Vilniaus universitetas

Matematikos ir informatikos fakultetas

DAUG KAS BUVO PAKEISTA DARANT ANTRĄ UŽDUOTĮ

Kurso „Operacinės Sistemos“ I-asis darbas

Virtualios ir realios mašinų projektai

*Darbą parengė:*

Jonas Daugalas (Kompiuterių mokslas, 3 kursas)

Laisvydas Skurevičius (Kompiuterių mokslas, 3 kursas)

*Pratybų dėstytojas:*

Vytautas Jančauskas

Vilnius, 2014 m.

Turinys

[1. Užduoties sąlyga (4 variantas) 3](#_Toc382183870)

[2. Reali mašina 4](#_Toc382183871)

[2.1. Realios mašinos modelis 4](#_Toc382183872)

[2.2. Realios mašinos centrinis procesorius (CPU) 4](#_Toc382183873)

[2.3. Realios mašinos atmintis 5](#_Toc382183874)

[2.4. Taimerio mechanizmas 5](#_Toc382183875)

[2.5 Pertraukimai 6](#_Toc382183876)

[2.6. Swapinimo mechanizmas 6](#_Toc382183877)

[3. Virtuali mašina 7](#_Toc382183878)

[3.1. Virtualios mašinos modelis 7](#_Toc382183879)

[3.2. Virtualios mašinos procesorius 7](#_Toc382183880)

[3.3. Virtualios mašinos atmintis 7](#_Toc382183881)

[3.4. Virualios mašinos procesoriaus komandos 8](#_Toc382183882)

[4. Puslapiavimo mechanizmas 9](#_Toc382183883)

[5. Programos failo formatas 12](#_Toc382183884)

[5.1. Programos failo pavyzdys 13](#_Toc382183885)

# 1. Užduoties sąlyga (4 variantas)

Projektuojama interaktyvi OS. Virtualios mašinos procesoriaus komandos operuoja su duomenimis, esančiais steko viršūnėje. Yra komandos duomenų persiuntimui iš atminties į steką ir atvirkščiai, aritmetinės (sudėties, atimties, daugybos, dalybos), sąlyginio ir besąlyginio valdymo perdavimo, įvedimo, išvedimo ir programos pabaigos komandos. Registrai yra du: komandų skaitiklio ir steko viršūnės. Atminties dydis yra 256 blokų po 256 žodžius (žodžio ilgį pasirinkite patys). Realios mašinos procesorius gali dirbti dviem režimais: vartotojo ir supervizoriaus. Virtualios mašinos atmintis atvaizduojama į vartotojo atmintį naudojant puslapių transliaciją. Numatytas swapping mechanizmo palaikymas, t.y. vartotojo atminties puslapiai gali būti iškeliami į išorinę atminti. Yra taimeris, kas tam tikrą laiko intervalą generuojantis pertraukimus. Įvedimui naudojama klaviatūra, išvedimui - ekranas. Yra išorinės atminties įrenginys - kietasis diskas. Vartotojas, dirbantis su sistema, programas paleidžia interaktyviai, surinkdamas atitinkamą komandą. Laikoma, kad vartotojo programos yra realios mašinos kietajame diske, į kurį jos patalpinamos „išorinėmis“, modelio, o ne projektuojamos OS, priemonėmis. Trūkstant atminties, sistema dalį vartotojo atminties puslapių (bet ne visą kažkurios VM atmintį) perkelia į išorinę atmintį, t.y. naudoja swapping'ą.

# 2. Reali mašina

Mūsų realios mašinos modelis yra virtualus kompiuteris, kurį sudaro tik esminiai komponentai:

1. Centrinis procesorius
2. Atmintis

* Vartotojo
  + Supervizorinė
  + Išorinė

1. Įvedimo įrenginys (klaviatūra)
2. Išvedimo irenginys (ekranas)

## 2.1. Realios mašinos modelis

## 2.2. Realios mašinos centrinis procesorius (CPU)

Centrinis procesorius skaito komandas iš atminties ir jas vykdo. Procesorius dirba dviem rėžimais:

* **Supervizoriaus** – aukšto lygio kalbos procesorius HLP betarpiškai vykdo iš supervizorinės atminties skaitomas komandas.
* **Vartotojo** – HLP vykdo užduoties programą imituodamas virtualų procesorių.

Centrinis procesorius turi šiuos registrus:

* **MODE** – 1 baito registras nusakantis centrinio procesoriaus darbo režimą (vartotojo arba supervizoriaus);
* **PC** – 3 baitų virtualios mašinos programos skaitiklis;
* **PTR** – 4 baitų puslapių lentelės adreso registras;
* **SP** – 3 baitų steko viršūnės adreso registras;
* **TI** – 1 baito taimerio registras;
* **SI** – 1 baito supervizorinių pertraukimų registras;
* **PI** – 1 baito programinių pertraukimų registras;
* **CH1** – nurodo įvedimo įrenginio (klaviatūra) kanalo užimtumą (0 – laisvas, 1 – užimtas);
* **CH2** – nurodo išvedimo įrenginio (ekranas) kanalo užimtumą (0 – laisvas, 1 – užimtas);
* **CH3** – nurodo išorinės atminties (kietasis diskas) kanalo užimtumą (0 – laisvas, 1 – užimtas).

## 2.3. Realios mašinos atmintis

Reali mašina turi vidinę ir išorinę atmintį. Vidinę atmintį skiriame į dvi:

* **Vartotojo atmintis** – sudaryta iš 256 blokų po 256 žodžius (žodis – 4 baitai), skirta virtualių mašinų atmintims.
* **Supervizorinė atmintis** – skirta operacinės sistemos poreikiams, puslapių lentelėms, mūsų modelyje ją valdo HLP.

Išorinė atmintis – kietasis diskas, jo viduje bus swapinimui skirtas failas (64 blokų), t.y. pritūkus vartotojo atminties, dalis jos bus iškeliama į išorinės atminties SWAP failą.

## 2.4. Taimerio mechanizmas

**Taimeris** skirtas užduočių išlygiagretinimui. Laikysime kad ta pati užduotis negali būti vykdoma ilgiau nei N laiko momentų. Laikysime kad įvedimo ir išvedimo instrukcijos atliekamos per 3 laiko momentus, visos kitos per 1 laiko momentą.

Pradedant vykdyti virtualios mašinos užduotį TI nustatoma sutarta reikšmė, pavyzdžiui TI = 20. Įvykdžius komandą TI mažinamas atitinkamu laiko momentų skaičiumi, priklausomai nuo komandos. Kai TI reikšmė pasiekia 0, kviečiamas pertraukimas.

## 2.5 Pertraukimai

**Pertraukimas** – tai signalas, kad įvyko kažkas kas reikalauja specialios paprogramės įvykdymo. Pertraukimai gali būti kelių rūšių, visi jie skirtingai apdorojami. Kiekviena pertraukimų rūšis turi savo požymių registrus, pagal kuriuos jie identifikuojami. Pertraukimai turi būti aptikti pačiai sistemai tikrinant registrus. Pertraukimai galimi tik vartotojo režime. Aptikus pertraukimą išsaugomi virtualios mašinos registrai ir pereinama i supervizorinį rėžimą, kuris įvykdo atitinkamą paprogramę. Vėliau atstatomi registrai ir darbas grąžinamas virtualiai mašinai.

Realizuosime trijų tipų pertraukimus:

1. **Programiniai** – kyla vykdant virtualią mašiną.

Požymių registras – PI(1 – neteisingas adresas, 2 – neegzistuoja operacijos kodas, 3 – nepakanka atminties išoriname diske)

1. **Supervizoriniai** – kyla, kai virtuali mašina nori vykdyti veiksmą, kuris leistinas tik supervizoriniame rėžime.

Požymių registras – SI(1 – komanda GETD, 2 – komanda PUTD, 3 – komanda HALT)

1. **Taimerio** – kyla taimerio registrui TI pasiekus 0.

## 2.6. Swapinimo mechanizmas

Operacinei sistemai pradedant darbą, supervizorinėje atmintyje išsaugomas SWAP failo adresas. SWAP failas sukuriamas tik 1, kuris operacinės sistemos darbo pabaigoje yra išvalomas. Jei pakraunant programą i vartotojo atmintį pritrūksta vietos, dalis atminties (iki 64 blokų) iškeliama į SWAP failą esantį išorinėje atmintyje. Jeigu SWAP faile nėra pakankamai laisvos vietos – įvyksta programinis pertraukimas. Vykdant programą, kai kreipiamasi į iškeltą atmintį (nustato puslapiavimo mechanizmas), SWAP faile esantis atminties blokas sukeičiamas su mažai naudojamu vartotojo atminties bloku. Atitinkamai atnaujinama puslapiavimo lentelė. SWAP failo blokai puslapiavimo lentelėje adresuojami nuo 0 iki 63, pažymint kad blokas yra iškeltas į išorinę atmintį.

# 3. Virtuali mašina

Virtuali mašina (VM) – tai virtualus realios mašinos vienetas vykdantis vartotojo programą.

## 3.1. Virtualios mašinos modelis

## 3.2. Virtualios mašinos procesorius

Virtualios mašinos procesorius yra supaprastintas realios mašinos procesoriaus variantas. Kiekviena virtuali mašina turi savo virtualų procesirių. VM procesoriaus paskirtis – vykdyti programą esančią virtualioje atmintyje.

VM procesoriaus registrai:

* **PC** – 3 baitų komandų skaitiklis (angl. Program counter)
* **SP** – 3 baitų steko viršūnės rodyklė (angl. Stack ponter)

## 3.3. Virtualios mašinos atmintis

Kiekvienas virtualios atminties blokas turi virtualų ir realų adresą. Virtualiais adresais operuoja virtuali mašina, realiais – reali mašina. Ryšiai tarp virtualių ir realių adresų nusakomi puslapių lentelėmis (žr. Puslapiavimo mechanizmas).

## 3.4. Virualios mašinos procesoriaus komandos

* **PUSH xy** – į steką įkeliami duomenys iš atminties:

[SP] = [256 \* x + y]; SP = SP + 1;

* **PSHC x** – į steką įrašoma konstanta x:

[SP] = x; SP = SP + 1;

* **POP xy** – iš steko viršūnės duomenys įrašomi į atmintį:

SP = SP - 1; [256 \* x + y] = [SP];

* **POP** – išmetami duomenys iš steko viršūnės:

SP = SP -1;

* **SET xy** – reikšmė adresu 256\*x+y įrašoma į steko viršūnę

[SP - 1] = [256 \* x + y];

* **SETC x** – į steko viršūnę užrašoma konstanta x:

[SP - 1] = x;

* **TOP xy** – adresu 256\*x+y patalpinama steko viršūnės reikšmė:

[256 \* x + y] = [SP - 1];

* **ADD**  - sudedami du skaičiai esantys steko viršuje:

[SP - 2] = [SP - 2] + [SP -1]; SP = SP - 1;

* **SUB** – iš skaičiaus po steko viršūne atimamas skaičius esantis steko viršūnėje:

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

* **MUL –** sudauginami skaičiai steko viršūnėje:

[SP - 2] = [SP - 2] \* [SP - 1]; SP = SP -1;

* **DIV** – skaičius esantis po steko viršūne dalinamas iš skaičiaus steko viršūnėje:

[SP - 2] = [SP - 2] / [SP - 1]; SP = SP - 1;

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

IF ([SP] == 0) PC = 256 \* x + y; SP = SP -1;

* **JP xy -** jei steko viršūnėje yra teigiamas skaičius (vyriausiojo baito vyriausiasis bitas yra 0), valdymas perduodamas nurodytu adresu:

IF (not BIT([SP – 1], 31)) PC = 256 \* x + y; SP = SP -1;

* **JN xy -** jei steko viršūnėje yra neigiamas skaičius (vyriausiojo baito vyriausiasis bitas yra 1), valdymas perduodamas nurodytu adresu:

IF (BIT([SP - 1], 31)) PC = 256 \* x + y; SP = SP -1;

* **JMP xy -** valdymas perduodamas nurodytu adresu:

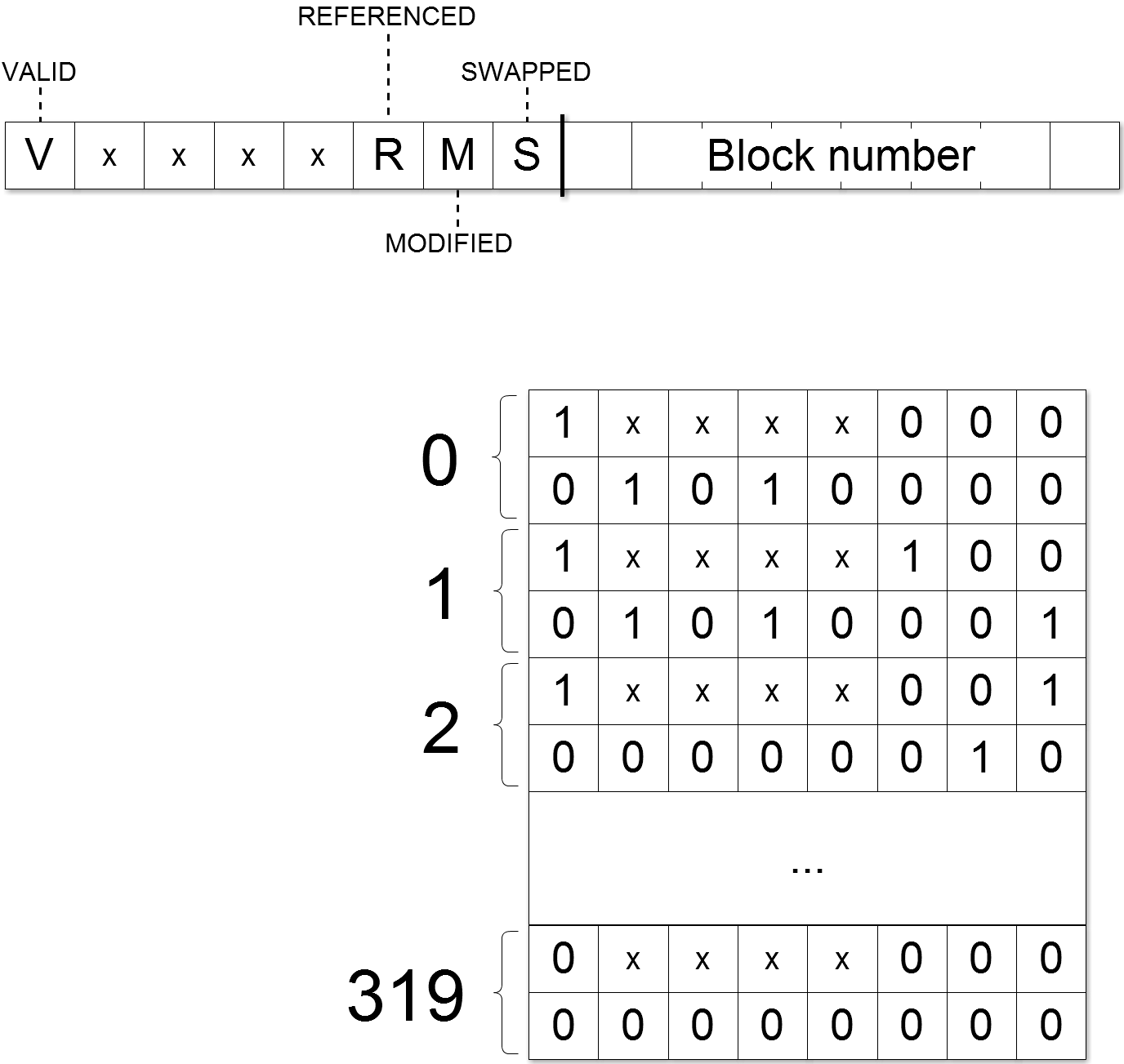
PC = 256 \* x + y; SP = SP -1;

* **GETD xy –** iš įvedimo įrenginio nuskaito 256 žodžius ir įrašo juos į atmintį adresais nuo 256\*x+y iki 256\*x+y+255 imtinai.
* **PUTD xy -**  į išvedimo įrenginį išveda žodžius adresais adresais nuo 256\*x+y iki pirmo nulinio baito einančio po nenulinio baito (t.y. nuliniai baitai esantys pradžioje yra ignoruojami, tada išvedami visi baitai iki pirmo nulinio) arba iki 256\*x+y+255 žodžio imtinai.
* **HALT**  - vartotojo programos vykdymo pabaiga.

4. Puslapiavimo mechanizmas

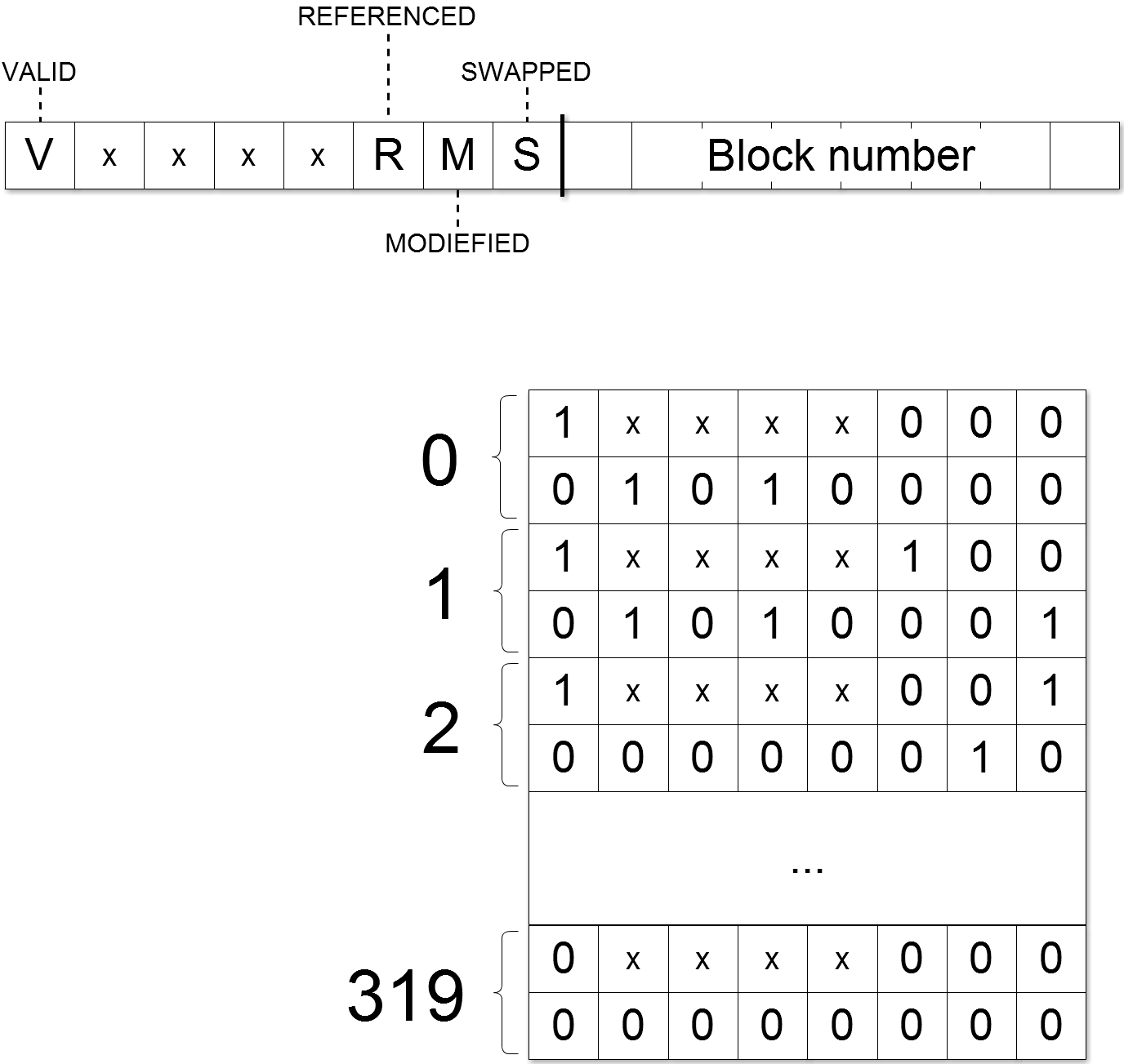
Realios mašinos vartotojo atminties dydis 256 blokų, prie jų pridėjus diskinėje atmintyje išskirtus iki 64 blokų swapping mechanizmo atminties iš viso turime 320 blokų, kuriuos gali naudoti vartotojo programos. Virtuali mašina šiems blokams naudoja savo virtualų adresavimą, kuris turi realaus adreso atitikmenį. Realaus adreso apskaičiavimui naudojamas puslapiavimo mechanizmas. Kiekvienai virtualiai mašinai sukuriama puslapių lentelė (160 žodžių) superivizorinėje atmintyje, kurioje virtualių adresų tvarka surašyti puslapiavimo lentelės įrašai~~. Kadangi minimalus vartotojo programos dydis yra 3 blokai (bent vienas blokas duomenims, bent vienas – stekui, ir bent vienas – komandoms), tai vienu metu gali būti užkrautos iki 106 virtualios mašinos (106 virtualios mašinos naudojančios po 3 blokus atminties užimtų 318 blokų iš 320 galimų), todėl supervizorinėje atmintyje yra išskirti 106 blokai puslapių lentelėms. Kadangi VM gali adresuoti iki 320 blokų, tai vienoje puslapių lentelėje bus 320 įrašų.~~ Vienas puslapių lentelės įrašas (pav. 4.1) užims 2 baitus: jaunesniajame baite bus laikomas realaus bloko numeris vartotojo atmintyje arba swapping mechanizmo faile, o vyresniajame baite išskirsime 4 bitus bendrai informacijai apie pslapį:

* **VALID** (8 bitas – vyriausias) – žymi ar šis puslapis išskirtas virtualiai mašinai. 1 – išskirtas, 0 - neišskirtas. Užkraunat vartotojo programą yra nustatomas reikalingų atminties blokų skaičius *n* ir tik pirmi *n* puslapių įrašų yra pažymimi bitu VALID = 1, visi kiti – neišskirti.
* **REFERENCED** (3 bitas) – yra nustatomas į 1, kai VM kreipiasi į šio puslapio atmintį skaitymui arba rašymui. Šis bitas naudojamas operacinei sistemai sprendžiant, kuriuos blokus perkelti į diskinę swap atmintį pritrūkus vartotojo atminties.
* **MODIFIED** (2 bitas) – nustatomas į 1, kai VM keičia atminties turinį šiame puslapyje. Šis bitas, kaip ir REFERNCED, taip pat naudojamas operacinei sistemai sprendžiant, kuriuos blokus perkelti į diskinę swap atmintį pritrūkus vartotojo atminties.
* **SWAPPED** (1bitas - jauniausias) – žymi ar puslapis yra iškeltas į swap atmintį. 1 – realus atminties blokas yra swap atmintyje, 0 – realus atminties blokas yra vartotojo atmintyje.



Pav.4.1.

Pateikiame puslapių lentelės pavyzdį (pav. 4.2). Iš lentelės matome, kad nulinis virtualios mašinos puslapis atitinka 80 vartotojo atminties bloką. Pirmasis VM puslapis atitinka 81 vartotojo atminties bloką, be to REFERENCED bitas nustatytas vienetu, kas reiškia, kad į duomenis šiame puslapyje jau buvo kreiptasi. Antrasis VM puslapis yra iškeltas swap atmintyje antrame bloke. 319 puslapio VALID bitas yra nustatytas 0, t.y. VM mašina neturėtų kreiptis į atmintį šiame puslapyje, nes atmintis tame puslapyje mašinai nebuvo išskirta.



Pav. 4.2.

Kuriant virualią mašiną, jai sukuriama puslapių lentelė. Pagal varotojo programos reikalavimus opercinė sitema bando surasti reikiamą atminties blokų kiekį vartotojo atmintyje ir užpildyti puslapių lentelę blokų numeriais. Jei vartotojo atminties neužtenka, sistema bando priskirti laisvus swap atminties blokus VM puslapiams. Jei neužtenka ir swap atminties, įvyksta programos užkrovimo klaida, VM nėra sukuriama.

Virtualiai mašinai bandant kreiptis adresu priklausančiu puslapiui, kuris atitinka realų bloką swap atmintyje, yra kviečiamas swapping mechanizmas, kuris sukeičia reikiamą bloką su mažai naudojamu bloku vartotojo atmintyje. Tumet atnaujinama puslapių lentelė ir tik tada atliekamas VM kreipinys į atmintį, tačiau jau veiksamai atliekami su vartotojo atmintimi.

Puslapių lentelė turi savo adresą supervizorinėje atmintyje, kuris darbo metu yra saugomas centrinio procesoriaus registre PTR. Prieš suteikiant virtualiai mašinai realų procesorių, PTR yra nustatomas pagal tos mašinos puslapių lentelės adresą.

Virtualiai mašinai atliekant kreipinius į atmintį, ji dirba su virtualiais adresais, tačiau realus procesorius turi dirbti realiais adresais. Tarkime VM atlieka kreipinį adresu x2x1x0 (x2x1 – puslapio numeris, x0 – poslinkis puslapyje), procesoriaus registre PTR reikšmė a3a2a1a0, tada transformacija į realų adresą vyksta tokiu būdu:

* Einamosios VM puslapių lentelės nulinio įrašo adresas:  
  Y1 = a3a2a1a0
* Reikiamo puslapio įrašo žodžio adresas:

Y2 = Y1+ (x2x1/2)

* Jei BIT([Y2], ((x2x1+1) mod 2)+8) = 1, kviečiamas swapping mechanizmas, kad būtų blokas iškeliamas iš swap atminties į vartotojo. Po to atnaujinama puslapių lentelė.
* Realaus bloko numeris:

Y3 = ([Y2] mod ((x2x1 mod 2)\*2^16 + 256) ) / ((x2x1 mod 2)\*2^16)

* Realaus bloko adresas:

Y4 = Y3\*256

* Realus kreipinio adresas:

Y5 = Y4 + x0

= (([a3a2a1a0+ (x2x1/2)] mod ((x2x1 mod 2)\*2^16 +256))/((x2x1 mod 2)\*2^16))\*256+ x0

Funkcija BIT(x, y) grąžina x reikšmės y-tojo bito reikšmę (čia nulinis bitas yra jauniausiojo baito jauniausiasis, o 31 bitas yra vyriausiojo baito vyriausiasis).

Laužtiniais skliaustais [x] žymime, kad paimama reikšmė realiu adresu x.

# ~~5. Programos failo formatas~~

~~Programos failas – tai tris segmentus aprašantys blokai atskirti specialiais žymėjimais. Pirmasis blokas aprašo duomenų segmentą, antrasis – steko, trečiasis – kodo.~~

~~Duomenų segmentą aprašantis blokas pradedamas antrašte~~ *~~DATA <DATASIZE>~~*~~, kur <DATASIZE> yra duomenų segmento dydis blokais. Duomenims skiriami adresai nuo~~ *~~0~~* ~~iki~~ *~~DATASIZE\*256~~*~~‑ 1. Po antrašte rašomi duomenis aprašantys elementai, kurių stuktūra yra:~~

~~<adresas> <reikšmių įrašymo būdas> <reikšmė1>, <reikšmė2>, ..., <reikšmėN>~~

~~<adresas> nurodo, kur bus išsaugota pirmoji reikšmė, <reikšmių įrašymo būdas> - gali būti „b“ arba „w“ atitinkamai nurodantis įrašyti reikšmes kaip baitus arba kaip žodžius. Reikšmės gali būti nurodomos sveikaisiais skaičiais, simboliais tarp viengubų kabučių (įrašomas to simbolio ASCII kodas), simbolių eilutėmis tarp dvigubų kabučių (iš eilės įrašomi kiekvieno eilutės simbolio ASCII kodai) arba komanda <size> dup(<reikšmė>), kuri <size> kartų nuosekliai įrašo <reikšmė>. Toks duomenis aprašantis elementas, priklausomai nuo pasirinkto įrašymo būdo (po baitą arba po žodį), pradedant nurodytu adresu iš eilės į atmintį surašo visas pateiktas reikšmes.~~

~~Steko segmentą aprašantis blokas pradedamas antrašte~~ *~~STACK <STACKSIZE>~~*~~, kur <STACKSIZE> yra steko segmento dydis blokais. Stekui skiriami adresai nuo~~ *~~DATASIZE~~*~~\*256 iki (~~*~~DATASIZE~~*~~+~~*~~STACKSIZE)~~*~~\*256 ‑ 1. Steko segmentą aprašantis blokas turi būti tuščias, todėl antraštė yra pirmas ir paskutinis bloką aprašantis elementas.~~

~~Kodo segmentą aprašantis blokas pradedamas antrašte CODE, po kurios seka programos komandos.~~

**­­­BUVO PAKEISTA (segmentų dydžiai nustatomi automatiškai, steko segmentas paskutinis ir plečiasi automatiškai, prireikus išskiriama papildoma atmintis). Žr. Antrą užduotį.**

## 5.1. Programos failo pavyzdys

DATA 4 ;duomenų segmentas nuo 0 iki 1023

;b - priskirti po baitą

;w - priskirti po žodį

0 b "Enter a digit", 10, 13, 0 ;16B

4 b "Enter another digit", 10, 13, 0 ;22B

12 w 0 ;4B

30 b " is equal to ", 0 ;14B

40 b " is greater than ", 0 ;18B

50 b " is less than ", 0 ;15B

255 b "end", 0 ;4B

256 b "star" ;4B

257 b 't', 0, 0, 0 ;4B

512 w 256 dup(0) ;1024B

STACK 1 ;steko segmentas nuo 1024 iki 1279

CODE ;kodo segmentas nuo 1280 iki 1307

PSHC 30 ;į steką įrašoma konstantą 30

PUTD 0 ;į ekraną išvedama eilutė adresu 0

GETD 512 ;nuskaitomi duomenys iš išorinio įrenginio į atmintį nuo 512 iki 767 žodžių

PUSH 512 ;į steką įrašomas žodį adresu 512

TOP 10 ;steko viršūnės žodis išsaugomas atmintyje adresu 10

ADD ;sudedamos dvi viršutines steko reikšmes (skaitmuo verčiamas į jo ASCII kodą)

TOP 11 ;rezultatas išsaugomas adresu 11

SET 10 ;steko viršūnėje esantis žodis perrašomas žodžiu, kurio adresas 10

PUTD 4 ;į ekraną išvedama eilutė adresu 4

GETD 512 ;nuskaitomi duomenys iš išorinio įrenginio į atmintį nuo 512 iki 767 žodžių

PUSH 512 ;į steką įrašomas žodį adresu 512

TOP 13 ;steko viršūnės žodis išsaugomas atmintyje adresu 13

SUB ;iš antros nuo viršaus steko reikšmės atimama steko viršūnės reikšmė

PUTD 11 ;į ekraną išvedama reikšmė adresu 11 (12 žodis turi 4 nulinius baitus)

JZ 1297 ;jei skirtumas lygus nuliui, šokama į komandą adresu 785

JP 1299 ;jei skirtumas - teigiamas skaičius, šokama į komandą adresu 787

JN 1301 ;jei skirtumas - neigiamas skaičius, šokama į komandą adresu 789

PUTD 30 ;(1297)

JMP 1302

PUTD 40 ;(1299)

JMP 1302

PUTD 50 ;(1301)

PSHC 30 ;(1302) į steką įrašoma konstantą 30

PUSH 13 ;į steką įdedamas antras nuskaitytas skaičius (skaitmuo)

ADD ;sudedamos dvi viršutines steko reikšmes (skaitmuo verčiamas į jo ASCII kodą)

TOP 11 ;rezultatas išsaugomas adresu 11

PUTD 11 ;į ekraną išvedama reikšmė adresu 11 (12 žodis turi 4 nulinius baitus)

HALT