**Reali mašina:**

1.Schema:

Central processor

IC

SP

PRG

IRG

ORG

HRG

TI

MODE

Spooler(Kanalų įrenginys)

1.Input device - Keyboard

2.Output device - Monitor

3.Output devie – Hard drive

Paging mechanism

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 000 |  |  |  |  |
| 001 |  |  |  |  |
| 003 |  |  |  |  |
| 004 |  |  |  |  |
| 005 |  |  |  |  |
|  |  |  |  |  |
| 006 |  |  |  |  |
| ... |  | | | |
| ... |
| ... |
| ... |
| ... |
| ... |
| ... |
| ... |
| 4091 |  |  |  |  |
| 4092 |  |  |  |  |
| 4093 |  |  |  |  |
| 4094 |  |  |  |  |
| 4095 |  |  |  |  |

Clients memory

Supervizorinė atmintis

SI

SM

2.Komponentai:

1. **Centrinis procesorius –** Įrenginys galintis dirbti dviejais rėžimais
   1. **Vartotojo rėžimas** –Yra imituojamas virtualios mašinos procesorius ir prieinama prie vartotojo atmintyje esančių programų per puslapiavimo mechanizmą.
   2. **Supervizoriaus rėžimas** - komandos iš supervizinės atminties yra apdorojamos procesoriaus ir saugoma:
      1. Kaip abdoroti sisteminius pertraukimus
      2. TI – pradinė reikšmė.
2. **Procesoriaus registrai(Dydis 4 baitai):**
   1. **IC** – Komandų skaitliukas
   2. **PRG –** Puslapiavimo registras
   3. **SP –** Stack viršūnės indeksas
   4. **HRG –** Skaitliukas(0/1) nusakantis ar vyksta įrašymas į kietąjį diską
   5. **ORG –** Skaitliukas(0/1) nusakantis ar vyksta atvaizdavimas į monitorių
   6. **IRG –** Skaitliukas(0/1) nusakantis ar vyksta įrašymas
   7. **TI** – Laiko skaitliukas
   8. **SI** – Pertraukimų registras
   9. **MODE –** Procesoriaus rėžimas
   10. **SM –** Registras rodantis į bendrą atmintį
3. **Atmintis –** Turime trijų rūšių atmintį
   1. **Vartotojo -** skirta virtualių mašinų atmintims bei puslapių lentelių laikymui.
      1. **Bendra -** kiekvienai virtualiai mašinai yra iškirta realios mašinos atmintis prieinama tik konkrečiai virtualiai mašinai. Taip pat išskirsime bendrai naudojamos atminties 1 bloką prieinamų visoms virtualioms mašinoms.(Pvz žemiau)
   2. **Supervizorinė -** atmintis, kurios reikia pačios OS poreikiams (komandos, sisteminiai kintamieji ir pan.), ji susideda iš 256 žodžių.
   3. **Išorinė -** šiuo atveju tai bus kietasis diskas. Jame gali būti koks failas.

|  |
| --- |
| 1 |
| 2 |
| 3 |
| Bendra atmintis |

SMP

VM1

VM2

VM3

1. **Kanalų įrenginys -** skirtas 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. jungiama klaviatūrą su vartotojo atmintimi.
   2. jungia ekraną su vartotojo atmintimi.
   3. jungia kietąjį diską su supervizorine bei vartotojo atmintimi.
2. **Puslapiavimo mechanizmas -** Virtualiai mašinai yra išskiriama 16 blokų (arba puslapių, bet toliau blokų). Tuos blokus kiekviena VM susinumeruoja nuo 1 iki 15, tačiau jai taip pat reikia žinoti realius šių blokų adresus. Tam yra naudojamas puslapiavimo mechanizmas. Realius bloko numerius talpinsime į puslapių lentelę. VM puslapių lentelei bus išskiriamas 1 blokas bendrojoje atminty. Lentelėje kiekvieno žodžio eilės numeris atitiks VM bloko numerį ir jame bus laikomas realus to bloko numeris.

Suskirstymas – **PRG**, sudarytas iš 4 baitų(Tegul **PRG** baitai b0b1b2b3), taigi bendrą atmintį paskirstysime tokiais intervalais

* VM(0) – [0-3FF].
* VM(1) – [0400-07FF].
* VM(2) – [0800 – 0BFF]
* VM(3) – [0C00 – 0FFF]
* VM(14) – [3800 – 3BFF]
* VM(15) – [3C00 – 3FFF]

Bendra formulė skaičiuot absoliučiam adresų intervalui.(Tegul VM numeris **x**):

Efektyvaus adreso skaičiavimui užtenka paskutinių 3 **PRG** baitų(b1b2b3):

1. **Taimeris -** skirtas užduotims suderinti. Užduotis negali trukti ilgiau nei tam tikrą T laiko momentų. Susitarsime, jog išvedimo / įvedimo operacijo reikalauja 2 laiko momentų, o kitos 1 ar 2.  
   Taigi kai VM pradeda darbą, speciali supervizorinės atminties ląstelė TI yra nustatoma tam tikrai reikšmei. Tarkime 50, tuomet kai yra įvykdoma instrukcija, TI yra mažinamas priklausomai nuo to kiek laiko momentų reikia instrukcijai. Kai TI tampa lygus nuliui, procesorius įvygdo pertraukimą  
   TI reikšmę galima nustatyti ar pakeisti supervizoriaus rėžime.
2. **Pertraukimai -** tai tam tikri signalai apie specialius įvykius. Jam įvykus vartotojo rėžime VM registrų reikšmės išsaugomos bendroje atmintyje ir procesorius perjungiamas į supervisoriaus rėžimą, kuriame nustatomas pertaukimo pobūdis bei kviečiama pertraukimą apdorojanti programa. Vėliau valdymas grįžta atgal į VM, vartotojo rėžimą ir atstatomi visi registrai.  
   Pertraukimus aptinka procesorius vygdydamas komandas.  
     
   Išskirsime trijų rūšių pertraukimus:
   1. Programiniai:
      1. SI = 1 – Atminties saugos pažeidimas
      2. SI = 2 – Blogas operacijos kodas
   2. Sisteminiai:
      1. SI = 3 – Komanda PRTS
      2. SI = 4 – Komanda PRTN
      3. SI = 5 – Komanda P
      4. SI = 6 – Komanda READ
      5. SI = 7 – Komanda RDH
      6. SI = 8 – Komanda STOP
      7. SI = 9 – Komanda LC
      8. SI = A – Komanda UC
   3. Timerio:
      1. SI = B – TI = 0
3. **Supervizorine atmintis – Joje išsaugoma SP, IC, ir VM numeris kurioje įvyko pertraukimas.**
   1. Supervizorinės atminties suskirstymas:
      1. 203 žodžiai
         1. Ti – adresas 0000
         2. Virtualių mašinų statusai – 0001-0010. Ar VM – aktyvi ar ne
         3. SI**x** – [0011 \*x; 0011\*x + 0010], x – pertraukimo numeris, x > 0
      2. 53 žodžiai – stekas, reikšmėms saugot
   2. Komandos dirbti su supervizorine atmintim(Dydis 4baitai):
      1. SVWxy – x reikšmę įrašo į supervizorinę atmintį adresu y.
      2. SVRx - nuskaito iš supervizorinės atminties reikšmę adresu x ir
      3. STOP(FFFF) – pertraukimo sustojimo komanda.
      4. MOV(register, value) – leidzia keisti registro reiskme.
      5. PUSH x – Įdėti x registrą į steko viršūnę
      6. POP x – Išimti iš steko viršūnės reikšmę ir įdėti į registą x

**Virtuali mašina:**

1. 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.

1. Virtualios mašinos schema atrodys taip:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 000 |  |  |  |  |
| 001 |  |  |  |  |
| 003 |  |  |  |  |
| 004 |  |  |  |  |
| 005 |  |  |  |  |
|  |  |  |  |  |
| 006 |  |  |  |  |
| ... |  | | | |
| ... |
| ... |
| ... |
| ... |
| ... |
| ... |
| ... |
| 251 |  |  |  |  |
| 252 |  |  |  |  |
| 253 |  |  |  |  |
| 254 |  |  |  |  |
| 255 |  |  |  |  |

Clients memory

Virtual processor

SP

IC

Virtual Input device

Virtual Output device

Plačiau apie komponentus:

I. **VM atmintis** susideda iš 256 žodžių. Kiekvienas žodis - 4 baitai. Bloką sudarys 16

žodžių. Taigi turime 16 blokų po 16 žodžių.

Atmintis bus suskirstyta į tris sritis į kurias bus įkeliamos atitinkamos programos

dalys parašytos VM programoje:

o Duomenys – 112 žodžiai (7 blokai, nuo 0-6) [tarp DATASEG ir CODESEG].

o Kodas – 112 žodžiai (7 blokai, nuo 7-13) [tarp CODESEG ir užbaigimo komandos STOP].

o Stekas – 32 žodžiai (2 blokai, nuo 14-15).

**II. VM procesorius** yra gerokai supaprastintas. Jo pagrindinė paskirtis vykdyti programą

kuri yra virtualioje atmintyje.

Procesoriaus registrai:

o SP - registras saugantis steko viršūnės žodžio indeksą.

o IC - komandų skaitliukas.

**1. VM komandų sistema.**

Visos komandos turi savo operacinį kodą. Komandų kodai gali būti 2 arba 4 baitų

o **Aritmetinės(Dydis – 2baitai):**

* ADD(0xAA) – sudeda du viršutinius steko elementus. Rezultatą padeda į steko

viršūnę ir steko rodyklę sumažina vienetu.

ST[SP – 1] = ST [SP – 1] + ST [SP]; SP--;

* SUB(0xAB) – atima steko viršūnėje esantį elementą iš antro nuo viršaus elemento.

Rezultatą padeda į steko viršūnę ir steko rodyklę sumažina vienetu.

ST [SP – 1] = ST [ SP – 1] - ST [SP]; SP--;

* MUL(0xAC) – sudaugina du viršutinius steko elementus. Rezultatą padeda į steko

viršūnę ir steko rodyklę sumažina vienetu.

ST [SP – 1] = ST [ SP – 1] \* ST [SP]; SP--;

* DIV(0xAD) – padalina antrą nuo viršaus steke esantį elementą iš viršūnėje esančio.

Rezultatą padeda į steko viršūnę ir steko rodyklę

sumažina vienetu.

[SP – 1] = [ SP – 1] / [SP]; SP--;

* NOT(0xAE) – atlieka [SP] esančio žodžio loginį neigimą (inversiją).

[SP] = !([SP])

o **Palyginimo(Dydis – 2baitai)**

* CMP(0xC0) – lygina steko viršūnėje esančius du žodžius. Ir rezultatą padeda į

steko viršūnę. 1 – jei lygūs, 0 – jei viršutinis mažesnis, 2 – jei didesnis.

[SP-1] = 0 jei ST [SP-1] > [SP]; SP--

[SP-1] = 1 jei ST [SP-1] == [SP]; SP--

[SP-1] = 2 jei [SP-1] < [SP]; SP--

o **Darbo su duomenimis / steko(Dydis 4baitai)**

* LDxy(0x80) – į steko viršūnę užkrauna reikšmę iš duomenų srities adresu 16 \*

x + y. SP--; 0 < x,y < 16

* PTxy(0x81) – steko viršūnėje esantį žodį deda į duomenų sritį nurodytu adresu

SP++; 16 \* x + y. 0 < x,y < 0xF

* PUNx(0x82) – x kaip skaičių patalpina į steko viršūnę.

SP++;[SP] = x.

* PUSx(0x83) – x kaip simbolį patalpina į steko viršūnę.

[SP] = x

o **Valdymo(Dydis – 4baitai)**

* JPxy(0xF0) – nesąlyginio valdymo perdavimo komanda. Valdymas

perduodamas kodo sričiai nurodytam adresui.

IC = 16\*x+y 0 < x,y < 0xF

* JExy(0xF1) – jei steko viršūnėje yra 1 valdymas perduodamas adresu

16\*x + y. 0 < x,y < 0xF

IF( [SP] == 1) IC = 16 \* x + y; SP--;

* JLxy(0xF2) – jei steko viršūnėje yra 0 valdymas perduodamas adresu

16\*x + y. 0 < x,y < 0xF

IF( [SP] == 0) IC = 16 \* x + y; SP--;

* JGxy(0xF3) – jei steko viršūnėje yra 2 valdymas perduodamas adresu

16\*x + y. 0 < x,y < 0xF

IF( [SP] == 2) IC = 16 \* x + y; SP--; 0 < x,y < 0xF

* STOP(FFFF) – programos sustojimo komanda.

o **Įvedimo bei išvedimo komandos(Dydis 4 baitai)**

* PRTS(0x50) – steko viršūnėje esantį žodį traktuoja kaip simbolius ir išveda į

išvedimo įrenginį.

* PRTN(0x51) – steko viršūnėje esantį žodį traktuoja kaip skaitinę reikšmę ir

išveda į išvedimo įrenginį.

* Pxyz(0x0) – į išvedimo įrenginį išveda x numeriu nurodyto atminties srities

bloko nuo y iki z žodžius (y < z). 0 < x,y,z < 0xF

Print([16\*x+i, i=y..z])

* Rxyz(0x1) – į išorinį įrenginį išveda x numeriu nurodyto atminties srities  
  bloko nuo y iki z žodžius (y < z).
* READ(0x52) – nuskaito vartotojo įvedimą kaip skaičių ir įrašo į steko

Viršūnę.

* RDHxy(0x20) – išorinėje atmintyje esantį žodį xy adresu įrašo į steką viršūnę.
* WRHxy(0x21) – steko viršūnėje esantį žodį įrašo į išorinę atmintį adresu xy.

**2. Bendravimas su įvedimo / išvedimo įrenginiais.**

Vartotojo užduoties įvedimo / išvedimo operacijos, tam yra specialios komandos

* StartIO(..) - ji kaip vieną iš argumentų ima kanalo numerį. Kanalų užimtumas nustatomas pagal kanalų (HRG,IRG,ORG) registrus. StartIO į steko viršūnę patalpina, 0 - laisvas arba 1- užimtas, reikšmę.
* FinishIO(..) – ji kaip vien1 iš argumentų ima kanalo numerį. FinishIO() į steko viršūnę patalpina, 0 - baigtas arba 1- nebaigtas, reikšmę.

o Įvedimo įrenginys – klaviatūra. Turi 16 žodžių buferį. Užsipildžius buferiui jo

turinys perkialiamas į atmintį. Tačiau jei buferis nebuvo pilnai užpildytas, visi

neužpildyti baitai užpildomi sutartiniu simboliu ‚$‘ - reiškiančiu srauto pabaigą.

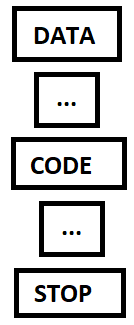
Įvedimas vyksta 1 kanalu.

o Išvedimo įrenginys – ekranas. Turi 16 žodžių buferį. Spausdina jį perduodant

informaciją iš atminties 2 kanalu.

o Išvedimo įrenginys – išorinė atmintis. Turi 16 žodžių buferį. Įrašo informaciją iš atminties 3 kanalo

1. **VM vykdymo failas.** Programos failas turi būti tam tikros struktūros:



- po DATA raktinio žodžio eina duomenys. Duomenims aprašyt naudojamos

šios „komandos“:

* + - DW X – išskiriams žodis į kurį talpinama skaitinė reikšmė X
    - DD dddd – išskiriamas žodis į kurį talpinami nurodyti simboliai
    - DN – talpinama simbolinė konstanta \n

Var = -(2+5); Programa skaiciuoja israiskos reiksme ir isveda rezultata

**DATASEG**

DW 2

DW 5

**CODESEG**

LW01

LW02

ADD

NOT

StartIO(ORG)

PRTN

**STOP**