# Kurso „Operacinės Sistemos“ I-asis darbas
## Virtualios ir realios mašinų projektai

2014 vasario 23 d.

*Darbą parengė:*
Lukas Klusis (Matematinė informatika, 3 kursas)
Justinas Gilvonauskas (Matematinė informatika, 3 kursas)

*Pratybų dėstytojas:*
Vytautas Jančauskas

# Turinys

# 1. Užduoties sąlyga

Penktoji užduotis iš pateiktų visų užduočių.

Projektuojama paketinė operacinė sistema:

- 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,
    - šakojimo (sukuria proceso kopiją; pagal jos grąžinamą reikšmę procesas nustato, ar jis yra tėvinis, ar vaikinis)
    - programos pabaigos komandos.
    - Registrai yra du:
        - komandų skaitiklio
        - steko viršūnės.
    - Atminties dydis yra 16 blokų po 16 žodžių (žodžio ilgį pasirinkite patys).

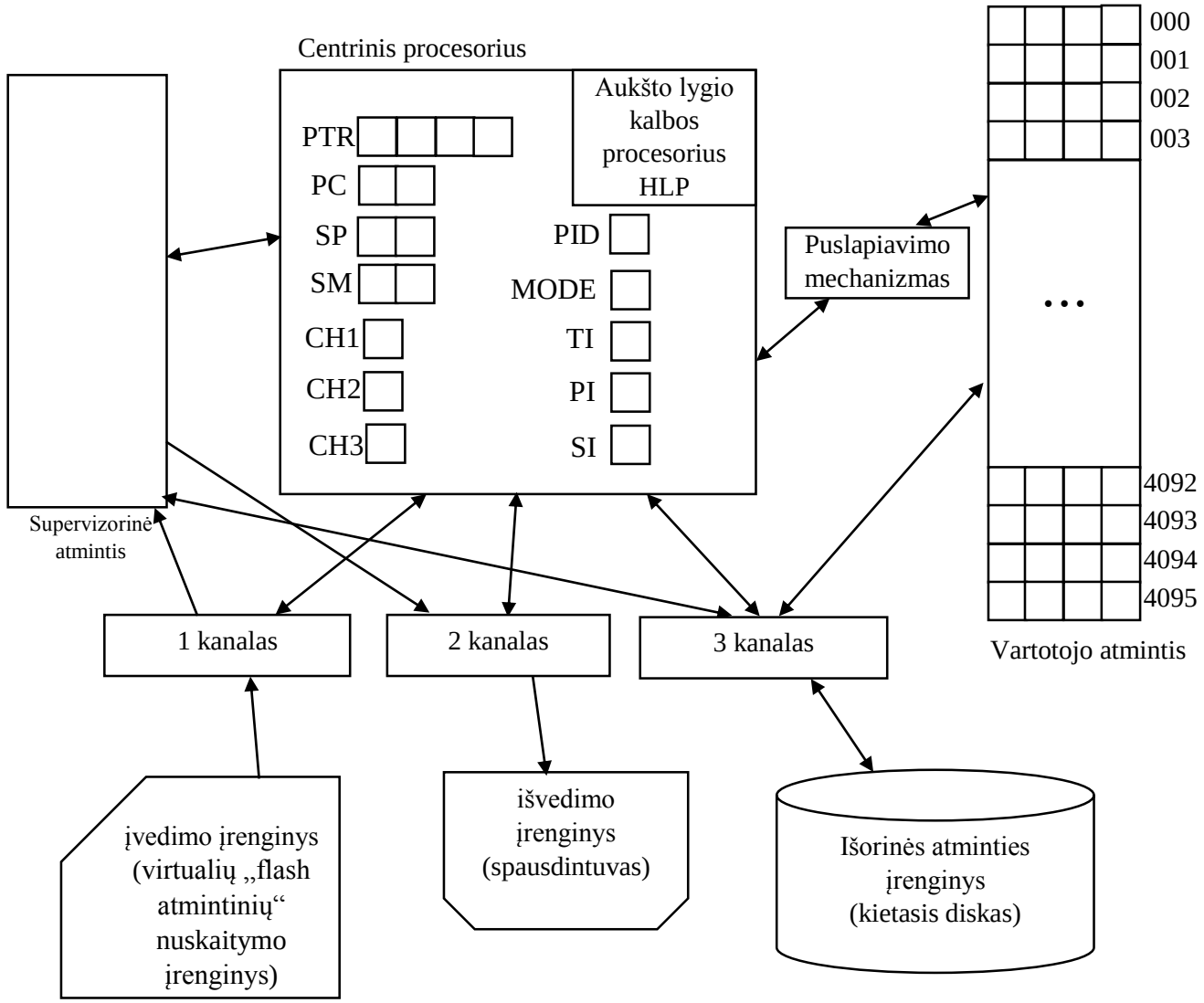
- Realios mašinos procesorius gali dirbti dviem režimais:
    - vartotojo
    - supervizoriaus.
    - Yra taimeris, kas tam tikrą laiko intervalą generuojantis pertraukimus.
    - Įvestis, išvestis
        - Įvedimui naudojamas virtualių „flash atmintinių“ nuskaitymo įrenginys,
        - išvedimui - spausdintuvas.
        - Yra išorinės atminties įrenginys - kietasis diskas.
- Virtualios mašinos atmintis atvaizduojama į vartotojo atmintį naudojant puslapių transliaciją.
- Vartotojas užduočių paketą pateikia „prijungęs“ atmintinę.
    - Sistema perkelia visas joje esančias užduotis į diską, patikrindama jų sintaksę, ir, tuo pat metu, jei tik yra reikiamų resursų, pradeda jas vykdyti.

# 2. Reali mašina

Reali mašina yra kompiuteris. Toliau nagrinėsime realią modelinę mašiną, kuri bus sudaryta tik iš esminių komponentų:

1. Centrinis procesorius
2. Atmintis
   - Vartotojo
   - Supervizoriaus
   - Išorinė (kietasis diskas)
3. Įvedimo įrenginys (virtualių „flash atmintinių“ nuskaitymo įrenginys)
4. Išvedimo įrenginys (spausdintuvas)

## *2.1. Realios mašinos modelis*




*Figūra 1 - Realios mašinos modelio schema*


Lukas Klusis
Justinas Gilvonauskas

## *2.2. Realios mašinos centrinis procesorius*

- **Centrinis procesorius –** skaito komandą iš atminties ir ją vykdo (interpretuoja).

Gali dirbti dviem rėžimais: vartotojo arba supervizoriaus.

- Vartotojo rėžime HLP vykdo tam tikrą užduoties programą. Yra imituojamas virtualios mašinos procesorius ir prieinama prie vartotojo atmintyje esančių programų per puslapiavimo mechanizmą.
- Supervizoriaus rėžime komandos iš supervizorinės atminties yra betarpiškai apdorojamos HLP.

HLP – bet kuris aukšto lygio kalbos procesorius (programavimo kalba).

Procesorius turi šiuos registrus:

- MODE – registras nusakantis darbo rėžimą.
- PTR – 4 baitų puslapių lentelės registras.
- PID – 1 baito proceso ID registras.
- SM – registras rodantis į bendrą atmintį.
- PC – komandų skaitliukas.
- SP – registras saugantis steko viršūnės žodžio indeksą.
- TI – timerio registras.
- PI – programinių pertraukimų registras.
- SI – supervizorinių pertraukimų registras.
- CH1, CH2, CH3 – kanalų registrai. Jų reikšmės 0 – kanalas laisvas, arba 1 – kanalas užimtas.

## *2.3. Realios mašinos atmintys*

- Vartotojo – skirta virtualių mašinų atmintims bei puslapių lentelių laikymui.
- Supervizorinė – trumpai tariant tai atmintis, kurios reikia pačios OS poreikiams (komandos, sisteminiai kintamieji ir pan.). Visa tai valdys HLP.
- Išorinė – šiuo atveju tai bus kietasis diskas. Jame gali būti koks failas.

## *2.4. Taimeris*

Skirtas užduotims suderinti. Yra sakoma, kad užduotis negali trukti ilgiau nei tam tikrą T laiko momentų. Susitarsime, jog išvedimo / įvedimo operacijos reikalauja 3 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, mikrokomanda Test() aptinka taimerio pertraukimą.

TI reikšmę galima nustatyti ar pakeisti supervizoriaus rėžime.

## *2.5. Pertraukimai*

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().

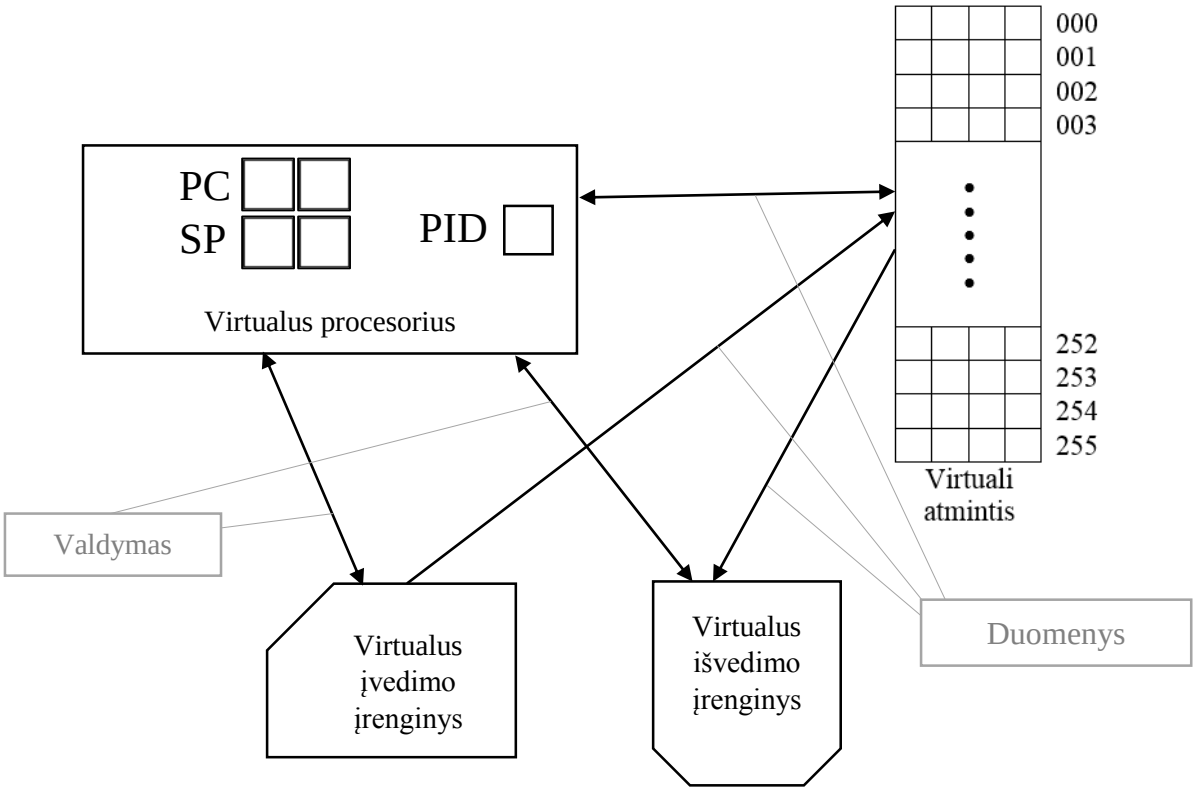
Išskirsime trijų rūšių pertraukimus:

- Programiniai, kurių registras yra PI. Galimi atvejai:
  - PI = 1 – neteisingas adresas.
  - PI = 2 – blogas operacijos kodas.
  - PI = 3 – neteisingas priskyrimas.
  - PI = 4 – perpildymas (overflow)
- Supervizoriniai, kurių registras SI. Galimi atvejai:
  - SI = 1 – komanda PRTS
  - SI = 2 – komanda PRTN
  - SI = 3 – komanda P
  - SI = 4 – komanda READ
  - SI = 5 – komanda STOPF
- Taimerio, kurio registras TI. Galimi atvejai:
  - TI = 0 – taimerio skaitliukas lygus 0.

# 3. 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.

## *3.1. Virtualios mašinos modelis*




*Brėžinys 1 - Virtualios mašinos modelio schema*

## *3.2. Virtualios mašinos centrinis procesorius*

Kaip galima pastebėti iš virtualios mašinos modelio schemos, centrinis virtualus procesorius yra gerokai paprastesnis realios mašinos atvejis. 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 du pagrindinius registrus:

**PC** – 2 baitų komandų skaitiklis (ang. Program Counter)
**SP** – 2 baitų steko rodyklė (ang. Stack Pointer)
**PID –** 1 baito proceso indeksas (process ID) – naudojamas šakojimui

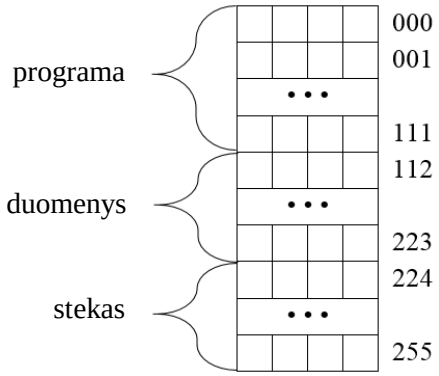
Visų registrų pradinė reikšmė lygi 0.

## *3.3. Virtualios mašinos atmintis*

Kiekvienai virtualiai mašinai yra skiriama 16 vartotojo atminties blokų. Tuose šešiolikoje blokų (256 žodžių po 4 baitus) turi tilpti programos kodas, duomenys ir stekas. 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. Tai bus aptarta skyrelyje „puslapiavimo mechanizmas“.

Kaip jau minėta, virtualios mašinos atmintis bus skirstoma į tris dalis, į kurias bus įkeliamos atitinkamos programos dalys:

1. Programa – 7 blokai, nuo 0 iki 6 (112 žodžiai)
2. Duomenys – 7 blokai, nuo 7 iki 13 (112 žodžiai)
3. Stekas – 2 blokai, nuo 14 iki 15 (32 žodžiai)




*Brėžinys 2 - virtualios mašinos atminties dalys*

## *3.4. Virtualios mašinos procesoriaus komandos*

- Aritmetinės komandos
  - ADD – sudeda du viršutinius steko elementus. Rezultatą padeda į steko viršūnę ir steko rodyklę sumažina vienetu.
    *[SP – 1] = [SP – 1] + [SP]; SP--;*
  - SUB – atima steko viršūnėje esantį elementą iš antro nuo viršaus elemento. Rezultatą padeda į steko viršūnę ir steko rodyklę sumažina vienetu.
    *[SP – 1] = [ SP – 1] - [SP]; SP--;*
  - MUL – sudaugina du viršutinius steko elementus. Rezultatą padeda į steko viršūnę ir steko rodyklę sumažina vienetu.
    *[SP – 1] = [ SP – 1] * [SP]; SP--;*
  - DIV – padalina (gauna sveikąją dalį) 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--;*
- Darbo su bendra atmintimi
  - WRx – į bendrą atminty nurodytu adresu įrašo steko viršūnėje esantį žodį.
  - RDx – iš bendrosios atminties nurodytu adresu nuskaito žodį ir įdeda į steko viršūnę.

- Palyginimo
  - CMP – 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 [SP-1] > [SP]; SP--*
    *[SP+1] = 1, jei [SP-1] == [SP]; SP--*
    *[SP+1] = 2, jei [SP-1] < [SP]; SP--*
- Darbo su duomenimis / steko
  - LDxy – į steko viršūnę užkrauna reikšmę iš duomenų srities adresu 16 * x + y, ($7 \leq x < 14$), SP--;
  - PTxy – steko viršūnėje esantį žodį deda į duomenų sritį nurodytu adresu *SP++;* 16 * x + y, ($7 \leq x < 14$)
  - PUNx – x kaip skaičių patalpina į steko viršūnę.
    *SP++;[SP] = x*
  - PUSx – x kaip simbolį patalpina į steko viršūnę.
    *[SP] = x*
- Valdymo
  - JPxy – nesąlyginio valdymo perdavimo komanda. Valdymas perduodamas kodo sričiai nurodytam adresui.
    *PC = 16*x+y*
  - JExy – jei steko viršūnėje yra 1 valdymas perduodamas adresu 16*x + y.
    *IF( [SP] == 1) PC = 16 * x + y; SP--;*
  - JLxy – jei steko viršūnėje yra 0 valdymas perduodamas adresu 16*x + y.
    *IF( [SP] == 0) PC = 16 * x + y; SP--;*
  - JGxy – jei steko viršūnėje yra 2 valdymas perduodamas adresu 16*x + y.
    *IF( [SP] == 2) PC = 16 * x + y; SP--;*
- Šakojimo komandos
  - FORK - sukuria proceso kopija ir tolimesnį darbą paskiria vaikiniam procesui.
    *PID++; SP++; [SP]=PC;*
  - ISP – palygina PID registro reikšmę su 0, jeigu lygu nuliui, tai į registro viršūnę įstatoma reikšmė 1, jeigu nelygi, įstatoma 0.
    *SP++; IF( PID == 0) THEN [SP] = 1 ELSE [SP] = 0*
  - ISC – palygina PID registro reikšmę su 0, jeigu nelygu nuliui, tai į registro viršūnę įstatoma reikšmė 1, jeigu lygi, įstatoma 0.
    *SP++; IF( PID <> 0) THEN [SP] = 1 ELSE [SP] = 0*
  - STOPF – besąlyginio (priverstinio) programos sustojimo komanda.

- STOP – programos proceso sustojimo komanda. Jeigu PID lygus 0, tai sustabdomas visas programos darbas, jeigu PID nėra lygus nuliui, tai iš steko paimama PC reikšmė ir tęsiamas darbas.
  *IF(PID<>0) THEN {PC=[SP]; SP--;} ELSE STOPF;*

- Įvedimo bei išvedimo komandos
  - PRTS – steko viršūnėje esantį žodį traktuoja kaip simbolius ir išveda į išvedimo įrenginį.
  - PRTN – steko viršūnėje esantį žodį traktuoja kaip skaitinę reikšmę ir išveda į išvedimo įrenginį.
  - Pxyz – į išvedimo įrenginį išveda x numeriu nurodyto atminties srities bloko nuo y iki z žodžius ($y < z$). ($7 \leq x < 14$)
    *Print([16*x+i, i=y..z])*
  - READ – nuskaito vartotojo pateiktą programą ir duomenis ir patalpina į atitinkamus atminties segmentus.

# 4. Puslapiavimo mechanizmas

Realios mašinos vartotojo atmintis siekia 256 takelių (arba blokų). Kiekvienai naujai sukurtai virtualiai mašinai reikia skirti 16 takelių iš tų 256. Jie gali būti parinkti bet kokiu būdu. Klausimas: kaip virtuali mašina gali sužinoti kokio nors jai priklausančio takelio realų adresą? Tam naudosime puslapiavimo mechanizmą.

Puslapiavimo mechanizmo esmė: sakykime, kuriama nauja virtuali mašina. Jai reikia dešimt takelių atminties. Mes parinkome takelius su numeriais: 2, 6, 8, 9, 13, 35, 49, 68, 69, 70, 100, 123, 180, 181, 199, 240. Šiais takeliais naudosis virtuali mašina. Pati virtuali mašina šiuos takelius mato sunumeruotus nuo 0 iki 15, t.y. 2 takelis jai yra nulinis, 6 takelis jai yra antras, o 240 takelis – penkioliktas. Kaip išlaikyti sąryšius tarp realių ir virtualių takelių adresų? Naudosime puslapių lentelę. Puslapių lentelė – tai vienas takelis (t.y. 16 žodžių). Kiekvieno žodžio eilės numeris atitiks virtualios mašinos takelio numerį, ir jame (žodyje) bus laikomas realus to takelio numeris. Pavyzdžiui, prieš tai pateikto pavyzdžio puslapių lentelė bus tokia:

| **Virtualus takelio numeris** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Realus takelio numeris** | 2 | 6 | 8 | 9 | 13 | 35 | 49 | 68 | 69 | 70 | 100 | 123 | 180 | 181 | 199 | 240 |

*Brėžinys 3 - Puslapių lentelės pavyzdys*

PTR yra 4 baitų ir simboliškai žymėsime taip $a_0a_1a_2a_3$. O $x_1x_2$ virtualius adresas.

- $a_0$ – nenaudojamas.
- $a_1$ – nenaudojamas.
- $16*a_2+a_3$ – puslapių lentelės bloko numeris vartotojo atmintyje.
- $16*(16*a_2+a_3)$ – puslapių lentelės bloko adresas.
- $16*(16*a_2+a_3) + x_1$ – bloko $x_1$ adresas puslapių lentelėje. Jame saugomas bloko numeris į kurį atvaizduotas yra $x_1$ blokas VM.
- $16*[16*(16*a_2+a_3) + x_1]$ – VM bloko $x_1$ realus bloko adresas.
- $16*[16*(16*a_2+a_3) + x_1] + x_2$ – realus adresas atitinkantis virtualų adresą $x_1x_2$.


Lukas Klusis
Justinas Gilvonauskas

# 5. Užduoties pateikimo formatas ir taisyklės

Užduotį įsivaizduosime kaip atminties žodžių paketą sudarytą iš programos ir duomenų blokų. Paketai pateikiami tokia tvarka

<paketas CODE>.<programa>.<paketas DATA>.<duomenys>.<paketas STOP>

<paketas CODE> sudarytas iš keturių laukų:

- $AMJ – vienas žodis
- Užduoties vardas – vienas žodis
- Laikas – vienas žodis
- Išvedimo eilučių skaičius – vienas žodis

Paketas <programa> Iki 108 žodžių užimantis programos kodas

<paketas DATA> turi vieną žodį $DTA, kuris reiškia, kad toliau seks duomenų blokas.

Pakete <duomenys> yra saugoma informacija, kurios dydis turi būti iki 109 žodžių

<paketas STOP> yra žodis $STO ir laukas, užduoties vardui, kuris yra identiškas <paketas CODE> užduoties vardui.