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 3](#_Toc475393366)

[1.1. Techninės įrangos komponentai sudarantys realią mašiną: 3](#_Toc475393367)

[1.2. Procesorius 3](#_Toc475393368)

[1.3. Realios mašinos procesorius turi tokius registrus: 4](#_Toc475393369)

[1.4. Atmintys 4](#_Toc475393370)

[1.5. Išvedimo ir įvedimo įrenginiai 4](#_Toc475393371)

[1.6. Puslapiavimo mechanizmas 4](#_Toc475393372)

[1.7. Duomenų perdavimo kanalai 4](#_Toc475393373)

[2. VIRTUALI MAŠINA 5](#_Toc475393374)

[2.1. Pagrindinė operacinės sistemos funkcija 5](#_Toc475393375)

[2.2. Modeliuojamos virtualios mašinos loginių komponentų aprašymas. 5](#_Toc475393376)

[2.2.1 Virtualios mašinos atmintis 5](#_Toc475393377)

[2.2.2 Virtualios mašinos procesorius. 6](#_Toc475393378)

[2.2.3 Virtualios mašinos komandų sistema. 6](#_Toc475393379)

[2.3. Virtualios mašinos bendravimo su įvedimo/išvedimo įrenginiais mechanizmo aprašymas. 8](#_Toc475393380)

[2.4. Virtualios mašinos interpretuojamojo ar kompiliuojamo vykdomojo failo išeities teksto formatas. 8](#_Toc475393381)

[2.5. Modeliuojamos virtualios mašinos loginių komponentų sąryšio su realios mašinos techninės įrangos komponentais aprašymas. 9](#_Toc475393382)

# REALI MAŠINA

Centrinis Procesorius

1 Kanalas

2 Kanalas

3 Kanalas

Klaviatūra  
(įvedimo įrenginys)

Ekranas  
(išvedimo įrenginys)

Kietasis  
diskas  
(išorinė atmintis)

Supervizorinė atmintis

Puslapiavimo mechanizmas

000

255

Vartotojo atmintis

.  
.  
.  
.  
.

APL

IC  
R1  
R2  
C  
DS  
CS

MODE

TI

PI

SI

IOI

PTR

CHST[1]

CHST[2]

CHST[3]

## 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. 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.

## 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
      * 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.

## 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.

## Išvedimo ir įvedimo įrenginiai

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

## 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 a0,a1,a2,a3), kuriame laikomi puslapių lentelės bloko realūs adresai. Naudojant PTR registro baitus a2 ir a3 , surandamas blokas. Pagal virtualaus adreso x1 ir x2 yra nustatomas puslapis. Bloko adresas kur yra x1 randamas pagal formulę [16\*(16\*a2 a3)+x1], o prie bloko adreso pridėjus x2 gauname virtualų adresą: 16\*[16\*(16\*a2 a3)+x1]x2.

## Duomenų perdavimo kanalai

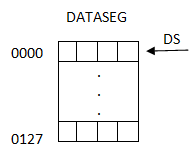
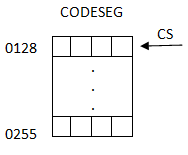
Skirti atminties ir įvedimo/išvedimo valdymui. Yra trys kanalai kiekvienas atsakingas už atitinkamus dalykus. Kanale vyksta tik rašymas arba skaitymas ir baigęs darbą jis informuoja centrinį procesorių ir sukelia pertraukimą padidindamas IOI registro reikšmę. Kanalų būsenos yra saugomos CHST[1] ... CHST[3]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

# VIRTUALI MAŠINA

* 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.

## Modeliuojamos virtualios mašinos loginių komponentų aprašymas.

* + 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:



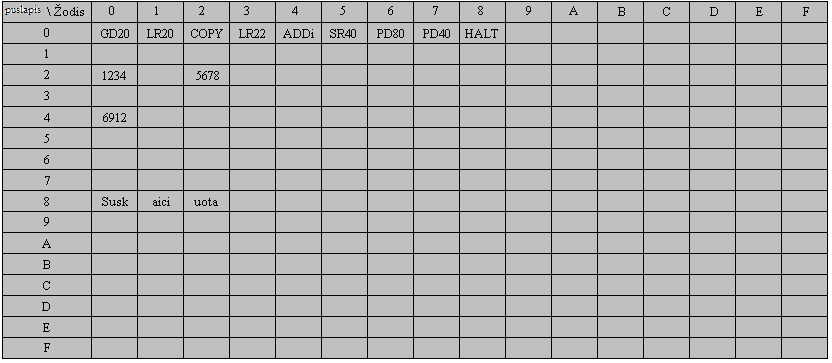
Kur,

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 a0,a1,a2,a3), kuriame laikomi puslapių lentelės bloko realus adresai. Naudojant PTR registro baitus a2 ir a3 surandamas blokas. Pagal virtualaus adreso x1 ir x2 yra nustatomas puslapis. Bloko adresas kur yra x1 randamas pagal formulę [16\*(16\*a2 a3)+x1], o prie bloko adreso pridėjus x2 gauname virtualų adresą: 16\*[16\*(16\*a2 a3)+x1]x2.

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 linked list‘u - kiekviena struktūra vienam, dabar egzistuojančiam, procesui.



* + 1. Virtualios mašinos procesorius.  
       Virtualios mašinos procesoriaus registrai:
       - Komandų skaitliukas:
         1. **IC** – 2 baitų virtualios mašinos programos skaitiklis
       - Bendrosios paskirties registrai:
         1. **R1** – bendrosios paskirties registras
         2. **R2** – bendrosios paskirties registras
       - Požymių registrai:
         1. **C** – 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:
         1. **DS** – Data segment – rodyklė į duomenų segmentą atmintyje.
         2. **CS** – Code segment – rodyklė į kodo segmentą atmintyje.

### 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:
   1. **LR** – Load Register – iš atminties baito x1x2 persiunčia į registrą R:  
      **LR x1x2** =) R:=[x1x2];
   2. **SR** – Save Register – iš registro R persiunčia į atminties baitą x1x2:   
      **SR x1x2** =) [x1x2]:=R;
2. Duomenų sukeitimui tarp registrų:
   1. **RR** – sukeičia registro R ir R2 reikšmes:  
      **RR** =) R:=R+R2, R2=R-R2, R=R-R2;
3. Aritmetinės komandos:
   1. **AD** – suma – prie esamos registro R reikšmės prideda reikšmę esančią x1x2 atminties baite, rezultatas patalpinamas registre R:   
      **AD x1x2** =) R:=R+[x1x2];
   2. **SB** – atimtis – iš esamos registro R reikšmės atimama reikšmė esanti x1x2 atminties baite, rezultatas patalpinamas registre R:   
      **SB x1x2** =) R:=R-[x1x2];
   3. **CR** – palyginimas – esamą registro R reikšmė yra lyginama su reikšme esančią x1x2 atminties baite, rezultatas patalpinamas registre C:   
      **CR x1x2**=)  
       if R>[x1x2] then C:=0;   
      if R=[x1x2] then C:=1;   
      if R<[x1x2] then C:=2;
4. Valdymo perdavimo:
   1. **JM** – besąlyginio valdymo perdavimas – valdymas perduodamas adresu 16\*x1+x2:  
      **JM x1x2**=) IC:=16\*x1+x2;
   2. **JA** – sąlyginio valdymo perdavimas (jeigu daugiau) – valdymas perduodamas jeigu C=0, valdymas perduodamas adresu 16\*x1+x2:   
      **JA x1x2**=) If C=0 then IC:= 16\*x1+x2;
   3. **JE** – sąlyginio valdymo perdavimas (jeigu lygu) – valdymas perduodamas jeigu C=1, valdymas perduodamas adresu 16\*x1+x2:   
      **JE x1x2**=) If C=1 then IC:= 16\*x1+x2;
   4. **JL** – sąlyginio valdymo perdavimas (jeigu mažiau) – valdymas perduodamas jeigu C=2, valdymas perduodamas adresu 16\*x1+x2:   
      **JL x1x2**=) If C=2 then IC:= 16\*x1+x2;
5. Darbo su bendra atminties sritimi (prieinama visoms vartotojo programoms; komandos leidžia į ją rašyti ir skaityti; sritis apsaugoma semaforais):
   1. **SM** – registro R įrašymas į bendrąją atmintį:   
      **SM x1x2** =)[16\*[16\*(16\*a2 a3)+x1]x2]:=R (pagal puslapiavimo mechanizmą);
   2. **LM** – iš bendrosios atminties įrašomas žodis į registrą R:   
      **LM x1x2** =) R:= [16\*[16\*(16\*a2 a3)+x1]x2] (pagal puslapiavimo mechanizmą);
6. Programos pabaigos:
   1. **HALT** – programos pabaigos komanda.
7. Įvedimo/Išvedimo:
   1. **GD** – įvedimas – iš įvedimo srauto paima 1 žodžio srautą ir jį įveda į atmintį pradedant atminties baitu 16\*x1+x2:   
      **GD x1x2**
   2. **PD** – išvedimas – iš atminties, pradedant atminties baitu 16\*x1+x2 paima 1 žodžio srautą ir jį išveda į ekraną:   
      **PD x1x2**

## 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.

* 1. 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 |

## 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.

# 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.

## Procesai

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.

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

start\_stop

2

1

JCL

Read\_from\_interface 2

4

Loader

6

5

Print\_Lines

Lines\_from\_ *hdd*

8

7

Chan3\_Device

Get\_Put\_Data

10

9

Main\_Proc

Interrupt

1. „*Read\_in\_Cards*“ – atsakingas už betarpišką užduoties įvedimą.
2. „*JCL*“ – analizuoja užduoties struktūrą – užduoties atpažinimas.
3. „Loader“ – paruoštas vykdymui užduotis perkelia į vartotojo atmintį.
4. „Lines\_from\_ *hdd*“ – įvykdytos užduotys ir rezultatai kaupiami išorinėje atmintyje. Jų paruošimo išvedimui procesas.
5. „Print\_Lines“ – fizinis eilučių išvedimas.
6. „Get\_Put\_Data“ – užduoties vykdymo metu gali prireikti I/O veiksmų, už tai atsakingas procesas.
7. „Chan3\_Device“ – darbo su tiesioginio priėjimo prie išorinių įrenginių procesas. Tai trečiojo kanalo aptarnavimo procesas.
8. „Interrupt“ – dirbant vartotojo režimu ir vykdant programą gali kilti pertraukimai. Jų signalai turi būti transformuojami į resursus. Tam skirtas „Interrupt“ procesas.
9. „Main\_Proc“ – užduočių atlikimo valdymo sistemos procesas.

## Detalus procesų funkcijų nagrinėjimas

### 6.2.0. Procesas „*Start\_Stop*“

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Sart\_Stop |  |
|  |  | | |
| 1 | Sistemos statinių resursų inicializavimas | | |
|  |  | | |
| 2 | Sisteminių statinių procesų inicializavimas | | |
|  |  | | |
| 3 | Blokavimas laukiant OS darbo pabaigos situacijos | | |
|  |  | | |
| 4 | Statistikos išvedimas | | |
|  |  | | |
| 5 | Sisteminių statinių procesų naikinimas | | |
|  |  | | |
| 6 | Sisteminių statinių resursų naikinimas | | |

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.

1. „*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.
2. „*Statistikos išvedimas*“ – dabar užduoties srautas jau įvykdytas ir sistema baigė darbą. Procesai atsiblokuoja. Statistikos išvedimas.
3. „*Sisteminių statinių procesų naikinimas*“ – veiksmas atliekamas kreipiantis į primityvą „*sunaikinti procesus*“, tai yra sunaikinti deskriptorius.
4. „*Sisteminių statinių resursų naikinimas*“ – kreipiamasi į primityvą „*naikinti resursus*“.

Taip pradedamas ir užbaigiamas sistemos darbas.

### Procesas „*Read\_from\_Interface*“

*Šį procesą kuria ir naikina „Start\_Stop“. „Read\_in\_Cards“* paskirtis: paimti eilutę iš įvedimo srauto, paimti resursą iš supervizorinės atminties ir perkelti eilutę į supervizorinę atmintį. Reikia imti eilutes ir formuoti užduotis. Jas ima „*JCL“* ir analizuoja ar jos teisingos.

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

|  |
| --- |
| Blokavimas, laukiant „*Iš vartotojo sąsajos*“ resurso |
|  |
| Failo nuskaitymas |
|  |
| Blokavimasis, laukiant „3-ojo kanalo“ resurso |
|  |
| Pranešimo procesui „Chan3\_Device“ atlaisvinimas |
|  |
| Blokavimas, laukiant proceso „Chan3\_Device“ darbo pabaigos |
| Blokavimasis, laukiant supervizorinės atminties resurso |
|  |
| Kopijavimas į supervizorinę atmintį |
|  |
| „Užduotis supervizorinėje atmintyje“ resurso atlaisvinimas |

### Procesas „*JCL*“

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

|  |  |
| --- | --- |
| Blokavimas, laukiant resurso „Užduotis supervizorineje atmintyje“ | |
|  | |
| Atlaisvinamas resursas „Programa klaidinga“  Taip | Ne  Užduotyje aptikta klaidų? |
|  | |
|  | |
| Atlaisvina resurą „Užduoties programa supervizorineje atmintyje“ | |

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 stabdytiProcesą(Procesas);

Public void pasiruosesProcesas(Procesas);

}

### Procesas „*Main\_Proc*“

Procesas pradeda savo veiklą laukdamas „Užuotis diske“. Kiekvienai vykdomai užduočiai turi būti sukurtas specialus procesas. „*Main\_Proc*“ sukuria užduotį prižiūrintį procesą – „*Job\_Governor*“. Šių procesų skaičius priklauso nuo atliekamų užduočių skaičiaus:

|  |  |  |
| --- | --- | --- |
|  | Main\_Proc |  |
|  | ... |
| Job\_Governor | Job\_Governor |
|  |  |  |
| Virtual Machine | ... | Virtual Machine |

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*.

1. **Pasiruošęs:** vienintelis trūkstamas resursas yra procesorius.
2. **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).
3. **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.

1. **Sustabdytas:** kito proceso sustabdytas procesas.

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 4 | | | Vykdomas | 1 | | |
|  | 2 | | 3 |  |  | |
| Pasiruošęs | | |  | Blokuotas | | |
| 7 | | 8 | 9 | 5 | | 6 |
| Pasiruošęs sustabdytas | | |  | Blokuotas sustabdytas | | |

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.

Procesas Main\_Proc, negaudamas resurso „*Užduotis išorinėje atmintyje*“, sukuria procesą „*Job\_Governor*“.

|  |  |  |
| --- | --- | --- |
| Blokavimas laukiant „*užduoties programa supervizorinėje atmintyje*“ resurso | | |
| Taip  Ne  Vykdymo laikas = 0? | | |
| Proceso „*Job\_Governor*“ sukūrimas, suteikiant jam „*užduoties programa supervizorinėje atmintyje*“ kaip pradinį resursą |  | Naikinti procesą „*Job\_Governor*“, sukūrusį gautąjį resurą. |

#### Procesas Job\_Governor

Kuriamas Main\_Proc proceso. Procesas virtualios mašinos tėvo, tvarkantis virtualios mašinos proceso darbą.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Blokavimas, laukiant „*vartotojo atminties*“ resurso | | | | | | |
|  | | | | | | |
| Resurso „*pakrovimo paketas*“ atlaisvinimas | | | | | | |
|  | | | | | | |
| Blokavimasis laukiant „Loader“ darbo pabaigos pranešimo | | | | | | |
|  | | | | | | |
| //„*Išorinės atminties*“ resurso atlaisvinimas | | | | | | |
|  | | | | | | |
| //Blokavimas laukiant „*supervizorinės atminties*“ resurso puslapių lentelei | | | | | | |
|  | | | | | | |
| Proceso „*Virtual\_Machine*“ sukūrimas, paduodame jai pointerį | | | | | | |
|  | | | | | | |
| Blokavimasis, laukiant proceso „*Interrupt*“ resurso | | | | | | |
|  | | | | | | |
| Stabdyti procesą „*Virtual\_Machine*“ | | | | | | |
| Ne  Ne  Taip  Taip  Įvedimas  Įvedimo/išvedimo pertraukimas? | | | | | | |
| Atlaisvinti resursą „eilutė supervizorinėje atmintyje“ |  | Blokavimasis laukinat 1-ojo kanalo resurso | | |  | Naikinti procesą Virtual\_Machine |
|  | | | | | | |
| Blokavimasis laukiant Print\_Lines darbo pabaigos |  | Eilutės nuskaitymas į vartotojo atminties buferį | | |  | Atlaisvinti „vartotojo atminties“ resursą |
|  |
| Atlaisvinti fiktyvų resursą „užduoties programa supervizorinėje atmintyje“ |
|  | | | | | | |
| Aktyvuoti procesą Virtual\_Machine | | |  | Blokavimasis laukiant neegzistuojančio resurso | | |

##### Virtual\_Machine

Procesas, atsakantis už vartotojiškos programos vykdymą.

|  |
| --- |
| Procesoriaus perjungimas į vartotojo režimą |
|  |
| Virtualios mašinos procesoriaus komandų emuliavimas |
|  |
| Sukuriamas pertraukimo įvykio resursas |

### Procesas „*Loader*“

|  |
| --- |
| Blokavimas, laukiant „pakrovimo paketo“ resurso |
|  |
| Užteko vartotojo atminties? |
|  |
| Sukuriama puslapių lentelė ir įrašoma |
|  |
|  |
|  |
| Duomenys iš supervizorinės atminties įrašomi į vartotojo pagal puslapių lentelę |
|  |
| Pranešimo procesui „Job\_Governor“, sukūrusiam „pakrovimo paketo“ resursą, sukūrimas |

### Procesas „*Print\_Lines*“

|  |
| --- |
| Blokavimas, laukiant „Eilutės supervizorinėje atmintyje“ resurso |
|  |
| Blokavimas, laukiant 2-ojo kanalo resurso |
|  |
| Eilutės kopijavimas iš supervizorinės atminties į išvedimo įrenginį |
|  |
| Atlaisvinama supervizorinė atmintis |
|  |
| 2-ojo kanalo atlasvinimas |

### Procesas „*Chan3\_Device*“

|  |
| --- |
| Blokavimas, laukiant pranešimo apie savo darbo inicijavimą |
|  |
| Kopijuoti nurodytą išorinės atminties takelių(blokų) skaičių į supervizorinę atminį |
|  |
| Generuoti pranešimą apie savo darbo pabaigą |

### 

### Procesas „*Interrupt*“

|  |
| --- |
| Blokavimas, laukiant „Interrupt“ |
|  |
| Pertraukimo tipo identifikavimas |
|  |
| Job\_Governor, atsakingo už pertrauktos VM darbą, nustatymas |
|  |
| Pranešimo apie Job\_Governor sugeneruotą pertraukimą sukūrimas |