Vilniaus Universitetas

Matematikos ir Informatikos Fakultetas

Kurso „Operacinės Sistemos“ III-asis darbas

Multiprograminės operacinės sistemos projektas.

*Darbą parengė:*

Domantas Masolas (Informatika, 3 kursas)

Joelis Jonas Šereika (Informatika, 3 kursas)

Turinys

[1. Užduoties sąlyga 4](#_Toc40981102)

[2. Procesai 5](#_Toc40981103)

[Procesų būsenos 5](#_Toc40981104)

[3. Planuotojas 6](#_Toc40981105)

[4. Procesų primityvai 7](#_Toc40981106)

[5. Resursai 8](#_Toc40981107)

[Bendras resursų aprašymas 8](#_Toc40981108)

[Resursų primityvai 8](#_Toc40981109)

[Resurso paskirstytojas 9](#_Toc40981110)

[6. Bazinių procesų paketas 10](#_Toc40981111)

[7. Procesas StartStop 11](#_Toc40981112)

[8. Procesas ReadFromInterface 12](#_Toc40981113)

[9. Procesas JCL 13](#_Toc40981114)

[10. Procesas MainProc 14](#_Toc40981115)

[11. Procesas JobGorvernor 15](#_Toc40981116)

[12. Procesas VirtualMachine 16](#_Toc40981117)

[13. Procesas Interrupt 17](#_Toc40981118)

[14. Procesas PrintLine 18](#_Toc40981119)

[15. Procesas MemoryGovernor 19](#_Toc40981120)

[Realios mašinos centrinis procesorius 20](#_Toc40981121)

[Realios mašinos atmintys 21](#_Toc40981122)

[Duomenų perdavimo kanalai 21](#_Toc40981123)

[Įvedimo/Išvedimo įrenginiai 21](#_Toc40981124)

[Taimeris 21](#_Toc40981125)

[Pertraukimų mechanizmas 21](#_Toc40981126)

[16. Virtuali mašina 22](#_Toc40981127)

[Virtualios mašinos modelis 22](#_Toc40981128)

[Virtualios mašinos centrinis procesorius 23](#_Toc40981129)

[Virtualios mašinos atmintis 23](#_Toc40981130)

[Virtualios mašinos procesoriaus komandos 24](#_Toc40981131)

[Bendravimas su įvedimo / išvedimo įrenginiais. 26](#_Toc40981132)

[17. Virtualios mašinos interpretuojamojo ar kompiliuojamojo vykdomojo failo išeities teksto formatas. 26](#_Toc40981133)

[18. Puslapiavimo mechanizmas 27](#_Toc40981134)

[19. Kaip virtuali mašina įsivaizduojama visos operacines sistemos kontekste. 27](#_Toc40981135)

[20. Remtasi literatūra 28](#_Toc40981136)

# Užduoties sąlyga

Projektuojama interaktyvi OS. (Variantas A)

Virtualios mašinos procesoriaus komandos operuoja su duomenimis, esančiais registruose ir ar atmintyje. Yra komandos duomenų persiuntimui iš atminties į registrus ir atvirkščiai, aritmetinės (sudėties, atimties, palyginimo), sąlyginio ir besąlyginio valdymo perdavimo, įvedimo, išvedimo, darbo su bendra atminties sritimi (prieinama visoms vartotojo programoms; komandos leidžia į ją rašyti ir skaityti; sritis apsaugoma semaforais) ir programos pabaigos komandos. Registrai yra tokie: komandų skaitiklis, bent du bendrosios paskirties registrai, požymių registras (požymius formuoja aritmetinės, o į juos reaguoja sąlyginio valdymo perdavimo komandos). Atminties dydis yra 16 blokų po 16 žodžių (žodžio ilgį pasirinkite patys).

Realios mašinos procesorius gali dirbti dviem režimais: vartotojo ir supervizoriaus. Virtualios mašinos atmintis atvaizduojama į vartotojo atmintį naudojant puslapių transliaciją. Yra taimeris, kas tam tikrą laiko intervalą generuojantis pertraukimus. Įvedimui naudojama klaviatūra, išvedimui - ekranas. Yra išorinės atminties įrenginys - kietasis diskas.

Vartotojas, dirbantis su sistema, programas paleidžia interaktyviai, surinkdamas atitinkamą komandą. Laikoma, kad vartotojo programos yra realios mašinos kietajame diske, į kurį jos patalpinamos „išorinėmis“, modelio, o ne projektuojamos OS, priemonėmis.

# Procesai

**Procesas** - tai vykdoma programa, kartu su esamomis registrų reikšmėmis ir savo kintamaisiais. Kiekvienam procesui sukuriamas atskiras virtualus procesorius. Programa nuo proceso skiriasi tuo, kad procesas – tai kokioje nors veiklumo stadijoje esanti programa, o programa – tik tam tikras baitų rinkinys. Veiklumo stadiją apibūdina deskriptorius. Jame laikomi visi procesui reikalingi parametrai – virtualaus procesoriaus registrų reikšmės, jam reikalingi kintamieji.

Sistemos veikimo eigoje turi būti vienas pagrindinis procesas, iš kurio seks likusieji.

Procesai gali būti sisteminiai arba vartotojo. Sisteminiai procesai, atlieka su MOS darbu susijusias funkcijas, o vartotojo rūpinasi vartotojo užduoties atlikimu.

Procesų būsenos

Procesas gauna procesorių tik tada, kai jam netrūksta jokio kito resurso. Gavęs procesorių jis tampa vykdomu. Šioje būsenoje jis turi procesorių tol, kol sistemoje neįvyksta pertraukimas arba einamasis procesas nepaprašo kokio nors resurso.Procesas blokuojasi priverstinai. Taip pat, jei procesas nereikalauja jokio resurso, iš jo gali būti atimtas procesorius, nes per ilgai dirbo. Procesų būsenos:

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

Diagrama, vaizduojanti, kaip procesas gali pakliūti i tam tikrą būseną ir iš jos išeiti:

Vykdomas

Blokuotas

Blokuotas sustabdytas

Pasiruošęs sustabdytas

Pasiruošęs

(4)

(2)

(1)

(7)

(8)

(9)

(5)

(6)

(3)

Kaip matyti iš diagramos yra devyni perėjimai. Juos trumpai pakomentuosime:

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 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.

# Planuotojas

Planuotojas – centrinio procesoriaus resursų paskirstytojas.

Planuotojo užduotis – skirstyti procesorių. Atimti procesorių iš vienų procesų ir duoti jį 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 varotojams.

Planuotojas kviečiamas, kai norima procesorių perduoti kitam procesui. Jo veiksmų seka galima būtų pavaizduoti tokia diagrama:

Ne

Ne

Yra pasiruošusių procesų?

Einamasis procesas blokuotas?

Taip

Taip

Einamasis procesas įtraukiamas į blokuotų proc. sąrašą

Imamas pirmas pasiruošęs procesas

Perduodamas valdymas šiam procesui. T.y. tik dabar naujas procesas tampa einamuoju

Vykdomas procesas

# Procesų primityvai

**Procesų primityvų paskirtis** – pateikti vienodą ir paprastą vartotojo sąsają darbui su procesais. Darbui su procesais skirti 4 primityvai:

1. **Kurti procesą**. Šiam primityvui perduodama nuoroda į jo tėvą, jo pradinė būsena, prioritetas, perduodamų elementų sąrašas ir išorinis vardas. Pačio primityvo viduje vyksta proceso kuriamasis darbas. Jis yra registruojamas bendrame procesų sąraše, tėvo-sūnų sąraše, skaičiuojamas vidinis identifikacijos numeris, sukuriamas jo vaikų procesų sąrašas (tuščias), sukurtų resursų sąrašas ir t.t.

2. **Naikinti procesą.** Pradedama naikinti proceso sukurtus resursus ir vaikus. Vėliau išmetamas iš tėvo sukurtų procesų sąrašo. Toliau išmetamas iš bendro procesų sąrašo ir, jei reikia, iš pasiruošusių procesų sąrašo. Galiausiai naikinami visi jam perduoti resursai ir proceso deskriptorius yra sunaikinamas.

3. **Stabdyti procesą.** Keičiama proceso būsena iš blokuotos į blokuotą sustabdytą arba iš pasiruošusios į pasiruošusią sustabdytą. Einamasis procesas stabdomas tampa pasiruošusiu sustabdytu.

4. **Aktyvuoti procesą.** Keičiama proceso būsena iš blokuotos sustabdytos į blokuotą, ar pasiruošusios sustabdytos į pasiruošusią. Pastaba: Procesai labai aktyviai naudojasi resurso primityvais “prašyti resurso” ir “atlaisvini resursą”. Nereikia jų painioti su procesų primityvais.

Kiekvieno primityvo programos gale yra kviečiamas planuotojas.

# Resursai

Bendras resursų aprašymas

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

1. • **Statinius resursus**. Kuriami sistemos kūrimo metu. Tai mašinos resursai, tokie kaip procesorius, atmintis ar kiti resursai, kurie sistemos veikimo metu nėra naikinami.
2. • **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.

Resursų primityvai

1. **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.
2. **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.
3. **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.
4. **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.

|  |  |  |
| --- | --- | --- |
| Pavadinimas | Kas sukuria/ sunaikiną procesą | Paskirtis |
| StartStop | Sukuriamas ir naikinamas  automatiškai kartu su sistema | Sukurti sisteminius procesus bei resursus, pasirūpinti korektiška sistemos darbo pradžia ir pabaiga |
| ReadFromInterface | StartStop | užduoties nuskaitymui iš įvedimo srauto („flash atmintinės“) |
| JCL | StartStop | suskaidyti užduoties programą blokais ir organizuoti juos kaip resursus; |
| MainProc | StartStop | valdyti JobGovernor procesus |
| JobGovernor | MainProc | virtualios mašinos proceso tėvas. valdyti virtualios mašinos proceso darbą |
| VirtualMachine | JobGovernor | atsakyti už vartotojiškos programos vykdymą |
| Interrupt | StartStop | apdoroti virtualios mašinos pertraukimą sukėlusią situaciją |
| PrintLine | StartStop | Persiųsti į išvedimo įrenginį eilutę (spausdintuvą) iš supervizorinės atminties |
| MemoryGovernor | StartStop | Valdo bendraja atminti |

Resurso paskirstytojas

Kaip kad procesorius yra skirstomas planuotojo, kiekvienas resursas taipogi yra skirstomas tam tikro paskirstytojo. Prašydamas resurso ar norėdamas jį atlaisvinti, procesas kreipiasi į atitinkamą resursų paskirstytoją, kuris privalo jį aptarnauti.

Resursų paskirstytojo paskirtis - suteikti paprašytą resurso elementų kiekį procesui. Resursų paskirstytojo algoritmas gali būti sudėtingas. Pavyzdžiui, turi būti numatyta galimybė atiduoti resurso elementą konkrečiam procesui, arba prašyti kelių resurso elementų. Resurso paskirstytojas peržvelgia visus laukiančius šio resurso procesų sarašą ir sutikęs galimybę aptarnauti procesą perduoda jam reikalingus resurso elementus ir pažymi jį pasiruošusiu. Paskirstytojo pabaigoje yra iškviečiamas planuotojas.

# Bazinių procesų paketas

Mūsų modelyje bus naudojami šie procesai:

* **StartStop** – šakninis procesas, sukuriantis bei naikinantis sisteminius procesus ir resursus;
* **ReadFromInterface** – užduoties nuskaitymo iš įvedimo srauto procesas
* **JCL** – užduoties programos, jos antraštės išskyrimas iš užduoties, ir jų organizavimas kaip resursų
* **MainProc** – procesas, valdantis JobGovernor procesus;
* **JobGovernor** – virtualios mašinos proceso tėvas, valdantis virtualios mašinos proceso darbą;
* **VirtualMachine** – procesas, atsakantis už vartotojiškos programos vykdymą;
* **Interrupt** – procesas, apdorojantis virtualios mašinos pertraukimą sukėlusią situaciją;
* **PrintLine** – į išvedimo įrenginį persiunčiamas blokas iš vartotojo atminties;
* **MemoryGovernor** – procesas atsakingas uz bendra atminti.

Visi procesai, išskyrus VirtualMachine (šį procesą sukuria JobGovernor) ir JobGovernor, yra sukuriami sistemos darbo pradžioje proceso StartStop.

Bendra procesų schema iš vartotojo užduoties pusės atrodytų taip:

ReadFromInterface

JCL

MainProc

JobGovernor

VirtualMachine

JobGovernor

VirtualMachine

Interrupt

.............

# Procesas StartStop

Šis procesas atsakingas už sistemos darbo pradžią ir pabaigą. Įjungus kompiuterį šis procesas pasileidžia automatiškai. Šio proceso paskirtis – sisteminių procesų ir resursų kūrimas.

**START**

**STOP**

Sisteminių permantinių procesų inicializacija (2)

Blokavimasis laukiant „MOS pabaiga“ resurso (3)

Sisteminių procesų naikinimas (4)

Sisteminių resursų naikinimas (5)

Sisteminių resursų inicializacija (1)

* Procesas, gavęs procesorių, savo darbą pradeda sukurdamas visus sisteminius resursus (1).
* Sukūręs resursus, procesas kuria nuolatinius procesus (2), t.y. tokius procesus, kurie bus aktyvūs visą MOS gyvavimo laiką (visi, išskyrus JobGovernor ir VirtualMachine).
* Prašo resurso „MOS pabaiga“ (3), t.y. šioje vietoje procesas blokuojasi ir laukia, kol bus atlaisvintas pranešimas apie MOS darbo pabaigą. Priklausomai nuo prioriteto šis procesas bus atblokuotas ir tęs darbą.
* Visų sisteminių procesų naikinimas (4).
* Sisteminių resursų naikinimas (5). MOS darbo pabaiga.

# Procesas ReadFromInterface

Šį procesą kuria ir naikina procesas StartStop. Šio proceso paskirtis – gavus informaciją iš įvedimo srauto ir atlikus pirminį jos apdorojimą, atiduoti informaciją tolesniam apdorojimui, kurį atliks procesas JCL.

Blokavimas laukiant „Iš vartotojo sasajos“ resurso (1)

Duomenų nuskaitymas ir suskaidymas blokais (2)

Blokavimasis laukiant “Supervizorinė atmintis“ resurso (3)

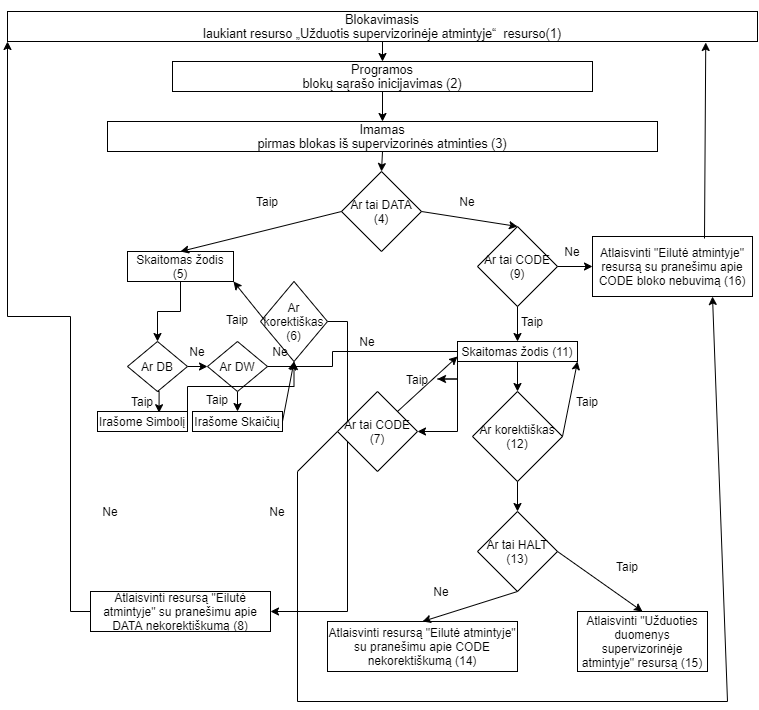
Duomenų įrašymas į supervizorinę atmintį (4)

Sukuriamas ir atlaivinamas resursas „Užduotis supervizorinėje atmintyje“. Skirta procesui JCL (5)

1. Pirmasis žingsnis – laukti įvedimo srauto (1)
2. Skaitymas ir skaidymas blokais (po 16 baitų) (2).
3. Apdorojimo rezultatas – blokų sąrašas yra laikomas supervizorinėje atmintyje, todėl mes jos prašome (3) ir kopijuojame blokus į atmintį (4).
4. Yra sukuriamas ir atlaisvinamas resursas, skirtas procesui JCL, kuriame yra informacija apie blokų padėtį atmintyje (5).
5. Proceso programa yra ciklinė, todėl ReadFromInterface vėl užsiblokuoja laukdamas.

# Procesas JCL

Procesą JCL kuria ir naikina procesas StartStop. Proceso JCL paskirtis – gautus blokus iš proceso ReadFromInterface suskirstyti į programos blokus, ir atidavus procesui MainProc, vėl blokuotis laukiant pranešimo iš ReadFromInterface.



Trumpas paaiškinimas:

1. Procesas pradeda savo darbą blokuodamasis, laukdamas pranešimo iš ReadFromInterface (1).
2. Sulaukus šio pranešimo, procesas pasiruošia darbui ir inicijuoja programos blokų sąrašą (2).
3. Imamas pirmas blokas (3).
4. Jei pirmi 4 baitai yra „DATA”(4) (Data segment) prijungiamas blokas prie programos sąrašo (išskyrus pirmus 4 baitus, kuriuose talpinamas „DATA“). Ir toliau skaitome žodžius/blokus ir jei jie korektiški jungiame juos prie programos sąrašo. Jei jie nekorektiški einame į (8) žingsnį. Taip darome kol sutinkame CODE tada einame į (11) žingsnį, jei jo nesutinkame einame į (16) žingsnį.
5. Jei pirmi 4 baitai yra „CODE“(9). (Code segment) skaitome toliau einančius žodžius/blokus kaip 11 žingsnyje, jei ne elgiamės kaip (16) žingsnįje.
6. Skaitome „CODE” dalį kol sutinkame „HALT”.
7. Pasiekus „HALT“ atlaisvinamas resursas „Užduoties vykdymo parametrai supervizorinėje atmintyje“ (tai sisteminis resursas), skirtas MainProc procesui su nuoroda į programos blokų sąrašą (9). Toliau JCL blokuojasi laukdamas pranešimo iš ReadFromInterface (1).

Taigi, apibendrinimui galima pasakyti, jei programa parašyta nekorektiškai, tai ji nebus vykdoma, nes ji paprasčiausiai nebus saugoma jokioje atmintyje.

# Procesas MainProc

Procesą MainProc kuria ir naikina procesas StartStop. Šio proceso paskirtis - kurti ir naikinti procesus.

Blokavimasis laukiant užduoties, kurią reikia įvykdyti (1)

Kuriamas procesas „JobGovernor“ ir jam perduodamas resursas “Užduoties duomenys supervizorineje atmintyje“ (2)

Naikinamas procesas „JobGovernor“ (3)

Vykdymo laikas=0?

NE

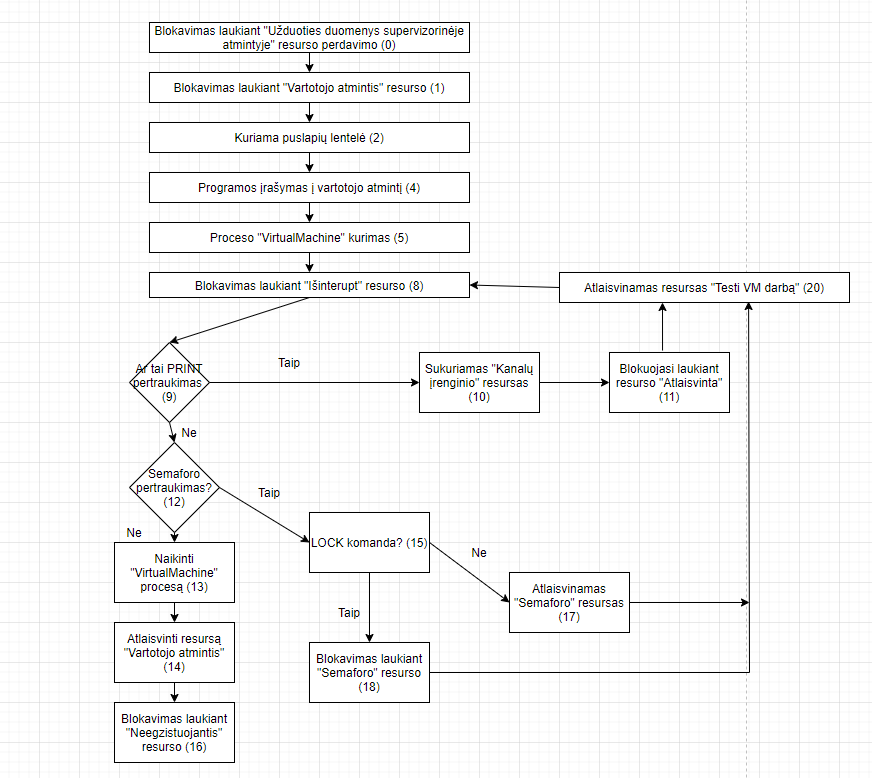
Taip

Trumpas paaiškinimas:

1. Procesas prašo užduoties, kurią reikia įvykdyti (1).
2. Tikrinama ar resurso vykdymo laikas nelygus 0, t.y. ar užduotis nevykdoma per ilgai.
3. Jei laikas nelygus nuliui - kuriamas procesas „JobGovernor“, priešingu atveju - naikinamas procesas „JobGovernor“, kuris atsiuntė pradinį resursą su nuliniu vykdymo laiku (2, 3).
4. Atlikęs savo darbą procesas MainProc vėl blokuojasi laukdamas resurso (1).

# Procesas JobGorvernor

Procesus (jų gali būti keli) kuria procesas MainProc. Šių procesų paskirtis – padėti procesui VirtualMachine atlikti savo darbą (atlikti veiksmus, kurių virtuali mašina procesoriui dirbant vartotojo režimu nesugeba atlikti).

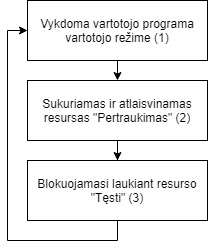


Trumpas paaiškinimas:

1. Blokavimas laukiant „Užduoties duomenys supervizorinėje atmintyje“ resurso perdavimo
2. Procesas prašo vartotojo atminties, kad galėtų patalpinti vartotojo užduoties programą (1)
3. Procesas prašo vieno takelio virtualios mašinos puslapių lentelei. Lentelė užpildoma išskirtos vartotojo atminties adresais (2).
4. Į vartotojo atmintį įrašomas programa ir sukuriamas procesas „VirtualMachine“ (4) ir (5).
5. Procesas JobGovernor blokuojasi ir laukia pertraukimo proceso VirtualMachine vykdymo metu (8).
6. Gavus pranešimą apie pertraukimą (vykdomos komandos PRINT,HALT) stabdomas procesas VirtualMachine, tikrinama ar tai kokie tai pertraukimai (9)(12).
7. Jei tai PRNT (išvedimas) pertraukimas, sukuriamas „Kanalų įrenginio resursas“ siganlizuojantis, kad yra siunčiami išvedimui skirti blokai, esantys vartotojo atmintyje.
8. Atlaisvina resursą „Tęsti VM darbą“ , kad virtuali mašina vykdytų vartotojo programą .
9. Vėliau aktyvuojamas procesas „VirtualMachine“ ir JobGovernor cikliškai grįžta blokuotis bei laukti „Interrupt“ pranešimo

# Procesas VirtualMachine

Procesą kuria ir naikina procesas JobGovernor. Proceso VirtualMachine paskirtis yra vykdyti vartotojo užduoties programą. Šių procesų yra tiek kiek yra procesų JobGovernor.

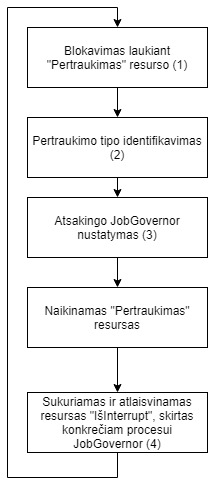


Trumpas paaiškinimas:

1. Interpretuojama programa, kol neįvyksta pertraukimas (1).
2. Įvykus pertraukimui, virtuali mašina išsaugo savo procesoriaus būseną, valdymas perduodamas pertraukimą apdorosiančioms programoms. Kuriamas resursas „Pertraukimas“ skirtas procesui Interrupt, kuris indentifikuos pertraukimą ir perduos informaciją procesui JobGovernor (2), ir laukiama leidimo tęsti vykdymą (pranešimo apie įvykdytą veiksmą iš įvedimo/išvedimo procesų).

# Procesas Interrupt

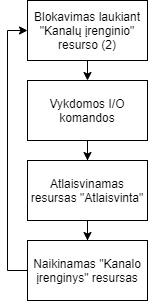
Procesą Interrupt kuria ir naikina procesas StartStop. Šio proceso paskirtis – reaguoti į pertraukimus, kilusius virtualios mašinos darbo metu.



* 1. Procesas Interrupt savo darbo pradžioje laukia „Pertraukimas“ resurso, kurį siunčia procesas VirtualMachine (1).
  2. Procesas nustato pertraukimo tipą apklausinėdamas pertraukimo programų nustatytas sisteminių kintamųjų reikšmes (2).
  3. Toliau procesas cikliškai blokuojasi vėl laukdamas „Pertraukimas“ resurso. Kiekviena virtuali mašina kūrimo metu laiko savo tėvo JobGovernor identifikatorių. Kartu su pranešimu apie pertraukimą yra perduodamas ir tas identifikatorius, kurį Interrupt naudoja jam reikalingo JobGovernor atskyrimui iš kitų (3).
  4. Galiausiai yra kuriamas ir atlaisvinamas „IšInterrupt“ resursas, kuris yra skirtas nustatytajam JobGovernor procesui (4).

# Procesas PrintLine

Procesą kuria ir naikina StartStop. Šio proceso paskirtis – siųsti listingo eilutes iš supervizorinės atminties į spausdinimo įrenginį t.y. generuoti išvedimo srautą.



# Procesas MemoryGovernor

MemoryGovernor ***-*** procesas valdantis bendrąją atminti.

Blokavimas laukiant pranešimo iš proceso JobGovernor

Ar tai įrašymas į bendrą atmintį?

Atlaisvina nuskaitytą bendrą atmintį

Sukuria pranešimą, kad įrašė į bendrą atmintį

Tikriname ar dar turime laisvos bendros atminties

Sukuria pranešimus su nuskaitytais duomenimis

Sukuria pranešimą, kad nėra laisvos bendros atminties

Rezervuojame bendrą atmintį ir ją įrašom

Ne

Taip

Taip

Ne

Realios mašinos centrinis procesorius

* + - **Centrinis procesorius –** skaito komandą iš atminties ir ją vykdo (interpretuoja).

Gali dirbti dviem rėžimais: vartotojo arba supervizoriaus.

* + - * Vartotojo rėžime HLP vykdo tam tikrą užduoties programą. Yra imituojamas virtualios mašinos procesorius ir prieinama prie vartotojo atmintyje esančių programų per puslapiavimo mechanizmą.
      * Supervizoriaus rėžime komandos iš supervizorinės atminties yra betarpiškai apdorojamos HLP.

HLP – bet kuris aukšto lygio kalbos procesorius (programavimo kalba).

Centrinis procesorius turi:

**Nuorodų registrai:**

* **PLR** – puslapiavimo registras, naudojamas atvaizdavimui tarp virtualios ir realios mašinos. 4 baitų registras.
* **IC** – 2 baitų virtualios mašinos komandų skaitliukas

**Duomenų registrai**:

* **R, D** – 4 baitų registrai operacijų darbui

**Loginiai registrai**:

* **C** – 1 baito loginis registras, gali įgyti reikšmes TRUE arba FALSE.

**Darbo režimo registrai**:

* **MODE** – registras, kurio reikšmė nusako procesoriaus darbo režimą (vartotojas, supervizorius).

**Pertraukimo registrai**:

* **PI** – programinio pertraukimo registras, signalizuoja, kad virtualioje mašinoje atrastas neegzistuojantis operacijos kodas arba adresas. 2 baitų registras.
* **SI** – supervizoriaus pertraukimo registras. Kai virtuali mašina negali įvykdyti veiksmo pati. Ji kreipiasi į OS servisą, virtuali mašina tik nustato pertraukimo registrą, o OS turi atlikti tą veiksmą. 2 baitų registras.
* **TI** – taimerio pertraukimo registras, nustato pertraukimą, kai pasiekiama tam tikra reikšmė. 2 baitų registras.

**Kanalū registras**:

* **CHN** – 3 baitų kanalų valdymo registras.

**Semaforų registrai(bendra atmintis)**:

* **SM** – registras rodantis į bendrą atmintį vartotojo atmintyje. 2 baitų registras.
* **S** – 2 baitų registras, kurio kiekvienas bitas nurodo ar atitinkamo skaičiaus ląstelė yra užimta. Pavyzdžiui, jei S yra 1000000000000001, tai pirmoji ir šešiolikta ląstelė yra užrakintos ir jei virtuali mašina bandytų užrakinti kažkurią iš tų ląstelių, tai ji turėtų laukti, kol ta ląstelė atsirakins, o kitos ląstelės yra atrakintos.

Realios mašinos atmintys

* + - * Vartotojo – skirta virtualių mašinų atmintims bei puslapių lentelių laikymui. Mes apibrėšime vartotojo atmintį taip: lentelės dydis – 4096 žodžių po 4 baitus. 16 žodžių laikysime bloku (takeliu). Taigi vartotojo atmintis lygi 256–čiai blokų, sunumeruotų nuo 0 iki 255, arba 4096 žodžių, sunumeruotų nuo 0 iki 4095.
      * Supervizorinė – trumpai tariant tai atmintis, kurios reikia pačios OS poreikiams (komandos, sisteminiai kintamieji ir pan.). Visa tai valdys HLP.
      * Išorinė atmintis – mūsų atveju tai kietasis diskas. Darbą su išorine atmintimi atliks HLP. Procesorius darbą su atmintimis valdo naudodamas kanalų įrenginį.

Duomenų perdavimo kanalai

Duomenų perdavimo kanalaiskirti darbui su atmintimi ir įvedimo/išvedimo įrenginiams valdyti. Jie reikalingi tam, kad centrinis procesorius būtų išlaisvintas nuo lėtų išvedimo ir įvedimo įrenginių. Yra trys kanalai, kuriais „bendraujama“:

* 1 – jungia klaviatūrą su vartotojo atmintimi.
* 2 – jungia ekraną su vartotojo atmintimi.
* 3 – jungia kietąjį diską su supervizorine bei vartotojo atmintim.

Įvedimo/Išvedimo įrenginiai

Įvedimo/išvedimo įrenginiai atitinkamai yra naudojami nuskaityti vartotojo įvestas komandas bei išvesti rezultatą. Įvedimui naudojama klaviatūra, išvedimui – ekranas.

Taimeris

Skirtas užduotims suderinti. Yra sakoma, kad užduotis negali trukti ilgiau nei tam tikrą T laiko momentų. Susitarsime, jog išvedimo / įvedimo operacijos reikalauja 3 laiko momentų, o kitos 1 ar 2.

Taigi kai VM pradeda darbą, speciali supervizorinės atminties ląstelė TI yra nustatoma tam tikrai reikšmei. Tarkime 15, tuomet kai yra įvykdoma instrukcija, TI yra mažinamas priklausomai nuo to kiek laiko momentų reikia instrukcijai. Kai TI tampa lygus nuliui, įvyksta pertraukimas. TI gali būti nustatytas supervizoriaus režime.

Pertraukimų mechanizmas

Išskirsime tryjų rūšių pertraukimus:

* + - * Programiniai, kurių registras yra PI. Galimi atvejai:
        + PI = 1 – atminties apsaugos pažeidimas.
        + PI = 2 – blogas operacijos kodas.
        + PI = 3 – neteisingas priskyrimas.
      * Supervizoriniai, kurių registras SI. Galimi atvejai:
        + SI = 1 – komanda GD
        + SI = 2 – komanda PD
        + SI = 3 – komanda HALT
        + SI = 4 – komanda BM
        + SI = 5 – komanda UM
        + SI = 6 – komanda SI
        + SI = 7 – komanda PI
      * Taimerio, kurio registras TI. Galimi atvejai:
      * TI = 0 – taimerio skaitliukas lygus 0.

# Virtuali mašina

Virtuali mašina (VM) tai realios mašinos modelis, kuris veikia kaip tam tikras tarpininkas. Ji smarkiai supaprastina tiek ir programų rašymą tiek ir pačią realizaciją. VM pagrindinė paskirtis vykdyti vartotojo programą.

Virtualios mašinos modelis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | 000 |
|  |  |  |  | 001 |
|  |  |  |  | 002 |
|  | | | | |
|  |  |  |  | 253 |
|  |  |  |  | 254 |
|  |  |  |  | 255 |

Virtualios mašinos schema:

Virtualus

įvedimo

įrenginys

R

IC

Virtualus procesorius

C

Virtualus

išvedimo

įrenginys

D

Virtualios mašinos centrinis procesorius

Virtualios mašinos procesoriaus paskirtis – vykdyti programą, kuri yra virtualioje atmintyje. Kiekvienas procesas turi savo virtualų centrinį procesorių, tačiau sisteminių procesų programas vykdo aukšto lygio kalbos procesorius.

Virtualaus procesoriaus registrai:

* R – 4 baitų bendro naudojimo registras;
* D – 4 baitų bendro naudojimo registras;
* IC – 2 baitų virtualios mašinos komandų skaitliukas;
* C – 1 baito loginis trigeris, kuris gali būti TRUE arba FALSE.

Virtualios mašinos atmintis

Kiekvienai virtualiai mašinai yra skiriama 16 vartotojo atminties blokų. Tuose šešiolikoje blokų (256 žodžių po 4 baitus) turi tilpti užduoties programa. Kiekvienas virtualios atminties blokas turi virtualų ir realų adresą. Virtualiais adresais operuoja virtuali mašina, realiais – reali mašina. Ryšiai tarp virtualaus ir realaus adreso nusakomi puslapių lentelėmis.

Virtualios atminties pvz:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | GD50 | LR60 | AD50 | SB00 | LD81 | CD00 | SR82 | PD90 | HALT |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 5 | 100 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 6 | 200 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 7 | 150 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 8 |  | 600 | 150 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 | Susk | aici | uota |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| B |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| C |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| F |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Virtualios mašinos procesoriaus komandos

Virtualios mašinos procesoriaus komandos:

**1) Aritmetinės komandos:  
 AD –** sudėti registrą R ir atminties lastelę

**AD x1 x2 :**

**R := R + [ 16 \* x1 + x2 ]**

**DA –** sudeda registrus R ir D

**DA x1 x2 :**

**R:= R + D  
 SB –** atima iš registro R reg D

**SB x1 x2**

**R : = R – D**

**BS** – atimti iš registro R atminties lastelę

**BS x1 x2 :**

**R:=R - [ 16 \* x1 + x2 ]**

**2) Palyginimo komandos:**

**CR –** Dviejų registrų palyginimas ( jei R daugiau už D ).

**CR x1 x2 :**

**If R > D then C := ‘T’ else C : = ‘F’**

**CD –** Dviejų registrų palyginimas (jei R lygus D)

**CD x1x2  :**

**If R = D then C := ‘T’ else C : = ‘F’**

**3) Darbui su atmintimi  
 LD –** pakrauti duomenų registrą

**LD x1 x2 :**

**D := [ 16 \* x1 + x2 ]**

**LR –** pakrauti rezultatų registrą

**LR x1 x2 :**

**R := [ 16 \* x1 + x2 ]**

**SR –** išsaugoti rezultatų registrą

**SR x1 x2 :**

**[ 16 \* x1 + x2 ]:= R**

**SD –** išsaugoti duomenų registrą

**SD x1 x2 :**

**[ 16 \* x1 + x2 ]:= D**

**4) Įvedimo išvedimo komandos  
 GD -** get data

**GD x1x2 :**

Iš įvedimo srauto perskaito 16 žodžių ir įrašo juos į ląsteles [x2 \*16 + i ], kur i = 0.. 15. Operandas “ x1” reikšmės neturi.

**PD** – put data

**PD x1 x2 :**

Išsiunčia išvedimui 16 žodžių srautą iš atminties ląstelių [x2 \*16 + i ], kur i = 0.. 15. Operandas “ x1” reikšmės neturi.

**PI –** atspausdinti skaičių iš atminties ląstelės

**PI x1 x2**

**SI –** išsaugoti skaičių iš įvedimo į atminties ląstelę.

**SI x1 x2**

**5) Valdymo perdabvimo komandos**

**JP x1x2 :**

Peršokti į atminties adresą xy (IC = 16\*x+y).

**JP x1x2 :**

Peršokti į atminties adresą xy (IC = 16\*x+y), jei C = T, jei ne – einam toliau.

**6) Darbo su bendra atminties sritimi komandos**

**BA -** Iš registro R įrašo į bendrą atmintį. Operandas “ x1” reikšmės neturi.

**BA x1x2**

**BR -** Iš bendros atminties įrašo į registrą R. Operandas “ x1” reikšmės neturi.

**BR x1x2**

Jei bandoma nuskaityti arba įrašyti į neužrakintą ląstelę, tai nustatoma PI = 1 – atminties apsaugos pažeidimas.

**7) Semaforų komandos**

**BM -** Užblokuojama **x2**-oji bendros atminties ląstelė. Operandas “ x1” reikšmės neturi.

**BM x1x2**

**UM -** Atblokuojama **x2**-oji bendros atminties ląstelė. Operandas “ x1” reikšmės neturi.

**UM x1x2**

Vykdant BM, nustatomas registras SI = 4, operacinė sistema patikrina ar **x2**-oji ląstelė laisva, ir jei laisva, tai nustato **x2**-ajį S registro bitą į 1 ir perduoda vykdymą atgal virtualiai mašinai, o jei užblokuota, tai laukia, kol atsilaisvins ir tik tada perduoda.

Vykdant UM, nustatomas registras SI = 5, operacinė sistema atlaisvina **x2**-ąją ląstelę nustatydamas **x2**-ajį S registro bitą į 0.

**7) Programos pabaigos komanda**

**HALT –** programos vygdymo pabaiga.

Bendravimas su įvedimo / išvedimo įrenginiais.

Vartotojo užduoties įvedimo / išvedimo operacijos atliekamos supervizoriaus rėžime. Ar kanalai užimti nustatoma per ju registrus CH1,CH2,CH3.

Įvedimo įrenginys – klaviatūra. Turi 16 žodžių buferį. Užsipildžius buferiui jo turinys perkialiamas į atmintį. Įvedimas vyksta 1 kanalu.

Išvedimo įrenginys – ekranas. Turi 16 žodžių buferį. Spausdina jį perduodant informaciją iš atminties 2 kanalu.

# Virtualios mašinos interpretuojamojo ar kompiliuojamojo vykdomojo failo išeities teksto formatas.

DATA

....

Bendras užduoties pavidalas bus iš šių dalių:

CODE

....

* Programos duomenys (DATA)
* Programos kodas (CODE)

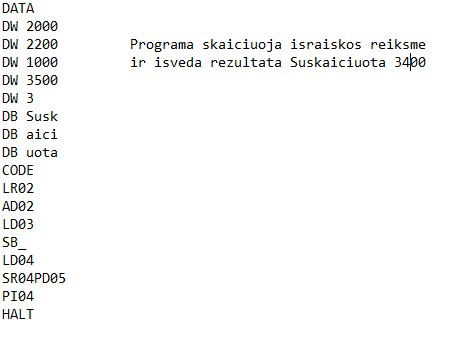
HALT

* Pabaigos žymė (HALT)

Kodo pavyzdys:

DW XXXX – Išskiriamas vienas žodis ir į jį talpinama nurodyta skaitinė reikšmė.

DB SSSS – Išskiriamas vienas žodis ir į jį talpinami keturi nurodyti simboliai.



# Puslapiavimo mechanizmas

Realios mašinos vartotojo atmintis siekia 256 takelių (arba blokų). Kiekvienai naujai sukurtai virtualiai mašinai reikia skirti 16 takelių iš tų 256. Jie gali būti parinkti bet kokiu būdu.

Sakykime, kuriama nauja virtuali mašina. Jai reikia šešiolikos takelių atminties. Mes parinkome takelius su numeriais: 2, 6, 8, 9, 13, 35, 49, 68, 69, 70, 100, 123, 180, 181, 199, 240. Šiais takeliais naudosis virtuali mašina. Pati virtuali mašina šiuos takelius mato sunumeruotus nuo 0 iki 15, t.y. 2 takelis jai yra nulinis, 6 takelis jai yra pirmas, o 240 takelis – penkioliktas. Tam, kad išlaikyti sąryšį tarp realiu ir virtualių adresų naudosime puslapiu lentele. Puslapių lentelė – tai vienas takelis (t.y. 16 žodžių mūsų atveju). Kiekvieno žodžio eilės numeris atitiks virtualios mašinos takelio numerį, ir jame (žodyje) bus laikomas realus to takelio numeris. Puslapių lenteles pvz:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Virtualus takelio numeris** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| **Realus takelio numeris** | 2 | 6 | 8 | 9 | 13 | 35 | 49 | 68 | 69 | 70 | 100 | 123 | 180 | 181 | 199 | 240 |

PLR yra 2 baitų ir simboliškai žymėsime taip a1a2. O x1x2 virtualus adresas.

* + - * 16\*a1+a2 – puslapių lentelės bloko numeris vartotojo atmintyje.
      * 16\*(16\*a1+a2) – puslapių lentelės bloko adresas.
      * 16\*(16\*a1+a2) + x1 – bloko x1 adresas puslapių lentelėje. Jame saugomas bloko numeris į kurį atvaizduotas yra x1 blokas VM.
      * 16\*[16\*(16\*a1+a2) + x1] – VM bloko x1 realus bloko adresas.
      * 16\*[16\*(16\*a1+a2) + x1] + x2 – realus adresas atitinkantis virtualų adresą x1x2.

# Kaip virtuali mašina įsivaizduojama visos operacines sistemos kontekste.

Virtuali mašina yra realios kopija, kuri veikia kaip tarpininkas. Viruali mašina paslepia realios mašinos realizaciją po virtualiais komponentais. Ji turi savo atmintį, kur kiekvienas blokas turi tiek virtualų tiek realų adresą. Ryšiai tarp šių adresų nusakomi puslapių lentelėmis. Virtuali mašina supaprastina vartotojo sąsają ir vykdo programą, kuri yra virtualioje atmintyje.  
  
Bendrai galėtume pavaizduoti tokia schema:

Reali mašina

Virtuali mašina

Virtuali mašina

Programa

Programa

# Remtasi literatūra

1. A. Mitašiūno „Operacinių sistemų“ konspektai.
2. Internetas.
3. Andrew S. Tanenbaum and Albert S. Woodhull „Operating Systems Design and Implementation“.
4. G.Šiaulio magistro darbas „Mokomoji operacinė sistema“.