Міністерство освіти і науки України
Ужгородський національний університет
Математичний факультет
Кафедра системного аналізу і теорії оптимізації

ОБЧИСЛЮВАЛЬНІ СИСТЕМИ

Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності "прикладна математика"

Ужгород – 2008

Чупов С.В., Брила А.Ю. Обчислювальні системи**.** Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності "прикладна математика". – Ужгород, 2008. – 31 с.

Розглядаються базові поняття "система", "інформаційна система", "обчислювальна система", структура, принципи роботи та властивості окремих елементів обчислювальної системи (обчислювальної машини), можливості програмного моделювання абстрактної обчислювальної системи на базі мови високого рівня.

Рецензенти: канд. фіз.-мат. наук, доц. Гече Ф.Е.,
канд. фіз.-мат. наук, доц. Мич І.А.

Рекомендовано до друку Вченою радою математичного факультету Ужгородського національного університету 26 лютого 2008 року, протокол №6.

І. Поняття обчислювальної системи

1.1. Поняття інформаційної та обчислювальної систем

Важливим поняттям сучасної науки є поняття системи. Під *системою* будемо розуміти сукупність взаємопов'язаних елементів, які утворюють єдине ціле і призначені (слугують) для досягнення єдиної мети.

Отже, поняття системи базується на трьох положеннях:

а) маємо множину взаємозв'язаних елементів;

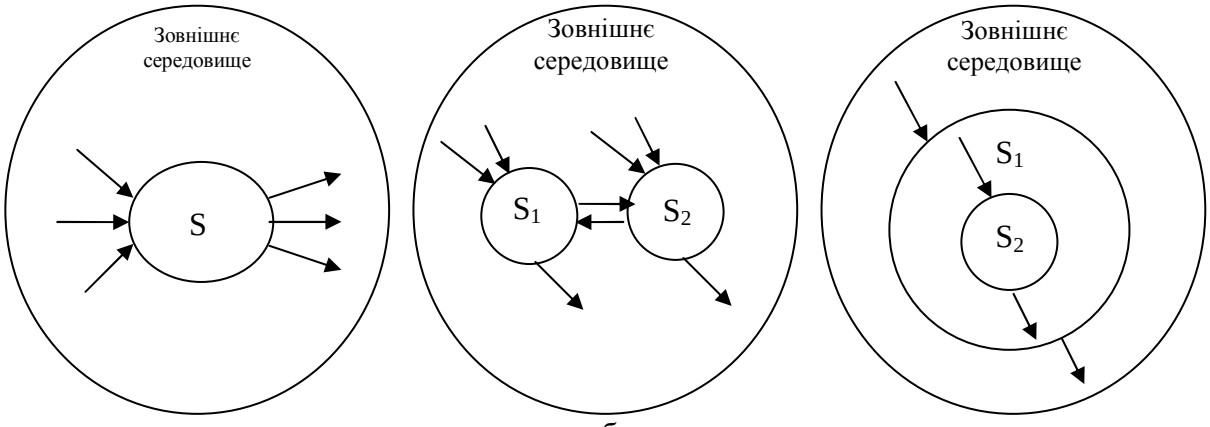
б) множина елементів утворює єдине ціле, тобто вилучення якого-небудь елемента порушить цілісність системи;

в) єдине ціле має певну мету, що характерна для усієї сукупності елементів системи, а не для якоїсь комбінації з них.

Система, як правило, характеризуються сукупністю властивостей, які утворюють *стан системи*.

Довільна система існує в певному середовищі, що її оточує. Дія середовища на систему проявляється за допомогою певних факторів, які особливим чином впливають на внутрішній стан системи. Дії факторів середовища на систему мають назву *вхідних дій*, а елементи до яких ці дії прикладені – *входами системи* по відношенню до середовища. У свою чергу система не є нейтральною по відношення до середовища. Її вплив на середовище характеризується значеннями *вихідних величин*.

Дослідження систем зручно здійснювати за допомогою розбиття (декомпозиції) системи на підсистеми, які в свою чергу є системами більш низького рівня. Таким чином, декомпозиція визначає *ієрархію систем*. Схематично це можна побачити на рис. 1.



а б в

Рис 1.

де на рис. 1.а S – система; на рис. 1.б S_1 і S_2 – системи одного рівня; на рис. 1.в S_2 є підсистемою системи S_1 .

При вивченні довільної системи необхідно з'ясувати найважливіші її характеристики: функцію, мету та структуру.

Під *функцією* розуміють дії системи, що виражаються зміною її можливих станів. Виконання системою своїх функцій називається *функціонуванням системи*. Функціонування системи – це динамічний процес, що полягає у переході системи з одного стану в один із можливих інших. Якщо позначити через P множину можливих станів системи, через F – функцію системи, а p_1 та p_2 – деякі стани системи, то можливе виконання співвідношення $p_2 = F(p_1)$. Множина усіх можливих станів P системи визначається кількістю її елементів, їх властивостями та різноманітністю зв'язків між ними. Тому функція системи характеризує її як єдине ціле, як результат взаємодії її елементів між собою та вхідними діями.

Метою системи називають бажаний (той, що задається ззовні або визначається самою системою) стан її виходів, тобто деяке значення або підмножину значень функції системи. Таким

чином, будь-яку систему слід вивчати із динамічної точки зору, тобто вивчати можливі послідовності зміни станів системи. При цьому вважається, що будь-яка система намагається перейти у так званий стан "*спокою*" за допомогою своєї функції, а зовнішні впливи і, можливо, внутрішні впливи виводять систему з цього стану. Схематично це зображено на рис. 2.

Рис. 2

де p_0 – початковий стан системи; p_n – стан системи, який характеризує мету системи.

Структура системи визначається розміщенням та взаємозв'язками елементів системи при виконанні останньою своїх функцій і залежить від *величини* та *складності*.

Величина системи характеризується кількістю її елементів та зв'язків, а *складність* – різноманітністю, неоднорідністю елементів системи та різною якістю зв'язків (прямі, обернені, нейтральні).

Великим та складним системам притаманні властивості *цілісності* та *емерджементності*. Цілісність означає, що усі її частини служать спільній цілі та сприяють формуванню найкращих результатів стосовно визначеного критерію ефективності. Емерджементність виражається в тому, що великі та складні системи мають властивості, яких немає в жодного з її елементів. Тобто встановити властивості системи неможливо, якщо вивчати тільки її окремі частини.

Система, в якій одним із зв'язків є інформаційний, називається *інформаційною системою*. Інформаційний зв'язок здійснює передачу даних або інформації між елементами цієї системи.

Схематично це виглядає так

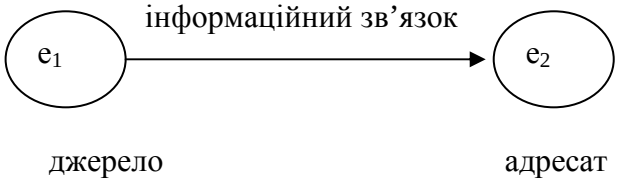


Рис. 3

У загальному довільна інформаційна система містить наступні складові: блок фіксації даних, блок обробки даних, блок аналізу – елементи системи, та два типи зв'язків (прямі та обернені). Схематично це можна зобразити так:

Рис. 4

Серед інформаційних систем можна виділити спеціальний тип, який будемо називати *обчислювальними системами*. *Обчислювальна система* – це сукупність апаратних і програмних засобів, що забезпечують автоматизацію збору, накопичення, опрацювання, систематизації, зберігання, подання, передачі інформації.

Оскільки програмні засоби можна поділити на системні, які будемо об'єднувати терміном операційної системи, та прикладні (або прикладні програми), то обчислювальна система містить три основні елементи: апаратна частина (АЧ), операційна система (ОС) та прикладні програми (ПП). Схематично це зображено на рис. 5.

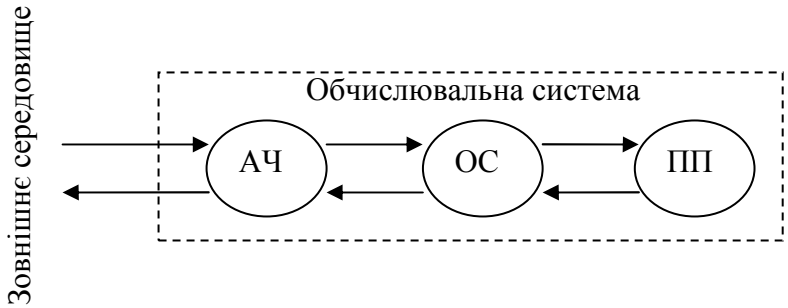


Рис. 5

Згадаємо, що мова програмування – це система позначень і правил, призначена для опису алгоритмів і структур даних. Оскільки з будь-якого обчислювальною системою зв'язана своя мова, яка призначена для створення інтерфейсу між обчислювальною системою та зовнішнім середовищем або користувачем, то можна дати інше означення обчислювальної системи. *Обчислювальна система (машина)* – це інтегрований набір алгоритмів і структур даних, що здатен зберігати та виконувати програми. Цей набір може бути реалізований в апаратній або програмній частинах обчислювальної машини (системи). В першому випадку таку машину будемо називати *апаратною обчислювальною машиною*, в другому випадку – *програмно-модельованою обчислювальною машиною*. Слід зазначити, що будь-яка сучасна обчислювальна система розроблена як програмно-модельована обчислювальна машина, для якої зовнішнім середовищем є апаратна обчислювальна машина.

1.2. Структурна схема обчислювальної машини

Структурну схему обчислювальної машини наведено на рис. 6.

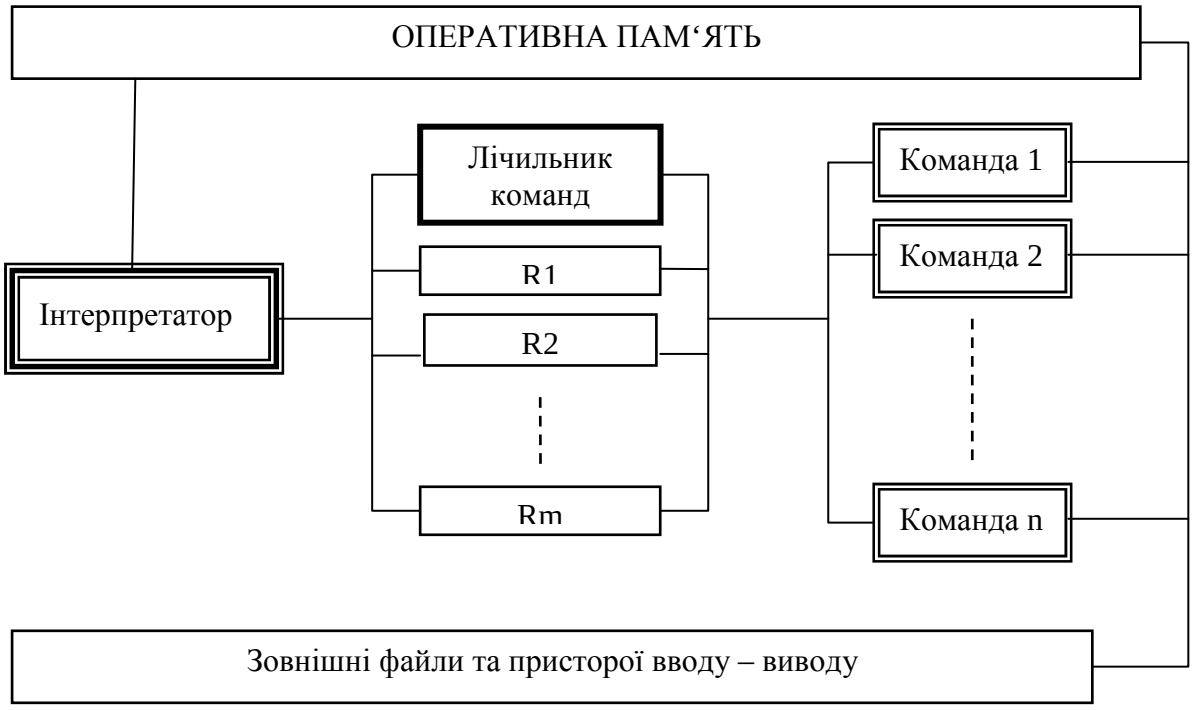


Рис. 6

1.3. Принцип програмного керування роботою обчислювальної машини

У кінці 40-х років американський математик Джон фон Нейман описав абстрактну обчислювальну машину, за допомогою якої можна було автоматично здійснювати керування процесом виконання програми. Ця машина складається з чотирьох основних елементів:

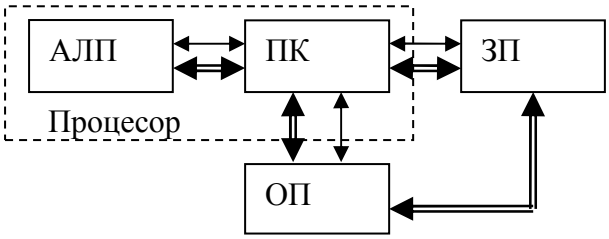
1) пристрою керування (ПК), який організовує покомандне виконання програми та керує ходом її виконання;

2) арифметико-логічного пристрою (АЛП), який призначений для виконання арифметичних та логічних команд;

3) оперативної пам’яті (ОП), де знаходиться програма і дані під час виконання команд програми;

4) зовнішньої пам’яті та пристроїв вводу/виводу (ЗП).

Між цими елементами встановлено два типи зв’язку: керуюча лінія (→) та лінія зв’язку (=>). Схематично абстрактну обчислювальну машину Джона фон Неймана можна представити наступним чином



Описання роботи такої машини назвали принципом програмного керування. Схематично роботу обчислювальної машини зображено на рис. 7.

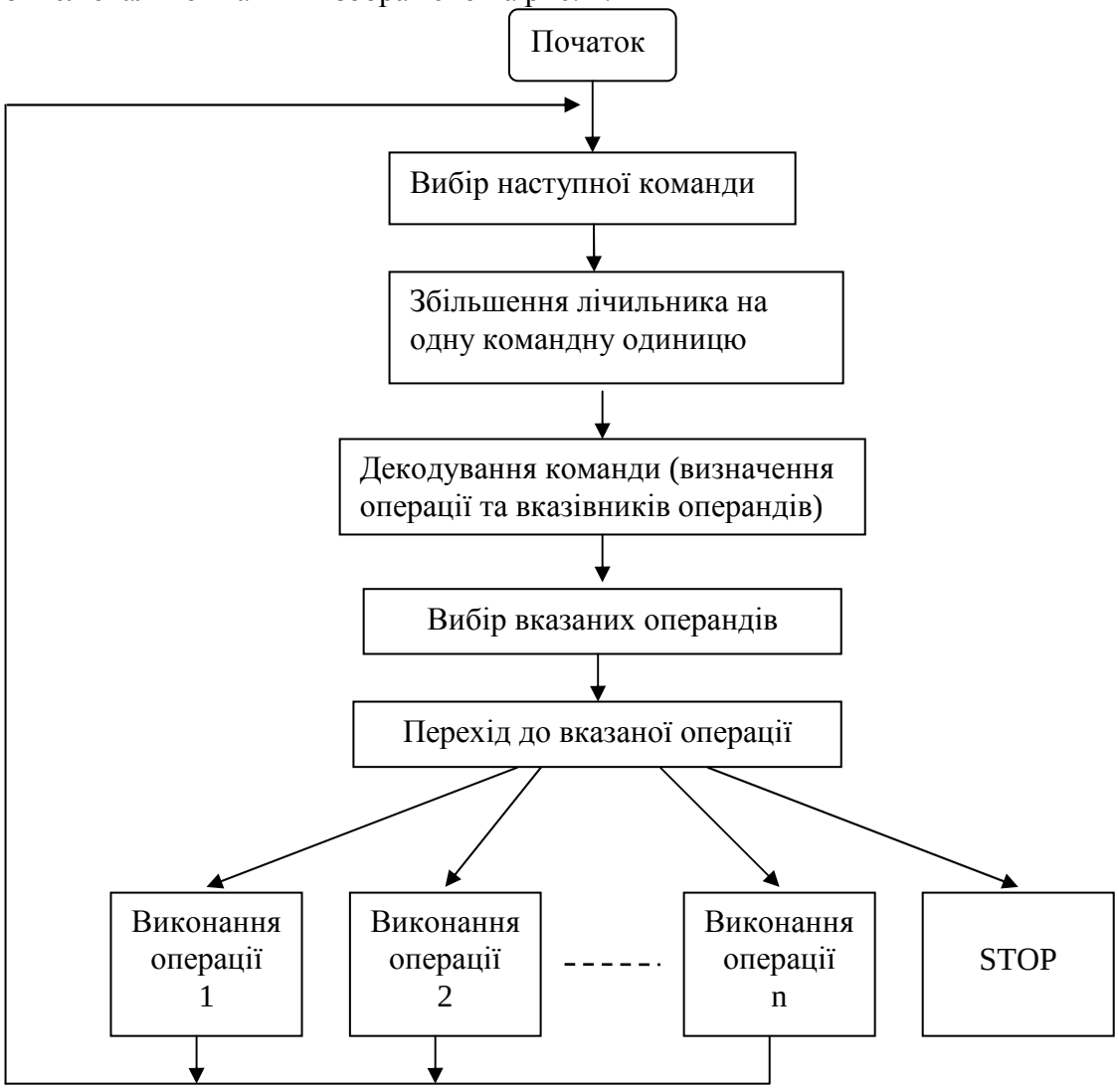


Рис. 7

1.4. Типи обчислювальних машин згідно формату машинних команд

Раніше зазначалося, що під час виконання програми обчислювальною машиною команди та дані розташовуються в оперативній пам'яті. Тому програму слід розглядати як окремий тип даних, які при виконанні програми розміщуються в оперативній пам'яті. Таким чином, кожна команда мусить мати своє внутрішнє двійкове представлення в оперативній пам'яті. Команда несе в собі наступну інформацію: яку операцію має виконати машина, над якими об'єктами, куди записати отриманий результат. Крім того, команда може містити також й інформацію про те, яку слід вибрати команду для виконання в якості наступної. Машинне слово, що містить команду розбивається на групи розрядів – поля, які використовуються для задавання інформації певного призначення. Одна група розрядів відводиться під код операції або номер операції. Інша група розрядів – адресне поле – відводиться під адреси операндів, що приймають участь в операції, під адресу результату і, можливо, під адресу наступної по порядку команди. Є команди, де в адресному полі вказується безпосереднє значення одного з операндів, а не адреса операнда, як це буває зазвичай. В загальному машинне слово, що місить команду можна зобразити так:

Поле операції	Поле адрес

Дешифруючи команду, пристрій керування визначає функціональне призначення окремих її розрядів, розглядаючи певні сукупності розрядів як поле операції або поле адрес. Наприклад, для виконання арифметичної дії у випадку, коли початкові дані містяться в пам'яті машини і результат потрібно помістити в пам'ять, а адреса наступної команди для виконання теж записана в команді, команда може містити код операції та чотири адреси: дві для операндів, одну для вказівки адреси результату і одну для визначення місцезнаходження в пам'яті наступної по порядку команди.

Машину, в якій команда має розглянуту вище структуру, називають *чотирьохадресною обчислювальною машиною*. Команда для неї має вигляд:

$$qA_1A_2A_3A_4,$$

тут q – код операції, A_1 – адреса першого операнда, A_2 – адреса другого операнда, A_3 –адреса результата, A_4 – адреса наступної команди. Наприклад, якщо всі дані і результати знаходяться в пам'яті (код команди займає 1 байт, а адреса 2 байти), то вся машинна команда займає 9 байтів. Якщо з 500-ї комірки будуть розташовані значення

	500	501	502	503	504	505	506	507	508	
…….	02	00	01	00	02	80	01	29	50	……
	q	A_1		A_2		A_3		A_4		

то команда буде розшифрована так: (код операції $q=02$) додати ($A_1=100;\ A_2=200$) значення 100-ї і 200-ї комірок і результат ($A_3=180$) записати у 180 комірку, ($A_4=529$) наступною буде виконуватись команда, що розташована з 529 комірки: $[180]:=[100]+[200]$.

Було виявлено, що в більшості програм машинні команди, як правило, ідуть послідовно одна за одною, тобто розташовані лінійно. Тому була запропонована трьохадресна машинна команда, для якої за замовчуванням вважається, що команда, яка має виконуватись наступною, іде одразу за тою, що виконується. Формат такої команди такий $qA_1A_2A_3$. Для того, щоб можна було змінювати природну лінійну послідовність виконання команд, в систему команд машини було включено команди умовних та безумовних переходів.

В багатьох випадках, після виконання команди результат зберігається або в одному із внутрішніх регістрів, або за адресою одного з операндів команди. Виходячи з цього, було запропоновано двохадресні машинні команди з форматом: qA_1A_2.

Наприклад, якщо визначити, що результат завжди має зберігатися за адресою 1-го операнда і 100-та комірка містить значення 5, 200-та містить значення 6

	100		200	
…….	5	…….	6	…….
	x		y	

то після виконання команди $qA_1A_2=2,100,200$ 100-та комірка буде містити значення 11 або (B)

$x := x + y$

	100		200	
…….	B	…….	6	…….
	x		*y*	

Нарешті, одноадресна каманда має формат qA_1. Поле адрес для неї розшифровується так: A_1 – адреса першого операнда, другий операнд завжди повинен розміщуватися у внутрішньому регістрі, який називають регістр-акумулятором. Після виконання команди результат записується також у внутрішній регістр-акумулятор.

У залежності від того, який формат команд використовується в обчислювальній машині, вони поділяються на чотири “чистих” типи обчислювальних машин: чотирьохадресні обчислювальні машини, трьохадресні, двохадресні та одноадресні обчислювальні машини.

II. Абстрактна одноадресна обчислювальна машина

2.1. Структурна схема абстрактної одноадресної обчислювальної машини

Абстрактна обчислювальна машина містить такі складові:

1. Інтерпретатор (***I***) – пристрій, що виконує команди машини.
2. Оперативна пам'ять (***M***), яка складається з трьох областей: області команд, де розташовується програма при виконанні, області даних, де містяться початкові, проміжні та результуючі дані при виконанні програми та область даних для стеку. Стековий сегмент поділяється на дві частини: стек даних (*DataStack*) та стек команд, в якому будуть зберігатися адреси точок повернення з підпрограм.
3. Внутрішні регістри обчислювальної машини:

a) ***IC*** – лічильник команд програми;

b) ***RA*** – регістр акумулятор – основний регістр машини, в який завжди записується значення першого операнда команди та в якому зберігається результат виконання команди;

c) ***RI*** – індексний регістр – допоміжний регістр для організації циклічної обробки даних;

d) ***SP*** – регістр стеку – завжди зберігає значення вершини стеку;

e) ***RF*** – регістр прапорців – містить відомості про стан виконання останньої команди. Він складається з трьох основних однорозрядних полів: *RFZ, RFL, RFG.*

RFZ – прапорець нуля. Завжди встановлюється в ***1***, якщо результат операції дорівнює *0* або при порівнянні двох значень вони виявились рівні.

RFL – прапорець менше. Встановлюється в ***1***, якщо результат операції менше за нуль або при порівнянні двох значень перше виявилось меншим за друге.

RFG – прапорець більше. Встановлюється в ***1***, якщо результат операції більше за нуль або при порівнянні двох значень перше виявилось більшим за друге.

4.Стандартні пристрої вводу/виводу даних або клавіатура та екран. Для роботи з цими пристроями у системі команд обчислювальної машини передбачено чотири команди: *RDRA* – ввід з клавіатури значення в регістр *RA, RDRI* – ввід з клавіатури значення в регістр *RI, WRRA* – вивід на екран значення з регістра *RA, WRRI* – вивід на екран значення з регістра *RI.*

5.Набір внутрішніх команд обчислювальної машини.

Схематично, структурна схема абстрактної обчислювальної машини зображена на рис 8.

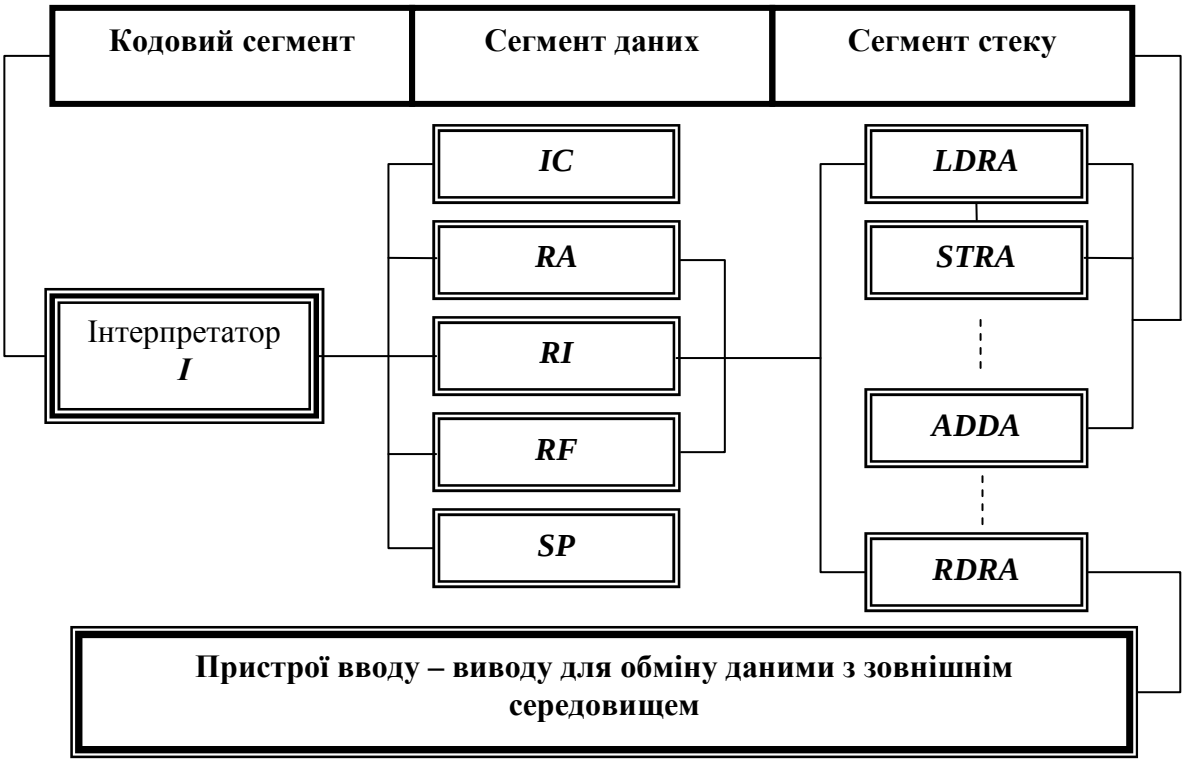


Рис. 8

2.2. Структура основної пам'яті та внутрішніх регістрів абстрактної одноадресної обчислювальної машини

Основна пам'ять складається з трьох областей: області даних (сегмент даних), області команд (сегмент коду) та області стеку (стековий сегмент). Кожна з цих областей має лінійну структуру, тобто представляє собою звичайний одновимірний масив. Сегмент даних – це масив дійсних чисел розмірності *DataCount*. Сегмент стеку – це також масив дійсних чисел розмірності *StackCount*. Кодовий сегмент – це також лінійний однорідний масив елементів розмірності *CodeCount*. Кожен елемент цього масиву представляє собою запис з двома полями. Перше поле містить код операції, а друге – вказівку операнда для виконання даної операції. Внутрішні регістри *RA* та *RI* мають структуру комірки області даних, тобто описуються як змінні типу *Real*. Регістр прапорців *RF* це запис, поля якого є значеннями логічного типу даних. Поля регістру *RF* (*RF1, RF2, RF3*) зарезервовані для подальшого використання. Оскільки регістри *IC* та *SP* містять номери комірок відповідних областей пам'яті, то вони описуються як змінні типу *Integer*. Відповідні описання на мові Turbo Pascal мають вигляд:

```
Const
 DataCount = 199; // Об'єм області даних: 200 комірок
 CodeCount  = 99; // Об'єм області команд: 100 комірок
 StackCount = 99; // Об'єм області стеку: 100 комірок

Type
 DataSeg  = array[0..DataCount] of Real;
 StackSeg = array[0..StackCount] of Real;

 Flags = Record
  Rfz:Boolean;
  Rfl:Boolean;
  Rfg:Boolean;
  Rf1, Rf2, Rf3:Boolean;
 End;

 Command = Record
      Cop:Byte; {код операції}
      Opr:Real; {операнд}
 End;
 CodeSeg = array[0..CodeCount] of Command;
Var
 Code:CodeSeg;
 Data:DataSeg;
 RA, RI: Real;
 IC,SP: Integer;
 RF: Flags;
```

Схематично структуру основної пам'яті абстрактної обчислювальної машини зображено на рис. 9. Наприклад, через q_2 позначено код команди, що розташована в 1-й комірці, або Code[1].Cop. Через A_2 позначений вказівник операнда для команди, яка записана в 1-у комірку кодового сегмента або Code[1].Opr. Через D_{101} позначено вміст 100-ї комірки області даних, або Data[100]. Тут через *CC* та *DC* позначено значення *CodeCount* та *DataCount* відповідно. Таким чином, команда абстрактної обчислювальної машини *LDRA 100* еквівалентна оператору мови Pascal *RA:=Data[101];*, *STRI 22 – Data[23]:=RI;*, *ADDA 50 – RA:=RA+Data[51];*. Описання набору машинних команд обчислювальної машини подано в наступному пункті.

Структура кодового сегменту

	коп	Операнд
0	q_1	A_1
1	q_2	A_2
2	q_3	A_3
…	…	…
…	…	…
50	q_{51}	A_{51}
…	…	…
…	…	…
CC		

Структура сегменту даних

	Дані
0	D_1
1	D_2
2	D_3
…	…
100	D_{101}
…	…
DC	

Рис 9.

2.3. Система команд одноадресної абстрактної обчислювальної машини

Усі команди можна поділити на п’ять груп:

- команди управління даними;
- команди маніпулювання даними;
- команди управління послідовністю дій;
- команди роботи зі стеком;
- команди роботи з підпрограмами.

Позначення, що використовуються в описаннях:

M – адреса комірки пам’яті обчислювальної машини;

N – числовий операнд;

[M] – вміст комірки оперативної пам’яті за адресою M;

CON – стандартний пристрій вводу – виводу (клавіатура та екран).

2.3.1. Команди управління даними

Таблиця 1

Код	Синтаксис	Зміст	
0	STOP	Завершення виконання програми	
1	LDRA M	Завантажити в RA значення комірки пам’яті, адреса якої задається в полі аргументу	**RA:= [M]**
2	LDRI M	Завантажити в RI значення комірки пам’яті, адреса якої задається в полі аргументу	**RI:= [M]**
3	LDAI M	Завантажити у регістр RA значення комірки пам’яті, адреса якої обчислюється як сума операнда M та регістру RI	**RA:= [M+RI]**
6	STRA M	З RA записати значення в комірку пам’яті, адреса якої задається аргументом M	**[M]:= RA**
7	STRI M	З RI записати значення в комірку пам’яті, адреса якої задається аргументом M	**[M]:= RI**
8	STAI M	В комірку пам’яті, адреса якої є сума M+RI записати значення RA	**[M+RI]:= RA**
11	ENTA N	Завантажити в регістр RA безпосереднє значення N	**RA:= N**
12	ENTI N	Завантажити в регістр RI безпосереднє значення N	**RI:= N**
13	WRRA	Значення регістру RA вивести на екран	**RA->CON (write(RA);)**
14	WRRI	Значення регістру RI вивести на екран	**RI->CON (write(RI);)**

Код	Синтаксис	Зміст	
15	RDRA	Завантажити значення регістру RA з інформаційної області (клавіатури)	**CON->RA (read(RA);)**
16	RDRI	Завантажити значення регістру RI з інформаційної області (клавіатури)	**CON->RI (read(RI);)**

2.3.2. Команди маніпулювання даними

Таблиця 2

Код	Синтаксис	Зміст	
50	ADDA M	До регістру RA додати значення, що зберігається в комірці пам'яті, адреса якої міститься в полі аргументу	**RA:=RA+[M]**
51	ADDI M	До регістру RI додати значення, що зберігається в комірці пам'яті, адреса якої міститься в полі аргументу	**RI:= RI + [M]**
52	ADAI M	До регістру RA додати значення комірки пам'яті, адреса якої обчислюється як сума значення, яке міститься в аргументі і регістрі RI	**RA:=RA+[RI+M]**
53	SUBA M	Від регістру RA відняти значення комірки пам'яті, адреса якої міститься у полі аргументу	**RA:=RA-[M]**
54	SUBI M	Від регістру RI відняти значення комірки пам'яті, адреса якої міститься у полі аргументу	**RI:= RI - [M]**
55	SBAI M	Від регістру RA відняти значення комірки пам'яті, адреса якої обчислюється як сума значення, яке міститься в аргументі і значення регістру RI	**RA:=RA-[RI+M]**
56	INCA N	До RA додати безпосереднє значення, яке міститься в полі аргументу	**RA:=RA + N**
57	INCI N	До RI додати безпосереднє значення, яке міститься в полі аргументу	**RI:=RI + N**
58	DECA N	Від RA відняти безпосереднє значення, яке міститься в полі аргументу	**RA:=RA – N**
59	DECI N	Від RI відняти безпосереднє значення, яке міститься в полі аргументу	**RI:=RI – N**
60	MULA M	RA помножити на значення комірки пам'яті, адреса якої міститься в аргументі	**RA:=RA*[M]**
61	MULI M	RI помножити на значення комірки пам'яті, адреса якої міститься в аргументі	**RI:=RI*[M]**
62	MLAI M	Значення регістру RA обчислюється як добуток RA і значення комірки пам'яті, адреса якої обчислюється як сума значення, яке міститься в аргументі і значення регістру RI	**RA:=RA*[RI+M]**
63	MLAN N	RA помножити на безпосереднє значення, яке міститься в полі аргументу	**RA:=RA*N**
64	MLIN N	RI помножити на безпосереднє значення, яке міститься в полі аргументу	**RI:=RI * N**
65	DIVA M	RA поділити на значення комірки пам'яті, адреса якої міститься в полі аргументу	**RA:=RA / [M]**
66	DIVI M	RI поділити на значення комірки пам'яті, адреса якої міститься в полі аргументу	**RI:=RI / [M]**
67	DVAI M	Значення регістру RA обчислюється як частка значення регістру RA і значення комірки пам'яті, адреса якої обчислюється як сума значення, яке міститься в аргументі і значення регістра RI	**RA:=RA / [RI+M]**
68	DVAN N	RA поділити на безпосереднє значення, яке міститься в полі аргументу	**RA:=RA / N**

Код	Синтаксис	Зміст	
69	DVIN N	RI поділити на безпосереднє значення, яке міститься в полі аргументу	**RI:=RI / N**
70	MODM M	RA – остача від ділення RA на значення комірки пам’яті, адреса якої міститься в полі аргументу	**RA:=RA mod [M]**
71	MODN N	RA – остача від ділення RA на безпосереднє значення, яке міститься в полі аргументу	**RA:=RA mod N**
72	IDVM M	RA поділити на значення комірки пам’яті, адреса якої міститься в полі аргументу	**RA:=RA div [M]**
73	IDVN N	RA поділити на безпосереднє значення, яке міститься в полі аргументу	**RA:=RA div N**

2.3.3. Команди управління послідовністю дій

Таблиця 3

Код	Синтаксис	Зміст	
20	GOTO M	Безумовний перехід до команди в комірці M кодового сегменту.	
21	CMPA M	Порівняння RA (x) із значенням комірки пам’яті, адреса якої міститься в полі аргументу M (y). Команди порівняння встановлюють прапорці у відповідному регістрі: RFZ= true, якщо операнди рівні (x=y); RFL= true, якщо перший операнд менший за другий (x<y); RFG= true, якщо перший операнд більший за другий (x>y).	
28	CMPI M	Порівняння RI (x) із значенням [M] (y)	
29	CMAI M	Порівняння RA (x) із значенням [M +RI] (y[i])	
40	CMAN N	Порівняння RA (x) із безпосереднім значенням N	
41	CMIN N	Порівняння RI (x) із безпосереднім значенням N	
22	JMEQ M	Перейти до команди в комірці M, якщо RFZ= true	**x = y**
23	JMNE M	Перейти до команди в комірці M, якщо RFZ= false	**x ≠ y**
24	JMGT M	Перейти до команди в комірці M, якщо RFG=true	**x > y**
25	JMGE M	Перейти до команди в комірці M, якщо RFZ= true або RFG=true	**x ≥ y**
26	JMLT M	Перейти до команди в комірці M, якщо RFL= true	**x < y**
27	JMLE M	Перейти до команди в комірці M, якщо RFZ= true або RFL= true	**x ≤ y**
74	LOOP M	Циклічна команда. Зменшити значення RI на 1 і, якщо воно не менше 0, перейти до команди в комірці M	

2.3.4. Команди роботи зі стеком

Таблиця 4

Код	Синтаксис	Зміст	
32	PUSH M	Занести в стек значення M-ої комірки	**[M]->DataStack**
33	PUSHA	Занести в стек значення регістру RA	**RA->DataStack**
34	PUSHI	Занести в стек значення регістру RI	**RI->DataStack**
35	POPM M	Взяти значення зі стеку та зберегти в комірці M	**DataStack ->[M]**
36	POPA	Взяти значення зі стеку та зберегти в RA	**DataStack ->RA**
37	POPI	Взяти значення зі стеку та зберегти в RI	**DataStack ->RI**

2.3.5. Команди роботи з підпрограмами

Таблиця 5

Код	Синтаксис	Зміст
30	CALL M	Виконати процедуру, яка розташована в пам’яті, починаючи з комірки M
31	RETM	Повернутися в головну програму

2.3.6. Розширення системи команд абстрактної обчислювальної машини

Оскільки абстрактна ОМ моделюється програмно, то ніщо не заважає розширювати систему її команд, тобто вводити у систему команд нові команди у форматі одноадресної системи команд. Наприклад, розглянемо команди, що реалізують деякі стандартні числові функції.

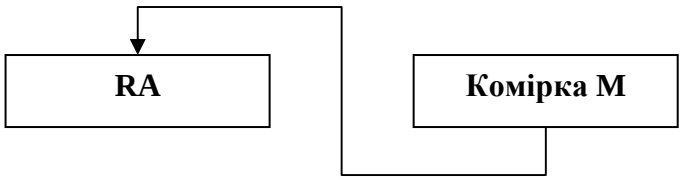
Код	Синтаксис	Зміст	
80	SINA	Знайти значення функції sin(*x*)	**RA:=SIN(RA)**
81	COSA	Знайти значення функції cos(*x*)	**RA:=COS(RA)**
82	EXPA	Знайти значення функції exp(*x*)	**RA:=EXP(RA)**
83	POWM M	Знайти значення *x^y*. Основа *x* міститься в RA, степінь *y* в комірці пам'яті за адресою M	**RA:= RA^[M]**
84	POWN N	Знайти значення *x^y*. Основа x міститься в RA, степінь *y* задається безпосередньо	**RA:= RA^N**

2.4. Описання найбільш вживаних команд

Надалі для ілюстрації аналогічних дій на мові Pascal, припустимо, що регістр RA відповідає змінній x, а комірка з номером M відповідає змінній y, регістр RI відповідає змінній j.

LDRA M

Дана команда записує в регістр *RA* значення, яке міститься в комірці пам'яті, адреса якої *M*. На мові Pascal ця дія еквівалентна оператору присвоєння *x:=y;*. Схематично дія команди *LDRA* зображена на рисунку.

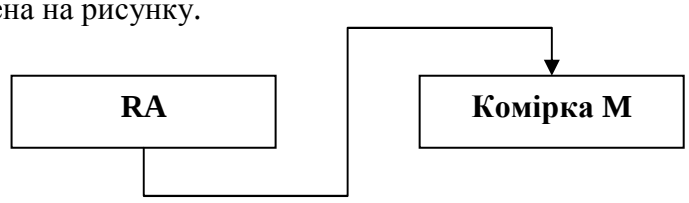


Команди, які виконуються аналогічно до команди *LDRA M*:

LDRI M

STRA M

Дана команда записує значення, яке міститься в регістрі *RA* в комірку пам'яті, адреса якої *M*. На мові Pascal ця дія еквівалентна оператору присвоєння *y:=x;*. Схематично дія команди *STRA* зображена на рисунку.



Команди які виконуються аналогічно до команди *STRA M*:

STRI M

ENTA N

Дана команда записує в регістр *RA* безпосереднє значення (числову константу) *N*. Якщо припустити, що значення *N* дорівнює *3.14*, то на мові Pascal це буде виглядати як *x:=3.14;*

Команди, які виконуються аналогічно до команди *STRA M*:

ENTI M.

INCA N

Дана команда збільшує значення регістра *RA* на безпосереднє значення *N*. Наприклад, якщо *N=1*, то оператор *x:=x+1;* буде еквівалентний збільшенню значення регістра *RA* на *1*.

Команди, які виконуються аналогічно до команди *INCA N*:

INCI N, DVAN N, MLAN N, DVIN N, MLIN N, IDVN N, MODN N.

RDRA

Дана команда вводить з клавіатури в регістр *RA* числове значення. Ця дія еквівалентна команді *readln(x);* на мові Pascal.

Команди, які виконуються аналогічно до команди *RDRA*:

RDRI.

WRRA

Дана команда виводить на екран значення регістра *RA*. Ця дія еквівалентна команді *writeln('RA=',x:8:3);* на мові Pascal.

Команди, які виконуються аналогічно до команди *WRRA*:

WRRI.

ADDA M

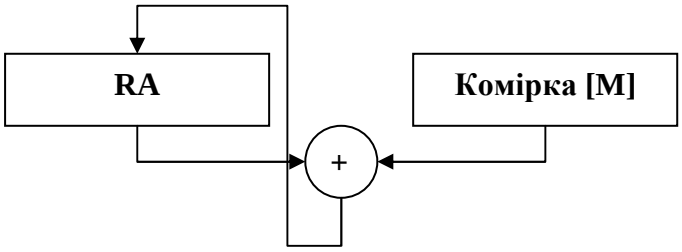
Дана команда сумує вміст регістра *RA* з вмістом комірки області даних з адресою (номером) *M* знову до регістру *RA*. В програмі на мові Pascal ця команда буде записана як *x:=x+y;*. Якщо припустити, що змінна *a* розміщена в *1*-ій комірці, *b* – в *2*-ій, а *c* – в *3*-ій, то для реалізації команди мови Pascal *a:=b+c;* треба виконати таку послідовність команд абстрактної обчислювальної машини:

LDRA 2 // значення *2*-ї комірки (*b*) записуємо в *RA*

ADDA 3 // додаємо значення *3*-ї комірки (*c*) *RA*

STRA 1 // результат з *RA* записуємо в *3*-ю комірку (*a*)

Схематично дія команди *ADDA* зображена на рисунку.



Команди, які виконуються аналогічно до команди *ADDA M*:

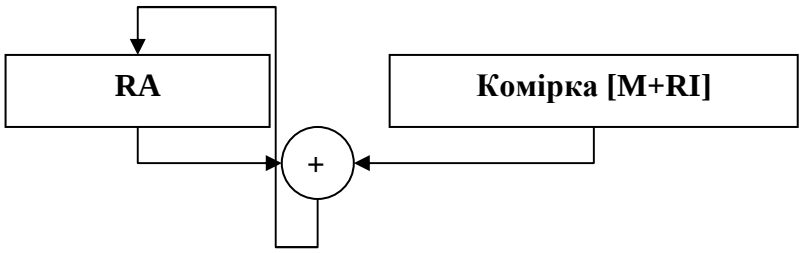
ADDI M, SUBA M, SUBI M, MULA M, MULI M, DIVA M, DIVI M, MOD M, IDVM M

ADAI M

При виконанні даної команди спочатку обчислюється адреса комірки, з якої буде вибиратися значення як сума регістра *RI* та операнда команди *M*. Після цього значення цієї комірки додається до значення регістру *RA*. Дана команда еквівалентна командам мови Pascal для обробки елементів одновимірних масивів. Тут регістр *RI* виступає в ролі лічильника номерів елементів масиву. Слід відзначити, що нумерація елементів масиву починається з *0*, тобто перший елемент має номер *0*, другий – *1* і т.д. В програмі на мові Pascal ця команда буде записана як *x:=x+y[j];*. Наприклад, якщо потрібно знайти середнє арифметичне 3-го та 7-го елементів масиву, то на мові Pascal це буде оператор *x:=(y[3]+y[7]) / 2;*. Якщо припустити, що масив *y* розміщується починаючи з *10*-ї комірки, то для реалізації аналогічних дій в системі команд абстрактної машини треба виконати наступні команди:

ENTI 2 // заносимо в *RI* номер 3-го елемента
LDAI 10 // записуємо в регістр *RA* значення цього елемента
ENTI 6 // заносимо в *RI* номер 7-го елемента
ADAI 10 // додаємо до 3-го елемента 7-ий
DVAN 2 // ділимо суму, яка знаходиться в *RA* на 2 (результат залишається в *RA*)

Схематично дія команди *ADAI* зображена на рисунку.



Команди, які виконуються аналогічно до команди *ADAI M*:
LDAI M, STAI M, SBAI M, MLAI M, DVAI M.

CMPA M

Дана команда порівнює значення регістра *RA* зі значенням, що міститься в комірці області даних, адреса якої *M*. Після порівняння встановлюються відповідні прапорці регістру *RF*. Якщо *RA* дорівнює значенню комірки області даних, адреса якої *M*, то прапорець *RFZ* прийме значення *TRUE*, якщо *RA* більше значення комірки області даних, адреса якої *M*, то прапорець *RFG* прийме значення *TRUE*, якщо *RA* менше значення комірки області даних, адреса якої *M*, то прапорець *RFL* прийме значення *TRUE*. Аналізуючи прапорці, можна здійснювати команди умовних переходів, або зміну природної лінійної послідовності виконання команд програми за допомогою команд JMxx. Слід відзначити, дана команда повинна використовуватися разом з командами умовних переходів JMxx і йти першою. Наприклад:

CMPA 10 // якщо значення *RA* більше за значення в 10–ії комірці, то перейти до
// команди в 20-ій комірці кодового сегмента
JMGT 20 // *if x>y then goto Label20;*

Пам'ятайте, що в команді *CMPA 10 10* – це номер комірки області даних, а в команді *JMGT 20 20* – це адреса комірки кодового сегменту, яка містить команду!

Команди, які виконуються аналогічно до команди *CMPA M*:
CMPI M, CMAI M.

JMEQ M

Дана команда здійснює передачу управління команді, що розташована в комірці кодового сегменту з номером *M*, якщо прапорець *RFZ* має значення *TRUE*. Якщо прапорець *RFZ* має значення *TRUE*, то виконання програми продовжується з наступної команди. Нехай поточна команда *JMEQ* має адресу m_1, значення регістра *RA* дорівнює *5*, а комірка області даних за адресою *10* містить значення *5*. Тоді після виконання команд

CMPA 10
m_1 : JMEQ 30
m_1+1: …
…
30 : …

наступною буде виконуватись команда, яка розташована в комірці 30 кодового сегменту. Якщо ж комірка області даних за адресою *10* містить значення, наприклад, *3*, то наступною буде виконуватися команда, яка міститься в комірці з номером m_1+1.

Команди, які виконуються аналогічно до команди *JMEQ M*:
JMNE M, JMGT M, JMGE M, JMLT M, JMLE M.

LOOP M

Дана команда призначена для зручної організації циклу з параметром. Після кожної ітерації значення регістра *RI* зменшується на 1 і, якщо воно більше або рівне нулю, то виконується наступна ітерація інакше цикл завершується. Наприклад, для відшукання суми *10* елементів масиву *y* можна скористатися наступними конструкціями мови Pascal:

```
1)   x:=0;
     For j:=1 to 10 do x:=x+y[j];
2)   x:=0; j:=10;
     Repeat
       x:=x+y[j];
       j:=j-1;
     Until j<1;
```

Якщо припустити, що масив *y* розміщується в області даних починаючи з *20* – ї комірки, то в системі команд абстрактної обчислювальної машини ця операція буде реалізована наступним чином:

```
1:   ENTA 0   // x:=0;
2:   ENTI 9   // j:=9;
3:   ADAI 20  // Repeat
              //   x:=x+y[j];
4:   LOOP 3   // Until j<0;
```

У відповідності до схеми, яка зображена на рис. 7 та кодуванням машинних команд з наведеної вище таблиці, опишемо приблизний варіант реалізації на мові Turbo Pascal одного кроку роботи інтерпретатора команд для виконання чергової команди, адреса якої знаходиться в регістрі *IC*. Представлена нижче функція повертає невід'ємне значення як номер комірки кодового сегменту, в якій міститься команда, що буде виконуватись наступною при умові, що поточна команда допустима (інтерпретатор розпізнав код команди та операндів) і це не команда завершення програми *STOP*. Якщо поточна команда – це команда завершення програми, то функція поверне значення *-1*. Якщо ж інтерпретатор не може виконати команду, то функція поверне значення *-2*.

```
Function Operation:Integer;
var kk,ki,k:integer;
begin
      k:=IC;          // Запам'ятовуємо значення лічильника команд.
 kk:=round(code[K].opr);  //Якщо операнд команди це номер комірки
                          //області даних, то запам'ятаємо його в
                          //цілочисловій змінній.
 ki:=round(RI);           //Якщо регістр використовується як
                          //лічильник, то він мусить бути
                          //цілочисловим.
  case code[K].cop of
  0:K:=-1;
  1:begin RA:=data[kk];inc(K);end;// LDRA
  2:begin RI:=data[kk];inc(K);end;// LDRI
  6:begin data[kk]:=RA;inc(K);end;// STRA
  7:begin data[kk]:=RI;inc(K);end;// STRI
  11:begin RA:=code[k].opr;inc(K);end;// ENTA
  12:begin RI:=code[k].opr;inc(K);end;// ENTI
  56:begin RA:=RA+code[K].opr;inc(K);end;// INCA
  57:begin RI:=RI+code[K].opr;inc(K);end;// INCI
  58:begin RA:=RA-code[K].opr;inc(K);end;// DECA
  59:begin RI:=RI-code[K].opr;inc(K);end;// DECI
```

```
    50:begin RA:=RA+data[kk];inc(K);end;// ADDA
    51:begin RI:=RI+data[kk];inc(K);end;// ADDI
    53:begin RA:=RA-data[kk];inc(K);end;// SUBA
    54:begin RI:=RI-data[kk];inc(K);end;// SUBI
    60:begin RA:=RA*data[kk];inc(K);end;// MULA
    61:begin RI:=RI*data[kk];inc(K);end;// MULI
    65:begin RA:=RA/data[kk];inc(K);end;// DIVA
    66:begin RI:=RI/data[kk];inc(K);end;// DIVI
    52:begin RA:=RA+data[kk+ki];inc(K);end;// ADAI
    55:begin RI:=RI-data[kk+ki];inc(K);end;// SBAI
    62:begin RA:=RA*data[kk+ki];inc(K);end;// MLAI
    67:begin RI:=RI/data[kk+ki];inc(K);end;// DVAI
    63:begin RA:=RA*code[K].opr;inc(K);end;// MLAN
    64:begin RI:=RI*code[K].opr;inc(K);end;// MLIN
    68:begin RA:=RA/code[K].opr;inc(K);end;// DVAN
    69:begin RI:=RI/code[K].opr;inc(K);end;// DVIN
    72:begin RA:=round(RA) div round(data[kk]);inc(K);end;
    70:begin RA:=round(RA) mod round(data[kk]);inc(K);end;
    73:begin RA:=round(RA) div kk;inc(K);end;
    71:begin RA:=round(RA) mod kk;inc(K);end;
    13:begin Writeln('RA=',RA:8:2); inc(k); end; // WRRA
    14:begin Writeln('RI=',RI:8:2); inc(k); end; // WRRI
    15:begin Writeln('RA=');Readln(RA); inc(k); end; // RDRA
    16:begin Writeln('RI=');Readln(RI); inc(k); end; // RDRI
    20:k:=kk;// GOTO
    21:begin // CMPA
       RF.RFZ:=false;RF.RFL:=false;RF.RFG:=false;
       if RA=data[kk] then RF.RFZ:=true
       else if RA<data[kk] then RF.RFL:=true
       else RF.RFG:=true; inc(K);
       end;
    28:begin // CMPI
       RF.RFZ:=false;RF.RFL:=false;RF.RFG:=false;
       if RI=data[kk] then RF.RFZ:=true
       else if RI<data[kk] then RF.RFL:=true
       else RF.RFG:=true; inc(K);
       end;
    22:if RF.RFZ then k:=kk else inc(k);// JMEQ
    23: if not RF.RFZ then k:=kk else inc(k);// JMNE
    24: if RF.RFG then k:=kk else inc(k);// JMGT
    25: if (RF.RFG or RF.RFZ) then k:=kk else inc(k);
    26: if RF.RFL then k:=kk else inc(k);// JMLT
    27: if (RF.RFL or RF.RFZ) then k:=kk else inc(k);
    else k:=-2;
  end;
Operation:=k;
end;
```

2.5. Приклади програм на мові абстрактної одноадресної обчислювальної машини

За замовчуванням вважається, що перша команда програми розташована в 0-ій комірці кодового сегменту. Номери комірок наводяться в текстах програм для полегшення розуміння змісту програми. При реалізації завдань текст програми вводиться без нумерації команд.

2.5.1. Реалізація лінійних алгоритмів

Проілюструємо реалізацію лінійних алгоритмів на мові абстрактної обчислювальної машини на прикладі.

Задача 1. Дано два числа *a* і *b*. Знайти середнє арифметичне цих чисел.

Розв'язання

Опишемо використання комірок області даних. Нехай в 0-ій комірці буде міститися значення *a*, в 1-ій – значення *b*, в 2-ій – результат. Тоді текст програми буде такий:

```
0: RDRA   //ввести значення a в RA
1: STRA 0 //зберегти RA в 0-й комірці
2: RDRA   //ввести значення b в RA
3: STRA 1 //зберегти b в 1-й комірці (значення b залишається в RA)
4: ADDA 0 //додати до RA значення з 0-ої комірки
5: DVAN 2 //поділити на 2 значення RA
6: STRA 2 //зберегти результат в 2-ій комірці(він залишається в RA)
7: WRRA   //вивести результат на екран
8: STOP   //кінець програми
```

Завдання для самостійної роботи

1. Дано два дійсних числа a і b. Знайти їх суму, різницю і добуток.
2. Кут α задано в градусах. Визначити його величину в радіанах.
3. Спекулянт, який має стартовий капітал k грн, зайнявся торгівлею, яка щомісячно збільшує капітал на p %. Через який час капітал спекулянта складатиме $2k$ грн.
4. Дано катети прямокутного трикутника. Знайти його площу та гіпотенузу.
5. Задано $x_1.x_2$ – корені квадратного рівняння. Знайти коефіцієнти цього квадратного рівняння
6. Визначити час падіння каменя масою m, який скинули з висоти h.
7. Довжину відрізка l задано в дюймах (1 дюйм ≈ 2,54см). Перевести значення довжини відрізка в метричну систему.
8. Дано довжину кола. Визначити площу круга, обмеженого колом та довжину кола.
9. Знайти координати вершини параболи $y = ax^2 + bx + c$.
10. Знайти площу кільця, внутрішній радіус якого дорівнює 10, а зовнішній – даному числу ($r > 10$).
11. Зростаючу арифметичну прогресію задано першим членом a, кроком h та загальною кількістю членів n. Знайти:
    1) суму геометричної прогресії; 2) k-й елемент арифметичної прогресії.
12. Задано двоцифрове натуральне число n. Знайти суму, різницю та добуток цифр, які складають дане число.
13. Задано чотирицифрове натуральне число n. Знайти суму першої та останньої цифри цього числа.
14. Задано трицифрове натуральне число n. Знайти середнє арифметичне цифр, які складають дане число.
15. Знайти відстань між двома точками з координатами (x_1, y_1) та (x_2, y_2).
16. Знайти скалярний добуток векторів $a = (a_1, a_2, a_3)$ та $b = (b_1, b_2, b_3)$.
17. Дано a, b, c – довжини сторін трикутника. Обчислити периметр та площу трикутника.

18. В рівнобічній трапеції a,b – довжини основ, l – довжина бічної сторони. Обчислити площу трапеції.
19. Визначити, яку суму одержить працівник за виконану роботу, якщо йому нараховано S гривень, а податок становить 25%.
20. У класі – N учнів. З них M – хлопці. Знайти співвідношення у відсотках кількості хлопців та кількості дівчат у цьому класі.
21. Дано значення змінних x, y, z. Обчислити значення змінної t:

1) $t = \frac{x+y}{x-2} + y$; 2) $t = (((x+2)y+3)y+4)y$; 3) $t = (x+y)((x+y)+3)$;

4) $t = \frac{(x+y)}{(x-y)} + \frac{(x-y)}{(x+y)}$; 5) $t = ((x+y)+2)(x+y)^2$; 6) $t = xy + x^2y + xy^2 + x^3y^3$.

2.5.2. Реалізація алгоритмів з розгалуженням

Проілюструємо реалізацію алгоритмів з розгалуженням на мові абстрактної обчислювальної машини на прикладах.

Задача 2. Дано два числа *a* і *b*. Знайти більше з цих чисел.

Розв'язання

Опишемо використання комірок області даних. Нехай в 0-ій комірці буде міститися значення *a*, в 1-ій – значення *b*, в 2-ій – результат. Тоді текст програми буде такий :

```
0: RDRA   //ввести значення a в RA
1: STRA 0 //зберегти RA в 0–й комірці
2: RDRA   //ввести значення b в RA
3: STRA 1 //зберегти b в 1-й комірці(значення b залишається в RA)
4: CMPA 0 //порівнюємо RA з 0-ою коміркою(b в RA, a в 0-ій комірці)
5: JMGT 9  // якщо b > a, то переходимо до команди в 9–ій комірці
           // якщо ні, то продовжуємо виконання з 6-ої команди
6: LDRA 0  // переписуємо значення a з 0-ої комірки в RA
7: STRA 2  // зберігаємо як результат в 2-ій комірці
8: GOTO 10 // переходимо до 10-ї команди
9: STRA 2  // значення b з RA зберігаємо в 2-ій комірці
10:WRRA    // тепер RA містить max(a,b); виводимо його на екран
11:STOP    // кінець програми
```

Задача 3. Дано дійсні числа *a,b,c*. З'ясувати чи належить число *c* інтервалу *[a,b]*.

Розв'язання

Нехай в 0-ій комірці буде міститися значення *a*, в 1-ій – *b*, а в 2-ій – *c*. Якщо число *c* належить вказаному інтервалу, то на екран буде виведено значення *1*, в іншому разі – значення *0*.

```
0:  RDRA   // вводимо a в RA
1:  STRA 0 // зберегти RA в 0–й комірці
2:  RDRA   // вводимо b в RA
3:  STRA 1 // зберегти RA в 1–й комірці
4:  RDRA   // вводимо c
5:  STRA 2 // зберегти RA в 2–й комірці (c залишається в RA)
6:  CMPA 0 // порівнюємо c і a
7:  JMLT 12 // якщо c < a, то виводимо 0
8:  CMPA 1  // порівнюємо c і b
9:  JMGT 12 // якщо c > b, то виводимо 0, інакше
10: ENTA 1  // заносимо 1 в RA та
11: GOTO 13 // переходимо на вивід
12: ENTA 0  // заносимо 0 в RA та
13: WRRA    // виводимо на екран результат
14: STOP    // кінець
```

Завдання для самостійної роботи

1. Задано три числа: a,b,c. Визначити, чи можуть вони бути сторонами трикутника.
2. Задано a,b,c – сторони трикутника. Визначити, чи є цей трикутник рівностороннім.
3. Задано вектор $x \in R^3$. Визначити, чи є цей вектор нульовим.
4. Задано вектори $x, y \in R^3$. З'ясувати, чи є вони паралельними або ж перпендикулярними.
5. Задано вектори $x, y, z \in R^3$. З'ясувати, чи виконується рівність $z = x + y$.
6. Перевірити, чи лежить коло $(x-a)^2 + (y-b)^2 = r^2$ повністю в середині кола $(x-a_1)^2 + (y-b_1)^2 = r^2$.
7. Визначити, чи є значення x_1 розв'язком квадратного рівняння $ax^2 + bx + c = 0$.
8. Дано двоцифрове натуральне число n. Чи вірно, що перша цифра цього числа є більшою за другу.
9. Дано чотирицифрове натуральне число n. Підрахувати кількість цифр цього числа, які рівні 1.
10. Дано трицифрове натуральне число n. Визначити суму двох деяких цифр цього числа, яка є найбільшою.
11. Дано два двоцифрові натуральні числа n і m. Знайти суму найменшої цифри в числі n та найбільшої цифри в числі m.
12. Турист може подолати ділянку шляху S км на таксі зі швидкістю v_1 км/год та оплатою p грн/км або йти пішки з швидкістю v_2 км/год. Як з найменшими витратами подолати шлях за час t_1.
13. Банк пропонує 3 види депозитних вкладів: на 3 місяці під p_1%, на 6 місяців під p_2% і на рік під p_3%. Визначити, який із вкладів є найкращим за рік.
14. Визначити, чи відносить рік r до XX століття.
15. Дано три дійсних числа: a, b, c. Знайти $\max(a,b) + (\min(b,c))^2$.
16. Дано три дійсних числа: a, b, c. Знайти $\max(a, \max(b,c)) + \min(c^2, \max(b,c))$.
17. Дано три дійсних числа: a, b, c. Визначити, чи є вони членами арифметичної прогресії.
18. Дано три дійсних числа: a, b, c. Визначити, чи є вони членами геометричної прогресії.
19. Дано числа b,c та x_1, x_2. Визначити, чи є числа b,c коефіцієнтами квадратного рівняння $x^2 + bx + c = 0$, де x_1, x_2 – розв'язки цього квадратного рівняння.
20. Дано дійсне число a. З'ясувати, чи належать це число інтервалу $[1,2] \cup (3,7)$.
21. Задано три числа a, b, c, які є довжинами сторін трикутника. Визначити, чи є цей трикутник рівнобедреним..

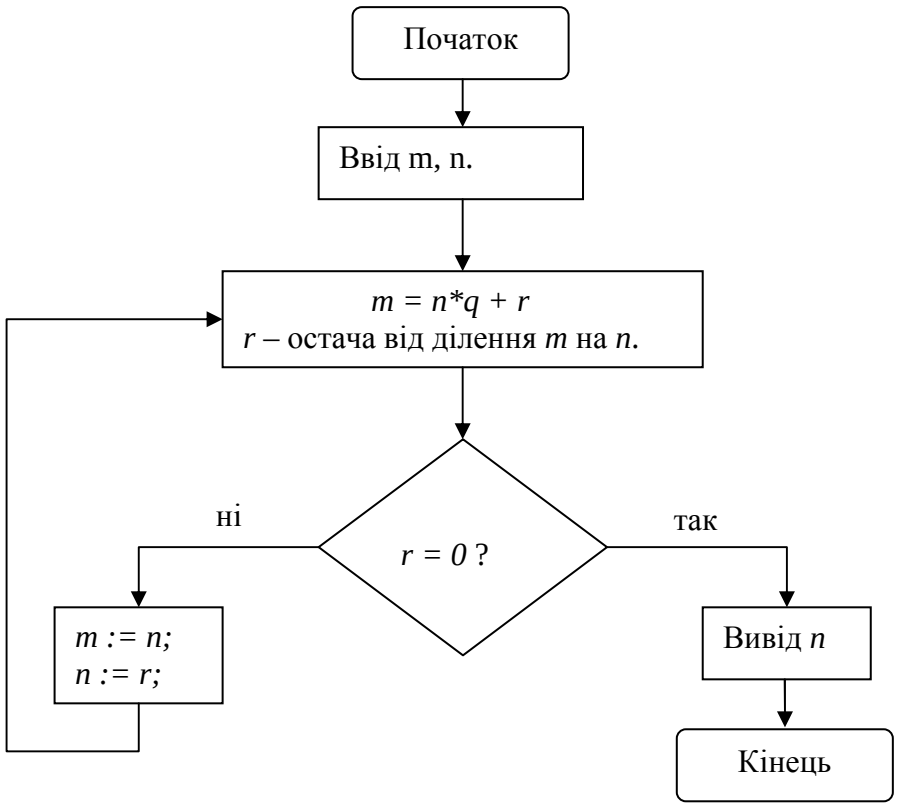
2.5.3. Реалізація циклічних алгоритмів

Проілюструємо реалізацію циклічних алгоритмів на мові абстрактної обчислювальної машини на прикладах.

Задача 4. Дано натуральні числа *m* і *n*. Знайти найбільший спільний дільник цих чисел.

Розв'язання

Нехай в 0-ій комірці міститься значення *m*, в 1-ій – значення *n*, в 2-ій – результат. В процесі роботи алгоритму значення 0-ої та 1-ої комірок можуть змінюватись відповідно до роботи алгоритму. Згадаємо блок-схему алгоритму:



На мові абстрактної обчислювальної машини даний алгоритм виглядає так:

```
0 : RDRA    // вводимо значення m та зберігаємо його в
1 : STRA 0  // 0-ій комірці
2 : RDRA    // вводимо значення n та зберігаємо його в
3 : STRA 1  // 1-ій комірці
4 : ENTA 0  // в RA заносимо 0
5 : STRA 2  // записуємо це значення в 2-гу та
6 : STRA 3  // 3-тю комірки
7 : LDRA 0  // в RA заносимо значення m
8 : MODM 1  // в RA записуємо остачу від ділення m на n
9 : CMPA 3  // якщо остача в RA дорівнює 0, то
10: JMEQ 15 // переходимо до команди в 15-ій комірці, інакше
11: LDRI 1  // переписуємо значення n в RI
12: STRI 0  // зберігаємо його в 0-ій комірці (m:=n;)
13: STRA 1  // значення остачі, що знаходиться в RA
            // переписуємо в 1-у комірку ( n:=r;)
14: GOTO 7  // і переходимо до 7-ої команди
15: LDRA 1  // записуємо в RA значення як результат
16: STRA 2  // зберігаємо його в 2-ій комірці
17: WRRA    // виводимо результат на екран
18: STOP    // кінець програми
```

Задача 5. Дано натуральне число *n*. Знайти суму його цифр.

Розв'язання

Згадаємо, що остача від ділення числа на 10 дає молодшу цифру цього числа, а цілочислове ділення цього числа на 10 дає число, в якому усі цифри будуть зсунуті на одну позицію вліво. Наприклад, якщо *n = 543*, то *n mod 10 = 3, n div 10 = 54*. Нехай значення *n* буде міститися в 0-ій комірці, результат – в 1-ій комірці.

```
0: RDRA    // Readln(RA)
1: STRA 0  // RA -> [0]
2: ENTI 0  // RI:=0
3: LDRA 0  // [0] -> RA
4: MODN 10 // RA:= RA mod 10
5: STRA 2  // RA -> [2]
6: ADDI 2  // RI:= RI + [2]
7: LDRA 0  // [0] -> RA
8: IDVN 10 // RA:= RA div 10
9: STRA 0  // RA -> [0]
10: CMAN 0 // якщо RA <> 0, тоді
11: JMNE 3 // перейти до 3-ої команди
12: STRI 1 // інакше зберігаємо результат RI -> [1]
13: WRRI   // Writeln(RI);
14: STOP   // кінець програми
```

Задача 6. Знайти суму перших n натуральних чисел.

Розв'язання

```
0: RDRA    // вводимо значення n та зберігаємо його в
1: STRA 0  // 0-ій комірці (RA -> [0])
2: ENTA 0  // RA:=0
3: LDRI 0  // RI:=[0] (в RI заносимо значення n)
4: STRI 1  // [1]:=RI
5: ADDA 1  // RA:=RA+[1]
6: LOOP 4  // зменшення RI (RI:=RI-1) та перехід до команди 4
7: WRRA    // виводимо значення суми на екран
8: STOP    // кінець програми
```

Завдання для самостійної роботи

1. Знайти n-тове ($n > 2$) число Фібоначчі.
2. Дано дійсне число x_0 та натуральне k. Знайти x_k згідно наступної рекурентної формули:

   1) $x_{i+1} = i*(x_i + 1)$; 2) $x_{i+1} = \frac{x_i}{x_i - 1}$;

   3) $x_{i+1} = x_{i-1} + 2x_i$; 4) $x_{i+1} = x_0 - \frac{1}{x_i}$.

3. Дано дійсне число a і натуральне число n. Обчислити:

   1) $a(a+1)...(a+n-1)$; 2) $\frac{1}{a} + \frac{1}{a(a+1)} + ... + \frac{1}{a(a+1)...(a+n)}$;

   3) $\sum_{i=1}^{20} i^2$; 4) $\sum_{i=1}^{10} \frac{1}{i}$;

   5) $\prod_{i=1}^{20} \frac{i+1}{i-2}$; 6) $\sum_{i=1}^{10} \frac{1}{i!}$

4. Задано натуральне число n. Знайти кількість цифр числа n.
5. Задано натуральне число n. Знайти кількість цифр числа n, які дорівнюють 1.
6. Задані натуральні числа n та m. Визначити, у якому з чисел цифра 2 зустрічається раніше (починаємо розгляд цифр з старших розрядів).
7. Задано натуральне число n. Чи утворюють цифри цього числа зростаючу послідовність.
8. Задано натуральне число n. Чи утворюють цифри цього числа арифметичну прогресію.

9. Обчислити значення функції

1) $f(x)=\min\left(x,\prod_{i=1}^{5}x^2\right)$;

2) $f(x)=x+x^2+x^3+...+x^{10}$

3) $f(x)=\frac{x+1}{1}*\frac{x+2}{2}*...*\frac{x+10}{10}$

4) $f(x)=\begin{cases}\sum_{i=1}^{9}x^i-i, & \text{ÿêùî } x>3;\\ \sum_{i=1}^{9}x^i-2i, & \text{â³íøîìó âèïàäêó}.\end{cases}$

10. Задано вектор $x\in R^n$. Підрахувати кількість координат цього вектора, які більші за середнє арифметичне координат цього вектора.
11. Задано вектор $x\in R^n$. Підрахувати кількість координат цього вектора, які дорівнюють нулю.
12. Задано $x,y\in R^n$. Визначити, чи є ці вектори паралельними.
13. Задано $x,y\in R^n$. Визначити вектор, у якого сума координат є найбільшою.
14. Задано $x,y\in R^n$. Визначити вектор, у якого нульова координата зустрічається раніше.
15. Відомий час початку та закінчення роботи маршрутного таксі. Відомий також час проходження маршруту в одну сторону. Вивести на екран розклад руху маршрутного таксі на одній з кінцевих зупинок.
16. Дано n натуральних чисел $a_1,a_2,...,a_n$. Визначити кількість чисел, які
    1) є непарними; 2) є більшими за 10;
    3) при діленні на 7 дають остачу 2; 4) мають парні порядкові номери та є непарними.
17. Дано $n\in N$. Побудувати алгоритм для знаходження $n!$.
18. Дано цілі числа $q_1,q_2,...,q_n$. Знайти суму тих членів послідовності, які:
    1) є додатними та парними; 2) не кратні 7;
    3) є меншими за n; 4) мають порядкові номери, кратні 3.
19. Дано натуральне число n і дійсні числа $x,a_1,a_2,...,a_n$. Визначити, скільки разів число x трапляється серед таких чисел:
    1) $a_1,a_2,...,a_n$; 2) $a_1,a_1+a_2,....,a_1+a_2+...+a_n$;
    3) $a_1^2,a_2^2,...,a_n^2$; 4) $a_1+1,a_2+2,...,a_n+n$.
20. Дано натуральне число n і дійсні числа $a_1,a_2,...,a_n$. Визначити:
    1) $\max(a_1,a_2,...,a_n)$; 2) $\min(a_1,a_2,...,a_n)$;
    3) $\max(a_1^2,a_2^2,...,a_n^2)$; 3) $\min(a_1,2a_2,...,na_n)$.
21. Дано натуральне число n і дійсні числа $a_1,a_2,...,a_n$. Обчислити:
    1) кількість додатних; 2) кількість від'ємних;
    3) кількість нульових; 3) середнє арифметичне.

2.5.4. Реалізація допоміжних алгоритмів та їх застосування

Нагадаємо, що *допоміжний алгоритм* – це алгоритм, який створюється наперед і викликається та повністю виконується в деякому іншому алгоритмі тоді, коли виникає в цьому необхідність. Саме з використанням допоміжних алгоритмів реалізується технологія процедурного програмування.

Для реалізації можливості застосування підпрограм розділимо стекову пам'ять на дві частини. Перша частина стекової пам'яті – ***стек даних*** – буде містити значення комірок сегменту даних для збереження параметрів підпрограм, результатів роботи підпрограм, а також для більш зручного маніпулювання даними в основній програмі. Друга частина стекової пам'яті – ***командний стек*** – буде містити адреси точок повернення з підпрограм. Наприклад, якщо в 25-ій комірці міститься команда CALL 50, тоді спочатку адреса повернення, або адреса наступної

команди, буде занесена в командний стек (тобто значення 26) і потім здійснюється перехід до підпрограми, яка розміщується з 50-ої комірки. Коли у підпрограмі зустрічається команда RETM, то з вершини командного стеку зчитується значення комірки кодового сегменту, до якої потрібно перейти, щоб продовжити виконання основної програми. За замовчуванням будемо вважати, що результат роботи підпрограми (якщо він є) повинен знаходитись в регістр-акумуляторі RA. При потребі результати роботи підпрограми, особливо якщо їх декілька, можна передавати через стек даних. Якщо підпрограма використовує параметри, тоді перед викликом підпрограми дані параметри слід помістити у стек (*1*-ий, *2*-ий, …, *n*-ий) за допомогою команд PUSHx. У підпрограмі для отримання цих параметрів необхідно прочитати їх зі стеку за допомогою команд POPx. Пам'ятайте, що значення зі стеку отримуються в оберненому порядку.

Проілюструємо реалізацію допоміжних алгоритмів на мові абстрактної обчислювальної машини на прикладі.

Задача 7. Знайти найбільший спільний дільник чотирьох натуральних чисел: n_1, n_2, n_3, n_4.

Розв'язання

Для розв'язання даної задачі використаємо раніше розроблену програму знаходження найбільшого спільного дільника двох чисел, оформивши її у вигляді підпрограми, яка розташована в оперативній пам'яті починаючи з комірки 19. Для передачі параметрів у підпрограму використаємо стек.

```
0 : RDRA    // вводимо значення n1 та зберігаємо його в
1 : STRA 0  // 0-ій комірці
2 : RDRA    // вводимо значення n2 та зберігаємо його в
3 : STRA 1  // 1-ій комірці
4 : RDRA    // вводимо значення n3 та зберігаємо його в
5 : STRA 2  // 2-ій комірці
6 : RDRA    // вводимо значення n4 та зберігаємо його в
7 : STRA 3  // 3-ій комірці.
8 : PUSH 0  // заносимо в стек значення n1
9 : PUSH 1  // заносимо в стек значення n2
10: CALL 19 // викликаємо підпрограму, яка починається з комірки 19
11: PUSH 2  // заносимо в стек значення n3 (в стеку вже
            // знаходиться найбільший спільний дільник n1 і n2)
12: CALL 19 // викликаємо підпрограму
13: PUSH 3  // заносимо в стек значення n4
14: CALL 19 // викликаємо підпрограму
15: POPA    // зчитуємо значення з стеку в RA
16: WRRA    // виводимо результат на екран
18: STOP    // кінець основної програми
            //  Підпрограма
19: POPA    // зчитуємо значення m з стеку в RA
20: STRA 4  // та зберігаємо його в 4-ій комірці
21: POPA    // зчитуємо значення n з стеку в RA
22: STRA 5  // та зберігаємо його в 5-ій комірці
23: LDRA 4  // в RA заносимо значення m
24: MODM 5   // в RA записуємо остачу від ділення m на n
25: CMAN 0   // якщо остача в RA дорівнює 0, то
26: JMEQ 31  // переходимо до команди в 31-ій комірці
27: LDRI 5   // інакше переписуємо значення n в RI
28: STRI 4   // зберігаємо його в 4-ій комірці (m:=n;)
29: STRA 5   // значення остачі, що знаходиться в RA
```

```
                // переписуємо в 5-у комірку ( n:=r;)
30: GOTO 23     // і переходимо до 23-ої команди
31: PUSHA       // заносимо результат в стек
32: RETM        // повернення в основну програму
```

Завдання для самостійної роботи

1. Дано три натуральних числа n, m, l. Обчислити:

   1) $\frac{n!+m!}{l!}$; 2) $\max\left(n^2, \max(n,m)\right)$; 3) $\frac{1+2+...+n}{m}+\frac{l}{1+2+...+m}$;

   4) $\sum_{i=1}^{n} l^i + \sum_{i=1}^{m} n^i + \sum_{i=1}^{l} m^i$; 5) $2^n + 3^m - 5^l$; 6) $n+\frac{n+m+\frac{n+m+l}{3}}{3}$.

2. Дано дійсні числа a, b, c, d. Обчислити:

   $$y = \left(p(a)+ p(b)+ p(c+2)+ p(2d)\right)/4,$$

   де $p(x) = 2x^2 + \frac{3}{x}$.

3. Дано ціле число a. Визначити:

   $$3f(a/2)+5f\left(a^2-2\right), \text{ де } f(x)=\sum_{i=1}^{10}\frac{x^i}{i}.$$

4. Дано вектори $x, y, z \in R^3$. Визначити вектор, довжина якого найбільша (найменша).
5. Дано вектори $x, y, z \in R^3$. Визначити пари перпендикулярних (паралельних) векторів.
6. Дано вектори $x, y, z \in R^3$. Визначити таку пару векторів, що вектор, який є їх сумою найдовший.
7. Дано вектори $x, y, z \in R^3$. Обчислити:

   1) $\langle x, y\rangle + \langle x, z\rangle - \langle y, z\rangle$; 2) $2x + 3y + 4z$;

   3) $|x| + |y| + 2|z|$; 4) $x + \left(x + \left(y + \left(y + z\right)\right)\right)$.

8. Дано вектори $x, y \in R^n$. У векторі x всі додатні координати замінити на 1, а у векторі y всі від'ємні координати замінити на 0.
9. Дано вектори $x, y, z \in R^n$. Визначити:
   1) у якому з векторів найбільша координата рівна 3;
   2) скільки серед них нульових векторів;
   3) в якому з них серед координат раніше зустрічається 1;
   4) в якому з них більше пар рівних сусідніх координат;
   5) координати скількох векторів утворюють зростаючу послідовність;
   6) координати скількох векторів утворюють арифметичну (геометричну) прогресію.
10. Трикутник задається координатами своїх вершин. Знайти довжини сторін трикутника.
11. Дано послідовність дійсних чисел з n елементів. Використовуючи допоміжний алгоритм, підрахувати кількість 1, кількість 3 та кількість 7.
12. На послідовність пар чисел $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Визначити дві пари чисел, які можуть утворити прямокутник найбільшої площі.
13. Дано натуральне число n і послідовності дійсних чисел $a_1, a_2, ..., a_n$ та $b_1, b_2, ..., b_n$. Обчислити:

    1) $\sum_{i=1}^{n} a_1 + \sum_{i=1}^{n} b_i$; 2) $\prod_{i=1}^{n} a_i^2 + \prod_{i=1}^{n} b_i^3$;

    4) $\coprod_{i=1}^{n}(a_i + b_i) + \coprod_{i=1}^{n}(a_i - b_i)$; 4) $a_1 b_1 + a_2 b_1 b_2 + ...a_n b_1 b_2 ... b_n$.

14. Дано дійсні числа $a_1, b_1, c_1, a_2, b_2, c_2, a_3, b_3, c_3$, які є довжинами сторін трьох трикутників. Визначити, який з трикутників має найбільшу площу.
15. Дано натуральне число n і дійсні числа $x_1, y_1, x_2, y_2, ..., x_n, y_n$. Знайти площу n-кутника, вершини якого при послідовному обході мають координати $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$.
16. Дано дійсне число x. Обчислити
$$y = \frac{\| x-2 | - | x-x^2 \|}{2},$$
використовуючи допоміжний алгоритм для знаходження модуля числа.
17. Дано дійсне число x та натуральне число n. Використовуючи допоміжний алгоритм, обчислити
$$y = \underbrace{(...(((}_{n} x^2 + 2x + 3)^2 + 2x + 3)^2 + 2x + 3)^2 + ... + 2x + 3.$$
18. Дано a, b, c – коефіцієнти біквадратного рівняння $ax^4 + bx^2 + c = 0$. Використовуючи допоміжний алгоритм для знаходження розв'язків квадратного рівняння, знайти розв'язки вказаного біквадратного рівняння.
19. Дано натуральні числа n, m, l. Використовуючи допоміжний алгоритм, знайти число, сума цифр якого є найбільшою.
20. Дано послідовність з m натуральних чисел $n_1, n_2, ..., n_m$. Обчислити:
    1) суму цифр всіх чисел;
    2) суму цифр всіх чисел, які більші за 5;
    3) суму цифр, які кратні 3;
    4) суму цифр, які є першими і останніми цифрами в кожному з чисел;
    5) добуток цифр всіх чисел.
21. Знайти площу поверхні піраміди, в основі якої трикутник. Піраміда задається координатами своїх вершин.

2.5.4. Використання динамічної структури стек

Проілюструємо використання динамічної структури стек на мові абстрактної обчислювальної машини на прикладі.

Задача 8. З клавіатури вводиться послідовність дійсних чисел. Організувати виведення чисел у порядку зворотному до порядку введення.

Розв'язання

```
0 : RDRA    // вводимо значення n та зберігаємо його в
1 : STRA 0  // 0-ій комірці (RA -> [0])
2 : LDAI 0  // в RI заносимо значення n  (RI:=[0])
3 : RDRA    // вводимо наступне значення
4 : PUSHA   // та заносимо його в стек
5 : LOOP 3  // зменшення RI (RI:=RI-1) та перехід до команди 3
6 : LDAI 0  // в RI заносимо значення n (RI:=[0])
7 : POPA    // зчитуємо значення з стеку в RA
8 : WRRA    // вводимо значення на екран
9 : LOOP 7  // зменшення RI (RI:=RI-1) та перехід до команди 7
10: STOP   // кінець програми
```

Завдання для самостійної роботи

1. Дано дійсні числа a, x. Використовуючи стек, обчислити:
   1) $y = (...(x_n + x_{n-1})a + x_{n-1} + x_{n-2})a + x_{n-2} + x_{n-3})a + ... + x_1)a$;
   2) $y = \left(1 + x + x^2 + x^3 + x^4\right)x^4 x^3 x^2 x$;

3) $y = \dfrac{(1+x)+(2+x^2)+...+(n+x^n)}{(1+x)(2+x^2)...(n+x^n)}$.

2. Дано натуральне число n. Отримати число m, у якого цифри йдуть у зворотному порядку по відношенню до n.
3. Дано послідовність n дійсних чисел $a_1, a_2, ..., a_n$. Використовуючи стек, знайти суму номерів чисел, які кратні 5.
4. Дано послідовність n дійсних чисел $a_1, a_2, ..., a_n$. Вивести значення:
   1) з парними індексами; 2) що є не додатними;
   3) що є парними; 4) що є кратними 7;
   5) що є більшими за a_1.
5. Дано послідовність n дійсних чисел $a_1, a_2, ..., a_n$. Вивести на екран всі можливі спадні послідовності, утворені з даних чисел $a_1, a_2, ..., a_n$.
6. Дано послідовність n дійсних чисел $x_1, x_2, ..., x_n$. Обчислити:
   $y = x_1 + (x_1 + x_2) + (x_1 + x_2 + x_3) + ... + (x_1 + x_2 + ... + x_n)$.
7. Задано функцію $f(x) = x + x^2 + x^3 + ... + x^n$, де $n \in N$, $x \in R$. Використовуючи стек, вивести на екран коефіцієнти похідної функції $f(x)$.
8. В стеку знаходиться послідовність n дійсних чисел $x_1, x_2, ..., x_n$. Знайти найменше (найбільше) серед них.
9. В стеку знаходиться послідовність n дійсних чисел $x_1, x_2, ..., x_n$, серед яких є два нулі. Вивести значення, які знаходяться між цими нульовими елементами.
10. Дано натуральне число n. Перевірити, чи серед цифр цього числа кількість одиниць відповідає кількості трійок.
11. Дано натуральне число n. Виписати всі цифри цього числа, які зустрічаються при переборі цифр після останньої одиниці і перед першою трійкою.
12. Дано послідовність n дійсних чисел $a_1, a_2, ..., a_n$. За один перегляд цих чисел знайти суму тих, які рівні найменшому серед них.
13. Дано дві послідовності з n дійсних чисел $a_1, a_2, ..., a_n$ та $b_1, b_2, ..., b_n$, які впорядковані за зростанням. Вивести на екран утворену з елементів вказаних двох послідовностей послідовність $c_1, c_2, ..., c_{2n}$, в якій елементи впорядковані за зростанням.
14. Дано не спадну послідовність n дійсних чисел $a_1, a_2, ..., a_n$. Вивести на екран послідовність утворену з даної, в якій елементи не повторюються.
15. Дано послідовність n дійсних чисел $a_1, a_2, ..., a_n$. Вивести на екран всі частини послідовності, що складаються з однакових елементів, які йдуть підряд один за одним і їх кількість більша або рівна двох.
16. Дано натуральне число n. Вивести на екран цифру, які при розгляді цифр числа повторюється першою та цифру, яка повторюється останньою.
17. Дано послідовність символів. Вибрати з цієї послідовності символи-цифри і дописати їх в кінці.
18. Дано послідовність символів. Вибрати з цієї послідовності символи-цифри і дописати їх з початку.
19. Дано послідовність символів, кількість яких парна. Поміняти місцями сусідні символи.
20. Дано послідовність n символів $a_1, a_2, ..., a_n$ (n – парне). Поміняти місцями символи з номерами *p-(j-1)* та *p+j, j=1,2,…,p, p = [n/2]*.
21. Дано послідовність символів, яка задає деякий складний арифметичний вираз, що записаний за правилами мови Pascal. Роздрукувати частину виразу, яка знаходиться між самими внутрішніми дужками.

Література

1. Гудман С. Введение в разработку и анализ алгоритмов. – М.: Мир, 1986.
2. Караванова Т.П. Інформатика: основи алгоритмізації та програмув.:777 задач з рек. та прикл.: Навч. посіб для 8-9кл. із поглибл. вивч. інф-ки/За заг. ред.. М.З.Згуровського – К.:Ґенеза, 2006. – 286 с.
3. Кнут Д. Искусство программирования на ЭВМ. – М.: Мир.
4. Пратт Т. Языки программирования. Разработка и реализация. – М.:Мир, 1986.

ЗМІСТ

Відповідальний за випуск: завідувач кафедрою системного аналізу і теорії оптимізації кандидат фізико-математичних наук, доцент О.І. Кузка

Автори: кандидат фізико-математичних наук, доцент Чупов С.В.,
викладач Брила А.Ю.

Рецензенти: кандидат фізико-математичних наук, доцент Гече Ф.Е.,
кандидат фізико-математичних наук, доцент Мич І.А.

ОБЧИСЛЮВАЛЬНІ СИСТЕМИ

Методичні вказівки до лабораторних робіт для студентів І-го курсу математичного факультету спеціальності "прикладна математика"