

VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Virtualios ir realios mašinos modelis

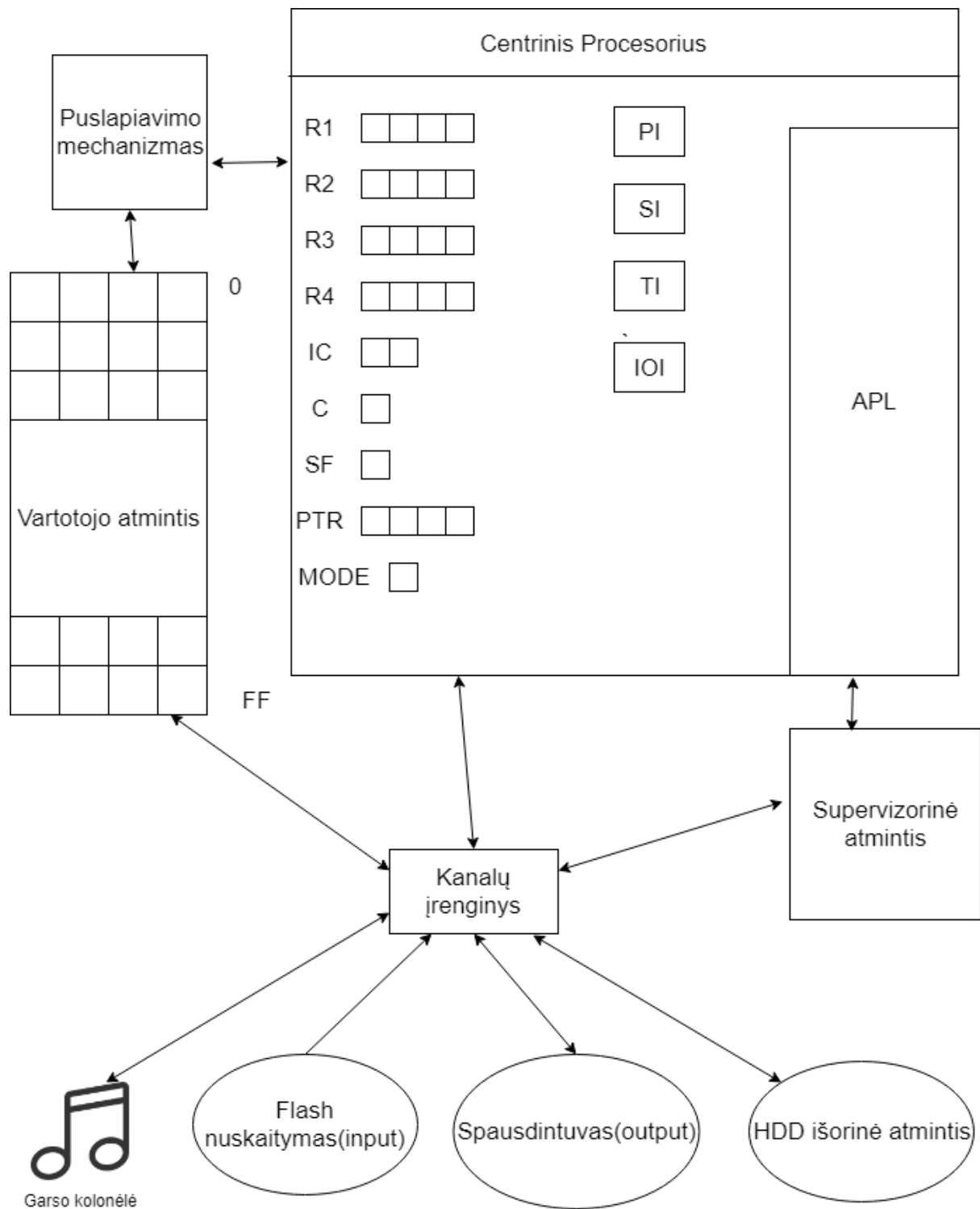
Virtual and real machine model

Atliko:	2 kurso 3 grupės studentai	
	Matas Savickis	(parašas)
	Justas Tvarijonas	(parašas)
	Greta Pyrantaitė	(parašas)
Darbo vadovas:	Mantas Grubliauskas Lekt.	(parašas)

TURINYS

1. REALI MAŠINA	2
1.1. Techninės įrangos elementai sudarantys realią mašiną	3
1.2. Centrinis procesorius	3
1.3. Procesoriaus registrai	3
1.4. Atmintys	4
1.5. I/O	4
1.6. Puslapiavimo mechanizmas	4
1.7. Garso kolonėlė	4
1.8. Pertraukimų mechanizmas	4
1.9. Timerio mechanizmas	4
1.10. Duomenų perdavimo kanalai	5
2. VIRTUALI MAŠINA	6
2.1. Funkcija	6
2.2. Modeliuojamos virtualios mašinos loginių komponentų aprašymas	6
2.2.1. Atmintis	6
2.2.2. Procesorius	7
2.2.3. Virtualios mašinos komandų sistema	7
2.3. Virtualios mašinos bendravimo su įvedimo/išvedimo įrenginiais mechanizmo aprašymas	9
2.4. Virtualios mašinos interpretuojamojo ar kompiliuojamo vykdomojo failo išeities teksto formatas	9
2.5. Modeliuojamos virtualios mašinos loginių komponentų sąryšio su realios mašinos techninės įrangos komponentais aprašymas	10
3. VIRTUALI MAŠINA OPERACINĖS SISTEMOS KONTEKSTE	11
4. MULTIPROGRAMINĖS OPERACINĖS SISTEMOS MODELIS	12
4.1. Procesai	12
4.2. Procesų būsenos	12
5. RESURSAI	14
5.1. Resurso primityvai	14
6. PLANUOTOJAS	15
7. PROCESŲ HIERARCHIJA	16
7.1. StratStop	16
7.2. Read	17
7.3. JCL	18
7.4. JobToDisk	19
7.5. MainProc	20
7.6. JobGovernor	21
7.7. SwapBack	22
7.8. Loader	23
7.9. VirtualMachine	24
7.10. Interrupt	25
7.11. ChanDevice	26
7.12. PrintLine(Printeris)	27
8. PROCESO DESKRIPTORIAUS STRUKTŪRA	29
9. RESURSO DESKRIPTORIAUS STRUKTŪRA	30

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 režimais - supervizoriaus ir vartotojo. Supervizoriaus režime komandos yra apdorojamos aukšto lygio procesoriaus. Komandos vykdomos supervizoriaus režime yra skirtos operacinės sistemos funkcionavimui palaikyti. Procesorius persijungia į šį režimą pertaukimais arba sisteminiais kreipiniais.

1.3. Procesoriaus registrai

- R1, R2, R3, R4 - 4 baitų bendros paskirties registrai
- IC - 2 baitų komandų skaitiklis
- C - 1 baito loginis registras (TRUE arba FALSE)
- SF - 1 baito požymių registras CF ZF SF IF OF XXX - carry flag, zero flag, sign flag, interrupt 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 sukuriamą 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: $\text{Realus adresas} = 16 * [16 * (16 * a2 + a3) + x1] + x2$.

1.7. Garso kolonėlė

Vartotojui įvedus atitinkamą komandą kompiuterio garso kolonėlė supypsi $x1$ kartų.

1.8. Pertraukimų mechanizmas

Vykdam komandą virtualioje mašinos gali kilti pertraukimai, tuo atvėju nustatoma atitinkamu registru 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;

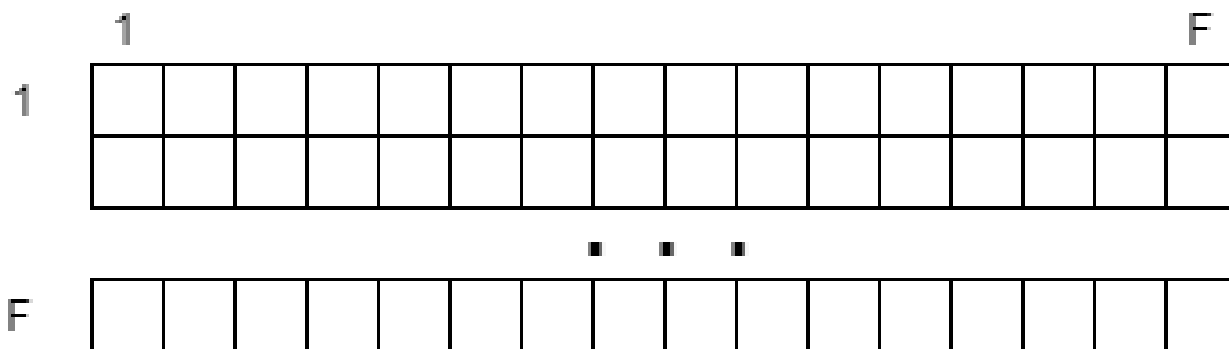
2. Virtuali mašina

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

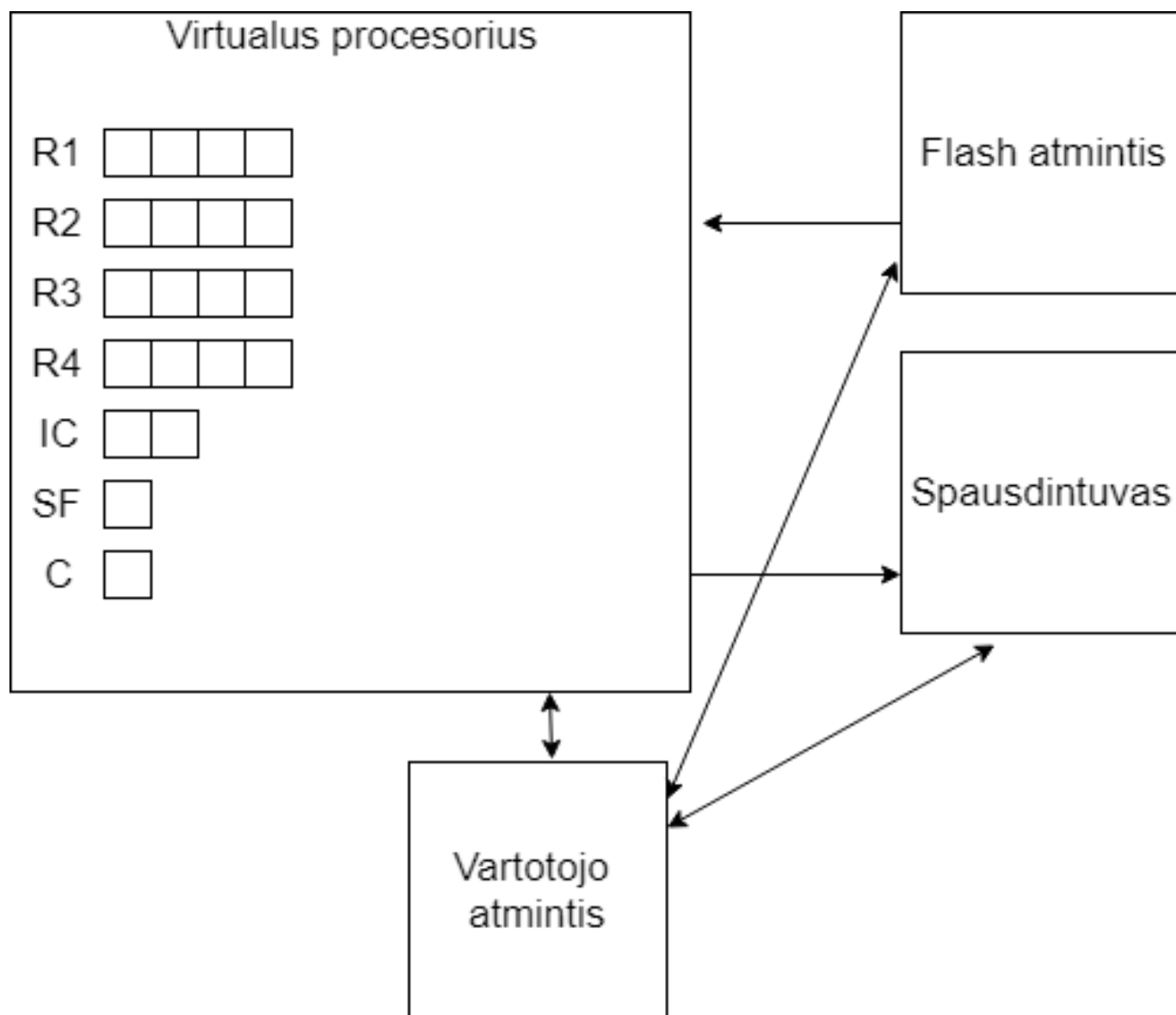
2.2.1. Atmintis



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: $\text{Realus adresas} = 16 * [16 * (16 * a2 + a3) + x1] + x2$

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
- C - loginis registras(TRUE arba FALSE)

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 baido $x1x2$ persiunčia į registrą R: $LR\ x1x2 \Rightarrow R := [x1x2]$;
- (b) SR – Save Register – iš registro R persiunčia į atminties baitą $x1x2$: $SR\ x1x2 \Rightarrow [x1x2] := R$;

2. Duomenų sukeitimui tarp registų:

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

3. Aritmetinės komandos:

- (a) AD – suma – prie esamos registro R reikšmės prideda reikšmę esančią $x1x2$ atminties baite, rezultatas patalpinamas registre R: $AD\ x1x2 \Rightarrow r1 := r1 + [x1x2]$;
- (b) SB – atimtis – iš esamos registro R1 reikšmės atimama reikšmė esanti $x1x2$ atminties baite, rezultatas patalpinamas registre R: $SB\ x1x2 \Rightarrow r1 := r1 - [x1x2]$;
- (c) CR – palyginimas – esamą registro R reikšmę yra lyginama su reikšme esančią $x1x2$ atminties baite, rezultatas patalpinamas registre C: $CR\ x1x2 \Rightarrow \text{if } r1 > [x1x2] \text{ then } cf := 0, zf := 0; \text{ if } r1 = [x1x2] \text{ then } zf := 1; \text{ if } r1 < [x1x2] \text{ then } cf := 1$;
- (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 supypsi $x1$ kartų.
- (b) PYC - Patikrina garso kolonėlės užimtumą.(1 - užimta, 0 - laisva).
- (c) JU – sąlyginio valdymo perdavimas – valdymas perduodamas adresu $16 * x1 + x2$: $JU\ x1x2 \Rightarrow IC := 16 * x1 + x2$;
- (d) JG – sąlyginio valdymo perdavimas (jeigu daugiau) – valdymas perduodamas jeigu SF bitai $ZF == 0$ ir $SF == OF$. Valdymas perduodamas adresu $16 * x1 + x2$: $JG\ x1x2 \Rightarrow \text{If } ZF == 0 \text{ AND } SF == OF \text{ then } IC := 16 * x1 + x2$;
- (e) JE – sąlyginio valdymo perdavimas (jeigu lygu) – valdymas perduodamas jeigu SF bitas $ZF == 1$. Valdymas perduodamas adresu $16 * x1 + x2$: $JE\ x1x2 \Rightarrow \text{If } ZF == 1 \text{ then } IC := 16 * x1 + x2$;
- (f) JL – sąlyginio valdymo perdavimas (jeigu mažiau) – valdymas perduodamas jeigu SF bitai SF ir OF nelygūs, valdymas perduodamas adresu $16 * x1 + x2$: $JL\ x1x2 \Rightarrow \text{If } SF \neq OF \text{ then } IC := 16 * x1 + x2$;

5. Programos pabaigos:

- (a) HALT – programos pabaigos komanda.

6. Įvedimo/Išvedimo:

- (a) GD – įvedimas – iš įvedimo srauto paima 4 žodžių srautą ir jį įveda į atmintį pradedant atminties baitu $16 * x1 + x2$: $GD\ x1x2$
- (b) CKP - patikrina printerio užimtumą.(1 - užimta, 0 - laisva)

- (c) PD – išvedimas – iš atminties, pradedant atminties baitu $16 \cdot x_1 + x_2$ paima 4 žodžių srautą ir jį išveda į ekraną: PD $x_1 x_2$

7. Loginės:

- (a) AND – $r_1 := r_1 \text{ and } r_2$
- (b) XOR – $r_1 := r_1 \text{ xor } r_2$
- (c) OR – $r_1 := r_1 \text{ or } r_2$.
- (d) NOT – $r_1 := \text{not } r_1$.

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 reiškinių „ $100 + 20 - 80$ “ reikšmę, bei ją išveda į ekraną.

000 | DATA

001 | 100

002 | 20

003 | 80

004 | Rezu

005 | ltat

009 | as y

00A | ra:

080 | CODE

081 | LR 01

082 | AD 02

083 | SB 03
084 | PD 04
085 | PD 05
086 | PD 06
087 | PD 07
088 | SR 10
089 | PD 10
08A | HALT
0FF |

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 režimu, tam naudojama iniciavimo operacija StarIO – kuria nustatomi kanalai, jų panaudojimas ir tikrinamas užimtumas.

Norint iš supervizorinio režimo į vartotojo režimą darbo pratęsimui virtualioje mašinoje reikalinga pakrauti būseną tam naudojama operacija Slave(plr,c,r,ic) – kur registrų panaudojimas sutampa su realios mašinos.

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. Vykdam 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 programas vienu metu. Būtų labai nepatogu jeigu norėdamas atsisiųsti failą iš interneto turėtum laukti ir negalėtum nieko daryti su kompiuteriu. Žinoma multiprogramiškumas nėra abstrakcija, kuri realiai neegzistuoja, o tik atrodo, kad keli procesai vyksta vienu metu. Tai palengvina programos vartotojiškumą, tačiau neparodo kaip ištikrųjų veikia operacinė sistema. OS veikimo struktūra bandysime išanalizuoti šiame darbe.

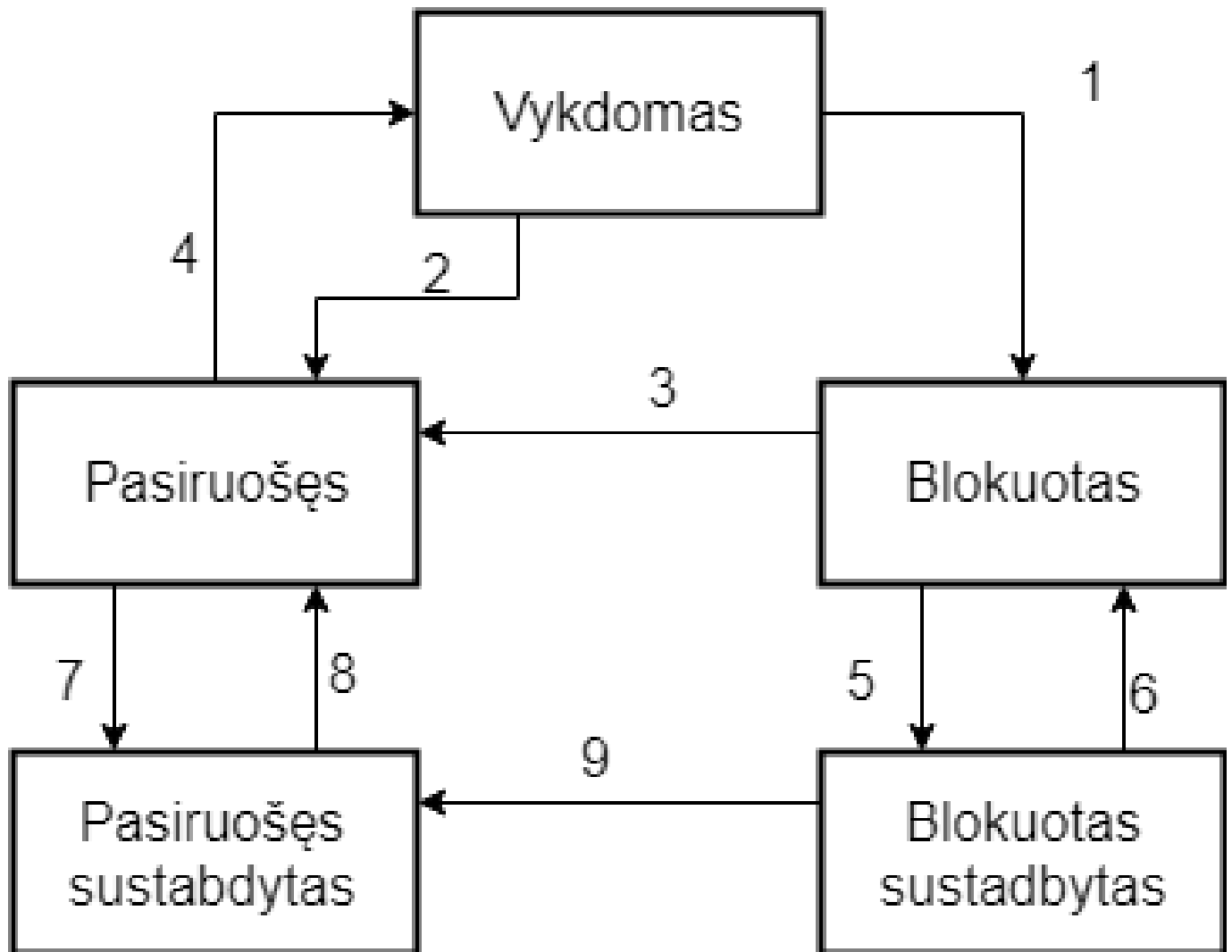
4.1. Procesai

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 nėra, procesas priverstas laukti, kol šis resursas taps prieinamu (bus atlaisvintas).

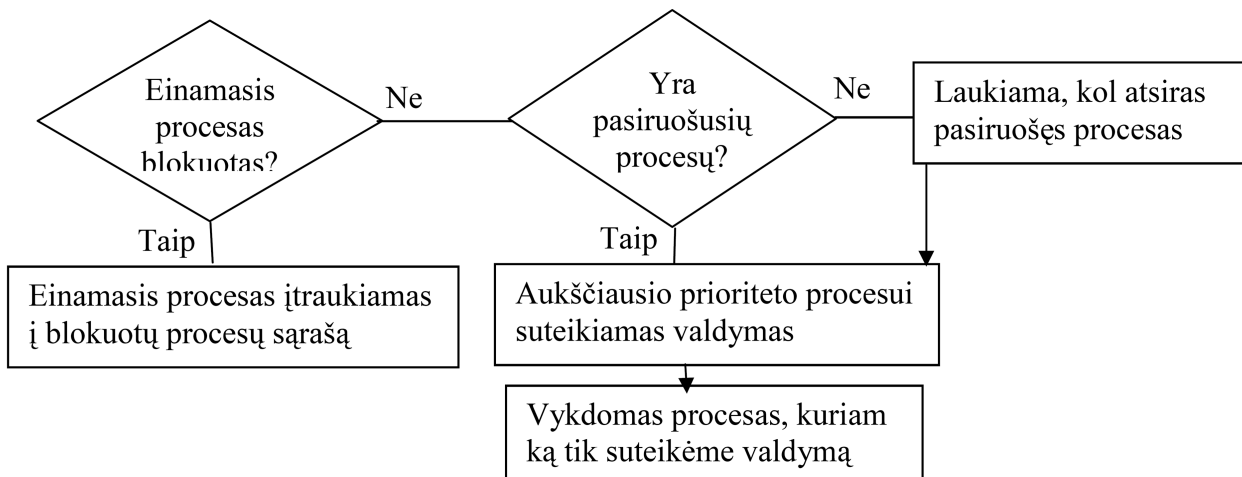
5.1. Resurso primityvai

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 sukurtų 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.

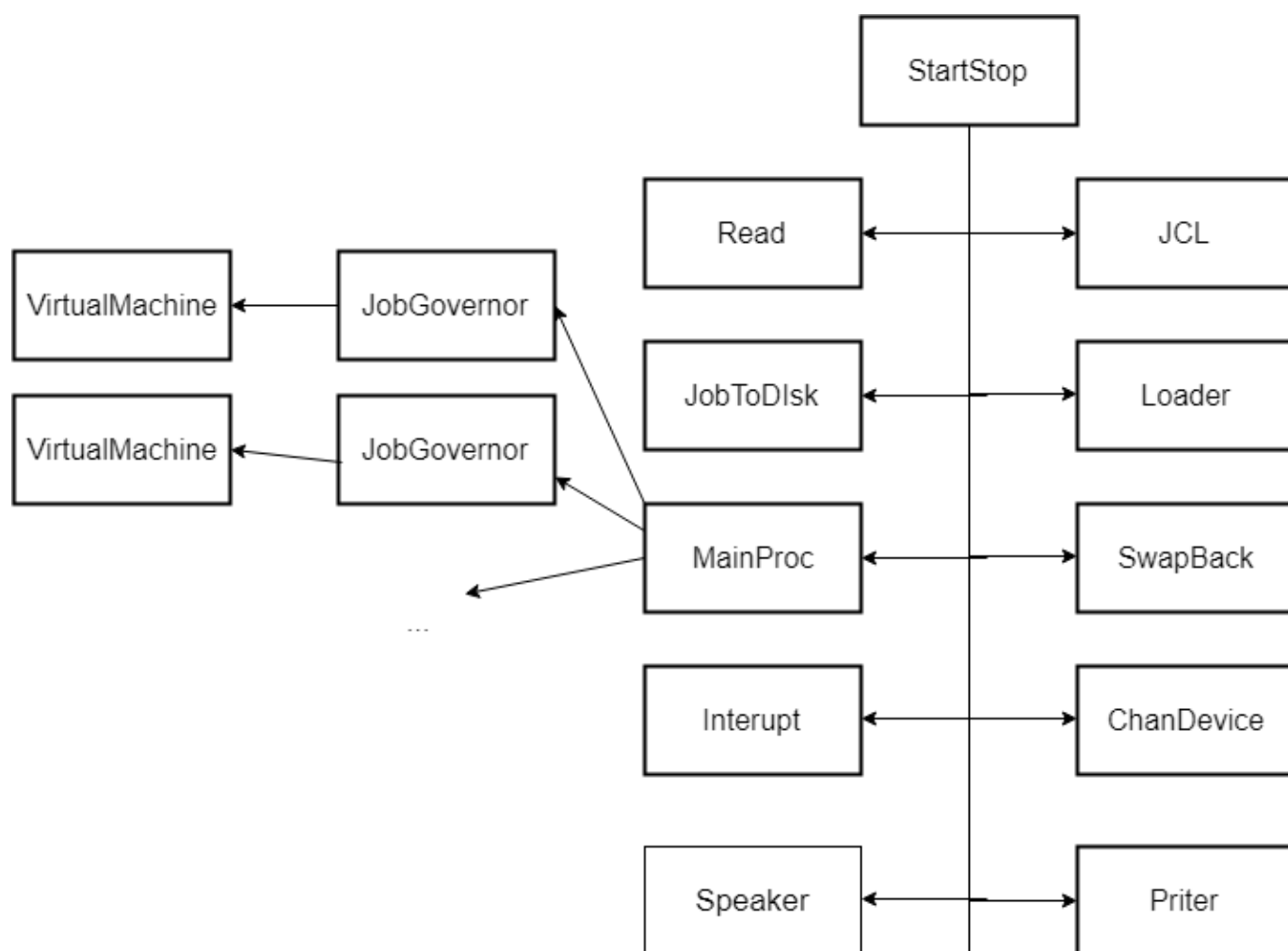
6. Planuotojas

Planuotojo paskirtis tvarkyti procesus, iš vieno procesų jis atima procesorių ir kitiems atiduoda. Jis dirba su branduolio primitivais, procesų sąrašu, procesų deskriptoriais ir procesoriaus resurso deskriptoriais. Planuotojas dirbs prioritetų principu, kur kiekvienas procesas turi prioriteto skaičių nuo 1 iki 100. Sisteminiams procesams skirsime aukštesnį prioritetą o vartotojiškiems žemesnį nes taip OS veiks greičiau.



5 pav. Procesu Veikimas

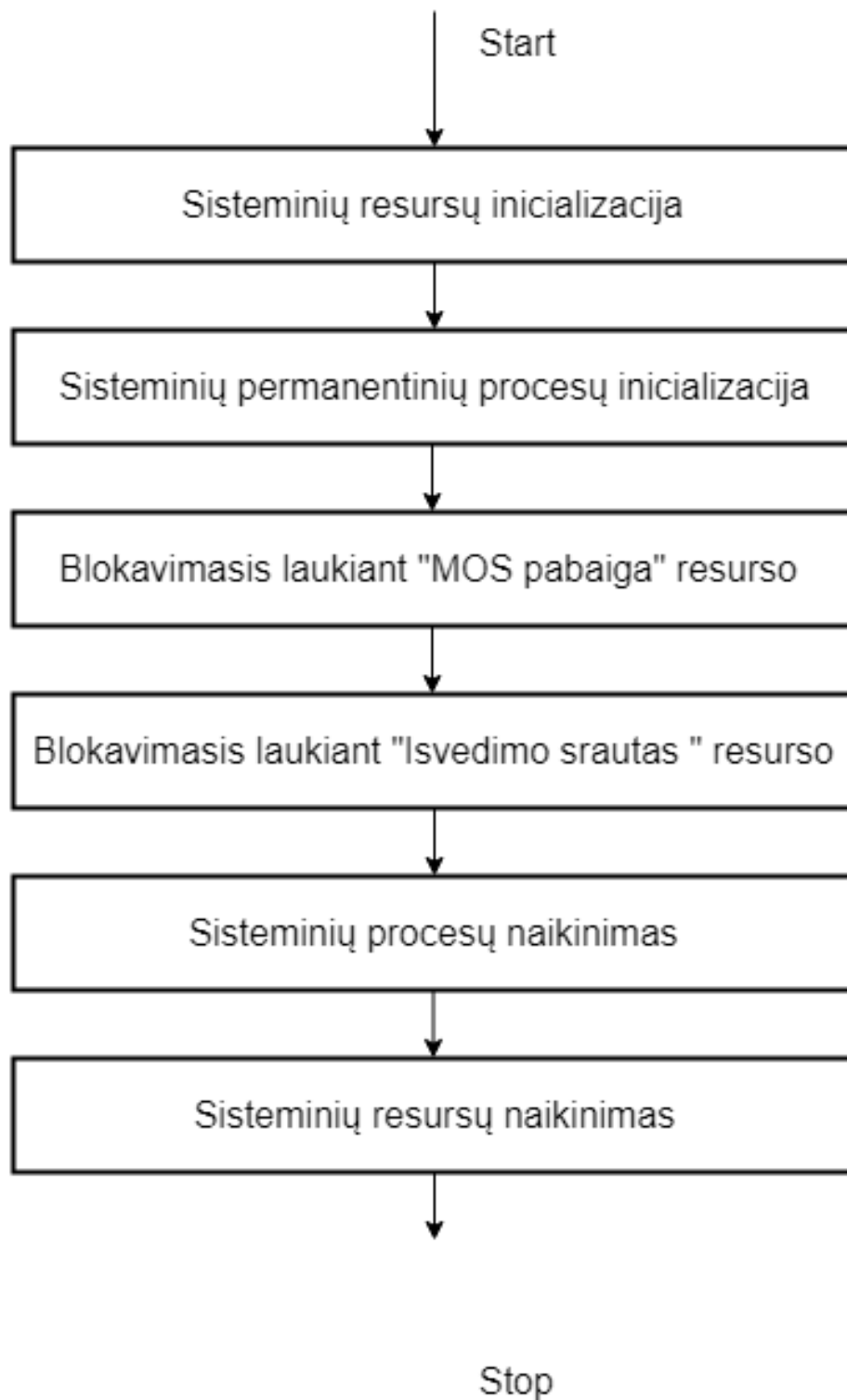
7. Procesų hierarchija



6 pav. Procesu Hierarchija

7.1. StratStop

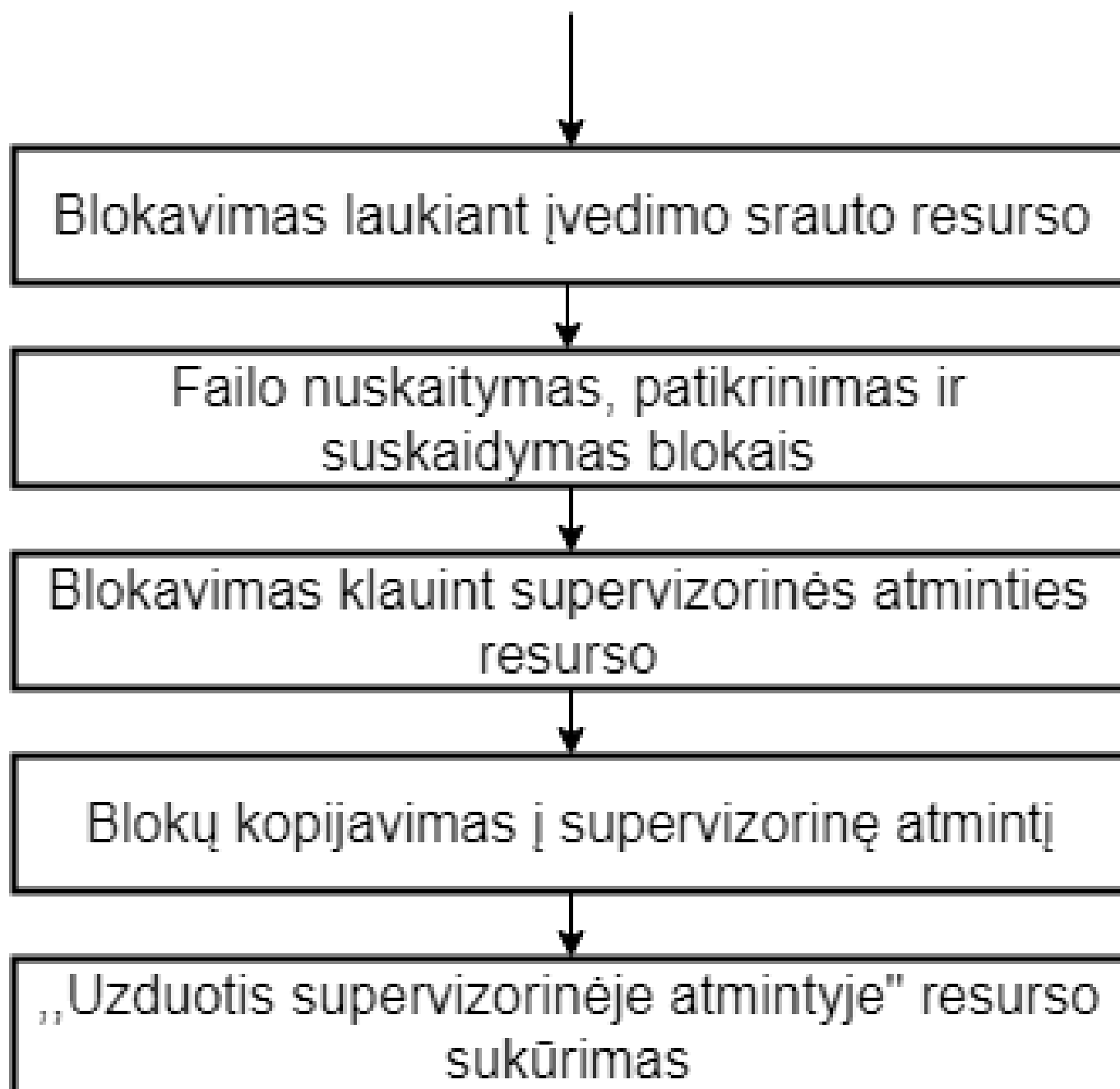
Procesas atsakingas už sistemos darbo pradžią ir pabaigą. Proceso paskirtis - sisteminių resursų kūrimas



7 pav. StartStop procesas

7.2. Read

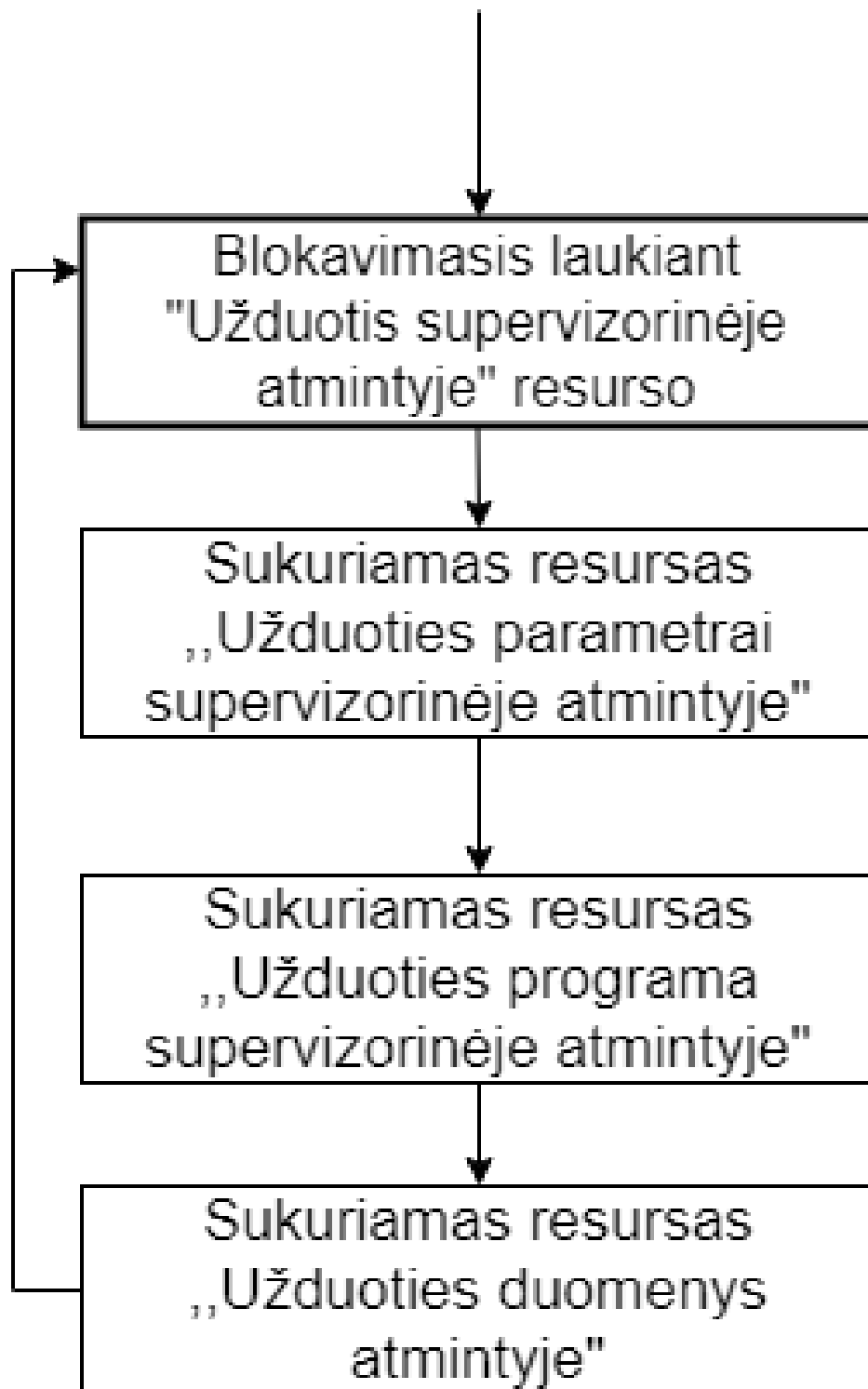
užduoties nuskaitymo iš įvedimo srauto procesas.



8 pav. Read procesas

7.3. JCL

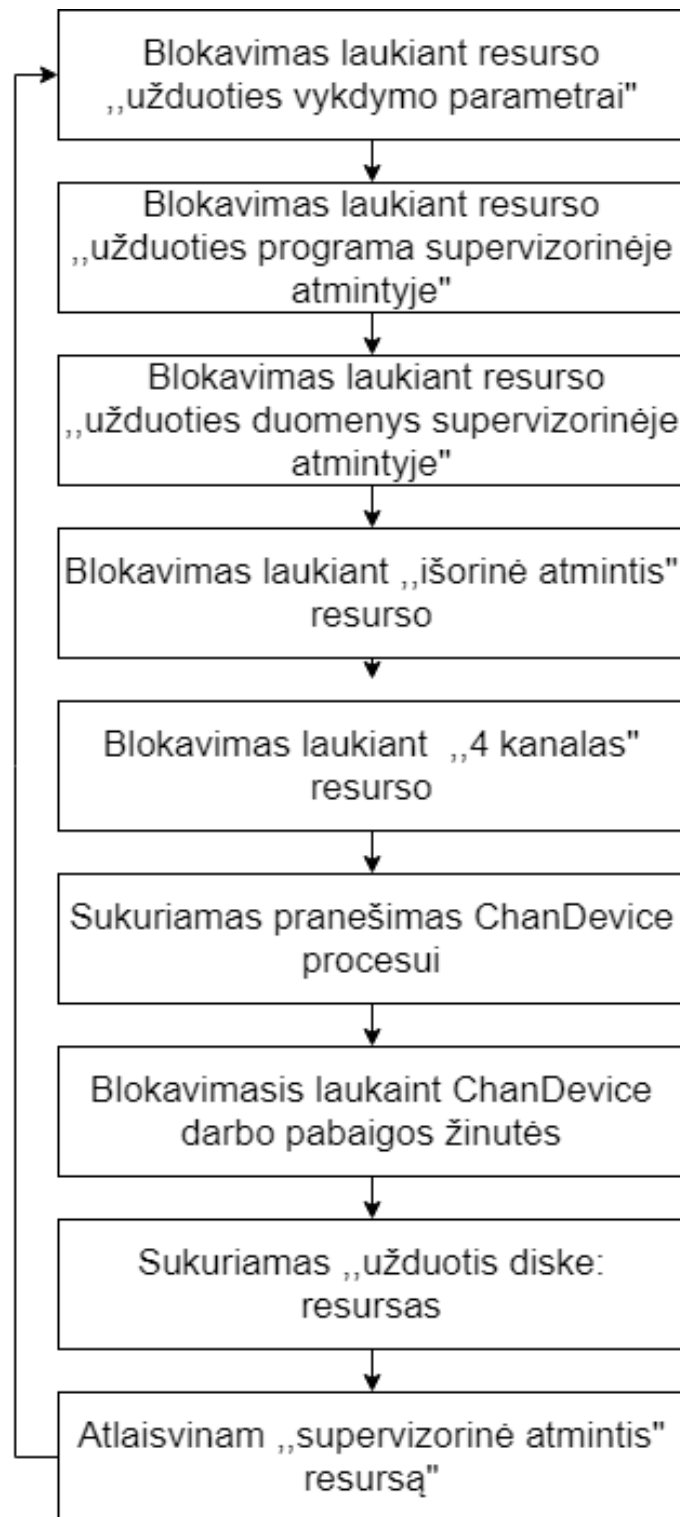
JCL paskirtis - interpretuoti užduoties tekstą, išskiriant duomenis ir paraketrus bei kuriant atitinkamus resursus.



9 pav. JLC

7.4. JobToDisk

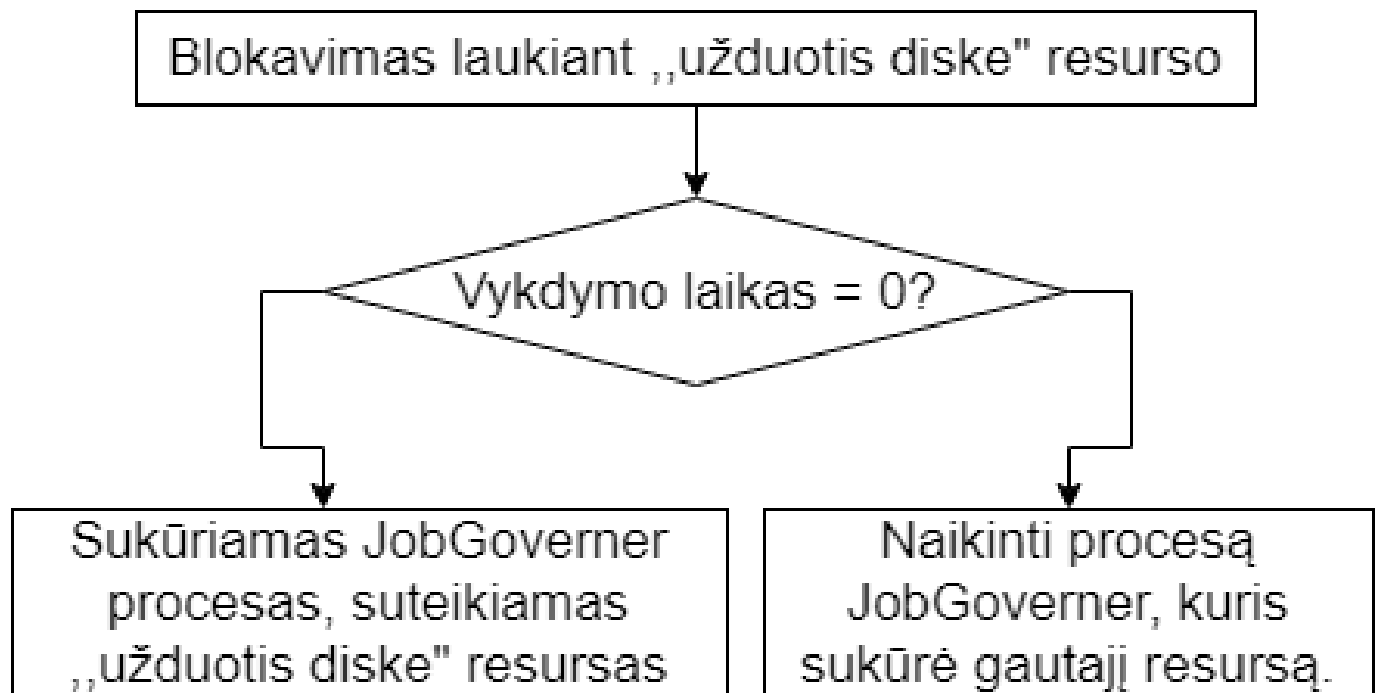
JobToDisk - procesas skirtas patarlpinti užduotį iš supervizorinės atminties į išorinę atmintį.



10 pav. JobToDisk procesas

7.5. MainProc

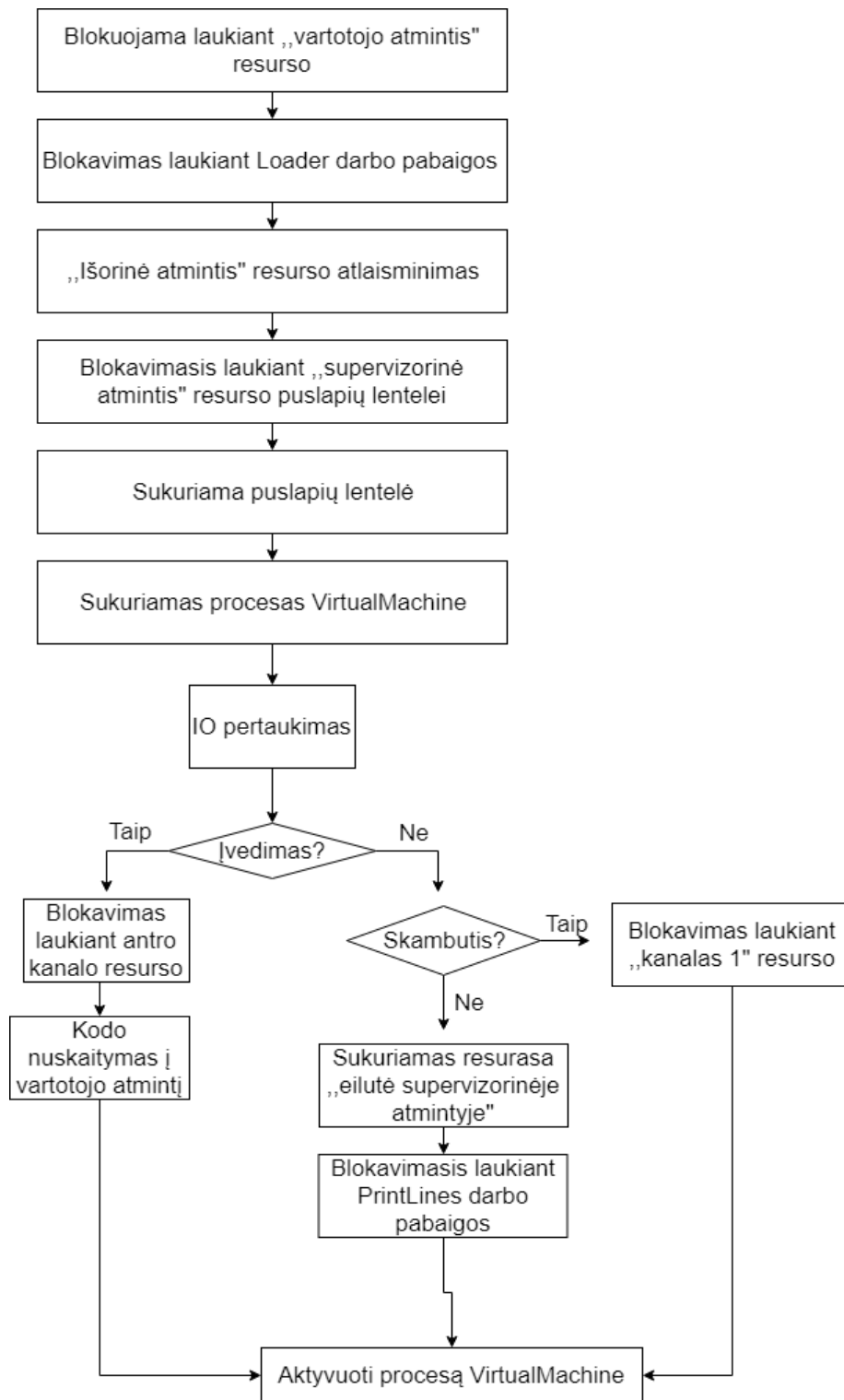
MainProc yra procesas atsakingas už užduočių vykdymo paruošimą ir sunaikinimą.



11 pav. MainProc

7.6. JobGovernor

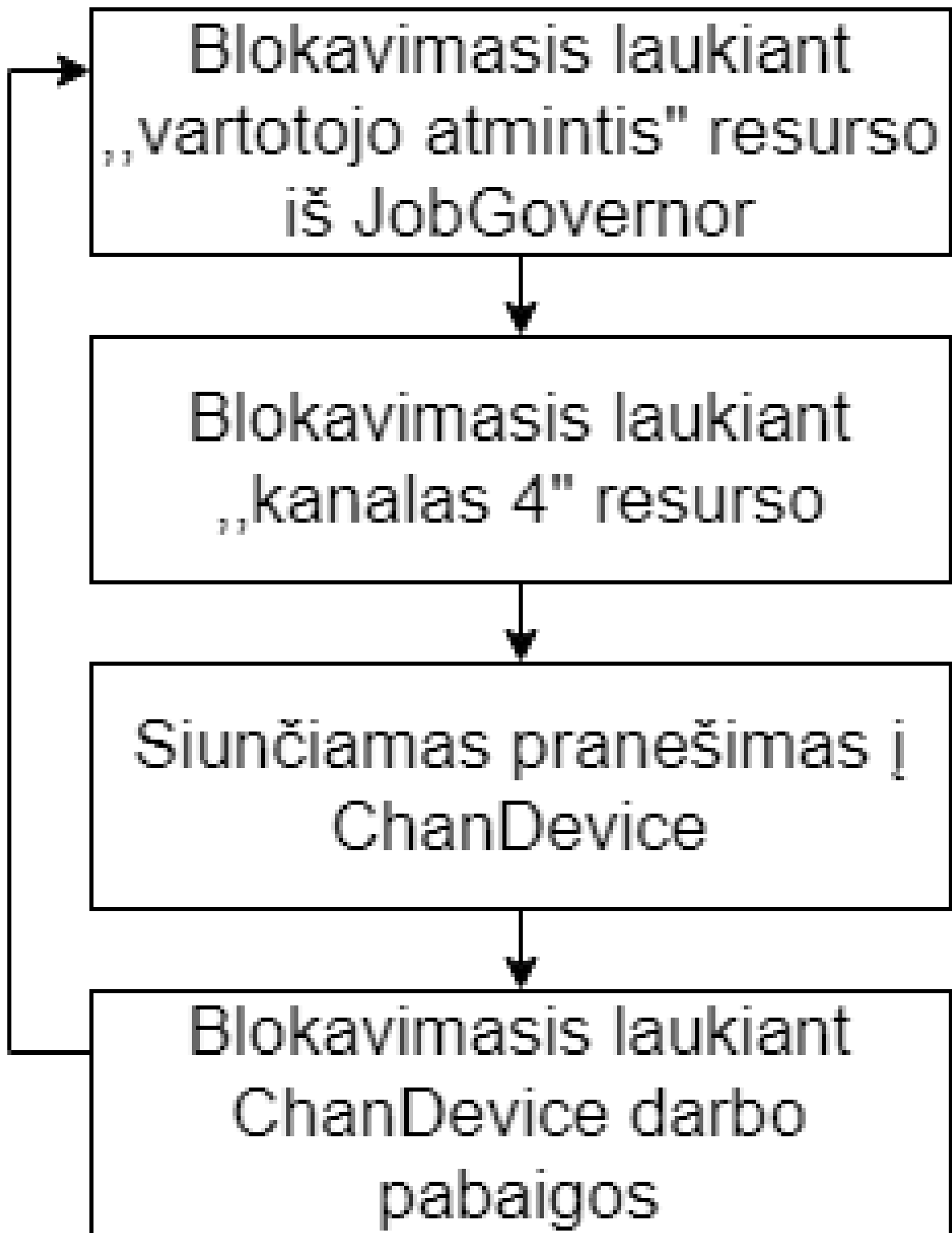
JobGovernor procesas atsakingas už virtualios mašinos proceso kūrimą ir tvarkantis jos darbą



12 pav. JobGovernor

7.7. SwapBack

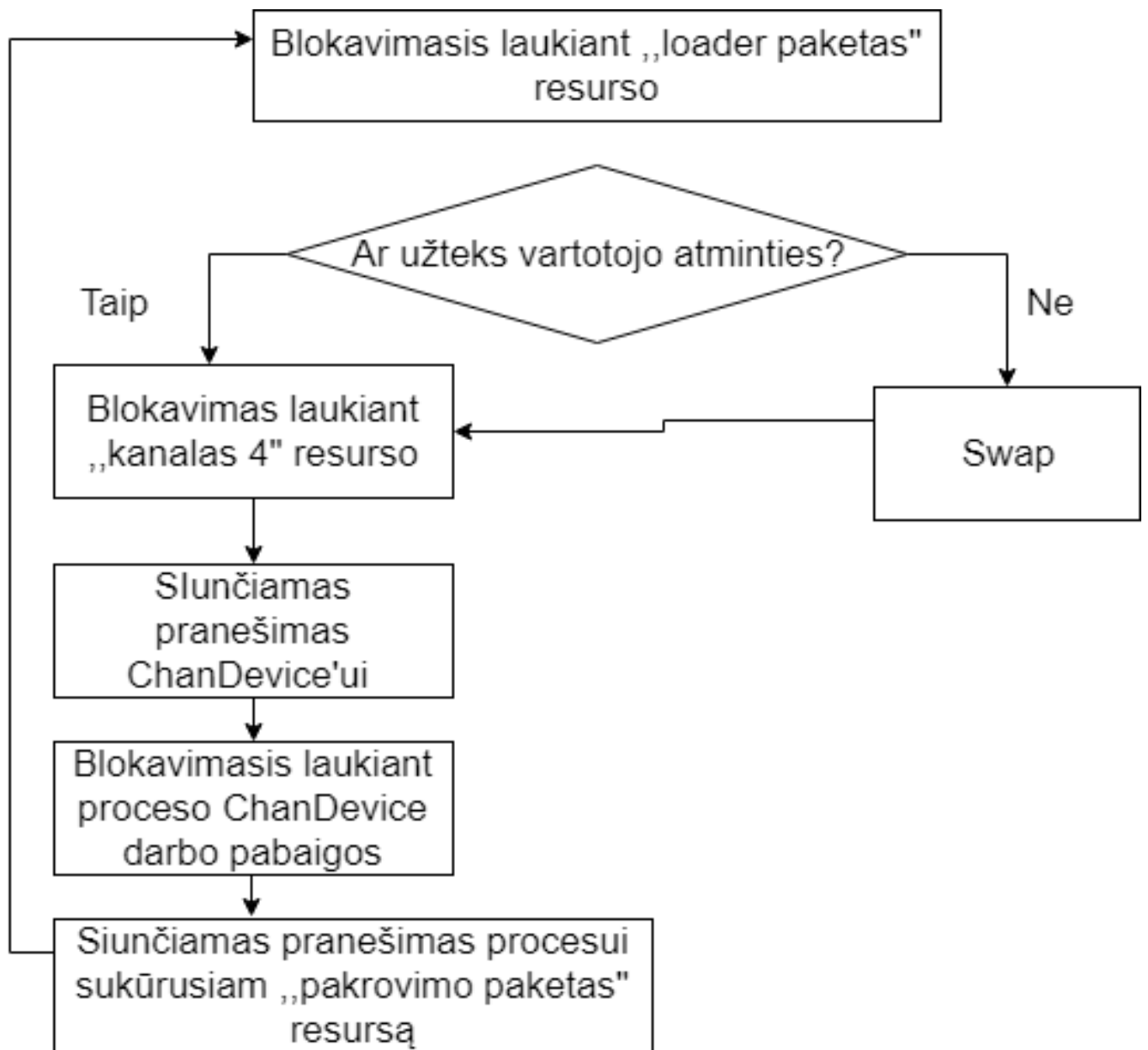
SwapBack - procesas gražinantis duomenis iš HDD swap bloko į vartotojo atmintį



13 pav. SwapBack procesas

7.8. Loader

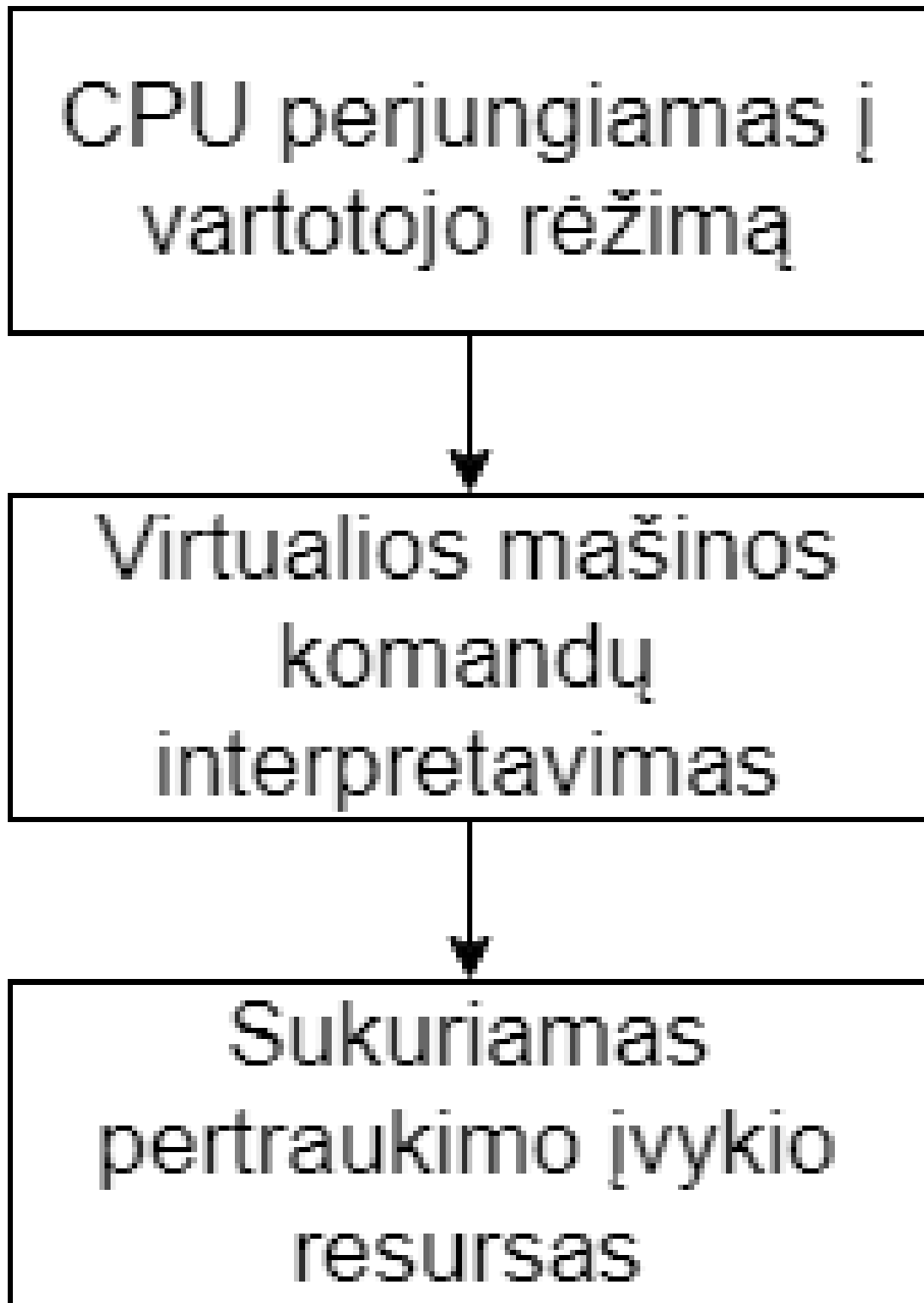
Loader - iš išorinės atminties programos tekstą perkelia į vartotojo atmintį



14 pav. Loader procesas

7.9. VirtualMachine

VirtualMachine - atsakingas už vartotojiškų procesų vykdymą.



15 pav. VirtualMachine procesas

7.10. Interrupt

Interrupt - atsakingas už VM pertraukimų interpretavimą.

Blokavimasis laukiant
resurso „pertraukimas“

Pertraukimo
indentifikavimas

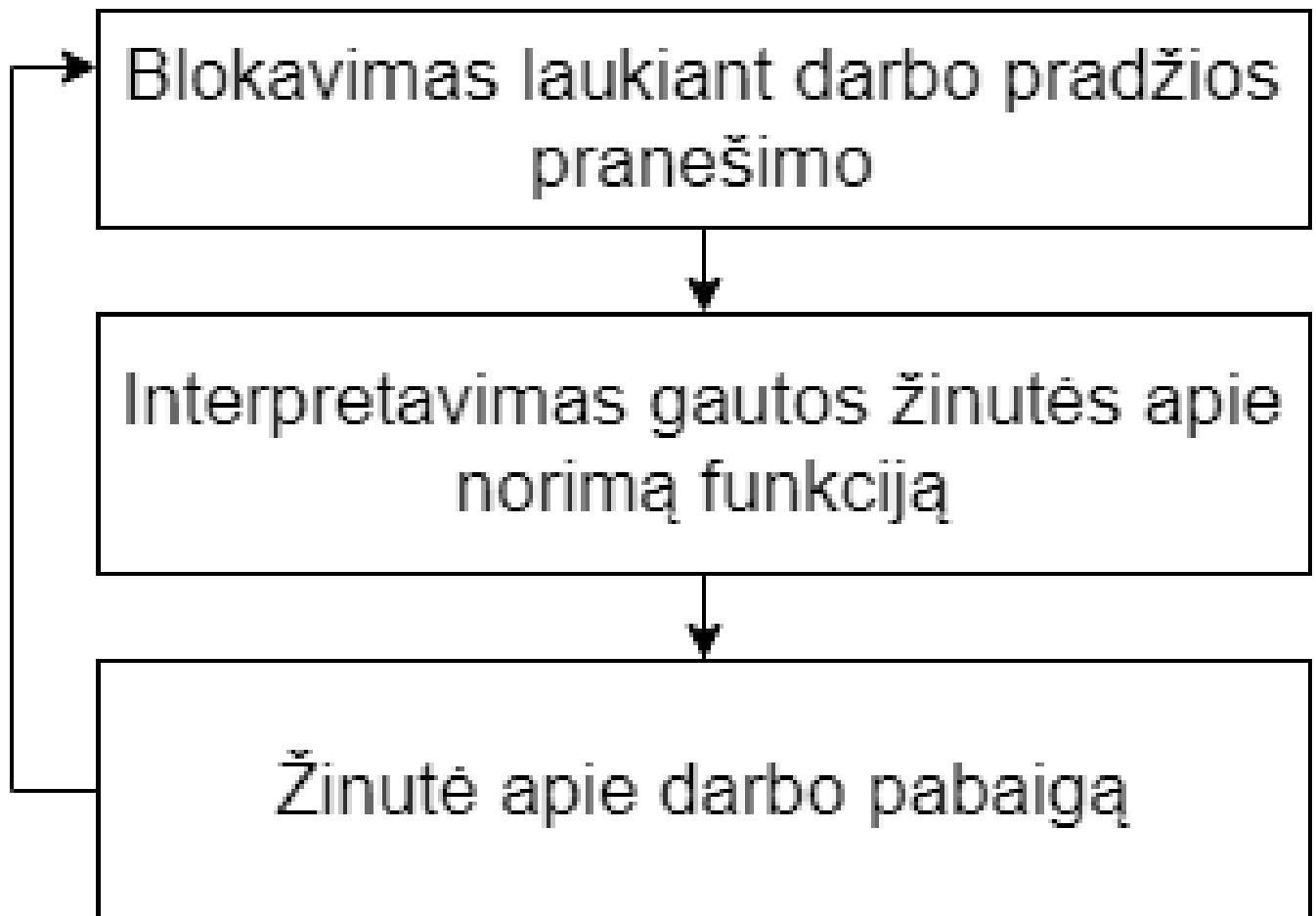
Pertrauktorios VM tėvinio
proceso JobGovernor
indentifikavimas

Pranešimo apie
pertraukimą procesui
JobGovernor sukūrimas

16 pav. Interrupt procesas

7.11. ChanDevice

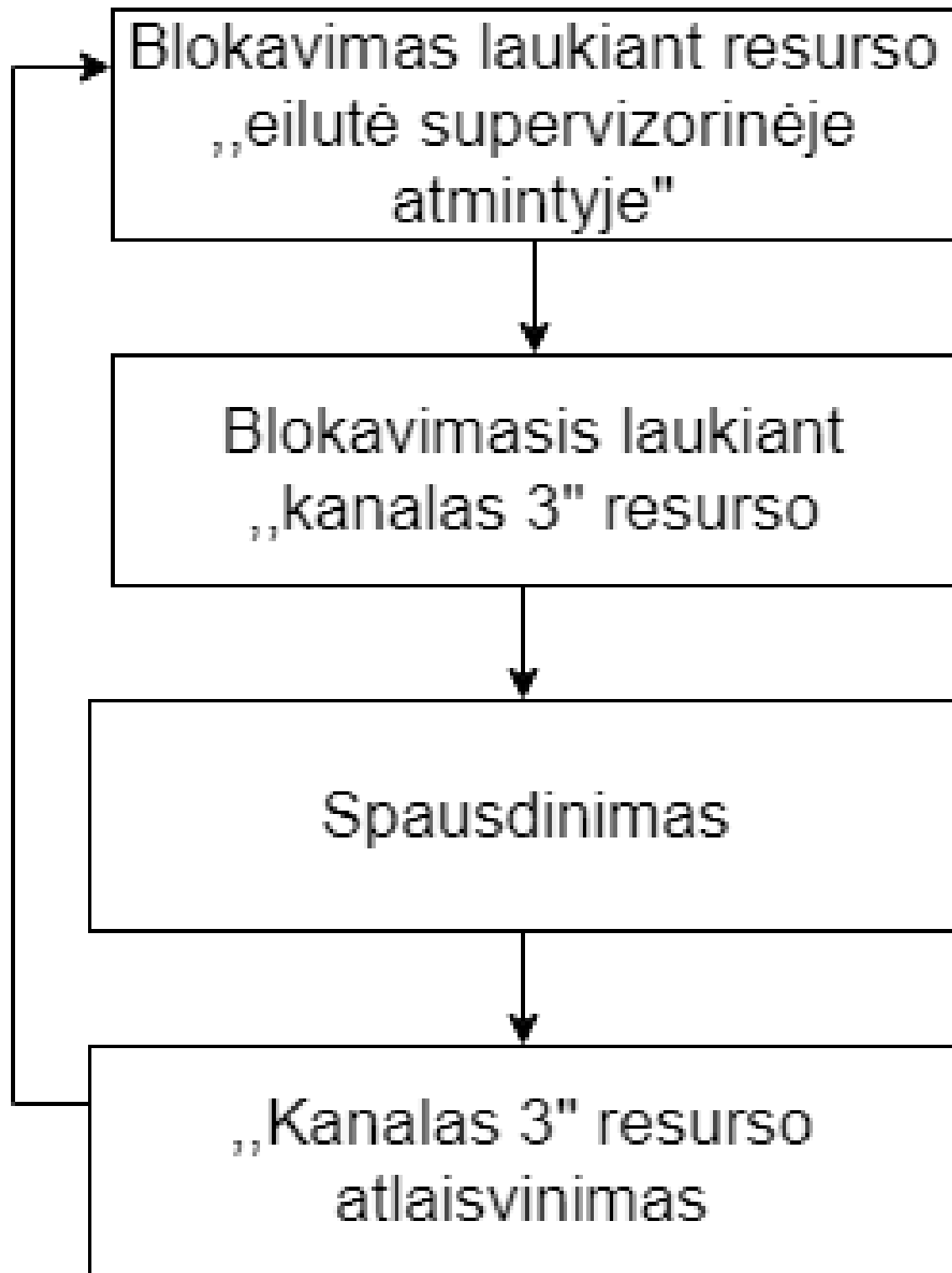
ChanDevice - kanalų įrenginio procesas skirtas valdyti 4 kanalus.



17 pav. ChanDevice procesas

7.12. PrintLine(Printeris)

- išvedimo įrenginiui siunčiama eilutė kurią reikės atspausdinti



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 būseną

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 būseną proceso metu

- mode: byte // supervizorinis/vartotojo darbo režimas
- 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)
- c: boolean // registras, atitinkantis VM loginį triggerį
- 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