VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS

OS projektas

Operacinės sistemos

Atliko: Tomas Kozakas, Daniil Samuilov

Vertino: Partn. Doc. Rokas Masiulis

Vilnius
2024

Turinys

1. Užduoties tikslai

1.1. Virtualios ir Realios mašinos projektas

Virtualios ir relios mašinos projektas[2]

- Sukurti virtualią mašiną, kuri būtų pritaikyta naudojimui su operacine sistema.
- Suteikti galimybę realizuoti operacinės sistemos tikslus naudojant tiek realią, tiek virtualią mašiną.
- Sukurti metodą operacinei sistemai dinaminių resursų valdymui.
- Užtikrinti, kad operacinė sistema galėtų veikti interaktyviai, leidžiant vartotojams paleisti programas ir bendrauti su jomis.
- Realizuoti Turingo pilnosios sistemos principus.
- Leisti programuotojams kurti ir iškviesti funkcijas naudojant rekursiją.

1.2. Multiprograminės OS projektas

Multiprograminės OS projektas[4]

- Pajėgi valdyti kelis procesus vienu metu, efektyviai paskirstant procesorių laiką.
- Užtikrinti efektyvų atminties valdymą, įskaitant fragmentavimo mažinimą ir virtualios atminties palaikymą.
- Sklandžiai paskirstyti ir valdyti įvairius resursus, įskaitant procesoriaus laiką, atmintį ir įvesties/išvesties įrenginius.
- Sukurti mechanizmus dinamiškam resursų valdymui, priklausomai nuo poreikių ir prioriteto.
- Robustiška failų sistema leis saugiai saugoti ir valdyti duomenis.
- Užtikrinti interaktyvumą, leidžiant vartotojams paleisti programas ir bendrauti su jomis.
- Sistema bus pagrįsta Turingo pilnosios sistemos principais.
- Leisti programuotojams kurti ir iškviesti funkcijas, naudojant rekursiją.

2. Reali mašina

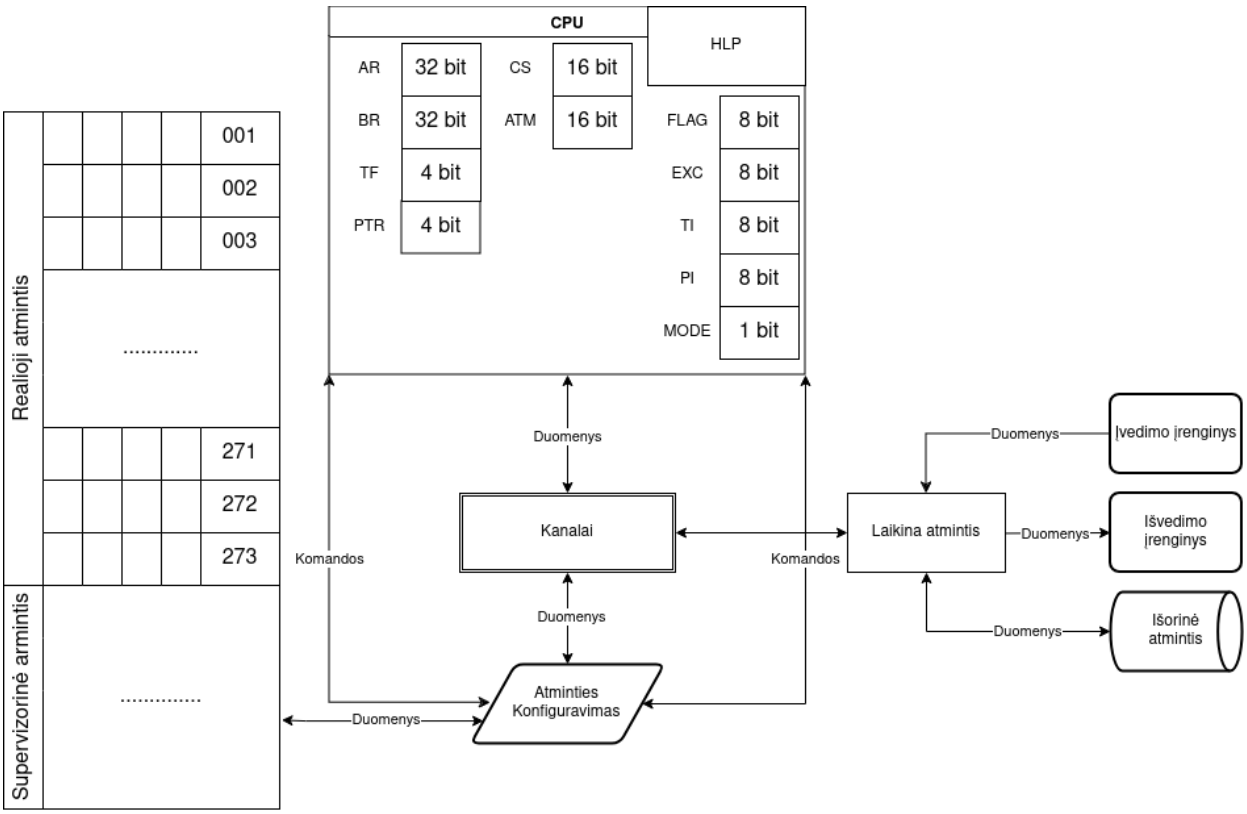
Reali mašina, arba kompiuteris, yra esminis įrankis, leidžiantis vykdyti įvairias operacijas. Kiekvienas procesas, kuris įvyksta šiame įrenginyje, susideda iš trijų būtinų elementų: programos, pradinių duomenų bei nustatymų, kurie leidžia šiai programai veikti.

2.1. Realios mašinos komponentai

Mūsų realioji mašina susideda iš septynių pagrindinių elementų:

1. Pagrindinis skaičiavimo modulis CPU
2. Atmintis
    - Supervisorinė atmintis
    - Vartotojo
    - Virtuali
3. Įvedimo/Išvedimo įrenginiai
4. Atminties konfigūratorius
5. Kanalų sistema
6. Išoriniai saugojimo įrenginiai

2.2. Realios mašinos modelis



2.1 pav. Realios mašinos modelis

2.2.1. Realios mašinos centrinis procesorius

Mūsų CPU turi 12 pagrindinius registrus (žr. 2.1 lentelė). Diagramoje parodytas kiekvieno iš jų užimamos atminties kiekis (žr. 2.1 pav.).

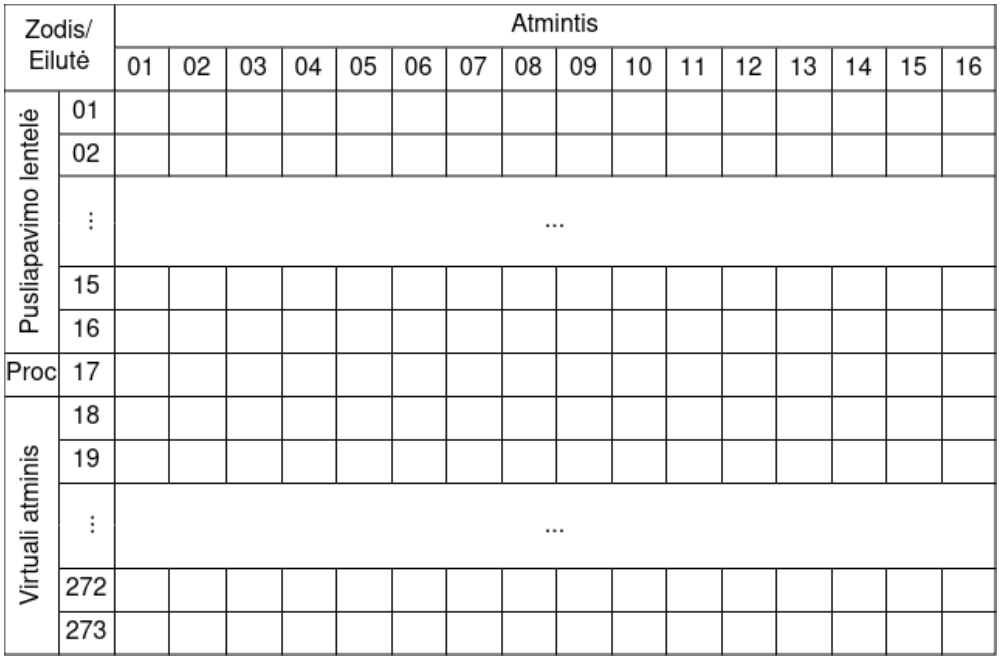
2.1 lentelė. CPU pagrindinių registrų apžvalga

Registras	**Aprašymas**
AR	Skirtas saugoti bendrą informaciją, su kuria šiuo metu dirbama.
BR	Taip pat kaip AR, skirtas saugoti bendrą informaciją.
PTR	Nurodo virtualios mašinos numerį.
TI	Laikmatis, nurodantis, kiek laiko procesas veikia. Po 10 ciklų vyksta perėjimas prie kito proceso.
TF	Teisinga arba neteisinga (sprendimo atsakymas).
CS	Komandų skaitiklis, nurodo, kuri komanda šiuo metu vykdoma. Kiekvieną kartą, kai vykdoma nauja komanda, skaitiklis padidėja.
ATM	Nurodo atminties ląstelę, su kuria šiuo metu dirbama.
FLAG	Dirba su išoriniais įrenginiais (įvedimo, išvedimo, laikina atmintis).
EXC	Pertraukimai.
PI	Visos išimtys, susijusios su programos (vm) vykdymu.
MODE	Procesoriaus režimas.

2.2.2. Virtualios mašinos komandos

Atminties langeliai, užimantys keturis baitus, gali būti suprantami kaip komanda ir kaip duomenys – tai priklauso nuo paskutinio bito. Mikroprogramos, interpretuojančios virtualaus procesoriaus komandas bus vykdomos HLP.

2.2.3. Realios mašinos atmintis



2.2 pav. Atmintis

Mūsų tikroji atmintis yra padalinta į du blokus: supervizoriaus atmintį, kurioje saugoma visa operacinės sistemos darbui reikalinga informacija, ir vartotojo atmintį, kurioje saugomos visos virtualios mašinos. Atmintis padalinta į 4 baitų blokus, kiekvienas blokas sudarytas iš 16 eilučių. Kiekvienos 16 eilučių dalis sudaro vieną skyrių, o yra 16 skyrių. Turint vietos, reikalingos puslapių numeravimui, yra 4368 blokų po keturis baitus, neskaičiuojant dalies, kurią užima operacinė sistema.

2.2.4. Atminties konfigūratorius

Atminties konfigūratorius atsakingas už tai, kad tarnautų kaip tam tikra kliūtis arba barjeras tarp centrinio procesoriaus ir išorinės informacijos, suteikdamas papildomo saugumo ir veikimo stabilumo. Sistema dirba su šiuo moduliu standartiškai, kai veikia neautomatiniu būdu. Pagrindinė šio modulio užduotis – kaupti informaciją, kol procesorius galės ją apdoroti.

2.2.5. Kanalų sistema

Kanalų sistema buvo perkelta į atskirą struktūrą, siekiant parodyti, kad turime tam tikrus modulius, atsakingus už tai, kad perduodamoje informacijoje nebūtų konfliktų. Papildoma pastaba: visa informacija būtinai praeina per šį modulį. Kitos rodyklės, jungiančios procesorių su atminties konfigūratoriumi bei procesorių su atminties prižiūrėtoju, yra skirtos parodyti, kad procesorius valdo šiuos modulius, tačiau jis negauna informacijos tiesiogiai iš jų.

2.2.6. Taimerio mechanizmas

Registeris **TI** (laikmačio registras) yra naudojamas operacinėse sistemose, kad apribotų vieno proceso veikimo laiką centrinio procesoriaus resursų naudojimo atžvilgiu. Tai užkerta kelią tam, kad vienas procesas negalėtų monopolizuoti CPU laiko ir užtikrina, kad kiti procesai taip pat galėtų

gauti procesoriaus laiko.

Štai kaip **TI** gali veikti sistemoje:

- Kai procesas pradeda vykdyti, **TI** registras nustatomas į tam tikrą pradinę reikšmę 16. Kiekvienas procesoriaus ciklas (arba kitas iš anksto apibrėžtas laiko vienetas) mažina šį registrą vienetu.
- Kai **TI** pasiekia nulį, tai reiškia, kad proceso laikas baigėsi. Operacinė sistema tai atpažįsta kaip signalą, kad reikia perjungti kontekstą – sustabdyti dabartinį procesą ir perduoti CPU kontrolę kitam procesui, kuris laukia savo eilės.
- Konteksto perjungimas gali apimti dabartinio proceso būsenos (įskaitant visus registro reikšmes) išsaugojimą, kad vėliau procesas galėtų būti tęsiamas nuo tos pačios vietos.
- Naujai pradedamam procesui **TI** registras vėl nustatomas į pradinę reikšmę, ir ciklas tęsiasi.

3. Virtuali mašina

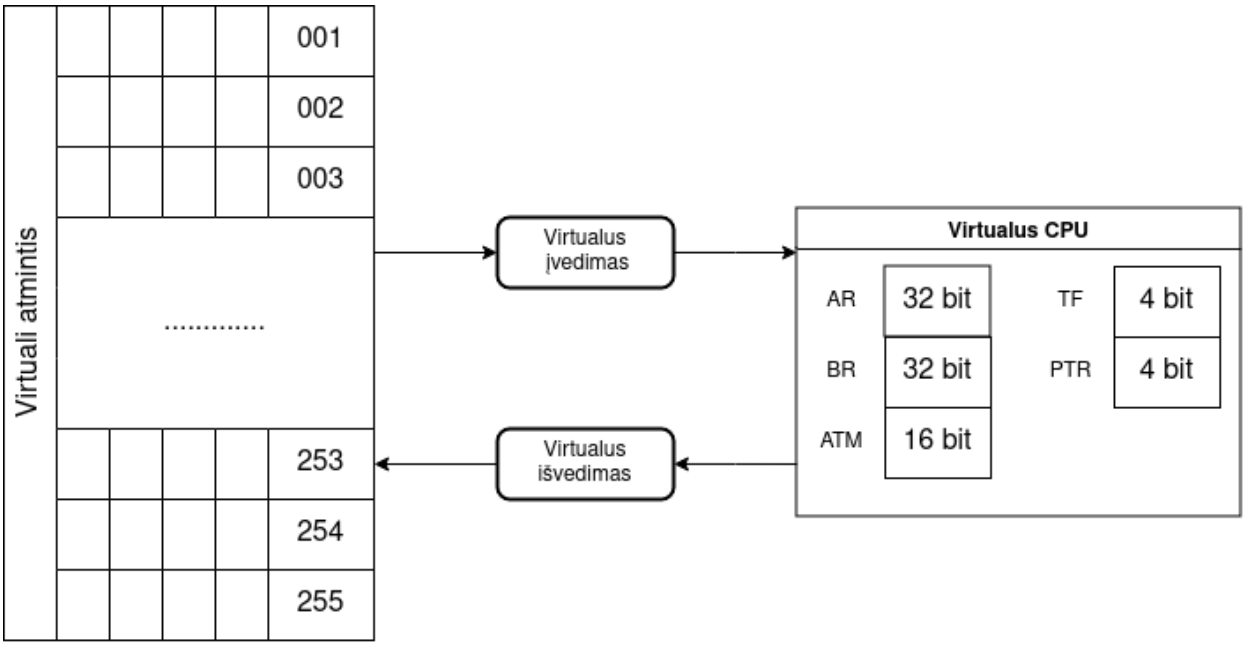
Virtuali mašina (vm) sukuria simuliaciją, kurioje programos gali veikti taip, tarsi kiekviena iš jų turėtų savo asmeninę kompiuterio sistemą, nepaisant to, kad realybėje jos bendrai naudoja tas pačias fizinio kompiuterio išteklius. Kiekvieną virtuali mašina konkuruoja dėl realaus procesoriaus laiko ir programoms veikia izoliuotai viena nuo kitos, taip užtikrinant didesnį saugumą, lankstumą ir resursų naudojimo efektyvumą.

3.1. Virtualios mašinos komponentai

Virtualios mašinos struktūra yra daug paprastesnė, palyginti su realiąja. Ji susideda iš keturių dalių, iš kurių dvi gali būti sujungtos.

1. Virtualus procesorius
2. Virtualus atmintis
3. Įvedimo/Išvedimo sistema

3.2. Virtualios mašinos modelis



3.1 pav. Virtualios mašinos modelis

3.2.1. Virtualios mašinos procesorius

Pradėkime nuo širdies, tiksliau, procesoriaus. Čia yra į registrai: **AR**, **BR** registrai bei **ATM**, **TF**, **PTR**. Jie atlieka tą patį vaidmenį kaip ir anksčiau (žr. 2.1 lentelė). Tokios kaip sudėjimas, atėmimas, palyginimas, bus atliekamos tarp dviejų registrų **AR** ir **BR**.

3.2.2. Virtualios mašinos atmintis

Kiekvienai virtualiai mašinai skiriame šešiolika eilučių po 16 blokų. Primenu, kad blokas susideda iš keturių baitų

Peršokimo operacijai naudojama informacija yra saugoma **PTR** registeryje, o **MODE** registras nurodo, ar operacija yra leidžiama, ir ši operacija nereikalauja jokių papildomų operatorių. **PTR** registras atsako už konkrečios atminties ląstelės nustatymą, kuri turi būti pasiekiama, o **MODE** registras nurodo, ar prieiga prie šios ląstelės yra leidžiama.

FLAG registras yra skirtas dirbti su sistemos vidine logika, pvz., nustatyti lyginimo operacijų rezultatus arba signalizuoti apie perpildymą atliekant aritmetines operacijas. Tuo tarpu **EX** registras yra susijęs su išoriniais įrenginiais, nurodydamas, pavyzdžiui, kada informacija gali būti įkelta į kompiuterį, CPU yra pasirengęs vykdyti kitą užduotį, arba signalizuojant apie gedimus.

Svarbu pabrėžti, kad **FLAG** registras yra saugomas virtualioje atmintyje, skirtingai nei **EX** registras, kas padidina sistemos saugumą.

3.2.3. Komandų sistema

3.1 lentelė. Virtualios mašinos komandų sistemos lentelė

Komanda	Argumentų perdavimas	Rezultatas
	Aritmetinės Operacijos	
ADD	AR, BR	AR
SUB	AR, BR	AR
MUL	AR, BR	AR
DIV	AR, BR	AR, BR (liekana)
NEG	AR	AR
	Logikos Operacijos	
AND	AR, BR	AR
OR	AR, BR	AR
NOT	AR	AR
	Palyginimo Operacija	
CMP	AR, BR	TF: 0 AR < BR, 1 AR > BR, 2 AR = BR
	Duomenų Užkrovimas ir Saugojimas	
LD	adresas	Įkrauna duomenis iš nurodyto adreso į AR registą
ST	ATM	Saugo duomenis iš AR registro į nurodytą adresą
MOVE	x_1, x_2 (registrai arba atminties adresai)	Užnulina x_1, perkelia x_2 į x_1
VAL	x_1 - adresas, x_2 - reikšmė	Užnulina x_1, perkelia x_2 į x_1
	Valdymo Operacijos	
JM	adresas	Perkelia adresą į ATM
JL	adresas	Perkelia adresą į ATM
JG	adresas	Perkelia adresą į ATM
JMR	adresas	Perkelia adresą į ATM
JLR	adresas	Perkelia adresą į ATM
JGR	adresas	Perkelia adresą į ATM
HALT	-	EXC
CHAR	kanalas paruoštas	EXC
MEMR	atmintis paruošta	EXC
PROR	programa paruošta	EXC
	Įvedimo/Išvedimo Operacijos	
PRINT	adresas	EXC

3.3. Puslapiavimo mechanizmas

Šiame skyriuje aptariamas virtualios atminties adresų konvertavimas į realius adresus naudojant puslapiavimo metodą. Mūsų atminties valdymo schema rezervuoja pirmąsias 256 atminties ląsteles puslapių lentelės duomenims. Virtuali mašina naudoja atminties ląsteles nuo 256 iki 272, o reali atmintis prasideda nuo 273 iki 4368 ląstelės.

Adreso konvertavimo procesą galima aprašyti šiais žingsniais:

1. **Puslapių lentelės indeksas** skaičiuojamas pagal formulę:

$$\text{Puslapių lentelės indeksas} = (PTR \times 16) + (\text{Adresas}/16),$$

   kur PTR yra puslapių lentelės registras, nurodantis į puslapių lentelę atmintyje, ir Adresas yra virtualus adresas.

2. **Realus Adresas** apskaičiuojamas pagal formulę:

$$\text{Realus Adresas} = (\text{Puslapio numeris} \times 16) + (\text{Adresas} \mod 16),$$

   kur:

   - **Puslapio numeris** gaunamas iš puslapių lentelės naudojant anksčiau apskaičiuotą indeksą.
   - Adresas $\mod 16$ nurodo baito numerį puslapyje.

3.4. Virtualios mašinos kūrimo ir veiklos scenarijus

1. Pradėta kurti virtuali mašina.
2. Virtuali mašina reikalauja 16 takelių atminties savo reikmėms.
3. Yra išskiriama 16 takelių virtualiai mašinai.
4. Yra išskiriamas 1 takelis virtualios mašinos puslapių lentelei.
5. Puslapių lentelė (t.y. tas takelis) yra užpildomas išskirtų 16 takelių realiais adresais.
6. Virtualios mašinos virtualaus registro PTR reikšmei priskiriamas puslapių lentelės takelio realus adresas.
7. Virtuali mašina baigiama kurti.
8. Virtuali mašina gauna procesorių.
9. Virtualiai mašinai prireikė paversti virtualų adresą 128 (t.y. 8 takelis, parinkta atsitiktinai) realiu.
10. Virtuali mašina iš puslapių lentelės nuskaito 5 žodį. Tai ir yra realus adresas.
11. Taip toliau

3.5. Pertraukimų mechanizmas

Veikiant virtualioms mašinoms, tokiu atveju gali kilti tam tikrų problemų. Atėjus laikui, operacinės sistemai veikiant, iškviečiami pertraukimai. Ji nuskaito duomenis iš registrų ir, priklausomai nuo jų turinio, iškviečia vieną iš integruotų programų, kurios yra operacinės sistemos dalis. Pertraukimai gali kilti šiais būdais:

- **PI Register:**
  - Netinkama atmintis (Memory Fault)
  - Atminties persipildymas (Memory Overflow)
- **FLAG Register:**
  - Reikšmės už ribų (Value Out of Range)
  - Dalijimosi iš nulio klaida (Division by Zero Error)
- **EXC Register:**
  - Neįprastų instrukcijų kodas (Illegal Instruction)

3.6. Bendravimo su įvedimo/išvedimo įrenginiais mechanizmas

Ryšys su įvesties ir išvesties įrenginiais vyksta tik per operacinę sistemą; tai reiškia, kad virtualios mašinos neturi tiesioginės prieigos prie išvesties įrenginių ne per operacinę sistemą. Komunikacija tarp CPU ir išorinių įrenginių vykdoma per **EXC** registrą, o informacija gaunama naudojant standartinius kanalus. Pirma, informacija patenka į **supervizorinę atmintį** ir lieka ten, kol procesorius gali ją priimti ar išvesties įrenginys yra pasiruošęs ją apdoroti. Be to, informacija perduodama per kanalų sistemą, kuri užkerta kelią informacijos praradimui perdavimo metu ir apsaugo nuo duomenų maišymosi, kai šie gaunami iš skirtingų šaltinių. Kai informacija patenka tiesiai į procesorių, jis nustato, kur ją įrašyti, ir tada informacijos įkėlimo operacija atliekama naudojant **PTR** registrą.

3.7. Virtualios mašinos programos pavyzdys

Šiame skyriuje pateikiamas virtualios mašinos programos kodas, kuris atlieka paprastą aritmetinę operaciją: dviejų skaičių sumavimą. Programa naudoja anksčiau aprašytas komandas (žr. 3.1 lentelė).

```
DATA_SEGMENT
VAL 3     ; Pirmas operandas
VAL 5     ; Antras operandas
result VAL 0   ; Rezultato kintamasis

CODE_SEGMENT
LD var1         ; Užkrauna reikšmę į AR registrą
MOVE BR, AR     ; Užkrauna reikšmę iš AR į BR registrą
LD var2         ; Užkrauna reikšmę į AR registrą
ADD             ; Prideda BR prie AR registro turinio
PT result       ; Saugo gautą sumą viduj AR į 'result' kintamąjį
PRNTC result    ; Spausdina gautą rezultatą
HALT            ; Sustabdo programos vykdymą
```

4. Multiprograminė operacinė sistema

4.1. Procesas

Procesas – tai vykdoma programa, kartu su esamomis registrų reikšmėmis ir savo atmintimi. Kiekvienas procesas turi savo virtualų procesorių. Nors skirtumas tarp programos ir proceso nėra didelis, bet jis svarbus. Procesas – tai kokioje nors veiklumo stadijoje esanti programa. O tuo tarpu programa – tik tam tikras baitų rinkinys. Veiklumo stadiją apibūdina proceso aprašas – deskriptorius. Apraše ir yra laikomi visi procesui reikalingi parametrai, tokie kaip registrų reikšmės.

4.1 lentelė. Procesų sąrašas

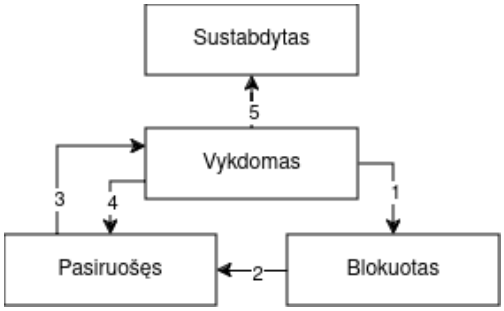
Procesas	**Aprašymas**
start_stop[4.9.1]	Paleidžia ir stabdo operacinę sistemą.
main_proc[4.9.2]	Valdo pagrindinius procesus.
job_governor[4.9.3]	Valdo ir kūrią virtualias mašinas.
interrupt[4.9.4]	Atlieka veiksmus susijusius su pertraukimu sistemoje.
environment_interaction[4.9.5]	Sąveika su aplinka ir užtikrina išvesties/išvesties operacijas.
jcl[4.9.6]	Valdo JCL ir užtikrina virtualių mašinų veikimą.
vm[4.9.7]	Atlieka veiksmus virtualioje mašinoje.
get_put_data[4.9.8]	Atlieka duomenų įkėlimą ir iškėlimą.

4.2. Procesų būsenos

Procesas gali gauti procesorių tik tada, jei jam netrūksta jokio kito resurso. Procesas gavęs procesorių tampa vykdomu. Procesorius iš proceso gali būti atimtas įvykus pertraukimui sistemoje arba jei procesui prireikia kokio nors resurso. Procesorius iš proceso gali būti atimtas dar ir dėl to, kad procesas procesorių turejo ilgą laiko tarpą. Procesų būsenos:

- Vykdomas – turi procesorių
- Blokuotas – prašo resurso (išskyrus procesoriaus)
- Pasiruošęs – turi visus reikalingus resursus ir prašo procesoriaus
- Sustabdytas – negali tęsti darbo dėl kilusio pertraukimo

Procesorius yra ypatingas resursas todėl, kad jis yra reikalingas visiems procesams ir be jo nė vienas procesas netaps vykdomu. Tik vykdomas procesas gali atlikti savo darbą. Procesų būsenų kitimo diagrama:



4.1 pav. Procesų būsenų kitimo diagrama

Diagramoje matome 5 perėjimus. Trumpai aptarsime kiekvieną:

1. Procesas yra vykdomas, kol jis prašo resurso ir tampa blokuotas.
2. Blokuotas procesas tampa pasiruošusiu, kai reikiamas resursas yra paskiriamas.
3. Pasiruošę procesai varžosi dėl procesoriaus. Kai kuris procesas gavęs procesorių, tampa vykdomu.
4. Vykdomas procesas vėl tampa pasiruošusiu po to, kai iš jo buvo pašalintas procesorius dėl įvairių priežasčių, išskyrus resurso trūkumą.
5. Vykdomas procesas gali būti sustabdytas, jei įvyksta tam tikras pertraukimas, pavyzdžiui, vartotojo programos komanda viršija savo atminties ribas. Kaip matome iš diagramos, sustabdyto proceso būsenos daugiau pakeisti nebegalima.

4.3. Planuotojas

Planuotojo užduotis – skirstyti procesorių. Atimti iš vienų procesų ir duoti kitiems, jo manymu, labiausiai vertiems procesoriaus. Planuotojo tikslai:

- Užtikrinti, kad kiekvienas procesas gautų procesorių reikiamą laiko tarpą
- Maksimaliai užimti procesorių
- Iki minimumo sumažinti atsakymo laiką vartotojams.

Vienas iš planuotojo algoritmų yra prioritetais pagrįstas modelis. Kiekvienam procesui suteikiamas prioritetas iš sutarto intervalo. Procesas, turintis didesnį prioritetą vykdomas pirmiau, nei mažesnį prioritetą turintis procesas.

Planuotojas kviečiamas, kai norima procesorių perduoti kitam procesui. Reikia pabrėžti, kad procesas yra einamasis (vykdomas) iki kol perduodamas valdymas.

Planuotojo žingsniai:

1. Einamojo proceso būsenos tikrinimas. Jei jis nėra blokuotas, įtraukiamas į pasiruošusių procesų sąrašą.
2. Toliau tikrinama, ar yra pasiruošusių procesų. Jei pasiruošusių procesų sąrašas netuščias, imamas pirmas sąraše procesas.
3. Tada perduodamas valdymas. Proceso apraše laikoma virtualaus procesoriaus būsena priskiriama realiam procesoriui, išsaugoma einamojo proceso aplinka, užkraunama pasirinktojo proceso aplinka. Naujasis procesas pažymimas kaip einamasis.

4.4. Procesų primityvai

Procesų primityvų paskirtis – pateikti vieningą sąsają dirbti su procesais. Išskirkime juos:

- Kurti procesą – sukuriamas bei padedamas į pasiruošusių procesų eilę proceso deskriptorius, skiriamas atminties resursas ir įkeliamos programos instrukcijos į atmintį.
- Naikinti procesą – pašalinamas atminties resursas ir proceso deskriptorius iš visų sąrašų.
- Blokuoti procesą – pakeičiama proceso būsena į blokuotą.
- Aktyvuoti procesą – pakeičiama proceso būsena į vykdomas.

4.5. Resursai

Resursas yra tai, dėl ko varžosi procesai. Dėl resursų trūkumo procesai blokuojasi, o gavę reikiamą resursą procesai tampa pasiruošusiais. Resursų sąrašas:

- Atmintis
- Procesorius
- Įvedimo srautas
- Išvedimo srautas

Atmintis skiriama proceso kūrimo metu. Procesoriaus resursą skirsto planuotojas. O įvedimo srauto resursą skirsto resursų paskirstytojas. Kiekvienas resursas turi jo laukiančių procesų sąrašą (išskyrus atmintį). Kai prašomas resursas yra suteikiamas, proceso aprašas yra padedamas į pasiruošusių procesų eilės pradžią. Resursų paskirstytojo darbo pabaigoje iškviečiamas planuotojas.

4.6. Resursų primityvai

- Kurti resursą: sukuria naują resursą su unikaliu vidiniu vardu ir prideda jį prie resursų sąrašų.
- Naikinti resursą: pašalina resursą iš sąrašų ir atlaisvina procesus, laukiančius šio resurso.
- Prašyti resurso: užblokuoja procesą ir prideda jį prie laukiančių procesų sąrašo.
- Atlaisvinti resursą: prideda resurso elementą prie sąrašo, kad jis būtų prieinamas kitiems procesams.

4.7. Resurso paskirstytojas

Resursų paskirstytojas skirsto resursus procesams, kaip ir planuotojas skirsto procesorius. Jis aptarnauja procesus, kurie prašo ar atlaisvina resursus. Paskirstytojo algoritmas gali būti sudėtingas, įskaitant galimybę atiduoti resursą konkrečiam procesui arba prašyti kelių resurso elementų.

Paskirstytojas peržvelgia laukiančių procesų sąrašą ir, aptarnaudamas procesą, pažymi jį pasiruošusiu. Pabaigoje, paskirstytojas kviečia planuotoją.

4.8. Lentelė

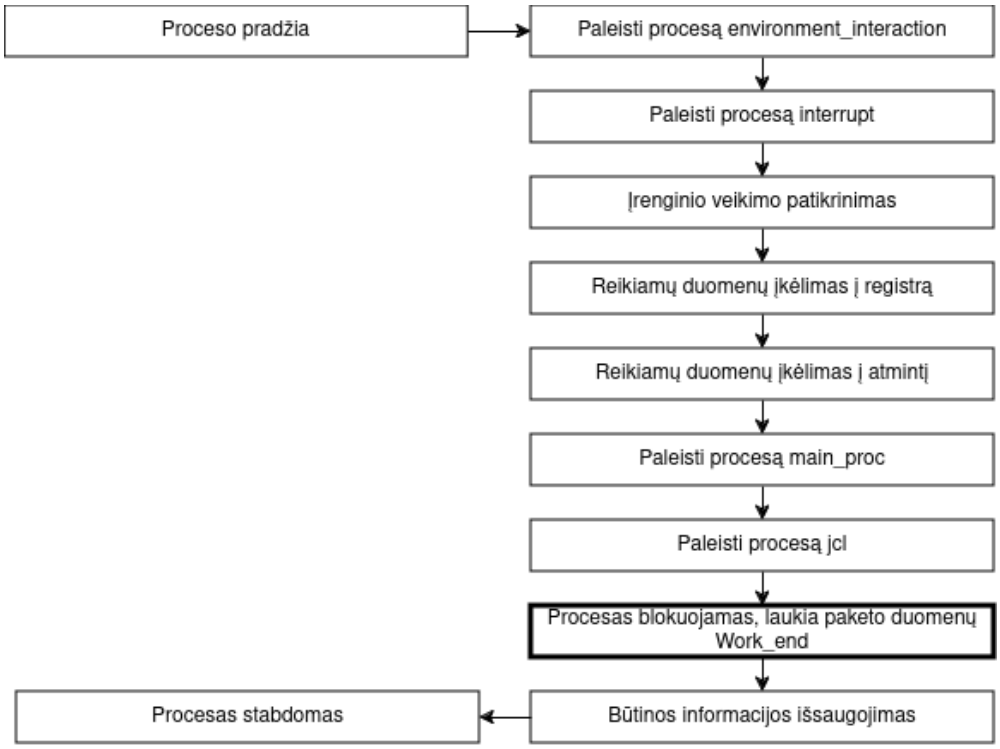
name	data	from	to
Work_end	-	main_proc	res_pask
Work_end	-	res_pask	start_stop
Work_end_U	“off”	environment_interaction	res_pask
Work_end_U	“off”	res_pask	main_proc
Work_end_U	“add”	environment_interaction	res_pask
Work_end_U	“add”	res_pask	main_proc
Work_end_U	“start”	environment_interaction	res_pask
Work_end_U	“start”	res_pask	main_proc
Stop	-	planoutojas	any proces
Continue_work	-	planoutojas	any proces
End_of_process	name	any proces	planoutojas
Start_new_process	name	any proces	planoutojas
Output	data	VM, main_proc,EXE	environment_interaction
Input	data	environment_interaction	VM, main_proc, interrupt
Input_I	-	environment	environment_interaction
Add_VM	-	main_proc	job_governor
New_VM_to_list	-	job_governor	jcl
ERROR	exe	any proces	interrup, planoutojas
All_done	-	interrupt	res_pask, planoutojas
Run_complete	-	get_put_data	job_governor

4.2 lentelė. Paskirstytojo veiksmai

4.9. Procesai

4.9.1. start_stop

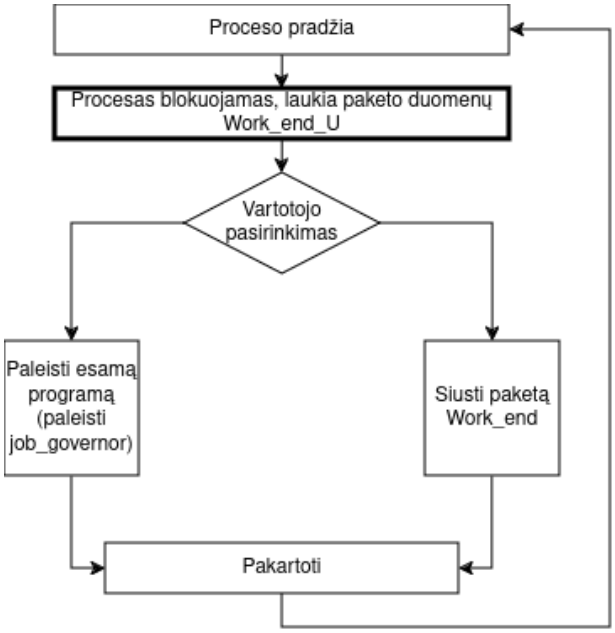
Proceso *start_stop* tikslas yra paleisti ir sustabdyti operacinę sistemą. Šis procesas yra pradinis ir vykdomas pirmiausia. Jo užduotis apima reikiamų duomenų įkėlimą į atitinkamas vietas, kitų procesų pasiruošimą darbui, pradinės registro būsenos patikrinimą ir įrenginio veikimo patikrinimą.



4.2 pav. start_stop Procesas

4.9.2. main_proc

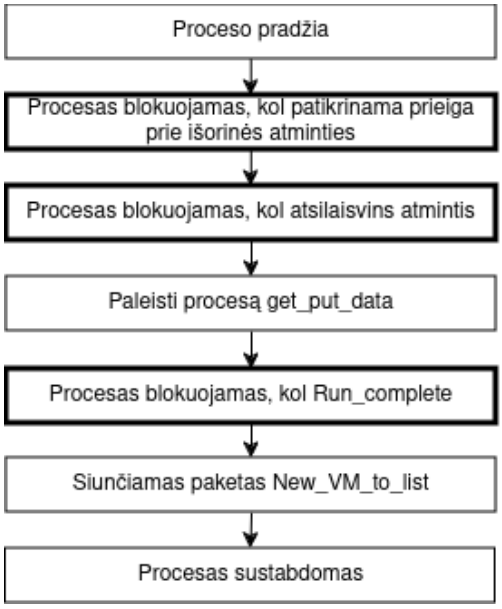
main_proc yra pagrindinis valdymo centras, kuris gali paleisti ir sustabdyti pagrindinius procesus.



4.3 pav. main_proc Procesas

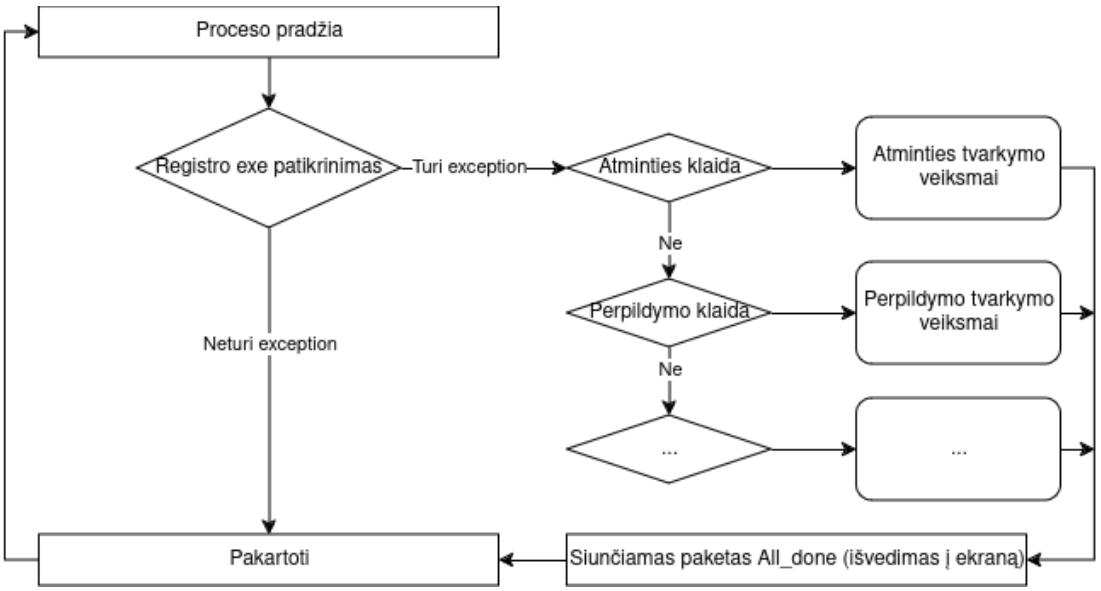
4.9.3. job_governor

job_governor proceso funkcija yra valdyti, kurti ir sunaikinti virtualiąsias mašinas bei suteikti pagalbą joms vykdyti jų darbą. Šios pagalbos veiksmai yra vykdomi tais atvejais, kai virtuali mašina, veikianti procesoriaus vartotojo režime, negali savarankiškai atlikti tam tikrų veiksmų. Kiekvienas *job_governor* procesas yra skirtas aptarnauti vieną virtualiąją mašiną.



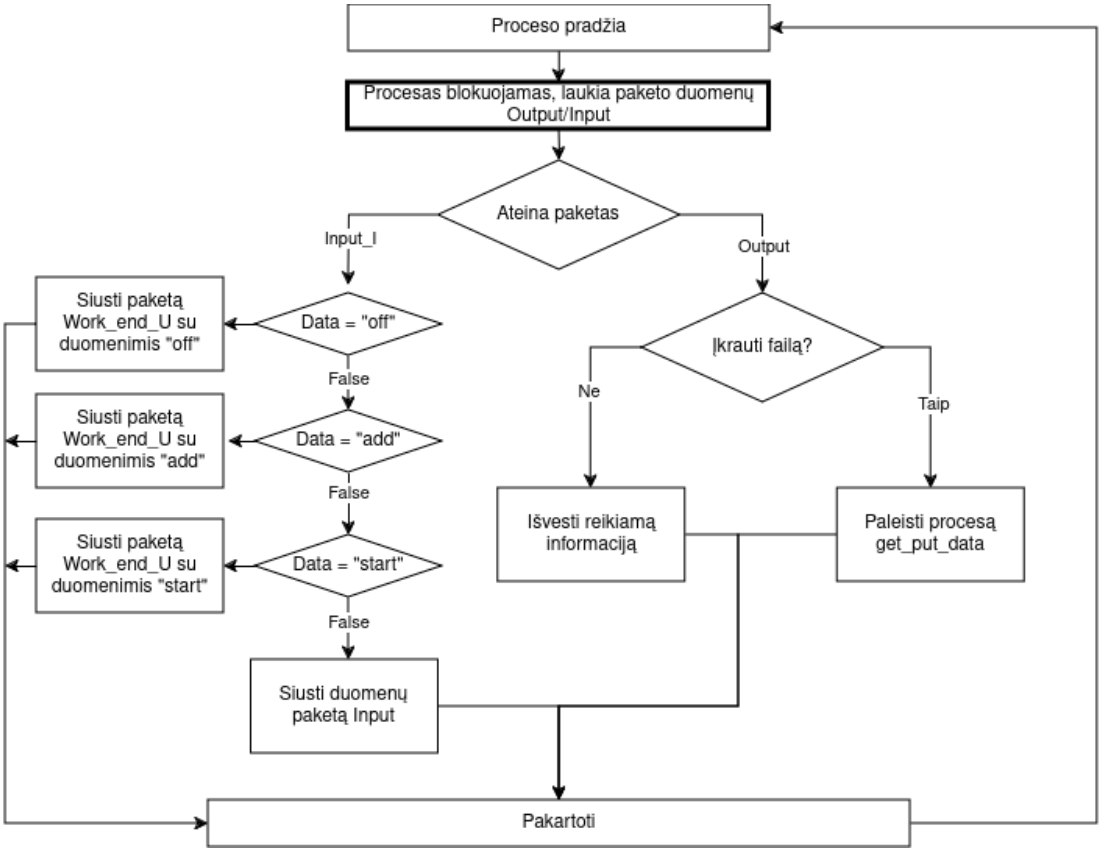
4.4 pav. job_governor Procesas

4.9.4. interrupt



4.5 pav. interrupt Procesas

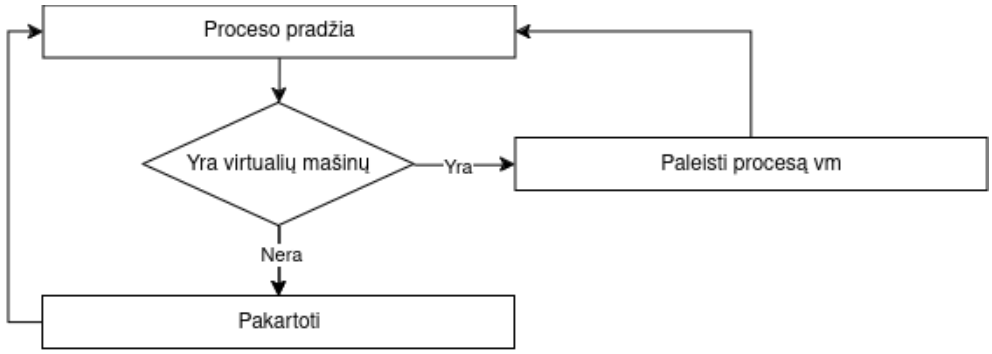
4.9.5. environment_interaction



4.6 pav. environment_interaction Procesas

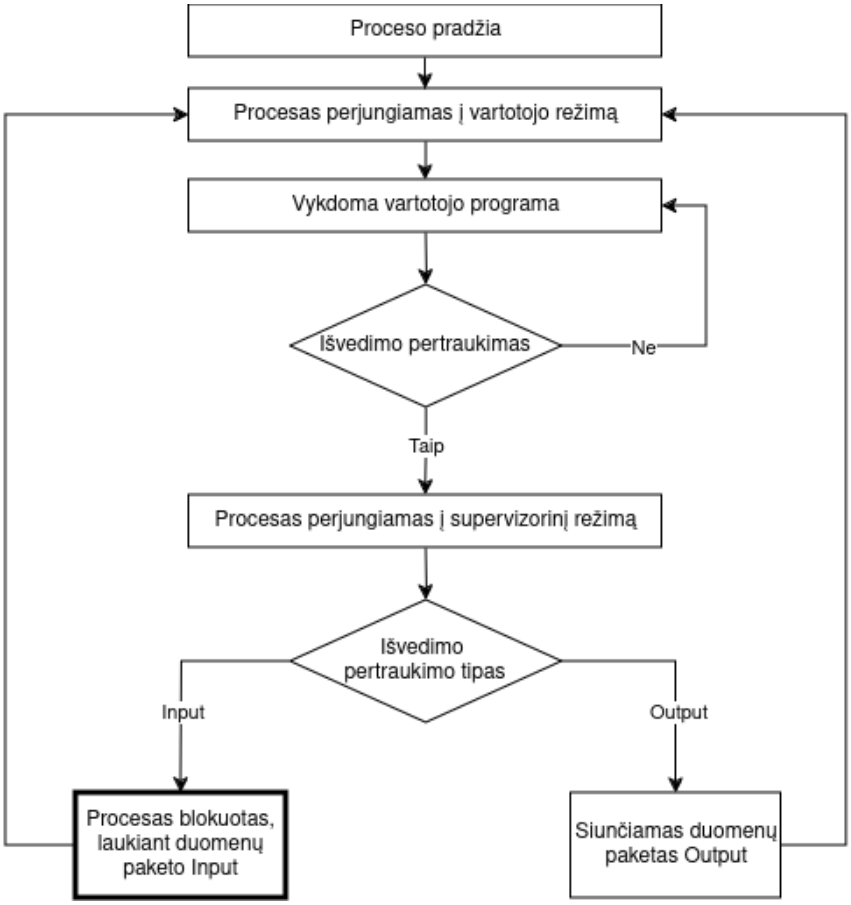
4.9.6. jcl

jcl paleidžia paruoštą virtualiąją mašiną ir užtikrina jų lygiagretų veikimą.



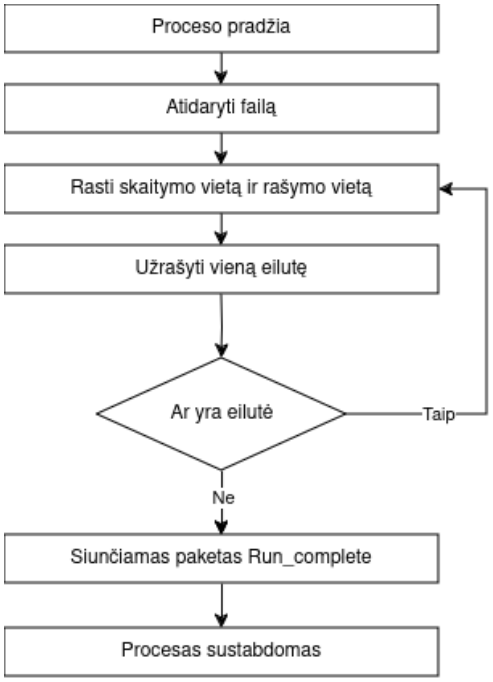
4.7 pav. jcl Procesas

4.9.7. vm



4.8 pav. vm Procesas

4.9.8. get_put_data



4.9 pav. get_put_data Procesas