(19) LIETUVOS RESPUBLIKA VALSTYBINIS PATENTŲ BIURAS V P B

(10) **LT IP385 A**

(12) **PARAIŠKOS APRAŠYMAS**

(21) Paraiškos numeris: **IP385**

(22) Paraiškos padavimo data: **1993 03 05**

(41) Paraiškos paskelbimo data: **1994 11 25**

(62) Paraiškos, iš kurios dokumentas išskirtas, numeris: —

(86) Tarptautinės paraiškos numeris: —

(86) Tarptautinės paraiškos padavimo data: —

(85) Nacionalinio PCT lygio procedūros pradžios data: —

(30) Prioritetas: **9002558, 1990 08 02, SE**
PCT/SE91/00514, 1991 08 01, SE

(51) Int. Cl. (2006): **G06F 7/00**
G06F 7/38
G06F 7/48
G06F 9/302
G06F 9/44
G06F 12/00
G11C 11/412
G11C 11/417
G11C 11/419
G11C 15/00

(71) Pareiškėjas:

CARLSTEDT ELEKTRONIK AB, Industrivagen 55, S-433 61 Partille, SE

(72) Išradėjas:

Lars Gunnar CARLSTEDT, SE

(74) Patentinis patikėtinis/atstovas:

Reda ŽABOLIENĖ, Advokatės Redos Žabolienės kontora METIDA, Verslo centras VERTAS, Gynėjų g. 16, LT-01109 Vilnius, LT

(54) Pavadinimas:

Aritmetinis blokas struktūrinei aritmetikai

(57) Referatas:

—

LT IP385 A

ARITMETINIS BLOKAS STRUKTŪRINEI ARITMETIKAI

Šis išradimas yra skirtas aritmetiniam struktūriniam blokui. Šis aritmetinis blokas toliau bus vadinamas šerdies ęele.

IŠRADIMO PAGRINDAI

Kompiuteris buvo išrastas 1940-jų metų laikotarpyje. Nuo to laiko kompiuterinė technika vystėsi revoliuciniu tempu. Nepaisant to, šiuo metu kompiuteriai turi tą pačią architektūrą, kaip ir pirmieji pavyzdžiai.

Didžiausi patobulinimai buvo padaryti aparatinėje dalyje. VLSI įgyvendinimas ir pasiekimai litografijos srityje įgalino sukurti kompiuterius, susidedančius iš vienos mikroschemos. Tai tokie kompiuteriai, kurie tik prieš penketą metų buvo vadinami superkompiuteriais. Linijų (mikroschemose) plotis siaurėjo eksponentiniu dėsniu ir dabar jis yra mažesnis, negu 1 mikrometras. Naudojamų impulsų dažnio dydis, o taip pat ir aktyvių tranzistorių galingumo diapazonas padidėjo daug kartų. Matyt, linijų plotis bus fiziškai ribojamas iki 0,2 mikrometrų dydžio.

Tuo pačiu metu kompiuterių architektūros kūrėjai nedaug pasiekė silicio panaudojimo srityje. Priešingai, dauguma kompiuterių naudoja daugiau, negu optimalų silicio kiekį, siekiant didesnio jų greitaeigingumo.

Abu šie faktai stabdo vieno atskiro procesoriaus greitaeigingumo vystymą bent artimiausių penkių metų laikotarpiui. Lygiagretieji procesoriai yra naudojami dėl didėjančios aparatinės dalies kainos ir dėl didėjančio sudėtingumo bei daugeliui programų tipų didėjančios programavimo kainos.

Peržvelgiant į jų tarpusavio santykį, aparatinės dalies kaina sumažėjo, tačiau naujų sistemų programavimo kaina pastebimai išaugo ir greitai taps neleistinai didelė.

Kompiuteris yra sudėtingas įvairių blokų įrenginys, tiek aparatine dalimi, tiek ir programine dalimi. Skirtingi jų pavyzdžiai ir vystymosi etapai skatino kurti specialius standartus, kurie buvo diegiami jų sistemose. Dėl šių nevienodumų atsirado daug įvairių interfeisų.

Visi šie interfeisai yra pavyzdys, kaip skirtinga kokybė ir naudojamas stilius sudaro sunkumus vartotojui arba progrmuotojui, besinaudojančiam įrenginiu - jis reikalauja daug žinių, o dėl jo sudėtingumo programuotojas gali pridaryti nepastebimų klaidų.

Pastaruoju metu atsirado ir yra tobulinami taip vadinamieji suskaidantieji (redukciniai) procesoriai. Suskaidantieji procesoriai atlieka programą, turinčią tam tikrą struktūrą, į kurią įeina aritmetinės lygtys, ši struktūra yra susmulkinama į eilę sumažintų žingsnių. Tokiu būdu, programa nėra vykdoma pagal nustatytą eilės tvarką, kaip kitų rūšių kompiuteriuose.

Kuriant suskaidančius procesorius, viršijančius nustatytą dydį, buvo susiduriama su tam tikrais sunkumais.

Programavimo kalbų paruošimas

Pirmo elektroninio kompiuterio sukūrimas davė pradžią kelių programavimo kalbų sukūrimui, kurios tiktų šio tipo kompiuteriams, tokių, kaip FORTRAN, COBOL, ALGOL, BASIC, Pascal. Šios kalbos buvo pavadintos imperatyvinėmis kalbomis, greta taip vadinamų įprastinių kalbų, daugiausia dėl to fakto, kad jos paprastai pateikia programą, susidedančią iš eilės komandų arba nurodymų, kurie turi būti vykdomi eilės tvarka įprastiniu kompiuteriu, pvz., kompiuteriu, sukurtu pagal John von Neumann principus. Didėjantis diskomfortas, naudojantis šiomis kalbomis, priverte išvystyti kitą kalbų serija: LISP, ISWIM, Scheme (LISP dialektas), ML, Hope, SASL ir tt. Šių kalbų sukūrimą skatino jų schematiškas paprastumas. Nei viena iš konkrečių mašinų įtakos jų kūrimui neturėjo.

Praėjo kiek laiko, kol dėmesį patraukė funkcinės kalbos. Viena iš to priežasčių buvo tai, kad funkcinės programos kompiuteryje buvo apdorojamos lėtai. Vėlesni darbai parodė, kad tam tikrais atvejais programų vykdymo greitis gali būti artimas arba toks pats, kaip ir įprastinių kalbų programų, atliekamų

įprastiniais kompiuteriais, nors funkcinės programos ir nėra skirtos šio tipo kompiuteriams.

Programinės įrangos krizė

Didėjantis diskomfortas, naudojantis imperatyvinio tipo kalbomis, plačiu mastu paskatino dėti pastangas sukurti funkcines kalbas. Kalba apie programinės įrangos krizę prasidėjo apie 1970m. Programos darėsi vis sudėtingesnės ir dažnai turėdavo daugybę klaidų, buvo sunkiai įvedamos ir ypač sunkiai tobulinamos. Viena iš tų keblumų priežasčių yra tai, kad buvo dedami per dideli lūkesčiai, jog aukšto lygio imperatyvinės kalbos turėtų supaprastinti programavimą. Šios kalbos nėra tokio aukšto lygio, kaip jos gali atrodyti. Imperatyvinės kalbos vis dar yra adaptuojamos pagal ankstyvųjų kompiuterių koncepciją - von Neumann tipo kompiuterių koncepciją, ir programavimo lygis vis dar yra artimas aparatiniam lygiui. Funkcinio programavimo kalbos turi kai kurias savybes, kurios mažina labiausiai įprastus programavimo kalbų nepatogumus.

Papildoma informacija ir paaiškinimai gali būti randami knygoje "Functional Programming Using Standard ML", Ake Wikstrom, Prentice Hall 1987.

Tam, kad pilnai būtų suprantami išradimo tikslai ir privalumai, svarbu suprasti, iš ko susideda funkcinis požiūris kompiuteriniame darbe. Istoriškai lyginant, labiausiai tiktų imperatyvinis požiūris.

Išsireiškimas "funkcinis požiūris" reiškia, kad programos yra parašytos funkcine kalba, įvedamos į atmintį ir atliekamos kompiuteryje, parenkant aparatinės dalies sudėtį, tinkamą tokiai kalbai. Taip pat, išsireiškimu "imperatyvinis požiūris" turima galvoje, kad programa yra parašyta imperatyvine (privalomaja) kalba, įvedama į atmintį ir atliekama kompiuteryje, parenkant aparatinės dalie sudėtį, tinkamą šiai kalbai.

Tačiau yra įmanoma įvesti ir atlikti įprastiniu kompiuteriu programas, parašytas ir funkcine kalba. Priešingas atvejis yra taip pat įmanomas - programos, parašytos privalomomis kalbomis gali būti atliekamos kompiuteriu, skirtu vykdyti programas, parašytas funkcinėmis kalbomis.

Funkcinių kalbų savybės

Į programą, parašytą funkcine kalba, gali būti žiūrima, kaip į rinkinį objekto savybių apibrėžimų, ir kaip į kompiuterines taisykles. Apibrėžimai yra deklaratyvioji dalis, o kompiuterinio darbo taisyklės, arba suskaidymas į dalis, arba perrašymo taisyklės, sudaro operatyvinę dalį, kurią naudoja kompiuteris darbo metu. Funkcinės kalbos sudaro aukštesnio lygio interfeisą kompiuteryje, kas įgalina programuotoją nepaisyti kompiuterio aparatinės dalies sudėties. Teigiamu papildomu efektu yra tai, kad funkcinės programos dažnai yra trumpesnės ir jas lengviau suprasti, negu įprastines privalomąsias programas. Vienas iš pagrindinių funkcinių kalbų trūkumų yra tai, kad funkcinės programos turi būti transliuojamas į įprastines kalbas tam tikslui, kad jas galima būtų atlikti įprastiniais kompiuteriais. Tai yra atliekama kompiliatorių arba interpretuojančių programų pagalba. Aišku, kad funkcinio požiūrio naudingumas yra šiek tiek menkinamas to fakto, kad apartinė dalis neturi priskirtos dalies funkcinių programų įvedimui ir jų atlikimui kuo efektyviau.

IŠRADIMO TIKSLAI

Išradimo tikslas yra sukurti ypatingą aktyvią atminties celę, toliau vadinamą šerdies cele, kuri yra aktyvios atminties sudėtyje ir kurios celės turi galimybę atlikti visų rūšių sus-kaidymus. Tuo tarpu aktyvi atmintis taip pat turi kito tipo at-minties celes, galinčias atlikti tik ribotą kiekį kai kurių tipų

suskaidymo operacijas. Tačiau skaitmeninės operacijos gali būti atliktos skaitmeniniame aritmetiniame loginiame bloke (ALU), sujungtame su šerdies cele, tačiau esminės suskaidymo operacijos yra atliekamos šerdies celėje.

Kitas išradimo tikslas yra pateikti šerdies celę, esančią saveikoje su asociatyvine atmintimi ir turinčią ribotą kiekį atminties celių, toliau vadinamą objekto atmintimi, ir kuri yra vienintelė celė, turinti tiesioginį ryšį su asociatyvine atmintimi informacijos perdavimo metu.

Dar kitas išradimo tikslas yra sukurti šerdies celę, kurioje gali būti saugoma daugybė reiškinio lygių. Šerdies celėje yra vykdoma daugybė įvestų į atmintį reiškinio bazinių komandų.

Kitas išradimo tikslas yra pateikti šerdies celę, kuri yra skirta aritmetikos struktūrai, kuri yra naudojama struktūros suskaidymui pagal kompiuterio programą, ir kurioje celės reiškinys yra tokio dydžio, kad atitiktų reprezentacinio simbolio šakai, dalyvaujančiai veiksme. Įprasto tipo kompiuteris gali būti aprūpintas šerdies cele, skirta aritmetinėms struktūroms, kuri bus ypač tobula, kai bus naudojama programinė kalba, kuri bazuojama aritmetinių struktūros koncepcijų pamatu, pavyzdžiui, LISP (LISP - LISt Processing language - sąrašus apdorojanti kalba), arba bet kuria kita funkcine arba aprašomaja kompiuterine kalba. LISP kalba yra ypatingai tinkama tvarkyti ir apdoroti sąrašus arba struktūras, sudarytas iš sąrašų, ji yra naudojama dirbtiniam intelektui, pavyzdžiui, sudarant ekspertines sistemas. Ji taip pat yra naudojama, dirbant su simboline algebra, VLSI projektavimui, robotikoje, kalbų mokymuisi ir tt. Šerdies celė gali būti kaip priemonė sąrašų apdorojimui, juos perrašant arba/ir jų suskaidymui.

Suskaidymo procesoriai simbolių skaidymui buvo nagrinėjami pastaraisiais metais, o speciali jų įgyvendinimo technika yra aprašyta straipsnyje: D. A. Turner "A New Implementation Technique for Applicative Languages", Software-Practice and

Experience," Vol. 9, pp 31-49, 1979. JAV patente PS-4,644,464 yra išvystytos šiame straipsnyje pateiktos idėjos ir tokiu būdu, jis liečia perdavimo mechanizmą lygiagrečiuose registruose, naudojamuose skaitmeniniuose procesoriuose, kurie yra pritaikyti apdoroti programas, pateiktas binarinių orientuotų simbolių pavidalu, tolygiai juos pakeičiant ekvivalentiniais simboliais. Neišvengiamai reikia įgyvendinti taikomąsias kalbas, tokias, kaip LISP arba SASL. Tuo tikslu, kad būtų padidintas programų vykdymo greitaeigingumas, JAV PS - 4,644,464 aprašo taip vadinamą simbolių menedžerį, kuris gali suskaidyti kombinatorinius simbolius, vaizduojančius taikomųjų kalbų, neturinčių kintamųjų, kodus, gaunamus, transformuojant aukšto lygio kodą per daugelį vienu metu atliekamų perdavimų registruose. Dviejų lygių susikirtimo taškai turi tinklelio pavidalo laukus. Simbolis yra susmulkinamas funkciniu pakeitimu, pertvarkymą registruose valdant iš valdymo įrenginio. Registruose gali būti saugomi dviejų celių susikirtimo taškai simbolyje. Registrai yra valdomi būsenos testavimo įrenginiu, kuris generuoja perkėlimo adresą. Valdymo atminties elementai išrenka eilę valdymo signalų, kurie paveikia perdavimą tarp registrų, turinčių atmintyje dviejų celių susikirtimo tinklelį ir tokiu būdu yra atliekamas funkcinis pakeitimas.

Programų vykdymas suskaidymo būdu

Norima atlikti programa gali būti pateikiama tiesioginiu uždarų reiškinių simboliu, kur kiekviena programos dalis yra atstovaujama uždaru reiškiniu. Vykdymo metu šis tiesioginis uždarų reiškinių simbolis yra laipsniškai skaidomas atitinkamai pagal naudojamos kalbos suskaidymo taisykles. Kai nebelieka vykdymui uždarų reiškinių, programos vykdymas yra baigtas.

Tiesioginiai uždarų reiškinių simboliai gali būti laikomi medžio struktūra, kur kiekvienas susikertantis medžio taškas yra

uždaras reiškinys ir kur pačioje viršūnėje esantis susikirtimo taškas (elementas) yra vadinamas šaknimi. Programos vykdymas suskaidymo būdu toliau atliekamas įprastu būdu, skaidant medžio struktūrą iš apačios į viršų, suskaidant medžio dalis, esančias toliausiai nuo šaknies ir einant šaknies link. Šis programos vykdymo būdas yra paprastai vadinamas vykdymu, paleidžiamu nuo užklausimo, t.y., programos dalių vykdymas priklauso nuo kitų dalių įvykdymo rezultato ir yra atidedamas tol, kol gaunamas tas rezultatas.

APIBRĖŽIMAI

Žemiau yra pateikiamas išsireiškimų sąrašas, naudojamas šiame aprašyme, ir jų reikšmės:

element elementas	šiokia tokia apimtį turinti dalis duomenų struktūroje;
list sąrašas	sutvarkyta elementų eilė, kurioje kiekvienas elementas gali būti sąrašu;
inserted list įterptas sąrašas	sąrašo dalis, per maža saugojimui viename uždarame reiškinyje. Ji pati gali atstovauti palyginti ilgus sąrašus;
closure uždaras reiškinys	hierarchinės struktūros kategorija, apsakanti procesą. Visos šios kategorijos turi šaknį, kuri unikaliai apibūdina uždaraįį reiškinį. Darbas suskaidančiu kompiuteriu yra atliekamas uždarų reiškinių pagalba. Visas įrenginio būvis yra transformuojamas suskaidymais;
object storage objektinė atmintis	atmintis, įskaitant lasteles objektų saugojimui. Pavyzdžiui, asociatyvi atmintis;

storage cell atminties lastelė	celė, skirta objektų saugojimui. Ji saugo atmintyje uždarų reiškinių lasteles, kurios gali turėti ryšį su kitomis uždarų reiškinių lastelėmis kitose atminties celėse;
cell closure uždara lastelė	atminties celės turinys;
storage cell field atminties celės laukas	laukas, esantis atminties celėje;
closure element uždaras elementas	duomenų elementas, saugomas atminties celės lauke;
closure identifier uždaro reiškinio identifikatorius	uždaro reiškinio elementas, vienareikšmiškai jį pažymintis;
canonical closure kanoninis uždaras reiškinys	uždaras reiškinys, kuris negali būti daugiau suskaidytas, pvz., celė, kuri neturi daugiau uždaro reiškinio identifikatorių, žyminčių kokias nors kitas uždaras celes, kurios gali būti suskaidytos tokiu būdu, kad ši uždara celė galėtų būti dar suskaidyta;
goal tikslas	apdorojamas (vykdomas) uždaras reiškinys, pvz., skaidomas;
father tėvas	uždaras reiškinys, turintis ne mažiau vieno identifikatoriaus savo vertės/žymėjimo lauke;
son sūnus	uždaras reiškinys, susijęs su kitu reiškiniu per jo identifikatorių, kuris žymi tą sūnų;

Sūnus taip pat gali būti ir tėvu. Tėvas taip pat gali būti sūnumi. Sūnus gali turėti daugiau, negu vieną tėvą. Tėvas gali turėti daugiau, negu vieną sūnų.

Closure position Uždaro reiškinio padėtis	rodo, ar reiškinys yra šaknis, ar susikirtimo (tinklelio) taškas;
root šaknis	pati viršutinė uždaro reiškinio celė, esanti reiškinio medyje;
node susikirtimo taškas	uždaro reiškinio celė, esanti reiškinio medyje ir nesanti šaknimi;
where kur	atminties celės laukas, užimantis uždaro reiškinio padėtį;
type tipas	tipo kodas, esantis uždaroje celėje, pvz., bito pavidalas, rodantis objekto savybę, pvz., komandos kodas;
lazy tingus	uždaros celės elementas, kuris rodo, ar galima su ja operuoti (vykdyti), ar tai reikia atidėti, arba ji yra neveikianti;
identifier identifikatorius	specialus uždaro elemento tipas, naudojamas saugomo atmintyje objekto pažymėjimui;
environment aplinka (periferija)	objektai gali būti grupuojami, suteikiant jiems tą pačią aplinką;
value/des. reikšmė/ žymėjimas	uždaras elementas, kuriame yra saugoma arba paprasta reikšmė, pvz., tiesioginis priskyrimas, arba nieko, arba žymėjimas, rodantis priklausymą kitam uždaram reiškiniui, pvz., netiesioginis priskyrimas;
core cell šerdies celė	aritmetinės struktūros vienetas pagal šį išradimą. Šerdies celė gali atlikti aritmetinius veiksmus, įskaitant

uždarų struktūrų skaidymą;

numeric ALU skaitmeninis loginis blokas	skaitmeninis aritmetinis blokas, galintis atlikti bazines skaitmenines arba logines operacijas. Centrinė ląstelė naudojasi skaitmeniniu ALU, atliekant skaitmenines operacijas;
full register pilnas registras	registro praplėtimas visame centrinės ląstelės plote;
core word šerdies žodis	pilno registro turinys, esantis centrinėje ląstelėje;
limited register ribotas registras	registras, esantis šerdies celės pratęsime, esančiame riboto ploto apimtyje, turintis apimtį, pakankamą priimti uždarą vertės/žymėjimo tipo celę;
element word elementarus žodis	riboto registro turinys arba pilna registro dalis, turinti tą patį praplėtimą, kaip ir ribotas registras;
num word skaitmeninis žodis	elementaraus žodžio dalis, rodantis reikšmę arba žymėjimą;
tag word požymio žodis	elementaraus žodžio dalis, turinti etiketę, rodančią reprezentacijos požymį skaitmeniniame žodyje;
reduction skaidymas (sumažinimas)	uždaro reiškinio perrašymas / performavimas pagal tam tikros naudojamos programavimo kalbos taisykles;

ĮSRADIMO SUVESTINĖ

Pagrindinis išradimo tikslas yra pasiekiamas aritmetinio struktūrinio proceso metodo dėka, į kurį įeina:

a) duomenų žodžių įvedimas į atmintį daugelyje registrų, kiekvienas duomenų žodis turi informacinę dalį,

b) minėti duomenų žodžiai yra surikiuoti sąrašuose, kiekvieną iš minėtų sąrašų įvedant į atmintį nustatytame kiekyje registrų kiekyje,

ir

c) kiekvienam duomenų žodžiui suteikiant žymės dalį, įskaitant žymę, rodančią, ar šis registras yra naudojamas, ar ne, kiekviena žymė kiekvieno minėto žodžio , esančio minėtuose sąrašuose, saugomuose minėtuose registruose, pažymimi vartojimo metu, parodant, kad vienas iš minėtų sąrašų turi bent dalį, saugomą tam tikrame . registre ir kad minėtas sąrašas, turintis dalį, saugoma tam tikrame registre, turi sąrašo komandą apie tai, kokio tipo tai yra sąrašas,

d) minėtus sąrašus, saugomus minėtuose registruose, sutvarkant, kaip sąrašų medį, kuriame vienas iš sąrašų yra šaknies sąrašas, kur santykis tarp minėtų sąrašų yra akivaizdus iš minėtų sąrašų esamos tvarkos minėtuose registruose,

e) minėtus registrus valdant, panaudojus minėto sąrašo komandas, kurios yra saugomos minėtuose sąrašuose, esančiuose minėtuose registruose, kad minėti sąrašai būtų pertvarkyti tarp minėtų registrų ir registrų turinio įvedimui/išvedimui atitinkamai pagal minėtas sąrašo komandas,

minėtoms sąrašo komandoms esant struktūrinės aritmetikos operatoriams, atitinkantiems aukšto lygio programavimo kalbų konstrukcijas.

Į aritmetikos bloką struktūrinės aritmetikos veiksmams pagal šį išradimą įeina:

a) bent viena įėjimo/išėjimo priemonė (v0, v1, v2, v3, env) duomenų sąrašų įvedimui ir išvedimui,

b) keletą registrų ($S_{0.0}$ iki $S_{3.3}$, F0 iki F3, ID, ENV), kiekvienas iš kurių yra pritaikytas saugoti duomenų žodį, minėti duomenų žodžiai surikiuoti minėtuose sąrašuose, kiekvienas duomenų žodis turi informacinę dalį, kiekvienas iš minėtų sąrašų saugomas nustatytame kiekyje minėtų registrų, ir kad

c) kiekvienas duomenų žodis turi žymės dalį, į minėtą žymės dalį įeina žymė, rodanti, ar šis registras yra naudojamas, ar ne,
kiekvieno minėto registro minėta žymės dalis pažymėta naudojimo metu, parodant, kad vienas iš minėtų sąrašų turi bent dalį, saugomą šiame registre ir kad minėtas sąrašas, turintis dalį, saugomą minėtame šiame registre, turi sąrašo komandą, rodančią, koks tai yra sąrašas,

d) minėti sąrašai, saugomi minėtuose registruose, yra sutvarkyti, kaip sąrašų medis, iš kurių vienas sąrašas yra šaknies sąrašas, o santykis tarp minėtų sąrašų yra akivaizdus iš minėtų sąrašų buvimo tvarkos minėtuose registruose,

e) valdymo priemonės (6p), valdyti minėtus registrus ir naudoti minėtas sąrašų komandas, saugomas minėtuose sąrašuose, kurie, savo ruožtu, yra saugomi minėtuose registruose, kad būtų pertvarkyti minėti sąrašai tarp minėtų registrų ir tam, kad registrų turinys būtų įvedamas/išvedamas atitinkamai pagal minėtas sąrašo komandas, šios minėtos sąrašo komandos yra struktūrinės aritmetikos operatoriai, atitinkantys aukšto lygio programavimo kalbų konstrukcijoms.

Pageidautina, kad sąrašai būtų sutvarkyti, saugant juos minėtuose registruose, kaip sąrašų medį, vienas iš kurių yra šaknies sąrašas. Saugomų atmintyje sąrašų medžio identifikatorius yra atskiras identifikavimo registras, taip pat įvestas į atmintį. Saugomų atmintyje sąrašų medžio aplinka gali būti saugoma atskirame aplinkos registre. Medžio šaknies sąraša pageidautina patalpinti skirtinguose registruose priklausomai nuo minėto įvedamo į atmintį medžio lygio. Kai kurie registrai yra suskirstyti į bazinių registrų matricą, įskaitant pagrindinių registrų eilę. Medį, į kurį įeina tik vienas lygis, pageidautina įvėsti saugojimui į pagrindinį registrą. Medį, į kurį įeina du lygiai, pageidautina įvesti saugojimui į minėtą pagrindinį registrą, o jo šakinius sąrašus - į bazinius registrus. Papildomas registrų rinkinys, vadinamas papildomais registrais, gali būti įrengtas už minėtos matricos ribų. Medį, į kurį įeina trys lygiai, pageidautina įvesti į minėto papildomo registro atmintį, o vieną iš jo elementų - minėtoje registrų matricoje.

Sąrašų medžio šaknies sąrašas gali būti padalintas į elementus, kur iš pirmojo šaknies sąrašo elemento gali būti gauta informacija valdymo priemonėms, kokio pobūdžio yra atliekamas suskaidymas, o kiti elementai rodo suskaidymui skirtus duomenis. Pirmajame minėto šaknies sąrašo elemente esanti informacija gali turėti komandos kodą, naudojamą valdymui, kai yra nustatoma, kokią reikia vykdyti komandos dalį, arba pirmajame minėto šaknies sąrašo elemente esanti informacija gali turėti sąrašų medžio šaknį, kurioje yra apibrėžiamos funkcijos, o esama informacija yra naudojama valdymo priemonėse, nustatant, kokių reikia imtis veiksmų, norint suskaidyti šaknies sąrašą.

Į informacija, esančią minėto šaknies sąrašo pirmame elemente, gali įeiti sąrašų medžio šaknis, kuri rodo funkcinius požymius.

Maksimalus žodžių, esančių sąraše, kiekis yra pageidautinas keturi. Maksimalus minėto sąrašų medžio gylis geriausiai yra

trijų lygių. Jei gylis yra trys lygiai ir jei minėto šaknies sąrašo, esančio minėtuose registruose, komanda rodo, kad minėtas šaknies sąrašas turi daugiau, negu vieną šaką, valdymo priemonė įveda į registrų atmintį tiktai vieną iš minėtų šakų.

Struktūrinis suskaidymas yra atliekamas su duomenų objektais, esančiais registruose, tokiuose, kaip bazinis registras arba baziniuose ir papildomuose registruose.

Duomenų stekai, esantys minėtuose registruose, pageidautina kad būtų sutvarkyti pjūvių forma, taip, kad kiekvienas steko elementas, turintis tą pačią vietą kiekviename duomenų steke, bitais būtų sujungiamas su kiekvienu kitu tokioje plokštumoje, į kurią įeina visi stekų elementai, priklausantys visiems minėtiems registrams, turintiems steko elementą šitoje vietoje. Kai kurie iš registrų turi ilgesnius stekus, negu kiti, taigi, kai kurios plokštumos turi tik steko elementus, priklausančius ilgesniems registrams.

Pagaliau, kai kurie registrai, vadinami baziniais, yra surikiuoti eilutėmis ir stulpeliais matricose N kartų, N registruose, kur N yra sveikas skaičius. Minėtų bazinių registrų stekų elementai yra tarpusavyje sujungti bitais. Kiekvienas minėto bazinio registro steko bitas yra prijungtas prie stulpelio šynos, o kiekviena eilutė - prie eilutės šynos. Kiekvienai stulpelio šynai ir kiekvienai eilutės šynai yra numatytas valdomas jungiklis, esantis kiekviename eilučių ir stulpelių, turinčių tuos pačius numerius, susikirtimo taške. Kiekvienas bazinis registras turi valdomą registro jungtį bent su artimiausia eilutės šyna ir su stulpelio šyna. Jungtis yra tarp gretimų bazinių registrų, tiek išilgai minėtų eilučių, tiek ir išilgai minėtų stulpelių. Valdymo priemonės valdo jungiklius ir registrų jungtis ir taip sudaro vieną iš trijų rūšių sujungimų, priklausomai nuo reikiamos vykdyti komandos:

a) paprasta jungtis nuo vieno registro į kitą viena kryptimi,

b) dvi atskiros jungtys tarp registrų, po vieną bet kurioje iš krypčių,

c) bendra laiko jungtis tarp registrų, per kurią yra perduodami esantys saraše elementai viena kryptimi ir kita kryptimi dviem sekančiom fazėmis.

Kiekviena celė, esanti registro steke, pageidautina, turi:

a) vidinį vieno bito registrą,

b) ne mažiau vienos vidinės vieno laido šynos, sujungtos su minėtu vieno bito registru,

c) ne mažiau vienos vidinės valdomos jungties, į kurią įeina jungtis, valdoma iš minėtų valdymo priemonių, įgalinant sujungti minėtą vieno laido šyną su vienu iš sekančių elementų: su šyna, esančia už celės ribų, viena iš celių, priklausančių kitam iš minėtų registro stekų. Ne mažiau, kaip vienas vidinis vieno bito registras gali turėti įvedimo buferį, tokį, kaip invertorius, ir išvedimo buferį, tokį, kaip invertorius bei valdomą jungiklį, sujungtą su minėtais buferiais. Įvedimo buferis ir minėtas išvedimo buferis per valdomus jungiklius atskirai yra sujungiami su ne mažiau, kaip viena iš minėtų vieno laido šyna.

Priklausomai nuo minėtų registrų padėties, kai kurios iš registrų celių turi fiksuotas reikšmes, pastoviai sujungtas su ne mažiau, kaip viena jos vidine jungtimi.

Komparatorius gali būti sujungtas taip, kad galėtų sulyginti minėtos dalies registrų turinį ir sulyginimo rezultatą pateikti į išorinės šynos buferį, vadinamą išrinkimu.

TRUMPAS BRĖŽINIŲ APRAŠYMAS

Šio išradimo pilnam suprtimui ir tolesniam tikslų ir pranašumų paaiškinimui yra duodamos nuorodos į sekančius aprašymus prie pridedamų brėžinių, kur:

FIG. 1 yra struktūrinė suskaidymo įrenginio schema, į

kurią įeina šerdies celės pagal išradimą schema;

FIG. 2A yra atminties celės, kurioje gali būti saugomas uždaras reiškinys, įgyvendinimas;

FIG. 2B iki 2D rodo šerdies registrus, kurie gali būti naudojami, įgyvendinant šerdies registrus pagal šį išradimą;

FIG. 2E rodo registrų šerdies galimą struktūrą, įgyvendinant registrų šerdį;

FIG. 3A iki 3F rodo skirtingas duomenų atminties formas šerdies celėse pagal šį išradimą;

FIG. 4, 5 ir 6 rodo šerdies celės darbo pavyzdį pagal šį išradimą;

FIG. 7 rodo darbinės plokštės, esančios šerdies celėje, pirmo įgyvendinimo blok - schemą;

FIG. 8 rodo pilnos registrų celės konstrukcijos įgyvendinimo schemą, turinčią visus galimus sujungimų tipus, skirtą pirmo tipo operacinei plokštei, parodytai FIG. 7.

FIG. 9A iki 9F parodyti įvairių tipų dalių, naudojamų registrų celėse, pavyzdžiai;

FIG. 10 ir 11 rodo įvairių registrų celių konstrukcijos schemas pirmo tipo operacinei plokštei, parodytai FIG.7;

FIG. 12 rodo antrojo tipo operacinės plokštės (požymių plokštės) pirmojo įgyvendinimo šerdies celėje pagal šį išradimą, blok-schemą;

FIG. 13 iki 15 rodo antrojo tipo operacinės plokštės, parodytos FIG 12, skirtingų registro celių konstruktyvinę schemą;

FIG. 16 ir 17 parodyti du perėjimo tarp registro celių schemos pavyzdžiai;

FIG. 18 iki 24 parodyti perdavimo šerdies celėje skirtingi pavyzdžiai;

FIG. 25 parodyta pirmojo tipo operacinės plokštės, esančios šerdies celėje, antrojo įgyvendinimo pagal šį išradimą, blok-schema;

FIG. 26 parodyta antrojo registro celės, esančios operacinėje plokštėje, įgyvendinimo konstruktyvinė schema;

SIŪLOMO ĮGYVENDINIMO APRAŠYMAS

Į atskiro suskaidymo procesoriaus įgyvendinimo schemą įeinanti šerdies celė yra parodyta FIG 1. Tai yra , palyginti, paprastas skaidymo procesorius, kuris gali būti naudojamas, kaip sudėtingesnio skaidymo procesoriaus dalis. Šio skaidymo procesorius struktūrinė schema matoma FIG.1. Tačiau reikia pažymėti, kad jo konstrukcija gali būti visiškai skirtinga, pavyzdžiui, šerdies celės atskiros plokštės gali būti sumontuotos greta viena kitos bendroje mikroschemoje, arba jos gali būti atliktos lygiagrečiuose sluoksniuose, tačiau kitaip, negu parodyta schematiškai.

Į skaidymo procesorių (FIG. 1) įeina širdies celė 2p, turinti keletą šerdies registrų 3p, objekto atmintį 4p, įskaitant daugybę atminties celių, kiekviena iš kurių galinti turėti atmintyje uždarą reiškinį ir galinti atlikti ribotą kiekį aritmetinių skaidymo veiksmų ir jei skaidymo procesorius (FIG.1) yra dalis sudėtingesnio procesoriaus, į kurį įeinà keletas skaidymo procesoriaus, dalis. Tokiu atveju, jis dar turi procesorių tinklo duomenų perdavimo priemones 5p, sudarančias ryšį tarp daugelio procesorių. Į duomenų perdavimo priemonę 5p įeina keletas registrų, pritaikytų laikinam uždaro reiškinio saugojimui tam, kad šis reiškinys būtų perduotas į kitą vienetinį procesorių. Kadangi ši priemonė 5p nėra tikra šerdies celės dalis pagal šį išradimą, toliau ji nebus aprašoma.

Valdymo blokas

Elementų, esančių šerdies plokštėje 2p, valdymui į procesorių taip pat įeina valdymo blokas 6p. Be to, į procesorių įeina skaitmeninis ALU 1p. Valdymo blokas 6p yra loginis ventilinis blokas, kuris valdo uždarų reiškinių turinį, kuris yra tvarkomas šerdies celėje. Nei valdymo blokas 6p, nei skaitmeninis ALU 1p nėra šio išradimo dalimis. Tokiu būdu, jie smulkiau nebus aprašomi, bus aprašoma tik signalai, einą iš jų arba į juos ir turį ryšį su toliau aprašomu išradimo įgyvendinimu. Platesnės informacijos apie valdymo blokų konstrukciją galima gauti iš knygos : Carver Mead and Lynn Conway "Introduction to VLSI Systems", Addison Wesey Publishing Company, 1980.

Objekto atmintis

Šerdies registrai turi struktūrą, įeinančią į objekto atmintį 4p. Objekto atmintis gali atmintyje turėti tik vieno lygio struktūrą, pvz., uždarą celę. Šerdies registrai yra sujungti su objekto atmintimi per šyną, kuri yra pakankamai plati vieno lygio struktūros perdavimui. Tačiau šerdies celė gali priimti ir laikyti iki trijų lygių objekto struktūros. Gali būti keturi atvejai: 0, 1, 2 arba 3 lygio objektas gali būti įvestas į šerdies celės atmintį. Kaip bus paaiškinta vėliau, jei yra įvedama į atmintį trijų lygių objekto struktūra, yra įvedamas tik jo viršutinis lygis (šaknis) ir viena iš jo šakų. Kitais atvejais yra įvedami į atmintį visi lygiai.

Šerdies celės struktūra

Šerdies celė visumoje yra aritmetinis prietaisas, skirtas struktūrinės aritmetikos veiksmams, turintis keletą šerdies registrų 3p. Šerdies registrai saugo atmintyje sąrašų medį. I kiekvieną sąrašą įeina žodžiai. Kai kurie iš registrų yra riboti registrai ir gali turėti atmintyje elemento žodį, į kurį įeina skaičiaus žodis ir žymės žodis. Žymės žodis rodo skaičiaus žodžio

priskyrimo savybes. Pavyzdžiui, jei skaičiaus (num) žodis atstovauja reikšmę, žymės žodis rodo reikšmės tipą, turimą skaičiaus žodyje, pavyzdžiui, ar tai yra sveikas skaičius, ar plaukiojančio kablelio skaičius ar pan. Šerdies celė gali tik apdoroti vienu metu tik tam tikro ilgio sąrašą, priklausomai nuo to, kiek joje yra registrų. Registrų skaičius, naudojamas sąrašo įvedimui į atmintį, yra pageidautinas keturi, kas reiškia, kad įvedamas į atmintį sąrašas gali turėti keturis elementus arba mažiau. Pavyzdžiui, į registrą gali įeiti skaičiaus žodis, užimantis 32 bitus ir 6 bitų žymės žodis. Šiuo atveju pageidautina, kad sąrašas, turintis keturis elementus, užimtų 4 kartus po 38 bitus.

Gali būti tvarkomi ir palyginti ilgi sąrašai, tačiau tokiu atveju, jie turi būti suskaidomi į keletą sąrašų, kiekvienas iš kurių turi ilgį, sutampantį su arba trumpesnį, negu maksimalus ilgis, kuris turi būti apdorojamas šerdies celeje. Šerdies celė vienu metu gali apdoroti medį, turintį tik tam tikrą ilgį. Gali būti tvarkomi medžiai, turintys ir didesnį ilgį, tačiau tik dalis medžio, turinčio ribotą gylį, yra įvedama į šerdies celę vienu metu, t.y., vienu metu yra įmanoma tvarkyti tik dalį medžio.

Nustatytas skaičius registrų gali įvesti į atmintį sąrašą. Nenaudojami registrai yra ypatingai pažymimi, kaip nenaudojami. Sąrašų medis yra naudojamas valdyti ir atlikti skaičiavimus, ir tai yra atliekama valdant iš valdymo bloko 6p ir, jei reikia, bendroje sąveikoje su skaitmeniniu ALU 1p. Skaičiavimai yra atliekami, turinį perrašant į sąrašų medį.

Interfeisas: objekto atmintis < - > šerdies celė

Objekto atmintis, kurios vietoje tinkamiausia yra asociatyvinė atmintis, ir šerdies celė yra sujungtos tarpusavyje per transformatorinį interfeisą 9p, tuo adaptuojant signalą ir

uždarą reiškinį perduodant į plačią šyną 8p, t.y., į šyną, į kurią įeina dalinė šyna OBJv (vertikali objekto šyna), sujungta

su registro plokštėmis NUM ir daline šyna TAG, sujungta su registro plokštėmis HEAD, tokiu būdu, šyna galės perduoti vieno lygio struktūrą. Interfeisas 9p stiprina ir perdirba signalus, ateinančius iš šerdies registrų 3p per plačią šyną OBJv 8p, į signalus, tinkamus perduoti į atminties celes, esančias objekto atmintyje. Jis taip pat ir stiprina ir transformuoja signalus, ateinančius iš objekto atminties į nuskaitymo operacijas, pritaikant šerdies celės registrams. Nors interfeisas yra parodytas esas objekto atmintyje 4p, jis taip pat gali būti patalpintas ir šerdies celėje. Tačiau, interfeisas 9p neturi būti laikomas šerdies celės dalimi pagal šį išradimą ir todėl toliau jis nėra aprašomas.

Šerdies celės plokštės

Kaip matoma iš FIG.1, šerdies registrai yra atskirti į keletą dalių, parodytų, kaip plokštės NUM, HEAD, BOOL, TYPE, WHERE, LAZXY, CLOS/SIMPLE. Skirtingos dalys turi skirtingus plokščių numerius. Tik keletas plokščių yra parodytos FIG.1 dėl aiškumo. Esančios žemiau plokštės TYPE, WHERE, LAZXY ir CLOS/SIMPLE yra vadinamos papildomomis (ATTRIBUTE) plokštėmis.

Skaičiaus žodis gali būti saugomas registro plokštėse NUM, o žymės žodis būti saugomas registro plokštėse HEAD. Pavyzdžiui, gali būti 32 NUM plokštės ir 6 HEAD plokštės.

Pavyzdžiui, yra penkios registro plokštėse TYPE, viena registro plokšteWHERE, dvi registro plokštės LAZY ir viena registro plokšte CLOS/SIMPLE toje procesoriaus konstrukcijoje, kuri yra parodyta FIG.1. Informacija, kuri yra pateikiama į šerdies registro dalis, esančias šioje plokštėje, bus aiški iš toliau einančio aprašymo, kuriame yra aprašomas šerdies celės darbas ir kuris pateikiamas prieš jo pilną aprašymą.

ŠERDIES CELĖS DARBAS

Tuo tikslu, kad bptų padarytas pamatas šerdies celės sukūrimui, mes aprašome tam tikrą objekto atmintį 4p, kuri buvo sukurta prieš sukuriant šerdies celę pagal šį išradimą ir toliau yra aprašoma mūsų pateiktose paraškose Nr PCT/SE91/00513, PCT/SE91/00516 ir PCT/SE91/00517. Tačiau reikia pastebėti, kad aritmetinis blokas, skirtas struktūrinės aritmetikos veiksmams pagal šį išradimą, taip pat gali būti sujungtas su įprasto tipo skaičiavimo ir atminties įtaisais. Reikia pastebėti, kad šerdies celė yra įtaisas, atliekantis tik struktūrinius aritmetikos veiksmus, o skaitmeniniams veiksmams atlikti ji naudoja skaitmeninį aritmetinį bloką (skaitmeninį ALU). Paprastai, procesorius yra montuojamas kartu su aritmetiniu loginiu bloku (ALU), atliekančiu tiek skaitmenines, tiek ir dalį struktūrinių operacijų ir šis ALU nėra suskirstytas į skirtingas funkcines dalis, skirtas struktūrinėms ir skaitmeninėms operacijoms. Naudojant įprastus ALU, reikia turėti nedidelę programą, kad būtų galima atlikti ekvivalentines operacijas šerdies celėje.

Objekto atminties galimybės

Objekto atmintis 4p turi iš esmės daugiau intelektinių savybių, negu įprasto RAM tipo atmintis. Ji yra asociatyvi, kas daro galimu teikti daugiau paslaugų, negu tik "skaityti" arba "įrašyti", kaip yra įprastinio tipo RAM atmintyje.

Kaip jau minėta aukščiau, objekto atmintis yra padalinta į atminties celes, kiekviena iš kurių turi keletą atminties elementų. Teikiamos paslaugos yra aukšto lygio. Pavyzdžiui, yra galima surasti visus įvykius su tam tikru duomenų elementu, kokioje jis būtų individualioje saugojimo ląstelių vietoje ir perrašyti tam tikrą rastą duomenų elementą globaliai, pvz., visoje objekto atmintyje, į naują reikšmę, naudojant vien tik vieną atminties komandą. Kadangi objekto atmintis yra asociatyvi,

ši perrašymo operacija gali būti atlikta dviem fiziniais atminties ciklais, nepriklausomai nuo liečiamų atminties ląstelių skaičiaus.

Atminties celė

Atminties celės realizavimas yra schematiškai parodytas FIG. 2A. Joje gali būti saugomi dviejų tipų elementai ir ji turi atminties laukus, ypatingai pritaikytus saugomiems elementams. Šie laukai FIG. 2A yra pažymėti tais pačiais pavadinimais, kaip ir juose saugomi elementai.

Pirmojo tipo uždari elementai

Pirmojo tipo elementai aprašo skirtingus atminties celių būvius. Vienas šio tipo elementų yra LAZY (tingus, neveikiantis), kuris rodo, ar ląstelė yra tuščia , tuo atveju jos likęs turinys yra laikomas pasyvia informacija, exec, t.y., yra vykdymo būvyje arba laukimo, t.y., celės įvertinimas yra atidėtas ir ji laukia rezultatų prieš tai, kai ji gali būti vykdoma. Kitas pirmo tipo elementas yra TYPE, į kurį įeina tipo kodas (par, seq, apply, list, unify ir tt.). Šie pirmojo tipo elementai yra pritaikyti įvedimui į atmintį šerdies registrų dalis, perduodamas plokštėse LAZY, WHERE ir TYPE. Tačiau šerdies celė turi papildomą plokštę, vadinamą CLOS/SIMPLE, į kurią yra priskiriama informacija, jei ji informacija registre yra uždaras reiškinys arba paprasta reikšmė. Priklausomai nuo pritaikymo, gali būti montuojamos papildomos plokštės.

Antrojo tipo uždari elementai

Antrojo tipo elementai nurodo identifikavimą, aplinką arba reikšmę. Tai yra IDENTIFIER, ENVIROMENT, VALUE/DES. Šie antrojo tipo elementai yra pritaikyti įvedimui į atmintį šerdies registrų dalis, perduodamas iš plokščių HEAD ir NUM. Į kiekvieną iš šių

elementų įeina elemento žodis, kuris , savo ruožtu, yra padalintas į žodį num (skaičiaus), saugomo plokštėje NUM šerdies celėje,

ir žymės žodį, saugomą plokštėse HEAD, šerdies celėje. Priklausomai nuo pritaikymo, gali būti montuojamos papildomos plokštės.

Kiekvienas antrojo tipo uždaras elementas turi požynio žodį, kuris rodo num žodžio savybę. Požymiai - etiketės yra dviejų tipų, netiesioginės etiketės, pvz., etiketės, naudojamos identifikatoriams ir aplinkai ir tiesioginės etiketės, naudojamos paprastoms reikšmėms ir pan.

Netiesioginių etikečių pavyzdžiai yra cls, canon ir open. Jei etiketės žodis yra cls, tai reiškia, kad num žodis atstovauja uždarą reiškinį, kuris gali būti skaidomas. Jei etiketės žodis yra canon, tai reiškia, kad num žodis atstovauja uždarą reiškinį, kuris negali būti toliau skaidomas. Jei etiketės žodis yra open, tai reiškia, kad num žodis atstovauja uždarą reiškinį, kuris yra įterptas sąrašas.

Tiesioginių etikečių pavyzdžiai yra discr, cont, unused ir nothing. Jei etiketės žodis yra discr, tai reiškia, kad num žodis yra sveikas skaičius. Jei etiketės žodis yra cont, tai reiškia, kad num žodis yra plaukiojančio kablelio reikšmė. Jei etiketės žodis yra unused, tai reiškia, kad num žodis reiškia nieką, pvz., uždaro reiškinio unifikavimas, turintis lauką, pažymėtą nothing, visada bus niekas. Priklausomai nuo poreikio, gali būti panaudojami ir papildomi tiesioginiai arba netiesioginiai požymių žodžiai.

Identifikatoriai

Jei atminties celėje yra identifikatoriaus elementas, uždara celė iš šios ląstelės gali būti perduodama į šerdies celę. Kiekviena iš atminties celių laukų VALUE/DES gali turėti

identifikatorių, rodantį kitą uždarąjį elementą, tokiu būdu sudarant ryšį su ta kita uždara cele. Atmintyje esančių uždarų celių rinkinys gali būti matomas, kaip tiesioginis šių celių medis, siejamas identifikatoriais.

Aplinka

Į aplinkos laukus gali įeiti identifikatorius, rodantis uždaro reiškinio šaknį sistemos dalyje, pvz., uždarų reiškinių medyje, sudarančiame aplinką. Tačiau aplinkos laukas taip pat gali turėti ir kitą panaudojimą. Aplinka gali būti panaudota išlaikyti struktūros kūrimo taką (pėdsakus), įvedant į visų sukurtų uždarų ląstelių aplinkos atmintį kuriamą identifikatorių. Pavyzdžiui, visos uždaros celės, esančios submedyje, kuriose visi simboliai, turintys tuos pačius pavadinimus , reiškia tą patį, gali būti sugrupuojamos su ta pačia aplinka. Tokiu būdu, visa struktūra gali būti išrenkama per vieną uždarą reiškinį medyje, per šaknį, vien tik vienos operacijos pagalba.

Tokiu būdu, jei uždaro reiškinio aplinka yra duota, šioje aplinkoje gali būti surasta uždaro reiškinio šaknis. Uždaros aplinkos šaknis gali turėti ypatingą žymę (pvz., "1"). Aplinkos uždaro reiškinio susikirtimo taškas gali turėti kitą žymę (pvz., "0") lauke WHERE.

Šerdies celės registrai

Registrai, kurie gali būti panaudoti, įgyvendinant šerdies celę yra parodyti FIG. nuo 2B iki 2D. Registrų struktūra (konfigūracija), kuri gali būti panaudota, raelizuojant šerdies celę, yra parodyta FIG. 2E.

FIG. 2B yra parodytas registras. Brėžinyje iliustruojama, kad registrai yra sudaryti iš registrų celių, kiekviena iš celių gali turėti atmintyje viena informacijos bitą. Tas būdas, kuriuo yra parodytas registras, parodo, kad registras tęsiasi per skirtingas šerdies celės matricas, kiekviena registro celė yra vienoje iš plokščių.

FIG. 2C parodytas registras, kuris tęsiasi per visas šerdies celės plokštes, t.y., pilnas registras. Šis registro tipas savo elementuose gali turėti identifikatorių arba reikšmę, kuri yra registro celėse NUM ir HEAD. Ten taip pat gali būti nurodytas būvis, kuris buvo aprašytas aukščiau, esantis registro celės plokštėse BOOL, TYPE, WHERE, LAZY ir CLOS/SIMPLE.

FIG. 2D parodytas registras, kuris yra tik NUM ir HEAD šerdies celės plokštėse t.y., ribotas registras.

FIG. 2E yra parodyta galima registrų struktūra šerdies celėje. Šerdies celė turi bazinius registrus, geriausiai, išdėstytus kvadratu, vadinamu bazinio registro matrica. Baziniai registrai yra išdėsyti pagrindinėje eilėje, išilgai vienos iš jų šonų, vadinamoje pagrindinių registrų pusėje. Pagrindinių registrų stulpeliai, kiekviename iš kurių yra vienas pagrindinis registras stulpelio apačioje, yra vadinami papildomais registrais. Pagrindinė plokštė taip pat gali turėti identifikatoriaus registra ir aplinkos registra. Bazinės registrų matricos šone yra išdėstyti papildomi registrai.

Šerdies celės realizavimo schemoje papildomi registrai gali būti tokio pavidalo, kaip yra parodyta FIG. 2D, t.y., riboti registrai, o likusieji registrai, parodyti FIG. 8D, gali būti tokie, kaip parodyti FIG. 2C, t.y., pilni registrai.

Trumpas duomenų įvedimo į atmintį formų aprašymas bus duotas, kalbant apie FIG. nuo 3A iki 3F, o kai kurie jų veikimo pavyzdžiai bus duoti, aprašant FIG. 4A iki 4H, 5A iki 5G ir 6A iki 6G.

Pprasta reikšmė

Kaip parodyta FIG. 3A, paprasta reikšmė 25, kaip suskaidymo rezultatas yra tam tikrame registre, pagrindinių registrų zonoje.

Vieno lygio struktūra

Tikslą sudaro tai, kas yra įvedama į šerdies celę tolesniam suskaidymui. Kaip parodyta FIG. 3B, tikslas, turintis tik vieną lygį, sudarantis tipinį uždarą reiškinį, be kreipimosi į kitas uždaras struktūras, yra saugomas pgrindiniame registre. Pavyzdys parodo paprastą aritmetinę operaciją, t.y., reikšmių 1, 2 ir 3 sudėtį. Skaitmeninė komanda (+) yra saugoma pirmame pagrindiniame registre, o elementai, kuriuos reikia apdoroti, yra įvedami į kitus pagrindinius registrus.

Dviejų lygių struktūra

Kaip parodyta FIG. 3C, medis, į kurį įeina dviejų lygių struktūra, gali turėti savo pagrindinį sąrašą, kuris yra vadinamas tėvu ir yra įvestas į pagrindinius registrus horizontaliai ir sąrašus, kurie vadinami sūnumis, vertikaliai baziniame registre. Šiame pavyzdyje struktūra, turinti sąrašo pavidalą ((1 2) (3 4)) yra įvedama į atmintį bazinio registro matricoje. Pagrindinis sąrašas, pvz., 1 ir 3, esantis pirmu elementu sąraše, yra laikomas pagrindinio registro atmintyje, o sūnų sąrašai, t.y., (1 2) ir (3 4), yra saugomi vertikaliai papildomuose registruose. Pavyzdys, kuriame toks įvedimo į atmintį būdas yra naudojamas, bus pateiktas toliau, žr. FIG. 4.

Trijų lygių struktūra

Kaip parodyta FIG. 3E, medis, į kurį įeina trijų lygių struktūra turi savo šaknį, įvestą į papildomus registrus, o jo vienintelis sūnus yra įvestas į pagrindinį registrą. FIG. 3D parodyta tikslo medžio šaknis, kurioje yra perstatymo komanda (Tr), įvesta į papildomo registro atmintį, o jos sūnus turintis sąrašo pavidalą (id1, id2, id7) yra įvedamas į atmintį baziniuose registruose. Kiekvienas šio sąrašo elementas yra savo ruožtu identifikatorius, rodantis sūnų. FIG. 3E šie sūnūs yra įvesti į bazinius registrus vertikaliai, kur id1 yra pakeičiamas į sąrašą, kurį jis žymi, pvz., (1 2 3), kur id2 yra pakeičiamas į sąrašą, kurį jis žymi, pvz., (11 12 13) ir kur id7 yra pakeičiamas sąrašu, kurį jis žymi, t.y., (21 22 23).

Magistralinis režimas

Kaip parodyta FIG. 3F, medis, kuris yra įvestas magistraliniame režime, yra įvedamas į pagrindinius registrus kartu su tikslo sąrašu, o tikslo tėvas - į papildomus registrus, be to, abiejuose registruose yra įvesta vykdymo komandos ir jų elementai. Operacinį magistralinį režimą geriausiai naudoti, skaidant skaitmeninius reiškinius. Vienas iš pranašumų yra tai, kad tarpiniai rezultatai gali būti laikinai įvesti į atmintį šerdies celėje, o ne objekto atmintyje.

1 PAVYZDYS

Pirmas pavyzdys, parodytas FIG. 4A iki 4H, yra lygiagrečių reikšmių unifikavimas, kuris yra pateikiamas, kaip suskaidomas uždaras reiškinys.

unify(par(1 par(1) 3) par(1 par(1) 2))

kas sudaro suskaidomą uždarą reiškinį, kuriame turi būti atlikti

keli lygiagretūs unifikavimai. Šis skaidomas reiškinys turi būti perrašytas, kaip unifikuota lygiagreti struktūra.

FIG. 4A parodytas vidinis suskaidomas uždaras reiškinys. FIG. 4B parodyta, kaip šis reiškinys yra įvedamas į objekto atmintį. Atminties ląstelės, į kurias yra įvedamas skaidomas reiškinys, yra pažymėtos FIG. 4A. Ryšiai tarp uždaro elemento ir uždaros ląstelės yra pažymėti FIG. 4B. Uždara ląstelė, turinti identifikaciją id_1, turi etiketę cls ir turi tipo kodą unify, nurodytą tipo lauke, o uždaros celės, turinčios identifikaciją id_2, id_3 ir id_4 turi tipo kodą par, nurodytą tipo lauke. Uždara celė, turinti identifikaciją id_1, turi du reikšmės/žymėjimo uždarus elementus, turinčius identifikavimą id_2 ir id_4. Šios uždaros celės turi etiketę canon. Uždara celė, turinti identifikaciją id_2, turi pirmą ir trečią reikšmės/žymėjimo uždarus elementus, turinčius diskretines reikšmes, kurios pažymėtos etikete discr, o jos antras reikšmės/žymėjimo uždaras elementas, turi identifikaciją id_3 ir turi etiketę canon. Uždara celė, turinti identifikaciją id_3, turi pirmą reikšmės/žymėjimo uždarą elementą, turintį sveiką skaičių ir yra pažymėta etikete discr. Uždara celė, turinti identifikaciją id_4, turi pirmą ir trečią reikšmės/žymėjimo uždarus elementus, turinčius diskretines reikšmes, kurios pažymėtos etikete discr, o jos antras reikšmės/žymėjimo uždaras elementas turi identifikaciją id_3 ir turi etiketę canon.

Kaip parodyta FIG. 4C, atminties celių turinys, turintis uždarą celę, identifikuotą id_1, pirmiausiai yra įvedamas į pagrindinę plokštę, patalpinant jos identifikatorių į identifikavimo registrą, kaip id_1, įskaitant tipo kodą unify ir reikšmės/žymėjimo elementus, kaip tikslą, pagrindiniame registre, pirmame operacijos žingsnyje.

Kaip parodyta FIG. 4D, sūnūs, turintieji identifikatorius id_2 ir id_4, į bazinius registrus yra įvedami vertikaliai, taip, kad turinys, esąs jų pirmame reikšmės/žymėjimo elemente yra patalpinamas į pagrindinį registrą, pažymėtą jo identifikatoriumi, o likusieji reikšmės/žymėjimo elementai - į virš

esančius registrus. Į pagrindinį registrą taip pat yra įvedami kiekvieno iš sūnų tipo kodai par. Tipo kodas yra įvedamas į registro celes, esančias plokštėse TYPE.

Kaip parodyta FIG 4E, bazinių registrų turinys yra perkeliamas 90° tokiu būdu, kad jų pirmoje eilutėje esantis turinys būtų patalpinamas į pagrindinius registrus, o antras vertikalus stulpelis būtų patalpinamas bazinių registrų eilutėje, lygiagrečiai pagrindiniams registrams. Perkelimo operacija yra parodyta FIG. 23 ir aprašoma žemiau. Tipo kodai par ir unify, esantys identifikatoriaus registre, ir pagrindiniai registrai yra sukeičiami, tai yra atliekama automatiškai iš valdymo bloko. Dabar į bazinius registrus įeina tėvas, turintis tris sūnus, patalpintus į stulpelius.

Dabar kiekvienas sūnus yra pakraunamas atgal į objekto atmintį, naudojant komandą make. Kaip atsakymas, identifikatoriai iš objekto atmintyje įvestų sūnų yra perduodami į pagrindinių registrų atmintį. Galima pastebėti, kad valdymo blokas 6p, būdamas ventilinės skleistinės tipo, peržiūri plokščių CLOS/SIMPLE iki TYPE registrų turinį ir duoda komandas, t.y., valdo perjungėjus ir ventilius atitinkamai pagal randamą informaciją. Sūnūs yra pavadinti pagal eilės numerį, einantį po id_1, o jau užimti vardai nėra naudojami. Tačiau vardų tvarka nėra svarbi ir gali būti laisvai pasirenkama.

Kaip parodyta FIG 4F, pirmasis sūnus gauna žymę id_2, sekantis sūnus, turintis uždarus elementus, užimančius identifikatorių id_3, gauna žymėjimą id_4, trečias sūnus gauna žymėjimą id_5. Tėvas, turintis uždarus elementus, susijusius su uždaromis lastelėmis, turinčiomis identifikatorius id_2, id_4, id_5, gauna identifikaciją id_1, ir po to yra įvedamas į objekto atmintį.

FIG 4G yra parodyta lastelė, kurios atmintyje yra įvestas skaidomas reiškinys

par(unify(1 1) unify(par(1) par(1)) unify(2 3))

Pats skaidomas reiškinys yra parodytas FIG 4H. FIG 4G ir 4H viskas yra pavaizduota taip pat, kaip ir FIG 4A ir 4B, todėl aiškios savaime.

FIG 4G taip pat yra parodyta, kad uždaros celės, turinčios tipo kodą unify, gavo nuorodą exec zonoje LAZY. o uždara celė, turinti identifikatorių id_1, gavo nuorodą wait (laukti), kas reiškia, kad uždara celė, pažymėta exec, turi būti vykdoma prieš celę, pažymėtą identifikatoriumi id_1, kai yra atliekamas jų suskaidymas reikšmėmis.

Uždaras reiškinys, parodytas FIG 4H, vėliau gali laiko bėgyje būti pakrautas atgal į pagrindinę plokštę tolesniam apdorojimui. Pavyzdžiui, uždara celė, turinti identifikatorių id_2, turi reikšmę 1, nes reikšmė 1 ir 1, esanti reikšmės/žymėjimo elementuose, yra ta pati, o uždara celė, turinti identifikatorių id_3, pavirs į nothing, nes reikšmės 2 ir 3, esančios jų reikšmės/žymėjimo elementuose, nėra tos pačios.

Kiekvienas unifikavimas siūlomame realizavimo variante gali būti atliekamas skaitmeniniame ALU (aritmetinis - loginis įrenginys), kuriame reikšmės sulyginamos komparatoriais, o lyginimo rezultatai perduodami į valdymo bloką 6p. Valdymo bloke yra ventilinė loginė skleistinė, per kurią informacija yra pateikiama į pirmąjį pagrindinį registrą, esantį pagrindinėje plokštėje. Kai skaidymas baigiasi kanonine reikšme, paprasta reikšme arba nieku, ši informacija yra globaliai paskirstoma į visas atminties zonas objekto atmintyje, paruošta antro tipo uždarų elementų įvedimui. Tokiu būdu, kiekvienas netiesioginis priskyrimas reiškinio suskaidymui yra pakeičiamas tiesioginiu reišmės žymėjimu.

2 PAVYZDYS

Šis pavyzdys yra aparatinės dalies komandų sąrašo pratęsimas – list expansion, reiškiantis, kad uždara celė turi įterptą sąrašą. Šios rūšies komanda sudaro papildomą žingsnį kitų skaidymo operacijų metu. Aparatinės dalies komandų sąrašo praplėtimas yra parodytas FIG. 24 ir ten aprašytas.

Įrenginys atlieka skaidymą, pateiktą pavyzdyje, vadinamame ex.type, kuris gali būti komandos formoje, į kurią įeina reikšmės ir sąrašai, turintys pavidalą

ex.type(i list(2 3 list(4 5 6))7)

Ši forma yra parodyta FIG 5A, o jos vidaus celės – FIG 5B. FIG 5A ir FIG 5B yra pavaizduota taip pat, kaip ir FIG 4A ir 4B, todėl aiškios savaime.

Kaip parodyta FIG 5C, uždara celė, turinti identifikatorių id_1, yra įvesta į pagrindinį registrą pagrindinėje plokštėje, turinčioje jos identifikaciją ir tipo kodą, esantį identifikavimo registre. Kadangi turinys antrajame pagrindiniame registre yra pažymėtas netiesioginiu elementu open, uždara celė id_2, su kuria jis yra susijęs, yra pakraunama vertikaliai į bazinius registrus, kaip sūnus, tai yra matyti iš FIG 11D.

Po to aparatinės dalies komanda list expand perkelia diskretinę reikšmę 7 iš trečio pagrindinio registro į padėtį šalia id_4, esančią trečiame baziniame stulpelyje ir perkelia sąrašo dalį iš antrojo stulpelio, esančio virš antrojo pagrindinio registro į trečią stulpelį, patalpinant jo žemiausią elementą (reikšmę 3) į trečią pagrindinį registrą, suteikiant jam tipo kodą list. Kadangi antrojo pagrindinio registro turinys yra diskretinė reikšmė, jis turi etiketę discr.

Po to sudaromas sarašo praplėtimas, perkeliant trečio stulpelio turinį, esantį virš pagrindinio registro į ketvirtą stulpelį, nurodant jo tipą, kaip list (sarašas). Kadangi trečiojo pagrindinio registro turinys yra diskretinė reikšmė, jis turi etiketę discr, kas yra matoma iš FIG 5F.

Po to ketvirtame stulpelyje esantis sarašas yra įvedamas į objekto atmintį, naudojantis aparatine komanda make. Jis yra įvedamas į atminties celę, turinčią identifikatorių id_2, kadangi ji tapo tuščia, o identifikatorius id_2 yra persiunčiamas atgal į pagrindinę lastelę, kad būtų įvestas į ketvirto pagrindinio registro atmintį, kaip yra parodyta FIG 5G.

Tokiu būdu yra atliekamas ex.type suskaidymas ir tada, kai yra pasiekiamas kanoninis suskaidymo rezultatas, jis yra pakraunamas atgal į objekto atmintį.

3 PAVYZDYS

Turi būti atliekama skaitmeninė komanda. Skaitmeninė komanda gali būti +, −, *, /, during, ir tt. Po komandos seka argumentai. Šiame pavyzdyje yra atliekamas sudėties veiksmas tarp skaičių, esančių sąraše. Įrenginys atlieka apply (pritaikymas) suskaidymą pagal formą

apply(+ list(1 2))

Pritaikymas yra parodytas FIG 6A, o jo uždaros ląstelės – FIG 6B. FIG 6A ir 6B turi tuos pačius pažymėjimus, todėl schemos paaiškina viena kitą.

Kaip yra parodyta FIG 6C, į pagrindinį registrą, turintį savo identifikatorių ir tipo kodą, yra pakraunamos uždaros ląstelės, turinčios identifikatorių id_1. Kaip komanda, yra duodama nuoroda (+). Kadangi antrame pagrindiniame registre esantis turinys turi etiketę, kaip netiesioginis elementas open, uždara ląstelė, su kuria jis yra susijęs, yra pakraunama vertikaliai, kaip sūnus, į bazinius registrus, kaip matoma iš FIG 6D.

Po to yra atliekamas sąrašo praplėtimas, uždedant etiketę discr ant diskretinės reikšmės, esančios antrame pagrindiniame registre, ir sąrašo praplėtimo reikšmę 2 žymint list kodo tipo lauke. Tai yra daroma todėl, kad įrenginys atlieka tą pačią operaciją, nepaisant to, ar sąrašas, turintis identifikatorių id_2 turi du, tris ar keturis elementus. Kadangi naujame sąraše yra tiktai vienas elementas, įrenginys perkelia žymę list su nuoroda, kad pagrindinis registras turi reikšmę, kuri yra discr, kas yra matoma iš FIG 6F.

Be to, į pagrindinį registrą įeina komandos žymė (+) ir dvi diskretinės reikšmės, o tai priverčia valdymo bloką, kuriame yra saugomos komandos, valdyti skaitmeninį ALU, kad būtų atlikta komanda (sudėtis) ir pirmame pagrindiniame registre būtų gautas skaitmeninės operacijos rezultatas, kaip kanoninė reikšmė, matoma FIG 6G. Reikia pastebėti, kad nuoroda apply, esanti kodo tipo lauke yra žymėjimas, kad turi būti atlikta pritaikymo funkcija.

Rezultato reikšmė, šiuo atveju, paprasta reikšmė 3, po to yra perskirstoma globaliai tuo tikslu, kad būtų išvengtas bet koks identifikatoriaus id_1 atsiradimas ties ta reikšme.

ŠERDIES CELĖS APARATINĖS DALIES KONSTRUKCIJA

Plokštės NUM ir HEAD turi šerdies registro celes, sujungtas su šynos OBJv laidais, id šyna, t.y., identifikavimo šyna ir env šyna, t.y., aplinkos šyna, esančiomis tarp plokštės 2p ir interfeiso 9p. I šyna OBJv įeina šynos dalys v0, v1, v2 ir v3.

Minėtų plokščių paskirtis ir schema bus aprašytos vėliau.

Registro celių feritinė matrica yra, galima sakyti, perpjauta statmenai registrams į plokštes, o registro celės, priklausančios toms pačioms NUM arba HEAD plokštumoms, tačiau skirtingiems šerdies registrams, viena su kita yra sujungtos tokiu būdu, kaip yra parodyta FIG. 7.

NUM ir HEAD plokščių konstrukcijoje, partodytoje FIG.7, registro celės kvadratas yra sumontuotas matricoje, turinčioje N x N registrų, nuo $S_{0,0}$ iki $S_{N-1,N-1}$. Registro celė, esanti šioje matricoje, yra vadinama baziniu registru, o registro celės yra vadinamos bazinio registro celėmis.

Baziniai registrai daugeliu atvejų yra naudojami laikinam uždarų elementų saugojimui. Registrų žymėjimas yra griežtai skirstomas tuo požiūriu, kad viena žymėjimo rūšis, tokia, kaip bazinis, pagrindinis ir papildomas registras, yra naudojama tada, kai aprašymas yra skirtas tikrai registro padėčiai, o kita rūšis, tokia, kaip sūnaus, tikslo ir tėvo registras yra naudojama tada, kai aprašymas yra skirtas registro funkcijos nurodymui.

Nurodytame pavyzdyje N = 4 yra pageidautina reikšmė, tačiau kitų matricų dydžiai gali būti pasiorenkami kitokie (neparodyta).

Žemiausia bazinių registrų celių eilutė $S_{0,0}$, $S_{1,0}$, $S_{2,0}$ ir $S_{3,0}$, kaip parodyta FIG. 7, yra sujungta per vieną laidą su šyna h0, skirtą tai plokštei, ir ši eilutė sudaro pagrindinių registrų celes. Pagrindinių registrų celės $S_{0,0}$, $S_{1,0}$, $S_{2,0}$ ir $S_{3,0}$ yra dažniausiai naudojamos, kip tikslo šaknies registrai ir yra sujungti su skaitmeniniu ALU ip per šyną NU, kuri susideda iš laidininkų NU0 iki NU3. Tačiau reikia pastebėti, kad yra galima sukurti paprastus procesorius, paremtus išradimo teiginiais, kuriuose nėra skaitmeninės aritmetikos poreikio. Tokiu atveju, skaitmeninio ALU ip galima nenaudoti (žr. FIG. 1).

Identifikavimo registro celė ID yra sujungta su laidu id, o aplinkos registro celė ENV yra sujungta su laidu env.

Šynos laidas hi yra sujungiamas su šynos laidu vi per jungiklį SW_{vi}, kur i yra skaičius nuo 0 iki 3. Šyna, į kurią įeina laidai h0, h1, h2, h3, yra pavadinta OBJh, t.y., horizontalia šyna. Tarp kitko, šyna OBJh yra naudojama duomenų įvedimui vertikalia kryptimi, t.y., į registrų stulpelį, esantį šerdies celėje ir iš kurio yra perduodami objekto atminties duomenys per šyną OBJv. Smulkiau tai yra aprašyta toliau, saryšyje su FIG. 20.

Šynos laidai id, env, v0, v1, v2, v3 gali būti prijungiami prie šynos laido h0 per jungiklius $SW_{id,h0}$, $SW_{env,h0}$, SW_{v0}, $SW_{v1,h0}$, $SW_{v2,h0}$ ir $SW_{v3,h0}$. Šyna res, į kurią įeina šynos laidai c_{id}, c_f, c_h ir c_v, yra prijungti prie valdymo bloko 6p, ir gali būti naudojami registro nustatymui ties konstanta, pavyzdžiui, į nulį. Šynos laidas c_{id} yra sujungtas su identifikavimo registro cele, o šynos laidas c_f yra sujungtas su registro celėmis F0, F1, F2, F3, o šynos laidas c_h gali būti jungiamas prie šynos laido h0 per jungiklį $SW_{ch,h0}$, o šynos laidas c_v gali būti jungiamas prie šynos laido vi per jungiklį $SW_{vi,cv}$, kur i yra skaičius nuo 0 iki 3.Kai kuriais atvejais šyna res gali būti praleista (neparodyta).

Papildomi registrai

Duomenų medžio aukščiausias lygis yra vadinamas tėvu. Tėvas kai kada yra įvedamas į papildomas registro celes F0, F1, F2, F3, kurios yra parodytos FIG. 7 kairėje pusėje. Parodytame atvejyje kiekvienas papildomas registras gali atmintyje laikyti žodį. Kiekviena papildomo registro celė yra prijungta prie šynos laido id ir prie vieno iš laidų h0, h1, h2, h3, atitinkamai, esančių šynoje OBJh, statmenoje šynos laidui id. Papildomo registro celės yra naudojamos nedaugeliui operacijų, kurias gali atlikti šerdies celė. Tokiu būdu, papildomas registras gali būti kai kuriais atvejais praleidžiamas, montuojant celę pagal šį išradimą (tai nėra parodyta. Taip pat yra galima sukurti šerdies celę, kuri turi daugiau, negu vieną papildomą registrų sytulpelį).

Kaip aišku iš aukščiau esančio aprašymo, kiekvienas registras turi registro celes, esančias keliose plokštumose 2p ir turinčiose tą pačią padėtį tose pačiose plokštėse. Todėl visi registrai bus žymimi nuorodomis, naudojamomis FIG. 7, nors ten vaizduojama tik viena celė. Kaip yra aišku iš FIG. 7, registrai yra išrikiuoti eilutėmis ir stulpeliais. Papildomų registrų zona F0, F1, F2, F3 yra stulpelis, o kiekvienas iš bazinių registrų $S_{0,0}$ iki $S_{0,3}$, $S_{1,0}$ iki $S_{1,3}$, $S_{2,0}$ iki $S_{2,3}$, $S_{3,0}$ iki $S_{3,3}$, yra stulpeliai ir turi galimybę atmintyje saugoti sūnų.

Registro celių tarpusavio sujungimas

Kiekvienoje plokštėje tarp gretimų bazinių registro celių yra sujungimai, tiek vertikaliai, tiek ir horizontaliai. Sujungimas, kuris turi fiksuotą užprogramuojamą reikšmę, schemoje yra parodytas raide f, tas pats yra ir su tolimiausiomis bazinio registro celėmis horizontalioje eilutėje prie objekto atminties. Eilutė yra prijungta prie registro celės N (north, šiaurė) terminalo (žr. FIG.8) ir jungtis yra naudojama tada, kai atliekami

perkėlimai šiaurės - pietų kryptimi. Sujungimas įstrižainės kryptimi tarp bazinio registro celių yra atliekamas taip, kad būtų

sujungiamos perkeliamos padėtys. Tai reiškia, kad celė $S_{i,j}$, kur i skiriasi nuo j, yra sujungiama su cele $S_{j,i}$. Kiekviena bazinio registro celė yra prijungta prie bazinio registro celės, esančios žemiau ir arčiausiai į dešinę. Kiekviena papildoma identifikavimo, aplinkos ir bazinio registro celė yra prijungta prie vienos iš plokščių BOOL per išvadas ACC_{Fx}, ACC_{id}, ACC_{env} ir $ACC_{mx,y}$, kur x ir y yra skaičiai tarp 0 ir 3.

Toliau yra aprašoma bendra registro celė, esanti -plokštėse NUM ir HEAD (FIG. 8). Jungiklių ir ventilių pavyzdžiai, naudojami bendroje registro celėje, yra parodyti nuo FIG. 9A iki 9F.

Nuo bendros registro celės priklauso ir papildomo registro celės (FIG. 10) ir identifikatoriaus registro celės (FIG. 11) konstrukcija.

Dar toliau yra aprašomi registro celių, kurios yra patalpinamos plokštėse ATTRIBUTE, įgyvendinimo pavyzdžiai. FIG. 13 yra parodyta identifikavimo registro celė. FIG. 14 yra parodyta papildomo registro celė. FIG. 15 yra parodyta pagrindinio registro celė.

Bendroji registro celė, esanti
plokštėse NUM ir HEAD

Pagal FIG. 8 į siūloma registro celę įeina dvi vidinės šynos a_R ir b_R ir centrinis vidinis registras r_R. Šynos a_R ir b_R yra prijungtos prie keleto išorinių jungčių, esančių už registro celės ribų. FIG. 8 parodytame pavyzdyje bendra registro celė turi visas galimybes jungtis su išore. Paprastai tam tikra registro celė neturi visų jungčių, kurios yra parodytos FIG. 8. Priklausomai nuo registro celės padėties, viena ar kita iš jų nėra reikalinga. Visi laidai, esantys tarp sujungimų terminalų, yra aiškūs iš schemos FIG. 7. Taip pat iš FIG. 7 yra matoma, kad ne visos registro celės turi visus išorinius sujungimus, parodytus FIG. 8. Visų registro celių ir jų jungtys yra parodytos FIG.8. Smulkesnis jų aprašymas čia nėra pateikiamas.

Šyna a_R

Šyna a_R yra prijungta prie laido vx, kur x yra skaičius nuo 0 iki 3, per jungiklį SW_{V1} ir terminalą V. Toliau yra prijungta prie horizontalaus laido hy, kur y yra skaičius nuo 0 iki 3, per jungiklį SW_{H1} ir terminalą H prie registro celės, esančios kairėje, per terminalą W (west, vakarai) sujungta su jungikliu SW_E (east, rytai) ir greta esančia registro cele ir, jei registro celė yra pagrindinė, tiesiogiai per terminalą NU su aritmetiniu bloku 1p.

Šyna a_R taip pat yra sujungta su registro cele, esančia žemiau, dešinėje, per terminalą Da prie jungiklio SW_{Db}, esančio toje registro celėje ir prie registro celės, esančios žemiau, per terminalą S (south, pietūs) prie jungiklio SW_N (north, šiaurė), esančio šioje registro celėje. Registro celė gali būti nustatoma arba numetama (set, reset) per terminalą C ir jungiklį SW_c, kuris yra sujungtas su šyna a_R. Šyna a_R yra taip pat per jungiklį SW_a sujungta su centrinio vidinio registro r_R įėjimu ir per jungiklį SW_{ao} su to paties registro išėjimu.

Šyna b_R

Šyna b_R yra prijungta prie registro celės, esančios dešinėje, per jungiklį SW_E ir terminalą E, o taip pat prie registro celių įstrižainės per jungiklį SW_{Db} ir terminalą Db, o taip pat prie registro celės, esančios viršuje, per jungiklį SW_N ir terminalą N.

Šyna b_R taip pat yra sujungta su centrinio vidinio registro r_R įvadu ir per jungiklį SW_{bo} prie to paties registro išėjimo.

Centrinis vidinis registras

Į centrinį vidinį registrą r_R įeina du invertoriai Q1 ir Q2, pageidautina CMOS tipo invertoriai, o tarp jų yra valdomas jungiklis SW_Q. Pilna registro celė taip pat turi šynas a_R ir b_R ir jungiklius SW_a, SW_{ao}, SW_{b1}, SW_{bo} ir jungiklius, jungiančius celę

su išore. Centrinio vidinio r_R registro išėjimas yra sujungiamas su horizontalia ir vertikalia šynomis per jungiklį SW_{Ho} ir terminalą H bei per jungiklį SW_{Vo} ir terminalą V. Centrinis vidinis r_R registras laiko atmintyje dinaminę būseną (bus paaiškinta toliau).

Jungiklio darbas

Visi valdomi jungikliai registro šerdies celėje yra valdomi per laidus, prijungtus prie valdymo bloko 6p, iš kurio ieina ventilinė matrica, tokia, kaip PAL (Programmable Array Logic, programuojama loginė matrica). Ventilinė matrica naudoja informaciją, įvestą į šerdies celę ir nustato, kurį jungiklį atidaryti ir kurį po to uždaryti. Ventilinės matricos darbas yra sinchronizuojamas laiko impulsais. Jungikliai yra dvipusiai, tačiau kai kurie iš jų yra naudojami tik niena kryptimi, pavyzdžiui, įėjimo ir išėjimo jungikliai SW_{Hi} ir SW_{Ho}.

Komparatoriaus įrenginys COMP

Į komparatoriaus įrenginį COMP įeina pirmasis NAND ventilis G1. Vienas įėjimas yra prijungtas prie neinvertuojamo įėjimo invertoriuje Q1, o kitas - prie invertoriaus Q2 įėjimo. Įtaisas COMP taip pat turi antrąjį NAND ventilį G2. Vienas jo įėjimas yra prijungtas prie invertoriaus Q1 išėjimo, o kitas - prie invertoriaus Q2 išėjimo. Ventilių G1 ir G2 išėjimai yra sujungti su vieno laido šyna ACC, einančia į vieną iš BOOL plokščių. Abu NAND ventiliai gali turėti dvi eiles suporintų MOS - Fet tranzistorių, nuosekliai sujungtų takeliais tarp žemės ir BOOL plokštės, jų ventiliai yra NAND ventilių įėjimai, o aukščiausio MOS - FET tranzistoriaus takelis yra išėjime (žr. FIG. 9D). Šis komparatoriaus įtaisas COMP yra naudojamas asociatyvinės paieškos metu, t.y., tada, kai šerdies celės elementas yra lyginamas su elementu, esančiu objekto atmintyje arba kitoje šerdies celės dalyje. Po to lyginamas elementas yra perduodamas į registro

celių įėjimus, kurie turi sulyginamą elementą ir kurie bus aprašomi toliau.

Invertoriai ir jungikliai

Invertoriai Q1 ir Q gali turėti arba du MOS -FET stiprinančius tranzistorius ir vieną praleidžiantį tranzistorių, sujungtus taip, kaip yra parodyta FIG.9B, arba du papildomus MOS - FET tranzistorius (FIG. 9C). Valdomi jungikliai, esantys registro celėje, gali turėti arba MOS - FET tranzistorių (FIG. 9E), arba du papildomus MOS - FET tranzistorius (slopinančio tipo, EE MOS), sujungtus, kaip parodyta FIG. 9F. Valdymo blokas 6p valdo jungiklių signalo c pagalba. Kaip galima matyti iš FIG. 9F, jungiklis gali būti valdomas tiek valdymo signalu, tiek ir papildomu signalu tuo tikslu, kad būtų atliekami greitesni būsenos perėjimai.

Pagrindinę registro celę, parodytą FIG. 8, galima laikyti visų registro celių baze, esančia šerdies celėje, t.y., jie yra sudaryti tuo pačiu principu. Registrų celės, esančios pagrindiniame registre, skiriasi tik terminalų skaičiumi ir su tuo susijusiais įėjimo ir išėjimo jungikliais. Šios celės turi tuos pačius nurodymo numerius, kaip ir celė, parodyta FIG.8.

Bazinės registro celės, esančios NUM ir HEAD plokštėse

Bazinio registro celės $S_{0,0}$, $S_{0,1}$, $S_{2,3}$ ir $S_{3,3}$ neturi jungiklio SW_{Db} ir terminalo Db, bazinio registro celės $S_{0,0}$, $S_{1,0}$, $S_{3,2}$ ir $S_{3,3}$ neturi terminalo Da, pagrindinio registro celės (nuo $S_{0,0}$ iki $S_{3,0}$) neturi S terminalo, o likusios bazinio registro celės (nuo $S_{0,1}$ iki $S_{3,3}$) neturi terminalo NU. Niekas iš bazinio registro celių neturi C terminalo arba SW_c jungiklio - vietoj jų yra naudojamos vertikalios ir horizontalios šynos, o terminalai V ir H yra naudojami nustatyti arba numesti registro celę pastovios reikšmės pagalba, perduodama per šynų laidus C_v arba c_h.

Papildomo registro celės, esančios NUM ir HEAD plokštėse

Papildomo registro celė, parodyta FIG. 10, turi tik terminalus H_y, V, C ir ACC, kur y yra skaičius nuo 0 iki 3 ir kur terminalas V yra sujungtas su šynos laidu ID ir kur C terminalas yra sujungtas su šynos laidu C_f.

Identifikatoriaus/aplinkos registro celės, celės, esančios NUM ir HEAD plokštėse

Identifikatoriaus registro celė, parodyta FIG. 11, turi tik terminalus V, C ir ACC, kur V terminalas yra sujungtas su šynos laidu ID, o C terminalas - su šynos laidu C_{id}. Aplinkos registro celė (neparodyta) yra ta pati, kaip ir identifikatoriaus registro celė, parodyta FIG. 11, nors šiame pavyzdyje ji yra be C terminalo ir be SW_c jungiklio. Kitame pavyzdyje aplinkos registro celė gali turėti C terminalą ir SW_c jungiklį.

Asociatyvi paieška ir BOOL plokštė

Asociatyvios paieškos metu sulyginimas yra atliekamas per išrinkimo schemos šyną, einančią į plokštę BOOL. Du IR ventiliai G1 ir G2 sulygina rakto reikšmę, t.y., reikšmę, kuri yra atmintyje ir turi būti sulyginama su išėjimo iš Q1 reikšme ir esanti atmintyje - su įėjimo į Q2 reikšme. To sulyginimo metu rakto reikšmė yra perduodama per vidinę šyną a_R arba b_R į Q1. Jungiklis SW_a turi būti išjungtas, t.y., atviras.

Jeigu perduota reikšmė, t.y., rakto reikšmė, neatitinka atmintyje esančiai reiokšmei, krūvis, esantis BOOL plokštėje, išsikrauna per NAND ventilius G1 ir G2. Jei reikšmės atitinka, BOOL plokštė lieka su krūviu.

Visi šynos laidai ACC, esantys registre, vienas šynos laidas ACC registro celėje, gali būti lygiagrečiai sujungti ir prijungti

prie to paties laido, esančio plokšteje BOOL, kaip alternatyva, šynos laidai ACC iš visų registro celių, esančių plokštėse NUM ir HEAD, gali būti sujungti su šynos laidu, esančiu BOOL plokšteje. Visos registro celės, esančios plokšteje ATTRIBUTE, gali būti sujungtos su atskiru šynos laidu, esančiu arba toje pačioje BOOL plokšteje, arba antroje BOOL plokšteje, skirtoje ATTRIBUTE plokštėms. Galima pasirinkti vieną ar dvi BOOL plokštes ir vieną arba du šynos laidus, tai priklauso tik nuo valdymo komandų, esančių valdymo bloke 6p, tipo. Išradimas apima ir tą atvejį, kai yra naudojamos ir daugiau, negu dvi BOOL plokštės. Naudojamų BOOL plokščių skaičius salygoja asociatyvinės paieškos smulkumą, t.y., skirtingų asociatyviniu paiešku skaičių, kurį galima atlikti ir tai, kokiu mastu ją galima atlikti, t.y., kokios registro dalys gali būti įtrauktos į darbą. Tokiu būdu, sulyginimas yra atliekamas vienu metu toje registro dalyje, kuri yra sujungta su tuo pačiu šynos laidu, esančiu BOOL plokšteje. Jei visi NAND ventiliai G1 ir G2 turi tą patį išėjimą (aukštą), tada sulyginimo rezultatai rodo sutapimą, priešingu atveju, rodo nesutapimą, o sutapimas reiškia, kad abi informacijos dalys yra identiškos. Tokiu būdu, BOOL plokštės yra plokštės, skirtos šynų laidams, ir jos gali būti laikomos "mąstančiomis" plokštėmis, t.y., šynų laidai nebūtinai turi būti plokšteje, o gali būti tiesiogiai sujungti su valdymo bloku 6p.

ATTRIBUTE plokščių konfigūracija

ATTRIBUTE plokštės turi kitokią konfigūraciją, negu plokštės NUM ir HEAD, kaip tai parodyta FIG. 12. Tą pačią sandarą turintiems elementams yra naudojami tie patys pažymėjimai, kaip ir FIG. 7. Į šio tipo plokštes įeina jungikliai $SW_{\vee 0}$, $SW_{\vee 1}$, $SW_{\vee 2}$ ir $SW_{\vee 3}$, viena identifikatoriaus registro celė ID_T, keturios papildomo registro celės $F0_T$, $F1_T$, $F2_T$ ir $F3_T$ ir bazinio registro

blokas, į kurį įeina vien tik pagrindinio registro celės $S_{0,0}$, $S_{1,0}$, $S_{2,0}$ ir $S_{3,0}$. Tokiu būdu, bazinio registro matrica yra sumažinta iki pagrindinio registro eilutės nuo $S_{0,0}$ iki $S_{3,0}$, sujungtos su atitinkamais šynų laidais v_0, v_1, v_2 ir v_3, bei

prie laido h0 prijungtais jungikliais SW_{v0}, $SW_{v1,h0}$, $SW_{v2,h0}$ ir $SW_{v3,h0}$, taip pat, kaip ir registro plokštėje, parodytoje FIG. 7. Šynos laidas hi yra sujungiamas su laidu vi per jungiklį SW_{vi}, kur i yra skaičius nuo 0 iki 3. Tačiau šynų laidai v0, v1, v2 ir v3 gali būti pravesti į kitus interfeiso 9p įvadus, negu šynų laidai, turintys tuos pačius žymėjimus plokštėse NUM ir HEAD. Tokiu būdu jie gali būti sujungti su kitomis objekto atminties 4p dalimis, geriausiai, su dalimi lazy, where ir type (žr. FIG. 1). Kaip alternatyva, šynos laidai v0, v1 ir v3 neturi būti sujungti su objekto atmintimi 4p, vietoj to, šynos laidas id gali būti naudojamas perduoti informaciją apie būseną (žr. FIG.1) iš objekto atminties, t.y., lazy, where ir type, esantys objekto atmintyje yra sujungti su šynos laidu id, esančiu atitinkamoje šerdies celės plokštėje. Taip pat, pagrindinio registro celės eilutė nuo $S_{0,0}$ iki $S_{3,0}$, identifikatoriaus registro celė ID_T ir papildomo registro celės nuo $F0_T$ iki $F3_T$ yra sujungtos su šyna res, į kurią įeina šynos laidai c_{id}, c_f, c_h ir c_v, tokiu pat būdu, kaip ir registro plokštėje, parodytoje FIG. 7.

Identifikatoriaus registro celės ID_T vertikalus stulpelis ir keturios papildomo registro celės $F3_T$, $F2_T$, $F1_T$ ir $F0_T$ yra sujungtos su šynos laidu id, kuris taip pat yra sujungtas su antru šynos laiducont (FIG. 1 neparodyta), einančiu į valdymo bloką 6p. Valdymo blokas 6p gali naudoti informaciją, kuri gali būti perduota į šią šyną, norint nuspręsti, kokio tipo turi būti atliktas suskaidymas. Kiekviena iš registro celių, tokia, kaip parodyta FIG. 12, greta jų išrinkimo ir šynos laido ACC, taip pat turi įėjimo laidą SD_i, kur i yra skaičius nuo 0 iki 3, arba žymes

ID, F iki F, kas yra naudojama. tiesioginiam registro celiu, sujungtu su išėjimo laidais, tikrinimui.

Identifikatoriaus registro celės, esančios ATTRIBUTE plokštėse

FIG. 13 yra parodytas identifikatoriaus registro celės ID_T pavyzdys, esantis plokštėse ATTRIBUTE. Ji turi keturis terminalus V, CONT, SD ir ACC. Terminalai V ir CONT yra sujungiami su šynos laidais id ir cont atitinkamai. Terminalas CONT yra sujungiamas su vidinio registro r_R išėjimu per jungiklį SW_{CONT}. Terminalas SD yra sujungiamas su vidinio registro r_R išėjimu, t.y., ties invertoriaus Q_2 išėjimu. Terminalas C yra sujungtas su šynos laidu c_{id}.

Papildomo registro celės, esančios ATTRIBUTE plokštėse

FIG. 14 yra parodytas papildomos registro celės Fy_T pavyzdys, esantis plokštėse ATTRIBUTE. y yra skaičius nuo 0 iki3. Lyginant su identifikatoriaus registro cele, ši registro celė turi viena papildoma terminala H. Terminalas H yra sujungiamas su šynos laidu hy, kur y yra skaičius nuo 0 iki3. Likę terminalai yra sujungti tokiu pat būdu su atitinkamais terminalais, analogiškai identifikatoriaus terminalams, esantiems registro celėje ID_T. Terminalas C sujungtas su šynos laidu c_f.

Pagrindinio registro celės, esančios ATTRIBUTE plokštėse

FIG. 15 yra parodytas pagrindinio registro celės $S_{x,o}$ pavyzdys, esantis plokštėse ATTRIBUTE. x yra skaičius nuo 0 iki3. Ji turi šešis terminalus V, E, H, W, SD ir ACC. Terminalas SD yra sujungtas su vidinio registro r_R išėjimu, t.y., ties invertoriaus Q_2 išėjimu. Likę terminalai yra sujungti tokiu pat būdu su atitinkamais terminalais, analogiškai identifikatoriaus terminalams, esantiems plokštėse NUM ir HEAD. Nei viena iš pagrindinio registro celė neturi C terminalo, nei jungiklio SW_C. Vietoj jų registro celės nustatymui ir numetimui pastovia reikšme

iš šynos laido c_v arba c_h yra naudojamos vertikali ir horizontali šynos ir terminalai V ir H.

Rezervinės atminties režimas

Rezervinės atminties schema yra sudaroma iš vienos arba abiejų schemų, esančių celėje. Vieną schema sudaro jungiklis SW_{bo}, šyna b_R, jungiklis SW_{b1}, invertorius Q1, jungiklis SW_a ir invertorius Q2. Kitą schemą sudaro jungiklis SW_{ao}, šyna a_R, jungiklis SW_{a1}, invertorius Q1, jungiklis SW_a ir invertorius Q2. Kai jungikliai, esantys vienoje arba abiejose schemose yra uždari, signalas gali praeiti pro du invertorius Q1 ir Q2, signalo lygis tampa stabilus invertoriaus Q1 įėjime ir invertoriaus Q2 išėjime, tokiu būdu į celę yra įvedami duomenys. Celės atmintyje yra saugoma dinaminė būsena.

Išvedimo režimas

Išvedimo režimo metu Q2 išėjimas gali būti pervedamas į vieną iš šynų a_R arba b_R, o iš čia tada gali būti valdomi reikiami jungikliai, kad išėjimas būtų perduodamas į vieną arba daugiau išėjimo terminalų (N, S, E, W ir tt.). Kita šyna b_R arba a_R gali būti naudojama pasirinktinai. Jei jungiklis SW_a yra išjungtas,t.y., atviras, išėjimas iš invertoriaus Q2 yra stabilus, t.y., jis negali būti pakeistas tol, kol jungiklis SW_a yra uždaras. Invertoriaus išėjimas gali būti perduotas į šyną b_R per jungiklį SW_{bo}, kai jis yra uždaras, į šyną a_R per jungiklį SW_{ao}, kai jis yra uždarytas, į išėjimą cont per jungiklį SW_{cont}, kai jis yra uždarytas, ir tiesiai į terminalą SD. Informacija šynose b_R ir a_R gali būti perduota į kiekvieną iš išoriniu šynų, prie kurios yra prijungta registro celė, valdant jungiklį, įjungtą tarp registro celės ir išorinės šynos, kaip tai bus parodyta toliau aprašomame pavyzdyje.

Įvedimo režimas

Įvedimo režimo metu vienas iš jungiklių SW_{a1} arba SW_{b1} yra įjungtas, t.y., uždaras. Tokiu būdu, vieno iš terminalų būsena (N, S, E, W ir tt.) yra perduodama į vietinę šyną a_R arba b_R ir iš čia - į centrinį nulinį registrą r_R.

Perdavimai

Iš bet kurio registro celės, esančios šerdies celėje, galima perduoti duomenis per terminalo jungtis į kitą registro celę, esančią šerdies celėje, dviejų fazių ciklo metu. Trijų fazių ciklo metu galima atlikti pakeitimą dviejų bazinių registrų celėse vertikalia, horizontalia arba įstrižainės kryptimi.

Jungiklis SW_Q yra paleidžiamas tiesiogiai nuo pagrindinių laiko impulsų ir vienu metu visoms registro celėms, taip, kad perdavimai tarp invertorių Q1 ir Q2 būtų atliekami tuoj pat visoje šerdies celeje. Likusieji jungikliai yra valdomi nuo signalų, gaunamų iš pagrindinių laiko impulsų, tačiau perduodamų skirtingais atitinkamos fazės laiko impulsų intervalais.

Laiko ciklai yra skirstomi į laiko fazes Q, a ir/arba b. Fazė Q yra pirmoji fazė, t.y., tada, kai centrinis vidinis registras r_R yra rezervinėje atmintyje ir kai duomenys yra stabilūs. a fazė yra naudojama perdavimo iš šynos a_R metu, o b fazė yra naudojama perdavimo iš šynos b_R metu.

Vienpusis perdavimas, t.y., tiktai iš arba į registro celę, yra atliekamas dviejų fazių laiko impulsų metu. Pirmoji fazė Q yra stabili. Dviejų fazių cikle perdavimui yra naudojama fazė a arba fazė b.

Dvipusio perdavimo, t.y., tarp dviejų registro celių, metu, jis yra atliekamas trijų fazių laiko ciklais. Fazė Q yra stabili. Fazių a ir b metu perdavimas yra atliekamas skirtingomis kryptimis. Reikia pastebėti, kad gali būti ir laiko ciklas, turintis daugiau, negu tris fazes, šis atvejas taip pat turimas omenyje šiame išradime, pavyzdžiui, esant dviem fazėms b.

Jungikliai SW_{a1} ir SW_{b1} yra normaliai uždari. Įvestą į atmintį registro būseną laiko abi vietinės šynos a_R ir b_R. Kai vidinė šyna a_R arba b_R bus naudojamos vietinės reikšmės įvedimui į atmintį, tada atidarant yra valdomas atitinkamas jungiklis SW_{a1} arba SW_{b1}. Jungiklis vienai iš išorinių, vertikaliai arba horizontaliai šynai, trumpo laiko intervalo metu yra uždaromas, tačiau to laiko pakanka, kad esanti toje šynoje informacija būtų perduota į vidinę šyną.

Taip pat yra galima naudoti ir perstumtą schemą, t.y., schemą, esančią tarp skirtingų registro celių, turinčių jungiklius, sujungtus su terminalais, kad būtų galima perduoti registro celių turinį N, S, W, E kryptimis.

Vienpusio perdavimo pavyzdys

FIG. 16A yra parodytos dvi kaiminystėje esančios bazinio registro celės, o duomenys yra perduodami iš kairiosios, siųstuvo, į dešiniąją, imtuvo pusę. Valdymo signalai iš valdymo bloko 6p valdo jungiklius. FIG. 16B yra parodyta kiekvieno jungiklio būsena duomenų perdavimo metu skirtingų fazių metu, žemesnioji būsena rodo atvirą jungiklį, o vyresnioji būsena - uždarą jungiklį. Realus perdavimas vyksta b fazės metu. Perdavimai yra vykdomi sekančiu būdu (skirtingi žingsniai, nurodyti tiek FIG. 16A, tiek ir FIG. 16B, yra pažymėti tais pačiais skaičiais):

0. Schema yra stabili, SW_{ω}, SW_{a0}, SW_{a1}, SW_{b0}, SW_{b1} yra uždaryti, visi kiti jungikliai yra atviri tiek į siųstuvo, tiek ir į imtuvo pusę (šis žingsnis nėra pažymėtas FIG. 16A, kadangi liečia visus jungiklius). Šis stabilus režimas FIG. 16B atitinka 0 fazę.
1. Pirmos laiko impulsų fazės metu (b fazės), kai jungiklis SW_{ω} tiek į siųstuvą, tiek ir į imtuvą yra atidarytas,

2. jungiklis SW_{ao} yra atidarytas, o jungiklis SW_{bo} tiek į siųstuvą, tiek ir į imtuvą yra uždarytas,
3. jungiklis SW_E, esantis tarp siųstuvo ir imtuvo yra uždarytas,
4. jungiklis SW_{b1} tiek į siųstuvą, tiek ir į imtuvą yra atidarytas, ir
5. jungiklis SW_{a1} į siųstuvą yra atidarytas, o jungiklis SW_{a1} į imtuvą yra uždarytas. Tai įgalina tai, kad duomenys iš siųstuvo vidinių registrų galėtų pakliūti į imtuvo vidinius registrus.
6. Antros laiko impulsų fazės metu (0 fazės), kai jungiklis SW_a tiek į siųstuvą, tiek ir į imtuvą yra uždarytas,
7. jungiklis SW_E tarp siųstuvo ir imtuvo yra atidarytas,
8. pirmiausiai yra uždaromi jungikliai SW_{bo} ir SW_{ao}, o po to jungikliai SW_{b1} ir SW_{a1}, tiek siųstuve, tiek ir imtuve. Tai gražina atgal į stabilų režimą, aprašytą aukščiau, žingsnyje 0, t.y., 0 faze.

Dvipusio perdavimo pavyzdys

FIG. 17A yra parodytos dvi kaiminystėje esančios bazinio registro celės, o duomenys yra perduodami tarp dviejų skirtingų bazinio registro celių dvipusio perdavimo operacijos būdu. Du valdymo signalai iš valdymo bloko 6p valdo jungiklius. FIG. 17B yra parodyta kiekvieno jungiklio būsena duomenų perdavimo metu skirtingų fazių metu, žemesnioji būsena rodo atvirą jungiklį, o vyresnioji būsena - uždarą jungiklį. Abi registro celės dirba, kaip siųstuvas ir kaip imtuvas, todėl jos toliau bus vadinamos "1 celė" ir "2 celė". Vienas perdavimas iš 2 celės į 1 celę vyksta

a fazės metu, o kita kryptimi, iš 1 į 2 celę, vyksta b fazės metu. Skirtingi žingsniai, nurodyti tiek FIG. 17A, tiek ir FIG. 17B, yra pažymėti tais pačiais skaičiais. Perdavimas vyksta sekančiu būdu:

0. Schema yra stabili, SW_a, SW_{ao}, SW_{a1}, SW_{bo}, SW_{b1} yra uždaryti, visi kiti jungikliai yra atviri abiejose celėse (šis žingsnis nėra pažymėtas FIG. 17A, kadangi jis liečia visus jungiklius). Šis stabilus režimas FIG. 16B atitinka 0 fazę.
1. Pirmos laiko impulsų fazės metu (a fazės), kai jungiklis SW_a celėse1 ir 2 yra atidarytas,
2. jungiklis SW_{ao} celėse 1 ir 2 yra uždarytas, o jungiklis SW_{bo} celėse 1 ir 2yra uždarytas,
3. jungiklis SW_E , esantis tarp celių, yra uždarytas,
4. jungiklis SW_{a1} celėse 1 ir 2 yra atidarytas, ir
5. jungiklis SW_{b1} celėje 1 yra uždarytas , o jungiklis SW_{b1} 2 celėje yra atidarytas. Tai duoda galimybę, kad duomenys galėtų pakliūti iš celės 2 į 1.

Antros laiko impulsų fazės metu (b fazės), kai jungiklis SW_a vis dar yra atidarytas,

6. jungiklis SW_{ao} celėse 1 ir 2 yra atidarytas, o jungiklis SW_{bo} celėse 1 ir 2 yra uždarytas,
7. jungiklis SW_{b1} celėse 1 ir 2 yra atidarytas,
8. jungiklis SW_{a1} celėje 1 yra atidarytas , o jungiklis SW_{a1} 2 celėje yra uždarytas. Tai duoda galimybę, kad duomenys galėtų pakliūti iš celės 1 į 2.
9. Trečios laiko impulsų fazės metu (0 fazės), kai jungiklis SW_a 1 ir 2 celėse yra uždarytas,
10. jungiklis SW_E , esantis tarp celių, yra atidarytas, ir
11. pirmiausiai yra uždaromi jungikliai SW_{bo} ir SW_{ao}, o po to jungikliai SW_{b1} ir SW_{a1}, abiejose celėse. Tai gražina atgal į stabilų režimą, aprašytą aukščiau, žingsnyje 0, t.y., 0 fazę.

Valdymo signalai, perduodami į jungiklius SW_{a0} ir SW_{b0}

Signalai yra įjungti, t.y., 0 fazės metu ventiliai yra uždaryti nuo einamosios, įregistruotos, reikšmės. Visos vietinės šynos tada laiko būseną, esančią atmintyje. Šyna, naudojama įvedimui, yra valdoma, nustatant valdymo signalą į išjungtą padėtį, t.y., atvirą jungikliuose SWQ ir SWx0, kur x yra a arba b. Įvedimo operacijos metu keletas šynų gali būti trumpai sujungtos per terminalus (E, V, D, H, ir tt.) neilgam laiko tarpui. Po kiek laiko šynos įgauna teisingą reikšmę.

Iš valdymo signalo krentančiosios dalies į jungiklį SW_a yra perduodamas uždelsimo laikas į jungiklio SW_{x0} (x yra a arba b) valdymo signalo krentančiąją dalį. Jei jis yra trumpas, tada problemų nekyla. Tačiau, jei jei laiko trukmė yra ms diapazone, šyna x_R (x yra a arba b) gali prarasti savo dinaminę būseną.

Iš valdymo signalo kylančiosios dalies į jungiklį SW_a yra perduodamas uždelsimo laikas į jungiklio SW_{x1} (x yra a arba b) valdymo signalo kylančiąją dalį. Jei jis tampa neigiamu, tada į vietinę šyną x_R iš invertoriaus Q2 ir invertoriaus Q1 gali patekti klaidinga reikšmė. Dėl tos priežasties yra naudojamas teigiamas uždelsimo laikas.

Valdymo signalai, perduodami į jungiklius

SW_E ir SW_V SW_D ir SW_H

Signalai yra normaliai išjungti, t.y., atviri. Tada visos vietinės šynos yra izoliuotos.Šyna, naudojama įvedimui ir išvedimui, yra valdoma, nustatant valdymo signalą terminalo jungiklyje, sujungtame su jo įjungimo padėtimi, t.y., uždarant jungiklį. Šios operacijos metu keletas šynų gali būti trumpai sujungtos per jungiklius (SW_E ir SW_V SW_D ir SW_H ir tt.) neilgam laiko tarpui. Po kiek laiko šynos įgauna teisingą reikšmę.

Iš valdymo signalo krentančiosios dalies į jungiklį SW_a yra perduodamas uždelsimo laikas į jungiklio SW_z (Z yra H, D, N, V, E ir tt., t.y., bet kuris iš terminalų, sujungtų su vidinėmis

Šynomis a_R ir b_R turi įjungiklį) valdymo signalo kylančiąją dalį. Jei jis yra neigiamas, tada vietinės šynos x_R (x yra a arba b) reikšmė gali pasikeisti. Po to gali būti nustatyta registro reikšmė. Tokiu būdu, šis uždelsimo laikas turi būti teigiamas.

Iš valdymo signalo kylančiosios dalies į jungiklį SW_Z (kur Z yra H, D, N, V, E ir tt., t.y., bet kuris iš terminalų, sujungtų su vidinėmis šynomis a_R ir b_R turi jungiklį) yra perduodamas uždelsimo laikas į jungiklio SW_{x1} valdymo signalo krintančiąją dalį. Jei jis tampa neigiamu, tada reikšmė negali patekti į įėjimą. Dėl tos priežasties yra naudojamas teigiamas uždelsimo laikas.

Iš valdymo signalo kylančiosios dalies į jungiklį SW_{x1} yra perduodamas uždelsimo laikas į jungiklį SW_Z. Jei jis tampa neigiamu, vietinė šyna gali pasikeisti ir registras gali nusistatyti ties klaidinga reikšme. Dėl tos priežasties yra naudojamas teigiamas uždelsimo laikas.

Valdymo signalai, perduodami į jungiklius SW_E ir SW_V SW_D ir SW_H

Q fazės metu signalai yra einamosios reikšmės (įregistruoti). Tačiau turi būti nedidelis vėlavimas nuo valdymo signalo kylančios dalies į jungiklio SW_Q valdymo signalo kylančiąją dalį, einančią į jungiklius SW_{a1} ir SW_{b1}. Jei šis vėlavimas tampa neigiamu, tada į vietinę šyną x_R (x yra a arba b) iš invertoriaus Q2 įėjimo negali patekti reikšmė. Dėl tos priežasties yra naudojamas teigiamas vėlavimo laikas.

Skaičiavimai, atliekami šerdies celėje

Vieno šio įrenginio ciklo metu yra atliekamos tipinės sąrašo komandos.

Kaip buvo anksčiau minėta, šerdies celė atlieka struktūrines aritmetikos operacijas. Visi žingsniai yra atliekami šerdies registruose, naudojant komandas, kurios yra sąrašuose. Komandų pavyzdžiai yra šie:

length ilgis	skaičiuojamas tikslo funkcijos ilgis,
map	funkcija yra taikoma sąrašo elementams.
planas	Jei į sąrašą įeina įterpti sąrašai, tada ši komanda yra taikoma ir šių įterptų sąrašų elementams. (Komandų planas bus dar vėliau paaiškintas).
filter filtras	funkcija, taikoma, filtruojant sąrašo elementus. Jei yra įterptų sąrašų, tada filtras yra taikomas ir jiems.
join sujungti	visi elementai yra perrašomi į įterptus sąrašo elementus. Ši komanda taip pat yra taikoma įterptiems, jei tokių yra, sąrašams.
transpose perkelti, sukeisti	yra sukeičiamos mažos matricos. Jei jos turi sąrašo elementų, jie yra sukeičiami. Tvarkomi įterptieji sąrašai. (Komanda transpose dar bus aiškinama vėliau).

Šerdies celės atmintis

Šerdies celės atmintyje yra saugoma

* tikslo funkcija, kuri turi būti suskaidoma keliuose registruose, pageidautina, baziniuose;

* kai kuriais atvejais, pavyzdžiui, yra suskaidoma trijų lygių struktūra, tikslo šaknis, pageidautina, papildomuose registruose, o likusi dalis - bazinio registro matricoje.

Šerdies celės laikinoji atmintis yra naudojama keturiais atvejais, t.y., saugant 0, 1, 2 arba 3 lygio objektus.

Paprastas medis, t.y., vienetinė reikšmė, yra įvedama į pirmojo pagrindinio registro atmintį.

Medis, į kurį įeina tik vienas lygis, yra įvedamas į pagrindinį registrą.

Medis, į kurį įeina du lygiai, gali turėti savo šaknies saraša, esantį tėvu, įvedamu į pagrindinio registro atmintį horizontaliai, ir sarašus - sūnus, įvedamus vertikaliai į bazinius registrus. Kaip alternatyva, šaknis gali būti įvedama į papildomų registrų atmintį, o vienas iš sūnų - į pagrindinius registrus. Reikia pastebėti, kad valdymo blokas ėp gali pasirinkti viena arba kitą iš šių alternatyvų, priklausomai nuo reikiamų atlikti operacijų.

Medis, į kurį įeina trys lygiai, turi savo šaknies saraša, įvedamą į vieną iš papildomų registrų, ir vieną iš dviejų lygių sūnų, įvedamų į bazinius registro matrica.

Tokiu būdu, tikslo medžio šaknies saraša yra pageidautina įvesti į skirtingus šerdies celės registrus, priklausomai nuo medžio struktūros lygio ir reikiamos atlikti operacijos.

Tikslo medžio šaknis yra uždaras reiškinys, skirtas suskaidymui, pavyzdžiui, unify. Funkciniams reikalams (apply...) yra skirtas pirmasis elementas, kuris reiškia komanda arba yra identifikatorius, netiesiogiai žymintis uždarą struktūrą, naudojamą funkcinio apibrėžimo vietoje, o likusieji elementai yra komandos/funkcijos apibrėžimo argumentai.

ŠERDIES CELĖS ATMINTIES VALDYMAS

Informacija, įvedama į šerdies registrus, yra perduodama iš objekto atminties 4p. Informacija į šerdies registrus yra įvedama sekančiu būdu.

Atminties valdymas ATTRIBUTE plokštėse

ATTRIBUTE plokštėse šerdies registrai yra sujungti su objekto atminties šyna OBJ. Įvedama į atmintį būsena susideda iš būsenos, kurioje yra identifikatoriaus registras ID_T, papildomi registrai nuo FO_T iki $F3_T$ ir baziniai registrai nuo $S_{0,0}$ iki $S_{3,0}$.

Valdymo žodis, perduodamas į šerdies registro celes, esančias ATTRIBUTE plokštėse, susideda iš mažesnių valdymo žodžių, perduodamų į jungiklius SW_{vi}, $SW_{vi,cv}$, kur i yra skaičius tarp 0 ir 3, $SW_{id,no}$, $SW_{ch,\ no}$, $SW_{v1,no}$, $SW_{v2,no}$ ir $SW_{v3,no}$, identifikatoriaus registro ID_T, papildomų registrų nuo FO_T iki $F3_T$ ir pagrindinių registrų $S_{0,0}$ iki $S_{3,0}$.

Valdymo žodžiai yra perduodami per keletą valdymo laidų, prijungtų prie valdymo bloko 6p. Valdymo laidai gali būti dvifazinių valdymo laidų poros arba vienos fazės pavieniai valdymo laidai, priklausomai nuo to, kokios rūšies yra naudojami jungikliai (žr. FIG. 9E ir 9F).

Valdymo žodis į kiekvieną celę, esančią pagrindiniuose registruose nuo $S_{0,0}$ iki $S_{3,0}$ yra perduodamas per vieną bendrą kanalą, atskirus kiekvienam baziniam registrui.

Bendras kanalas valdo jungiklius SW_{mo}, SW_{bo} ir SW_{a}, esančius šerdies celėje (žr. FIG. 15). Tačiau reikia pažymėti, kad tai, kas čia aprašyta, turi būti laikoma vien kaip pavyzdys ir kad yra galimi kai kurie kiti realizavimo atvejai.

Atminties valdymas HEAD ir NUM plokštėse

HEAD ir NUM plokštėse šerdies registrai yra sujungti su objekto atminties šyna OBJ, išrinkimo šyna ACC, šyna res ir skait-

meninio ALU šyna NUM. Ivedama į atmintį būsena susideda iš dviejų vienetinių registrų ID ir ENV būsenos, papildomų registrų nuo F0 iki F3 būsenos ir bazinių registrų nuo $S_{0,0}$ iki $S_{3,0}$.

Valdymo žodis, perduodamas į šerdies registro celes, susideda iš valdymo žodžių, perduodamų į jungiklius SW_{vi}, $SW_{vi,cv}$, kur i yra skaičius tarp 0 ir 3, $SW_{id,no}$, $SW_{ch,no}$, $SW_{v1,no}$, $SW_{v2,no}$ ir $SW_{v3,no}$, identifikatoriaus registrą ID_T, papildomus registrus nuo $F0_T$ iki $F3_T$ ir pagrindinius registrus $S_{0,0}$ iki $S_{3,0}$.

Valdymo žodžiai yra perduodami per keletą valdymo laidų, prijungtų prie valdymo bloko 6p. Valdymo laidai gali būti dvifazinių valdymo laidų poros arba vienos fazės pavieniai valdymo laidai, priklausomai nuo to, kokios rūšies yra naudojami jungikliai (žr. FIG. 9E ir 9F).

Valdymo žodis, perduodamas į kiekvieną bazinio registro celę, turi vieną bendrą kanalą ir atskiras dalis kiekvienam baziniam registrui. Bendroji dalis valdo jungiklius SW_{ao}, SW_{bo} ir SW_{o}, esančius šerdies celėje (žr. FIG. 8). Tačiau reikia pažymėti, kad tai, kas čia aprašyta, turi būtĩ laikoma vien kaip pavyzdys ir kad yra galimi kai kurie kiti realizavimo atvejai.

ŠERDIES REGISTRO DARBO PAVYZDŽIAI

Schemos, parodytos FIG. nuo 18 iki 24, yra susijusios su FIG. 7. Nuorodos, esančios FIG. 7, taip pat tinka FIG nuo 18 iki 24. Tačiau dauguma nuorodų yra praleista dėl aiškumo. FIG. 18 iki 24 aprašyme registrų celių pažymėjimai reiškia visą registrą, einantį per visas 2p plokštes.

1. Objekto atminties 4p išrinkimas

Komanda mpx_mv:

Skaitoma objekto atmintis 4p ir nustatomi pagrindiniai registrai nuo operacijos mpx_mv. Išrinktas objektas yra perduodamas per šynas v0, v1, v2, v3, į pagrindinius registrus $S_{0,0}$, $S_{1,0}$, $S_{2,0}$, $S_{3,0}$, per šyną id į registrą ID ir per šyną env į

registrą ENV, kaip parodyta FIG. 18 storomis linijomis su strėlėmis, nukreiptomis į registro celes, į kurias yra perduodama. Tuo pačiu metu senas turinys, esantis pagrindiniuose registruose, yra įvedamas į objekto atmintį 4p, kaip uždaras reiškinys. Taigi, mpx_mv komanda įveda į atmintį esamą šerdies celės uždarą reiškinį ir pakrauna sekantį objekto atminties uždarą reiškinį į šerdies celę vykdymui.

Komanda fetch (iškvietimo, išrinkimo):

. FIG. 19 ir 20 yra parodyta situacija, kada į vieną iš pagrindinių registrų yra įvestas identifikatorius ir jis turi būti sukeistas su ta informacija, kurią jis pažymi. Identifikatorius, pvz., įvestas į $S_{2,o}$, žr. FIG. 19, perduodamas į objekto atmintį 4p, tada objekto atmintis randa identifikatorių ir tą turinį, kurį jis pažymi. Tas turinys yra perkeliamas į šynos laidą nuo v0 iki v3 ir galiausiai įvedamas į vertikalių stulpelį, esantį baziniame registre, t.y., $S_{2,o}$ iki $S_{2,3}$, žr. FIG. 20.

Ši operacija yra pradedama, perduodant identifikatorių į bazinį registrą $S_{2,o}$, į vertikalią šyną id, per šyną h0 ir jungiklį $id_{id,h0}$ (FIG. 19). Esanti atmintyje reikšmė gali būti perduodama iš bet kurio kito registro tokiu pat būdu.

Operacija tęsiasi (žr. FIG. 20), įvedant reikšmes, perduodamas iš objekto etminties 4p į šynų laidus v0, v1, v2, v3 į atitinkamus registrus $S_{2,o}$, $S_{2,1}$, $S_{2,3}$, reikšmes perduodant per jungiklius SW_{v0}, SW_{v1}, SW_{v2}, SW_{v3} ir šynas h0, h1, h2, h3.

Objekto atminties operacija make ir unify_id gali būti naudojamas tada, kai šerdies celėje esantį turinį reikia patalpinti į objekto atmintį.

Komanda make (daryti):

Pirmuoju operacijos make žingsniu norimo registro turinys yra perkeliamas taip, kaip parodyta FIG. 20, tik priešinga kryptimi. Operacijos metu taip pat yra perkeliamas ir aplinkos registro turinys. Objekto atmintyje yra atliekama asociatyvinė paieška, norint surasti objektą, turintį tą pačią informaciją, kaip ir informacija, esanti šerdies celėje. Jei objektas yra rastas, identifikatorius, žymintis objektą, yra gražinamas, priešingu atveju, jei objektas nėra rastas, yra sugražinamas nepanaudotas identifikatorius. Abiem atvejais identifikatorius yra perduodamas iš objekto atminties į identifikatoriaus registrą, esantį šerdies celėje, naudojant šynos laidą id. Kaip alternatyva, identifikatorius gali būti perduodamas į pagrindinį registrą esantį šio registro stulpelyje. Tokiu būdu, yra sudaromas ryšys tarp šerdies celės turinio ir identifikatoriaus.

Komanda unify_id

Operacija unify_id yra pavaizduota FIG. 21 ir jos metu identifikatorius yra perduodamas iš vieno registro, pvz., $S_{2,o}$ į visas vertikalias šynas id, env, v0, v1, v2, v3, sujungiant tą registro celę su horizontalia šyna h0 ir visas vertikalias šynas sujungiant su horizontalia šyna h0 per jungiklius $SW_{id,ho}$, $SW_{env,ho}$, SW_{vo}, $SW_{v1,ho}$ ir tt. Tai yra operacija, kuri gali būti naudojama tada, kai yra atliekama asociatyvi paieška ir pakeitimas, į kurį gali įeiti identifikatoriaus paieška, pakeičiant rastą reikšmę nauja, suskaidyta iki paprastosios reikšmės. Operacija, analogiška operacijai unify_id, gali savo pirmuoju žingsniu panaudoti komandą make, kad šerdies celės turiniui būtų priskirtas vienas identifikatorius, o antruoju žingsniu perkelti tą turinį į šynos kanalus, sujungtus su objekto atmintimi tam, kad į ją būtų įvestas identifikatorius ir tas turinys, kurį jis nurodo.

Skaitmeninis skaidymas

Skaitmeninio skaidymo metu objektas, t.y., tikslas, yra patalpinamas į pagrindinius registrus. Iš esmės, bendrame suskaidymo procese dalyvauja visa tikslo funkcija. Paprastai, registre $S_{0,0}$ yra laikomas komandos kodas, kuris yra skirtingas bitas, skirtas kitoms komandoms. Registrai $S_{1,0}$, $S_{2,0}$ yra naudojami dviguboms operacijoms, atliekamoms su dviem operandais, o registras $S_{1,0}$ yra naudojamas vienetinėms operacijoms, atliekamoms su vienu operandu. Bendrai paėmus, sarašų formoms yra naudojami juostiniai registrai, o jų turinys po suskaidymo yra perkeliamas į kairę.

Esminiai skaitmeniniai aritmetiniai veiksmai vyksta tarp tikslo registrų $S_{1,0}$ ir $S_{2,0}$. Pagrindinis skaitmeninio ALU sumavimo įtaisas yra prijungtas prie šių dviejų registrų. Kiti registrai gali būti naudojami papildomiems tikslams, tokioms komandoms vykdyti, kaip mul, div ir mod (dauginimo, dalijimo, pertvarkymo).

Gali būti naudojami sekantys komandų tipai:

vienetinės komandos

komanda yra patalpinta registre $S_{0,0}$, o registre $S_{1,0}$ yra operandas. Registrai $S_{2,0}$ ir $S_{3,0}$ nėra naudojami. Rezultatai iš skaitmeninio ALU yra gražinami į visus pagrindinius registrus. Jei nėra kanalų, jie yra įvedami į tarpinę atmintį, į papildomą registrą, arba į bazinį registrą.

dvigubos komandos

komanda yra patalpinta registre $S_{0,0}$, o registruose $S_{1,0}$ ir $S_{2,0}$ yra operandai. Registras $S_{3,0}$ nėra naudojamas. Rezultatai yra gražinami į visus pagrindinius registrus. Jei yra kanalai, rezultatai yra įvedami į tarpinę atmintį,- į papildomą registrą, arba į bazinį registrą.

komandos mul, div, mod

komanda yra patalpinta registre $S_{0,0}$, o registruose $S_{1,0}$ ir $S_{2,0}$ yra operandai. Registras $S_{3,0}$ gali būti naudojamas tarpinių rezultatų laikinam saugojimui. Galutinis rezultatas yra įvedamas į registrą $S_{1,0}$.

unify suskaidymas

unify suskaidymo metu yra naudojamas skaitmeninis ALU turinio, esančio registre $S_{0,0}$ sulyginimui su turiniu, esančiu registre $S_{1,0}$. Gali būti naudojami ir kiti pagrindiniai registrai, atliekant suvienodinimą. Etikečių (žymių) žodžiai, esantys registrų HEAD plokštėse, yra naudojami drauge su sulyginimo rezultatu, įvertinant sekantį veiksmą.

Komandos mul, div, mod atlieka pilną vidinį ryšį skaitmeniniame aritmetiniame bloke. Apskaičiuotos tarpinės reikšmės gali būti saugomos dinaminėje atmintyje, laidininkuose, esančiuose tarp skaitmeninio aritmetinio bloko ir pagrindinių registrų šerdies celėje, t.y., šynoje NU.

Struktūrinis skaidymas

Struktūrinio skaidymo metu objektas, t.y., tikslas, yra patalpinamas į pagrindinius registrus. Iš esmės, bendrame suskaidymo procese dalyvauja visi arba dalis pagrindinių registrų. Paprastai, pagrindiniame registre $S_{0,0}$ yra laikomas komandos kodas, kuris yra skirtingas bitas, skirtas kitoms komandoms.

Komanda map (planas, schema) turi funkciją f ir sąrašą $(e_1,\ldots,e_n)$, o argumentu yra sąrašas $(e_1,\ldots,e_n)$. Funkcija gražina sąrašą $(e_1,\ldots,e_n)$ po kiekvieno rezultatų panaudojimo, kur fe_1 rodo rezultatą, gautą panaudojus e_1 funkcijoje f_1.

Komandos mapping

Komanda map yra įvedama į papildomą registrą F0. Apdorojama funkcija yra įvedama į papildomą registrą F1. Sąrašas yra įvedamas į pagrindinius registrus $S_{0,0}$ iki $S_{3,0}$. Kaip parodyta FIG. 22a, elementai įvedami į pagrindinius registrus, yra perkeliami per du žingsnius aukštyn į bazinio registro matricą, t.y., turinys, esantis registruose $S_{x,0}$, yra perduodamas į registrą $S_{x,2}$, kur x yra skaičius tarp 0 ir 3. Perkėlimas yra atliekamas, naudojant vertikalius šynų laidus nuo v0 iki v3. Kaip parodyta FIG. 22b, turinys tada yra pertransliuojamas horizontaliai į bazinius registrus, pvz., turinys iš F0 yra nukopijuojamas į $S_{0,0}$ iki $S_{3,0}$, o turinys iš F1 yra nukopijuojamas į $S_{0,1}$ iki $S_{3,1}$. Jei elementas yra paprasta reikšmė (ne sąrašas), registro celės turinys yra patalpinamas į, pvz., $S_{1,2}$, o žemiau esančio registro celės turinys, pvz., $S_{1,1}$ yra perstumiamas per vieną žingsnį žemyn. Apdorojama funkcija dabar yra patalpinama į pagrindinio registro celę, pvz., $S_{1,0}$, o naudojamas elementas dabar yra patalpinamas į registro celę, esančią virš to pagrindinio registro celės, t.y., į $S_{1,1}$. Jei elementą sudaro sąrašas, tada šių registrų stulpelyje perstūmimai nėra daromi. Schemoje, parodytoje FIG.22c, yra laikoma, kad e_1, e_2 ir e_3 atstovauja paprastas reikšmes, o e_4 rodo įterptą sąrašą. Kiekvienas stulpelis, esantis bazinio registro matricoje, tada yra įvedamas į objekto atmintį, kaip uždaras reiškinys. Toliau, kiekvienas iš tų uždarų reiškinių yra pakraunami į šerdies celę tolesniam apdorojimui. Jei įvestas į atmintį uždaras reiškinys turi paprastą reikšmę, jis yra įvedamas į šerdies celę įprastu būdu, t.y., f yra įvedamas į $S_{0,0}$, o ei – į $S_{1,0}$, kaip parodyta FIG. 22d. Jei, iš kitos pusės, saugomas atmintyje uždaras reiškinys turi savyje įterptą sąrašą, jis yra

pakraunamas taip, kaip buvo aprašyta aukščiau ir parodyta FIG. 22a, tačiau čia e_1 yra pirmuoju sąrašo, kurį sudaro e_4, e_2, elementu, antruoju elementu sąraše, kurį sudaro e_4 ir tt. Tai įgalina map komandą veikti rekursyviai (pasikartojančiai) įterptuose sąrašuose.

Tokiu būdu, komanda map, turinti dviejų lygių struktūrą

(map, f, (e_1,.., e_n))

yra perrašoma į reiškinį

((f, e_1,.., f, e_n)),

kuris, jį apdorojus, yra perrašomas į vieno lygio struktūrą

(fe_1,.., fe_n),

kur fe_1 rodo rezultatą, taikant f e_1 atveju.

Komanda map, turinti trijų lygių (arba daugiau) struktūrą

(map, f, par(e_1,.., (e_k,..,e_m),.., e_n)),

kur (e_k,.., e_m) yra įterptas sąrašas, yra perrašomas į

par((f, e_1),.., (map, f, (e_k, ..,e_m)), .., (f, e_n))

kaip tarpinis žingsnis ir po to į

par((f, e_1), .., ((f, e_k), ..,(f, e_m)), ..,
(f, e_n)),

kuris po apdorojimo yra perrašomas į dviejų lygių struktūrą

par((fe_1, .., (fe_k, ..,fe_m), .., fe_n),

kur fe_1 rodo rezultatą, taikant f e_1 atveju ir

kur (fe_k, .., fe_m) yra įterptas sąrašas.

Tokiu būdu, funkcija f yra taikoma rekursyviai taikoma visiems elementams, esantiems argumentų sąraše.

Žemiau yra aprašomas iliustruojamas pavyzdys, kaip šerdies celė pertvarko ir vykdo map komandą. Sutrumpinimas reg yra naudojamas vietoje "registras", ident vietoj "identifier" (identifikatorius) ir storage (saugojimas atmintyje) vietoj "object storage" tam, kad aprašymas būtų trumpesnis. Komandos pavyzdys:

(map f (-1 -2 (-7 -8))),

kur f yra apibrėžiama, kaip f(x) = abs (x)+1. Mašininis

pavaizdavimas, naudojant mašininius identifikatorius, gali būti:

id1 : (map f id2)

id2 : (-1 -2 id3)

id3 : (-7 -8)

kur žymė id1 rodo uždarą reiškinį, turintį (map fid2) struktūrą ir tt.

Žemiau, i yra skaičius tarp 0 ir 3. Yra atliekami sekantys žingsniai:

1 žingsnis: map yra įvedamas į reg F0, f į reg F1, ir ident į reg $S_{0,0}$. S_{11} S_{11}

2 žingsnis: ident yra praplečiamas, t.y., reg $S_{0,0}$ turi -1, reg $S_{1,0}$ turi -2 ir reg $S_{2,0}$ turi ident id3.

3 žingsnis: reg $S_{1,0}$ turinys yra perkeliamas į reg $S_{1,2}$. Registrai, kurie yra pažymėti unused, lieka (nenaudojamas) nepaliesti.

4 žingsnis: map ir f yra transliuojami horizontaliai, t.y., reg $S_{1,1}$ turi savyje f ir reg $S_{1,0}$ turi map. Registrai, kurie yra pažymėti unused (nenaudojamas), lieka nepaliesti.

5 žingsnis: stulpeliai, kuriuose yra paprastoji reikšmė, įrašyta reg $S_{1,2}$, yra paspaudžiami per vieną žingsnį žemys, t.y., reg $S_{0,1}$ turi -1, reg $S_{0,0}$ turi f ir reg $S_{1,1}$ turi -2, reg $S_{1,0}$ turi f, o trečias stulpelis lieka nepaliestas.

6 žingsnis: kiekvienas stulpelis, esantis bazinio registro matricoje, yra iš naujo įvedamas į atmintį, kaip:

id1 : (id6 id7 id8)

id6 : (f -1)

id7 : (f -2)

id8 : (map f id3)

7 žingsnis: uždaras reiškinys, pažymėtas žyme id6, yra įvedamas į pagrindinius registrus, f – į reg $S_{1,o}$, o –1 į reg $S_{1,o}$.

8 žingsnis funkcija, t.y., f(x) = abs(x)+1, yra taikoma argumentui, turinčiam 2, kuris yra laikomas reg $S_{o,o}$.

9 žingsnis: atmintyje yra atliekama asociatyvinė žymės id6 paieška ir visos pasitaikiusios žymės id6 yra pakeičiamos 2:

id1 : (2 id7 id8)

id7 : (f –2)

id8 : (map f id3)

10 žingsnis: žingsniai nuo 7 iki 9 yra atliekami su žyme id7 ir rezultatu 3. Atmintis:

id1 : (2 3 id8)

id8 : (map f id3)

11 žingsnis: žingsniai nuo 1 iki 6 yra atliekami su žyme id8 ir id10, to rezultate du bazinės matricos stulpeliai yra įvedami į atmintį:

id1 : (2 3 id8)

id8 : (id9 id10)

id9 : (f –7)

id10 : (f –8)

12 žingsnis: žingsniai nuo 7 iki 9 yra atliekami su žymėm id9 ir id10 ir rezultatais 8 ir 9 atitinkamai. Atmintis:

id1 : (2 3 id8)

id8 : (8 9)

Tai reiškia: (2 3 (8 9)) – funkcija f buvo pritaikyta visiems elementams, esantiems argumento saraše.

Reikia pastebėti, kad aprašyti žingsniai gali būti atliekami šerdies celėje ir skirtingu, efektyvesniu būdu. Pavyzdžiui, vietoj tarpinio rezultato įvedimo į objekto atmintį, gali būti toliau atliekamas skaidymas šerdies celėje.

transpose (perkelti)

Formatas: (perkėlimo sąrašas).

Perkėlimo komanda yra įvedama į vieną iš papildomų registrų, pvz., F0, o sąrašo argumentas (sąrašų sąrašas) yra įvedamas į bazinio registro matricą, žr. FIG. 23. Perkeliamas bazinio registro matricos turinys. Tokiu būdu, komanda transpose, turinti trijų lygių struktūrą

(transpose,
((e1,1, .., e1,m),
....
((en,1, .., en,m)))

yra vykdoma taip, kad rezultatas yra perrašomas į dviejų lygių struktūrą

((e1,1, .., en,1),
....
(e1,m, .., en,m))

Pavyzdys paaiškinimui:

sąrašas - struktūra

((1 2 3 4),
(5 6 7 8),
(9 10 11 12),
(13 14 15 16), kur pirmasis sąrašas, t.y., (1 2 3 4), yra įvedamas į bazinio registro pirmąjį syulpelį, t.y., į bazinius registrus nuo $S_{0,0}$ iki $S_{0,3}$, o antrasis sąrašas, t.y., (5 6 7 8) įvedamas į antrąjį bazinio registro syulpelį, t.y., į bazinius registrus nuo $S_{1,0}$ iki $S_{1,3}$ ir tt.,

perkeliamas į

((1 5 9 13),
(2 6 10 14),
(3 7 11 15),
(4 8 12 16), kur pirmasis sąrašas, t.y., (1 5 9 13), yra įvedamas į bazinio registro pirmąjį syulpelį, t.y., į bazinius registrus nuo S0,0 iki S0,0 ir tt.

swap

(sukeitimas vietomis)

Formatas: (swap m list) (sukeisti vietomis m saraša).

Komanda swap yra vykdoma taip, kad komanda swap, turinti trijų lygių struktūra

(swap m ((e$_{1,1}$, ..),

..

(e$_{m,1}$, ..),

(e$_{m+1,1}$, ..),

...

(e$_{n,1}$, ..)))

kur sąrašų sąrašas, turintis elementus e$_{i,j}$, kai i ir j reiškia elemento padėties žymę bazinio registro matricoje, yra perrašomas į dviejų lygių struktūra

((e$_{1,1}$, ..),

..

(e$_{m+1,1}$, ..),

(e$_{m,1}$, ..),

...

(e$_{n,1}$, ..))

tokia, kad elementai (e$_{m,1}$, ..) pasikeičia vietomis su elementu (e$_{m+1,1}$, ..).

skip

praleisti

Formatas: (skip m list) (praleisti m saraša).

Komanda skip yra vykdoma taip, kad komanda skip, turinti trijų lygių struktūra

(skip m ((e$_{1,1}$, ..),

..

(e$_{m-1,1}$, ..),

(e$_{m,1}$, ..),

$(e_{m+1,1}, ..)$,

...

$(e_{n,1}, ..)))$

kur sąrašų sąrašas, turintis elementus $e_{i,j}$, kai i ir j reiškia elemento padėties žymėjimą bazinio registro matricoje, yra perrašomas į dviejų lygių struktūrą

$((e_{1,1}, ..)$,

..

$(e_{m-1,1}, ..)$,

$(e_{m+1,1}, ..)$,

...

$(e_{n,1}, ..))$

taip, kad elementai $(e_{m,1}, ..)$ yra pašalinami.

4. Sąrašo išskyrimas

Tikslo funkcija, į kurią įeina sąrašas, yra patalpinama į pagrindinį registrą. Jei sąrašas turi elementus, kurie yra įterptieji sąrašai, šie sąrašai yra saugomi vertikaliuose pagalbiniuose registruose.

Operacija expand_list (praplėsti sąrašą) gali būti atliekama viemnu ciklu. Bazinių registrų turinys yra perstumiamas įstrižainės krytimi žemyn ir į dešinę per vieną žingsnį, išskyrus turinį, esantį pagrindiniame registre, kuris yra perkeliamas į vertikalią šyną ir įterptas į aukščiausią bazinį registrą tame stulpelyje (žr. FIG. 24). Gali būti pakartotinai panaudota komanda expand_list, kad pagrindinio registro celės būtų pripildytos duomenimis.

Nors čia aprašytas vienas iš tinkamesnių pagal šį išradimą šerdies celės ir registrų celių struktūrų, reikia suprasti, kad išradimas neapsiriboja vien tik šia konstrukcija ir kad gali būti daromi pakeitimai, nenutolstant nuo išradimo esmės.

221

KITAS ŠERDIES CELĖS VARIANTAS

Fig. 25 yra parodytas kitas šerdies celės variantas. Tiems patiems šerdies celės elementams žymėti yra naudojami tie patys pažymėjimai, kaip ir FIG. 7. Šerdies celės elementai, kurie yra skirtingai montuojami, negu FIG. 7, yra pažymėti ženklu ". Pagrindinis skirtumas tarp varianto, parodyto FIG. 7, kuris yra pranašesnis, ir varianto, parodyto FIG. 25, yra tai, kad bazinio registro celės S"$_{0,0}$, S"$_{1,0}$, S"$_{2,0}$ ir S"$_{3,0}$ yra sujungiamos su laidu res" be jungiklių. Šynų laidai C_{1d} ir C_{f} yra praleisti. Be to, terminalas W (west, vakarai), esantisb azinio registro celėje S"$_{0,y}$, nėra sujungtas su terminalu E (east, rytai), esantis bazinio registro celėse $S_{3,y}$, kur y yra skaičius nuo 0 iki 3. Vietoj to, šie terminalai turi signalą f (false, klaidingas). Sujungimo takeliai, esantys šerdies celės viduje, gali būti padaryti kiek skirtingai kai kurių komandų atžvilgiu, tačiau tai nėra funkcinis skirtumas, tik skirtumas yra tame, kokios šerdies registro celės turi būti valdomos. Jungtys, esančios tarp registro celių taip pat truputį skiriasi, tačiau tai irgi priklauso nuo to, kokie vidiniai jungikliai, esantys kiekvienoje registro celėje, yra valdomi.

Fig. 26 yra parodytas kitas šerdies celės variantas tuo tikslu, kad būtų pateiktas pavyzdys tokio fakto, kad jungikliai, kurie turi būti celėje valdomi, gali būti išdėstyti skirtingai, tačiau celės gali turėti tas pačias funkcijas. Tiems patiems šerdies celės elementams žymėti yra naudojami tie patys pažymėjimai, kaip ir FIG. 8. Šerdies celės elementai, kurie yra skirtingai montuojami, negu FIG. 8, yra pažymėti ženklu ". Pagrindinis skirtumas tarp varianto, parodyto FIG. 8 ir 26, yra tai, kad parodytame FIG. 26 terminalas C ir jungiklis SW_c yra praleisti, o terminalai V ir H kiekvienas turi tik vieną jungiklį, o terminalai L, L' ir D su jungikliais yra naudojami vietoj terminalų Da ir Db su jungikliais.

Nors išradimas ir yra aprašytas, remiantis tam tikru variantu, specialistas turi suprasti, kad gali būti padaryti įvairūs pakeitimai ir panaudojami kiti ekvivalentiški elementai, nenutolstant nuo pagrindinės idėjos ir apimties. Be to, gali būti atlikti patobulinimai, nenutolstant nuo esminių išradimo nurodymų.

APIBRĖŽTIS

1. Aritmetinių veiksmų struktūrinio apdorojimo metodas, į kurį įeina:

a) duomenų žodžių įvedimas į keletą registrų, o kiekvienas iš duomenų žodžių turi informacinę dalį,

b) minėti duomenų žodžiai yra surikiuoti į sąrašus, laikančius nustatyto skaičiaus minėtų registrų atmintyje kiekvieną iš minėtų sąrašų,

c h a r a k t e r i z u o j a m a s tuo, kad

c) kiekvienas duomenų žodis turi dalį su žyme, šios dalies žymė rodo, ar šis registras yra naudojamas, ar ne, minėta kiekvieno minėto žodžio dalis, turinti žymę ir esanti minėtuose sąrašuose, esančiuose minėtuose registruose, yra sužymėta naudojimui, kad vienas iš minėtų sąrašų turi ne mažiau, kaip dalį, saugomą viename iš registrų ir kad minėtas sąrašas, turintis dalį, saugomą viename iš registrų, turi komandų sąrašą, rodantį, kokio tipo tai yra sąrašas,

d) minėti sąrašai, saugomi minėtuose registruose, yra yra sutvarkyti sąrašų medžio pavidalu, iš kurių vienas iš sąrašų yra šaknies sąrašas ir kur santykis tarp minėtų sąrašų yra akivaizdus iš minėtų sąrašų tvarkos minėtuose registruose,

e) minėti registrai yra valdomi, naudojant minėtas sąrašo

komandas, kurios yra įvestos į minėtus sąrašus, sąugomus minėtuose registruose, siekiant pertvarkyti minėtus sąrašus tarp minėtų registrų ir tam, kad registrų turinys būtų įvedamas/išvedamas atitinkamai pagal minėtas sąrašo komandas,

minėtos sąrašo komandos yra struktūriniai aritmetiniai operatoriai, atitinkantys aukšto lygio programinės kalbos konstrukciją.

2. Metodas pagal 1 punktą, c h a r a k t e r i z u o j a m a s tuo, kad įvesto į atmintį sąrašų medžio identifikatorius yra saugomas atskirame identifikatoriaus registre.

3. Metodas pagal 1 punktą, c h a r a k t e r i z u o j a m a s tuo, kad įvesto į atmintį sąrašų medžio aplinka yra saugoma atskirame aplinkos registre.

4. Metodas pagal bet kurį iš punktų nuo 1 iki 3, c h a r a k t e r i z u o j a m a s tuo, kad įvedamo į atmintį minėto sąrašų medžio šaknies sąrašas yra patalpinamas skirtinguose registruose priklausomai nuo minėto įvedamo į atmintį medžio lygio.

5. Metodas pagal 4 punktą, c h a r a k t e r i z u o j a m a s tuo, kad kai kurie iš minėtų registrų yra surikiuojami bazinių registrų matricoje, įskaitant ir pagrindinių registrų eilutę.

6. Metodas pagal 5 punktą, c h a r a k t e r i z u o j a m a s tuo, kad medis, turintis tik vieną lygį, yra įvedamas į pagrindinio registro atmintį.

7. Metodas pagal 5 punktą, c h a r a k t e r i z u o j a m a s tuo, kad medžio, turinčio tris lygius, šaknies sąrašas yra įvedamas į minėtą pagrindinį registrą, o jo šakiniai sąrašai – į minėtus bazinius registrus.

8. Metodas pagal 5 punktą, c h a r a k t e r i z u o j a m a s tuo, kad minėtų registrų išorėje turi papildoma

rinkinį registrų, vadinamą pagalbiniais registrais.

9. Metodas pagal 8 punktą, c h a r a k t e r i z u o j a m a s tuo, kad medžio, turinčio tris lygius, jo šaknies sąrašas yra įvedamas į minėtą pagalbinį registrą, o vienas iš jo elementų įvedamas į minėtą registrų matricą.

10. Metodas pagal bet kurį iš punktų nuo 1 iki 9, c h a r a k t e r i z u o j a m a s tuo, kad informacija apie tai, kokio tipo turi būti atliktas suskaidymas, gali būti gaunama iš minėto šaknies sąrašo.

11. Metodas pagal bet kurį iš punktų nuo 1 iki 9, c h a r a k t e r i z u o j a m a s tuo, kad informacija apie tai, kokio tipo turi būti atliktas suskaidymas, gali būti gaunama iš minėto šaknies sąrašo.

12. Metodas pagal punktą 10, c h a r a k t e r i z u o j a m a s tuo, kad jei minėtas tipas yra skirtas funkciniam naudojimui, pirmasis minėto šaknies sąrašo elementas turi komandos kodą arba medžio sąrašų šaknį, rodančią funkcijos priskyrimą.

13. Metodas pagal bet kurį iš punktų nuo 1 iki 12, c h a r a k t e r i z u o j a m a s tuo, kad maksimalus žodžių skaičius sąraše yra keturi.

14. Metodas pagal bet kurį iš punktų nuo 1 iki 13, c h a r a k t e r i z u o j a m a s tuo, kad maksimalus minėto sąrašų medžio gylis yra trys lygiai.

15. Metodas pagal punktą 13, c h a r a k t e r i z u o j a m a s tuo, kad jei minėtas gylis yra trys lygiai ir jei minėtuose registruose minėta sąrašo komanda nurodo, kad minėtas šaknies sąrašas turi daugiau, negu vieną šaką, į minėtus registrus yra įvedama tik viena iš minėtų šakų.

16. Metodas pagal bet kurį iš punktų nuo 1 iki 12, c h a r a k t e r i z u o j a m a s tuo, kad struktūrinis suskaidymas yra atliekamas su duomenimis, esančiais minėtuose registruose, tokiuose, kaip baziniai registrai arba esančiais baziniuose ir pagalbiniuose registruose.

17. Metodas pagal punktą 16, c h a r a k t e r i z u o j a m a s tuo, kad komanda mapping yra atliekama tokiu būdu, kaip komanda map turinti dviejų lygių struktūra

(map, f, (e_1,.., e_n))

kur komanda map yra įrašoma į vieną iš papildomų registrų, funkcija map yra įrašoma į antrąjį iš minėtų papildomų registrų, o komandos elementų sąrašas e_i – minėtus bazinius registrus, visa tai yra perrašoma, valdant nuo valdymo bloko į

((f, e_1,.., f, e_n)),

kur yra perduodama funkcija, o minėti komandos elementai yra pertvarkomi tokiu būdu, kad kiekviename stulpelyje tarp minėtų bazinių registrų funkcija yra patalpinama į žemiausią bazinį registrą minėtame stulpelyje, o vienas iš minėtų komandos elementų yra patalpinamas į antrąjį žemiausią bazinį registrą minėtame stulpelyje.

18. Metodas pagal punktą 16, c h a r a k t e r i z u o j a m a s tuo, kad komanda transpose, turinti trijų lygių struktūrą, yra vykdoma taip

(transpose,

(($e_{1,1}$, .., $e_{1,m}$),

....

(($e_{n,1}$, .., $e_{n,m}$)))

kur komanda transpose yra įrašoma į papildomą registrą, o komandos elementų $e_{i,j}$ sąrašų sąrašas, kur i ir j yra komandos padėties minėtose matricose nuorodos, į bazinių registrų matricą tokiu būdu, kad bazinio registro komplektą sudaro elementų kvadratas, o perrašymas yra atliekamas nuo valdymo bloko komandos į dviejų lygių struktūrą

(($e_{1,1}$, .., $e_{n,1}$),

..

($e_{1,m}$, .., $e_{n,m}$))

kur papildomų registrų reikšmės yra paliekamos tuščios, o bazinių registrų matricos yra perkeliamos taip, kad atsidurtų veidrodinėje padėtyje matricos įstrižainės atžvilgiu.

19. Metodas pagal punktą 16, c h a r a k t e r i z u o j a m a s tuo, kad komanda swap yra atliekama tokiu būdu, kaip komanda swap, turinti trijų lygių struktūrą

(swap m ((e$_{1,1}$, ..),
..
(e$_{m,1}$, ..),
(e$_{m+1,1}$, ..),
...
(e$_{n,1}$, ..)))

kur sarašų sarašas, turintis elementus e$_{i,j}$, kai i ir j reiškia elemento padėties žymę bazinio registro matricoje, kai bazinio registro komplektą sudaro elementų kvadratas, yra perrašomas į dviejų lygių struktūrą, valdant nuo valdymo bloko

((e$_{1,1}$, ..),
..
(e$_{m+1,1}$, ..),
(e$_{m,1}$, ..),
...
(e$_{n,1}$, ..))

taip, kad elementas (e$_{m,1}$, ..) pasikeičia vietomis su elementu (e$_{m+1,1}$, ..).

20. Metodas pagal punktą 16, c h a r a k t e r i z u o j a m a s tuo, kad komanda skip yra atliekama tokiu būdu, kaip komanda skip, turinti trijų lygių struktūrą

(skip m ((e$_{1,1}$, ..),
..
(e$_{m-1,1}$, ..),
(e$_{m,1}$, ..),
(e$_{m+1,1}$, ..),
...
(e$_{n,1}$, ..))

kur sąrašų sąrašas, turintis komandos elementus $e_{i,j}$, kai i ir j reiškia komandos elemento padėties žymėjimą bazinio registro matricoje, yra perrašomas į dviejų lygių struktūrą, valdant nuo valdymo bloko komandos

$((e_{1,1}, ..),$

$..$

$(e_{m-1,1}, ..),$

$(e_{m+1,1}, ..),$

$...$

$(e_{n,1}, ..))$

taip, kad elementai $(e_{m,1}, ..)$ yra pašalinami.

21. Aritmetinis blokas, skirtas struktūriniams aritmetiniams veiksmams, į kurį įeina:

a) ne mažiau, kaip viena įvedimo/išvedimo priemonė (v0, v1, v2, v3, id, env), skirta duomenų sąrašų įvedimui ir išvedimui,

b) keletas registrų ($S_{0,0}$ iki $S_{3,3}$, F0 iki F3, ID, ENV),

kiekvienas iš kurių yra pritaikytas duomenų žodžių saugojimui, o minėti duomenų žodžiai surikiuoti minėtuose sąrašuose,

kiekvienas duomenų žodis turintis informacinę dalį,

kiekvienas iš minėtų sąrašų galimas saugoti nustatytame skaičiuje minėtų registrų,

c h a r a k t e r i z u o j a m a s tuo, kad

c) kiekvienas duomenų žodis turi žymės vietą, minėta žymės vieta turi žymę, rodančią, ar šis registras yra naudojamas, ar ne,

minėta kiekvieno minėto žodžio dalis, turinti žymę ir esanti minėtuose registruose,

yra sužymėta naudojimui, nurodant, kad vienas iš minėtų sąrašų turi ne mažiau, kaip dalį, saugomą viename iš registrų ir kad minėtas sąrašas, turintis dalį, saugomą viename iš registrų, turi komandų sąrašą, rodantį, kokio

tipo yra tas sąrašas,

d) minėti sąrašai, saugomi minėtuose registruose, yra sutvarkyti sąrašų medžio pavidalu, iš kurių vienas iš sąrašų yra šaknies sąrašas ir kur santykis tarp minėtų sąrašų yra akivaizdus iš minėtų sąrašų tvarkos minėtuose registruose,

e) minėti registrai yra valdomi, naudojant minėtas sąrašo komandas, kurios yra įvestos į minėtus sąrašus, saugomus minėtuose registruose, siekiant pertvarkyti minėtus sąrašus tarp minėtų registrų ir tam, kad registrų turinys būtų įvedamas/išvedamas atitinkamai pagal minėtas sąrašo komandas,

minėtos sąrašo komandos yra struktūriniai aritmetiniai operatoriai, atitinkantys aukšto lygio programinės kalbos konstrukciją.

22. Aritmetinis blokas pagal 21 punktą, c h a r a k t e r i z u o j a m a s tuo, kad vienas iš sąrašų yra šaknies sąrašas.

23. Aritmetinis blokas pagal 21 arba 22 punktą, c h a r a k t e r i z u o j a m a s tuo, kad maksimalus numatytų registrų, kuriuose yra saugomas sąrašas, skaičius yra keturi.

24. Aritmetinis blokas pagal punktus nuo 21 iki 23, c h a r a k t e r i z u o j a m a s tuo, kad maksimalus minėtų sąrašų medžio gylis yra trys lygiai.

25. Aritmetinis blokas pagal 24 punktą, c h a r a k t e r i z u o j a m a s tuo, kad jei minėtas gylis yra trys lygiai ir jei minėto sąrašo komanda, esanti minėtame šaknies sąraše, saugomame minėtuose registruose rodo, kad minėtas šaknies sąrašas turi vieną arba daugiau negu vieną šaką, minėtos valdymo priemonės (6p) yra pritaikytos laikyti minėtų registrų atmintyje tik vieną iš minėtų šakų.

26. Aritmetinis blokas pagal punktus nuo 21 iki 25, c h a r a k t e r i z u o j a m a s tuo, kad yra ne mažiau, kaip vienas papildomas registras (ENV), kuriame yra saugoma įvesto į atmintį medžio aplinka.

27. Aritmetinis blokas pagal punktus nuo 21 iki 26, c h a r a k t e r i z u o j a m a s tuo, kad yra ne mažiau kaip vienas papildomas registras (ENV), kuriame yra saugoma įvesto sąrašų medžio aplinka.

28. Aritmetinis blokas pagal punktus nuo 21 iki 27, c h a r a k t e r i z u o j a m a s tuo, kad į jį įeina registrų matrica ($S_{0,0}$ iki $S_{3,3}$), turinti periferinę eilutę ($S_{0,0}$ iki $S_{3,0}$) su pagrindiniais registrais ir minėtų matricų stulpelius, turinčius bazinius registrus.

29. Aritmetinis blokas pagal 28 punktą, c h a r a k t e r i z u o j a m a s tuo, kad turi papildomų registrų, (nuo F0 iki F3), easnčių už minėtos matricos.

30. Aritmetinis blokas pagal punktus nuo 21 iki 29, c h a r a k t e r i z u o j a m a s tuo, kad minėtos valdymo priemonės (6p) yra pritaikytos patalpinti minėto sąrašų medžio šaknies sąrašą skirtinguose registruose priklausomai nuo minėto įvedamo į atmintį medžio lygio.

31. Aritmetinis blokas pagal 28 punktą, c h a r a k t e r i z u o j a m a s tuo, kad minėtos valdymo priemonės yra pritaikytos įvedimui į atmintį medžio, turinčio tik vieną lygį, kuris yra įvedamas į pagrindinio registro atmintį.

32. Aritmetinis blokas pagal 28 punktą, c h a r a k t e r i z u o j a m a s tuo, kad minėtos valdymo priemonės yra pritaikytos įvesti į atmintį medį, turintį du lygius, jo šaknies sąrašą įvedant į minėtus pagrindinius registrus (nuo $S_{0,0}$ iki $S_{3,0}$), o jo šakinius sąrašus - į minėtus bazinius registrus (nuo $S_{0,0}$ iki $S_{0,3}$

33. Aritmetinis blokas pagal 28 punktą, c h a r a k t e r i z u o j a m a s tuo, kad minėtos valdymo priemonės (6p) yra pritaikytos įvesti į atmintį medį, turintį tris lygius, jo šaknies sąrašą įvedant į minėtus papildomus registrus (nuo F0 iki F3), o vieną iš jo elementų įvedant į minėtų registrų matricą.

34. Aritmetinis blokas pagal bet kurį iš punktų nuo 21 iki 33, c h a r a k t e r i z u o j a m a s tuo, kad informacija apie tai, kokio tipo turi būti atliktas suskaidymas, gali būti gaunama iš minėto šaknies sąrašo tipo per minėtas valdymo priemones (6p).

35. Metodas pagal punktą 34, c h a r a k t e r i z u o j a m a s tuo, kad minėta informacija, esanti minėtame šaknies sąrašo tipe turi komandos kodą, rodantį, kokio tipo komanda turi būti atlikta.

36. Metodas pagal punktą 34, c h a r a k t e r i z u o j a m a s tuo, kad pirmasis minėto šaknies sąrašo elementas turi komandos kodą arba medžio sąrašų šaknį, rodančią funkcijos priskyrimą, kurią naudoja valdymo priemonė (6p), nustatant, kokio veiksmo reikia imtis, norint suskaidyti minėtą šaknies sąrašą.

37. Aritmetinis blokas pagal punktus nuo 21 iki 36, c h a r a k t e r i z u o j a m a s tuo, kad minėti registrai yra sutvarkyti logine tvarka plokščių skerspjūvio požiūriu (NUM, HEAD, TYPE, LAZY, CLOS/SIMPLE), esančių šerdies celėje, kiekviena plokštė turi daugiausiai vieną registrų celę iš kiekvieno registro, kur kiekviena registro celė gali turėti atmintyje vieną informacijos bitą ir kur plokštėje esančios registro celės yra sujungiamos viena su kita.

38. Aritmetinis blokas pagal punktą 37, c h a r a k t e r i z u o j a m a s tuo, kad kai kurie iš registrų yra ilgesni už kitus rtegistrus, tokiu būdu, kai kurios iš

minėtų plokščių (TYPE, WHERE, LAZY, CLOS/SIMPLE), turi

tik registrų celes, priklausančias minėtiems ilgesniams registrams (FIG. 12).

39. Aritmetinis blokas pagal punktą 37, c h a r a k t e r i z u o j a m a s tuo, kad bent keletas iš minėtų registrų, vadinamų baziniais, yra surikiuoti eilutėmis ir stulpeliais matricose N kartų, N registruose, kur N yra sveikas skaičius; minėtų bazinių registrų celių elementai yra tarpusavyje sujungti bitais, o kiekvienas minėto bazinio registro bitas yra prijungtas prie stulpelio laido (v0, v1, v2, v3), o kiekviena eilutė (h0, h1, h2, h3) yra prijungta prie eilutės laido, kiekvienam stulpelio laidui ir kiekvienam eilutės laidui yra numatytas valdomas jungiklis (SW_{v0}, SW_{v1}, SW_{v2}, SW_{v3}), esantis kiekviename eilučių ir stulpelių, turinčių tuos pačius numerius, susikirtimo taške, kiekvienas bazinis registras turi valdomą registro jungtį bent su artimiausiu eilutės laidu ir su stulpelio laidu; ir kad tarp gretimų bazinių registrų, tiek išilgai minėtų eilučių, tiek ir išilgai minėtų stulpelių, yra jungtis.

40. Aritmetinis blokas pagal punktą 39, c h a r a k t e r i z u o j a m a s tuo, kad minėtos valdymo priemonės (6p) yra pritaikytos valdyti jungiklius ir registrų jungtis ir taip sudaro vieną iš trijų rūšių sujungimų, priklausomai nuo reikiamos vykdyti komandos:

a) paprasta jungtis nuo vieno registro į kitą viena kryptimi,

b) dvi atskiros jungtys tarp registrų, po vieną bet kurioje iš krypčių,

c) bendra laiko jungtis tarp registrų, per kurią yra perduodami esantys saraše elementai viena kryptimi ir kita kryptimi per dvi fazių sekas.

41. Aritmetinis blokas pagal punktą 40, c h a r a k t e r i z u o j a m a s tuo, kad kiekviena registro celė registre turi:

a) vidinį vieno bito registrą (r_R),

b) ne mažiau, kaip vieną vidinę vieno laido šyną (a_R, b_R), sujungtą su minėtu vieno bito registru,

c) ne mažiau, kaip vieną vidinę valdomą jungtį, į kiekvieną iš kurių įeina jungtis (SW_N, SW_E, SW_{vo}, SW_{v1}, SW_{H1}, SW_{HO}, SW_{Db}, SW_{ao}, SW_{a1}, SW_{bo}, SW_{b1}), valdoma iš minėtų valdymo priemonių (6p), įgalinant sujungti minėtą vieno laido šyną su vienu iš sekančių elementų: su šyna, esančia už celės ribų, viena iš celių, priklausančių kitam iš minėtų registrų.

42. Aritmetinis blokas pagal punktą 41, c h a r a k t e r i z u o j a m a s tuo, kad ne mažiau, kaip vienas vidinis vieno bito registras (r_R) turi įvedimo buferį (Q1), tokį, kaip invertorius, ir išvedimo buferį (Q2), tokį, kaip invertorius bei valdomą jungiklį (SW_Q), sujungtą su minėtais buferiais.

43. Aritmetinis blokas pagal punktą 42, c h a r a k t e r i z u o j a m a s tuo, kad įvedimo buferis ir minėtas išvedimo buferis per valdomus jungiklius atskirai yra sujungiami su ne mažiau, kaip viena iš minėtų vieno laido šynų (a_R, b_R).

44. Aritmetinis blokas pagal punktą 39, c h a r a k t e r i z u o j a m a s tuo, kad kiekviena registro celė turi:

a) pirmą (a_R) ir/arba antrą (b_R) vidinę vieno laido šyną,

b) vidinį vieno bito registrą (r_R),

c) pirmą grupę vidinių valdomų jungčių, turinčių nuo valdymo priemonės valdomus jungiklius ir tuo įgalinančių sujungti minėtą pirmą šyną su minėto vidinio registro pirmuoju įėjimu ir kai kurių išorinių šynų grupę sujungti su kitomis registro celėmis,

d) antrą grupę vidinių valdomų jungčių, į kurias įeina jungtys, valdomos iš minėtų valdymo priemonių, įgalinant sujungti minėtą antrąją šyną su antruoju minėto vidinio registro įėjimu ir su sekančia grupe iš keleto išorinių šynų, sujungtų su kitomis registro celėmis.

e) trečią grupę vidinių valdomų jungčių, kurias įeina jungtys, valdomos iš minėtų valdymo priemonių, įgalinant sujungti minėto vidinio vieno bito registro išėjimą su trečiąja keletos išorinių šynų grupe.

45. Aritmetinis blokas pagal punktą 44, c h a r a k t e r i z u o j a m a s tuo, kad priklausomai nuo minėtų registrų padėties, kai kurios iš registrų celių turi fiksuotas reikšmes (f), pastoviai sujungtas su ne mažiau, kaip viena jos vidine jungtimi, nesančia minėtu vidiniu registru.

46. Aritmetinis blokas pagal punktą 44, c h a r a k t e r i z u o j a m a s tuo, kad kiekviena iš minėtų valdomų jungčių yra ne mažiau, kaip viena valdoma jungtis, turinti vieną iš sekančių įrenginių: MOS-FET (FIG> 9E), du papildomus MOS-FET (FIG. 9F).

47. Aritmetinis blokas pagal punktą 42, c h a r a k t e r i z u o j a m a s tuo, kad kiekvienas invertorius turi vieną iš sekančių įrenginių: du papildomus MOS-FET (FIG. 9C), du MOS-FET padidinto tipo (FIG. 9A), vieną MOS-FET padidinto tipo ir vieną praleidžiamo tipo MOS-FET (FIG. 9B).

48. Aritmetinis blokas pagal punktą 42, c h a r a k t e r i z u o j a m a s tuo, kad komparatorius (G1, G2) yra prijungtas minėtų registrų turinio sulyginimui ir sulyginamų rezultatų perdavimui į išorinės šynos, vadinamos išrinkimo, laidą.

49. Aritmetinis blokas pagal punktą 48, c h a r a k t e r i z u o j a m a s tuo, kad minėtas komparatorius turi:

a) pirmą NAND ventilį, turintį pirmąjį įėjimą, prijungtą prie minėto pirmojo buferio (Q1) šono, esančio iš minėto vidinio registro jungiklio pusės, o antrąjį įėjimą, prijungtą prie minėto vidinio registro jungiklio jungties, esančios tarp jo ir minėto antrojo buferio (Q2),

b) antrą NAND ventilį, turintį pirmąjį įėjimą, prijungtą prie minėto antrojo buferio (Q2) šono, esančio iš minėto vidinio registro jungiklio pusės, o antrąjį įėjimą, prijungtą prie minėto vidinio registro jungiklio jungties, esančios tarp jo ir minėto pirmojo buferio (Q1);

minėtų NAND ventilių išėjimai yra sujungti tarpusavyje ir sujungti su minėtais išorinės šynos laidais, esančiais minėtoje šynoje, vadinamoje išrinkimo šyna.

50. Aritmetinis blokas pagal punktą 49, c h a r a k t e r i z u o j a m a s tuo, kad kiekvienas iš minėtų NAND ventilių turi dvi eiles nuosekliai sujungtų MOS-FET tranzistorių, turinčių nuosekliai sujungtus kanalus, jų ventilius sudaro NAND ventilių įėjimai, o aukščiausiai esančio MOS-FET tranzistoriaus išvedimo kanalas yra minėtos išorinės šynos laidas.

51. Aritmetinis blokas pagal punktą 49, c h a r a k t e r i z u o j a m a s tuo, kad jis turi minėtas registrų matricas, valdomas automatiškai iš minėtų valdymo priemonių (6p), kad būtų atliekamas duomenų objekto, patalpinto minėtuose baziniuose registruose, struktūrinis suskaidymas.

52. Aritmetinis blokas pagal punktą 49, c h a r a k t e r i z u o j a m a s tuo, kad komanda mapping yra pritaikyta vykdyti tokiu būdu, kad komanda map, turinti dviejų lygių struktūrą

$$(map,\ f,\ (e_1,..,\ e_n))$$

kur komanda map yra perrašoma į vieną iš minėtų papildomų registrų (nuo F0 iki F3), funkcija map yra perrašoma į antrąjį minėtą papildomą registrą, o komandos elementų e_i sąrašas, esantis minėtuose baziniuose registruose (nuo $S_{0,0}$ iki $S_{3,3}$), yra pritaikyta perrašymui, valdant nuo minėtų valdymo priemonių, į

$$((f,\ e_1,..,\ f,\ e_n));$$

kur perduodama funkcija ir minėti komandų elementai yra pertvarkomi tokiu būdu, kad kiekvienam stulpeliui tarp minėtų bazinių registrų skirta funkcija yra patalpinama į žemiausią bazinį registrą, esantį minėtame stulpelyje, o vienas iš minėtų komandos elementų yra patalpinamas į antrą žemiausią bazinį registrą, esantį minėtame stulpelyje.

53. Aritmetinis blokas pagal punktą 52, c h a r a k t e r i z u o j a m a s tuo, kad komanda transpose yra pritaikyta vykdyti tokiu būdu, kad komanda transpose, turinti trijų lygių struktūrą,

(transpose,
((e1,1, .., e1,m),
....
((en,1, .., en,m)))

yra įrašoma į papildomą registrą, o komandos elementų $e_{i,j}$ sąrašų sąrašas, kur i ir j yra komandos padėties minėtose matricose nuorodos, įrašomas į bazinių registrų matricą tokiu būdu, kad bazinio registro komplektą sudaro elementų stačiakampis, o perrašymas yra atliekamas nuo valdymo bloko komandos į

((e1,1, .., en,1),
..
(e1,m, .., en,m))

kur papildomų registrų reikšmės yra paliekamos tuščios, o bazinių registrų matricos yra perkeliamos taip, kad atsidurtų veidrodinėje padėtyje matricos įstrižainės atžvilgiu.

54. Aritmetinis blokas pagal punktą 51, c h a r a k t e r i z u o j a m a s tuo, kad komanda swap yra atliekama tokiu būdu, kad komanda swap, turinti trijų lygių struktūrą

(swap m ((e1,1, ..),
..
(em,1, ..),
(em+1,1, ..),
...
(en,1, ..)))

kur sąrašų sąrašas, turintis elementus $e_{i,j}$, kai i ir j reiškia elemento padėties žymę bazinio registro matricoje, kai bazinio registro komplektą sudaro elementų stačiakampis, yra pritaikytas perrašyti, valdant nuo valdymo bloko, į

236

$((e_{1,1}, ..),$

$..$

$(e_{m+1,1}, ..),$

$(e_{m,1}, ..),$

$...$

$(e_{n,1}, ..))$

taip, kad elementas $(e_{m,1}, ..)$ pasikeičia vietomis su elementu $(e_{m+1,1}, ..)$.

55. Aritmetinis blokas pagal punktą 51, c h a r a k t e r i z u o j a m a s tuo, kad komanda skip yra pritaikyta vykdyti tokiu būdu, kad komanda skip, turinti trijų lygių struktūrą

$(skip\ m \quad ((e_{1,1}, ..),$

$..$

$(e_{m-1,1}, ..),$

$(e_{m,1}, ..),$

$(e_{m+1,1}, ..),$

$...$

$(e_{n,1}, ..)))$

kur sąrašų sąrašas komandų elementų $e_{i,j}$, kai i ir j reiškia komandos elemento padėties žymėjimą bazinio registro matricoje, kai bazinių registrų komplektas turi elementų stačiakampį, yra pritaikytas perrašyti, valdant nuo valdymo bloko, į

$((e_{1,1}, ..),$

$..$

$(e_{m-1,1}, ..),$

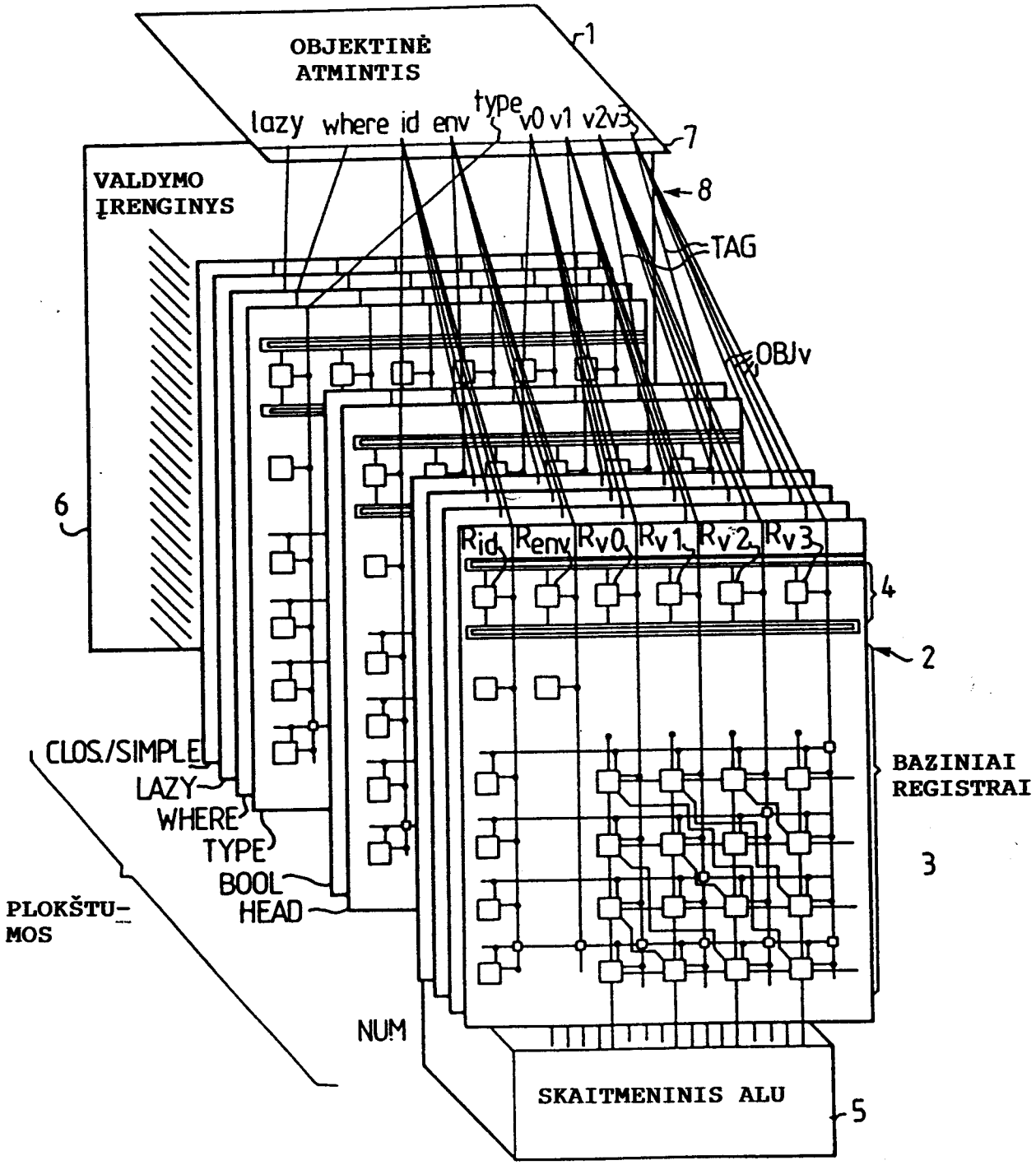
$(e_{m+1,1}, ..),$

$...$

$(e_{n,1}, ..))$

taip, kad elementai $(e_{m,1}, ..)$ yra pašalinami.

55. Aritmetinis blokas pagal punktą 21, c h a r a k t e r i z u o j a m a s tuo, kad jis sudaro centrinio procesoriaus dalį.


OBJEKTINĖ ATMINTIS
1
lazy where id env type v0 v1 v2 v3
7
VALDYMO ĮRENGINYS
8
TAG
OBJv
6
Rid Renv Rv0 Rv1 Rv2 Rv3
4
2
BAZINIAI REGISTRAI
3
CLOS/SIMPLE
LAZY
WHERE
TYPE
BOOL
HEAD
PLOKŠTU-MOS
NUM
SKAITMENINIS ALU
5

PIEŠ.1

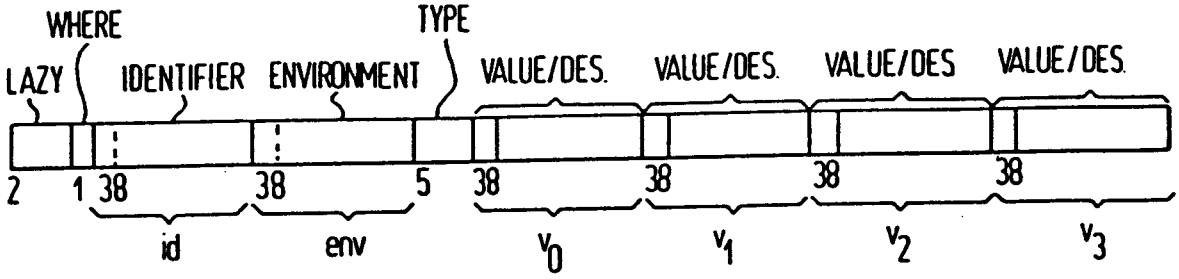


FIG.2A

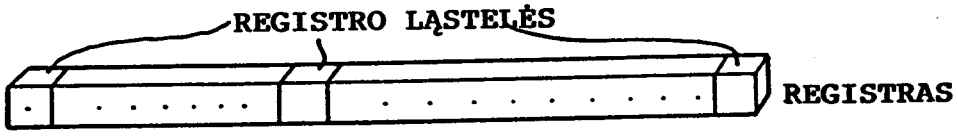
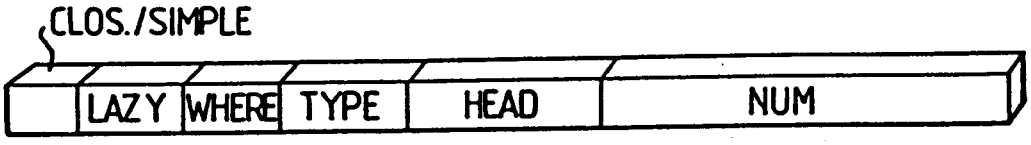
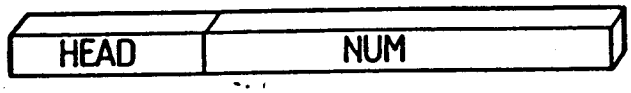


FIG.2B



PILNAS REGISTRAS FIG.2C



RIBOTAS REGISTRAS FIG.2D

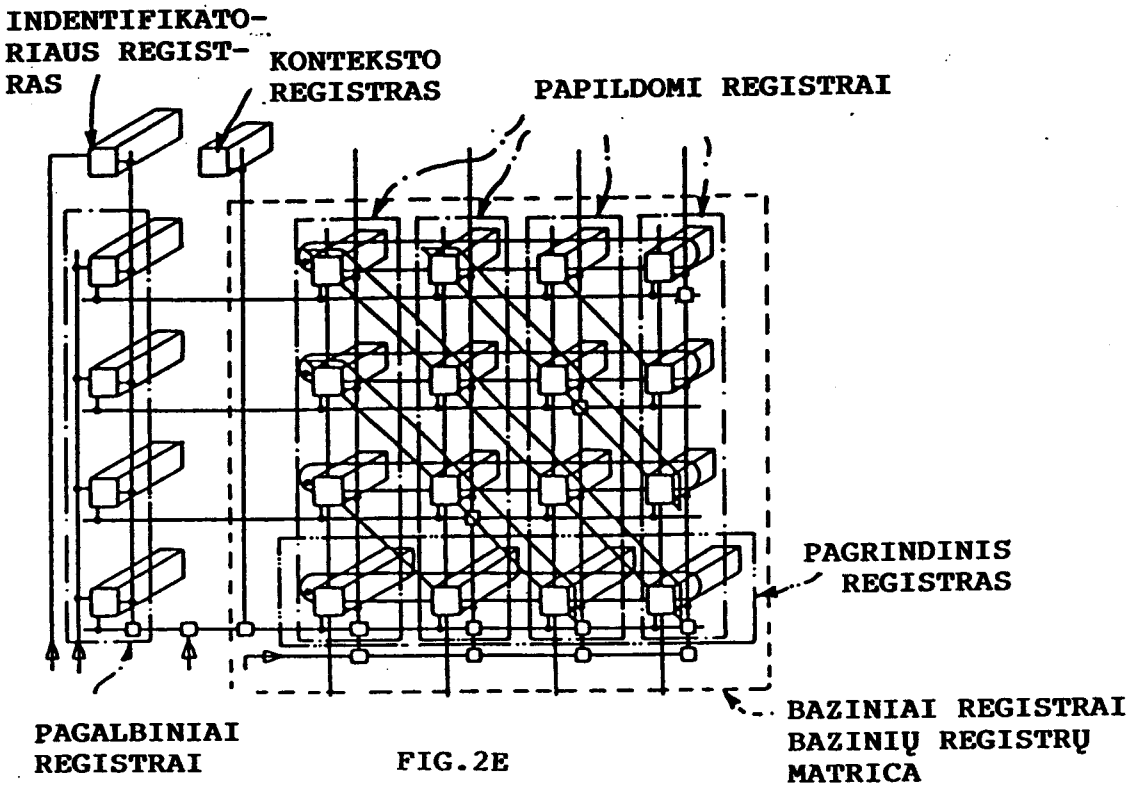


FIG.2E

PAPRASTA REIKŠMĖ ARBA MAŠINOS IDENTIFIKATORIUS

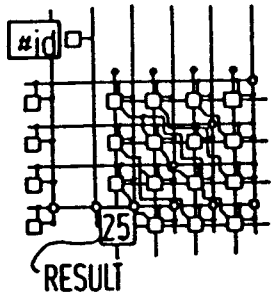


FIG.3A

VIENO LYGIO STRUKTŪRA

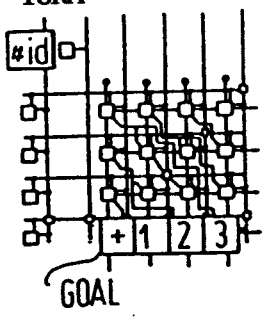


FIG.3B

DVIEJŲ LYGIŲ STRUKTŪRA

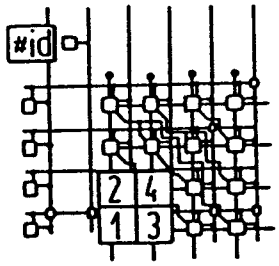


FIG.3C

DVIEJŲ LYGIŲ STRUKTŪRA

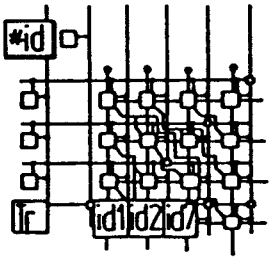


FIG.3D

TRIJŲ LYGIŲ STRUKTŪRA

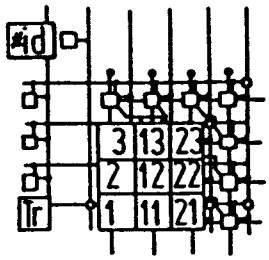


FIG.3E

KONVEJERINIAME REŽIME

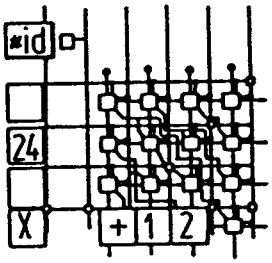


FIG.3F

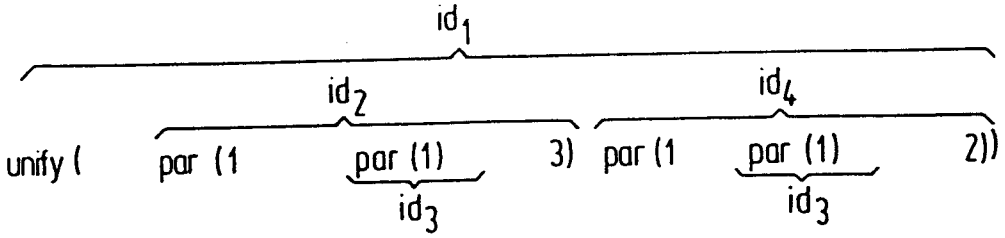


FIG.4A

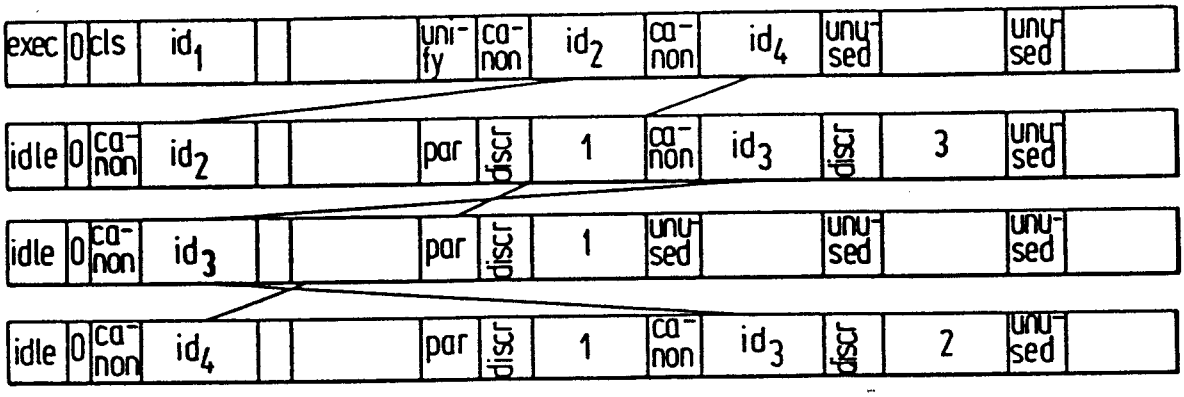


FIG.4B

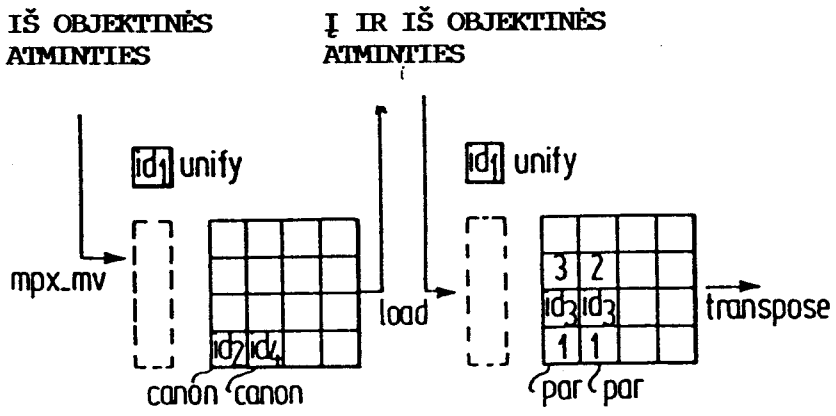


FIG.4C

FIG.4D

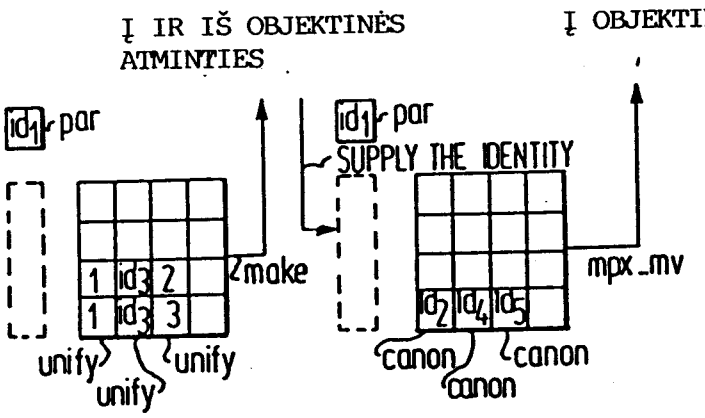


FIG.4E FIG.4F

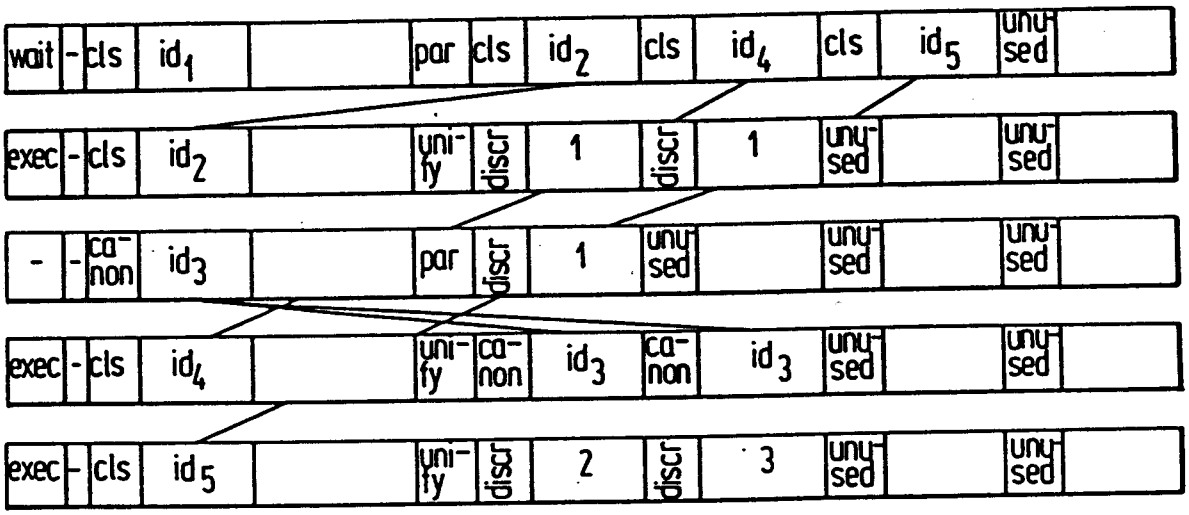


FIG.4G

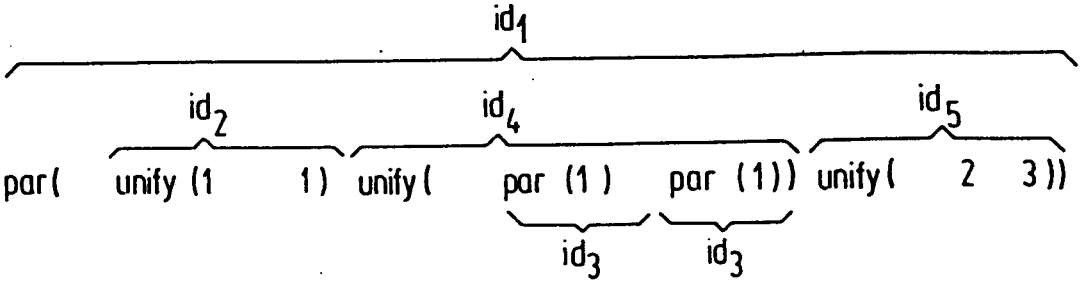


FIG.4H

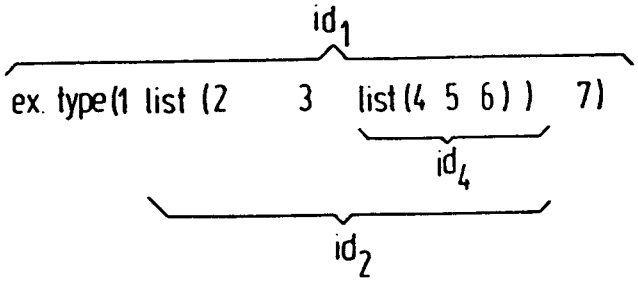


FIG.5A

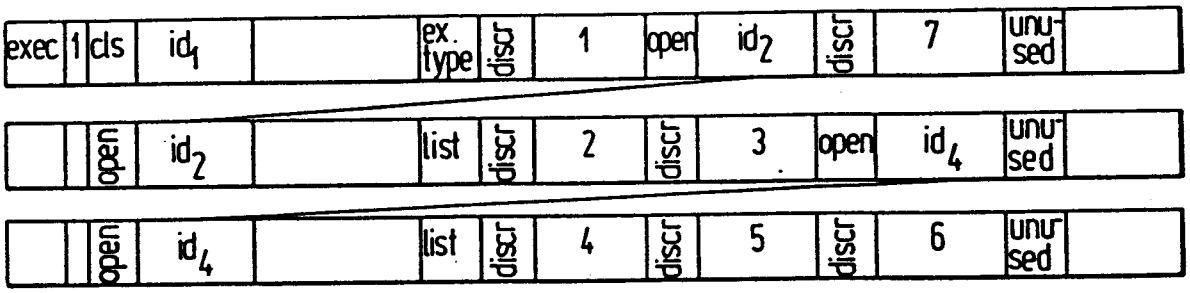


FIG.5B

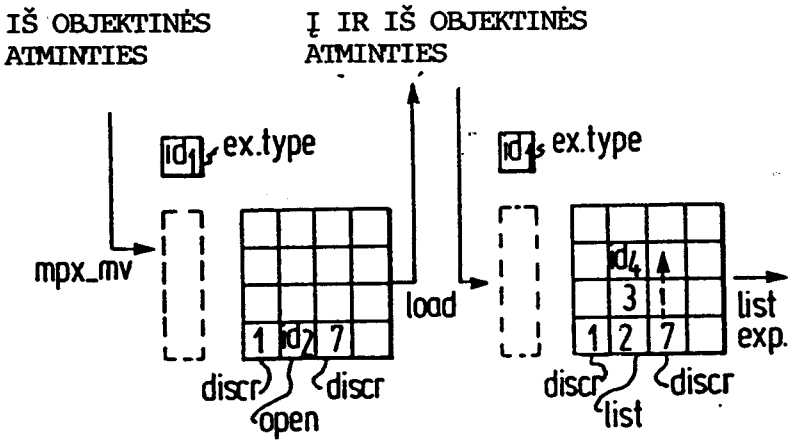


FIG.5C FIG.5D

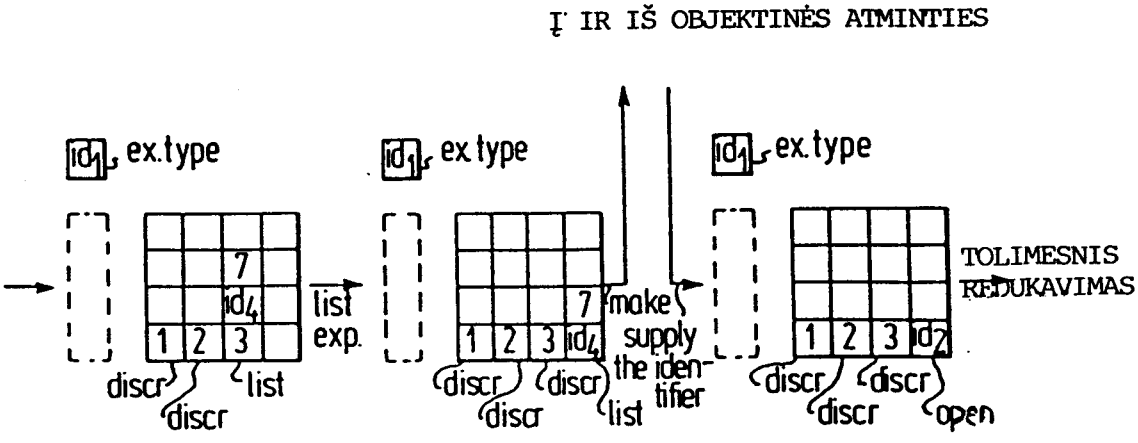


FIG.5E FIG.5F FIG.5G

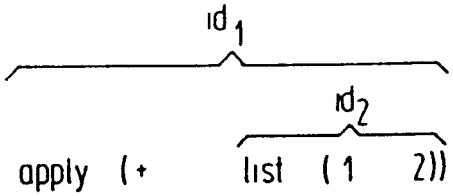


FIG.6A

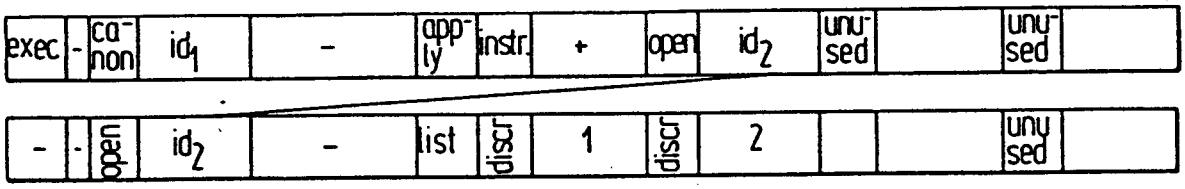


FIG.6B

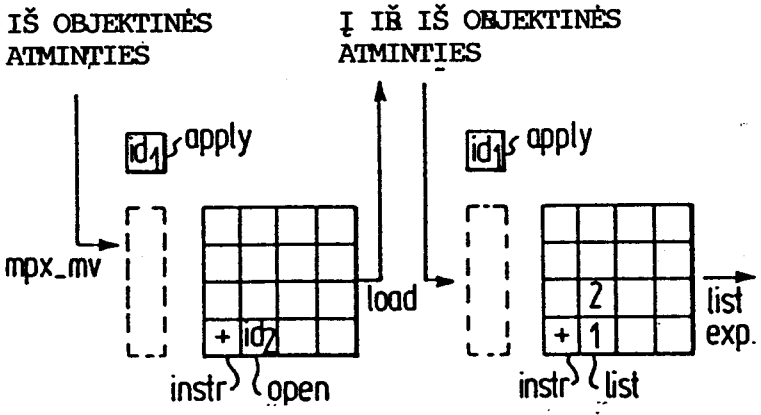


FIG.6C FIG.6D

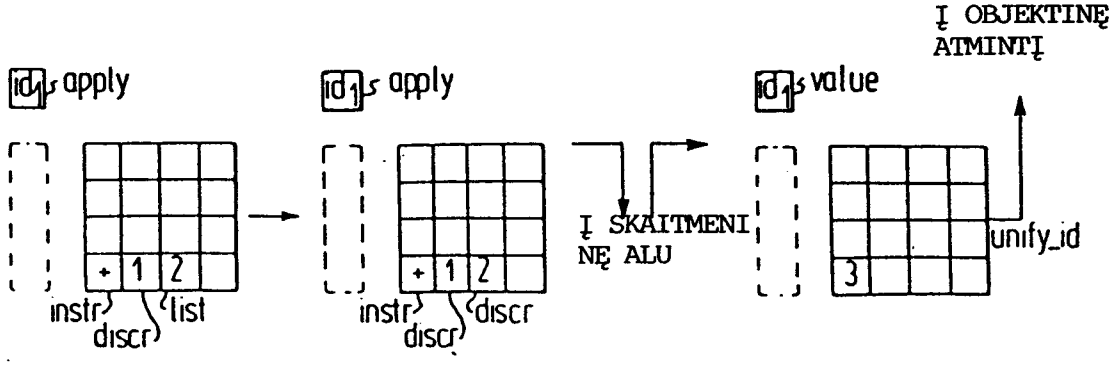
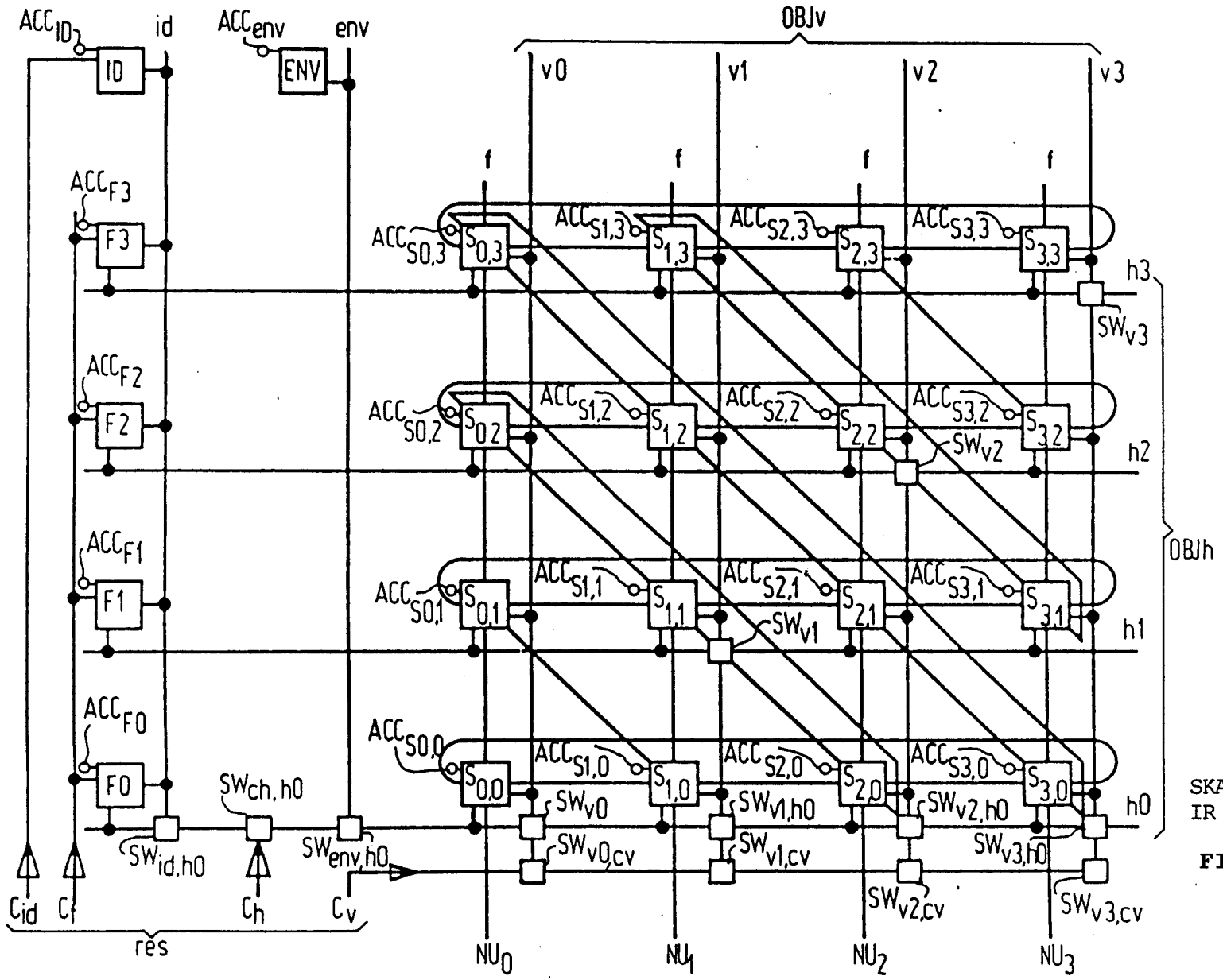


FIG.6E FIG.6F FIG.6G



SKAITMENINĖ
IR VIRŠŪNĖS PLOKŠTUMOS

FIG.7

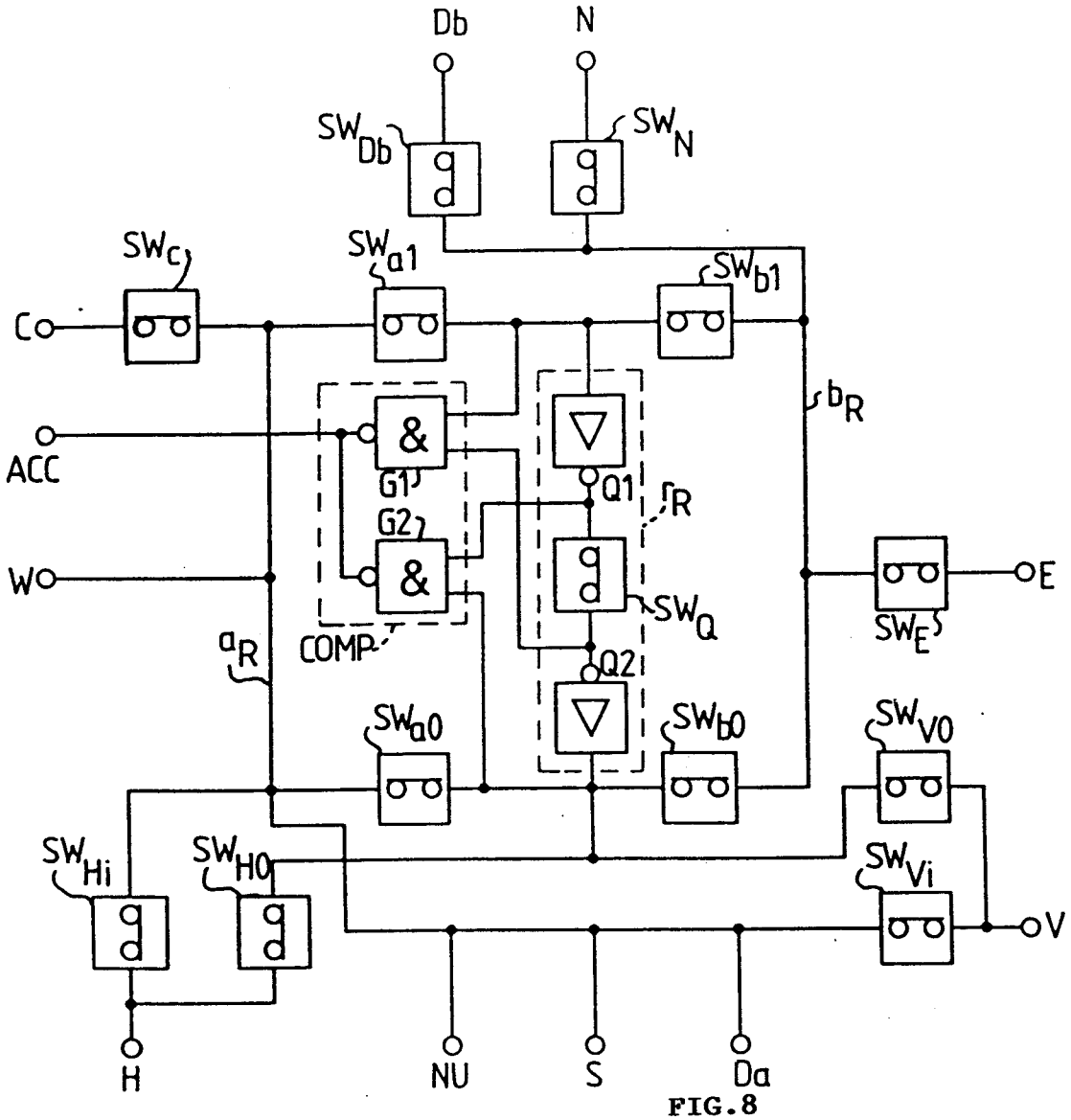


FIG.8

BENDRA REGISTRO LĄSTELĖ (SKAITMENINĖ IR VIRŠŪNĖS PLOKŠTUMOS)

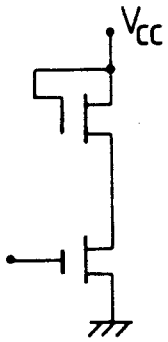


FIG.9A

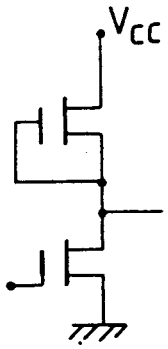


FIG.9B

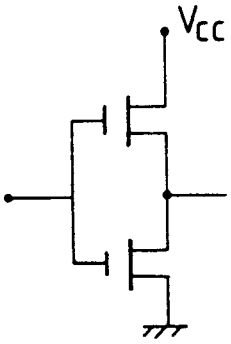


FIG.9C

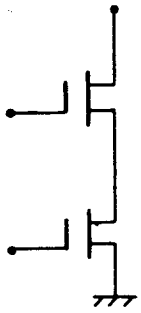


FIG.9D

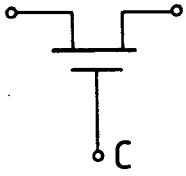


FIG.9E

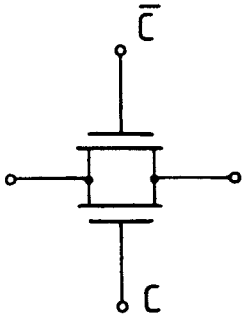
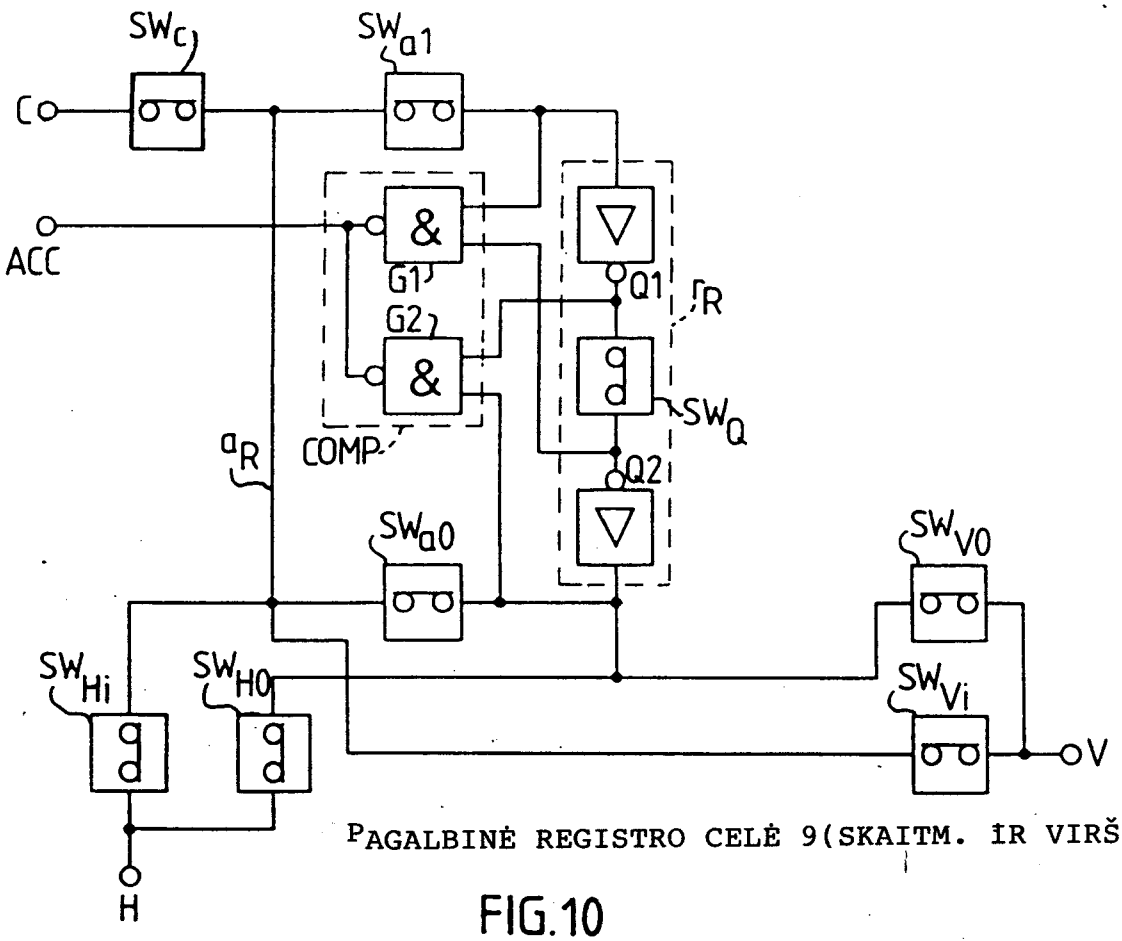


FIG.9F



PAGALBINĖ REGISTRO CELĖ 9(SKAITM. IR VIRŠ.PLOKŠ)

FIG.10

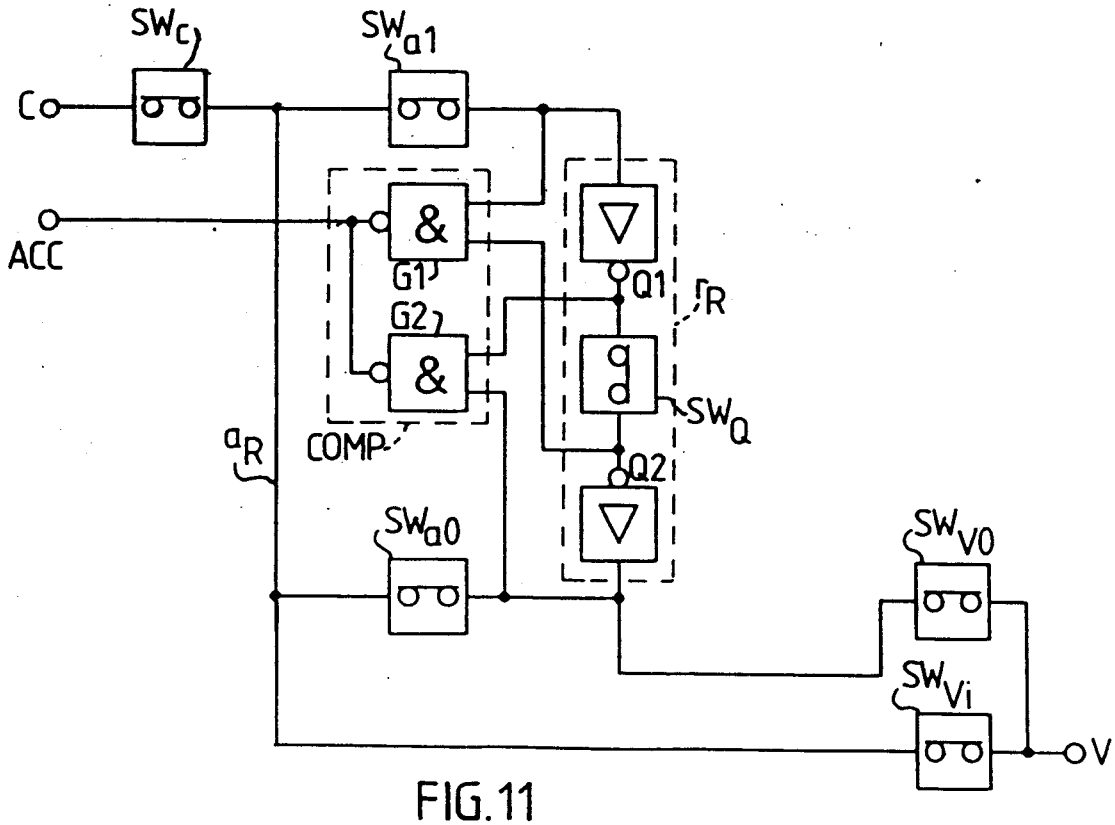


FIG.11

IDENTIFIKATORIAUS REGISTRO CELĖ (SKAITM. IR VIRŠ.PLOKŠ.)

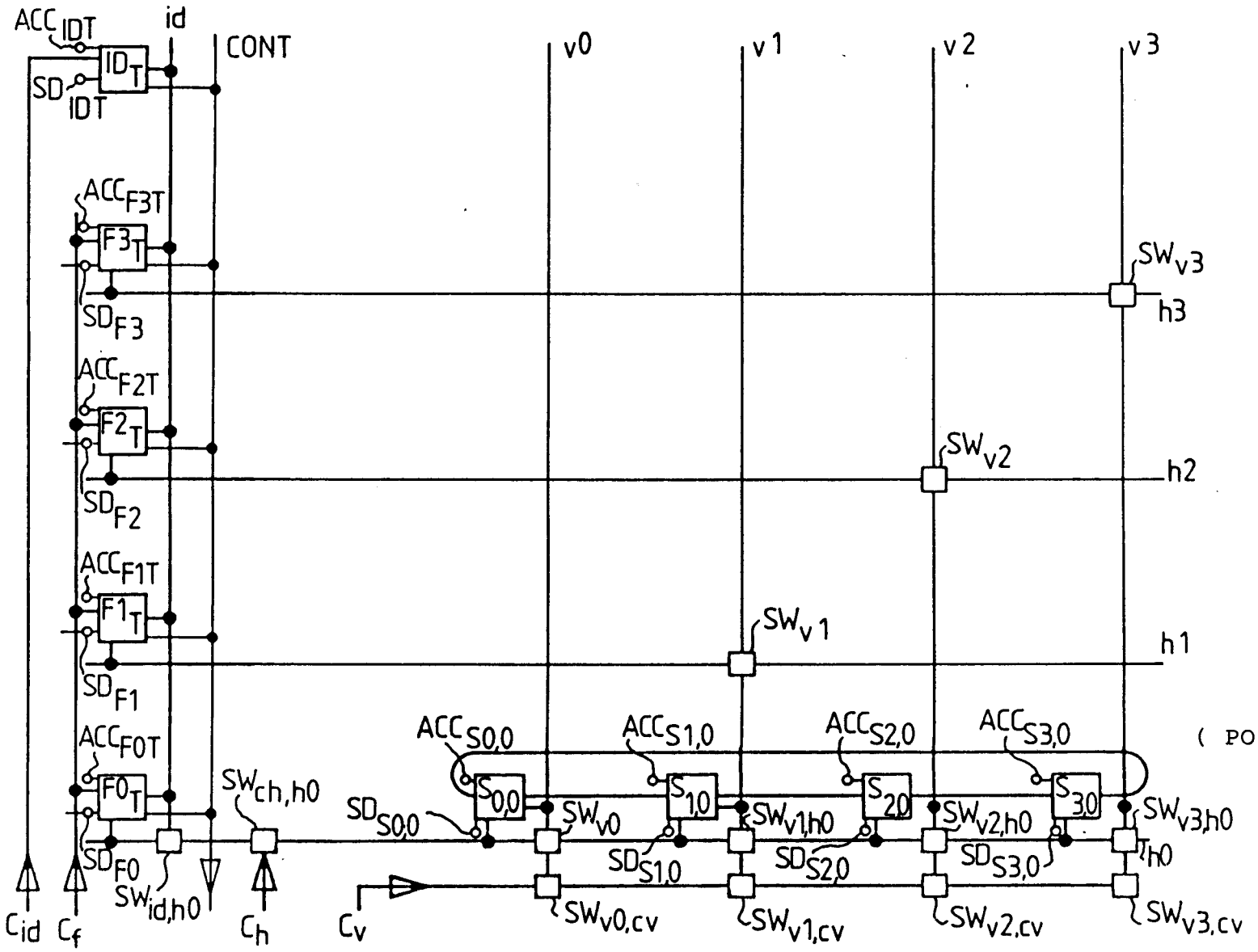


FIG.12

(POŽYMIŲ PLOKŠTĖS)

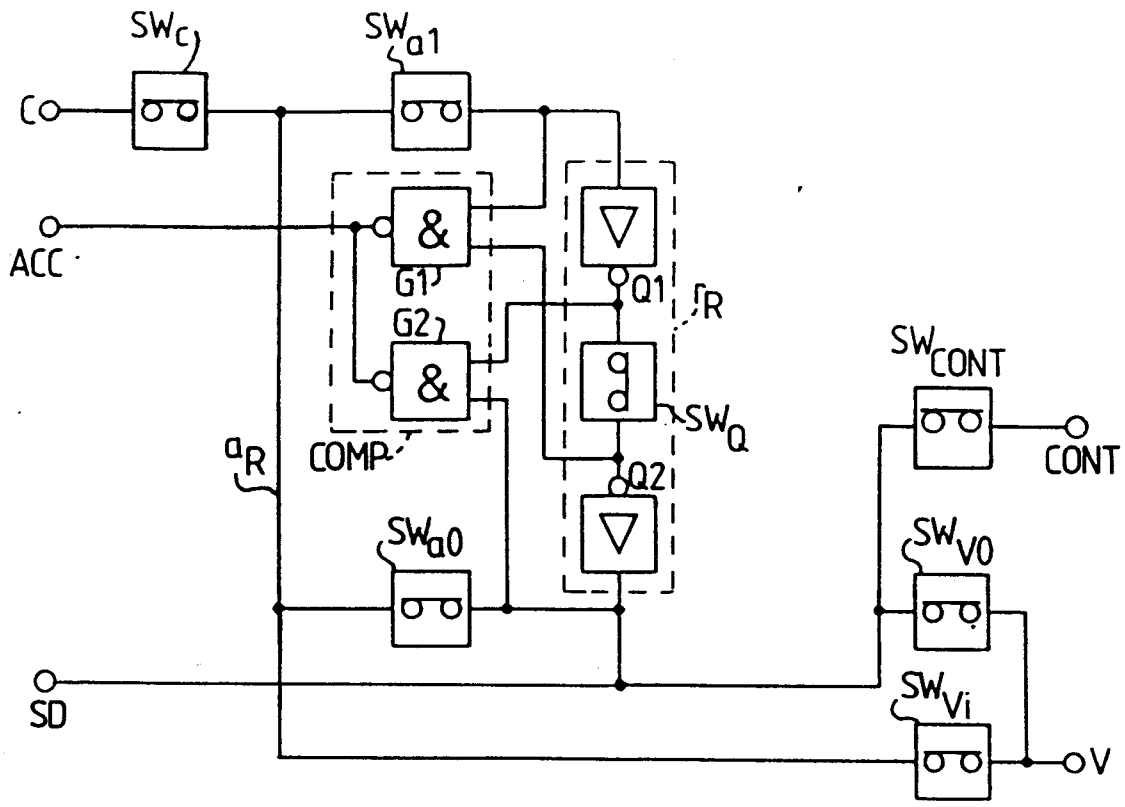


FIG.13

IDENTIFIKATORIAUS REGISTRO CELĖ
(POŽYMIŲ PLOKŠTĖS)

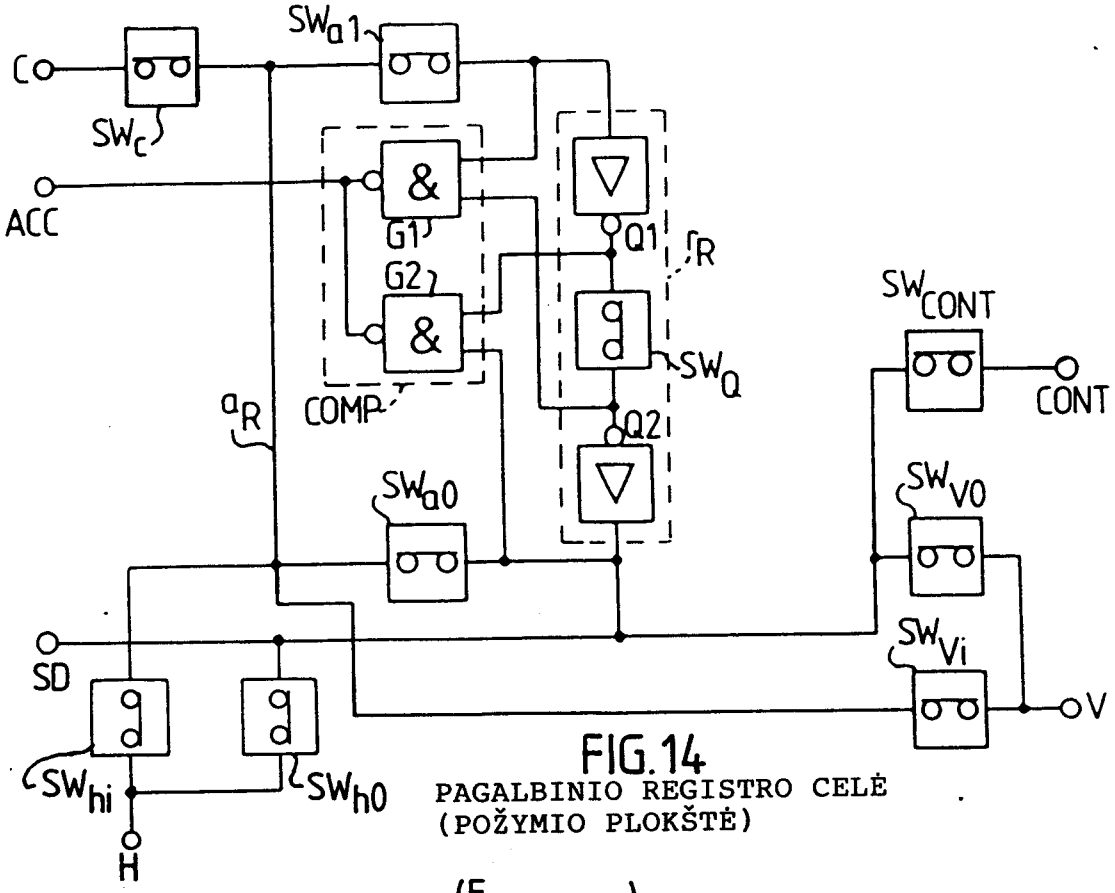


FIG. 14
PAGALBINIO REGISTRO CELĖ
(POŽYMIO PLOKŠTĖ)

$(F_{YT,Y:0..3})$

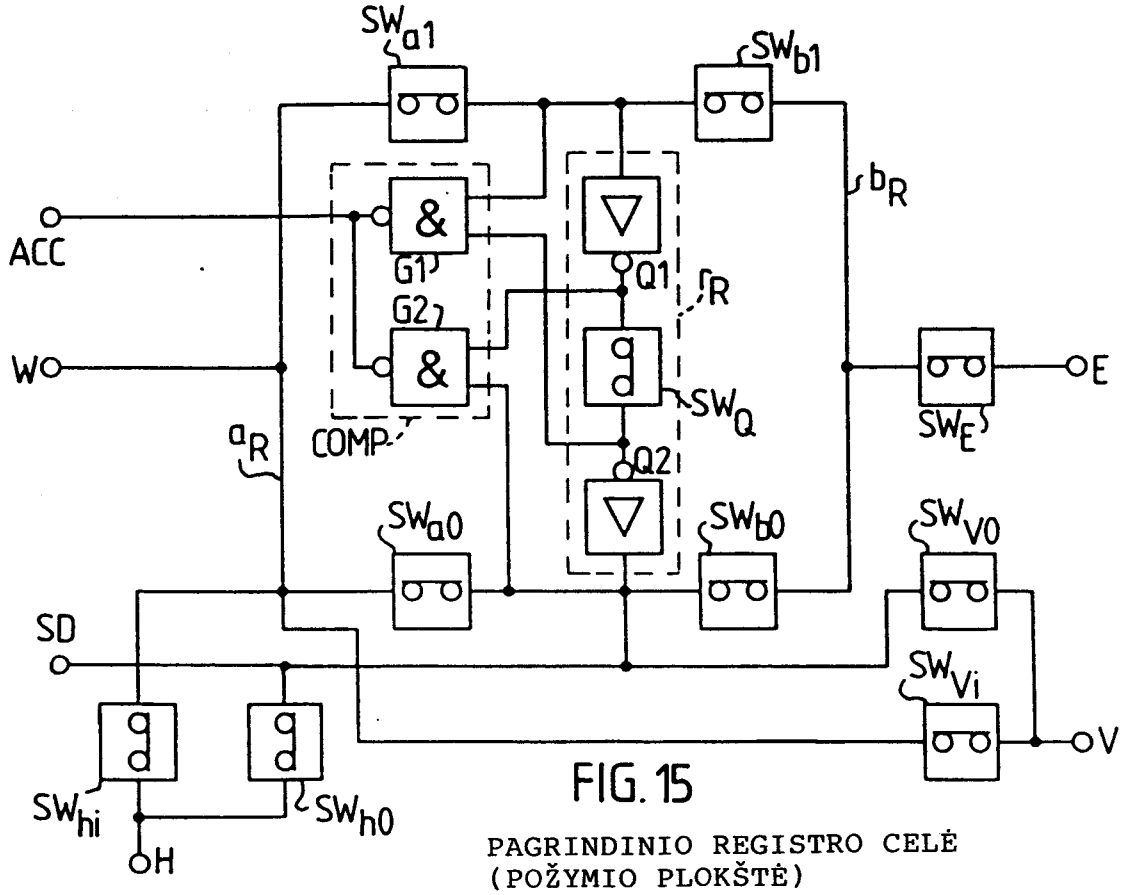


FIG. 15
PAGRINDINIO REGISTRO CELĖ
(POŽYMIO PLOKŠTĖ)
$(S_{X,0X:0...3})$

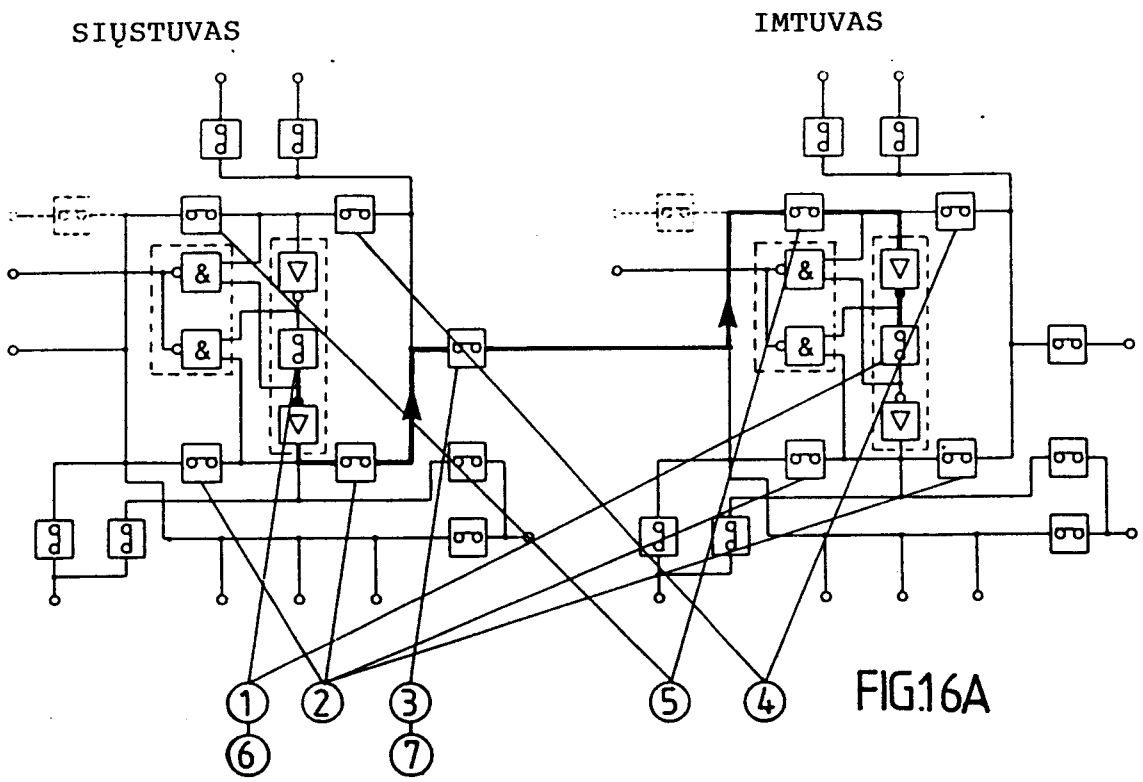


FIG.16A

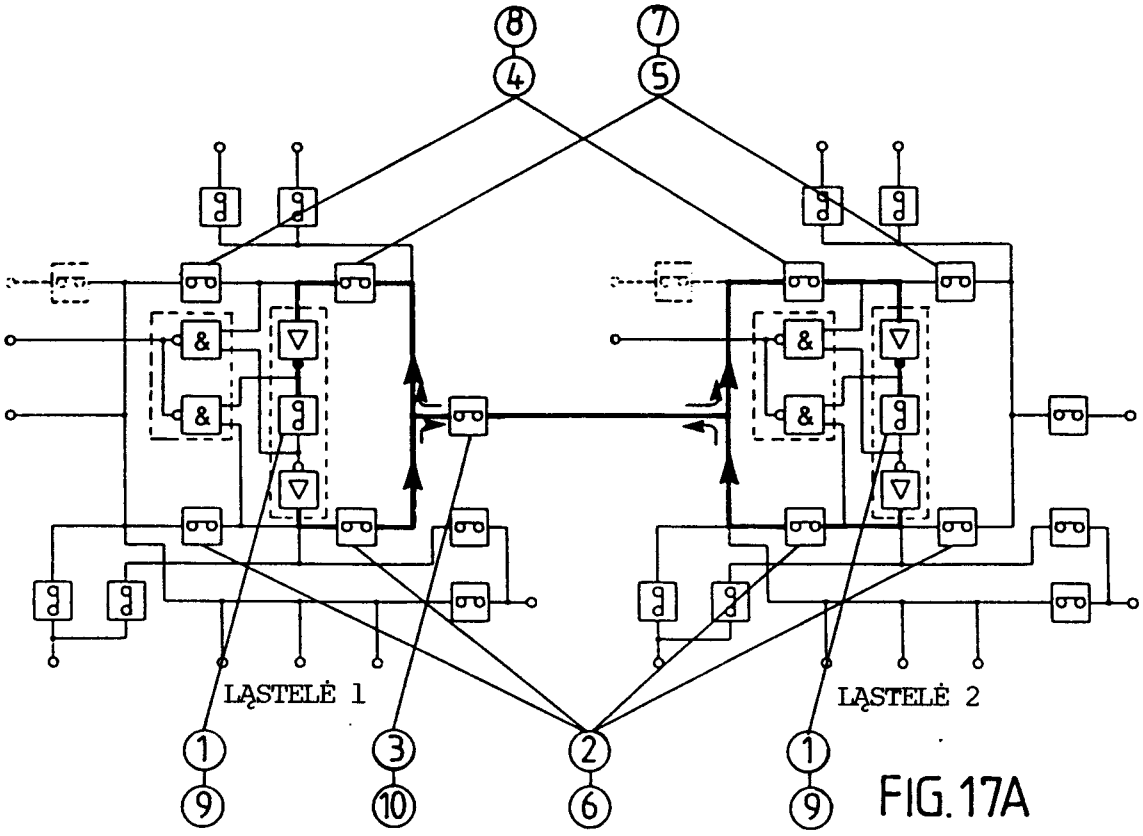


FIG.17A

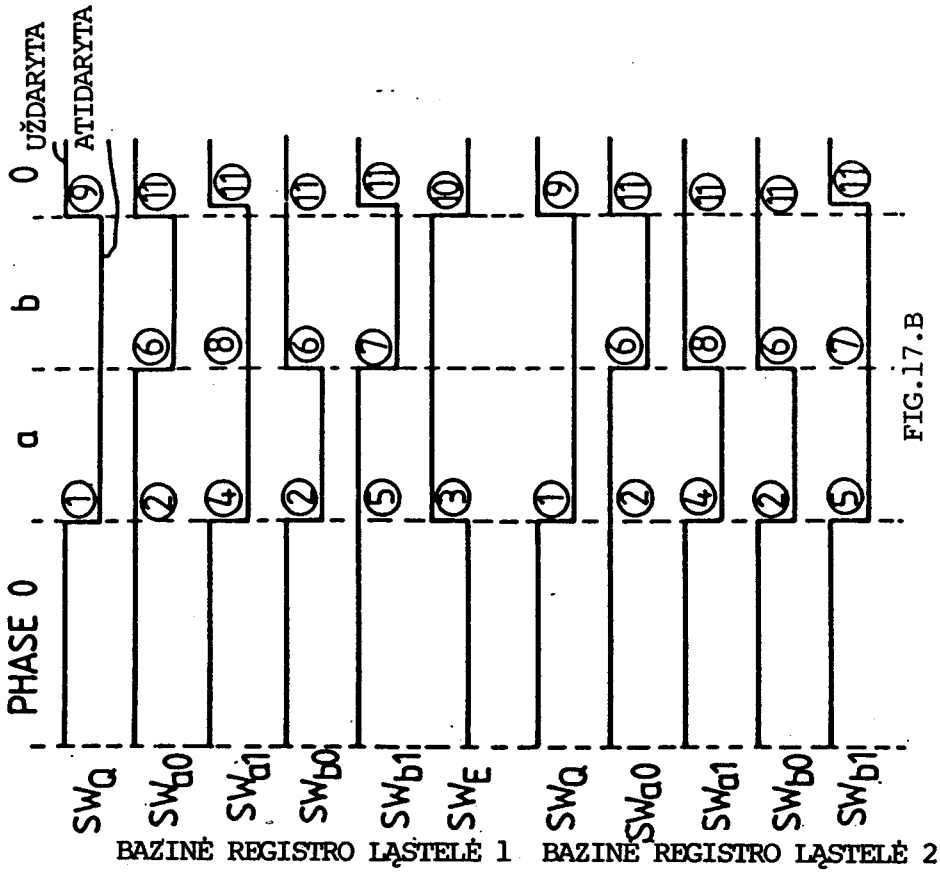


FIG.17.B

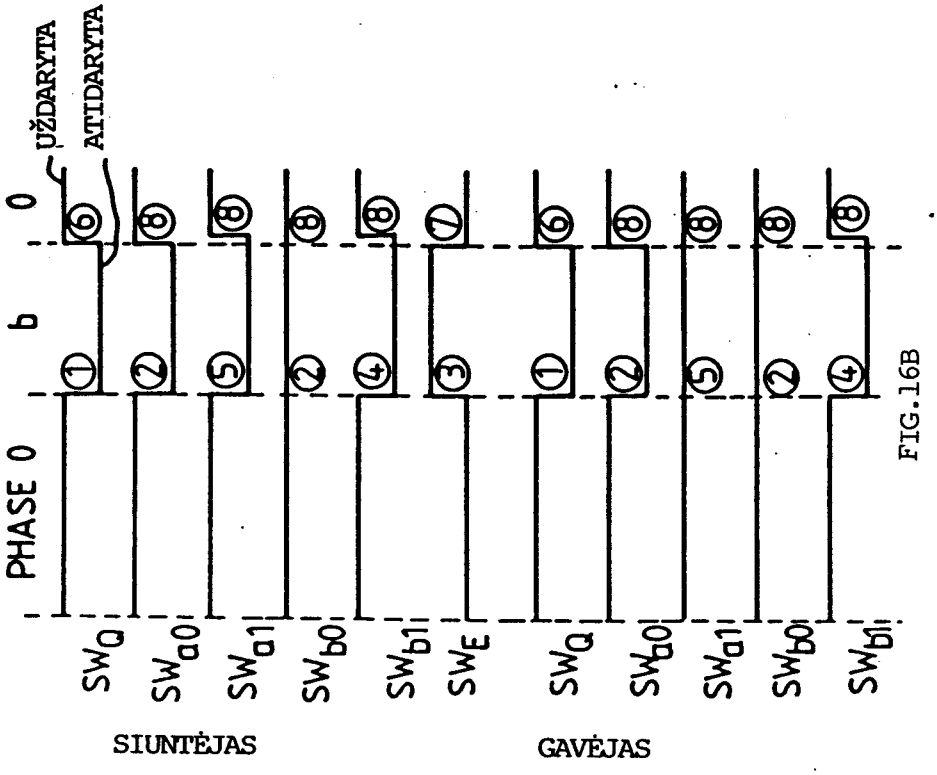


FIG.16B

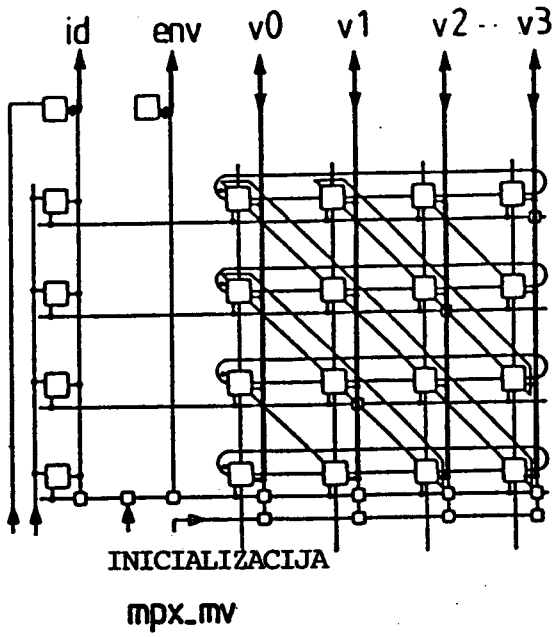


FIG.18

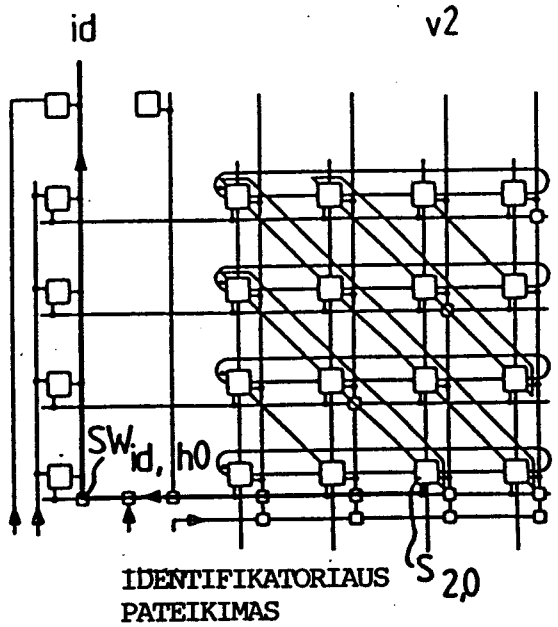


FIG.19

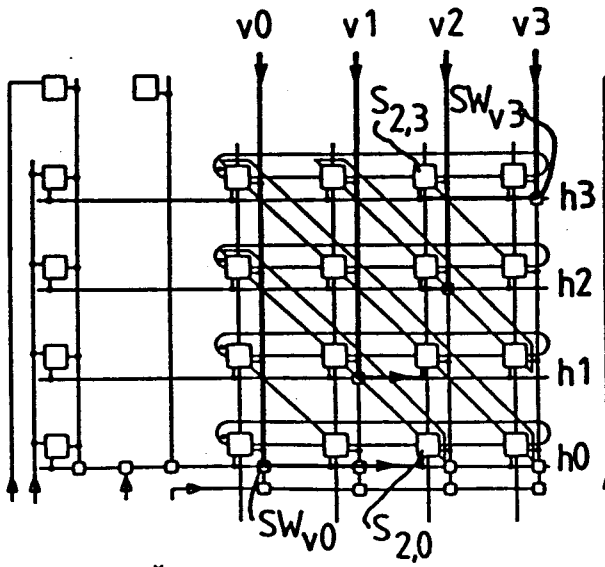


FIG.20

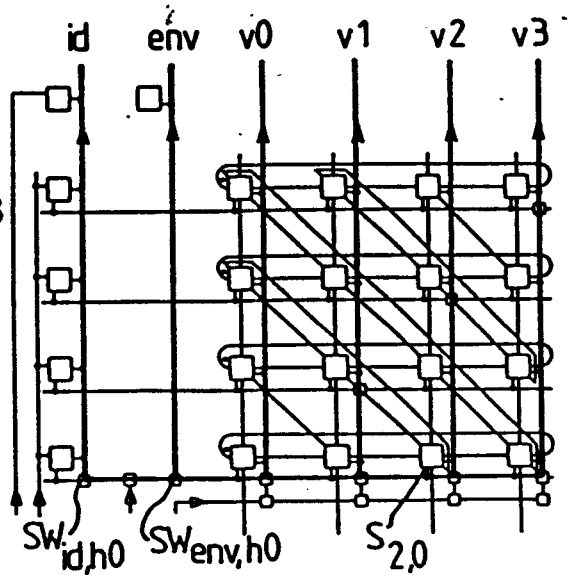


FIG.21

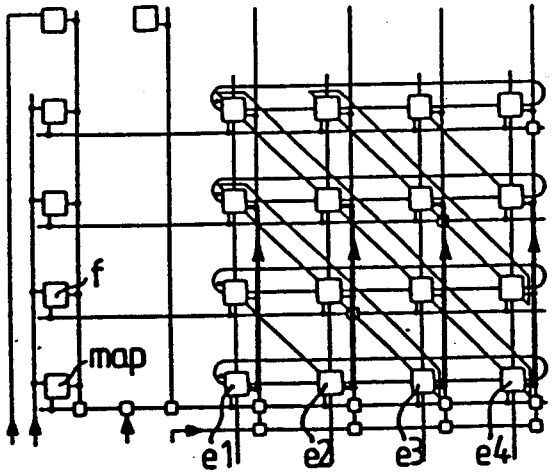


FIG.22A

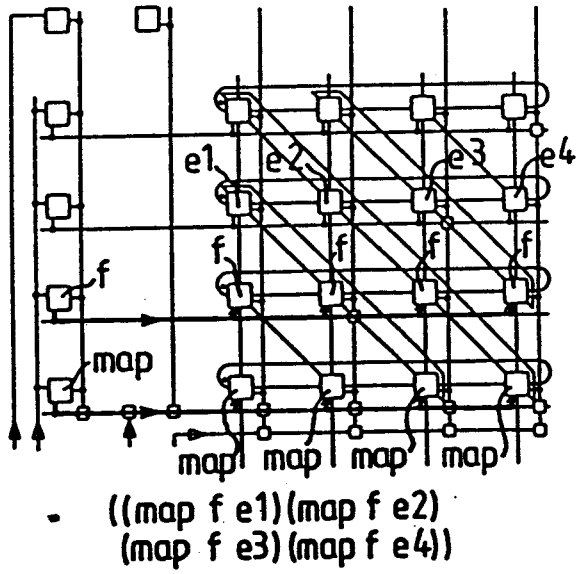


FIG.22B

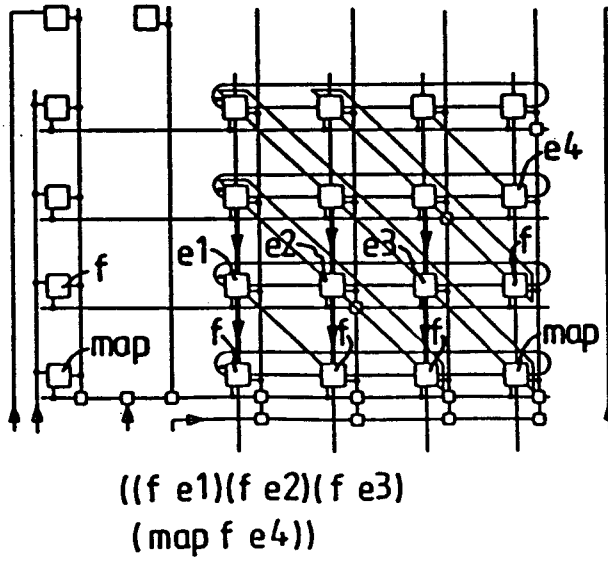
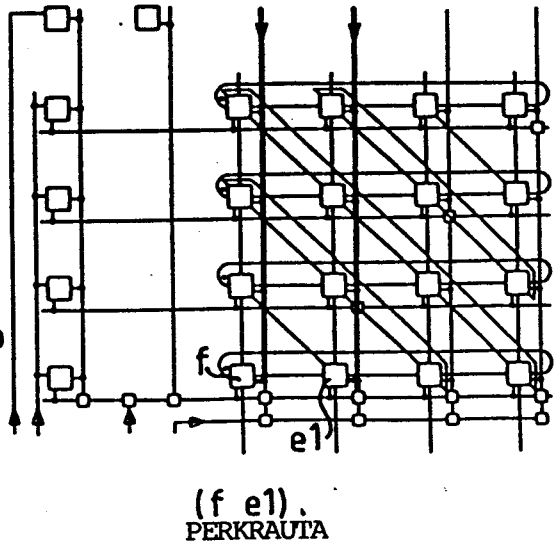
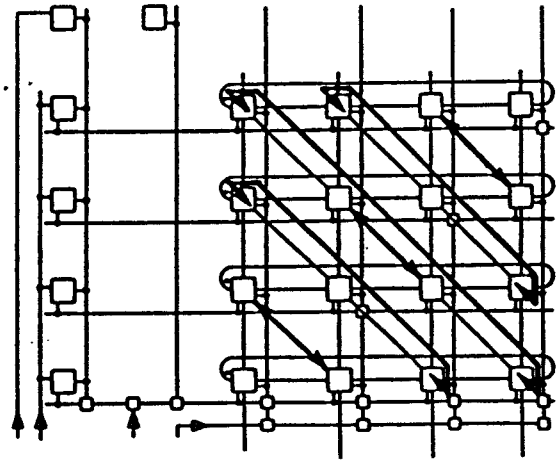


FIG.22C



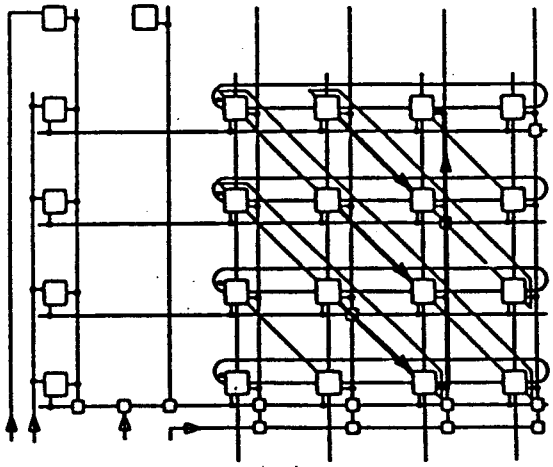
PERKRAUTA

FIG.22D



TRANSPONAVIMAS

FIG.23



IŠPLEČIAMAS sąrašas

FIG.24

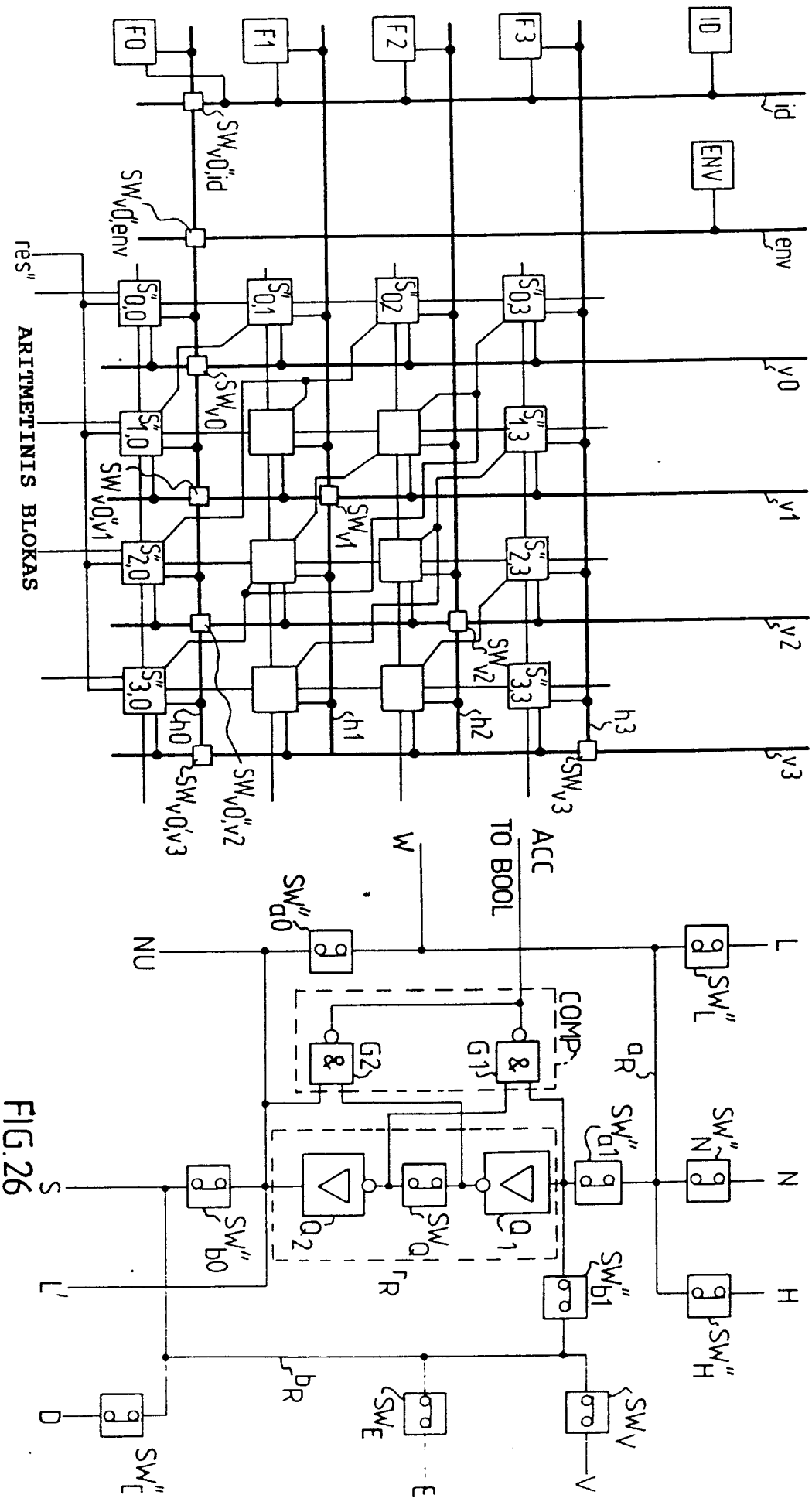

ARITMETINIS BLOKAS
F0
F1
F2
F3
ID
ENV
id
env
res"
v0
v1
v2
v3
h0
h1
h2
h3
W
ACC
TO BOOL
NU
COMP
G1
G2
Q1
Q2
L
N
H
S
L'
D
E
V
R
bR
aR

FIG. 25

FIG. 26