VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

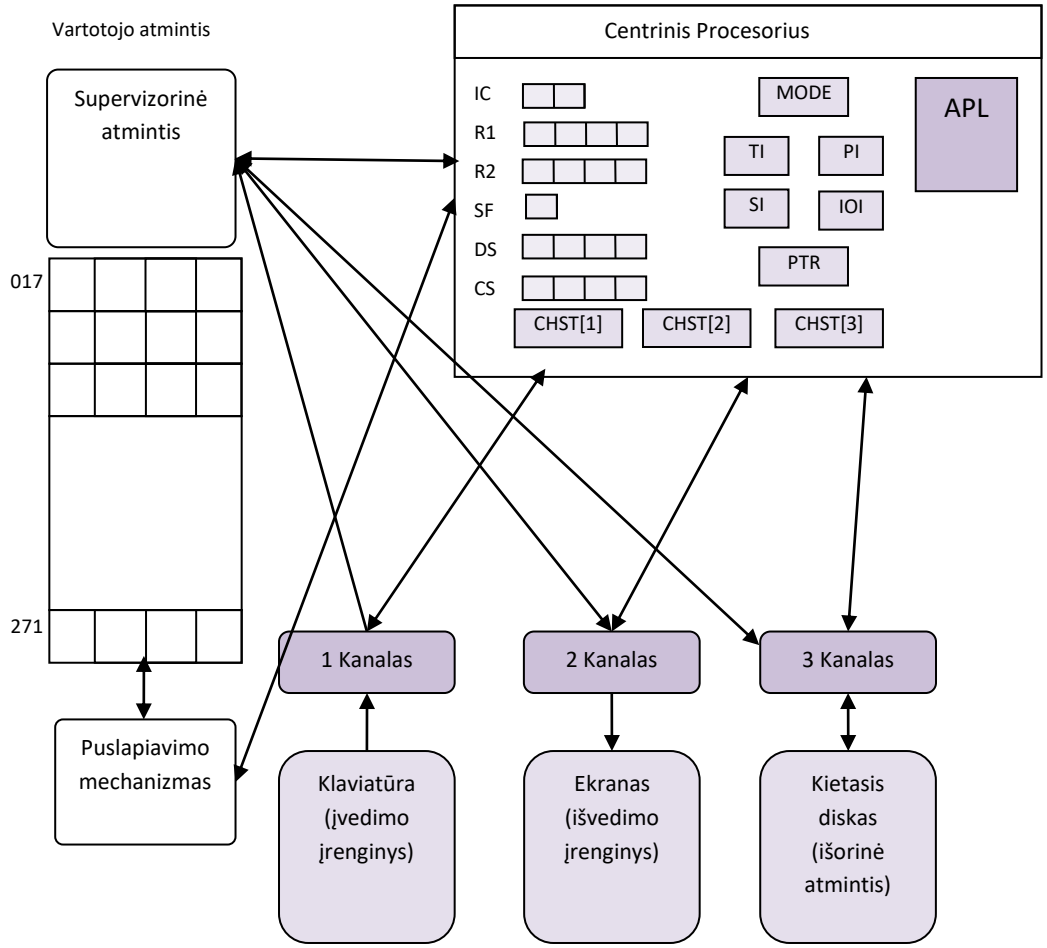
VIRTUALIOS IR REALIOS MAŠINOS MODELIS

Anastasija Kiseliova
Evelina Bujytė
MI, 3 kursas

2017 m.
VILNIUS

TURINYS

1. REALI MAŠINA



1.1. Techninės įrangos komponentai sudarantys realią mašiną:

- Centrinis procesorius
- Vartojo atmintis
- Supervizorinė atmintis
- Išorinė atmintis
- Duomenų perdavimo kanalai
- Įvedimo įrenginys - klaviatūra
- Išvedimo įrenginys - ekranas
- Puslapiavimo mechanizmas

1.2. Centrinis procesorius

Procesoriaus paskirtis yra skaityti komandas iš atminties ir jas interpretuoti.

Procesorius gali būti naudojamas dviem rėžimais:

- Supervizoriaus
- Vartotojo

Supervizoriaus rėžime procesorius vykdo komandas, kurios yra atsakingos už operacinės sistemos funkcionavimą, bet ne už vartotojo užduočių programas. Naudojantis

supervizoriaus rėžimu darbas su ALP yra atliekamas naudojantis supervizoriaus atmintimi. Procesoriaus persijungia į supervizoriaus režimą pertraukimais arba sisteminiais kreipiniais.

1.3. Realios mašinos procesorius turi tokius registrus:

- R1, R2– 4 baitų bendrosios paskirties registrai
- IC – 2 baitų komandų skaitiklis
- C -– 1 baito loginis registras, gali įgyti reikšmes TRUE arba FALSE.
- SF – 1 baito požymių registras. CF OF XX XXXZF – carry flag, overflow flag, tušti ir zero flag.
- PTR – 4 baitų puslapių lentelės registras (aktyvus puslapių lentelės bloko numeris)
- MODE – 1 baito registras, skirtas nustatyti mašinos režimą (supervizoriniu arba vartotojo)
- TI – 2 baitų taimerio registras, kiekvienas atliktas komandos žingsnis padidina jį laiko vienetu
- PI,SI – 2 baitų programinių ir supervizorinių pertraukimų registrai
- IOI - 2 baitų įvedimo ir išvedimo pertraukimų registras
- CHST[1] ... CHST[3] – kanalų būsenos registrai.

1.4. Atmintys:

Vartotojo ir supervizorinė atmintys dalijasi realios mašinos atmintį.

- Vartotojo Atmintis-pagrindinė atmintis, naudojama vartotojo; joje saugomos užduotys rezultatai, puslapio lentelės ir t.t.
- Supervizoriaus atmintis -ją naudoja APL.
- Išorinė atmintis-papildoma atmintis, šiuo atvėju kietasisi diskas.

1.5. Išvedimo ir įvedimo įrenginiai

- Klaviatūra - įvedimo įrenginys; naudojamas nuskaityti įvestas komandas
- Ekranas- išvedimo įrenginys; naudojamas informacijos išvedimui ir atvaizdavimui ekrane

1.6. Puslapiavimo mechanizmas

Puslapiavimo mechanizmas yra naudojamas virtualaus adreso atvaizdavimui į realų adresą atmintyje. Virtualiai mašinai išskiriama 16 atminties blokų. Kiekvienas lentelės puslapis (1 blokas) užpildomas realiais adresais. Naudojamas registras PTR(4 baitų a_0,a_1,a_2,a_3), kuriame laikomi puslapių lentelės bloko realūs adresai. Naudojant PTR registro baitus a_2 ir a_3, surandamas blokas. Pagal virtualaus adreso x_1 ir x_2 yra nustatomas puslapis. Bloko adresas kur yra x_1 randamas pagal formulę [16*(16*a_2 a_3)+x_1], o prie bloko adreso pridėjus x_2 gauname virtualų adresą: 16*[16*(16*a_2 a_3)+x_1]x_2.

1.7. Duomenų perdavimo kanalai

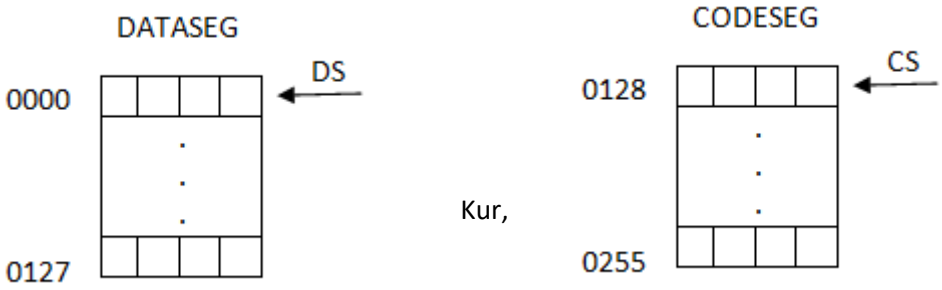
Skirti atminties ir įvedimo/išvedimo valdymui. Yra trys kanalai kiekvienas atsakingas už atitinkamus dalykus. Kanale vyksta tik rašymas arba skaitymas ir baigęs darbą jis informuoja centrinį procesorių ir sukelia pertraukimą padidindamas IOI registro reikšmę. Kanalų būsenos yra saugomos CHST[1] ... CHST[3] registruose.

2. VIRTUALI MAŠINA

2.1. Pagrindinė operacinės sistemos funkcija – paslėpti visą jos sudėtingumą ir duoti programuotojui instrukcijų, su kuriomis jis galėtų dirbti, sąrašą. Todėl vienas iš operacinės sistemos tikslų yra paslėpti realią mašiną ir pateikti mums virtualią. Virtuali mašina - tai tarsi virtuali realios mašinos kopija, su kuria dirba vartotojas. VM yra lyg trapininkas tarp konkrečios mašinos ir jai taikomos programinės įrangos.

2.2. Modeliuojamos virtualios mašinos loginių komponentų aprašymas.

2.2.1 Virtualios mašinos atmintis susideda iš 16 blokų po 16 žodžių (iš viso 256 žodžiai), kurie yra po 4 B (32 bit). Atmintis turi 2 lygius segmentus: duomenų (DATASEG) – 8 blokai (128 žodžių), kodo (CODESEG) – 8 blokai (128 žodžių), į kuriuos bus įkeliamos atitinkamos programos dalys.
Kodo segmente esančios programos paskutinė komanda turi būti HALT.
Virtualios mašinos atmintis vizualiai atrodo taip:



DS – registras, kurio reikšmė yra rodyklė į duomenų segmentą atmintyje, o CS – registras, kurio reikšmė yra rodyklė į kodo segmentą atmintyje.
Puslapiavimo mechanizmas yra naudojamas virtualaus adreso atvaizdavimui į realų adresą atmintyje. Virtualiai mašinai išskiriama 16 atminties blokų. Kiekvienas lentelės puslapis (1 blokas) užpildomas realiais adresais. Naudojamas registras PTR(4 baitų a_0,a_1,a_2,a_3), kuriame laikomi puslapių lentelės bloko realus adresai. Naudojant PTR registro baitus a_2 ir a_3 surandamas blokas. Pagal virtualaus adreso x_1 ir x_2 yra nustatomas puslapis. Bloko adresas kur yra x_1 randamas pagal formulę $[16*(16*a_2\ a_3)+x_1]$, o prie bloko adreso pridėjus x_2 gauname virtualų adresą: $16*[16*(16*a_2\ a_3)+x_1]x_2$.

Virtualiais adresais operuoja virtuali mašina, o realiais adresais – reali mašina. Ryšiai tarp realaus ir virtualaus adreso nusakomi per puslapiavimo mechanizmą.

Informacija apie kiekvieną procesą yra saugoma operacinės sistemos lentelėje, kuri yra realizuojama struktūrų list‘u - kiekviena struktūra vienam, dabar egzistuojančiam, procesui.

puslapis \ Žodis	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	GD20	LR20	COPY	LR22	ADDi	SR40	PD80	PD40	HALT							
1																
2	1234		5678													
3																
4	6912															
5																
6																
7																
8	Susk	aici	uota													
9																
A																
B																
C																
D																
E																
F																

2.2.2 Virtualios mašinos procesorius.

Virtualios mašinos procesoriaus registrai:

- Komandų skaitliukas:
  a. **IC** – 2 baitų virtualios mašinos programos skaitiklis
- Bendrosios paskirties registrai:
  a. **R1** – bendrosios paskirties registras
  b. **R2** – bendrosios paskirties registras
- Požymių registrai:
  a. **SF** – 1 baito loginis registras, požymius formuoja aritmetinės, o į juos reaguoja sąlyginio valdymo perdavimo komandos. Įgyjamos reikšmės: 0 – jeigu daugiau, 1 – jeigu lygu, 2 – jeigu mažiau.
- Segmentų registrai:
  a. **DS** – Data segment – rodyklė į duomenų segmentą atmintyje.
  b. **CS** – Code segment – rodyklė į kodo segmentą atmintyje.

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 x_1x_2 persiunčia į registrą R:
      LR x_1x_2 =) R:=[x_1x_2];
   b. **SR** – Save Register – iš registro R persiunčia į atminties baitą x_1x_2:
      SR x_1x_2 =) [x_1x_2]:=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 x_1x_2 atminties baite, rezultatas patalpinamas registre R:
   AD x_1x_2 =) r1:=r1+[x_1x_2];
b. **SB** – atimtis – iš esamos registro R reikšmės atimama reikšmė esanti x_1x_2 atminties baite, rezultatas patalpinamas registre R:
   SB x_1x_2 =) r1:=r1-[x_1x_2];
c. **CR** – palyginimas – esamą registro R reikšmė yra lyginama su reikšme esančią x_1x_2 atminties baite, rezultatas patalpinamas registre C:
   CR x_1x_2 =)
   if r1>[x_1x_2] then cf:=0, zf:=0;
   if r1=[x_1x_2] then zf:=1;
   if r1<[x_1x_2] then cf:=1;
d. **MU x1x2** – daugyba, r1 =r1* [x1x2].
e. **DI x1x2** – dalyba, r1=r1/[x1x2] , r2=r1%[x1x2].

4. Valdymo perdavimo:
   a. **JU** – besąlyginio valdymo perdavimas – valdymas perduodamas adresu $16*x_1+x_2$:
      JU x_1x_2 =) IC:=$16*x_1+x_2$;
   b. **JG** – sąlyginio valdymo perdavimas (jeigu daugiau) – valdymas perduodamas jeigu C=0, valdymas perduodamas adresu $16*x_1+x_2$:
      JG x_1x_2 =) If C=0 then IC:= $16*x_1+x_2$;
   c. **JE** – sąlyginio valdymo perdavimas (jeigu lygu) – valdymas perduodamas jeigu C=1, valdymas perduodamas adresu $16*x_1+x_2$:
      JE x_1x_2 =) If C=1 then IC:= $16*x_1+x_2$;
   d. **JL** – sąlyginio valdymo perdavimas (jeigu mažiau) – valdymas perduodamas jeigu C=2, valdymas perduodamas adresu $16*x_1+x_2$:
      JL x_1x_2 =) If C=2 then IC:= $16*x_1+x_2$;
5. Darbo su bendra atminties sritimi (prieinama visoms vartotojo programoms; komandos leidžia į ją rašyti ir skaityti; sritis apsaugoma semaforais):
   a. **SM** – registro R įrašymas į bendrąją atmintį:
      SM x_1x_2 =)[$16*[16*(16*a_2\ a_3)+x_1]x_2$] :=R (pagal puslapiavimo mechanizmą);
   b. **LM** – iš bendrosios atminties įrašomas žodis į registrą R:
      LM x_1x_2 =) R:= [$16*[16*(16*a_2\ a_3)+x_1]x_2$] (pagal puslapiavimo mechanizmą);
6. Programos pabaigos:
   a. **HALT** – programos pabaigos komanda.
7. Įvedimo/Išvedimo:
   a. **GD** – įvedimas – iš įvedimo srauto paima 1 žodžio srautą ir jį įveda į atmintį pradedant atminties baitu $16*x_1+x_2$:
      GD x_1x_2
   b. **PD** – išvedimas – iš atminties, pradedant atminties baitu $16*x_1+x_2$ paima 1 žodžio srautą ir jį išveda į ekraną:
      PD x_1x_2
8. Loginės:
   a. **AND** – r1 := r1 and r2
   b. **XOR** – r1 := r1 xor r2
   c. **OR -** r1 := r1 or r2.
   d. **NOT -** r1 := not r1.

9. Operavimo failais:
    a. **FC -** uždaromas failas, kurio handleris yra r1.
    b. **FO x_1x_2** – atidaromas [x1x2[pavadinimo failas. Handleris įrašomas į r1.
    c. **FR x_1x_2 –** r1 – handleris, r2 – adresas, iš kur skaitome. 16*x1+x2 – virtualios atminties vieta, į kurią įrašysime.
    d. **FW x_1x_2 –** handleris, r2 – adresas, iš kur skaitome. 16*x1+x2 – virtualios atminties vieta, iš kurios rašysime į failą.
    e. **FD –** ištrinamas failas, kurio handleris yra r1.

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

VM duomenis skaito iš išorinės atminties (realizuotos failu kietajame diske), o rezultatą išveda į kompiuterio ekraną. Įvedimą/išvedimą kontroliuoja kanalų įrenginys.

2.4. Virtualios mašinos interpretuojamojo ar kompiliuojamo vykdomojo failo išeities teksto formatas. **Pavyzdžiui, kaip išskiriamas duomenų segmentas, kodo segmentas, kaip aprašomi duomenys ir t.t.)**

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).

Duomenų segmento apraše galimi tokie atvejai:
DW - Išskiriamas vienas tuščias žodis skaitinei reikšmei.
DW X - Išskiriamas vienas žodis ir į jį talpinama nurodyta skaitinė reikšmė.
DB ssss - Išskiriamas vienas žodis ir į jį talpinami keturi nurodyti simboliai.
DB nnnn - Tai rezervuota simbolinė konstanta, reiškianti \n.

Programa apskaičiuoja reiškinio „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 rėžimu, tam naudojama iniciavimo operacija StarIO – kuria nustatomi kanalai, jų panaudojimas ir tikrinamas užimtumas.

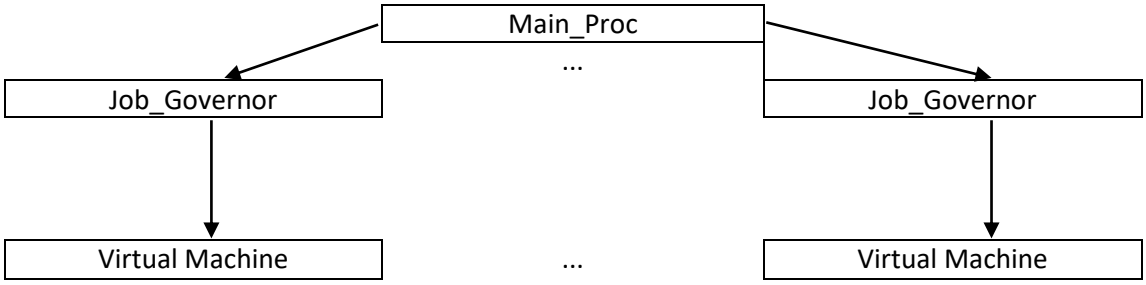
Norint iš supervizorinio rėžimo į vartotojo rėžimą darbo pratęsimui virtualioje mašinoje reikalinga pakrauti būsena tam naudojama operacija Slave(plr,c,r,ic) – kur registrų panaudojimas sutampa su realios mašinos.

3. OPERACINĖS SISTEMOS MODELIS

Dabartinės operacinės sistemos yra multiprograminės t.y. gali atlikti keletą skirtingų veiksmų vienu metu. Nors ir atrodo, kad operacinė sistema vykdo visus procesus lygiagrečiai, nes viskas atliekama užtrunkant kažkurią sekundės dalį. Dėl greito veikimo ir atsiranda butaforija, kad procesai vyksta lygiagrečiai. Kadangi taip tik atrodo naudojant, o ne kuriant OS, dokumente bandysime išsiaiškinti tikslų OS veikimą ir jį struktūriškai aprašyti.

Procesas – tai programa, turinti savo registrų reikšmes ir kintamuosius. Kiekvienas procesas turi savo virtualų procesorių. Nors galėtume sakyti, kad procesas ir programa yra adekvatus veiksmai. Visgi yra vienas esminis skirtumas: procesas, tai programa esanti veikimo būsenoje.

Kiekvienai vykdomai užduočiai turi būti sukurtas specialus procesas. „*Main_Proc*“ sukuria užduotį prižiūrintį procesą – „*Job_Governor*“. Šių procesų skaičius priklauso nuo atliekamų užduočių skaičiaus:



Turime pastovius sisteminius procesus, kurie egzistuoja visą sistemos darbo laiką, sukuriami sistemos darbo pradžioje, o pabaigoje sunaikinami. Galioja vienintelė išimtis – „*Job_Governor*“ procesui. Jis kuriamas prieš užduoties vykdymą, o naikinamas baigus vykdyti programą.

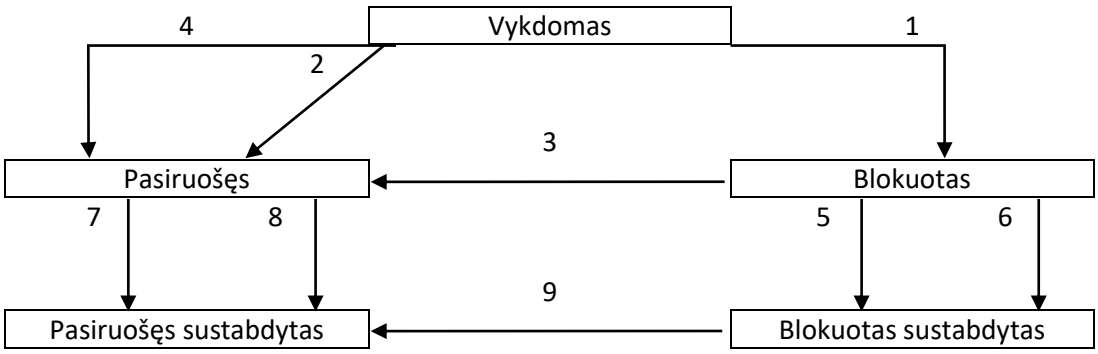
„*Virtual Macine*“ – VM vykdymo procesas. Šio proceso galimos būsenos: *pasiruošęs, vykdomas, blokuotas, sustabdytas*.

Procesų būsenos

a) **Pasiruošęs:** vienintelis trūkstamas resursas yra procesorius.
b) **Vykdomas:** procesas gali gauti procesorių tik tada, kai jam netrūksta jokio kito resurso. Procesas gavęs procesorius tampa vykdomu. Procesas, esantis šioje būsenoje, turi procesorių, kol sistemoje neįvyksta pertraukimas arba einamasis procesas nepaprašo kokio nors resurso (priklausomai nuo situacijos tai galėtų būti: prašymas įvedimo iš klaviatūros).
c) **Blokuotas:** procesas blokuojasi priverstinai. Tačiau, jei procesas nereikalauja jokio resurso, iš jo gali būti atimamas procesorius, pavyzdžiui dėl per ilgo darbo.

   Galima situacija, kad tam tikram procesui negalima leisti gauti procesorių, nors jis ir pasiruošęs. Tokį procesą vadinsime sustabdytu.
d) **Sustabdytas:** kito proceso sustabdytas procesas.

Kaip procesas patenka ir išeina iš tam tikros būsenos matysime žemiau pavaizduotoje schemoje.

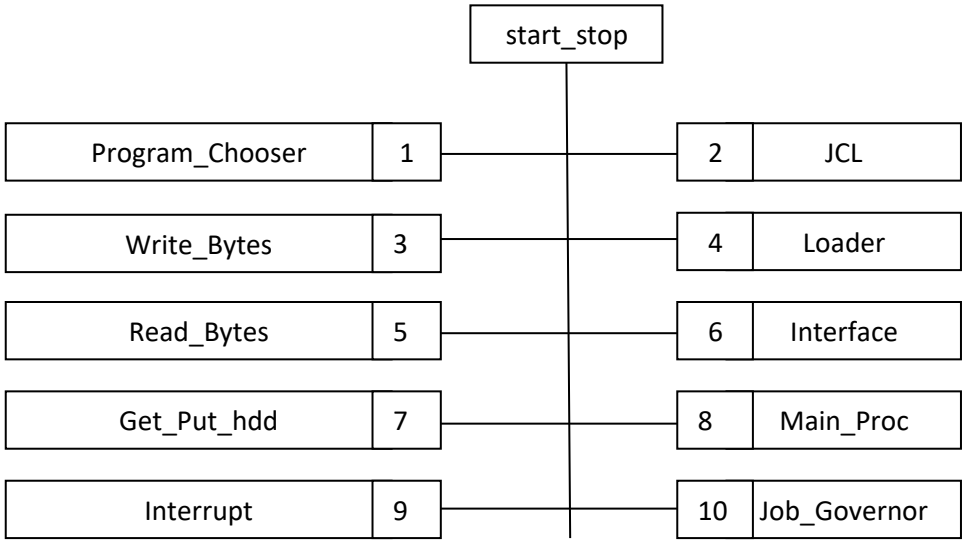


Yra galimi devyni skirtingi perėjimo būdai:

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 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 nuima būseną 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.

3.1. Procesai

Mūsų OS modelyje pirmasis procesas „*start_stop*“ bus atsakingas už darbo pradžią ir pabaigą. Proceso tikslas, sukurti statinius MOS procesus ir resursus.



1. „*Read_from_Interface*“ – atsakingas už betarpišką užduoties įvedimą.
2. „*JCL*“ – analizuoja užduoties struktūrą – užduoties atpažinimas.

3. „Write_Bytes“ – išveda tam tikrą baitų kiekį į ekraną.
4. „Loader“ – paruoštas vykdymui užduotis perkelia į vartotojo atmintį.
5. „Read_Bytes“ – failai nuskaitomi iš išorinės atminties.
6. „Chan3_Device“ – darbo su tiesioginio priėjimo prie išorinių įrenginių procesas. Tai trečiojo kanalo aptarnavimo procesas.
7. „Get_put_hdd“ – failai skaitomi/ įrašomi į išorinę atmintį.
8. „Main_Proc“ – užduočių atlikimo valdymo sistemos procesas.
9. „Interrupt“ – dirbant vartotojo režimu ir vykdant programą gali kilti pertraukimai. Jų signalai turi būti transformuojami į resursus. Tam skirtas „Interrupt“ procesas. Detalus procesų funkcijų nagrinėjimas
10. „Job_Governor“ - jų bus daug, jie tvarkys VM.

1.1.1 Procesų prioritetai

Prioritetų numeriai bus nuo 1 iki 100, didesnį numerį skirdami sisteminiams , o ne vartotojo, procesams. Numeris mažės įvykus procesui.

ID	Pavadinimas	Prioritetas
1	Start_Stop	100
2	Main_Proc	99
3	Job_Governor	98
4	Interrupt	97
5	Loader	96
6	Get_put_hdd	90
7	Program_Chooser	85
8	Read_Bytes	80
9	Write_Bytes	75
10	JCL	65
11	Virtual machine	50

Planuotojas pagal sudarytą prioritetų lentelę nustato kiekvieno proceso prioritetą ir atsižvelgdamas į juos renkasi iš pasiruošusių procesų sąrašo sekantį.

```
public class planuotojas {

    ArrayList<Procesas> pasiruose;
    Procesas vykdomas;
    ArrayLisy<Procesas> blokuoti;
    ArrayList<Procesas> sustabdyti;

    Public void surikiuotiIEile();

    Public void stabdytiProcesą(Procesas);

    Public void paleistiProcesą();

    Public void blokuotiProcesą(Procesas);

    Public void pasiruosesProcesas(Procesas);

}
```

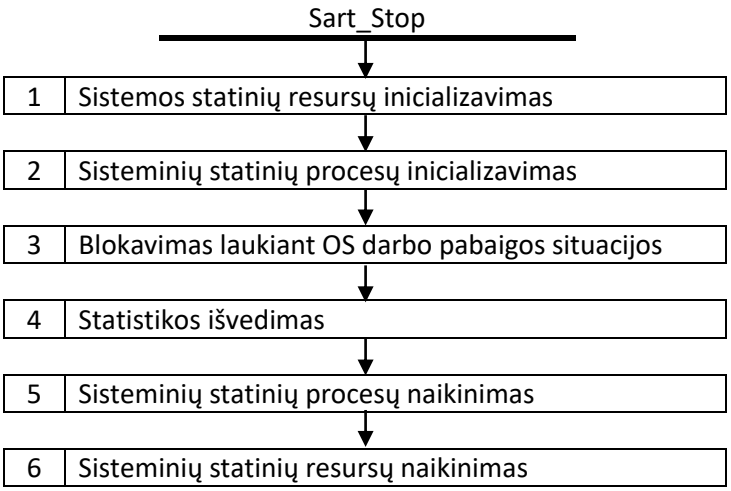
Resursai

Pavadinimas	**Laukia**	**Sukuria/atlaisvina**
Main_Proc	Programa ok	[sukuria Job_Governor]
Program_Chooser	*Iš vartotojo sąsajos* *3-io kanalo* *Supervizorinė atmintis*	Programos tikrinimas Supervizorinė atmintis 3-iojo kanalo
JCL	Programos tikrinimas	*Programa ok;* *Programa klaidinga;*
Job_Governor	Vartotojo atmintis Supervizorinė atmintis Loader pabaiga Interrupt Read_bytes pabaiga Write_bytes pabaiga Get_put_HDD pabaiga	Vartotojo atmintis; VM atmintis Žodis rašymui Žodis skaitymui Darbas su failais Užduotis ok (fiktyvus) Os pabaiga [sukuria Virtual Machine ir sukuria jos lentelę] [gali sustabdyti Virtual Machine]
Loader	Programos tikrinimas	Programa ok Programa klaidinga
Virtual Machine	-	Interrupt
Program_Chooser	Iš vartotojo sąsajos 3-io kanalo Supervizorinė atmintis	Programos tikrinimas Supervizorinė atmintis
Write_Bytes	Žodis rašymui 2-ojo kanalo	[atlaisvina 2-ąjį kanalą] 2-ojo kanalo
Read_Bytes	Žodis skaitymui 1-ojo kanalo	[atlaisvina 1-ąjį kanalą] 1-ojo kanalo
Interrupt	Interrupt call	Pertraukimo tipas
Get_put_hdd	Darbas su failais Išorinė atmintis 3-iojo kanalo	3-ią kanalą
Start_Stop	Os pabaiga	
Interface		Os pabaiga Iš vartotojo sąsajos

Pavadinimas	**Sukuria/atlaisvina**	**Laukia**	**S ar D?**
OS pabaiga	Interface	Start_Stop	Dinaminis
Supervizorinė atmintis	*Start_Stop* *Program_Chooser* *Job_Governor* Get_Put_hdd	Job_Governor *Program_Chooser* Get_Put_hdd	Statinis
Vartotojo atmintis	*Start_Stop* *Job_Governor* 2-ojo kanalo	*Job_Governor*	*Statinis*
1-ojo kanalo	Start_Stop Read_Bytes	Read_Bytes	*Statinis*
2-ojo kanalo	Start_Stop Write_Bytes	Write_Bytes	*Statinis*

3-ojo kanalo	Start_Stop Get_Put_hdd Program_Chooser	Get_Put_hdd Program_Chooser	*Statinis*
Programos tikrinimas	Program_Chooser	JCL	Dinaminis
Programa klaidinga	JCL	-	Dinaminis
Programa ok	JCL	Main_Proc	Dinaminis
VM atmintis	Job_Governor	Loader	Dinaminis
Loader pabaiga	Loader	Job Governor	Dinaminis
Iš vartotojo sąsajos	Interface	Program_Chooser	Dinaminis
Žodis rašymui	Job_Governor	Write_Bytes	Dinaminis
Žodis skaitymui	Job_Governor	Read_Bytes	Dinaminis
Darbas su failais	Job_Governor	Get_Put_hdd	Dinaminis
Interrupt	Start_Stop Virtual_Machine	Job_Governor	Statinis

3.1.1 Procesas „*Start_Stop*“



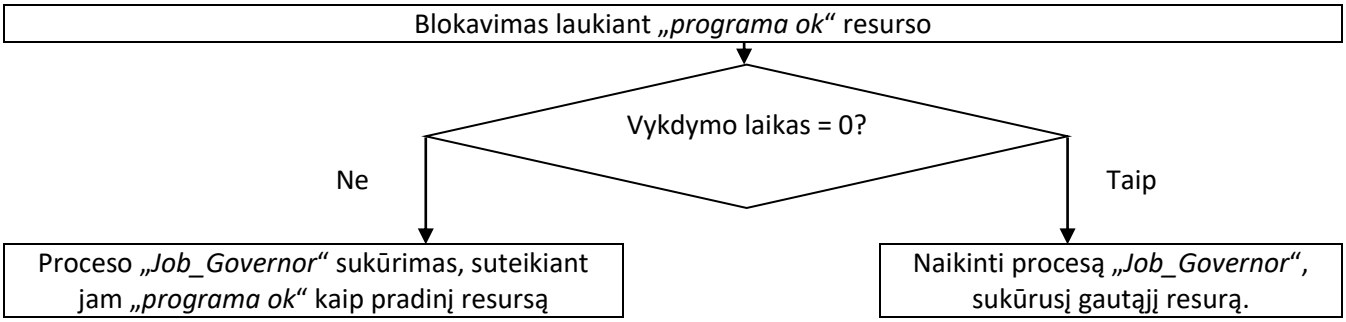
1. „*Sistemos statinių resursų inicializavimas*“ – tiek procesai, tiek resursai yra atstovaujami deskriptoriais. Inicializuoti resursą, reiškia sukurti deskriptorių. Darbas su deskriptoriais galimas tik per specialias operacijas – OS branduolio primityvus: „sukurti resursą“, „naikinti resursą“, prašyti resurso“, atlaisvinti resursą“.
2. „*Sisteminių statinių procesų inicializavimas*“ – yra penki primityvai darbui su procesais: „sukurti“, „naikinti“, „aktyvuoti“ , „keisti“ , „stabdyti“.
   Antrasis procesas įregistruoja resursą ar procesą deskriptoriuje. Primityvui reikia tai apibūdinti parametrais.
3. „*Blokavimas laukiant OS darbo pabaigos situacijos*“ - toliau procesui „*Start_Stop*“ darbas atsiras tik sistemai pabaigus darbą. O tai yra blokavimas (laukimas). Sistemos darbo pabaigoje bus sukurtas resursas, kurio lauke procesas „*Start_Stop*“. Tada jis atsiblokuos ir baigs savo darbą.
4. „*Statistikos išvedimas*“ – dabar užduoties srautas jau įvykdytas ir sistema baigė darbą. Procesai atsiblokuoja. Statistikos išvedimas.
5. „*Sisteminių statinių procesų naikinimas*“ – veiksmas atliekamas kreipiantis į primityvą „*sunaikinti procesus*“, tai yra sunaikinti deskriptorius.
6. „*Sisteminių statinių resursų naikinimas*“ – kreipiamasi į primityvą „*naikinti resursus*“.

   Taip pradedamas ir užbaigiamas sistemos darbas.

3.1.1. Procesas „*Main_Proc*“

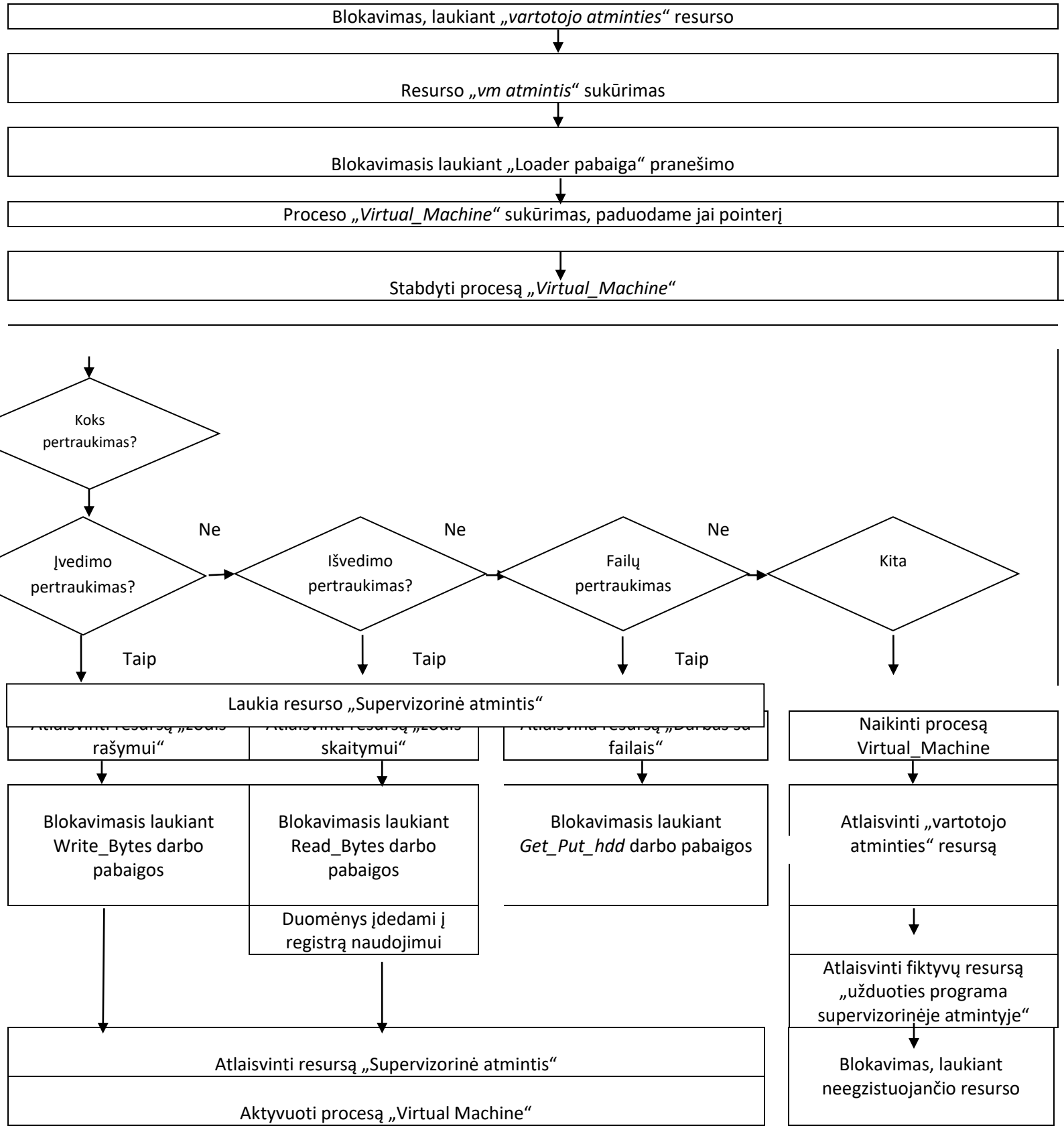
Procesas pradeda savo veiklą laukdamas „*programa ok*“.

Procesas Main_Proc, sulaukęs resurso „*programa ok*“, sukuria procesą „*Job_Governor*“.



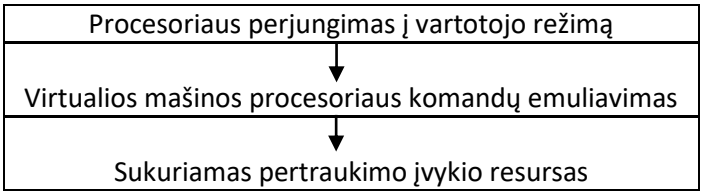
3.1.1.1.Procesas Job_Governor

Kuriamas Main_Proc proceso. Procesas virtualios mašinos tėvo, tvarkantis virtualios mašinos proceso darbą.



3.1.1.1.1. Virtual_Machine

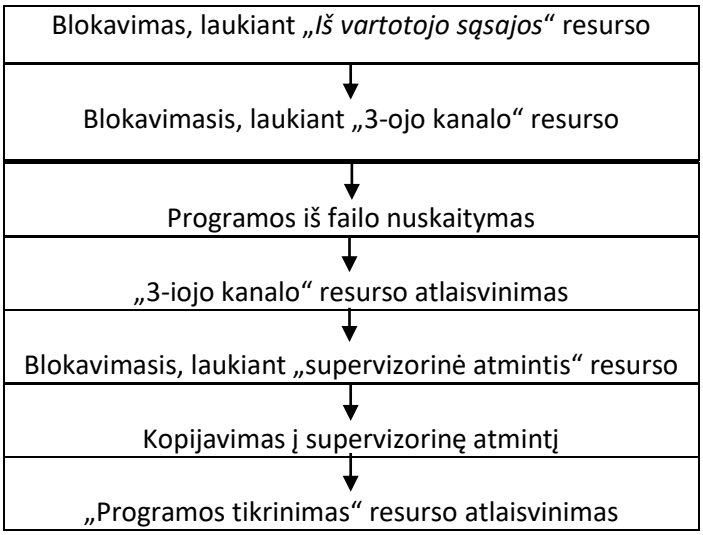
Procesas, atsakantis už vartotojiškos programos vykdymą.



3.1.2. Procesas „*Program_chooser*“

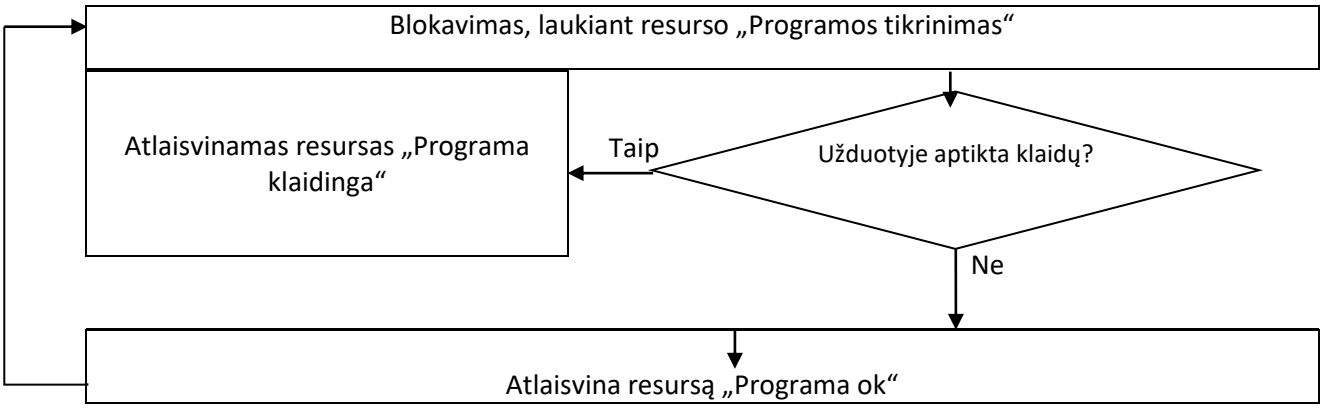
Šį procesą kuria ir naikina „Start_Stop“. „Program_chooser“ paskirtis: paimti eilutę iš įvedimo srauto, paimti resursą iš supervizorinės atminties ir perkelti programą į supervizorinę atmintį, patikrinti ją. Jas ima *„JCL“* ir analizuoja, ar jos teisingos.

Su įvedimo srautu susijęs tik procesas *„Program_chooser“*, todėl specialūs resursai reikalingi tik tam, kad nebūtų rūpesčių su įvedimo įrenginiais.

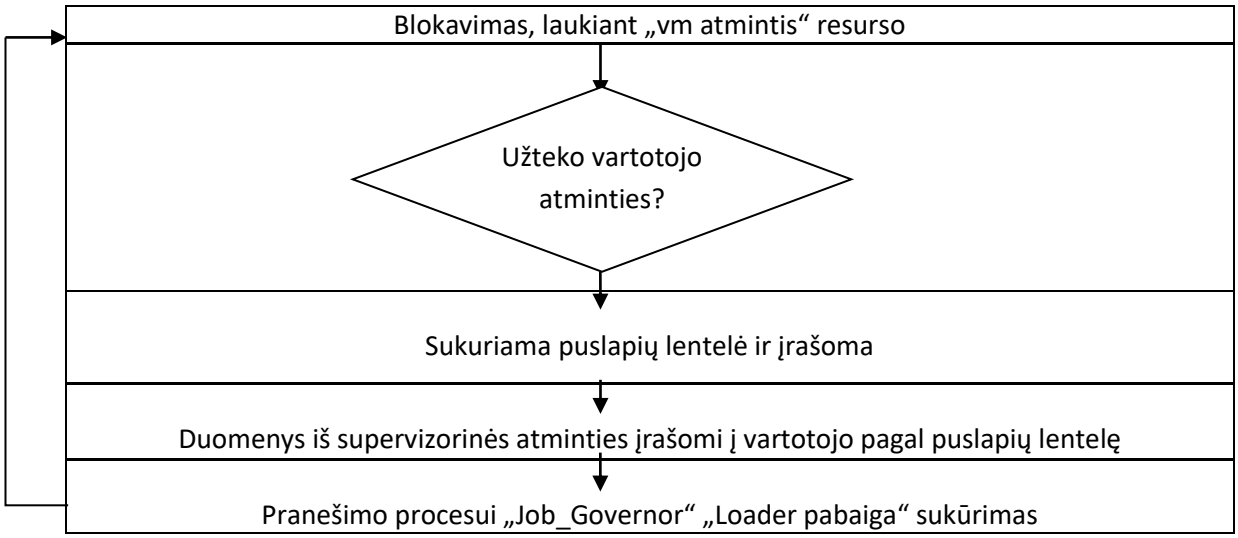


3.1.3. Procesas „*JCL*“

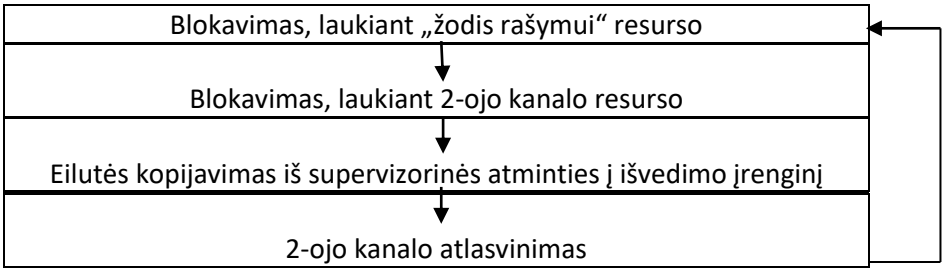
Procesą sukuria ir naikina „Start_Stop“. Procesas patikrina užduoties, esančios supervizorinėje atmintyje, struktūrą (sintaksę).



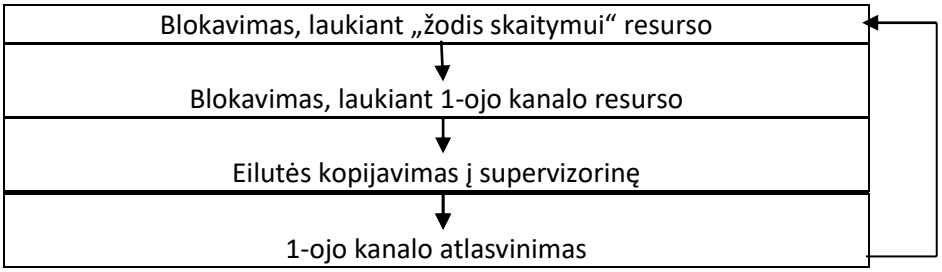
3.1.4. Procesas „*Loader*“



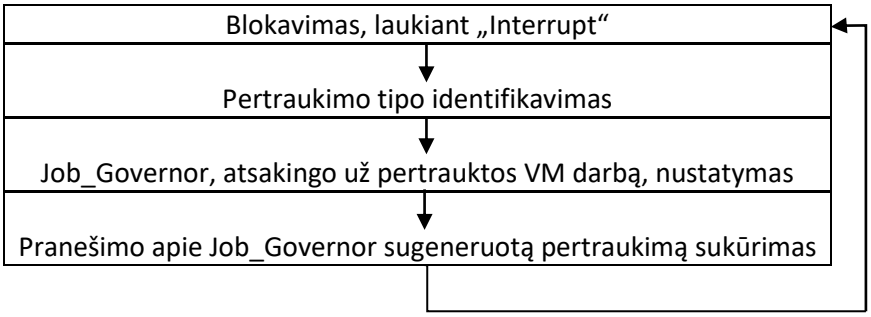
3.1.5. Procesas „*Write_Bytes*“



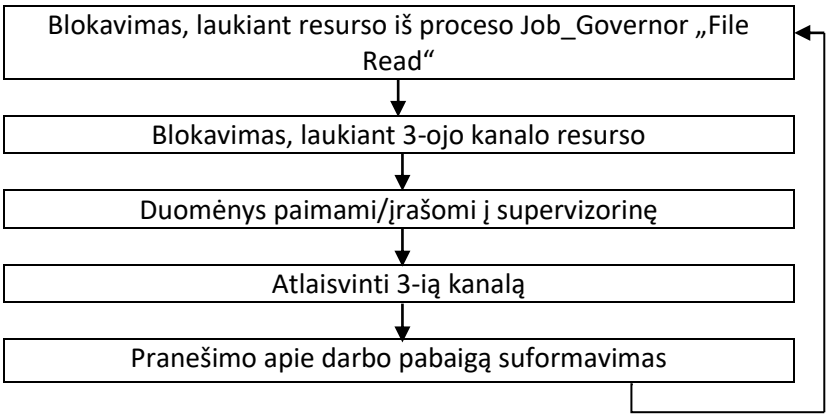
3.1.6. Procesas „*Read_Bytes*“



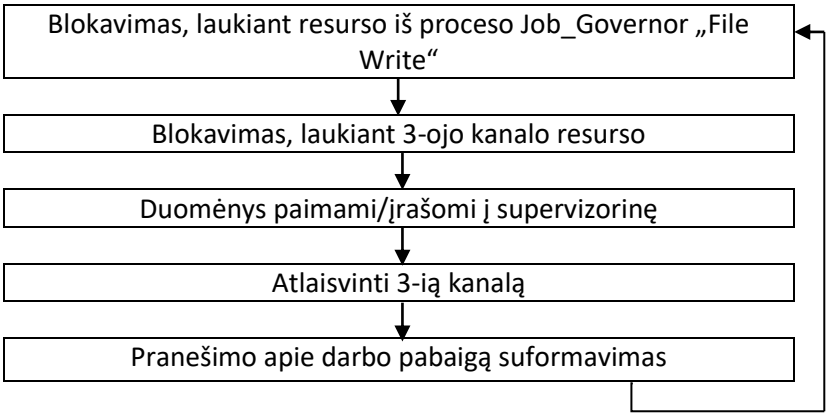
3.1.7. Procesas „*Interrupt*“



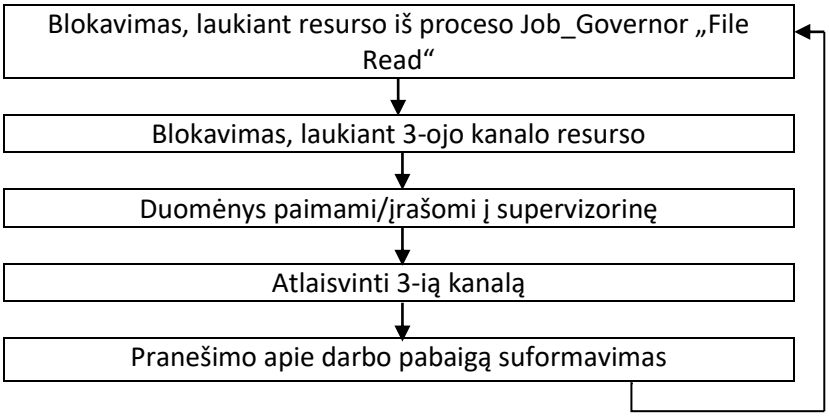
3.1.8. Procesas „*Get_hdd*“



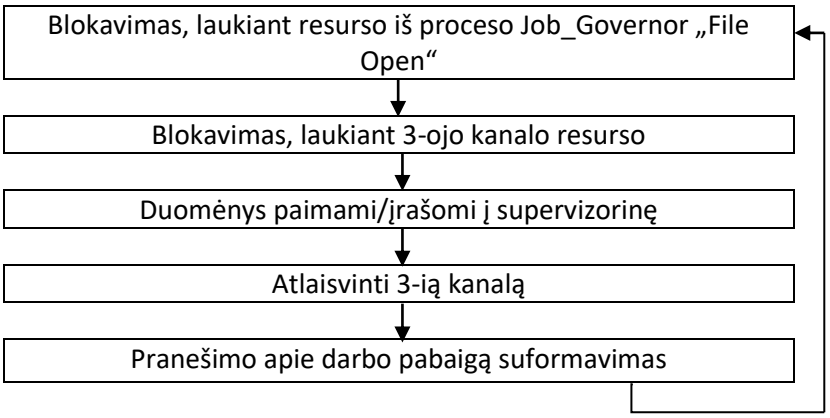
3.1.9. Procesas „*Put_hdd*“



3.1.10. Procesas „*Close_hdd*“



3.1.11. Procesas „*Open_hdd*“



3.1.12. Procesas „*Delete_hdd*“