

VILNIAUS UNIVERSITETAS  
MATEMATIKOS IR INFORMATIKOS FAKULTETAS

## **Realios ir virtualios mašinos projektas**

Operacinės sistemos

Atliko: Tomas Kozakas, Daniil Samuilov

Vertino: Partn. Doc. Rokas Masiulis

Vilnius  
2024

# Turinys

<b>1. Užduoties tikslas</b>	<b>3</b>
1.1. Užduotys . . . . .	3
<b>2. Realioji mašina</b>	<b>4</b>
2.1. Realios mašinos komponentai . . . . .	4
2.2. Realios mašinos modelis . . . . .	4
2.2.1. Realios mašinos centrinis procesorius . . . . .	5
2.2.2. Virtualios mašinos komandos . . . . .	5
2.2.3. Realios mašinos atmintis . . . . .	6
2.2.4. Atminties konfigūраторius . . . . .	6
2.2.5. Kanalių sistema . . . . .	6
2.2.6. Taimerio mechanizmas . . . . .	6
<b>3. Virtualioji mašina</b>	<b>8</b>
3.1. Virtualios mašinos komponentai . . . . .	8
3.2. Virtualios mašinos modelis . . . . .	8
3.2.1. Virtualios mašinos procesorius . . . . .	9
3.2.2. Virtualios mašinos atmintis . . . . .	9
3.2.3. Komandų sistema . . . . .	10
3.3. Puslapiavimo mechanizmas . . . . .	11
3.4. Virtualios mašinos kūrimo ir veiklos scenarijus . . . . .	11
3.5. Pertraukimų mechanizmas . . . . .	12
3.6. Bendravimo su įvedimo/išvedimo įrenginiais mechanizmas . . . . .	12
3.7. Virtualios mašinos programos pavyzdys . . . . .	13

## **1. Užduoties tikslas**

Projekto tikslas - sukurti virtualią mašiną, tinkamą naudoti kartu su operacine sistema.

### **1.1. Užduotys**

- Sukurti virtualią mašiną, kuri būtų pritaikyta naudojimui su operacine sistema.
- Suteikti galimybę realizuoti operacinės sistemos tikslus naudojant tiek realią, tiek virtualią mašiną.
- Sukurti metodą operacinei sistemai dinaminį resursų valdymui.
- Užtikrinti, kad operacinė sistema galėtų veikti interaktyviai, leidžiant vartotojams paleisti programas ir bendrauti su jomis.
- Realizuoti Turingo pilnosios sistemos principus.
- Leisti programuotojams kurti ir iškviesti funkcijas naudojant rekursiją.

## 2. Reali mašina

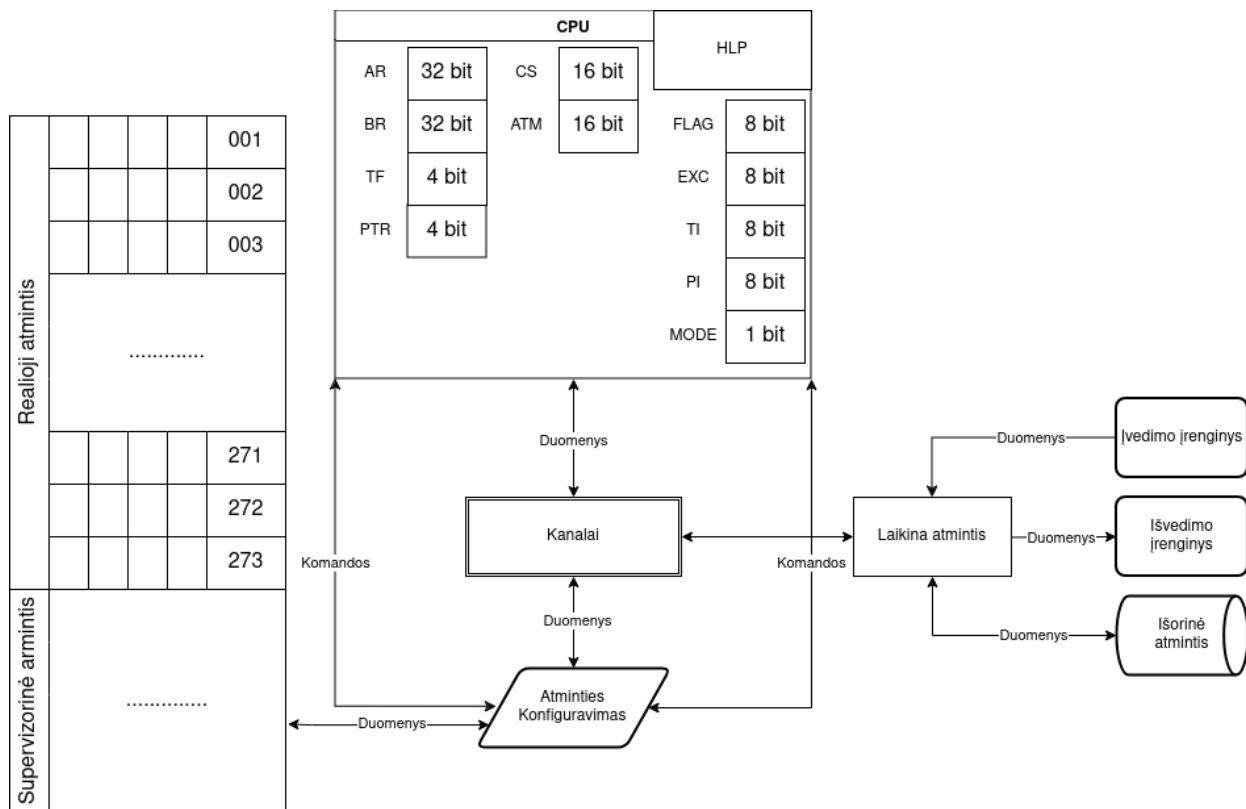
Reali mašina, arba kompiuteris, yra esminis įrankis, leidžiantis vykdyti įvairias operacijas. Kiekvienas procesas, kuris įvyksta šiame įrenginyje, susideda iš trijų būtinų elementų: programos, pradinųjų duomenų bei nustatymų, kurie leidžia šiai programai veikti.

### 2.1. Realios mašinos komponentai

Mūsų realioji mašina susideda iš septynių pagrindinių elementų:

1. Pagrindinis skaičiavimo modulis CPU
2. Atmintis
  - Supervisorinė atmintis
  - Vartotojo
  - Virtuali
3. Įvedimo/Išvedimo įrenginiai
4. Atminties konfigūраторius
5. Kanalo sistema
6. Išoriniai saugojimo įrenginiai

### 2.2. Realios mašinos modelis



2.1 pav. Realios mašinos modelis

### 2.2.1. Realios mašinos centrinis procesorius

Mūsų CPU turi 12 pagrindinius registrus (žr. 2.1 lentelė). Diagramoje parodytas kiekvieno iš jų užimamos atminties kiekis (žr. 2.1 pav.).

2.1 lentelė. CPU pagrindinių registrų apžvalga

Registras	Aprašymas
AR	Skirtas saugoti bendrą informaciją, su kuria šiuo metu dirbama.
BR	Taip pat kaip AR, skirtas saugoti bendrą informaciją.
PTR	Nurodo virtualios mašinos numerį.
TI	Laikmatis, nurodantis, kiek laiko procesas veikia. Po 10 ciklų vyksta perėjimas prie kito proceso.
TF	Teisinga arba neteisinga (sprendimo atsakymas).
CS	Komandų skaitiklis, nurodo, kuri komanda šiuo metu vykdoma. Kiekvieną kartą, kai vykdoma nauja komanda, skaitiklis padidėja.
ATM	Nurodo atminties ląstelę, su kuria šiuo metu dirbama.
FLAG	Dirba su išoriniais įrenginiais (įvedimo, išvedimo, laikina atmintis).
EXC	Pertraukimai.
PI	Visos išimtys, susijusios su programos (VM) vykdymu.
MODE	Procesoriaus režimas.

### 2.2.2. Virtualios mašinos komandos

Atminties langeliai, užimantys keturis baitus, gali būti suprantami kaip komanda ir kaip duomenys – tai priklauso nuo paskutinio bito. Mikroprogramos, interpretuojančios virtualaus procesoriaus komandas bus vykdomos HLP.

### 2.2.3. Realios mašinos atmintis

Zodis/ Eilutė		Atmintis															
		01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
Puslapavimo lentelė	01																
	02																
	:	...															
	15																
	16																
Proc	17																
Virtuali atmintis	18																
	19																
	:	...															
	272																
	273																

2.2 pav. Atmintis

Mūsų tikroji atmintis yra padalinta į du blokus: supervizoriaus atmintį, kurioje saugoma visa operacinės sistemos darbui reikalinga informacija, ir vartotojo atmintį, kurioje saugomos visos virtualios mašinos. Atmintis padalinta į 4 baitų blokus, kiekvienas blokas sudarytas iš 16 eilučių. Kiekvienos 16 eilučių dalis sudaro vieną skyrių, o yra 16 skyrių. Turint vietas, reikalingos puslapių numeravimui, yra 4368 blokų po keturis baitus, neskaičiuojant dalies, kurią užima operacinė sistema.

### 2.2.4. Atminties konfigūраторius

Atminties konfigūраторius atsakingas už tai, kad tarnautų kaip tam tikra kliūtis arba barjeras tarp centrinio procesoriaus ir išorinės informacijos, suteikdamas papildomo saugumo ir veikimo stabilumo. Sistema dirba su šiuo moduliu standartiškai, kai veikia neautomatiniu būdu. Pagrindinė šio modulio užduotis – kaupti informaciją, kol procesorius galės ją apdoroti.

### 2.2.5. Kanalo sistema

Kanalo sistema buvo perkelta į atskirą struktūrą, siekiant parodyti, kad turime tam tikrus modulius, atsakingus už tai, kad perduodamoje informacijoje nebūtų konfliktų. Papildoma pastaba: visa informacija būtinai praeina per šį modulį. Kitos rodyklės, jungiančios procesorių su atminties konfigūраторiumi bei procesorių su atminties prižiūrėtoju, yra skirtos parodyti, kad procesorius valdo šiuos modulius, tačiau jis negauna informacijos tiesiogiai iš jų.

### 2.2.6. Taimerio mechanizmas

Registeris **TI** (laikmačio registras) yra naudojamas operacinėse sistemose, kad apribotų vieno proceso veikimo laiką centrinio procesoriaus resursų naudojimo atžvilgiu. Tai užkerta kelią tam, kad vienas procesas negalėtų monopolizuoti CPU laiko ir užtikrina, kad kiti procesai taip pat galėtų

gauti procesoriaus laiko.

Štai kaip **TI** gali veikti sistemoje:

- Kai procesas pradeda vykdyti, **TI** registras nustatomas į tam tikrą pradinę reikšmę 16. Kiekvienas procesoriaus ciklas (arba kitas iš anksto apibrėžtas laiko vienetas) mažina šį registrą vienetu.
- Kai **TI** pasiekia nulį, tai reiškia, kad proceso laikas baigėsi. Operacinė sistema tai atpažįsta kaip signalą, kad reikia perjungti kontekstą – sustabdyti dabartinį procesą ir perduoti CPU kontrolę kitam procesui, kuris laukia savo eilės.
- Konteksto perjungimas gali apimti dabartinio proceso būsenos (įskaitant visus registro reikšmes) išsaugojimą, kad vėliau procesas galėtų būti tęsiamas nuo tos pačios vietos.
- Naujai pradedamam procesui **TI** registras vėl nustatomas į pradinę reikšmę, ir ciklas tęsiasi.

### 3. Virtuali mašina

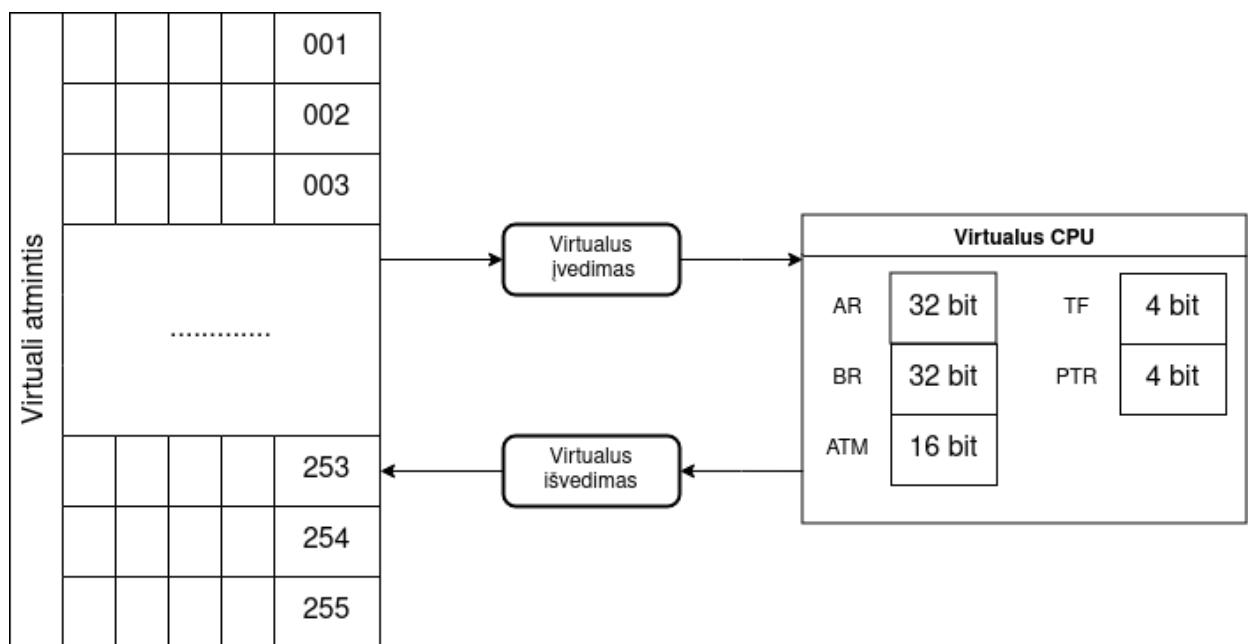
Virtuali mašina (VM) sukuria simuliaciją, kurioje programos gali veikti taip, tarsi kiekviena iš jų turėtų savo asmeninę kompiuterio sistemą, nepaisant to, kad realybėje jos bendrai naudoja tas pačias fizinio kompiuterio išteklius. Kiekvieną virtuali mašina konkuruoja dėl realaus procesoriaus laiko ir programoms veikia izoliuotai viena nuo kitos, taip užtikrinant didesnę saugumą, lankstumą ir resursų naudojimo efektyvumą.

#### 3.1. Virtualios mašinos komponentai

Virtualios mašinos struktūra yra daug paprastesnė, palyginti su realiaja. Ji susideda iš keturių dalių, iš kurių dvi gali būti sujungtos.

1. Virtualus procesorius
2. Virtualus atmintis
3. Įvedimo/Išvedimo sistema

#### 3.2. Virtualios mašinos modelis



3.1 pav. Virtualios mašinos modelis



### 3.2.1. Virtualios mašinos procesorius

Pradėkime nuo širdies, tiksliau, procesoriaus. Čia yra į registrai: **AR**, **BR** registrai bei **ATM**, **TF**, **PTR**. Jie atlieka tą patį vaidmenį kaip ir anksčiau (žr. 2.1 lentelė). Tokios kaip sudėjimas, atėmimas, palyginimas, bus atliekamos tarp dviejų registrų **AR** ir **BR**.

### 3.2.2. Virtualios mašinos atmintis

Kiekvienai virtualiai mašinai skiriame šešiolika eilučių po 16 blokų. Primenu, kad blokas susideda iš keturių baitų

Peršokimo operacijai naudojama informacija yra saugoma **PTR** registeryje, o **MODE** registras nurodo, ar operacija yra leidžiama, ir ši operacija nereikalauja jokių papildomų operatorių. **PTR** registras atsako už konkrečios atminties ląstelės nustatymą, kuri turi būti pasiekiamą, o **MODE** registras nurodo, ar prieiga prie šios ląstelės yra leidžiama.

**FLAG** registras yra skirtas dirbti su sistemos vidine logika, pvz., nustatyti lyginimo operacijų rezultatus arba signalizuoti apie perpildymą atliekant aritmetines operacijas. Tuo tarpu **EX** registras yra susijęs su išoriniais įrenginiais, nurodydamas, pavyzdžiui, kada informacija gali būti įkelta į kompiuterį, CPU yra pasirengęs vykdyti kitą užduotį, arba signalizuojant apie gedimus.

Svarbu pabrėžti, kad **FLAG** registras yra saugomas virtualioje atmintyje, skirtingai nei **EX** registras, kas padidina sistemos saugumą.

### 3.2.3. Komandų sistema

3.1 lentelė. Virtualios mašinos komandų sistemos lentelė

Komanda	Argumentų perdavimas	Rezultatas
<b>Aritmetinės Operacijos</b>		
ADD	AR, BR	AR
SUB	AR, BR	AR
MUL	AR, BR	AR
DIV	AR, BR	AR, BR (liekana)
NEG	AR	AR
<b>Logikos Operacijos</b>		
AND	AR, BR	AR
OR	AR, BR	AR
NOT	AR	AR
<b>Palyginimo Operacija</b>		
CMP	AR, BR	TF: 0 AR < BR, 1 AR > BR, 2 AR = BR
<b>Duomenų Užkrovimas ir Saugojimas</b>		
LD	adresas	Įkrauna duomenis iš nurodyto adreso į AR registrą
ST	ATM	Saugo duomenis iš AR registro į nurodytą adresą
MOVE	$x_1, x_2$ (registrai arba atminties adresai)	Užnulina $x_1$ , perkelia $x_2$ į $x_1$
VAL	$x_1$ - adresas, $x_2$ - reikšmė	Užnulina $x_1$ , perkelia $x_2$ į $x_1$
<b>Valdymo Operacijos</b>		
JM	adresas	Perkelia adresą į ATM
JL	adresas	Perkelia adresą į ATM
JG	adresas	Perkelia adresą į ATM
JMR	adresas	Perkelia adresą į ATM
JLR	adresas	Perkelia adresą į ATM
JGR	adresas	Perkelia adresą į ATM
HALT	-	EXC
CHAR	kanalas paruoštas	EXC
MEMR	atmintis paruošta	EXC
PROR	programa paruošta	EXC
<b>Įvedimo/Išvedimo Operacijos</b>		
PRINT	adresas	EXC

### 3.3. Puslapiavimo mechanizmas

Šiame skyriuje aptariamas virtualios atminties adresų konvertavimas į realius adresus naudojant puslapiavimo metodą. Mūsų atminties valdymo schema rezervuoja pirmąsias 256 atminties ląsteles puslapių lentelės duomenims. Virtuali mašina naudoja atminties ląsteles nuo 256 iki 272, o reali atmintis prasideda nuo 273 iki 4368 ląstelės.

Adreso konvertavimo procesą galima aprašyti šiais žingsniais:

1. **Puslapių lentelės indeksas** skaičiuojamas pagal formulę:

$$\text{Puslapių lentelės indeksas} = (PTR \times 16) + (\text{Adresas}/16),$$

kur  $PTR$  yra puslapių lentelės registras, nurodantis į puslapių lentelę atmintyje, ir  $\text{Adresas}$  yra virtualus adresas.

2. **Realus Adresas** apskaičiuojamas pagal formulę:

$$\text{Realus Adresas} = (\text{Puslapio numeris} \times 16) + (\text{Adresas} \bmod 16),$$

kur:

- **Puslapio numeris** gaunamas iš puslapių lentelės naudojant anksčiau apskaičiuotą indeksą.
- $\text{Adresas} \bmod 16$  nurodo baido numerį puslapyje.

### 3.4. Virtualios mašinos kūrimo ir veiklos scenarijus

1. Pradėta kurti virtuali mašina.
2. Virtuali mašina reikalauja 16 takelių atminties savo reikmėms.
3. Yra išskiriama 16 takelių virtualiai mašinai.
4. Yra išskiriamas 1 takelis virtualios mašinos puslapių lentelei.
5. Puslapių lentelė (t.y. tas takelis) yra užpildomas išskirtų 16 takelių realiais adresais.
6. Virtualios mašinos virtualaus registro  $PTR$  reikšmei priskiriamas puslapių lentelės takelio realus adresas.
7. Virtuali mašina baigiama kurti.
8. Virtuali mašina gauna procesorių.
9. Virtualiai mašinai prireikė paversti virtualų adresą 128 (t.y. 8 takelis, parinkta atsitiktinai) realiu.
10. Virtuali mašina iš puslapių lentelės nuskaity 5 žodį. Tai ir yra realus adresas.
11. Taip toliau

### 3.5. Pertraukimų mechanizmas

Veikiant virtualioms mašinoms, tokiu atveju gali kilti tam tikrų problemų. Atėjus laikui, operacinės sistemai veikiant, iškviečiami pertraukimai. Ji nuskaito duomenis iš registrų ir, priklausomai nuo jų turinio, iškviečia vieną iš integruotų programų, kurios yra operacinės sistemos dalis. Pertraukimai gali kilti šiais būdais:

- **PI Register:**
  - Netinkama atmintis (Memory Fault)
  - Atminties persipildymas (Memory Overflow)
- **FLAG Register:**
  - Reikšmės už ribų (Value Out of Range)
  - Dalijimosi iš nulio klaida (Division by Zero Error)
- **EXC Register:**
  - Neįprastų instrukcijų kodas (Illegal Instruction)

### 3.6. Bendravimo su įvedimo/išvedimo įrenginiais mechanizmas

Ryšys su įvesties ir išvesties įrenginiais vyksta tik per operacinę sistemą; tai reiškia, kad virtualios mašinos neturi tiesioginės prieigos prie išvesties įrenginių ne per operacinę sistemą. Komunikacija tarp CPU ir išorinių įrenginių vykdoma per **EXC** registrą, o informacija gaunama naudojant standartinius kanalus. Pirma, informacija patenka į **supervizorinę atmintį** ir lieka ten, kol procesorius gali ją priimti ar išvesties įrenginys yra pasiruošęs ją apdoroti. Be to, informacija perduodama per kanalų sistemą, kuri užkerta kelią informacijos praradimui perdavimo metu ir apsaugo nuo duomenų maišymosi, kai šie gaunami iš skirtingų šaltinių. Kai informacija patenka tiesiai į procesorių, jis nustato, kur ją įrašyti, ir tada informacijos įkėlimo operacija atliekama naudojant **PTR** registrą.

### 3.7. Virtualios mašinos programos pavyzdys

Šiame skyriuje pateikiamas virtualios mašinos programos kodas, kuris atlieka paprastą aritmetinę operaciją: dviejų skaičių sumavimą. Programa naudoja anksčiau aprašytas komandas (žr. 3.1 lentelė).

#### DATA\_SEGMENT

```
VAL 3      ; Pirmas operandas
VAL 5      ; Antras operandas
result VAL 0 ; Rezultato kintamasis
```

#### CODE\_SEGMENT

```
LD var1      ; Užkrauna reikšmę į AR registrą
MOVE BR, AR  ; Užkrauna reikšmę iš AR į BR registrą
LD var2      ; Užkrauna reikšmę į AR registrą
ADD          ; Prideda BR prie AR registro turinio
PT result    ; Saugo gautą sumą viduj AR į 'result' kintamąjį
PRNTC result ; Spausdina gautą rezultatą
HALT         ; Sustabdo programos vykdymą
```