VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Virtualios ir realios mašinos modelis

Virtual and real machine model

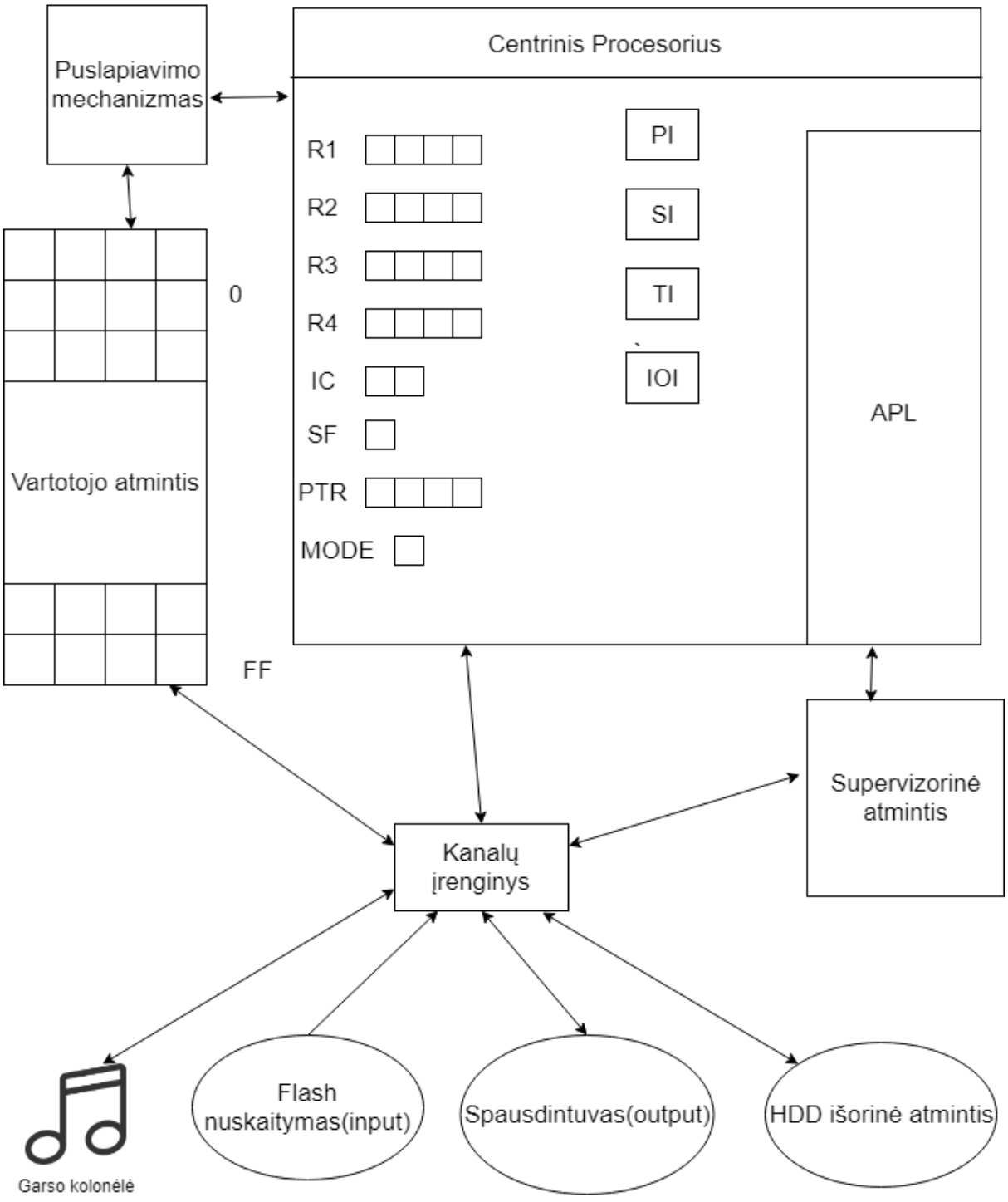
Atliko: 2 kurso 3 grupės studentai
Matas Savickis (parašas)
Justas Tvarijonas (parašas)
Greta Pyrantaitė (parašas)

Darbo vadovas: Mantas Grubliauskas Lekt. (parašas)

Vilnius – 2018

TURINYS

1. Reali mašina



1 pav. Reali mašina

1.1. Techninės įrangos elementai sudarantys realią mašiną

- Centrinis procesorius
- Vartotojo atmintis
- Supervizorinė atmintis
- Išorinė atmintis
- Duomenų perdavimo kanalas
- Įvedimo įrenginys - flash atmintinis
- Išvedimo įrenginys - spausdintuvas
- Papildomas išvedimo įrenginys - garso kolonėlė
- Puslapiavimo mechanizmas

1.2. Centrinis procesorius

Centrinio procesoriaus paskirtis yra skaityti komandas iš atminties ir jas interpretuoti. Procesorius gali dirbti dviem rėžimais - supervizoriaus ir vartotojo. Supervizoriaus rėžime komandos yra apdorojamos aukšto lygio procesoriaus. Komandos vykdomos supervizoriaus režime yra skirtos operacinės sistemos funkcionavimui palaikyti. Procesorius persijungia į šį rėžimą pertaukimais arba sisteminiais kreipiniais.

1.3. Procesoriaus registrai

- R1, R2, R3, R4 - 4 baitų bendros paskirties registrai
- IC - 2 baitų komandų skaitiklis
- SF - 1 baito požymių registras CF ZF SF IF OF XXX - carry flag, zero flag, sign flag, interupt flag, overflow flag
- PTR - 4 baitų puslapių lentelės registras(naudojamo puslapių lentelės adresas)
- MODE - registras, kurio reikšmė nusako procesoriaus darbo režimą(supervizorinis, vartotojo)
- PI - programinių pertraukimų registras
- SI - supervizorinių pertraukimų registras
- TI - taimerio registras
- IOI - 2 baitų įvedimo ir išvedimo pertraukimų registras

1.4. Atmintys

Pagrindinę realios mašinos atmintį dalinasi vartotojo ir supervizorinė atmintis. Kaskart sukuriant naują virtualią mašiną panaudojama dalis realios mašinos atminties. Atminties dydis 256 blokų po 16 žodžių. Žodžio ilgis 4 baitai. Supervizorinę atmintį naudoja aukšto lygio procesorius, tačiau šios atminties savo rašomoje programoje nenaudosim. Taip pat yra išorinė atmintis kietojo disko pavidalu.

1.5. I/O

Komandų įvedimui naudojamas „flash atmintinių" nuskaitymo įrenginys. Išvedimui naudojamas spausdintuvas. Abu šie įrenginiai tėra modeliai ir didelio skirtumo nedarys.

1.6. Puslapiavimo mechanizmas

Puslapiavimo mechanizmas yra metodas skirtas virtualios atminties adreso parodymui į realios atminties adresą. Kiekvienai virtualiai mašinai skiriama 16 atminties blokų. Sukuriant virtualią mašiną sukuriama puslapių lentelė. Naudojamas PTR(4 baitų a0,a1,a2,a3) registras kuriame laikomas puslapių lentelės adresas. Baitai a0 ir a1 yra nenaydojami, o a2 * 16 + a3 žymi puslapių lentelės adresą. Dabar galime pateikti formulę, kuri virtualiam adresui x1x2 gražina realų adresą: Realus adresas = 16*[16*(16 * a2 + a3) + x1] + x2.

1.7. Garso kolonėlė

Vartotojui įvedus atitinkamą komandą kompiuterio garso kolonėlė supypsi x1 kartų.

1.8. Pertraukimų mechanizmas

Vykdant komandą virtualioje mašinos gali kilti pertraukimai, tuo atvėju nustatoma atitinkamu registrų reikšė ir nutraukiamas komandos vykdymas. Kviečiant test() metodą, patikrinama ar įvyko pertraukimas, jam įvykus nutraukiamas virtualios mašinos darbas ir kviečiamas metodas Test(), kuris apdorojo pertraukimą. Apdorojus pertraukimą sprendžiama ar grąžinti darbą atgal virtualiai mašinai. Galimi pertraukimai:

- Operacijos GD, PD, PY ir HALT iššauks supervizorinius pertraukimus. SI = 1 - komanda GD, SI = 2 - komanda PD, SI = 3 - komanda PY, SI = 4 – komanda HALT.
- Programiniai pertraukimai:PI = 1 – neteisingas adresas, PI = 2 – neteisingas operacijos kodas, PI = 3 – neteisingas priskyrimas, PI = 4 – perpildymas (overflow).
- Esant TI = 0 bus fiksuojamas taimerio pertraukimas.

1.9. Timerio mechanizmas

Vygdant virtualią mašiną kas komandą mažinama timerio registro reikšmė, jai pasiekus 0 įvyksta pertraukimas, o jį apdorojus timerio reikšmė nustatoma 10.

1.10. Duomenų perdavimo kanalai

Skirti I/O ir atminties valdymui.Modelyje yra vienas kanalas kuriam galima padaryti skirtingas paskirtis: rašyti, skaityti arba abu. Duomenys eina per kanalą ir pats kanalo įrenginys juos nukreipia į reikiamą kryptį ir pasako kiek duomenų gali pasiimti o kiek reikia nukirpti. Kanalų įrenginio registrai:

SB: Žodžio, nuo kurio kopijuosime numeris.
DB: Žodžio, į kurį pradėsime rašyti numeris
ST: Objekto, iš kurio kopijuosime, numeris
1. Vartotojo atmintis;
2. Supervizorinė atmintis;
3. Išorinė atmintis;
4. Įvedimo srautas;

DT: Objekto, į kurį kopijuosime, numeris
1. Vartotojo atmintis;
2. Supervizorinė atmintis;
3. Išorinė atmintis;
4. Išvedimo srautas;

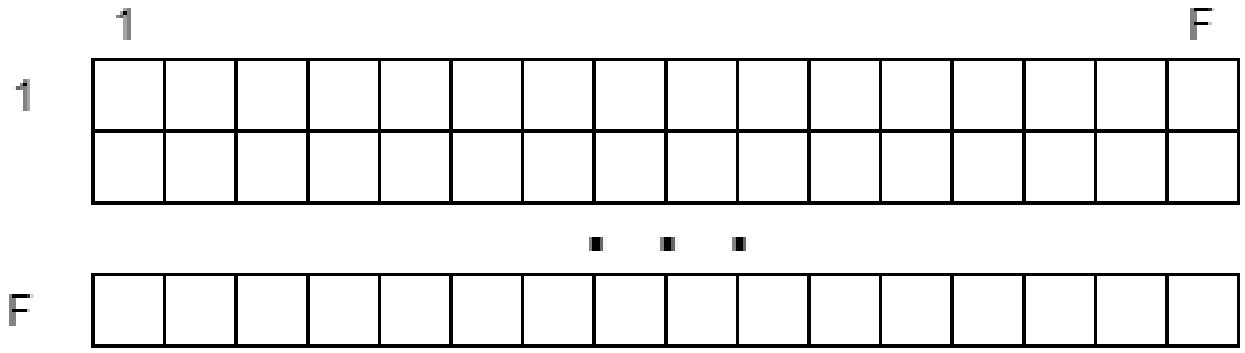
2. Virtuali mašina

2.1. Funkcija

Virtuali mašina yra skirta paslepti realios mašinos sudėtingumą ir suteikti vartotojui instrukcijų sąrašą su kuriuo jis galėtų dirbti. Todėl operacinės sistemos paskirtis ir yra paslėpti realią mašiną ir duoti mums virtualią. Virtuali mašina taip pat suteikia darbų pasidalijimą kurio dėka galima paleisti kelias virtualias mašinas ir tokiu būdu ant kiekvienos iš jų atlikti skirtingas užduotis.

2.2. Modeliuojamos virtualios mašinos loginių komponentų aprašymas

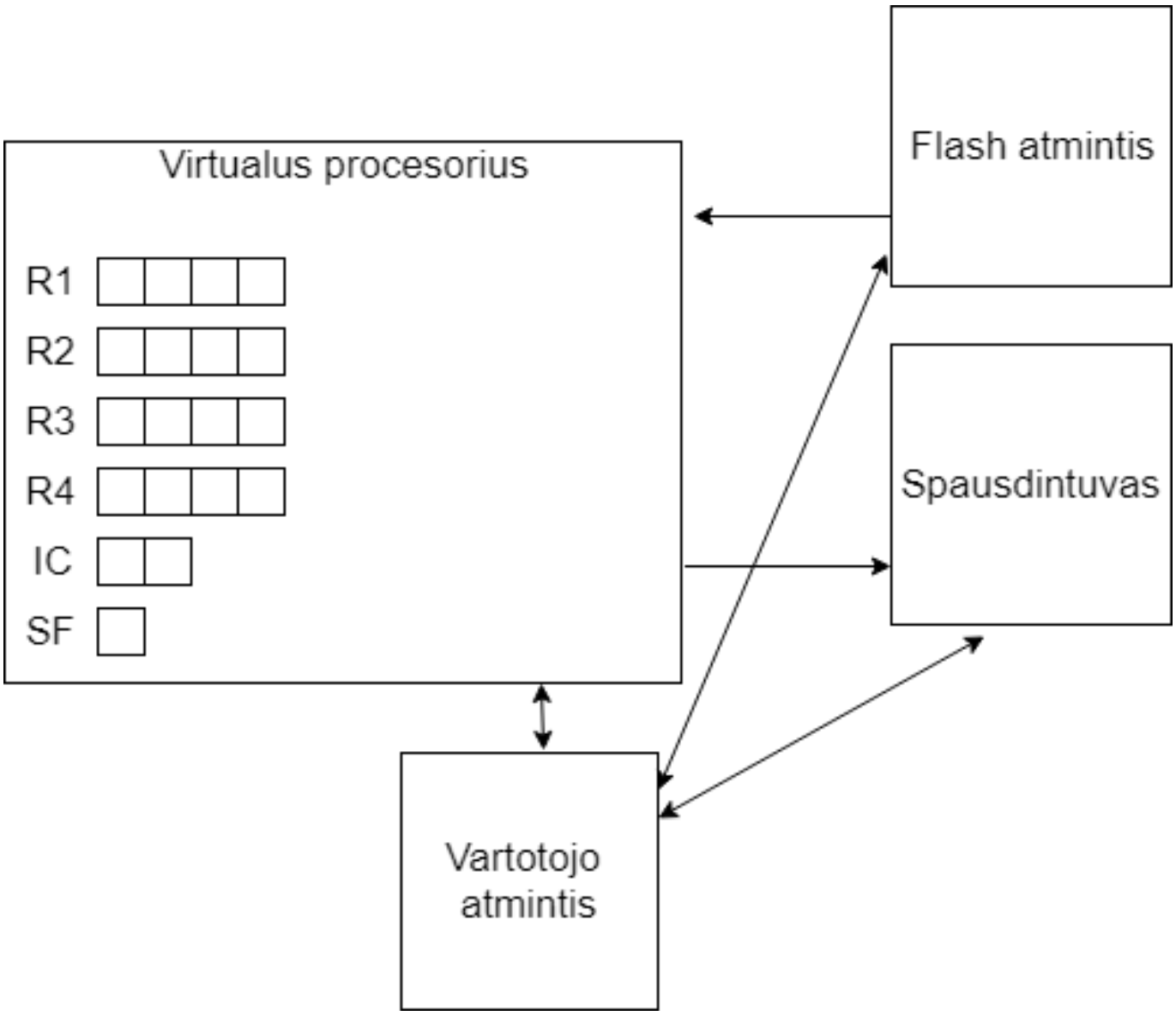
2.2.1. Atmintis



2 pav. Virtualios mašinos atmintis

Virtualios mašinos atmintis susideda iš 16 blokų po 16 žodžių iš viso 256 žodžiai po 4 baitus arba 32 bitus. Naudosime registrą PTR(4 baitai a0 a1 a2 a3) kuriame laikomas adresas į puslapių lentelę. Baitai a0 ir a1 nenaudojami, o a2 * 16 + a3 žymi puslapių lentelės adresą. Dabar galime pateikti formulę, kuri virtualiam adresui x1x2 gražina realų adresą: Realus adresas = 16*[16*(16 * a2 + a3) + x1] + x2

2.2.2. Procesorius



3 pav. Virtualios mašinos procesorius

Registrai:

- R1...R4 - Bendros paskirties registrai
- IC - komandos skaitliukas, rodo sekančios komandos adresą
- SF - status flagas rodantis programos būseną

2.2.3. Virtualios mašinos komandų sistema

Kiekvieną virtualios mašinos komandą sudaro 4B, tačiau priklausomai nuo komandos ne visi baitai turi būti užimti – jie gali būti ir tušti. Komandos:

1. Duomenų persiuntimui iš atminties į registrus ir atvirkščiai:
   (a) LR – Load Register – iš atminties baito x1x2 persiunčia į registrą R: LR x1x2 =) R:=[x1x2];

(b) SR – Save Register – iš registro R persiunčia į atminties baitą x1x2: SR x1x2 =) [x1x2]:=R;

2. Duomenų sukeitimui tarp registrų:

   (a) RR – sukeičia registro R ir R2 reikšmes: RR =) R:=R+R2, R2=R-R2, R=R-R2;

3. Aritmetinės komandos:

   (a) AD – suma – prie esamos registro R reikšmės prideda reikšmę esančią x1x2 atminties baite, rezultatas patalpinamas registre R: AD x1x2 =) r1:=r1+[x1x2];

   (b) SB – atimtis – iš esamos registro R1 reikšmės atimama reikšmė esanti x1x2 atminties baite, rezultatas patalpinamas registre R: SB x1x2 =) r1:=r1-[x1x2];

   (c) CR – palyginimas – esamą registro R reikšmė yra lyginama su reikšme esančią x1x2 CR x1x2 =) if r1>[x1x2] then cf:=0, zf:=0; if r1=[x1x2] then zf:=1; if r1<[x1x2] then cf:=1;

   (d) MU x1x2 – daugyba, r1 =r1* [x1x2].

   (e) DI x1x2 – dalyba, r1=r1/[x1x2] , r2=r1%[x1x2].

4. Valdymo perdavimo:

   (a) PY x1 - Garso kolonėlė pypsi x1 sekunčių.

   (b) JU – besąlyginio valdymo perdavimas – valdymas perduodamas adresu 16*x1+x2:JU x1x2 =) IC:=16*x1+x2;

   (c) JG – sąlyginio valdymo perdavimas (jeigu daugiau) – valdymas perduodamas jeigu SF bitai ZF == 0 ir SF == OF. Valdymas perduodamas adresu 16*x1+x2:JG x1x2 =) If ZF == 0 AND SF == OF then IC:= 16*x1+x2;

   (d) JE – sąlyginio valdymo perdavimas (jeigu lygu) – valdymas perduodamas jeigu SF bitas ZF == 1.Valdymas perduodamas adresu 16*x1+x2:JE x1x2 =) If ZF == 1 then IC:= 16*x1+x2;

   (e) JL – sąlyginio valdymo perdavimas (jeigu mažiau) – valdymas perduodamas jeigu SF bitai SF ir OF nelygūs, valdymas perduodamas adresu 16*x1+x2:JL x1x2 =) If SF != OF then IC:= 16*x1+x2;

5. Programos pabaigos:

   (a) HALT – programos pabaigos komanda.

6. Įvedimo/Išvedimo:

   (a) GD – įvedimas – iš įvedimo srauto paima 4 žodžų srautą ir jį įveda į atmintį pradedant atminties baitu 16*x1+x2: GD x1x2

   (b) PD – išvedimas – iš atminties, pradedant atminties baitu 16*x1+x2 paima 4 žodžių srautą ir jį išveda į ekraną: PD x1x2

7. Loginės:

   (a) AND – r1 := r1 and r2

(b) XOR – r1 := r1 xor r2

(c) OR - r1 := r1 or r2.

(d) NOT - r1 := not r1.

2.3. Virtualios mašinos bendravimo su įvedimo/išvedimo įrenginiais mechanizmo aprašymas.

VM duomenis skaito iš flash atminties (realizuotos failu kietajame diske), o rezultatą išveda spausdintuvas. Įvedimą/išvedimą kontroliuoja kanalų įrenginys.

2.4. Virtualios mašinos interpretuojamojo ar kompiliuojamo vykdomojo failo išeities teksto formatas.

VM modelio įvedimo įrenginiui pateikiamas programos failas turi būti tokios struktūros:
DATASEG
.
.
.
CODESEG
.
.
.
HALT
Atmintis yra išdėstyta nuosekliai: 128 žodžiai skirti DATASEG (nuo 0 iki 127) ir 128 žodžiai CODESEG (nuo 128 iki 255).
Programa apskaičiuoja reiškinio „100 + 20 – 80“ reikšmę, bei ją išveda į ekraną.
| DATA
000 | 100
001 | 20
002 | 80
003 | Rezu
004 | ltat
008 | as y
009 | ra:
| CODE
081 | LR 00
082 | AD 01
083 | SB 02
084 | PD 03
085 | SR 10
086 | PD 10
087 | HALT

2.5. Modeliuojamos virtualios mašinos loginių komponentų sąryšio su realios mašinos techninės įrangos komponentais aprašymas.

Virtualiai mašinai atliekant komandas gali kilti pertraukimai. Jie apdorojami tik tada kai VM baigia vykdyti komandą.Tuomet reali mašina persijungia iš vartotojo režimo į supervizorinį.
Įvedimo/ Išvedimo veiksmas atliekamas supervizoriniu rėžimu, tam naudojama iniciavimo operacija StarIO – kuria nustatomi kanalai, jų panaudojimas ir tikrinamas užimtumas.
Norint iš supervizorinio rėžimo į vartotojo rėžimą darbo pratęsimui virtualioje mašinoje reikalinga pakrauti būsena.

3. Virtuali mašina operacinės sistemos kontekste

Operacinei sistemai turi būti pateikiamas užduočių rinkinys (programa). Tam reikalinga specifinė užduočių pateikimo kalba. Kiekviena užduotis suformuojama kaip failas. Užduotis sudaryta iš pateikiamų duomenų ir rezultatų. Vykdant užduotį ji yra išskaidoma į dalis: užduotis saugoma išorinėje atmintyje, kai ji paruošta vykdymui. Užduotis tiesioginės sąveikos su fiziniais įrenginiais neturi, tik su virtualiais.

4. Multiprograminės operacinės sistemos modelis

Šiuolaikinės multiprograminės operacinės sistemos gali vykdyti kelias programos vienu metu. Būtų labai nepatogu jeigu norėdamas atsisiųsti failą iš interneto turėtum laukti ir negalėtum nieko daryti su kompiuteriu. Žinoma multiprogramiškumas nėra abstakcija, kuri realiai neegzistuoja, o tik atrodo, kad keli procesai vyksta vienu metu. Tai palengvina programos vartotojiškuma, tačiau neparodo kaip ištikrūjų veikia operacinė sistema. OS veikimo struktūra bandysime išanalizuoti šiame darbe.

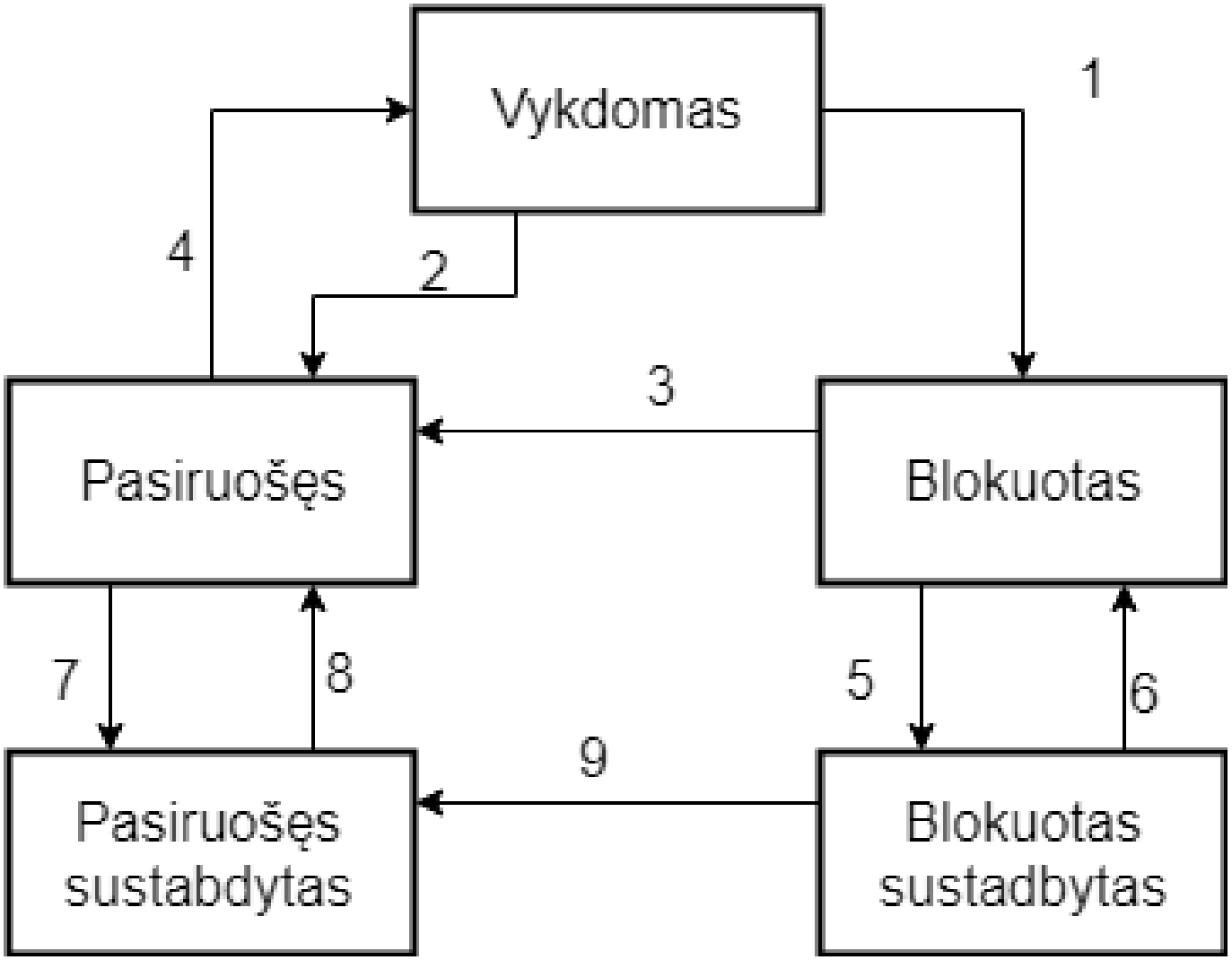
4.1. Procesai

Procesas tai yra programa kuri turi registrų reikšmes, kintamuosius ir virtualų procesorių. Procesas ir programa yra labai panašios savokos, skirtumas tik tas, kad procesas turi savo veiklumo būseną, o programa tėra baitų seka.

4.2. Procesų būsenos

Kiekvienas procesas turi jam priskirtą būseną:

- Pasiruošęs: vienintelis trūkstamas resursas yra procesorius
- Vykdomas - turi procesorių
- Blokuotas - prašo resurso (išskyrus procesorių)
- Sustabdytas – kito proceso sustabdytas procesas.



4 pav. Virtualios mašinos procesorius

- 1. Vykdomas procesas blokuojasi jam prašant ir negavus resurso.
- 2. Vykdomas procesas tampa pasiruošusiu atėmus iš jo procesorių dėl kokios nors priežasties (išskyrus resurso negavimą).
- 3. Blokuotas procesas tampa pasiruošusiu, kai yra suteikiamas reikalingas resursas.
- 4. Pasiruošę procesai varžosi dėl procesoriaus. Gavęs procesorių procesas tampa vykdomu.
- 5. Procesas gali tapti sustabdytu blokuotu, jei einamasis procesas jį sustabdo, kai jis jau ir taip yra blokuotas.
- 6. Procesas tampa blokuotu iš blokuoto sustabdyto, jei einamasis procesas nuimabūseną sustabdytas.
- 7. Procesas gali tapti pasiruošusiu sustabdytu, jei einamasis procesas jį sustabdo,kai jis yra pasiruošęs.
- 8. Procesas tampa pasiruošusiu iš pasiruošusio sustabdyto, jei einamasis procesas nuima būseną sustabdytas
- 9. Procesas tampa pasiruošusiu sustabdytu iš blokuoto sustabdyto, jei procesui yra suteikiamas jam reikalingas resursas.

5. Resursai

Resursai yra tai, dėl ko varžosi procesai. Dėl resursų trūkumo procesai blokuojasi, gavę reikiamą resursą, procesai tampa pasiruošusiais. Resursus galima skirstyti į:

- Statinius resursus. Kuriami sistemos kūrimo metu. Tai mašinos resursai, tokie kaip procesorius, atmintis ar kiti resursai, kurie sistemos veikimo metu nėra naikinami. Šie resursai gali būti laisvi, kai nei vienas procesas jų nenaudoja, arba ne, kada juos naudoja vienas ar keli, jei tą resursą galima skaldyti, procesai.
- Dinaminius resursus. Kuriami ir naikinami sistemos darbo metu. Šie resursai naudojami kaip pranešimai. Kartu su jais gali ateiti naudinga informacija. Kartais šio tipo resursas pats yra pranešimas. Pavyzdžiui, esantis laisvas kanalo resursas žymi, kad bet kuris procesas gali naudotis kanalu. Jei jo n÷ra, procesas priverstas laukti, kol šis resursas taps prieinamu (bus atlaisvintas).

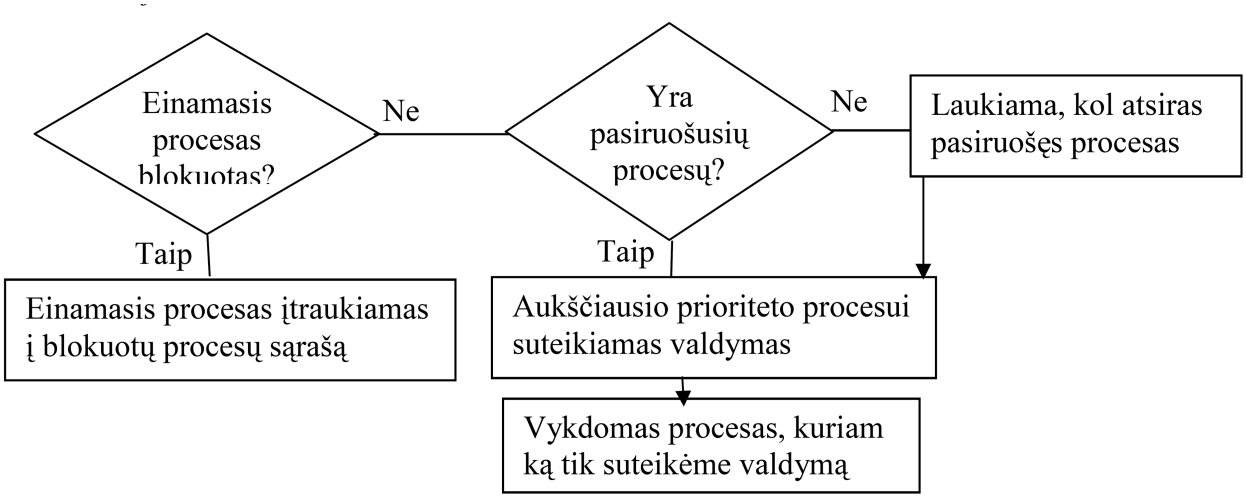
5.1. Resurso primityvai

Kiekvienas resursas turi keturis primityvus

- Kurti resursą. Resursus kuria tik procesas. Resurso kūrimo metu perduodami kaip parametrai: nuoroda į proceso kūrėją, resurso išorinis vardas. Resursas kūrimo metu yra: pridedamas prie bendro resursų sąrašo, pridedamas prie tėvo suskurtų resursų sąrašo, jam priskiriamas unikalus vidinis vardas, sukuriamas resurso elementų sąrašas ir sukuriamas laukiančių procesų sąrašas.
- Naikinti resursą. Resurso deskriptorius išmetamas iš jo tėvo sukurtų resursų sąrašo, naikinamas jo elementų sąrašas, atblokuojami procesai, laukiantys šio resurso, išmetamas iš bendro resursų sąrašo, ir, galiausiai naikinamas pats deskriptorius
- Prašyti resurso. Šį primityvą kartu su primityvu “atlaisvinti resursą” procesai naudoja labai dažnai. Procesas, iškvietęs šį primityvą, yra užblokuojamas ir įtraukiamas į to resurso laukiančių procesų sąrašą. Sekantis šio primityvo žingsnis yra kviesti resurso paskirstytoją.
- Atlaisvinti resursą. Šį primityvą kviečia procesas, kuris nori atlaisvinti jam nereikalingą resursą arba tiesiog perduoti pranešimą ar informaciją kitam procesui. Resurso elementas, primityvui perduotas kaip funkcijos parametras, yra pridedamas prie resurso elementų sąrašo. Šio primityvo pabaigoje yra kviečiamas resursų paskirstytojas.

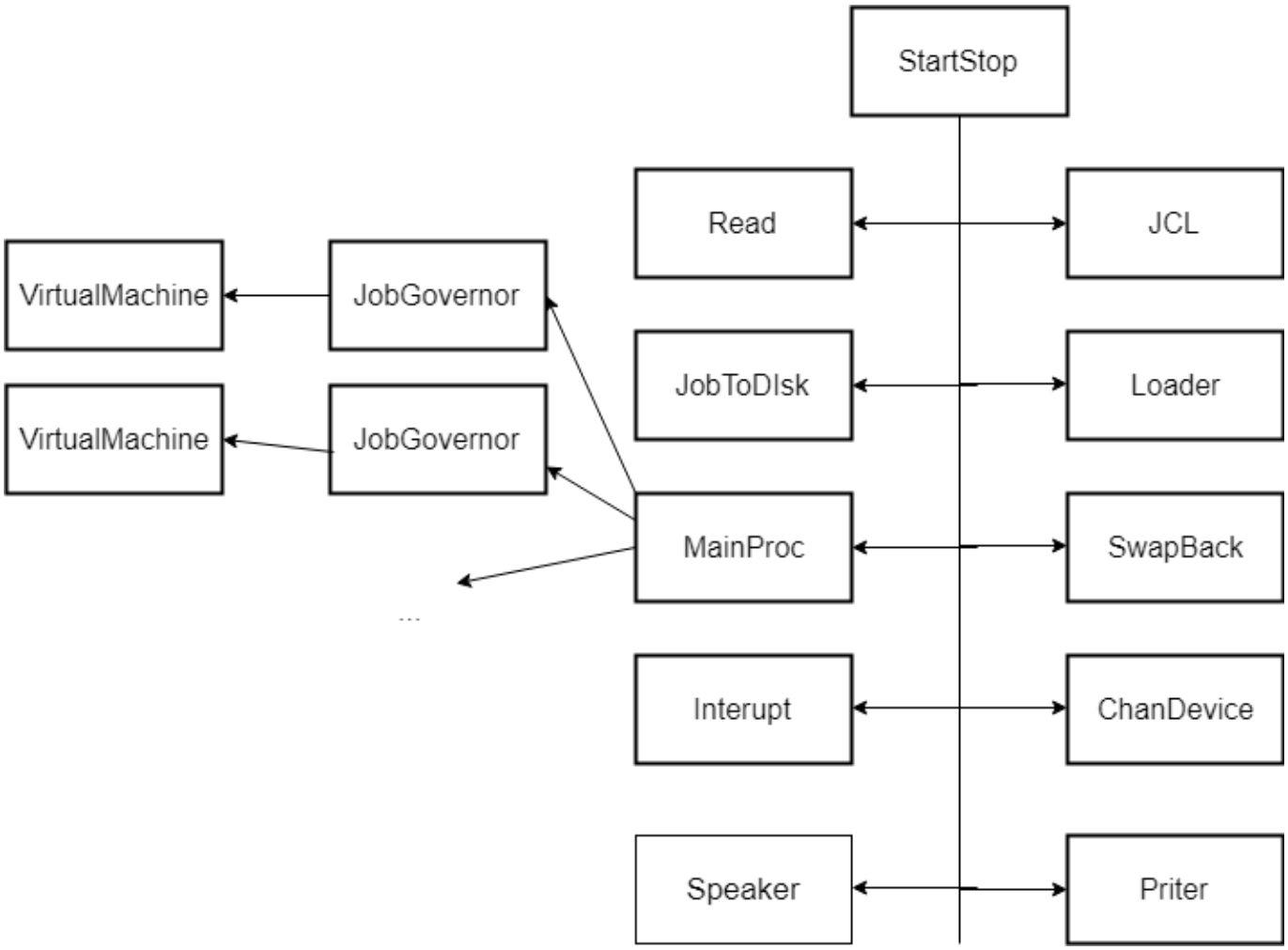
6. Planuotojas

Planuotojo paskirtis tvarkyti procesus, iš vienų procesų jis atiima procesorių ir kitiems atiduoda. Jis dirba su branduolio primityvais, procesų sarašu, procesų deskriptoriais ir procesoriaus resurso deskriptoriais. Planuotojas dirbs prioritetų principu, kur kiekvienas procesas turi prioriteto skaičių nuo 1 iki 100. Sisteminiasm procesams skirsime aukštenį prioriteta o vartotojiešiems žemesnį nes taip OS veiks greičiau.



5 pav. Procesu Veikimas

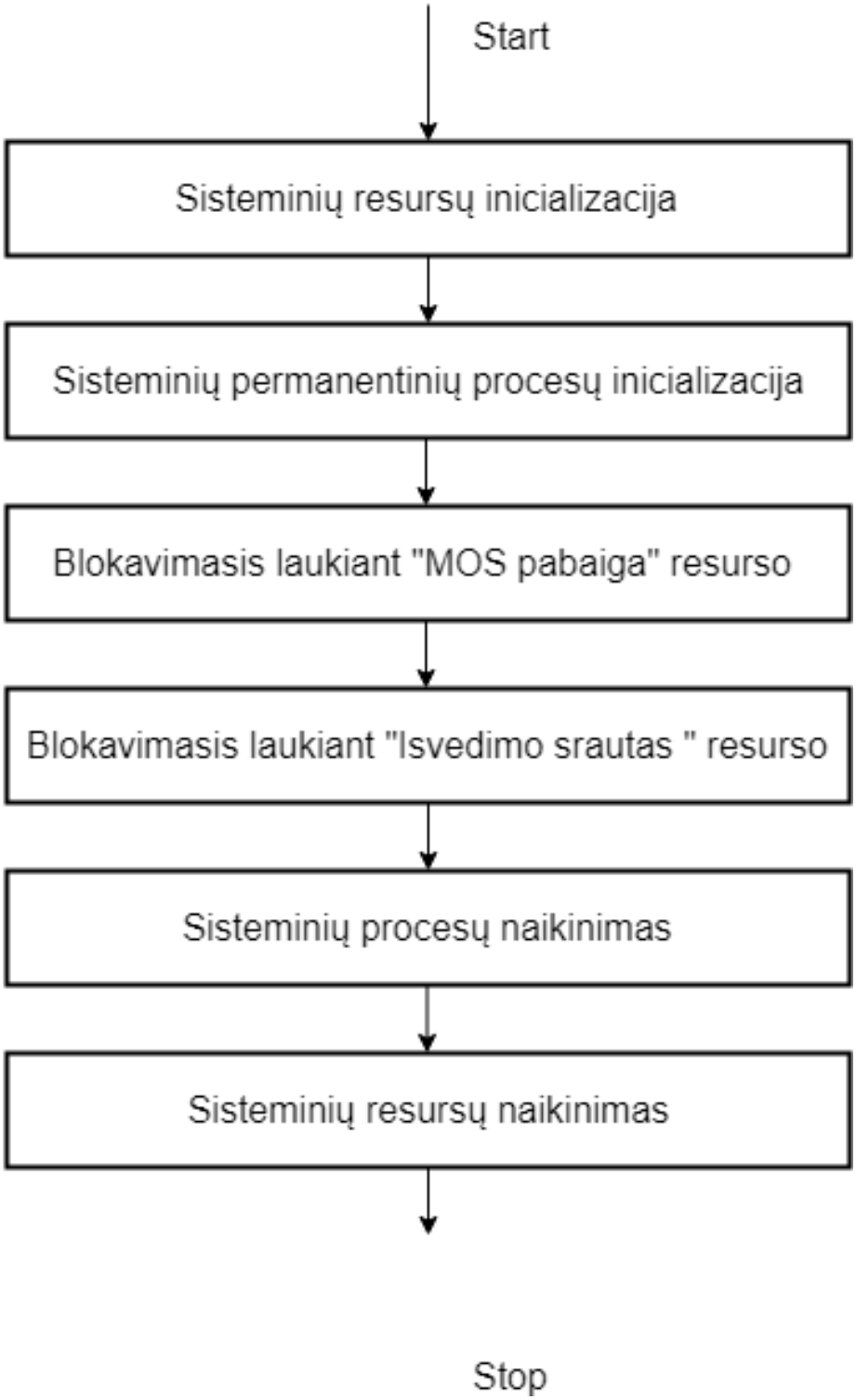
7. Procesų hierarchija



6 pav. Procesu Hierarchija

7.1. StratStop

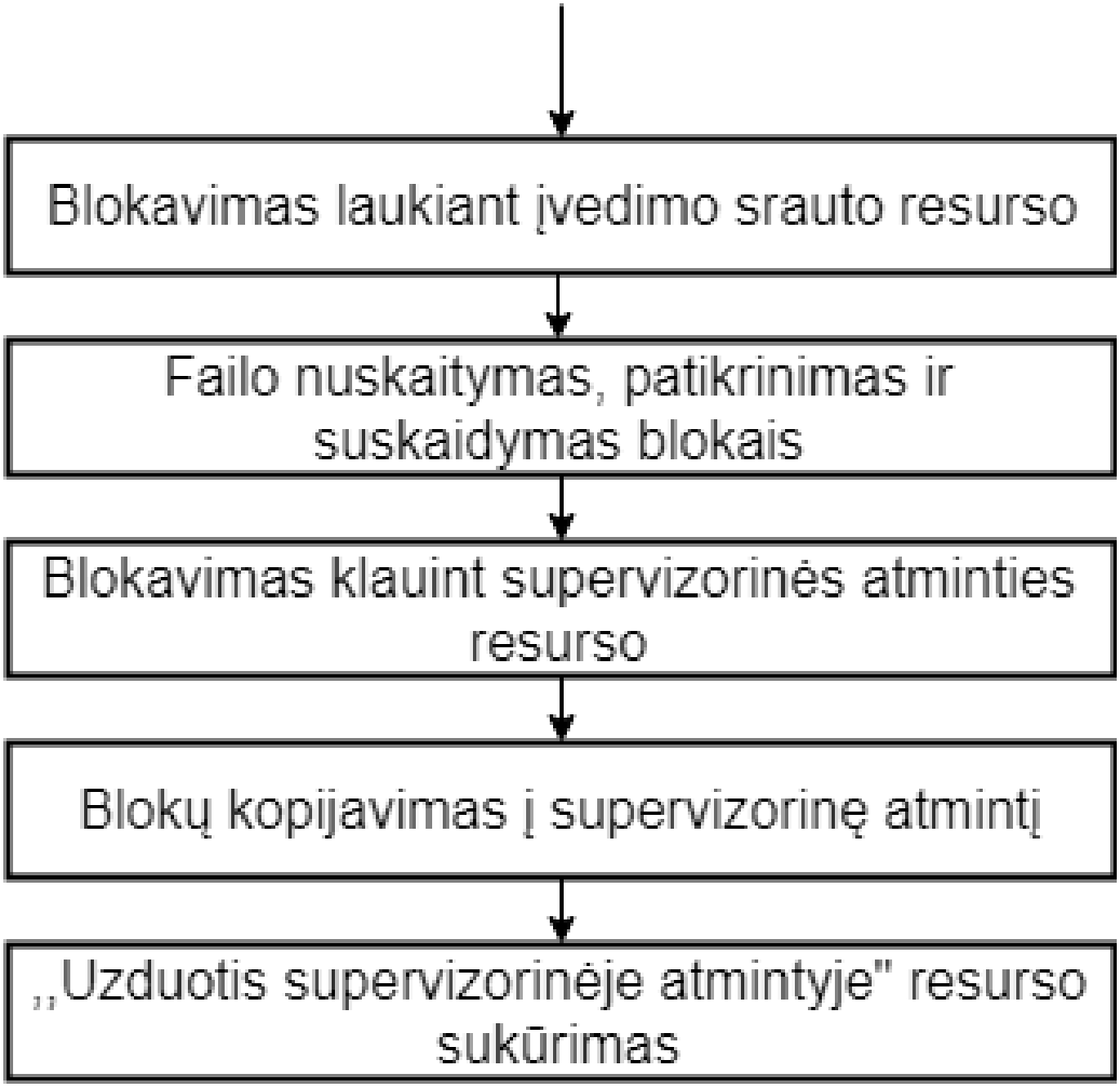
Procesas atsakingas už sistemos darbo pradžią ir pabaigą. Proceso paskirtis - sisteminių resursų kūrimas



7 pav. StartStop procesas

7.2. Read

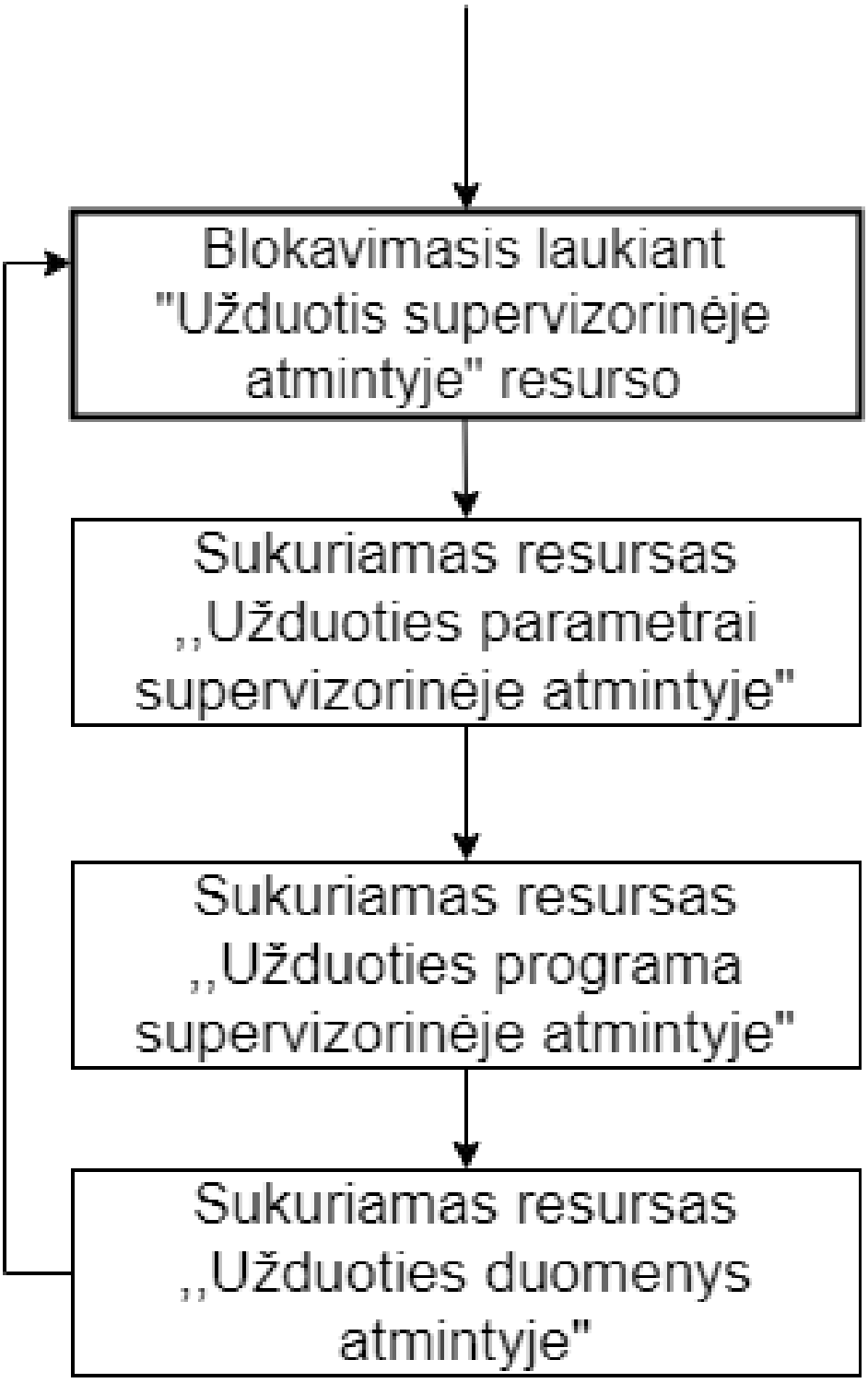
užduoties nuskaitymo iš įvedimoi srauto procesas.



8 pav. Read procesas

7.3. JCL

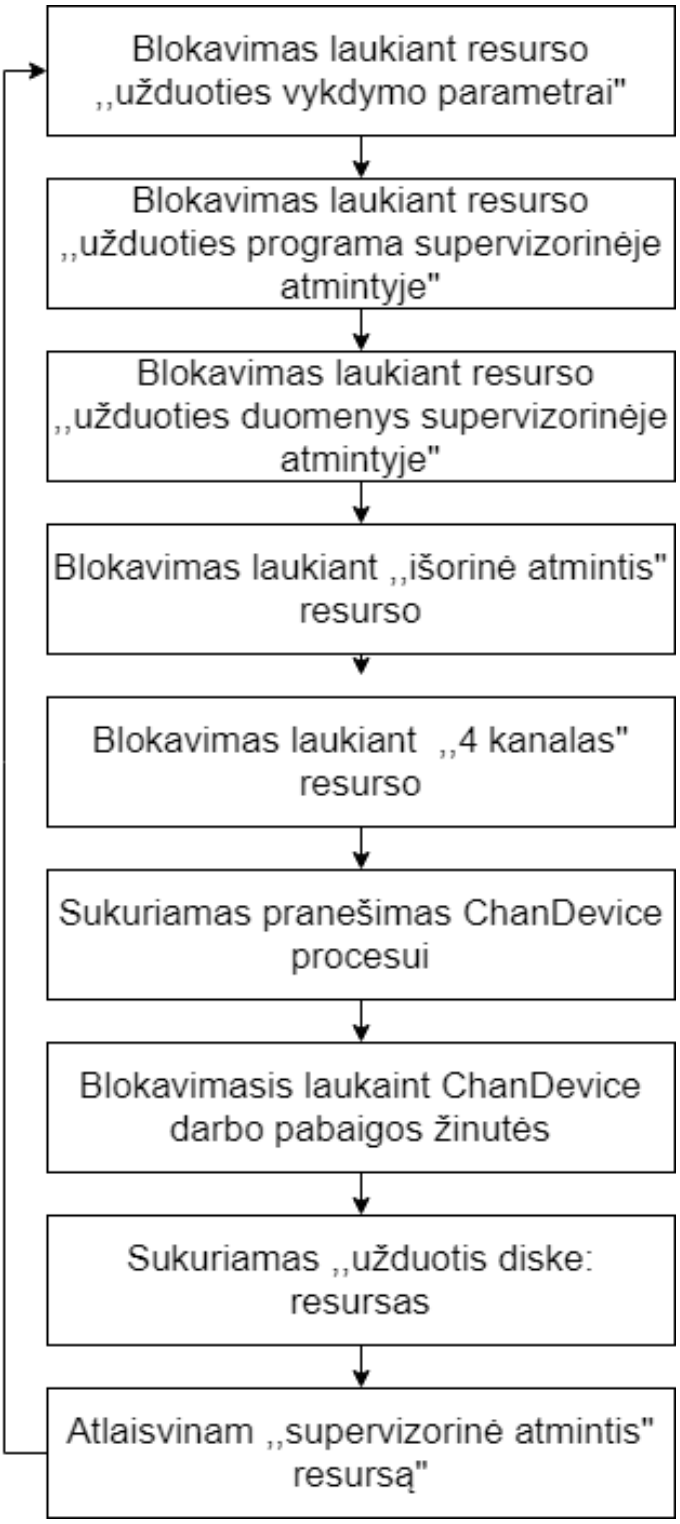
JCL paskirtis - interpretuoti užduoties tekstą, išskiriant duomenis ir paraketrus bei kuriant atitinkamus resursus.



9 pav. JLC

7.4. JobToDisk

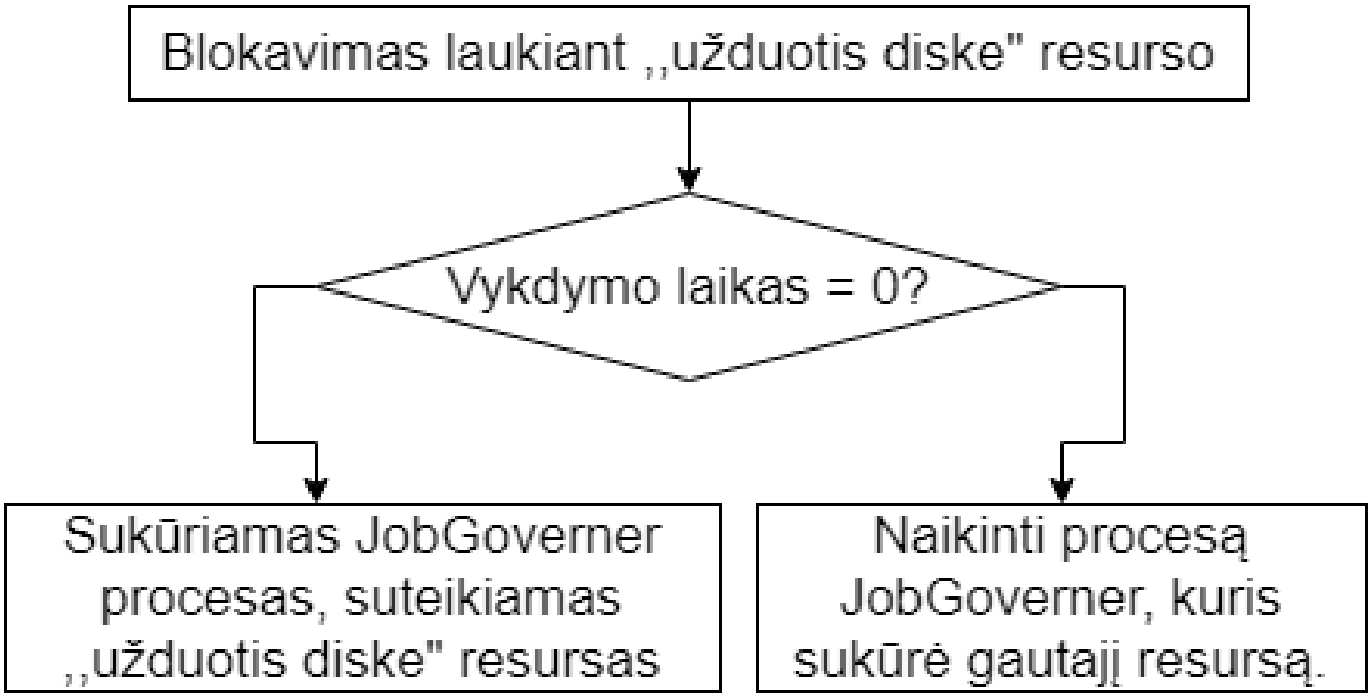
JobToDisk - procesas skirtas patarlpinti užduotį iš supervizorinės atminties į išorinę atmintį.



10 pav. JobToDisk procesas

7.5. MainProc

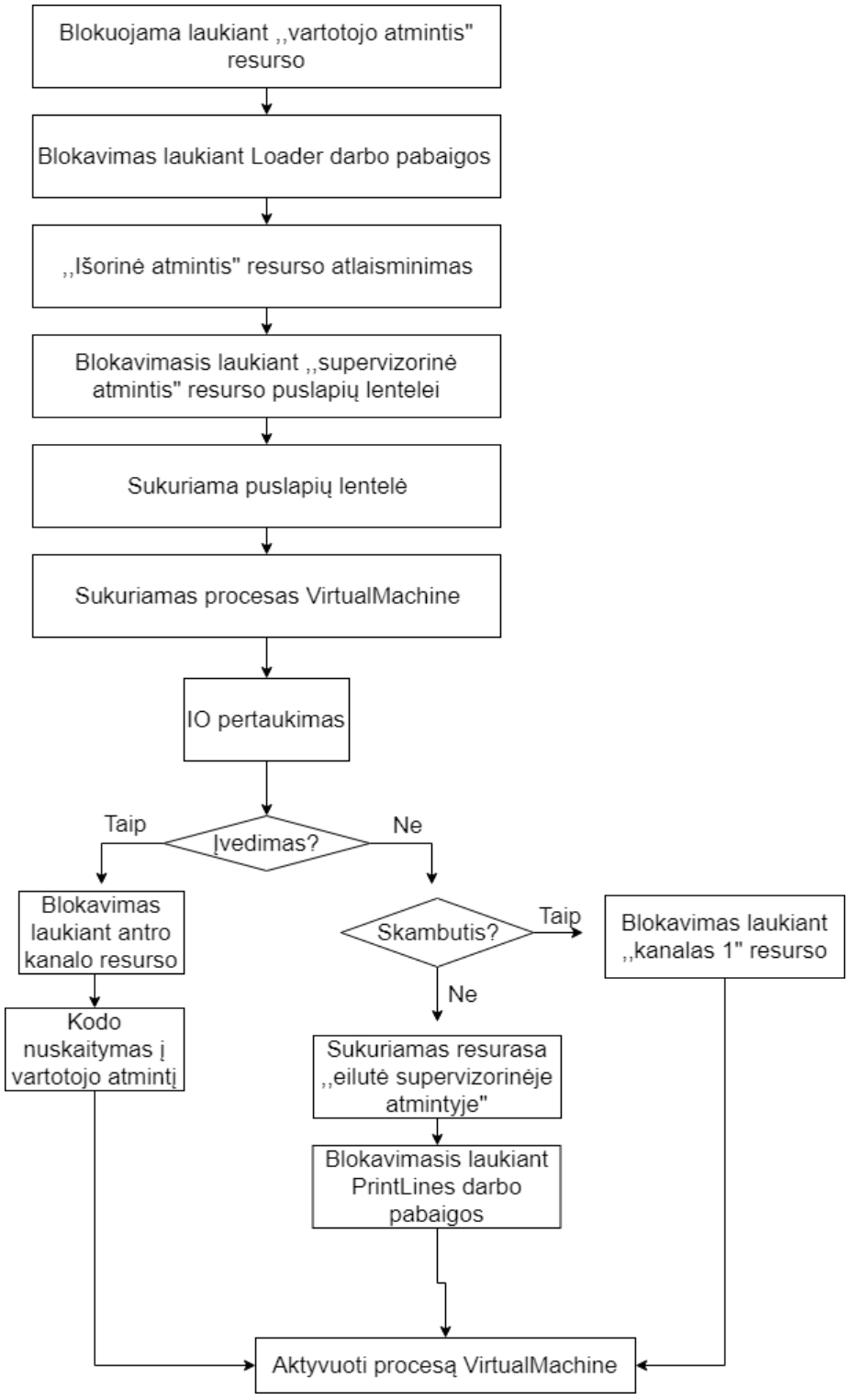
MainProc yra procesas atsakingas už užduočių vykdymo paruošimą ir sunaikinimą.



11 pav. MainProc

7.6. JobGovernor

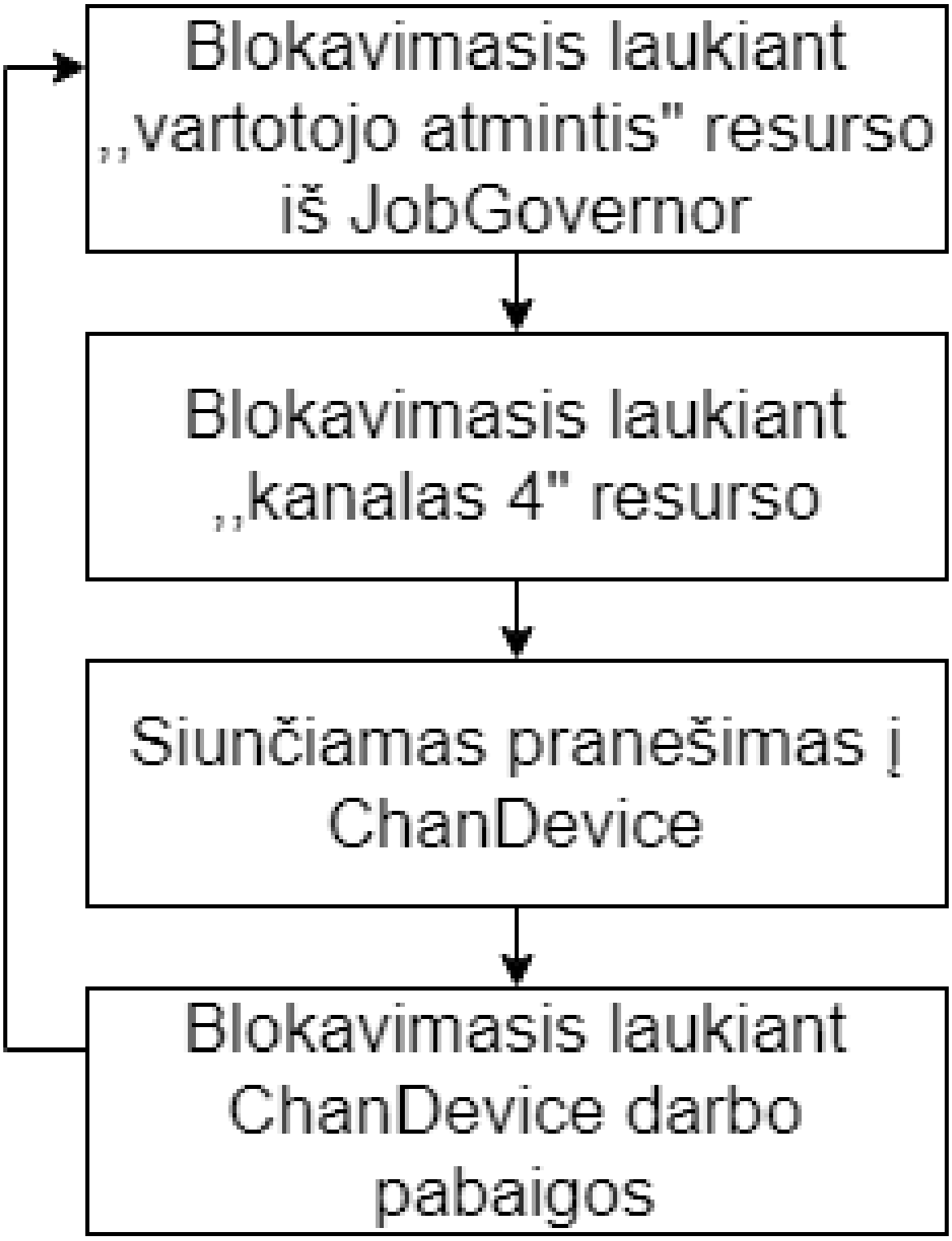
JobGovernor procesas atsakingas už virtualios mašinos proceso kūrimą ir tvarkantis jos darbą



12 pav. JobGovernor

7.7. SwapBack

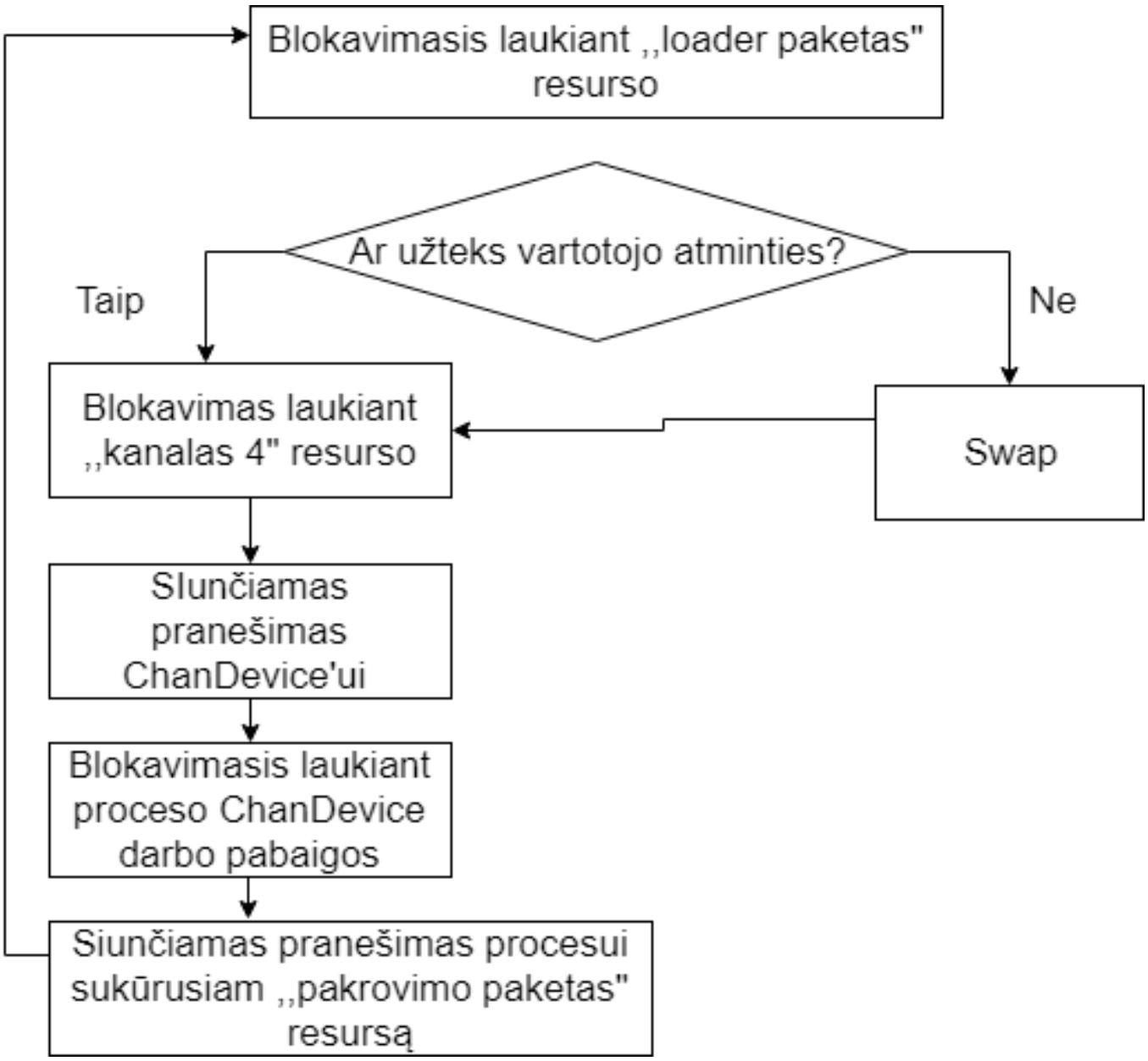
SwapBack - procesas gražinantis duomenis iš HDD swap bloko į vartotojo atmintį



13 pav. SwapBack procesas

7.8. Loader

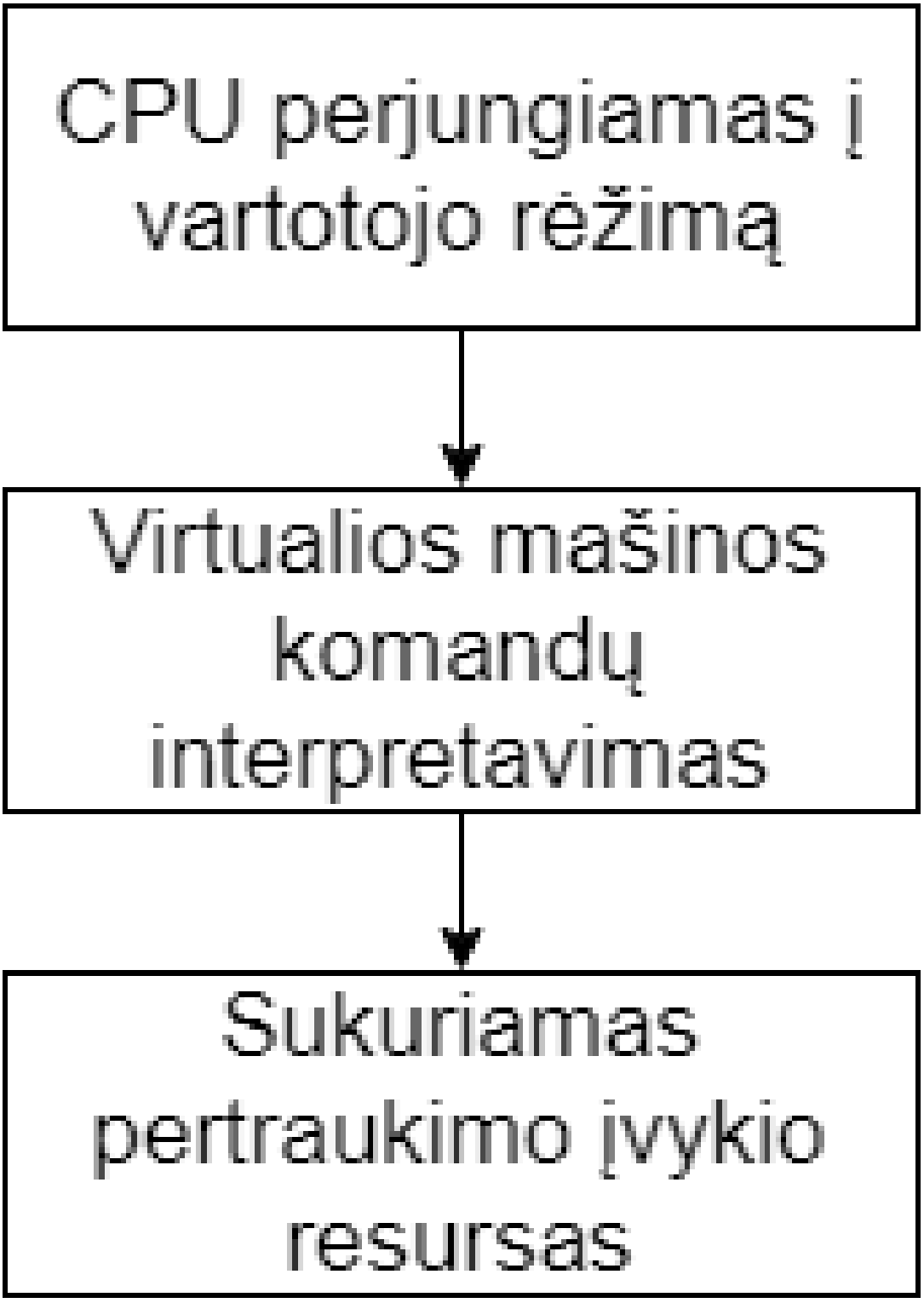
Loader - iš išorinės atminties programos tekstą perkelia į vartotojo atmintį



14 pav. Loader procesas

7.9. VirtualMachine

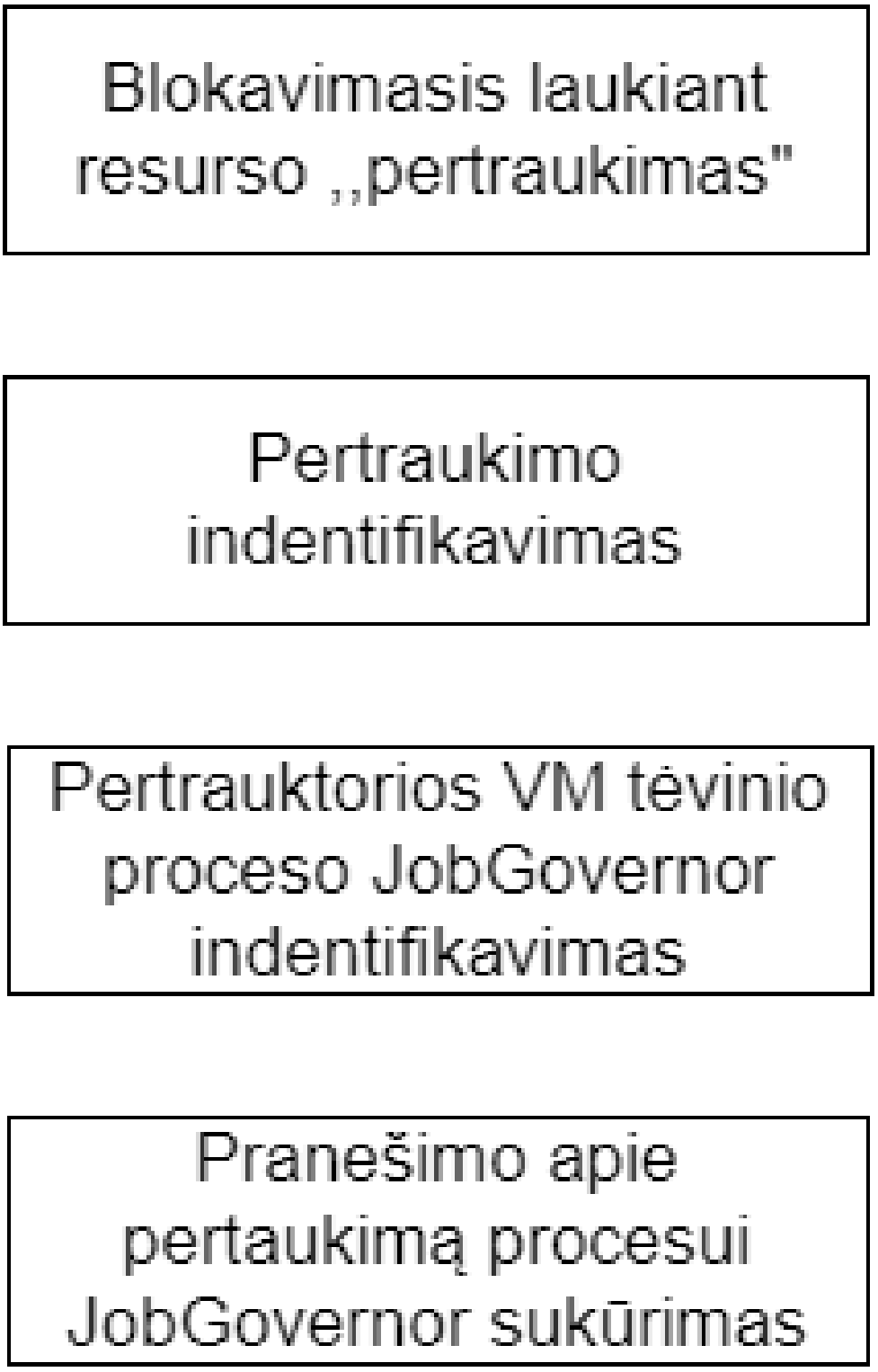
VirtualMachine - atsakingas už vartotojiškų procesų vykdymą.



15 pav. VirtualMachine procesas

7.10. Interupt

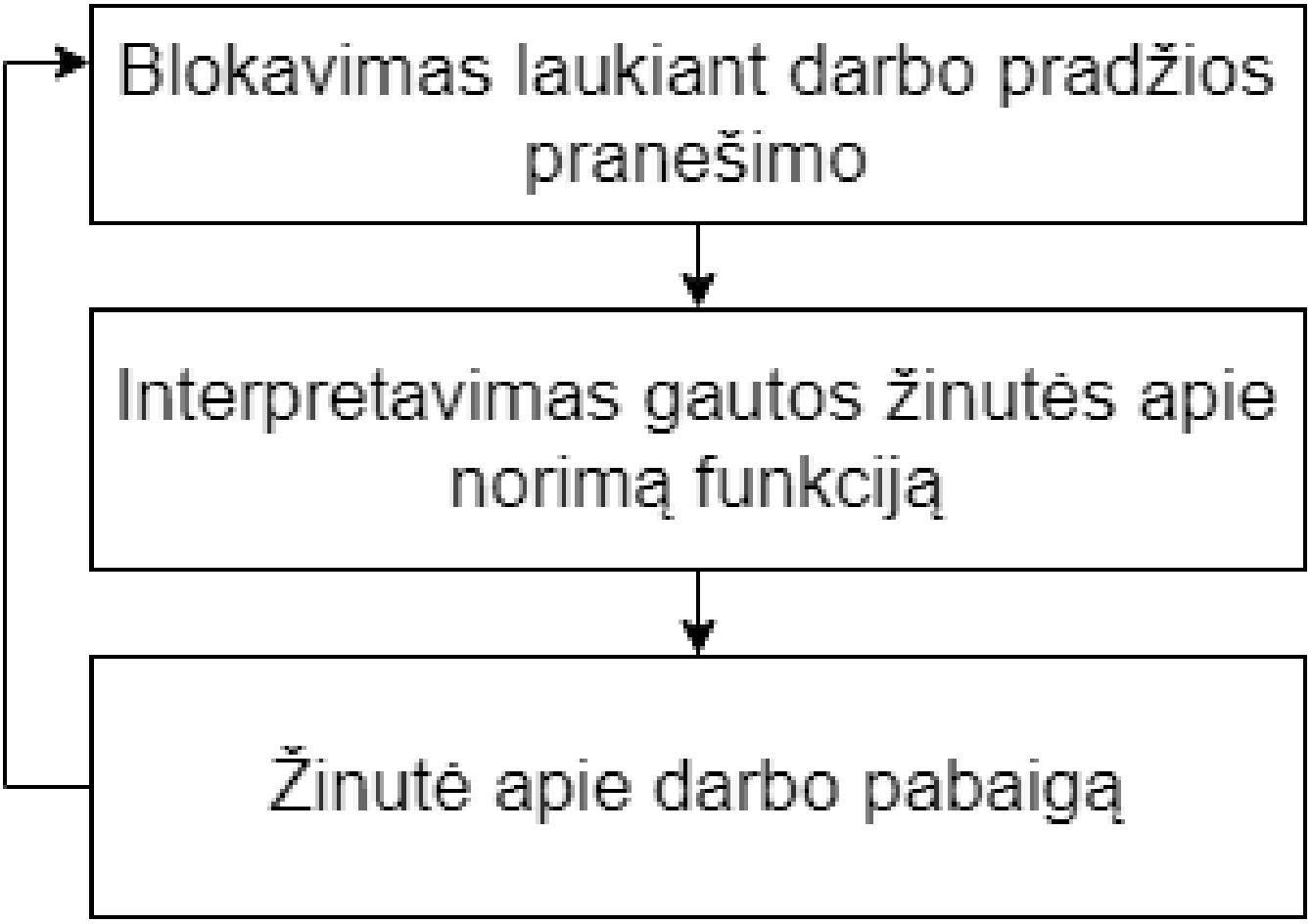
Interupt - atsakingas už VM pertraukimų interpretavimą.



16 pav. Interupt procesas

7.11. ChanDevice

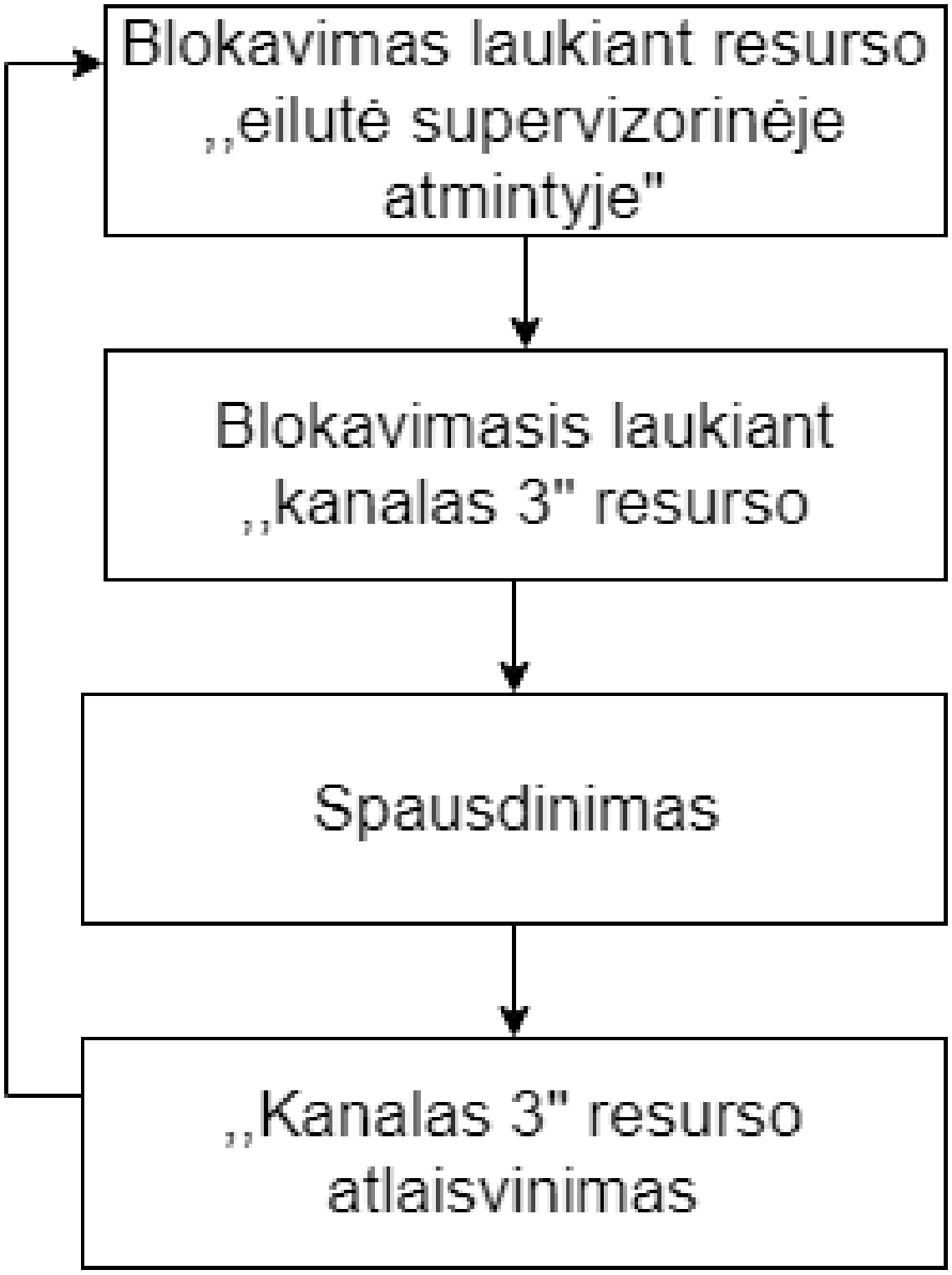
ChanDevice - kanalų įrenginio procesas skirtas valdyti 4 kanalus.



17 pav. ChanDevice procesas

7.12. PrintLine(Printeris)

- išvedimo įrenginiui siunčiama eilutė kurią reikės atspauisdinti



18 pav. PrintLine procesas

8. Proceso deskriptoriaus struktūra

ID: String // išorinis vardas

Resources: LinkedList // turimų resursų elementų sąrašas

State: String // RUN, READY, BLOCK, READYS, BLOCKS - proceso būsena

ProcessList: Queue // procesų sąrašas, kuriam šiuo metu priklauso procesas

T: int // tėvinio proceso vidinis vardas

S: LinkedList // sūnų sąrašas

Priority: int // proceso prioritetas

CPU: // procesoriaus einamoji būsena proceso metu

- mode: byte // supervizorinis/vartotojo darbo rėžimas
- entry: int // (sisteminiam procesui) - įėjimo taškas, nuo kurio bus pratęstas proceso darbas
- r1: int // registras, atitinkantis VM registrą R1
- r2: int // registras, atitinkantis VM registrą R2
- r3: int // registras, atitinkantis VM registrą R3
- r4: int // registras, atitinkantis VM registrą R4
- ic: ushort // registras, atitinkantis VM komandų skaitliuką (2 baitai)
- sf: byte // registras, atitinkantis VM būsenų flag'ą
- ti: ushort // registras, atitinkantis RM laikmatį (darbo trukmės skaitliuką)
- pi: short // registras, atitinkantis RM programinių pertraukimų registrą
- si: short // registras, atitinkantis RM sisteminių pertraukimų registrą
- ioi: byte // registras, atitinkantis RM kanalų registrą

9. Resurso deskriptoriaus struktūra

RID: string // resurso vardas

ID: int // sukūrusio proceso vidinis vardas

Reusability: boolean // ar resursas pakartotinio naudojimo

Availability: // resurso prieinamumo aprašymas
msg: boolean // ar pranešimas, ar išmatuojamas resursas
ElementList: List // resurso elementų sąrašas
amount: // kiekis prieinamų išmatuojamojo resurso elementų
receiver: int // pranešimo gavėjo (proceso) vidinis vardas

WaitingList: Queue // resurso laukiančių procesų sąrašas
ProcessID: int // proceso vidinis vardas
amount: int // norimas kiekis resurso elementų
msg: string // laukiamo pranešimo/ reikiamo kanalo identifikatorius

Distributor: method() // paskirstytojo funkcija