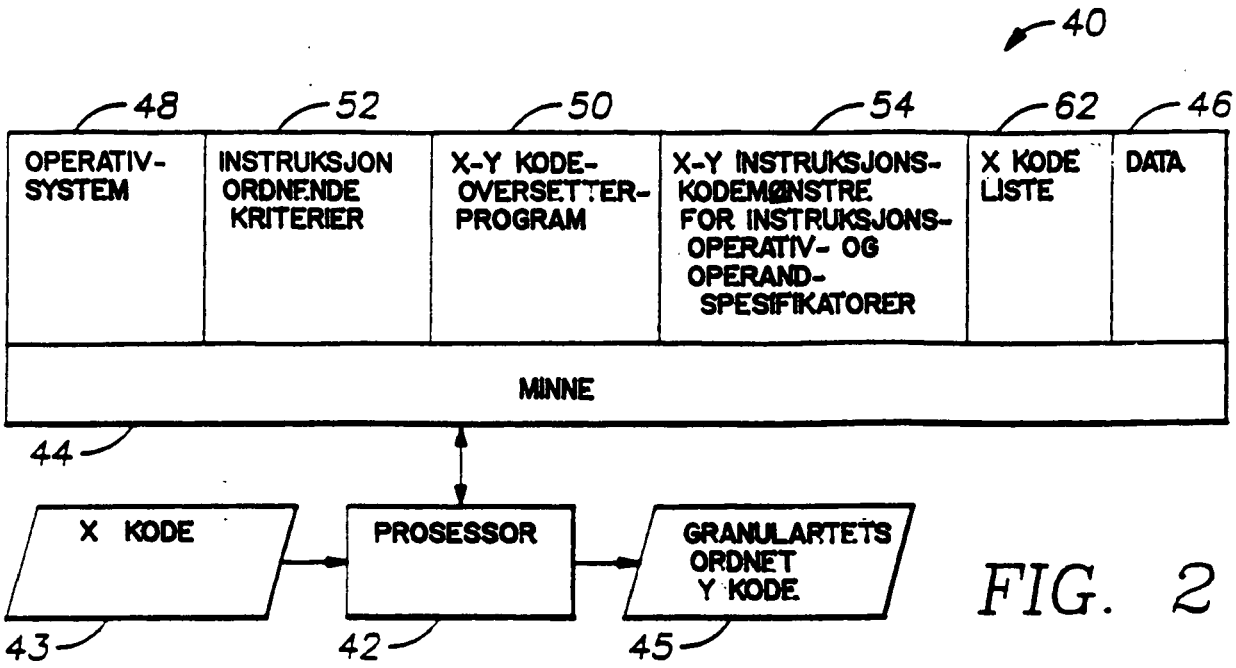

40
48
52
50
54
62
46
OPERATIV-SYSTEM
INSTRUKSJON ORDNENDE KRITERIER
X-Y KODE-OVERSETTER-PROGRAM
X-Y INSTRUKSJONS-KODEMØNSTRE FOR INSTRUKSJONS-OPERATIV- OG OPERAND-SPESIFIKATORER
X KODE LISTE
DATA
MINNE
44
X KODE
PROSESSOR
GRANULARTETS ORDNET Y KODE
43
42
45

FIG. 2

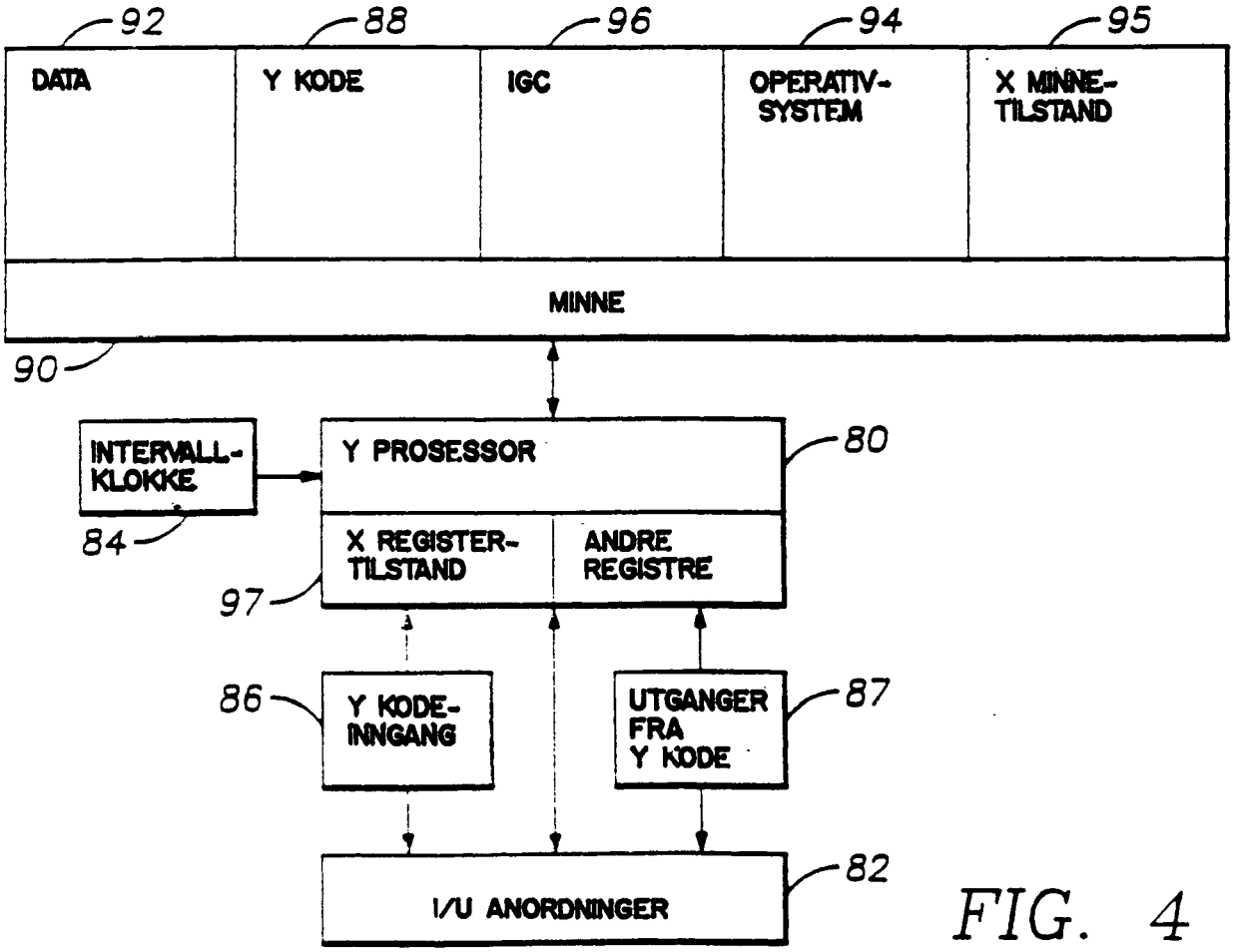

92
88
96
94
95
DATA
Y KODE
IGC
OPERATIV-SYSTEM
X MINNE-TILSTAND
MINNE
90
INTERVALL-KLOKKE
Y PROSESSOR
80
84
X REGISTER-TILSTAND
ANDRE REGISTRE
97
86
Y KODE-INNGANG
UTGANGER FRA Y KODE
87
I/U ANORDNINGER
82

FIG. 4

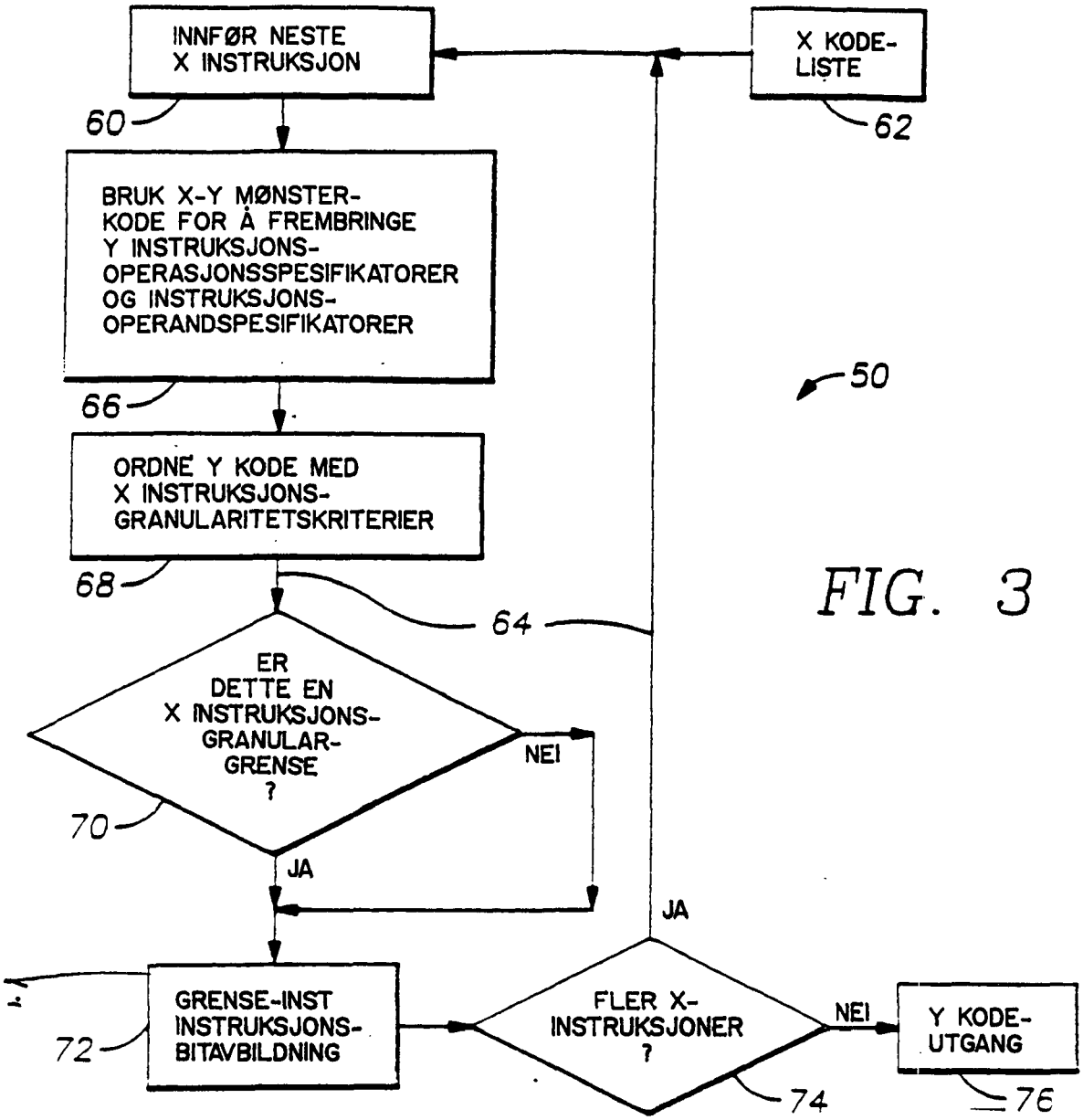

INNFØR NESTE X INSTRUKSJON
60
X KODE-LISTE
62
BRUK X-Y MØNSTER-KODE FOR Å FREMBRINGE Y INSTRUKSJONS-OPERASJONSSPESIFIKATORER OG INSTRUKSJONS-OPERANDSPESIFIKATORER
66
50
ORDNE Y KODE MED X INSTRUKSJONS-GRANULARITETSKRITERIER
68
64
ER DETTE EN X INSTRUKSJONS-GRANULAR-GRENSE ?
70
NEI
JA
JA
GRENSE-INST INSTRUKSJONS-BITAVBILDNING
72
FLER X-INSTRUKSJONER ?
74
NEI
Y KODE-UTGANG
76

FIG. 3

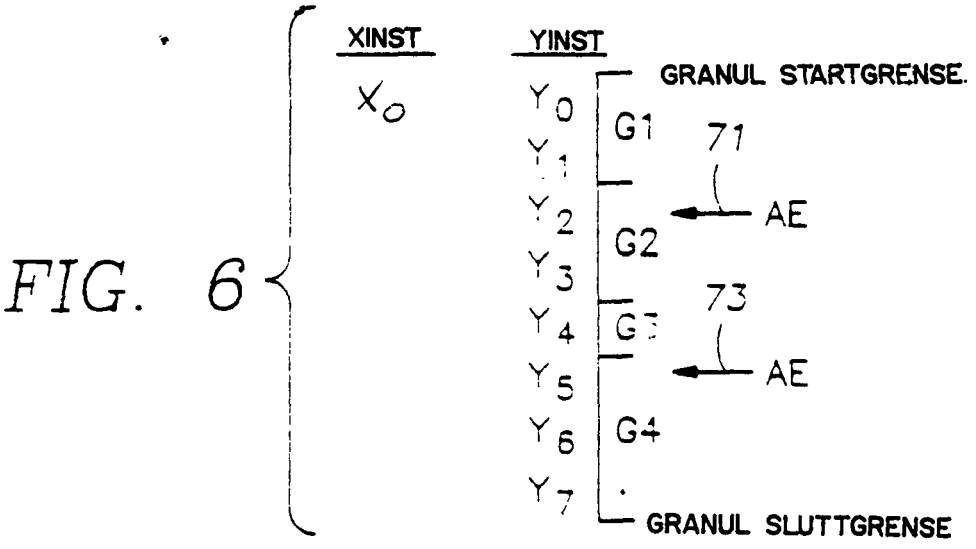

XINST
YINST
X_0
Y_0
Y_1
Y_2
Y_3
Y_4
Y_5
Y_6
Y_7
G1
G2
G3
G4
GRANUL STARTGRENSE
GRANUL SLUTTGRENSE
71
73
AE
AE

FIG. 6

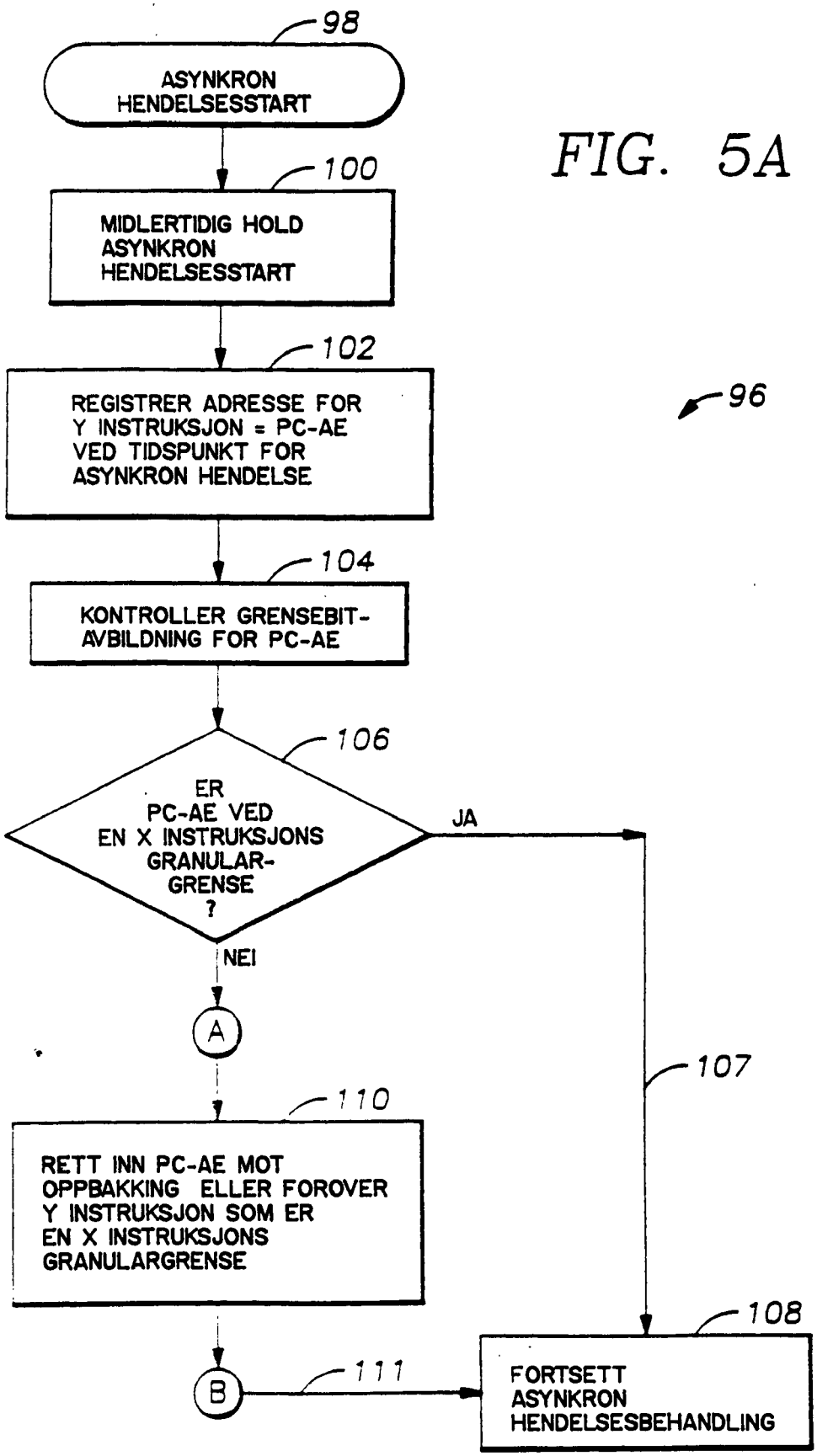

98
ASYNKRON HENDELSESSTART
100
MIDLERTIDIG HOLD ASYNKRON HENDELSESSTART
102
REGISTRER ADRESSE FOR Y INSTRUKSJON = PC-AE VED TIDSPUNKT FOR ASYNKRON HENDELSE
96
104
KONTROLLER GRENSEBIT-AVBILDNING FOR PC-AE
106
ER PC-AE VED EN X INSTRUKSJONS GRANULAR-GRENSE ?
JA
NEI
A
107
110
RETT INN PC-AE MOT OPPBAKKING ELLER FOROVER Y INSTRUKSJON SOM ER EN X INSTRUKSJONS GRANULARGRENSE
108
B
111
FORTSETT ASYNKRON HENDELSESBEHANDLING

FIG. 5A

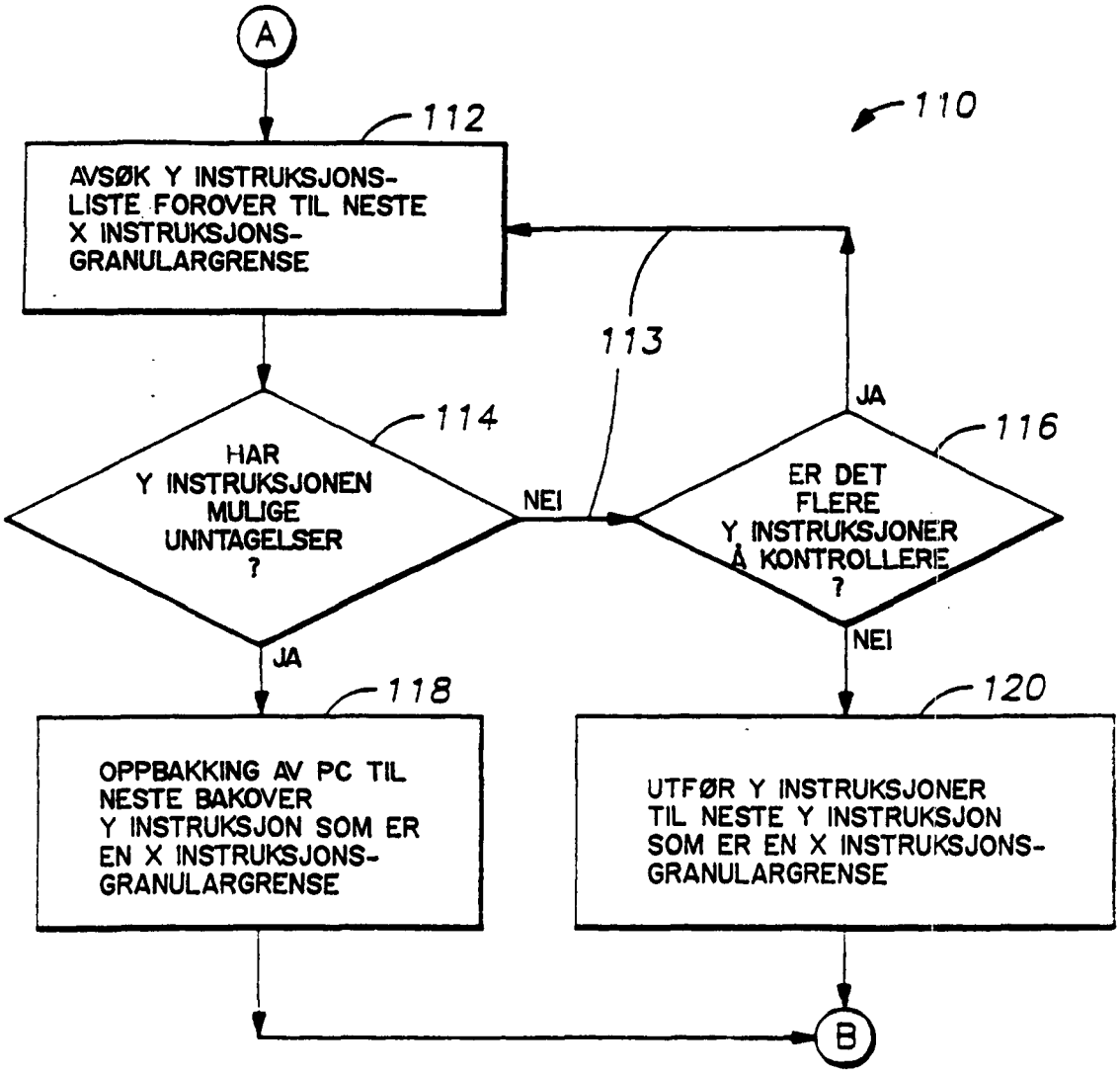

A
110
112
AVSØK Y INSTRUKSJONS-
LISTE FOROVER TIL NESTE
X INSTRUKSJONS-
GRANULARGRENSE
113
114
HAR
Y INSTRUKSJONEN
MULIGE
UNNTAGELSER
?
NEI
JA
116
ER DET
FLERE
Y INSTRUKSJONER
Å KONTROLLERE
?
JA
NEI
118
OPPBAKKING AV PC TIL
NESTE BAKOVER
Y INSTRUKSJON SOM ER
EN X INSTRUKSJONS-
GRANULARGRENSE
120
UTFØR Y INSTRUKSJONER
TIL NESTE Y INSTRUKSJON
SOM ER EN X INSTRUKSJONS-
GRANULARGRENSE
B

FIG. 5B

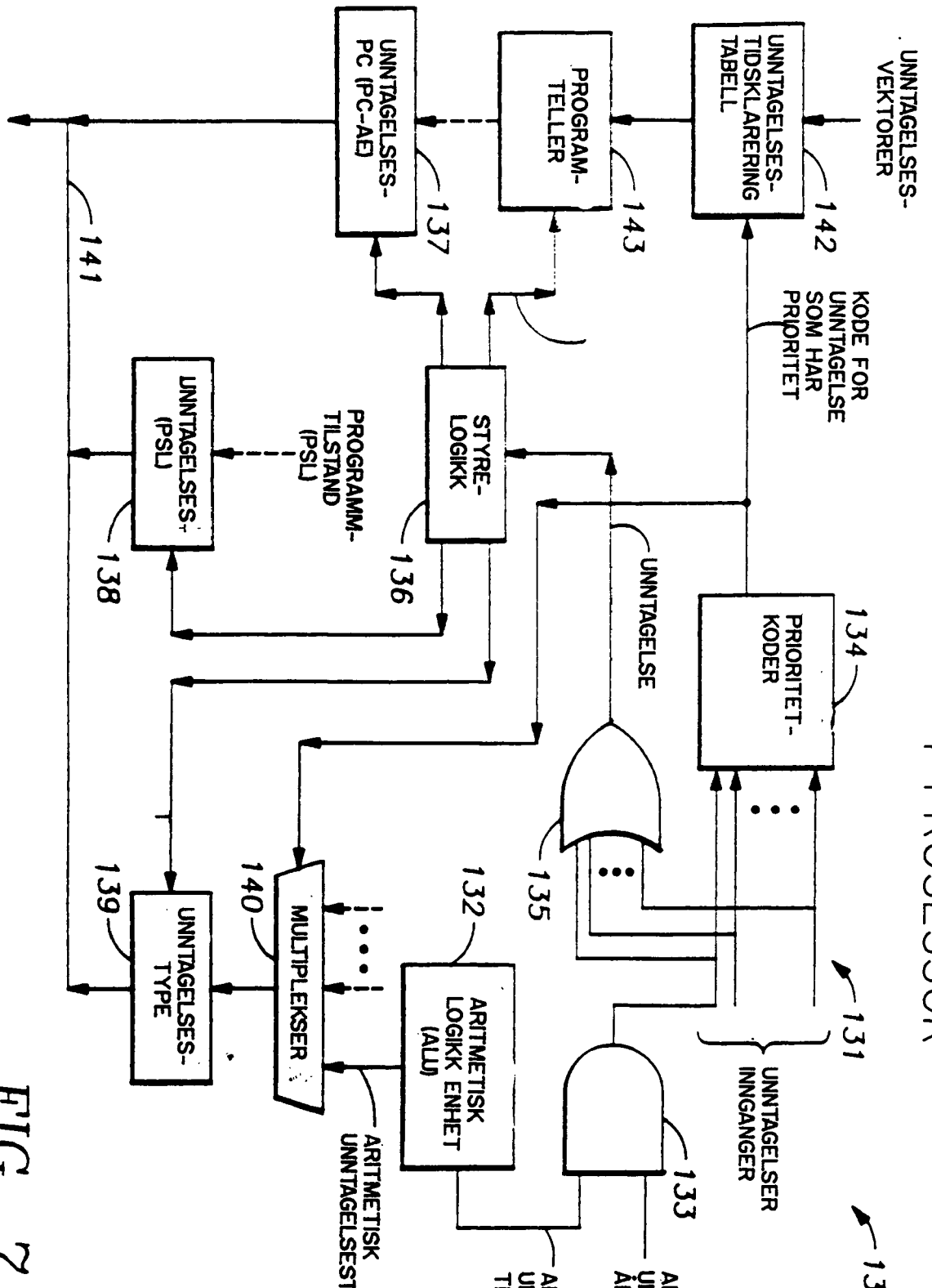

Y PROSESSOR
130
131
UNNTAGELSES-VEKTORER
UNNTAGELSES-TIDSKLARERING TABELL
142
PROGRAM-TELLER
143
UNNTAGELSES-PC (PC-AE)
137
141
KODE FOR UNNTAGELSE SOM HAR PRIORITET
PRIORITET-KODER
134
UNNTAGELSER INNGANGER
UNNTAGELSE
135
STYRE-LOGIKK
136
PROGRAMM-TILSTAND (PSL)
UNNTAGELSES-(PSL)
138
ARITMETISK LOGIKK ENHET (ALU)
132
133
ARITMETISK UNNTAGELSE ÅPNER
ARITMETISK UNNTAGELSES-TILSTAND
MULTIPLEKSER
140
ARITMETISK UNNTAGELSESTYPE
UNNTAGELSES-TYPE
139

FIG. 7

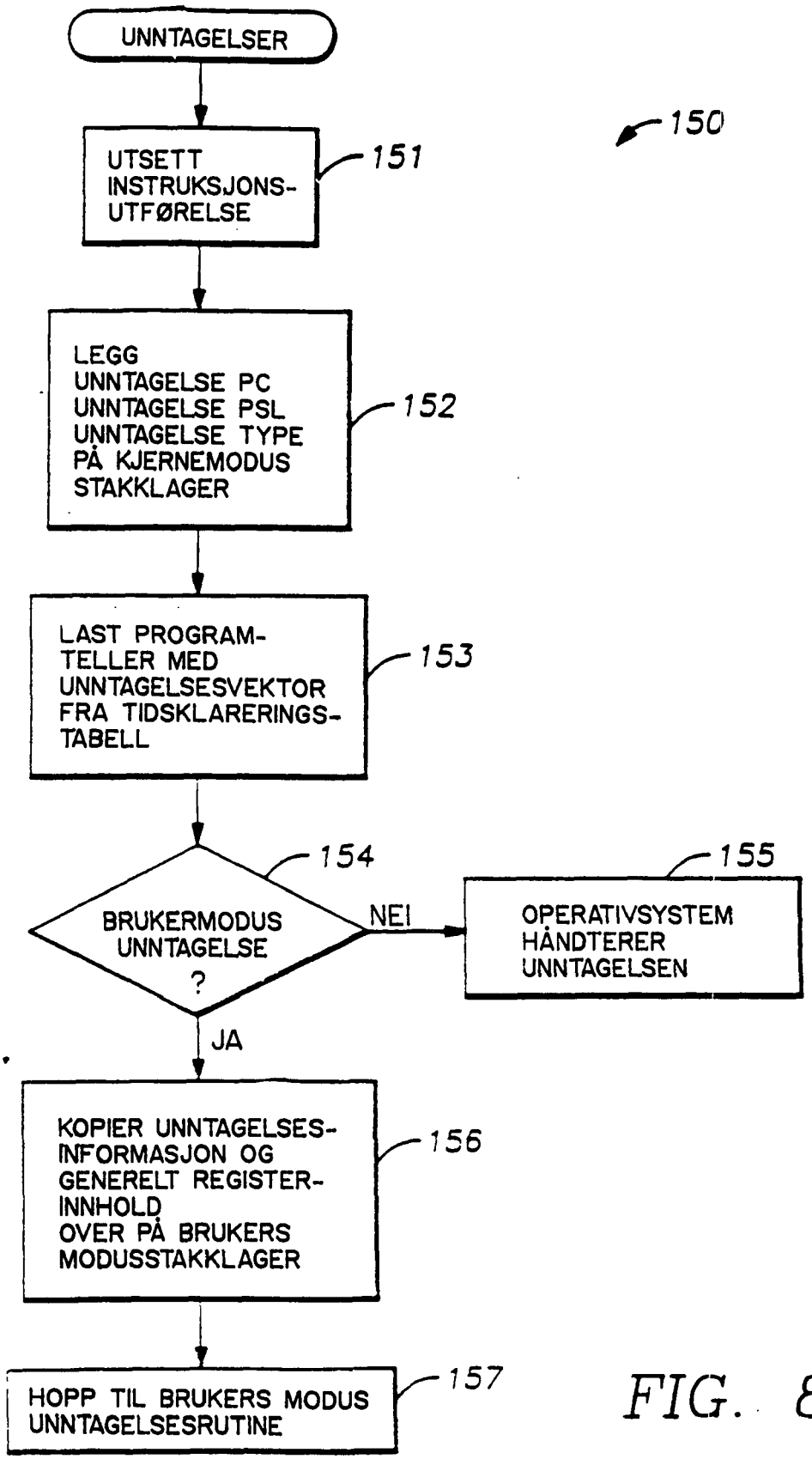

UNNTAGELSER
150
UTSETT INSTRUKSJONS-UTFØRELSE
151
LEGG UNNTAGELSE PC UNNTAGELSE PSL UNNTAGELSE TYPE PÅ KJERNEMODUS STAKKLAGER
152
LAST PROGRAM-TELLER MED UNNTAGELSESVEKTOR FRA TIDSKLARERINGS-TABELL
153
BRUKERMODUS UNNTAGELSE ?
154
NEI
OPERATIVSYSTEM HÅNDTERER UNNTAGELSEN
155
JA
KOPIER UNNTAGELSES-INFORMASJON OG GENERELT REGISTER-INNHOLD OVER PÅ BRUKERS MODUSSTAKKLAGER
156
HOPP TIL BRUKERS MODUS UNNTAGELSESRUTINE
157

FIG. 8

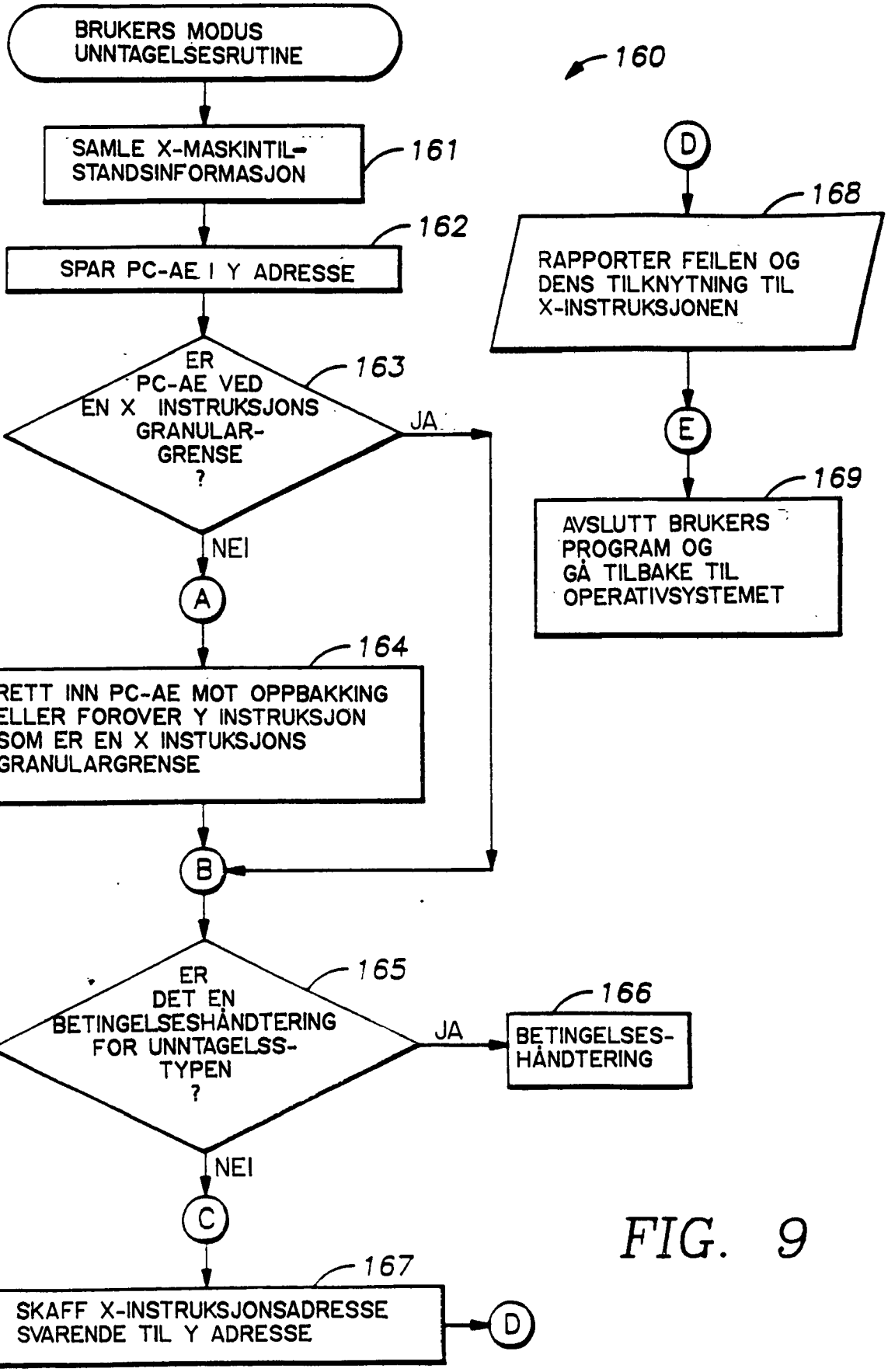

BRUKERS MODUS UNNTAGELSESRUTINE
160
SAMLE X-MASKINTIL-STANDSINFORMASJON
161
SPAR PC-AE I Y ADRESSE
162
ER PC-AE VED EN X INSTRUKSJONS GRANULAR-GRENSE ?
163
JA
NEI
A
RETT INN PC-AE MOT OPPBAKKING ELLER FOROVER Y INSTRUKSJON SOM ER EN X INSTUKSJONS GRANULARGRENSE
164
B
ER DET EN BETINGELSESHÅNDTERING FOR UNNTAGELSS-TYPEN ?
165
JA
BETINGELSES-HÅNDTERING
166
NEI
C
SKAFF X-INSTRUKSJONSADRESSE SVARENDE TIL Y ADRESSE
167
D
D
RAPPORTER FEILEN OG DENS TILKNYTNING TIL X-INSTRUKSJONEN
168
E
AVSLUTT BRUKERS PROGRAM OG GÅ TILBAKE TIL OPERATIVSYSTEMET
169

FIG. 9

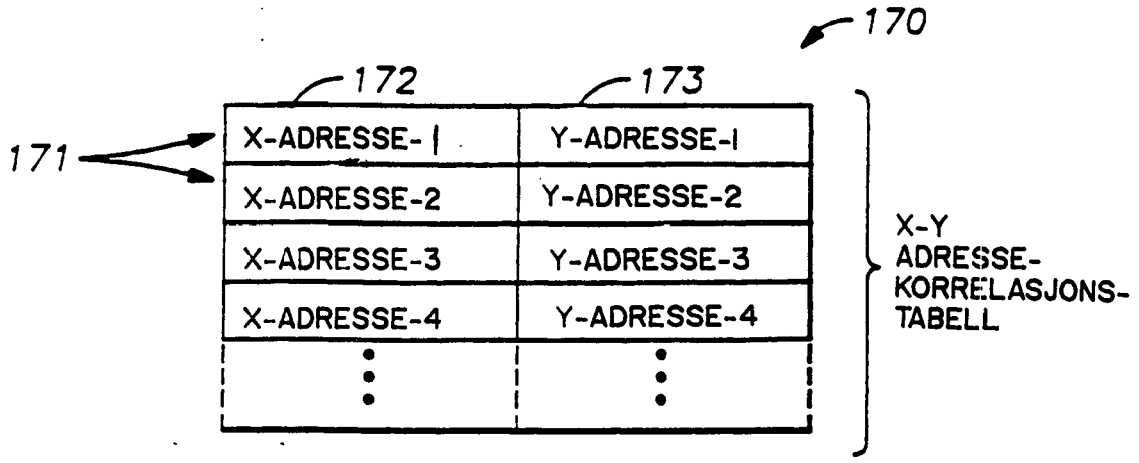

170
172
173
171
X-ADRESSE-1
Y-ADRESSE-1
X-ADRESSE-2
Y-ADRESSE-2
X-ADRESSE-3
Y-ADRESSE-3
X-ADRESSE-4
Y-ADRESSE-4
X-Y
ADRESSE-
KORRELASJONS-
TABELL

FIG. 10

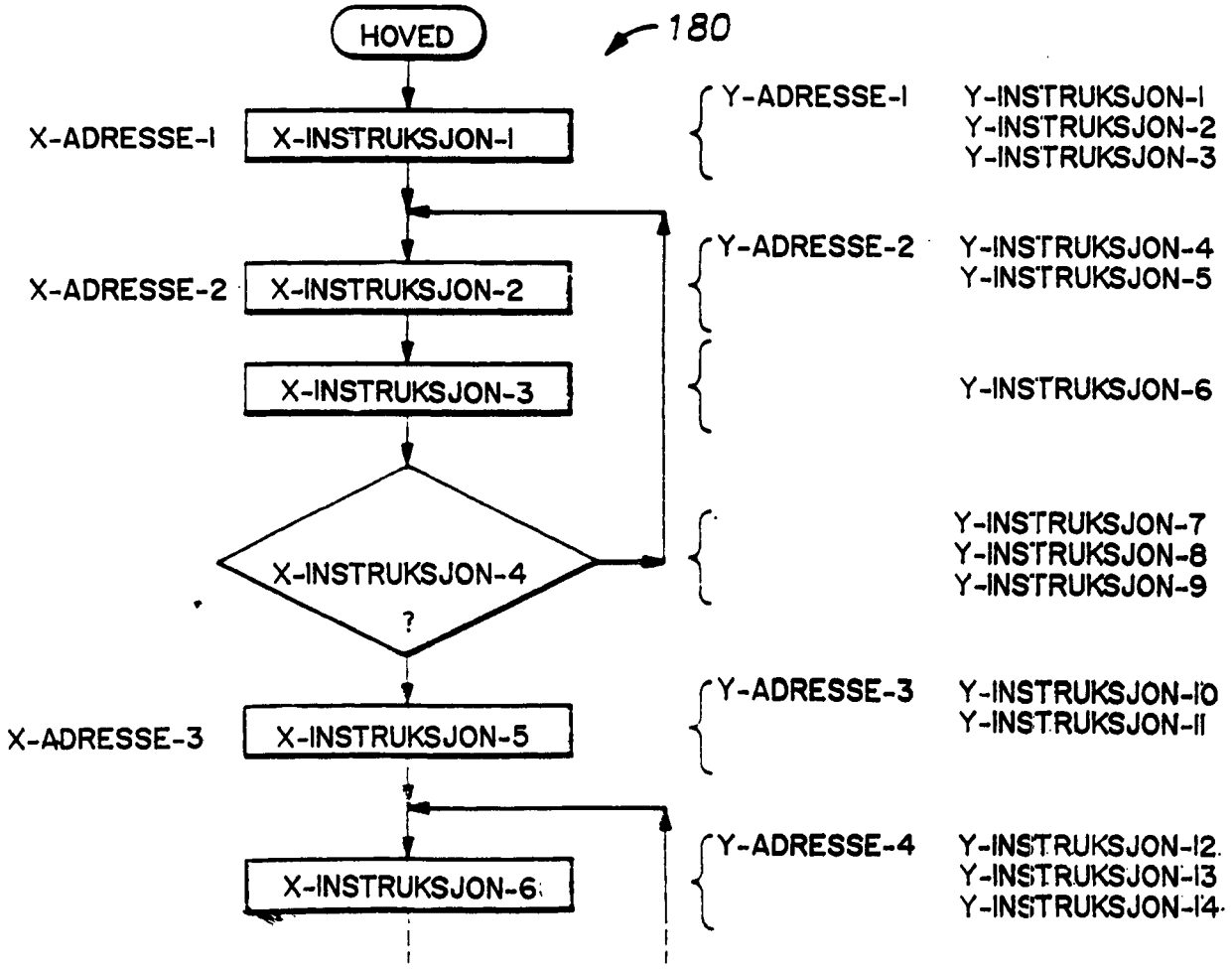

HOVED
180
X-ADRESSE-1
X-INSTRUKSJON-1
Y-ADRESSE-1
Y-INSTRUKSJON-1
Y-INSTRUKSJON-2
Y-INSTRUKSJON-3
X-ADRESSE-2
X-INSTRUKSJON-2
Y-ADRESSE-2
Y-INSTRUKSJON-4
Y-INSTRUKSJON-5
X-INSTRUKSJON-3
Y-INSTRUKSJON-6
X-INSTRUKSJON-4
?
Y-INSTRUKSJON-7
Y-INSTRUKSJON-8
Y-INSTRUKSJON-9
X-ADRESSE-3
X-INSTRUKSJON-5
Y-ADRESSE-3
Y-INSTRUKSJON-10
Y-INSTRUKSJON-11
X-INSTRUKSJON-6
Y-ADRESSE-4
Y-INSTRUKSJON-12
Y-INSTRUKSJON-13
Y-INSTRUKSJON-14

FIG. 11

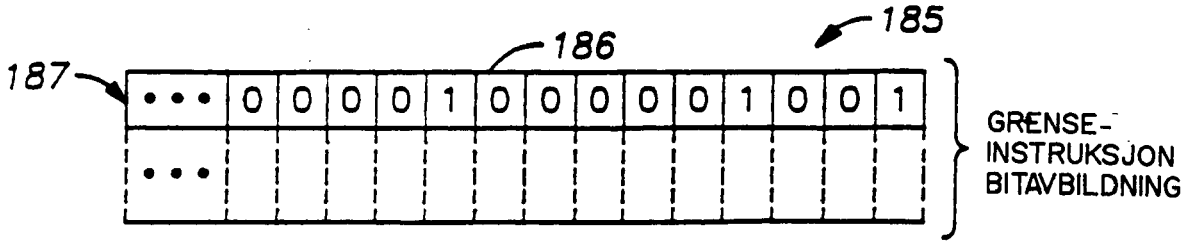

185
186
187
• • •
0 0 0 0 1 0 0 0 0 0 1 0 0 1
• • •
GRENSE-
INSTRUKSJON
BITAVBILDNING

FIG. 12

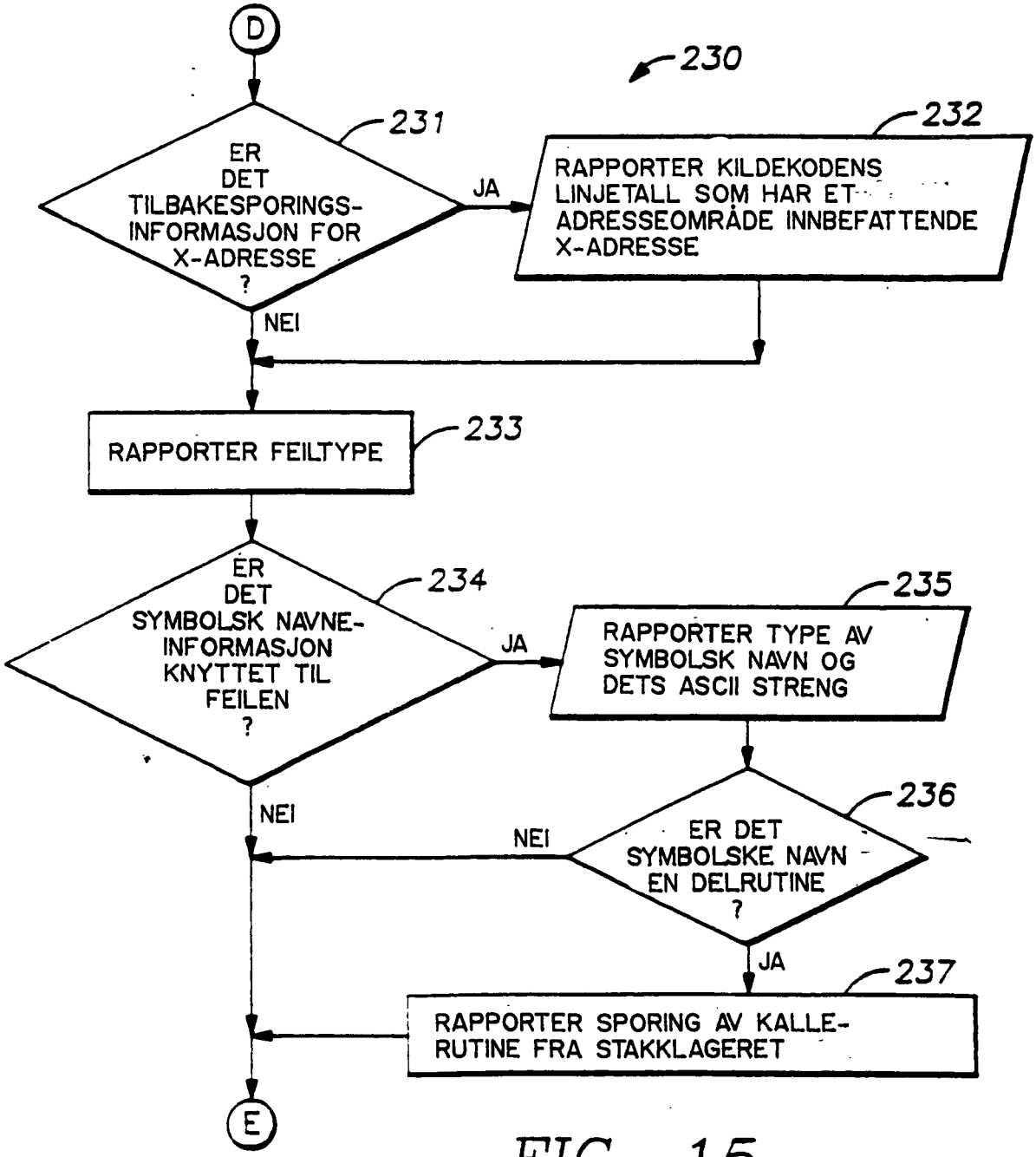

D
230
231
ER
DET
TILBAKESPORINGS-
INFORMASJON FOR
X-ADRESSE
?
JA
232
RAPPORTER KILDEKODENS
LINJETALL SOM HAR ET
ADRESSEOMRÅDE INNBEFATTENDE
X-ADRESSE
NEI
RAPPORTER FEILTYPE
233
ER
DET
SYMBOLSK NAVNE-
INFORMASJON
KNYTTET TIL
FEILEN
?
234
JA
235
RAPPORTER TYPE AV
SYMBOLSK NAVN OG
DETS ASCII STRENG
NEI
NEI
ER DET
SYMBOLSKE NAVN
EN DELRUTINE
?
236
JA
237
RAPPORTER SPORING AV KALLE-
RUTINE FRA STAKKLAGERET
E

FIG. 15

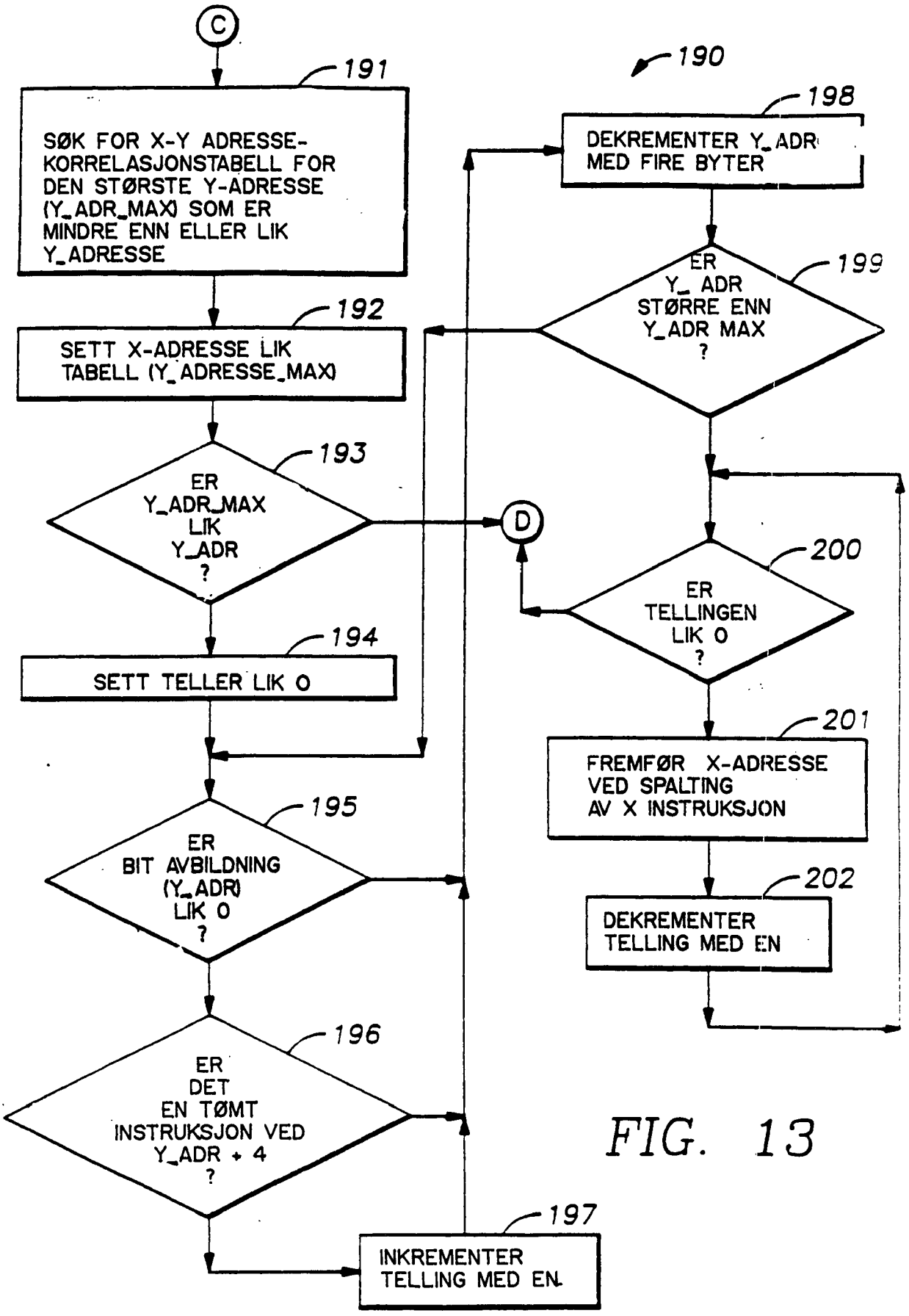

C
191
SØK FOR X-Y ADRESSE-KORRELASJONSTABELL FOR DEN STØRSTE Y-ADRESSE (Y_ADR_MAX) SOM ER MINDRE ENN ELLER LIK Y_ADRESSE
192
SETT X-ADRESSE LIK TABELL (Y_ADRESSE_MAX)
193
ER Y_ADR_MAX LIK Y_ADR ?
D
194
SETT TELLER LIK 0
195
ER BIT AVBILDNING (Y_ADR) LIK 0 ?
196
ER DET EN TØMT INSTRUKSJON VED Y_ADR + 4 ?
197
INKREMENTER TELLING MED EN
190
198
DEKREMENTER Y_ADR MED FIRE BYTER
199
ER Y_ADR STØRRE ENN Y_ADR MAX ?
200
ER TELLINGEN LIK 0 ?
201
FREMFØR X-ADRESSE VED SPALTING AV X INSTRUKSJON
202
DEKREMENTER TELLING MED EN

FIG. 13

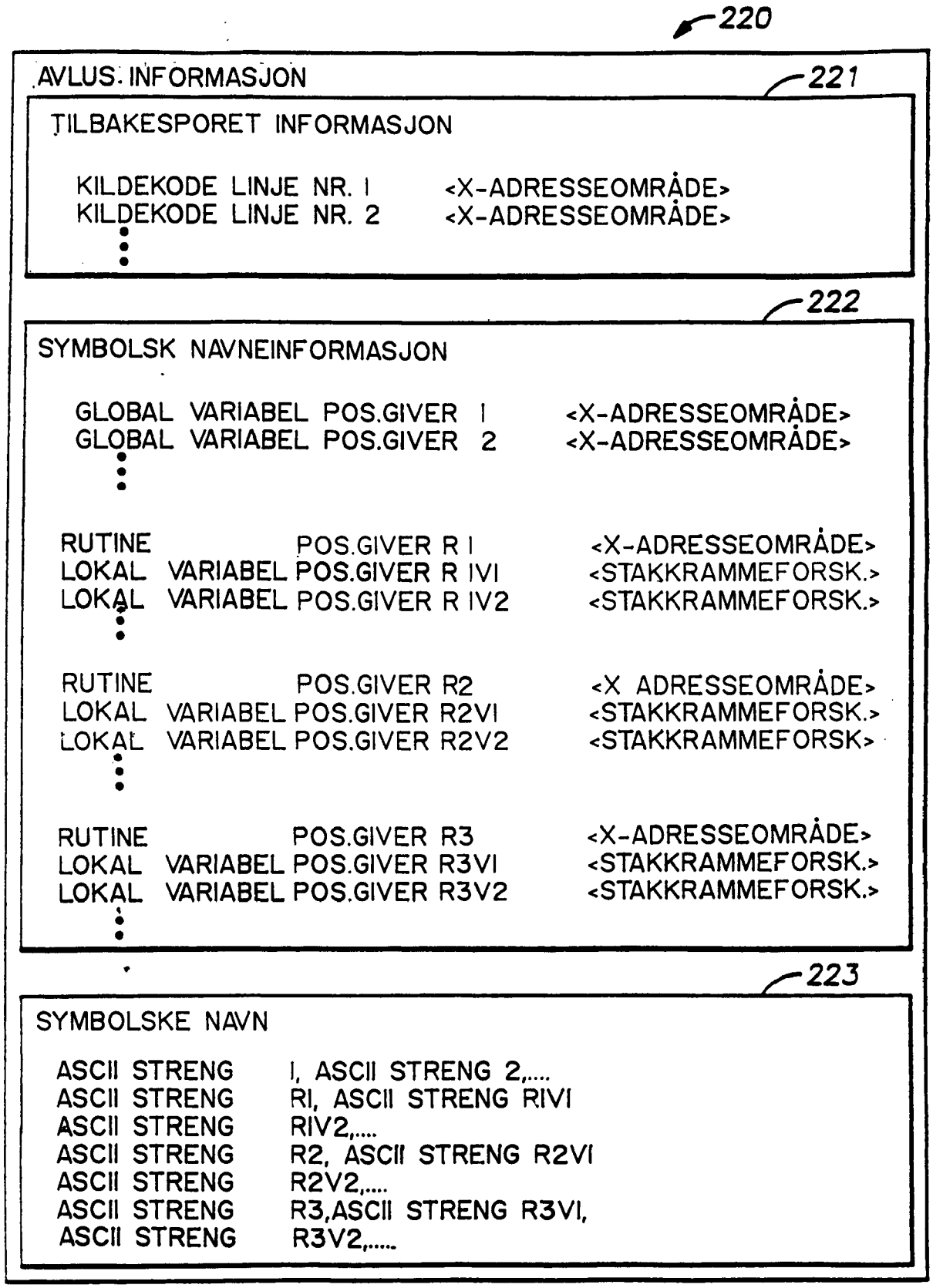

220
AVLUS. INFORMASJON
221
TILBAKESPORET INFORMASJON
KILDEKODE LINJE NR. 1 <X-ADRESSEOMRÅDE>
KILDEKODE LINJE NR. 2 <X-ADRESSEOMRÅDE>
222
SYMBOLSK NAVNEINFORMASJON
GLOBAL VARIABEL POS.GIVER 1 <X-ADRESSEOMRÅDE>
GLOBAL VARIABEL POS.GIVER 2 <X-ADRESSEOMRÅDE>
RUTINE POS.GIVER R 1 <X-ADRESSEOMRÅDE>
LOKAL VARIABEL POS.GIVER R IVI <STAKKRAMMEFORSK.>
LOKAL VARIABEL POS.GIVER R IV2 <STAKKRAMMEFORSK.>
RUTINE POS.GIVER R2 <X ADRESSEOMRÅDE>
LOKAL VARIABEL POS.GIVER R2VI <STAKKRAMMEFORSK.>
LOKAL VARIABEL POS.GIVER R2V2 <STAKKRAMMEFORSK>
RUTINE POS.GIVER R3 <X-ADRESSEOMRÅDE>
LOKAL VARIABEL POS.GIVER R3VI <STAKKRAMMEFORSK.>
LOKAL VARIABEL POS.GIVER R3V2 <STAKKRAMMEFORSK.>
223
SYMBOLSKE NAVN
ASCII STRENG I, ASCII STRENG 2,....
ASCII STRENG RI, ASCII STRENG RIVI
ASCII STRENG RIV2,....
ASCII STRENG R2, ASCII STRENG R2VI
ASCII STRENG R2V2,....
ASCII STRENG R3,ASCII STRENG R3VI,
ASCII STRENG R3V2,.....

FIG. 14