|  |
| --- |
|  |
| Virtualiosios ir realiosios mašinų projektavimas |
| Dalyko “Operacinės Sistemos” pirma pratybų užduotis |
|  |
|  |
| **2/22/2019** |

|  |
| --- |
|  |

Centrinis procesorius

TI

PI

SI

MODE

PC

SP

PTR

Įvedimo įrenginys

Išvedimo įrenginys

Išorinė atmintis

Realios mašinos modelis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 |  |  |  |  |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| Vartotojo atmintis  …… | | | | |
| 3FD |  |  |  |  |
| 3FE |  |  |  |  |
| 3FF |  |  |  |  |
| Supervizorinė atmintis  ..... | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 |  |  |  |  |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| Virtuali atmintis  …… | | | | |
| FD |  |  |  |  |
| FE |  |  |  |  |
| FF |  |  |  |  |

Virtualios mašinos modelis

PC

SP

Virtualus procesorius

Virtualus įvedimo įrenginys

Virtualus išvedimo įrenginys

# Reali mašina

Reali mašina yra kompiuteris. Toliau nagrinėsime realią modelinę mašiną, kuri bus sudaryta

tik iš esminių komponentų:

## Centrinis procesorius

Procesoriaus paskirtis - skaityti komandą iš atminties ir ją vykdyti (interpretuoti). Dabar apžvelgsime procesoriaus registrus:

• PTR – 2 baitų puslapių lentelės registras.

• SP – 2 baitų registras saugantis steko viršūnės žodžio indeksą.

• PC – 2 baitų komandų skaitliukas.

• MODE – registras, kurio reikšmė nusako procesoriaus darbo režimą.

• PI – programinių pertraukimų registras.

• SI – supervizorinių pertraukimų registras.

• TI – taimerio registras.

## Atmintis:

## Vartotojo

Skirta virtualių mašinų atmintims bei puslapių lentelėms laikyti. Vartotojo atmintį sudaro lentelė, laikanti visą atmintį žodžiais. Žodis – 4 baitų atminties laukas. Mūsų lentelę iš viso sudarys 1024 žodžiai, kurie skirstomi į blokus po 16 žodžių. Viso lentelė turės 64 tokius žodžių blokus. Blokai sunumeruoti nuo 0 iki 63. Žodžiai blokuose sunumeruoti nuo 0 iki 15. Žodžiai lentelėje sunumeruoti nuo 0 iki 1023.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Žodžiai | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0 | 1 | 2 | 3 | 4 | . | . | . | . | . | . | . | B | C | D | E | F |  |
| Blokai | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  | . | . | . | . | . | . | . |  |  |  |  |  |  |
|  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | . |  |  | . |  |  | . |  |  |  |  |  | . |  |  | . |  |  | . |
|  | . |  |  | . |  |  | . |  |  |  |  |  | . |  |  | . |  |  | . |
|  | . |  |  | . |  |  | . |  |  |  |  |  | . |  |  | . |  |  | . |
|  | . |  |  | . |  |  | . |  |  |  |  |  | . |  |  | . |  |  | . |
|  | 3C |  |  |  |  |  | . | . | . | . | . | . | . |  |  |  |  |  |  |
|  | 3D |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 3E |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 3F |  |  |  |  |  | . | . | . | . | . | . | . |  |  |  |  |  |  |

1 pav. Vartotojo atmintis

## Supervizorinė

Supervizorinė atmintis yra skirta pačios operacinės sistemos poreikiams. Supervizorinėje atmintyje laikomi sisteminiai procesai, sisteminiai kintamieji, resursai, mikroprogramos, interpretuojančios virtualaus procesoriaus komandas.

## Išorinė

Išorinė atmintis skirta programoms laikyti. Jas galima užkrauti į realią atmintį, kur bus vykdoma. Šiuo atveju išorinė atmintis bus realizuota failu kietajame diske. Operacinė sistema žinos kelią (path) iki šio failo ir naudos jį kaip virtualų kietąjį diską - talpins programas. Šiame faile bus galima talpinti iki 256 blokų atminties (arba 4096 žodžių). Schematiškai išorinę atmintį vaizduojame analogiškai vartotojo atminčiai (Pav. 2). Procesorius tiek su atmintim, tiek su išorine atmintim bendrauja per kanalų įrenginį,Įvedimo įrenginys (klaviatūra).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Žodžiai | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | 0 | 1 | 2 | 3 | 4 | . | . | . | . | . | . | . | B | C | D | E | F |  |
| Blokai | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  | . | . | . | . | . | . | . |  |  |  |  |  |  |
|  | 3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | . |  |  | . |  |  | . |  |  |  |  |  | . |  |  | . |  |  | . |
|  | . |  |  | . |  |  | . |  |  |  |  |  | . |  |  | . |  |  | . |
|  | . |  |  | . |  |  | . |  |  |  |  |  | . |  |  | . |  |  | . |
|  | . |  |  | . |  |  | . |  |  |  |  |  | . |  |  | . |  |  | . |
|  | FC |  |  |  |  |  | . | . | . | . | . | . | . |  |  |  |  |  |  |
|  | FD |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | FE |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | FF |  |  |  |  |  | . | . | . | . | . | . | . |  |  |  |  |  |  |

2 pav. Išorinė atmintis

## Išvedimo įrenginys

## Taimerio mechanizmas

Laikysim kad įvedimo/išvedimo instrukcijos atliekamos per 3 taktus, visos kitos per 1 taktą. Dabar apie veikimo principą. Pradedant virtualios mašinos užduoties vykdymą TI registro reikšmė nustatoma tam tikrai reikšmei. Tarkim N = 10. Įvykdžius eilinę instrukciją TI reikšmė mažinama priklausomai nuo to per kiek taktų ši instrukcija yra atliekama. Kuomet TI reikšmė yra lygi nuliui, mikrokomanda Test () aptinka taimerio pertraukimą.

## Pertraukimai

Petraukimai - tai tam tikri signalai apie specialius įvykius. Gali būti aptikti tik vartotojo rėžime. Jam įvykus VM registrų reikšmės išsaugomos 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. Pertaukimus aptinka komanda Test().

Modelyje bus realizuoti 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. Apie taimerio pertraukimą jau buvo rašyta aukščiau.

Pertraukimai kils šiais būdais:

• Operacijos GET, PUT ir HALT iššauks supervizorinius pertraukimus. SI = 1 – komanda GET, SI = 2 - komanda PUT, SI = 3 – komanda HALT.

• Programiniai pertraukimai: PI = 1 – neteisingas adresas, PI = 2 – neteisingas operacijos kodas, PI = 3 – neteisingas priskyrimas, PI = 4 – perpildymas (overflow)

• Esant TI = 0 bus fiksuojamas taimerio pertraukimas.

Esant situacijai SI = 0 ir PI = 0 ir TI <> 0, pertraukimų sistema neaptiks. Pertraukimai nustatomi paprasčiausiai virtualaus procesoriaus registrams priskiriant atitinkamas reikšmes (pavyzdžiui, komandų interpretatoriui vykdant komandą GET, jis priskiria SI:= 1) Kiekvieną kartą komandų interpretatoriui įvykdžius programą, kviečiama komanda test(), kuri apklausia registrus, ir, jei kilo pertraukimas, gražina informaciją apie tai.

Po kiekvienos virtualios mašinos komandos vykdymo test() funkcija tikrina pertraukimų registrų reikšmes:

PUSH SP ; Išsaugome SP registro reikšmę steke

PUSH PC ; Išsaugome PC registro reikšmę steke

SETM ; Keičiame rėžimą iš vartotojo į supervizoriaus.

IF SI = 1 THEN

//GET funkcijos argumentas puslapiavimo mechanizmu yra paverčiamas realiu adresu ir priskiriamas kanalų įrenginio SB registrui

//Kanalų įrenginio ST registro reikšmė tampa 1 (imame takelį iš vartotojo atminties)

//Kanalų įrenginio DT registro reikšmė tampa 4 (perduodame takelį į išvedimo įrenginį)

IF SI = 2 THEN

//PUT funkcijos argumentas puslapiavimo mechanizmu yra paverčiamas realiu adresu ir priskiriamas kanalų įrenginio DB registrui

//Kanalų įrenginio ST registro reikšmė tampa 4 (imame duomenis iš įvedimo srauto)

//Kanalų įrenginio DT registro reikšmė tampa 1 (perduodame duomenis į vartotojo atimtį)

IF SI = 3 THEN

// Sustabdome virtualios mašinos darbą

IF PI != 0 THEN

// Įvyksta programinis pertraukimas, virtuali mašina baigia darbą

IF TI == 0 THEN

// Taimeriui pasiekus 0 sustabdome virtualios mašinos darbą

SETM ; Keičiame rėžimą iš vartotojo į supervizoriaus.

POP PC ; Atstatome PC registro reikšmę

POP SP ; Atstatome SP registro reikšmę

## Kanalų įrenginys

Kanalų įrenginys leidžia dirbti su atmintimis. Priklausomai nuo nustatytų registrų kanalų įrenginys gali vykdyti apsikeitimą duomenimis visomis galimomis kryptimis.Veiksmai su kanalų įrenginiu atliekami tik supervizoriaus režime. Dabar bus pateikta kanalų įrenginio vartotojo sąsaja:

Kanalų įrenginio registrai:

**CH1** : įvedimo kanalo naudojimo registras. Reikšmė keičiasi iš 0 į įvedimo įrenginio mygtuko reikšmę.

**SB**: Takelio, iš kurio kopijuosime, numeris.

**DB**: Takelio, į kurį kopijuosime, numeris

**ST**: Objekto, iš kurio kopijuosime, numeris

1. Vartotojo atmintis;

2. Supervizorinė atmintis;

3. Išorinė atmintis;

4. Įvedimo srautas;

**DT**: Objekto, į kurį kopijuosime, numeris

1. Vartotojo atmintis;

2. Supervizorinė atmintis;

3. Išorinė atmintis;

4. Išvedimo srautas;

## Realios mašinos komandos

Realios mašinos registrų nustatymo komandos (set):

* PTRxy – nustatome registro PTR reikšmę, kuris rodys puslapių lentelės bloko numerį.

PTR = x \* 16 + y.

* SPxy – nustatome steko viršūnės adresą. SP = x \* 16 + y.
* PCxy – nustatome registro PC reikšmę, PC = x \* 16 + y.
* PIx – nustatome registro PI reikšmę. PI = x, čia x = [0;4].
* SIx – nustatome registro SI reikšmę. SI = x, čia x = [0;3].
* TIx – nustatome registro TI reikšmę. TI = x.
* SETM – keičiame registro MODE reikšme jai priešinga.

IF MODE = 1 THEN MODE = 0 ELSE MODE = 1.

# 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.

## Centrinis procesorius

Centrinis virtualus procesorius yra gerokai paprastesnis. Virtualios mašinos procesoriaus paskirtis - vykdyti programą, kuri yra virtualioje atmintyje. Kiekvienas procesas turi savo virtualų centrinį procesorių, tačiau modelyje sisteminių procesų programas vykdys aukšto lygio kalbos procesorius. Taigi realiai mūsų projekte virtualius procesorius turės tik procesai – virtualios mašinos. Virtualus procesorius turi tris pagrindinius registrus:

• SP – 2 baitų registras saugantis steko viršūnės žodžio indeksą.

• PC – komandų skaitliukas.

## Virtualios mašinos atmintis

Kiekvienai virtualiai mašinai išskiriama atmintis, kurioje turi tilpti užduoties programa. Ši atmintis vadinama virtualia atmintimi. Mūsų atveju kiekvienai virtualiai mašinai bus išskiriami 16 realios atminties blokų (256 žodžiai). Kiekvienas toks virtualios atminties blokas turi realų ir virtualų adresą, kuriais atitinkamai operuoja reali ir virtuali mašinos. Ryšiai tarp realios ir virtualios atminties puslapių lentelėmis, apie kurias toliau ir kalbėsime.

## Puslapiavimo mechanizmas

Kiekvienai virtualiai mašinai išskiriame 16 iš 64 realios mašinos blokų. Šie blokai realioje atmintyje gali būti išskirti bet kuria tvarka. Puslapiavimo mechanizmas nusako ryšį tarp realaus ir virtualaus bloko adreso. Tam naudosime puslapių lentelę. Jai išskiriamas vienas blokas atminties (16 žodžių). Kiekvienas bloko žodžio indeksas nurodo virtualios mašinos atminties bloko indeksą ir kiekviename iš jų (žodyje) yra laikomas bloko numeris realioje atmintyje.

Puslapių lentelės pavyzdys:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Virtualaus bloko indeksas | 0 | 1 | 2 |  |  |  | D | E | F |
| Realaus bloko indeksas | 2 | A | 11 | . | . | . | 2A | 2F | 3D |

3 pav. Puslapių lentelė

Pavyzdyje turime virtualios atminties bloką su indeksais nuo 0 iki 15. Pagal puslapių lentelę 0 virtualaus bloko indeksas atitiks 2-ą bloko indeksą virtualioje atmintyje. Tai reiškia, kad šis realios atminties blokas (2-as) yra išskirtas virtualiai mašinai. Toliau 1-as virtualaus bloko indeksas atitiks 10-ą indeksą realioje atmintyje ir t.t.

Puslapių lentelė yra atminties blokas, todėl jis turi būti saugomas realioje atmintyje. Registras PTR laiko puslapių lentelės realaus adreso reikšmę (bloko numerį). Prieš pradedant darbą kiekviena virtuali mašina nustatys šią PTR registro reikšmę atitinkama bloko adreso reikšme, kur laikoma reikalinga puslapių lentelė.

## Registras PTR

PTR yra 2 baitų registras. Kiekvieną baitą atitinkamai pažymėkime simboliais a0 ir a1. Taip pat turime virtualios mašinos adresą, kurį simboliškai žymėsime x0x1. Čia simboliai atitinka dešimtainės sistemos skaičius. Pateiksime puslapiavimo mechanizmo reikšmių lentelė:

|  |  |
| --- | --- |
| Reikšmė | Aprašymas |
| (a0 \* 16) + a1 | Puslapių lentelės bloko numeris realioje (vartotojo) atmintyje |
| 16\*((a0 \* 16) + a1) | Puslapių lentelės bloko adresas |
| 16\*((a0 \* 16) + a1) + x0 | Bloko x0 adresas puslapių lentelėje. Jame saugomas bloko numeris į kurį atvaizduotas yra x2 blokas virtualioje mašinoje. |
| 16\*(16\*((a0 \* 16) + a1) + x0) | Virtualios mašinos bloko x0 realus bloko adresas |
| 16\*(16\*((a0 \* 16) + a1) + x0) + x1 | Realus adresas atitinkantis virtualų adresą x0x1 |

# Virtualios mašinos procesoriaus komandos

**PUSH xy** – iš atminities į steką įkeliami duomenys

**PSHC x** – į steką įkeliama konstanta

**POPM xy** – iš steko į atmintį įkeliami duomenys

**POP** – iš steko ištrinami duomenys

**TOP xy** – steko viršūnės reikšmė įkeliama į atmintį

**ADD** – sudedamos steko viršūnėje esančios reikšmės

**SUB** – atimamos steko viršūnėje esančios reikšmės

**MULT** – sudauginamos steko viršūnėje esančios reikšmės

**DIV** – padalinamos steko viršūnėje esančios reikšmės

**CMP** – palyginamos steko viršūnėje esančios reikšmės

**JZ xy** – valdymas perduodamas, jei steko viršūnėje yra 0

**JP xy** – valdymas perduodamas, jei steko viršūnėje yra teigiamas skaičius

**JN xy** – valdymas perduodamas, jei steko viršūnėje yra teigiamas skaičius

**JMP xy** – valdymas perduodamas

**PUT x** – iš įvedimo įrenginio nuskaito 16 žodžių ir įrašo juos į atmintį adresais nuo iki .

**GET x** - į išvedimo įrenginį išveda žodžius adresais adresais nuo iki pirmo nulinio baito einančio po nenulinio baito

**HALT** – programos vygdymo pabaiga

## Užduoties formatas

Užduotis, tai programa, su savo vykdymo parametrais ir duomenimis. Užduotys bus laikomos failuose. Norint sukurti naują užduotį, užtenka sukurti ir teisingai užpildyti naują tekstinį failą modelio išorėje.

**Programos antraštė.** Pradedama „$HDR”. Po to sekantys žodžiai iki žymės “$BDY” yra programos pavadinimas.

**Programos dalis**. Šiai daliai skirta 160 \* 4 = 640 baitų. Tai yra viskas, kas seka žymę “$BDY” ir baigiasi prieš pabaigos žymę “$END”.

## Programos pavyzdys

Žemiau pateikiamas programos, kuri paskaičiuoja 10 Fibonačio sekos skaicių:

$HDR

FIBONACCI

$BDY

PSHC 9 ; Dedame į steką ciklo skaitliuką (9).

PSHC 0 ; Dedame į steką nulinį Fibonačio sekos skaičių.

PSHC 1 ; Dedame į steką pirmą Fibonačio sekos skaičių.

POP E1 ; steko viršūnės elementą (1) dedame į atmintį adresu E1.

POP E0 ; steko viršūnės elementą (0) dedame į atmintį adresu E0.

JZ 1B ; ciklo pabaigos sąlyga, kai steko viršūnės elementas lygus 0, valdymas ;perduodamas adresu 1B.

PUSH E0 ; iš atminties adresu E0 dedame į steką n-2 Fib. Sekos skaičių.

PUSH E1 ; iš atminties adresu E1 dedame į steką n-1 Fib. Sekos skaičių.

ADD ; sudedame paskutinius du steke esančius elementus.

PUSH E1 ; iš atminties adresu E1 dedame į steką n-1 Fib. Sekos skaičių.

POP E0 ; steko viršūnės elementą(1) dedame į atmintį adresu (n-1 Fib. Sekos ;skaičius tampa n-2).

POP E1 ; steko viršūnės elementą(1) dedame į atmintį adresu (n Fib. Sekos ;skaičius tampa n-1).

PSHC 1 ; Dedame į steką reikšmę (1), kurią naudosime ciklo skaitliukui mažinti.

SUB ; atliekame atimties veiksmą (sumažiname ciklo skaitliuką (9-1)).

JMP 0A ; perduodame valdymą adresu 0A (tikriname ciklo sąlygą).

PUSH E1 ; išėjus iš ciklo, dedame atsakymą iš adreso E1 į steką.

POP F0 ; atsakymo reikšmę perkeliame į išvedimo takelio pradžią (F0).

GET F ; išvedame takelio F eilutę į ekraną.

HALT ; baigiame darbą.

$END

