VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ KATEDRA

**Multiprograminės operacinės sistemos projektas**

Vilnius – 2013

**Turinys**

[1. Realios mašinos aprašas 4](#_Toc357498095)

[1.1. Realios mašinos schema 4](#_Toc357498096)

[1.2. Techninės įrangos komponentai 4](#_Toc357498097)

[1.3. Techninės įrangos detalizacija 5](#_Toc357498098)

[1.3.1. Procesorius 5](#_Toc357498099)

[1.3.2. Centrinio procesoriaus registrai 5](#_Toc357498100)

[1.3.3. Pertraukimų registrai 5](#_Toc357498101)

[1.3.4. Išorinių komponentų registrai 6](#_Toc357498102)

[1.4. Atmintis 6](#_Toc357498103)

[1.5. Išoriniai įrenginiai 7](#_Toc357498104)

[1.6. Realios mašinos komandos 8](#_Toc357498105)

[1.7. Realios mašinos veikimo principas 9](#_Toc357498106)

[2. Virtualios mašinos aprašas 10](#_Toc357498107)

[2.1. Virtualios mašinos samprata 10](#_Toc357498108)

[2.2. Virtualios mašinos procesoriaus paskirtis 10](#_Toc357498109)

[2.3. Virtualiosios mašinos atmintis 10](#_Toc357498110)

[2.4. Puslapiavimo mechanizmas 11](#_Toc357498111)

[2.5. Virtualios mašinos kūrimo ir veiklos scenarijus: 11](#_Toc357498112)

[2.6. Taimerio mechanizmas 11](#_Toc357498113)

[2.7. Pertraukimų mechanizmas 12](#_Toc357498114)

[2.8. Kanalai 12](#_Toc357498115)

[2.9. Virtualios mašinos komandų sistema 13](#_Toc357498116)

[2.9.1. Aritmetinės darbo su sveikaisiais skaičiais komandos 13](#_Toc357498117)

[2.9.2. Darbo su duomenimis komandos 13](#_Toc357498118)

[2.9.3. Įvedimo/išvedimo komandos 13](#_Toc357498119)

[2.9.4. Valdymo perdavimo komandos 14](#_Toc357498120)

[2.9.5. Palyginimo komanda 14](#_Toc357498121)

[3. Multiprograminė OS 15](#_Toc357498122)

[4. Procesai 15](#_Toc357498123)

[4.1. Procesų būsenos 15](#_Toc357498124)

[4.1.1. Procesų būsenų sąrašas 15](#_Toc357498125)

[4.1.2. Procesų būsenų diagrama 16](#_Toc357498126)

[4.2. Procesų planuotojas 17](#_Toc357498127)

[4.2.1. Eilės principas 17](#_Toc357498128)

[4.2.2. Planuotojo veikimo algoritmas 18](#_Toc357498129)

[4.2.3. Proceso deskriptorius 19](#_Toc357498130)

[4.2.4. Procesų primityvai 19](#_Toc357498131)

[4.3. Resursai 19](#_Toc357498132)

[4.3.1. Resurso deskriptorius 20](#_Toc357498133)

[4.3.2. Resursų primityvai 20](#_Toc357498134)

[4.3.3. Resurso paskirstytojas 20](#_Toc357498135)

[4.3.4. Resurso paskirstytojo veikimas 21](#_Toc357498136)

[4.4. Resursų lentelė 22](#_Toc357498137)

[4.5. Procesų paketas 23](#_Toc357498138)

[4.5.1. Sisteminių procesų sąrašas 23](#_Toc357498139)

[4.5.2. Įvedimas ir išvedimas 24](#_Toc357498140)

[4.5.3. Procesas StartStop 24](#_Toc357498141)

[4.5.4. Procesas ReadFromInterface 25](#_Toc357498142)

[4.5.5. Procesas Validator 26](#_Toc357498143)

[4.5.6. Procesas MainProc 27](#_Toc357498144)

[4.5.7. Procesas JobGoverner 28](#_Toc357498145)

[4.5.8. Procesas VirtualMachine 30](#_Toc357498146)

1. Realios mašinos aprašas

* 1. Realios mašinos schema

**Supervizorinė**

**atmintis**

**Centrinis procesorius**

**Vartotojo**

**atmintis**

15536

65535

**1 kanalas**

**2 kanalas**

**3 kanalas**

**Įvedimo įrenginys**

**Išvedimo renginys**

**PTR**

**R**

**IC**

**MODE**

**SF**

**PI**

**TI**

**SI**

**IOI**

**CHST[1]**

**CHST[2]**

**CHST[3]**

**Puslapiavimo mechanizmas**

**Išorinė atmintis**

0

15535

* 1. Techninės įrangos komponentai
* Procesorius
* Atmintis
* Išoriniai įrenginiai
  1. Techninės įrangos detalizacija
     1. Procesorius

Procesorius dirba dviem režimais: supervizoriaus arba vartotojo.

Dydžiams nusakyti naudojami šie matai: 1 baitas yra 8 bitai; 1 žodis yra 4 baitai; 1 blokas yra 10 žodžių.

* + 1. Centrinio procesoriaus registrai

**R – bendro naudojimo registras**.  Bendros paskirties registras. Registro dydis – 4 baitai.

**SF – būsenos registras**. Registras įgyja reikšmę, nusakančią įvykdytos komandos rezultatą. Registro dydis – 1 baitas.

**IC – komandų skaitliuko registras**. Nurodo sekančios komandos arba jos operando žodžio realų arba virtualų adresą atmintyje, priklausomai nuo procesoriaus režimo. Jei komanda nenurodo kitaip, šis registras automatiškai padidinamas vienetu. Registro dydis -  2 baitai. Registras pasiekia 216 atminties žodžių.

**PTR – puslapių lentelės registras**.  PTR reikšmė – einamosios vartotojiškos užduoties puslapių lentelės bazinis adresas ir ilgis. 4 PTR baitai a0 a1 a2 a3 turi sekančias reikšmes: a0 – nenaudojamas, a1 – puslapių lentelės ilgis blokais, a2a3 – atminties blokas, kur yra puslapių lentelė. Registro dydis – 4 baitai.

**MODE registras**. Procesoriaus darbo režimo registras. Registras įgyja reikšmes 1 (supervizoriaus režimas) arba 0 (vartotojo režimas). Registro dydis – 1 bitas.

* + 1. Pertraukimų registrai

**PI – programinių pertraukimų registras**. Programinis pertraukimas kyla aptikus vartotojo užduoties programoje klaidingą operacijos kodą arba neleistiną virtualų adresą. PI įgyja reikšmę 1, kai pažeista atminties apsauga, 2 – klaidingas operacijos kodas. Dydis - 1 baitas.

**TI – taimerio pertraukimo registras**. Kai taimerio skaitliukas lygus 0, įvyksta pertraukimas. Registro dydis – 1 baitas.

**SI – supervizorinių pertraukimų registras**. Komandos RD, PD, HL iššaukia supervizorinius pertraukimus (kviečia supervizorių). SI gali įgyti tokias reikšmes : 1 – komanda RD, 2 – komanda PD, 3 – komanda HL.  Dydis – 1 baitas.

**IOI – Įvedimo/išvedimo pertraukimų registras.** Įvedimo – išvedimo operacijos pabaigos registras. Įgyja tokias reikšmes: 1 – darbą baigė I kanalas, 2 – darbą baigė II kanalas, 4 – darbą baigė III kanalas.  Jei įvyksta keli tokie pertraukimai, tai reikšmės sudedamos. Registro dydis – 1 baitas.

* + 1. Išorinių komponentų registrai

**CHST[1] – pirmojo kanalo būsenos registras**. Rodo pirmojo kanalo būseną, kuris jungia duomenų skaitymo įrenginį ir supervizoriaus atmintį. Dydis – 1 bitas.

**CHST[2] – antrojo kanalo būsenos registras**. Rodo antrojo kanalo būseną, kuris jungia duomenų išvedimo įrenginį ir supervizoriaus atmintį. Dydis – 1 bitas.

**CHST[3] – trečiojo kanalo būsenos registras**. Rodo trečiojo kanalo būseną, kuris jungia išorinę atmintinį ir supervizoriaus atmintį. Registro dydis – 1 bitas.

* 1. Atmintis

Atmintis skaidoma į supervizorinę, vartotojo atmintį bei išorinę atmintį.

**Supervizorinė atmintis**

0

15535

Nuo 12770-tojo žodžio puslapiavimo lentelė

Nuo 14779-tojo žodžio

PTR reikšmių lentelės

Nuo 15279-tojo žodžio

pertraukimų lentelė

Bendroji supervizorinė atmintis

**Supervizorinė atmintis** skirta pačios operacinės sistemos reikmėms. Supervizoriaus atmintyje laikomi sisteminiai procesai, sisteminės programos, puslapių lentelės ir kiti sistemai reikalingi resursai. Supervizoriaus atminčiai išskiriama 15536 žodžiai po 4 baitus. Ties 12770-tu žodžiu atmintis išskiriama saugoti puslapiavimo lentelėmis, kuriu maksimalus užimamas dydis – 2009 žodžiai. Ties 14779-tu žodžiu atmintis išskiriama saugoti VM PTR reikšmes, kuriu maksimali užimama atmintis 500 žodžių. Ties 15279-tu žodžiu atmintis išskiriama pertraukimų lentelei, kurios dydis 256 žodžių.

**Vartotojo atmintis** procesoriui pasiekiama per puslapiavimo mechanizmą. Ji skirta laikyti virtualių mašinų duomenims. Vartotojo atminties dydis yra 50000 žodžiai po 4 baitus.

**Išorinės atminties** struktūra tokia pati kaip ir vidinės, t.y. yra sudaryta iš žodžių po 4 baitus, kur kiekvienas jų turi po 8 bitus. Maksimalus pasiekiamas dydis – 216.

* 1. Išoriniai įrenginiai

**Įvedimo įrenginys**, skirtas informacijos įvedimui (pvz. klaviatūra).

**Išvedimo įrenginys**, skirtas informacijos išvedimui (pvz. monitorius).

**Išorinė atmintis**, skirta informacijos laikymui išorinėje atmintyje (pvz. failas).

* 1. Realios mašinos komandos

Komandos užima vieną žodį, jei nėra nurodyta kitaip. Pirmųjų dviejų komandos baitų reikšmės nurodo komandos tipą, atitinkamai pagal ASCII lentelę. Jei komanda neatpažįstama, IC reikšmė padidinama vienetu ir vykdomas sekantis žodis. Šios komandos galimos tik esant MODE = 1. RM taip pat naudoja ir toliau aprašytą VM komandų sistemą.

**IO x y z w** – 2 žodžių įvedimo/išvedimo komanda. Kur x – žodžio vieta iš kurios kopijuosime 2 baituose, y – vieta į kurią kopijuosime 2 baituose, z – žodžių kiekis, w – naudojamo kanalo numeris. Jei komanda duodama užimtam kanalui, tai centrinis procesorius laukia kanalo atsilaisvinimo. Kanalų būsenos nustatomos pagal būsenos registrus CHST[i], i = 1,2,3. CHST[i] = 0 jei kanalas laisvas ir CHST[i] = 1, jei užimtas. Duomenų kryptis nustatoma pagal SF registrą (0 – duomenys nuskaitomi iš atminties, 1 – įrašomi). Jei parametrai 0 0 0 0, tai parametrai imami iš RD/PD komandą naudojusios VM.

**SL x –** komanda priskirianti žodį PTR = [14779 + x], baitus SF = [10 \* 4 \* n – 6], IC = [10 \* 4 \* n – 5],

R = [10 \* 4 \* n – 3], ir MODE = 0. Jeigu x = 0, darbas atiduodamas sekančiai VM. Kur n – PTR registro antrojo baito reikšmė.

**ST x y** – 2 žodžių komanda priskirianti registrui x (1 baitas) žodžio y reikšmę, jei registras gali talpinti tokią reikšmę. Atitinkamas registras nustatomas pagal lentelę:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *reg(x)*reikšmė | 0 | 1 | 2 | 3 | 4 | 5 | 6 | i = 7, 8, 9 | 10 |
| registras | PTR | IC | SF | PI | TI | SI | IOI | CHST[i] | MODE |

**GT x y** – 2 žodžių komanda nukopijuojanti registro x (1 baitas) reikšmę į žodį ties adresu y (2 baitai). Atitinkamas registras nustatomas pagal lentelę:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *reg(x)*reikšmė | 0 | 1 | 2 | 3 | 4 | 5 | 6 | i = 7, 8, 9 | 10 |
| registras | PTR | IC | SF | PI | TI | SI | IOI | CHST[i] | MODE |

* 1. Realios mašinos veikimo principas

Pradžia

MODE = 1, IC = 0

TT

Vykdoma komanda adresu IC per PTR

Keičiamas IC atitinkamai pagal vykdytą komandą

TRUE

FALSE

MODE = 1

Vykdoma komanda adresu IC

Keičiamas IC atitinkamai pagal vykdytą komandą

Išsaugomi VM duomenys

MODE = 1,

IC = [15279 + poslinkis]

FALSE

TRUE

1. Virtualios mašinos aprašas
   1. Virtualios mašinos samprata

Virtuali mašina yra paprastesnis realios mašinos modelis, kurios operacijos yra apdorojamos operacinės sistemos. Virtuali mašina tampa tarsi tarpininku tarp konkrečios mašinos ir jai taikomos programinės įrangos. VM skirtos izoliuoti atskiras programas ir optimaliai joms paskirstyti kompiuterio resursus.

* 1. Virtualios mašinos procesoriaus paskirtis

Paskirtis yra vykdyti programą, kuri yra vartotojo atmintyje. Kiekviena VM turi savo virtualų centrinį procesorių. Virtualios mašinos procesoriaus registrai yra R – bendrojo naudojimo registras, IC – komandų skaitiklis, SF – būsenos registras.

* 1. Virtualiosios mašinos atmintis

Kiekvienai virtualiai mašinai yra skiriama 10n vartotojo atminties blokų, n – blokais reikalinga atmintis programai, kurią apsibrėžia pats vartotojas. Šios atminties dešimtasis ir devintasis baitai nuo pabaigos yra skirtas steko viršūnės reikšmei saugoti, kuri naudojama procedūrų komandoms. Ši reikšmė – tai steko elementų skaičius, kurie yra saugomi vartotojo atmintyje nuo šių dviejų baitų link vartotojo atminties pradžios. Kiekvienas steko elementas užima 2 baitus. 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.

.

IOI adresas „to“

**VM**

**Atmintis**

**. .**

**. .**

**. .**

Saugomas VM registras R

Saugomas VM registras IC

Saugomas VM registras SF

Saugomas VM steko elementų skaičius

10n blokų

(100n žodžių)

1

IOI kanalo numeris

IOI adresas „from“

IOI žodžių kiekis

* 1. Puslapiavimo mechanizmas

Pagrindinis puslapiavimo mechanizmo elementas – puslapių lentelės, jos saugomos supervizorinėje atmintyje.Jos išlaiko sąryšį tarp realių ir virtualių blokų adresų. Norint sužinoti realų adresą, reikia kreiptis į puslapio lentelę ir nuskaityti reikšmę esančią žodyje su bloko numeriu. Tokiu būdu virtuali mašina gali pasiekti visą jai išskirtą atmintį realioje atmintyje. Realusis adresas nusakomas pagal formulę:

*Realus\_adresas := 10 \* [PTR \* 10 + x1] + x2*

*kur PTR – registro PTR reikšmė 3 ir 4 baito reikšmė, x1 = IC div 10, x2 = IC mod 10*

*[x] – žodžio reikšmė adresu x*

* 1. Virtualios mašinos kūrimo ir veiklos scenarijus:

Virtuali mašina reikalauja 10n blokų atminties savo reikmėms, kur n – vartotojo nustatytas dydis programai. VM sukūrimas yra atliekamas pačios operacinės sistemos, vadovaujantis šiuo scenarijumi.

1. surandama laisva vieta vartotojo atmintyje pagal jau esamus PTR, kurie nurodo jau užimtą vietą;
2. išskiriama 10n blokų virtualiai mašinai, sukuriant naują puslapių lentelę, užimančia n blokų supervizorinėje atmintyje;
3. puslapių lentelė (n blokų) užpildoma iš skirtų 10n blokų realiais adresais;
4. 4 baitų registro PTR reikšmei priskiriamas puslapių lentelės bloko realus žodžio adresas ir jos dydis blokais;
5. PTR registro reikšmė išsaugoma supervizorinėje atmintyje;
6. VM baigiama kurti;
7. procesorius tampa prieinamas VM reikmėms, įkėlus šios VM registrų reikšmes iš supervizorinės atminties ir procesorių perjungus į MODE = 0;
8. VM vykdoma paverčiant virtualų adresą IC realiu pagal formulę: *Realus\_adresas := 10 \* [PTR + x1] +* *x2, kur PTR – registro PTR reikšmė 3 ir 4 baito reikšmė, x1 = IC div 10, x2 = IC mod 10, [x] – žodžio reikšmė adresu x*
   1. Taimerio mechanizmas

VM pradedant darbą su procesoriumi, priskiriama TI = 10, t.y. vykdoma ne daugiau kaip 10 einamosios užduoties taktų. I/O komandos atliekamos per 3 taktus, visos kitos – per 1 taktą. Atlikus komandas, atitinkamai sumažinama ir TI reikšmė pagal taktų kiekį. Juos įvykdžius įvyksta pertraukimas TI = 0.

* 1. Pertraukimų mechanizmas

Pertraukimas – tai signalas apie įvykusį įvykį. Kiekvienas pertraukimas turi savo identifikaciją pagal kurią sistema atskiria pertraukimo tipą. Aptikus pertraukimą, tolesnis sistemos valdymas yra perduodamas to pertraukimo nustatytai programai.

Realizuojami trijų tipų pertraukimai – programiniai, supervizoriniai ir taimerio. Programinių pertraukimų registras yra PI, supervizorinių pertraukimų registras – SI, taimerio - TI. Programiniai pertraukimai kyla vykdant virtualią mašiną, bandant įvykdyti kokį nors neleistiną veiksmą arba nuskaičius neleistiną reikšmę. Supervizoriniai pertraukimai kyla virtualiai mašinai norint įvykdyti veiksmą, kuris gali vykti tik supervizoriaus režime.

Pertraukimai gali būti aptikti tik vartotojo režime. Supervizoriniame režime centrinio procesoriaus darbo pertraukti negalima.

Pertraukimai kils šiais būdais:

* Operacijos RD, PD ir HL iššauks supervizorinius pertraukimus. SI = 1 – komanda RD, SI = 2 – komanda PD, SI = 3 – komanda HL. Šių pertraukimų poslinkiai atitinkamai yra 0, 1 ir 2.
* Programiniai pertraukimai: PI = 1 – pažeista atminties apsauga, PI = 2 – klaidingas operacijos kodas. Šių pertraukimų poslinkiai atitinkamai yra 3 ir 4.
* Esant TI = 0 bus fiksuojamas taimerio pertraukimas. Šio pertraukimo poslinkis yra lygus 5.
* Po komandos IT, kurios parametras x yra traktuojamas kaip poslinkis. T.y. poslinkis = x. Jei x > 28 - 1, tai poslinkis = 3 (pažeista atminties apsauga);

Esant situacijai SI = 0 ir PI = 0 ir TI <> 0, pertraukimų sistema neaptiks.

Aptikus pertraukimą VM mašinos registrų SF, IC, R reikšmės yra paeiliui išsaugomos virtualios mašinos atmintyje, paskutiniuose 18-oje baituose, kartu su IOI informaciją. Pertraukimas apdorojamas procesoriui persijungiant į MODE = 1, o IC priskiriant reikšmę esančią ties 15279 atminties žodžiu + poslinkis.

* 1. Kanalai

Norint prie kompiuterio prijungti išorinius įrenginius, tam reikalingi kanalai. Kanalai naudojami keitimuisi duomenimis tarp sistemos ir išorinio įrenginio. Šiuo atveju turime 3 kanalus. Pirmasis kanalas sistemą jungia su įvedimo įrenginiu, antrasis kanalas - su išvedimo įrenginiu, o trečiasis - su išorine atmintimi. Įvesties/išvesties instrukcijai atlikti, siunčiama komanda, specifikuojanti tam tikrą kanalą, darbinės informacijos vietą bei kiekį. Vykdant šias komandas, procesoriuje esantys registrai įgyja reikšmę 1 (*TRUE*), atitinkamai pagal užimto kanalo numerį. Operacijos vykdymo metu įvesties/išvesties modulis daugiau nesiunčia centriniam procesoriui jokių įspėjimo pranešimų ir centrinio procesoriaus nėra darbas pertraukiamas. Taigi centrinis procesorius periodiškai tikrina kanalų būklę, kol būklės registre bus aptikta informacija, kad operacija įvykdyta.

* 1. Virtualios mašinos komandų sistema

Šios sistemos komandos galimos kai MODE = 0 arba MODE = 1. Ši komandų sistema taip pat yra ir dalis RM komandų sistemos (išskyrus PER, RE, IT ir HL). Jei komanda tiesiogiai nekeičia IC reikšmės, tai IC reikšmė yra padidinama 1-tu. Esant MODE = 0, visos komandos yra operuojamos naudojant puslapiavimo mechanizmą.

[a] – reikšmė adresu a.

x – 2 baitų reikšmė.

n – PTR registro antrojo baito reikšmė.

s – [10 \* 4 \* n – 18] – 2 baitų dydžio reikšmė vartotojo atmintyje išskirtame procedūrų steke.

Įvedimo ir išvedimo komandos yra dviejų žodžių

* + 1. Aritmetinės darbo su sveikaisiais skaičiais komandos

**AD x** – 2 sveikų skaičių, užimančių žodį atmintyje, sudėties komanda (R := R + [a], kur [a] = x).

**SB x** – 2 sveikų skaičių, užimančių žodį atmintyje, atimties komanda (R := R – [a], kur [a] = x).

**ML x** – 2 sveikų skaičių, užimančių žodį atmintyje, sandaugos komanda (R := R \* [a], kur [a] = x).

**DD x** – 2 sveikų skaičių, užimančių žodį atmintyje, dalybos komanda (R := R div [a] , kur [a] = x).

**DM x** – 2 sveikų skaičių, užimančių žodį atmintyje, dalybos komanda (R := R mod [a] , kur [a] = x).

* + 1. Darbo su duomenimis komandos

**LR x** – žodžio, kurio adresas x, turinio kopijavimas į registrą R.

**SR x** – registro R reikšmės įrašymas į atmintį su adresu x.

* + 1. Įvedimo/išvedimo komandos

**RD x y z w** – z žodžių kiekio iš w kanalo x žodžio rašymas į atmintį ties adresu y. [s + 10] = x,

[s + 12] = y, [s + 16] = z, [s + 17] = w

Iššaukiamas pertraukimas SI = 1.

**PD x y z w** – z žodžių kiekio iš w kanalo x žodžio rašymas į atmintį ties adresu y. [s + 10] = x,

[s + 12] = y, [s + 16] = z, [s + 17] = w

Iššaukiamas pertraukimas SI = 2.

* + 1. Valdymo perdavimo komandos

**JM x** – nesąlyginio valdymo perdavimo komanda. Valdymas perduodamas nurodytu adresu x. IC = x.

**JE x** – sąlyginio valdymo perdavimo komanda. Jei registro SF reikšmė lygi 1, tai IC = x.

**JL x** – sąlyginio valdymo perdavimo komanda. Jei registro SF reikšmė lygi 2, tai IC = x.

**JG x** – sąlyginio valdymo perdavimo komanda. Jei registro SF reikšmė lygi 3, tai IC = x.

**PR x** – nesąlyginio valdymo perdavimo komanda. Valdymas perduodamas nurodytu adresu x, prieš tai išsaugant IC reikšmę 2-juose baituose vartotojo atmintyje išskirtam procedūrų steke pagal adresą [s – 2 \* [s]], tuomet [s] reikšmę padidinant vienetu ir priskiriant IC = x.

**RE** – nesąlyginio valdymo perdavimo komanda. IC priskiriama 2 baitų reikšmė iš procedūrų steko viršaus adresu [s - 2 \* [s]], reikšmę adresu [s] sumažinant vienetu.

**HL** – vartotojo programos vykdymo pabaiga. Iššaukiamas pertraukimas SI = 3.

**IT x** – Iššaukiamas pertraukimas su parametru x.

**TT** – komanda, esant būsenai (PI <> 0 arba SI <> 0 arba IOI <> 0 arba TI = 0), paeiliui išsaugo registrų SF, IC, R reikšmes nuo adreso s + 1 VM atmintyje, pakeičiant į MODE = 1, o IC pagal įvykusį pertraukimą.

* + 1. Palyginimo komanda

CM x – palygina registro R reikšmę su x žodžio reikšme atmintyje. Jei reikšmės lygios, tai SF = 1, jei R esanti reikšmė yra didesnė, tai SF = 2, jei mažesnė, tai SF = 3.

1. Multiprograminė OS

Multiprograminėje operacinėje sistemoje centrinis procesorius yra perjungiamas iš vieno proceso į kitą. Kiekvieną iš procesų vykdo tik kelias dešimtąsias ar šimtąsias sekundės dalis. Todėl net ir sekundės bėgyje procesorius sugeba aptarnauti keletą procesų, ir tai suteikia vartotojui lygiagretaus veikimo įspūdį.

1. Procesai

**Procesas** – tai vykdoma programa, kartu su esamomis registrų reikšmėmis ir savo kintamaisiais. Kiekvienas procesas turi savo virtualų procesorių. Procesą laikysime kokioje nors veiklumo stadijoje esančią programą. **Programa** – tai tik tam tikras baitų rinkinys. Veiklumo stadiją apibūdina **proceso aprašas** – **deskriptorius**. Apraše ir yra laikomi visi procesui reikalingi parametrai, tokie kaip virtualaus procesoriaus registrų reikšmės, ar jam reikalingi kintamieji. Procesų aprašai yra dinaminiai objektai. Tai reiškia, kad jie gali būti sukurti ar sunaikinti jau sistemos veikimo metu.

Procesus galima suskirstyti į sisteminius ir vartotojiškus. Sisteminių procesų paskirtis – aptarnauti vartotojiškus. Tuo tarpu vartotojiško proceso paskirtis yra vykdyti vartotojo programą.

* 1. Procesų būsenos

Procesas gali gauti procesorių tik tada, kai jam netrūksta jokio kito resurso. Procesas gavęs procesorių tampa vykdomu ir turi procesorių, kol sistemoje neįvyksta pertraukimas arba einamasis procesas nepaprašo kokio nors resurso. Procesoriaus resursas yra reikalingas visiems procesams. Nė vienas procesas netaps vykdomu neturėdamas procesoriaus.

* + 1. Procesų būsenų sąrašas
* **Vykdomas** – turi procesorių
* **Blokuotas** – prašo resurso (išskyrus procesorių)
* **Pasiruošęs** – vienintelis trūkstamas resursas yra procesorius
  + 1. Procesų būsenų diagrama

1. Vykdomas procesas blokuojasi jam prašant resurso.
2. Vykdomas procesas tampa pasiruošusiu atėmus iš jo procesorių dėl kokios nors priežasties (pvz.: skirstant laiką).

Vykdomas

Pasiruošęs

Blokuotas

1.

2.

3.

4.

1. Blokuotas procesas tampa pasiruošusiu, kai yra suteikiamas reikalingas resursas.
2. Pasiruošęs procesas gavęs procesorių tampa vykdomu.
   1. Procesų planuotojas

Planuotojo paskirtis yra atimti procesorių iš proceso, peržvelgti pasiruošusių procesų sąrašą, išrinkti pasiruošusį procesą, kuris planuotojo manymu yra tinkamiausias, ir perduoti procesorių jam. Planuotojas kviečiamas kai norima pakeisti einamąjį procesą kitu.

**PS** – procesų sąrašas veikia paprastos eilės principu. T.y. pasiruošę procesai yra įdėdami į sąrašo pabaigą, o procesas, kuris bus vykdomas, paimamas iš sąrašo viršaus. Procesai nėra prioretizuojami, tad nei sisteminiai, nei vartotojo procesai, negali būti dirbtinai stumdomi sąraše.

* + 1. Eilės principas

Procesas

Procesas

Procesas

Procesas

Procesas

Procesas

Procesas

Procesas

**...**

Nauji procesai keliauja į eilės galą

Vykdymui paimamas pirmasis pasiruošęs

procesas nuo eilės priekio

**PS**

* + 1. Planuotojo veikimo algoritmas

Procesas keliauja į procesų eilę

Procesas tampa vykdomu

Vykdomasis procesas tampa pasiruošusiu arba blokuotu

Planuotojas baigia darbą

Parenkamas pirmasis pasiruošęs procesas iš procesų sąrašo

* + 1. Proceso deskriptorius

**ID** – unikalus vidinis proceso numeris

**state** – proceso būsena

**externalName** – išorinis proceso vardas, naudojamas patogesniam procesų identifikavimui

**savedState** – išsaugota proceso būsena, naudojama tęsti proceso darbą

**resCreated** – proceso sukurti resursai

**procChildren** – vaikinių procesų sąrašas

* + 1. Procesų primityvai

Procesų primityvų paskirtis – pateikti vienodą ir paprastą vartotojo sąsają darbui su procesais. Darbui su procesais skirti 2 primityvai:

1. **Kurti procesą**. Šiam primityvui perduodamas išorinis proceso vardas ir tėvinio proceso ID. Pačio primityvo viduje vyksta proceso kuriamasis darbas. Jis yra registruojamas bendrame procesų sąraše, skaičiuojamas vidinis identifikacijos numeris, sukuriamas tuščias jo vaikų procesų sąrašas, sukurtų resursų sąrašas, būsena nustatoma į pasiruošusį.
2. **Naikinti procesą**. Pradedama naikinti proceso vaikiniai procesai ir proceso bei jo vaikų sukurti resursai. Sunaikinti procesai bei resursai išmetami iš bendrųjų sąrašų.

Kiekvieno primityvo programos gale yra kviečiamas planuotojas.

* 1. Resursai

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

Resursai yra laisvi, kai nė vienas procesas jų nenaudoja, arba užimti, kada juos naudoja procesai arba jie yra pažymėti kaip užimti.

Resursas iš tikrųjų tėra tik aprašas (deskriptorius), galintis saugoti ir atiduoti informaciją. Kiekvienas resursas turi laukiančių procesų sąrašą (jis gali būti ir tuščias). Kiekvienas procesas prašęs resurso yra ne tik užblokuojamas, bet ir įdedamas į resurso laukiančių procesų sąrašą.

* + 1. Resurso deskriptorius

**ID** – unikalus vidinis resurso numeris

**externalName** – išorinis resurso vardas, naudojamas patogesniam resursų identifikavimui

**createdBy** – resursą sukūrusio proceso identifikacinis numeris

**usingBy** – resursą naudojančio proceso identifikacinis numeris arba resurso būseną nusakanti reikšmė

**waitingProc** – procesų laukiančių resurso sąrašas

* + 1. Resursų primityvai

Resursas turi keturis primityvus:

1. **Kurti resursą**. Resursus kuria tik procesas. Resurso kūrimo metu perduodami kaip parametrai išorinis resurso vardas ir kuriančio proceso vidinis numeris. Resursas kūrimo metu yra: pridedamas prie bendro resursų sąrašo, pridedamas prie tėvo sukurtų resursų sąrašo, jam priskiriamas unikalus vidinis vardas, sukuriamas laukiančių procesų sąrašas.
2. **Naikinti resursą**. Resursas išmetamas iš jo tėvo sukurtų resursų bei bendrojo sąrašo. Resursas sunaikinimas.
3. **Prašyti resurso**. Procesas, iškvietęs šį primityvą, yra užblokuojamas ir įtraukiamas į to resurso laukiančių procesų sąrašą.
4. **Atlaisvinti resursą**. Šiam primityvui perduodamas resurso vardas. Resursas atlaisvinamas – nustatomas pradinis identifikacijos numeris.
   * 1. Resurso paskirstytojas

Kaip kad procesorius yra skirstomas planuotojo, kiekvienas resursas taipogi yra skirstomas paskirstytojo.

Resursų paskirstytojo paskirtis – suteikti paprašytą resursą procesui. Resurso paskirstytojas peržvelgia visus laukiančius šio resurso procesų sąrašą, ir, sutikęs galimybę aptarnauti procesą, perduoda jam reikalingus resurso elementus ir pažymi jį pasiruošusiu. Paskirstytojas kviečiamas po kiekvieno planuotojo.

* + 1. Resurso paskirstytojo veikimas

Laukia

Procesai įtraukiami į resursų belaukiančiųjų sąrašus

Patikrinami naujai sukurti resursai, ar jų nelaukia blokuoti proc.

Laisvi resursai atiduodami procesams

Nelaukia

* 1. Resursų lentelė

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Resursas | Kas sukuria | Kas atlaisvina | Kas naudoja | Kas naikina |
| Atmintis | StartStop | StartStop, JobGorvernor | JobGoverner | StartStop |
| PirmasisKanalas | StartStop | StartStop, JobGorvernor, ReadFromInterface | ReadFromInterface | StartStop |
| AntrasisKanalas | StartStop | StartStop, JobGorvernor | JobGoverner | StartStop |
| TrečiasisKanalas | StartStop | StartStop, Validator, Loader | Validator, Loader | StartStop |
| MOSPabaiga | StartStop | StartStop, ReadFromInterface | ReadFromInterface | StartStop |
| Užduotisišorinėje atmintyje | StartStop | ReadFromInterface | Validator | StartStop |
| VartotojoIvedimas | StartStop | Validator | PrintLine | StartStop |
| Validi programa išorinėje atmintyje | Validator | Validator, JobGorvernor | MainProc | MainProc |
| Pertraukimas | VirtualMachine | VirtualMachine | JobGoverner | JobGoverner |

* 1. Procesų paketas
     1. Sisteminių procesų sąrašas

• **StartStop** – šakninis procesas, sukuriantis bei naikinantis sisteminius procesus ir resursus.

• **ReadFromInterface** – užduoties nuskaitymo iš įvedimo srauto procesas

• **Validator** – užduoties programos validumo patikrinimo procesas

• **MainProc** – Procesas valdantis JobGorvernor procesus.

• **JobGorvernor** – virtualios mašinos proceso tėvas, tvarkantis virtualios mašinos proceso darbą

• **Virtual Machine** – procesas atsakantis už vartotojiškos programos vykdymą.

Beveik visi procesai yra sukuriami sistemos darbo pradžioje proceso StartStop. StartStop nekuria tik 2 procesų – JobGorvernor (kiekvienai naujai vartotojo užduočiai MainProc kuria po naują procesą JobGorvernor) ir VirtualMachine, kurį kuria JobGorvernor.

Validator

MainProc

JobGovernor

JobGovernor

ReadFromInterface

VirtualMachine

VirtualMachine

**………**

StartStop

* + 1. Įvedimas ir išvedimas

Vykdant vartotojo užduotis, būtina turėti įvedimo ir išvedimo galimybes. Mūsų modelyje už jas bus atsakinga vartotojo sąsaja. Bus laikoma, kad yra galimybė iš vartotojo sąsajos pasiųsti pranešimą (atlaisvinti resursą). Šis veiksmas atstos įvedimo srautą. Be to, pasiųsta į išvedimo srautą eilutė, bus atvaizduota vartotojo sąsajoje. Vartotojo sąsaja – tai būdas stebėti ir įtakoti modelį. Įvedimo ir išvedimo srautai jungia modelį su vartotojo sąsaja.

Vartotojo sąsajoje yra konsolinis langas, kurioje matomi sistemos siunčiami pranešimai, bei yra galimybė perduoti komandas sistemai naudojantis komandinė eilutė. Komandinė eilutė leidžia įvesti bet kokią simbolių seką, kuri, paspaudus mygtuką <enter>, yra pasiunčiama į įvedimo srautą(atlaisvinamas resursas). Konsolinis langas išveda informaciją, modelio pasiųstą į išvedimo srautą ir ją kaupia.

* + 1. 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.

Procesas StartStop, gavęs procesorių, savo darbą pradeda sukurdamas visus sisteminius resursus. Laikoma, kad procesoriaus resurso kurti nereikia, jis kuriamas ir naikinamas įjungiant ir išjungiant kompiuterį. Sukūręs resursus, StartStop kuria permanentinius procesus, t.y. tuos procesus, kurie bus aktyvūs visą MOS gyvavimo laiką. Mūsų modelyje tokie procesai bus: ReadFromInterface, Validator, Loader, MainProc, Interrupt ir PrintLine.

Sekantis StartStop etapas yra prašyti resurso “MOS pabaiga”. Šioje vietoje StartStop blokuojasi ir laukia kol bus atlaisvintas pranešimas apie MOS darbo pabaigą. Priklausomai nuo prioriteto anksčiau ar vėliau StartStop bus atblokuotas ir tęs darbą. Sekantis etapas yra sisteminių procesų naikinimas. Jeigu procesų primityvas “naikinti procesą” yra realizuotas korektiškai, bus sunaikinti visi sistemoje esantys procesai (t.y. StartStop vaikų vaikai ir t.t.). Galiausiai yra naikinami sisteminiai resursai. Tuo MOS darbas ir pasibaigia.

Sisteminų resursų inicializacija

Sisteminių permanentinių procesų inicializacija

Blokavimasis laukiant „MOSpabaiga” resurso

Sisteminių procesų naikinimas

Sisteminių resursų naikinimas

START

STOP

* + 1. Procesas ReadFromInterface

Šį procesą kuria ir naikina procesas StartStop. ReadFromInterface pirmasis žingsnis – patikrinti kanalų registrus žiūrint ar įvyko įvedimo srautas. Jei taip, procesas blokuojasi laukdamas resurso „Pirmasis kanalas“. Gavus šį resursą yra nuskaitoma komanda. Jei tai komanda „Load X“, kur x yra takelio numeris išorinėje atmintyje, atlaisvinamas resursas„Užduotis išorinėje atmintyje“ su parametru x. Jei tai tiesiog vartotojo informacija, atlaisvinamas resursas „Vartotojo įvedimas“ su informacija.

Atlaisvinamas resursas „Vartotojo įvedimas“

Kas tai?

Paprastas įvedimas

Užduotis išorinėje atmintyje

Atlaisvinamas resursas „Užduotis išorinėje atmintyje”

Atlaisvinamas resursas „Pirmasis kanalas“

Blokavimasis laukiant „Pirmasis kanalas“ resurso

Ar įvyko įvedimas?

Ne

Taip

Atlaisvinamas fiktyvus resursas „Fiktyvus”

Atlaisvinamas resursas „MOSPabaiga“

Shutdown

* + 1. Procesas Validator

Procesą Validator kuria ir naikina procesas StartStop. Proceso Validator paskirtis – validuoti programos korektiškumą ir vėl blokuotis laukiant pranešimo iš ReadFromInterface.

Procesas pradeda savo darbą blokuodamasis, laukdamas pranešimo iš ReadFromInterface. Sulaukus šio pranešimo yra vėl blokuojamasi ir laukiama resurso „Trečiasis kanalas“. Jį gavus yra žiūrima ar pirmas žodis yra blokų, reikalingų vartotojo programai skaičius. Jei ne, yra kuriamas ir atlaisvinamas resursas „Eilutė atmintyje”, su parametrais: eilute “Nenurodyti reikiamų blokų skaičiaus vartotojo programoje” ir informacijos, kad ši eilutė yra supervizorinėje atmintyje. Atlaisvinęs pranešimą procesas vėl blokuojasi. Jei reikiamas blokų skaičius rastas, einama per programą, kol nesutinkamas HALT žodis arba praeinami visi vartotojo programai išskirti žodžiai. Jei bloko HALT nėra, yra kuriamas ir atlaisvinamas resursas „Eilutė atmintyje“ su parametrais “Nėra HALT bloko”. Jei programa yra validi, atlaisvinamas resursas, skirtas MainProc procesui, kuriame laikoma atminties vieta išoriniame faile, kur yra vartotojo programa. Dabar Validator vėl blokuojasi laukdamas pranešimo iš ReadFromInterface.

Blokavimas laukiant „Užduotis išorinėje atmintyje“ resurso

Blokavimas laukiant „Trečiasis kanalas“ resurso

Ar tai blokų kiekis reikalingas programai?

Skaitomas žodis iš išorinės atminties

Skaitomas žodis iš išorinės atminties

Ar tai HALT arba programos pabaiga?

Ne

Taip

Ne

Ar tai HALT?

Atlaisvinamas resursas „Validi programa išorinėje atmintyje“

Atlaisvinamas resursas „Trečiasis kanalas“

* + 1. Procesas MainProc

Procesą MainProc kuria ir naikina procesas StartStop. Šio proceso paskirtis - kurti ir naikinti procesus JobGorvernor. Šis procesas pradeda savo veiklą prašydamas „Validi programa išorinėje atmintyje” resurso, kurį atlaisvina procesas Validator arba JobGorvernor. Gavęs reikalingą resursą, MainProc žiūri ar jo vykdymo laikas nėra 0. Nenulinė reikšmė – naujo JobGorvernor kūrimo ženklas, kai tuo tarpu nulinė reikšmė yra JobGorvernor naikinimo ženklas. Kurdamas JobGorvernor procesas suteikia jam „Validi programa išorinėje atmintyje ” kaip pradinį resursą. Naikinamas būtent tas JobGorvernor, kuris atsiuntė „Validi programa išorinėje atmintyje ” resursą, su nuliniu vykdymo laiku. Atlikęs savo darbą MainProc vėl blokuojasi laukdamas „Validi programa išorinėje atmintyje ”

Blokavimas laukiant „Validi programa išorinėje atmintyje“ resurso

Ar vykdymo laikas ne 0?

Kuriamas procesas JobGovernor, suteikiant jam „Validi programa išorinėje atmintyje“ kaip pradinį resursą

Naikinamas procesas JobGovernor, sukūręs gautąjį resursą

* + 1. Procesas JobGoverner

Procesus JobGoverner, kurių vienu metu gali būti keli, kuria procesas MainProc. Proceso JobGorvernor paskirtis – kurti, naikinti ir padėti procesui Virtuali mašina atlikti savo darbą. “Padėti – tai atlikti veiksmus, kurių Virtuali mašina, procesoriui dirbant vartotojo režimu, nesugeba atlikti. Vienas JobGorvernor aptarnauja vieną virtualią mašiną.

Procesas JobGorvernor pradeda darbą laukdamas varotojo atminties, kur bus perkelta vartotojo užduoties programa. Prašoma tiek takelių, kiek yra blokų. Gavęs „Vartotojo atmintis” resursą procesas blokuojasi laukdamas resurso „Trečiasis kanalas“. Gavęs šį, procesas dirba su trečiuoju kanalu nuskaitinėdamas takelius. Perkėlus vartotojo užduoties programą į vartotojo atmintį, „Trečiojo kanalo“ resursas yra atlaisvinamas ir paprašoma tiek blokų supervizorinės atminties, kiek blokų buvo išskirta virtualiai mašinai, kur bus laikoma virtualios mašinos puslapių lentelė. Dabar puslapių lentelė yra užpildoma išskirtosios vartotojo atminties adresais. „Supervizorinė atmintis“ yra atlaisvinama. Dabar yra pasiruošta virtualios mašinos kūrimui, taigi sekantis žingsnis yra kurti procesą Virtuali mašina. Dabar JobGorvernor blokuosis tol, kol negaus pranešimo iš proceso Interrupt apie įvykusį pertraukimą proceso Virtuali mašina metu.

Gavęs pranešimą apie pertraukimą, JobGorvernor stabdo procesą Virtuali mašina. Yra tikrinama ar tai įvedimo - išvedimo pertraukimas. Jei tai nėra įvedimo – išvedimo pertraukimas, naikinamas procesas “Virtuali mašina”, atlaisvinama jos užimta vartotojo atmintis ir duodamas ženklas procesui MainProc apie tai, kad šis JobGorvernor jau baigė darbą ir turi būti sunaikintas. Ženklas - sukurtas ir atlaisvintas fiktyvus resursas “Validi programa išorinėje atmintyje ” su nuliniu vykdymo laiku. Kadangi šis procesas savo darbą jau baigė, tačiau dar nėra sunaikintas, jis blokuojasi laukdamas „Neegzistuojantis” resurso. Šio resurso JobGorvernor negaus, taigi jis blokuosis tol kol nebus sunaikintas.

Anksčiau ar vėliau procesas MainProc gaus JobGorvernor atlaisvintą fiktyvų „Validi programa išorinėje atmintyje ” resursą ir sunaikins šį JobGorvernor procesą.

Jeigu tai įvedimo pertraukimas, procesas blokuojasi laukdamas „Vartotojo įvedimas” resurso,skirto būtent šiam JobGorvernor. Šioje vietoje JobGorvernor prašo resurso, kurį įves vartotojas į komandinę eilutę. Taigi vartotojas mato, kaip jo programa laukia jo įvedimo, tuo tarpu vykdydama kitas užduotis.

Gavęs reikalaujamą resursą, procesas Virtuali mašina yra aktyvuojamas ir procesas JobGorvernor cikliškai grįžta toliau blokuotis dėl “Iš interrupt” resurso.

Jei tai buvo išvedimo pertraukimas, pasiunčiame į antrąjį kanalą informaciją, esančią vartotojo atmintyje. Reikalingas adresas ateina kartu su “Interuptas ” resursu – kaip vienas jo parametrų. Ši informacija yra išvedame konsoliniame lange.

Atlikus šiuos veiksmus yra aktyvuojama virtuali mašina ir procesas užsiblokuoja laukdamas “Interuptas ” resurso.

Blokavimas laukiant „Atmintis“ resurso

Naikinti procesą „Virtuali mašina“

Išvedimo-įvedimo pertraukimas?

Ar tai išvedimo pertraukimas?

Ne

Taip

Sukuriama puslapių lentelė

Proceso „Virtuali mašina“ kūrimas

Blokavimasis laukiant „Pertraukimas“ resurso

Proceso „Virtuali mašina“ stabdymas

Atlaisvinti resursą „Vartotojo atmintis“

Atlaisvinti fiktyvų resursą „Validi užduotis išorinėje atmintyje“

Blokavimasis laukiant „Neegzistuojantis“ resurso

Blokavimasis laukiant „Antrasis kanalas“ resurso

Į antrąjį kanalą pasiunčiama informacija iš nurodytų atminties žodžių

Taip

Ne

Blokavimasis laukiant „Vartotojo įvedimas“ resurso

Blokavimas laukiant „Trečiasis kanalas“ resurso

Vyksta blokų perkėlimas

Vyksta blokų perkėlimas

Atlaisvinamas „Trečiasis kanalas“ resursas

Atlaisvinamas „Antrasis kanalas“

* + 1. Procesas VirtualMachine

Procesą Virtuali mašina kuria ir naikina procesas JobGorvernor. Virtualios mašinos paskirtis yra vykdyti vartotojo užduoties programą. Procesų VirtualMachine yra tiek, kiek yra procesų JobGorvernor.

Virtualios mašinos darbo pradžioje procesorius yra perjungiamas į vartotojo režimą. Vėliau vykdoma virtualaus procesoriaus komandų interpretatoriaus programa. Ši programa veikia tol, kol sistemoje neaptinkamas pertraukimas. Tada virtuali mašina išsaugoja savo procesoriaus būseną, perduodamas valdymas pertraukimą apdorosiančioms programoms.Apdorojus pertraukimą valdymui sugrįžus į virtualią mašiną, yra atlaisvinamas resursas “Pertraukimas”, skirtas procesui JobGoverner. Procesas identifikuos pertraukimą ir atliks nustatytus veiksmus ir priklausomai nuo situacijos arba aptarnaus arba sunaikins procesą “Virtuali mašina”A

Procesorius perjungiamas į vartotojo rėžimą

Vykdoma vartotojo programa

Vyksta blokų perkėlimas

Aptikus pertraukimą, atlaisvinamas „Pertraukimas“ resursas