МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ МОРСЬКИЙ УНІВЕРСИТЕТ

Кафедра «Технічна кібернетика й інформаційні технології

ім. проф. Р.В. Меркта»

Затверджено
НМК Інституту інформаційних технологій та інноваційного підприємництва
Протокол № 3 від 04.12.2025
Керівник НМК

_____ Андрій ІВАНОВ

«4» грудня _____2025 р.

Методичні вказівки

для лабораторних робіт

з дисципліни «Схемотехніка та архітектура комп'ютерів».

Робота з 3-адресною навчальною машиною НМ-3.

Для здобувачів вищої освіти спеціальності

F5 «Кібербезпека та захист інформації» першого (бакалаврського)

рівня для денної та заочної форми навчання.

Одеса 2025

Методичні вказівки для лабораторних робіт з дисципліни «Схемотехніка та архітектура комп'ютерів». Робота з 3-адресною навчальною машиною НМ-3 підготовлені Ларіним Дмитром Георгійовичем, к.т.н., доцентом кафедри «Технічна кібернетика й інформаційні технології ім. проф. Р.В. Меркта» Одеського національного морського університету та Берковим Юрієм Миколайовичем, старшим викладачем кафедри «Комп'ютерних систем та технологій» Одеського національного університету ім. І.І. Мечникова.

Методичні вказівки до лабораторних робіт з дисципліни « Схемотехніка та архітектура комп'ютерів» схвалено кафедрою «Технічна кібернетика й інформаційні технології ім. проф. Р.В. Меркта» ОНМУ

«_06_»_____11_____ 2025 року, протокол № _8_

Завідувач кафедрою _____ Павло НОСОВ

НАВЧАЛЬНІ МАШИНИ

Короткі теоретичні відомості

Принципи організації та роботи ЕОМ зручно вивчати, розглядаючи моделі, а не реальні комп'ютери. Справа в тому, що на конструкцію будь-якої реальної обчислювальної машини крім основних проектних ідей, впливає безліч факторів, навіть випадковостей, що ускладнюють і заплутують пристрій машини. А модель може демонструвати ідею в чистому вигляді.

Будемо розглядати машини, що відповідають принципам фон Неймана, машини так званої фон-Нейманівської архітектури.

Структура ЕОМ

Основними елементами будь-якої обчислювальної машини є центральний процесор, оперативна пам'ять і зовнішні пристрої. Схематично комп'ютер зображений на рис 1.

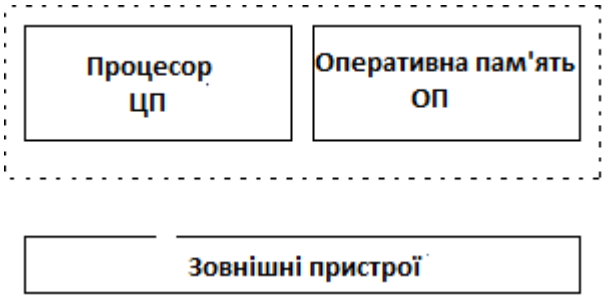


Рисунок 1. Схема ЕОМ

Центральний процесор (ЦП) виконує програму Користувача і управляє всіма пристроями ЕОМ. Оперативна пам'ять (ОП) під час роботи комп'ютера зберігає дані і виконувану програму. Зовнішні пристрої служать для зв'язку обчислювальної машини із зовнішнім світом: для введення і виведення даних, для довготривалого зберігання даних і програм.

Розглянемо структуру і роботу кожної з компонент.

Процесор

Центральний процесор-основна компонента обчислювальної машини. Саме процесор виконує програму Користувача. Крім цього, процесор організовує взаємодію між усіма частинами обчислювальної машини, так що вони утворюють єдине ціле – комп'ютер.

Елементами центрального процесора є пристрої і регістри. Регістри-це запам'ятовуючі елементи (осередки), розташовані всередині процесора; вони призначені для зберігання інформації. Схема центрального процесора наведено на рис. 2

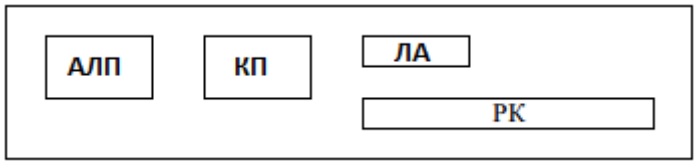


Рисунок 2. Схема центрального процесора

Керуючий пристрій (КП) управляє роботою процесора в цілому, координує роботу всіх інших пристроїв, організовує процес виконання програми. Арифметико-логічний пристрій (АЛП) виконує арифметичні і логічне операції, власне, виробляє обчислення. Регістр лічильник адреси (ЛА) містить адресу чергової команди програми. Регістр команди (РК) містить команду, яка виконується процесором в поточний момент часу.

Будемо вважати, виконання однієї машинної команди відбувається за один такт роботи процесора.

Розглянемо, як процесор виконує машинну команду на прикладі команди

Нехай 01 відповідає коду операції (КОП) "додавання"; А1, А2, А3 - адреси першого операнда, другого операнда і результату відповідно.

1. У РК зчитується з ВП команда, адреса якої записаний в ЛА.

2. Зміст ЛА збільшується на 1, так що тепер в ЛА вийшов адресу наступної команди програми.

3. КП аналізує вміст РК і організовує виконання команди.

Визначається, що треба виконати операцію"додавання". Визначаються А1, А2, А3. Вміст комірок ВП з адресами А1, А2 пересилаються в АЛП. Далі АЛП виконує дію додавання. Результат додавання з АЛП пересилається в комірку пам'яті з адресою А3.

Комірка ОП може містити дані або команду. Тепер зрозуміло, як процесор відрізняє дані від команди: якщо адреса комірки зустрівся в команді як адреса операнда, процесор обробляє вміст комірки як дані; якщо адреса комірки вийшов в лічильнику адреси, процесор обробляє вміст комірки як команду. Вміст однієї і тієї ж комірки в один момент роботи процесора може трактуватися як дані, а в іншій - як команда.

Перед початком роботи процесора в регістр ЛА записується апаратно завжди один і той же адресу, і перша команда програми повинна розташовуватися в ОП в осередку саме з цією адресою.

Оперативна пам'ять

Оперативна пам'ять зберігає програму, яку виконує обчислювальна машина, і дані для програми. Оперативна пам'ять може зберігати дані тільки під час роботи машини. Коли машина вимикається, вміст оперативної пам'яті втрачається.

Оперативна пам'ять складається з елементів – комірок. Кількість комірок визначає об'ємом оперативної пам'яті. Комірки пронумеровані, номер комірки називається її адресою. Схематично Оперативна пам'ять зображена на рис. 3.

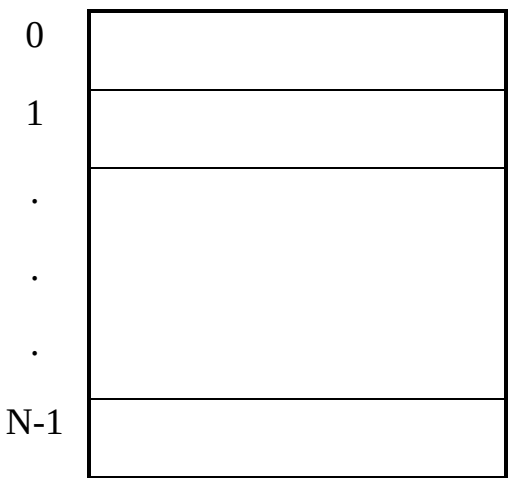


Рисунок 3. Оперативна пам'ять

Зауважимо, що обсяг ОП і розмір регістра ЛА взаємопов'язані: кількість розрядів в ЛА має бути достатньо для зберігання будь-якого можливого адреси.

Комірки складаються з розрядів. Кількість розрядів у всіх комірках однаково і називається розрядністю машини. Кожен розряд містить одну двійкову цифру.

Комірки можуть зберігати дані та команди. Доступ до комірки здійснюється за її адресою, номером. Працювати з частиною комірки не можна, тому часто поняття комірка визначають так: комірка – це мінімальна адресується одиниця пам'яті.

У будь-який момент часу можна звернутися до будь-якої комірки, незалежно від того, з якою коміркою працювали раніше. Час доступу до комірки не залежить від розташування комірки в пам'яті. Обидви ці обставини мають на увазі, кажучи, що Оперативна пам'ять – пристрій прямого (або довільного) доступу.

Слід мати на увазі, що Оперативна пам'ять набагато повільніший пристрій, ніж центральний процесор.

Зовнішні пристрої

Зовнішні пристрої призначені для зв'язку обчислювальної машини із зовнішнім світом (завантаження програми в оперативну пам'ять, введення даних і виведення результату рахунку) і для довготривалого зберігання програм і даних.

До пристроїв введення-виведення відносяться клавіатура, монітор, миша, принтер, сканер і т.п. пристрої введення-виведення працюють набагато повільніше процесора і оперативної пам'яті. Прикладами зовнішніх запам'ятовуючих пристроїв є гнучкі і жорсткі диски, флеш-накопичувач. У порівнянні з оперативною пам'яттю зовнішня пам'ять має набагато більший обсяг, але працює істотно повільніше. Час доступу до даних може залежати від їх розташування на носії.

Важливим фактором є те, що існує велика різноманітність зовнішніх пристроїв, кожне з них має свої фізичними характеристиками і особливостями функціонування, які потрібно враховувати при організації взаємодії обчислювальної машини і зовнішніх пристроїв.

Система команд

Кожен процесор має набір вбудованих, так званих машинних, операцій. Машинна операція – це елементарна дія, що виконується апаратним шляхом. Машинні операції реалізовані у вигляді електронних схем в процесорі. Приклади машинних операцій: пересилання вмісту комірки з оперативної пам'яті в процесор, додавання, віднімання. Будь-яке обчислення, яке потрібно зробити на обчислювальній машині, необхідно розкласти на елементарні дії, представити у вигляді послідовності машинних операцій.

Конкретний набір машинних операцій процесора залежить від специфіки класу задач, для вирішення яких передбачається використовувати обчислювальну машину, і від економічних факторів. Дії, що часто зустрічаються при вирішенні завдань, доцільно реалізовувати апаратно, рідкісні операції можна реалізувати програмно, висловивши їх через машинні операції.

Чим ширше набір машинних операцій, тим, взагалі кажучи, простіше програмувати для процесора, тим процесор складніше і дорожче. Чим вже набір машинних операцій, тим простіше процесор.

Наказ виконати машинну операцію для зазначених даних називається машинною командою. Правила запису машинної команди називається форматом команди. Список машинних операцій разом з форматами команд називається системою команд процесора. Машинна програма являє собою алгоритм рішення задачі, реалізований у вигляді послідовності машинних команд.

Варто зауважити, що часто слово "команда" використовують як синонім словосполучення "машинна операція". Однак ця багатозначність слова "команда" плутанини не викликає, так як з контексту завжди зрозуміло, про що йде мова – про апаратно реалізованої операції ("в процесорі є команда додавання") або про виконання дії з конкретними даними.

Як правило, в набір машинних операцій входять наступні групи команд.

- пересилання: призначені для обміну даними між процесором і оперативною пам'яттю;
- арифметичні команди: додавання, віднімання, множення, ділення та інші операції;
- переходи: команди передачі управління, що дозволяють програмувати умовні оператори і цикли.

Схема роботи процесора

Розглянемо, як функціонує обчислювальна машина. Оскільки операції введення-виведення не входять в тему обговорення, домовимося вважати, що на початку роботи обчислювальної машини в її оперативну пам'ять якимось

чином завантажена програма. Вихідні дані знаходяться в осередках пам'яті з відомими адресами. Результати також потрібно записати в відомі осередки оперативної пам'яті.

Робота обчислювальної машини складається з кроків – тактів. Виконання однієї машинної команди відбувається за один такт роботи процесора.

Робота відбувається наступним чином.

При включенні комп'ютера перш за все в регістр лічильник адреси (ЛА) записується деяке число. Ця дія виконується апаратно, незалежно від програміста, причому при кожному запуску процесора в ЛА записується одне і те ж число. Будемо вважати, що це число 0.

Після завантаження початкового значення в регістр ЛА, процесор починає послідовно, крок за кроком, виконувати команди програми, поки не зустріне команду зупину.

На кожному такті процесор виконує дії:

1. У регістр команди (РК) записується вміст комірки, адреса якої знаходиться в регістрі ЛА.

2. Значення ЛА збільшується на 1, так що тепер ЛА вказує на наступну команду програми.

3. Пристрій управління розшифровує команду, що знаходиться в РК, і організовує її виконання. Як саме виконується команда, залежить від виду самої команди.

Далі робота повторюється з першого кроку.

Процес закінчується, коли виконалася команда зупинки процесора.

Зауважимо, що, оскільки при включенні машини в ЛА записується число 0 і далі саме вміст комірки з адресою 0 поміщається в ЦП, розшифровується і виконується, в нульовій комірці повинна знаходитися перша команда програми, що виконується. Отже, або всі програми повинні починатися за нульовою адресою, тобто з комірки з адресою 0, або в комірку з адресою 0 повинна бути поміщена команда переходу на початок програми.

Підіб'ємо підсумок: регістр лічильник адреси зберігає адресу команди, яка буде виконуватися на наступному кроці роботи процесора; регістр команди містить команду, виконувану на поточному кроці.

Навчальна машина НМ-3

Розглянемо конкретизацію абстрактної машини фон Неймана на прикладі навчальної машини, яку будемо називати НМ–3 (сенс цієї назви – навчальна машина 3-адресна). Наша навчальна машина буде задовольняти всім принципам фон Неймана.

Пам'ять навчальної машини складається з 512 комірок, що мають адреси від 0 до 511, по 32 двійкових розряду кожна. У кожній комірці може бути записано ціле або дійсне число (представляються вони по-різному) або команда. Команда в комірці буде представлятися в наступній формі:

КОП	A1	A2	A3
5 разрядів	9 разрядів	9 разрядів	9 разрядів

Тут КОП – це число від 0 до 31, яке задає номер (код) операції, а A1, A2 і A3 – адреси операндів. Таким чином, в кожній команді задаються адреси аргументів (це A2 і A3)і адреса результату операції A1. Конкретизуємо регістри пристрою управління:

* Ra – регістр , званий лічильником адреси, він має 9 розрядів і зберігає адресу команди, яка буде виконуватися слідом за поточною командою;

* RK – регістр команд має 32 розряду і містить поточну виконувану команду (код операції КОП і адреси операндів A1, A2 і A3);

* ω – регістр «омега», в який після виконання деяких команд (у нас це будуть арифметичні команди додавання, віднімання, множення і ділення) записується число від 0 до 2 за правилом (S – результат арифметичної операції):

$$\omega := \begin{cases} 0, & S = 0, \\ 1, & S < 0, \\ 2, & S > 0; \end{cases}$$

* Err – регістр помилки, що містить нуль в разі успішного виконання чергової команди і одиницю в іншому випадку.

У таблиці 1 наведені всі команди навчальної машини НМ–3.

Схема виконання команд

Всі бінарні операції (тобто ті, які мають два аргументи і один результат) виконуються в нашій навчальній машині за схемою: <A1>:=<A2>□<A3> (□– будь-яка бінарна операція). Кожна команда виконується за наступним алгоритмом:

1. RK = <RA>; читання чергової команди з комірки пам'яті з адресою RA на регістр команд RK в пристрої управління.

2. RA = RA + 1; збільшення лічильника адреси на одиницю, щоб наступна команда (якщо поточна команда не є командою переходу) виконувалася з наступної комірки пам'яті.

3. Виконання операції, заданої в коді операції (КОП) в АЛП, при необхідності знак результату записується на регістр ω. При неіснуючому коді операції або інших помилках при виконанні команди (наприклад, розподілі на нуль) виконання команди припиняється і проводиться присвоювання Err=1.

4. if (Err=0) and (КОП ≠ ЗУП) then goto 1 else кінець.

Тепер нам залишилося визначити умову початку роботи програми. Для завантаження програми в пам'ять і формування початкових значень регістрів в пристрої управління на пристрої введення є спеціальна кнопка ПУСК (F9). При натисканні цієї кнопки пристрій введення самостійно (без сигналів з боку пристрою управління) виробляє наступну послідовність дій:

1. Проводиться введення розташованого на пристрої введення масиву машинних слів в пам'ять, починаючи з першого осередку; цей масив машинних слів закінчується спеціальною ознакою кінця введення.

2. RA = 1; першою буде виконуватися команда з комірки з номером один.

3. ω = 0; початкове значення ознаки результату нульове.

4. Err = 0; ознака помилки скидається (встановлюється рівним false).

Далі все готово для автоматичної роботи центрального процесора по завантаженої в пам'ять програмі. Таким чином, ми повністю визначили умови початку і кінця роботи нашої алгоритмічної системи.

За своєю архітектурою наша навчальна машина дуже схожа на перші ЕОМ, побудовані відповідно до принципів фон Неймана.

Емулятор роботи навчальної машини-3 (Education mashine-3)

Програма em3.exe є емулятором машини НМ-3 (навчальної 3-адресної машини).

МІНІМАЛЬНІ ВИМОГИ:

Intel 286 compatible

100 Kb HD space

16 Mb RAM

MS-DOS 3.0

Програма ЕМ-3 може працювати під управлінням будь-якої версії MS-DOS, FreeDOS або з під віртуальної машини DOSBox, запущеної в середовищі Windows XP/7/8/10. У деяких випадках може знадобитися змінити налаштування конфігурації DOSBox.

Загальний опис

ЕМ-3 підтримує повний стандартний набір команд, за винятком команди СТОП. Тобто, такої команди в мові ЕМ-3 немає. Зате є команда ЗУП, що означає Зупинка (це зроблено для того, щоб стандартизувати довжину всіх команд).

Таблиця 1.

Повний перелік підтримуваних команд

0	ПЕР	– пересилання: <A1>=<A3>
1	ДОД	– додавання дійсних чисел
2	ВДД	– віднімання дійсних чисел
3	МНД	– множення дійсних чисел
4	ДІД	– ділення дійсних чисел
5	ВВД	– введення A2 дійсних чисел в пам’ять, починаючи з адреси A1 for i=1 to A2 do Readln(<A1+i-1>)
6	ВВЦ	– введення массиву цілих чисел, аналогічно ВВЦ

7	???	

Продовження табл.1

8	???	
9	БЕЗ	– безумовний перехід: goto A2, т.е. RA=A2
10	ЦІЛ	– перетворення дійсне в ціле: <A1>=Round(<A3>)
11	ДОЦ	– додавання цілих чисел
12	ВДЦ	– віднімання цілих чисел
13	МНЦ	– множення цілих чисел
14	ДІЦ	– ділення цілих чисел (округлення вниз)
15	ВИД	– виведення A2 дійсних чисел, починаючи з адреси A1 for i=1 to A2 do Writeln(<A1+i-1>)
16	ВИЦ	– виведення масиву цілих чисел, аналогічно ВИД
17	???	
18	???	
19	УМВ	– умовний перехід: ω=0: RA=A1; ω=1: RA=A2; ω=2: RA=A3
20	ДСН	– перетворення ціле в дійсне: <A1>=Real(<A3>)
21	???	
22	???	
23	???	
24	МОД	– залишок від ділення цілих чисел
25	???	
26	???	
27	???	
28	???	

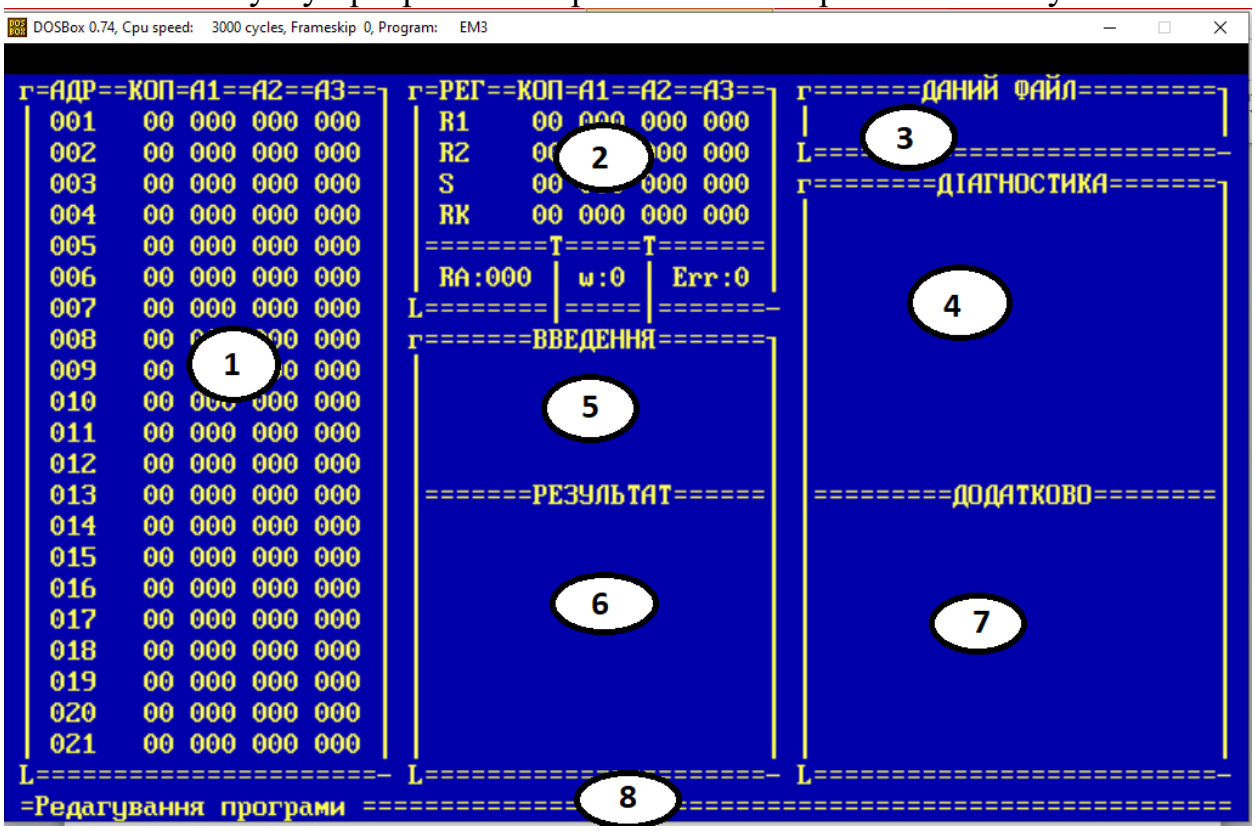
29	???	
30	???	

Продовження табл.1

31	ЗУП	– зупинка виконання програми

«???"- команда в даній версії не визначена.

Після запуску програми на екрані з'явиться приблизно наступне:



1 - Редактор
2 - Регістри
3 - Ім'я поточного файлу
4 - Відображення даних, що вводяться
5 - Висновок діагностичних повідомлень
6 - Висновок результатів роботи програми
7 - Опис того, що буде робити наступна команда (покроковий режим)
8 - Рядок стану

У вікні редактора ви побачите курсор. Його можна переміщати клавішами ←,→,↑,↓, <Home>, <End>, <PgUp>, <PgDn>, і <Enter>. Якщо натиснути цифрову клавішу 0-9, то в позиції курсора запишеться нова цифра. При цьому програма не дасть ввести безглуздий код. Тобто, наприклад, якщо в поле A1

було 421, то при спробі на місце 4 поставити 5, весь адреса автоматично заміниться на 511, тому що адреси 521 в НМ-3 немає. Аналогічна ситуація буде, якщо в поле коду операції спробувати ввести, наприклад, 35.

Вийти з програми можна, натиснувши <F10>.

Припустимо, програма написана. Як тепер її запустити? Є два режими:

1) звичайний (натиснути <F9>).

У цьому режимі емулятор просто почне виконувати програму, запитуючи, при необхідності, дані у вікні введення (5) і виводячи результат у вікно (6). Якщо програма зациклилася, можна її завершити натисканням клавіші <Esc>.

У разі якщо під час виконання програми зафіксована помилка, відбувається останов і комп'ютер видає повідомлення про помилку в вікно (4). Там же буде вказано рядок (тобто адреса команди), при виконанні якої сталася помилка.

2) покроковий(натиснути <Ctrl>+<F9>).

Аналогічний першому режиму, з очевидними змінами. Перед виконанням кожної команди, у вікні (1) буде виділятися рядок, в якій команда розташована. У вікні (7) виводиться короткий опис того, що команда буде робити. Команда не буде виконана, поки ви не натиснете якусь клавішу.

Мнемонічний режим

Машинні коди НМ - 3 (ЕМ-3) досить важко запам'ятати. Тому переважній більшості буде зручніше працювати в мнемонічному режимі (МР). Для входу/виходу з нього треба натиснути <F6>.

У МР все без винятку команди будуть відображатися з трьома буквами замість двох цифр в поле коду операції. При цьому їх можна і редагувати, але тут є одна особливість. КОП починайте вводити або змінювати з найпершої літери. При цьому 2-я і 3-я букви поміняються на ті, які присутні в коді операції команди з вашої першої буквою і найменшим кодом операції. Потім можна змінити другу букву, потім третю. Однак так не завжди буває зручно робити. Наприклад, програма зрозуміє вас в разі, якщо ви захочете в копі ВВЦ або ДОЦ замінити третю букву на 'в'.

І ще, не перемикайтеся в українську розкладку! Програма "розуміє", що Латинській літері d на клавіатурі відповідає українська В. Також не натискайте клавішу Shift при введенні.

Відкриття / збереження

Тепер про завантаження / збереження програми на диск. Використовуйте наступні комбінації клавіш:

< F2> - Зберегти

<Alt> + <F2> - Зберегти як...

<Ctrl> + <F2> - зберегти текст програми у файл...

<F3> - відкрити файл...

Процес відкриття файлу:

1. Натисніть <F3>. Курсор з'явиться у вікні (3).

2. а) якщо ви передумали відкривати файл, натисніть <Esc>.

б) введіть повний шлях до файлу(можна без розширення) і натисніть <Enter>.

3. Якщо програма змогла відкрити файл, то у вікні (3) з'явиться або повний шлях до файлу, або тільки ім'я файлу, якщо повний шлях у вікні (3) не поміщається. Тепер при натисканні < F2> програма буде збережена в той файл, ім'я якого відображено у вікні (3).

Збереження тексту програми у файл.

Після натискання <Ctrl> + <F2> у вікні (7) з'являться послідовно запити на введення:

1) Номер рядка, з якого починати збереження

2) Номер останнього рядка програми, що зберігається

3) Повний шлях до файлу, в який потрібно зберегти текст програми

Після виконання всіх зазначених дій вміст кінцевого файлу виявиться приблизно наступним (зберігався файл з 5 по 11 рядки не в мнемонічному режимі):

```
АДР     КОП   А1    А2   А3
005      19   006  012  012
006      11   002  002  016
007      11   004  004  016
008      12   000  017  002
009      19   010  010  002
010      16   100  005  000
011      09   000  013  000
```

Якщо в момент збереження був включений МР, то і текст програми буде відповідним, тобто коди команд заміняться їх мнемонічними позначеннями.

Зміна опцій

Натисніть <F7> і подивіться на вікно (7). Перемикання між опціями здійснюється клавішами ← та →, а зміна - клавішами ↑ та ↓. Для підтвердження зміни натисніть <Enter>, для скасування - <Esc>. Що яка опція означає, можна подивитися, натиснувши <F1>. До речі, контекстна довідка може бути викликана не тільки при зміні опцій, а, наприклад, й при аварійному завершенні програми на ЕМ-3 для роз'яснення виниклої помилки. Можна викликати довідку практично в будь-який момент.

У програмі є можливість зберегти опції за замовчуванням, щоб не змінювати їх при кожній новій завантаженні. Для цього в режимі зміни опцій натисніть клавішу F2. Вам буде запропоновано підтвердити збереження. У разі підтвердження, в файл options.dat будуть записані нові установки за замовчуванням.

Додаткові можливості

Ще про редагування програми.

У вікні редактора (1) діють клавіші:

<Ins> - вставити порожній рядок перед тим, на якому знаходиться курсор

< Del> - Видалити рядок, на якому знаходиться курсор

<Ctrl> + <Ins> - те ж, що і <Ins>, але з виправленням адрес - "розумна вставка"

<Ctrl> + <Del> - те ж, що і <Del>, але з виправленням адрес - "розумне видалення"

Коротко про те, що мається на увазі під "розумною вставкою і видаленням". Коли ви виконуєте, наприклад, звичайну вставку рядка клавішею <Ins> в середину вашої програми, виявляється неприємний момент: половину програми доводиться переписувати заново, тобто як мінімум міняти значення адрес в командах. Якщо ж використовувати" розумний " варіант вставки, то емулятор постарається автоматично замінити адреси (тільки необхідні) так, щоб програма могла виконуватися так само, як і до вставки.

Треба зауважити, що все, що знаходиться після першої команди ЗУП, буде розглядатися емулятором як числа, тобто адреси у відповідних комірках не будуть змінені. Оскільки студенти, як правило, мають у своєму розпорядженні програми, що виконують команди до команди ЗУП, в більшості випадків операції "розумної" вставки і видалення можуть істотно скоротити рутинні дії по виправленню адрес. Якщо ви хочете, щоб всі адреси у всіх командах змінювалися при натисканні комбінацій <Ctrl>+<Ins> і <Ctrl>+<Del>, вимкніть опцію "використовувати розумну вставку/видалення".

Ще в емуляторі передбачений зручний введення чисел в пам'ять ЕМ-3. У вікні редактора натисніть: <Alt>+<i> - для перегляду вмісту комірки в позиції курсора як цілого числа. Для виходу з режиму перегляду натисніть будь-яку клавішу крім <Enter>. Якщо натиснути <Enter>, то буде запропоновано змінити вміст комірки (тобто ввести нове значення). <Alt > + <f> - те саме, що і <Alt> + <i>, але для дійсних чисел. При цьому можна, наприклад, вводити так:

-1.2345
1.2 E-5 (одна ціла і дві десятих на десять в мінус п'ятої)
+ NAN (+НЕ_ЧИСЛО)
- NAN (- НЕ_ЧИСЛО)
+ INF (+НЕСКІНЧЕННІСТЬ)
- INF (- НЕСКІНЧЕННІСТЬ)

ЛАБОРАТОРНА РОБОТА №1.
Вивчення роботи емулятора 3-х адресної навчальної машини.
Створення лінійних програм.

Мета роботи: Ознайомитися з інтерфейсом емулятора трехадресної навчальної машини.

Обладнання: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBoxPortable, емулятор навчальної машини ЕМ-3.

Оператор присвоювання.

Складемо програму, яка реалізує арифметичний оператор присвоювання.

$y = (x+1)^2 \bmod (x-1)^2$.

Спочатку необхідно вирішити, в яких осередках пам'яті будуть розташовуватися наші змінні x і y. ця робота називається розподілом пам'яті під зберігання змінних. При програмуванні на мовах високого рівня ця робота виконується автоматично при виявленні оператора присвоювання.

Тепер нам доведеться розподіляти пам'ять самим. Зробимо припущення, що наша програма буде займати не більше 100 комірок пам'яті (нагадаємо, що програма вводиться, починаючи з першого осередку пам'яті при натисканні кнопки ПУСК(<F9>). Тоді, починаючи з 101 комірки, пам'ять буде вільна. Нехай для зберігання значення змінних ми виділимо 101 комірку, а змінної y – 102 комірку. Решта змінних, за необхідністю, будемо розміщувати в наступних комірках пам'яті. У наведеному прикладі нам знадобляться додаткові (як кажуть, робочі) змінні r1 і r2, які ми розмістимо в комірках 103 і 104 відповідно.

При програмуванні на машинній мові, у нас не будуть існувати константи. Дійсно, в яку би комірку ми не помістили значення константи, ніщо не завадить нам записати в цю комірку нове значення. Тому ми будемо називати константами такі змінні, які мають початкові значення, і які ми не плануємо змінювати в ході виконання програми. Звідси випливає, що такі константи, як і змінні з початковим значенням, слід розташовувати в тексті програми і завантажувати в пам'ять разом з програмою при натисканні кнопки ПУСК. Зрозуміло, такі змінні з початковими значеннями слід розташовувати в такому місці програми, щоб пристрій управління не почало б виконувати їх як команди. Щоб уникнути цього ми будемо розміщувати їх в кінці програми, після команди ЗУП.

Слід звернути увагу і на той факт, що спочатку програміст не знає, скільки комірок в пам'яті буде займати його програма. Тому адреси програми, що посилаються на змінні з початковим значенням, до завершення написання програми залишаються незаповненими, і вже потім, розмістивши ці змінні в пам'яті відразу слідом за командами програми, слід вказати їх адреси в тексті програми. У наведеному прикладі ті адреси програми, які заповнюються в останню чергу, будуть позначатися підкресленням.

Запис програми складається з рядків, кожен рядок забезпечується номером комірки, куди буде поміщатися це машинної слово (команда або змінна з початковим значенням) при завантаженні програми. Слідом за номером задаються всі поля команди, потім програміст може вказати коментар. Номери комірок, кодів операцій і адреси операндів будемо записувати в десятковому вигляді, хоча перші програмісти використовували для цього 8-у або 16-у системи числення. Крім того, так як числа не відрізняються за зовнішнім виглядом від команд, то будемо записувати їх теж найчастіше у вигляді команд. Текст нашої першої програми з коментарями наведено на рис.

```
№    Команда                    Коментар
1    ВВЦ 6  101 001 000         Read(x)
2    ДОЦ 11 103 101 008         r1 = (x + 1)
3    МНЦ 13 103 103 103         r1 = (x+1)2
4    ДОЦ 11 104 101 101         r2 = (x+x)=2*x
5    МОД 24 102 103 104         y = R1 mod r2
6    ВИЦ 16 102 001 000         Write (y)
7    ЗУП 31 000 000 000         Стоп
8           000 000 001         ціла константа 1
```

Після написання програми залишилося помістити на пристрій введення два масиви - саму програму (9 машинних слів) і число x (одне машинне слово) і натиснути кнопку ПУСК. Як ми вже говорили, перший масив закінчувався спеціальної рядком-ознакою кінця введення, так що пристрій введення знає, скільки машинних слів треба ввести в пам'ять по кнопці ПУСК.

Завдання для домашньої підготовки.

Вказівки до змісту звіту

1. Текст програми. Кожен рядок програми супроводити коментарями.
2. Результат роботи програми (скріншот виконаної програми).

Варіанти для лабораторної роботи №1

Використовувати в якості значення x номер свого варіанту.

Лабораторна робота № 2.
Реалізація умовного переходу.

Мета роботи: Ознайомитися з методами організації умовного переходу в 3-х адресній навчальній машині

Обладнання: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBox Portable, емулятор навчальної машини ЕМ-3.

Умовний оператор.

Складемо тепер програму, що реалізує умовний оператор присвоювання. Нехай цілочисельна змінна y приймає значення в залежності від введеної цілочисельної змінної x відповідно до правила:

$$y := \begin{cases} x+2, & \text{при } x < 2, \\ 2, & \text{при } x = 2, \\ 2*(x+2), & \text{при } x > 2; \end{cases}$$

В даному прикладі при записі програми на місці коду операції ми будемо для зручності замість числа вказувати його мнемонічне позначення.

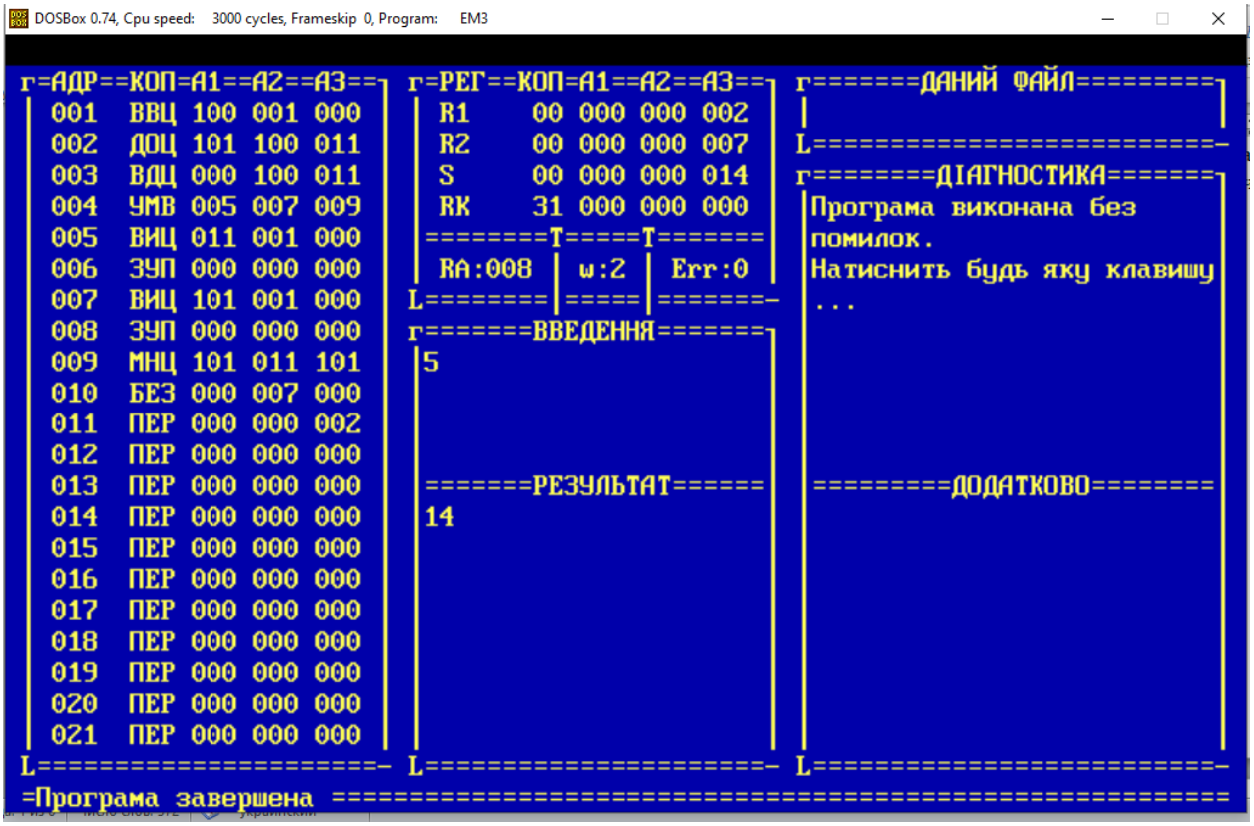
Для визначення того, чи є значення змінної x більше, менше або рівним константі 2, ми будемо виконувати операцію віднімання x–2, отримуючи в регістрі ω значення 0 при x=2, 1 при x<2 і 2 при x>2. При цьому сам результат операції віднімання нам не потрібен, але на наш формат команд вказівка на адресу комірки для запису результату є обов'язковою. Для запису таких непотрібних значень ми будемо найчастіше використовувати комірку за номером 0. Відповідно до принципу однорідності пам'яті, ця комірка нічим не відрізняється від інших, тобто, доступна як для запису, так і для читання даних. У деяких реальних ЕОМ цей принцип порушується: при зчитуванні з цієї комірки завжди повертався нуль, а запис в комірку з адресою нуль фізично не здійснюється (на практиці такий принцип роботи с цією коміркою іноді зручніше).

```
 №  Команда                 Комментарі
001 ВВЦ    100   001   000  Read(x)
```

```
002 ДОЦ    101   100   011   y = x+2
003 ВДЦ    000   100   011   <000> = x-2; формування ω
004 УМВ    005   007   009   Case ω of
                               0: goto 005;
                               1: goto 007;
                               2: goto 009 end
005 ВИЦ    011   001   000   Write(2)
006 ЗУП    000   000   000   Кінець роботи
007 ВИЦ    101   001   000   Write(y)
008 ЗУП    000   000   000   Кінець роботи
009 МНЦ    101   011   101   y = 2*y
010 БЕЗ    000   007   000   Goto 007
011 00     000   000   002   Ціла константа 2
```

Для зберігання змінних x і y виділимо комірки 100 і 101 відповідно. Програміст сам визначає порядок розміщення в програмі трьох гілок умовного оператора присвоювання. Нижче наведено текст цієї програми.

Нижче показаний вид програми в ЕМ-3:



Зверніть увагу, що константа 2 не відрізняється від команди пересилання змісту другої комірки пам'яті в нульову комірку, саме така команда і буде виконуватися, якщо ця константа буде обрана на регістр команд пристрою управління.

Завдання для домашньої підготовки.

Вказівки до змісту звіту

1. Текст програми. Кожен рядок програми супроводити коментарями.
2. Результат роботи програми (скріншот виконаної програми).

Варіанти для лабораторної роботи №2

Варіант 1

$$Y=\begin{cases} ax^2+b & \textit{при } x<0 \\ \dfrac{x-a}{x-c} & \textit{при } x>0 \\ \dfrac{x}{c} & \textit{в інших випадках} \end{cases}$$

Варіант 2

$$Y=\begin{cases} \dfrac{1}{ax}-b & \textit{при } x+5<0 \\ \dfrac{x-a}{x} & \textit{при } x+5>0 \\ \dfrac{10x}{c-4} & \textit{в інших випадках} \end{cases}$$

Варіант 3

$$Y=\begin{cases} ax^2+bx+c & \textit{при } x<0 \\ \dfrac{-a}{x-c} & \textit{при } x>0 \\ a(x+c) & \textit{в інших випадках} \end{cases}$$

Варіант 4

$$Y=\begin{cases} -ax-c & \textit{при } x<0 \\ \dfrac{x-a}{-c} & \textit{при } x>0 \\ \dfrac{bx}{c-a} & \textit{в інших випадках} \end{cases}$$

Варіант 5

Варіант 6

$$Y=\begin{cases} ax^2+b^2x & \textit{при } x<0 \\ \dfrac{x+a}{x+c} & \textit{при } x>0 \\ \dfrac{x}{c} & \textit{в інших випадках} \end{cases}$$

Варіант 7

$$Y=\begin{cases} -ax^2-b & \textit{при } x<5 \\ \dfrac{x-a}{x} & \textit{при } x>5 \\ \dfrac{-x}{c} & \textit{в інших випадках} \end{cases}$$

Варіант 8

$$Y=\begin{cases} -ax^2 & \textit{при } x<0 \\ \dfrac{a-x}{cx} & \textit{при } x>0 \\ \dfrac{x}{c} & \textit{в інших випадках} \end{cases}$$

Варіант 9

$$Y=\begin{cases} ax^2+b^2x & \textit{при } x<0 \\ x-\dfrac{a}{x-c} & \textit{при } x>0 \\ 1+\dfrac{x}{c} & \textit{в інших випадках} \end{cases}$$

Варіант 10

$Y=\begin{cases} a-\dfrac{x}{10+b} & \text{при } x<0 \\ \dfrac{x-a}{x-c} & \text{при } x>0 \\ 3x+\dfrac{2}{c} & \text{в інших випадках} \end{cases}$	$Y=\begin{cases} ax^2-bx+c & \text{при } x<3 \\ \dfrac{x-a}{x-c} & \text{при } x>3 \\ \dfrac{x}{c} & \text{в інших випадках} \end{cases}$

Варіант 11

$$Y=\begin{cases} ax^2+\dfrac{b}{c} & \text{при } x<15 \\ \dfrac{x-a}{(x-c)^2} & \text{при } x>15 \\ \dfrac{x^2}{c^2} & \text{в інших випадках} \end{cases}$$

Варіант 12

$$Y=\begin{cases} ax^3+b^2+c & \text{при } x<0 \\ \dfrac{x-a}{x-c} & \text{при } x>0 \\ \dfrac{x}{c}+\dfrac{x}{a} & \text{в інших випадках} \end{cases}$$

Варіант 13

$$Y=\begin{cases} ax^2+b & \text{при } x-1<0 \\ \dfrac{x-a}{x} & \text{при } x-1>0 \\ \dfrac{x}{c} & \text{в інших випадках} \end{cases}$$

Варіант 14

$$Y=\begin{cases} -ax^3-b & \text{при } x+c<0 \\ \dfrac{x-a}{x-c} & \text{при } x+c>0 \\ \dfrac{x}{c}+\dfrac{c}{x} & \text{в інших випадках} \end{cases}$$

Варіант 15

$$Y=\begin{cases} -ax^2+b & \text{при } x<0 \\ \dfrac{x}{x-c}+5 & \text{при } x>0 \\ \dfrac{x}{-c} & \text{в інших випадках} \end{cases}$$

Варіант 16

$$Y=\begin{cases} a(x+c)^2-b & \text{при } x>0 \\ \dfrac{x-a}{-c} & \text{при } x<0 \\ a+\dfrac{x}{c} & \text{в інших випадках} \end{cases}$$

Варіант 17

$$Y=\begin{cases} ax^2-cx+b & \text{при } x+10<0 \\ \dfrac{x-a}{x-c} & \text{при } x+10>0 \\ \dfrac{-x}{a-c} & \text{в інших випадках} \end{cases}$$

Варіант 18

$$Y=\begin{cases} ax^3+bx^2 & \text{при } x<0 \\ \dfrac{x-a}{x-c} & \text{при } x>0 \\ \dfrac{x+5}{c(x-10)} & \text{в інших випадках} \end{cases}$$

Варіант 19

$$Y=\begin{cases} a(x+7)^2-b & \text{при } x<5 \\ \dfrac{x-cd}{ax} & \text{при } x>5 \\ \dfrac{x}{c} & \text{в інших випадках} \end{cases}$$

Варіант 20

$$Y=\begin{cases} -\dfrac{2x-c}{cx-a} & \text{при } x<10 \\ \dfrac{x-a}{x-c} & \text{при } x>10 \\ -\dfrac{x}{c}+\dfrac{-c}{2x} & \text{в інших випадках} \end{cases}$$

Лабораторна робота № 3.
Організація циклів.

Мета роботи: Ознайомитися з методами організації циклічних операцій в 3-х адресній навчальній машині

Обладнання: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBox Portable, емулятор навчальної машини ЕМ-3.

Завдання:

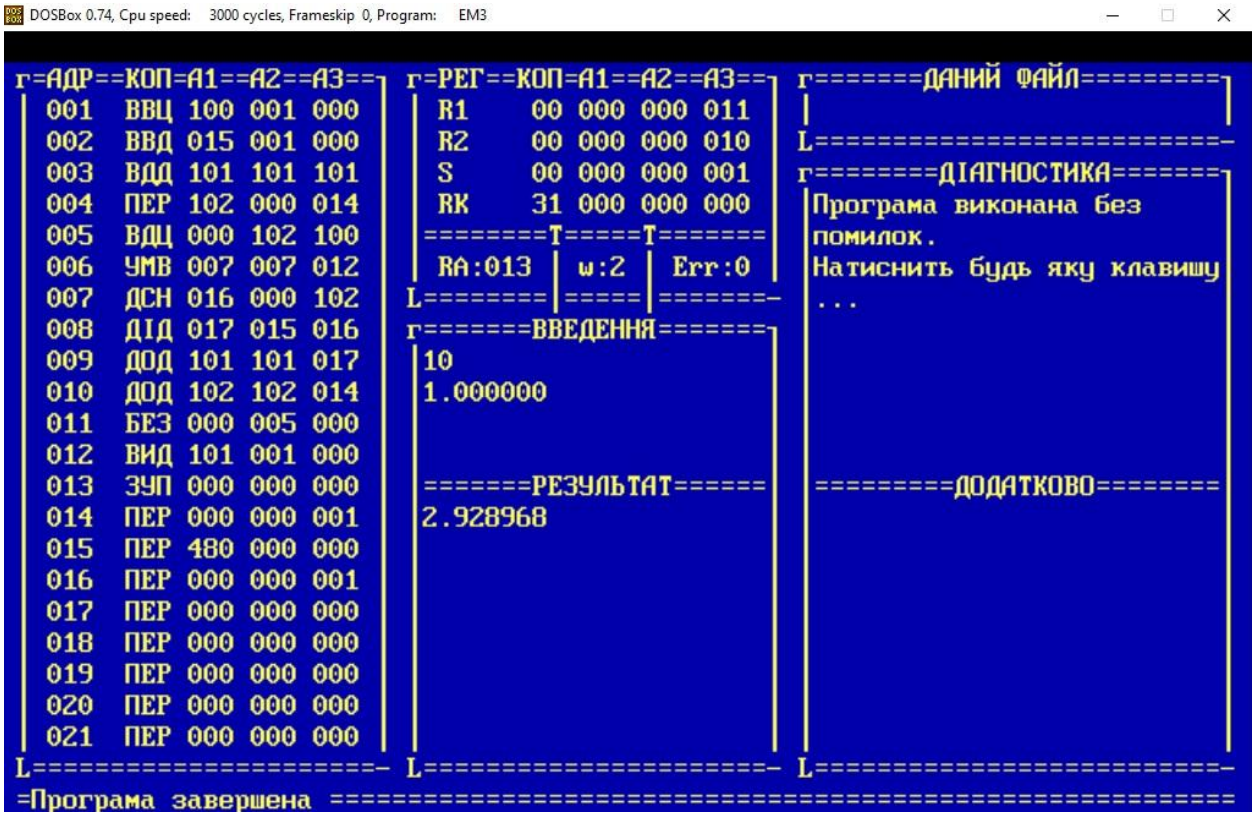
В якості наступного прикладу розглянемо програму для обчислення початкового відрізка гармонійного ряду:

$$y = \sum_{i=1}^{n} 1 / i$$

Для зберігання змінних *n, y* і *i* виділимо комірки 100, 101 і 102 відповідно. У цьому алгоритмі ми реалізуємо цикл з передумовою, тому при введенні $n<1$ тіло циклу не буде виконуватися жодного разу, і наша програма буде видавати нульовий результат. Нижче наведена можлива програма для вирішення цього завдання.

Зробимо деякі зауваження до цієї програми. У нашій мові у нас немає команди ділення цілого числа на дійсне, тому при обчисленні величини 1.0 / *i* нам довелося окремою командою ДСН 000 000 102 перетворити значення цілої змінної *i* в дійсної значення (код команди 20). Зверніть також увагу, що для нашої навчальної машини ми ще не визначили формат представлення дійсних чисел, тому в комірці з адресою 14 варто поки просто умовне позначення константи 1.0, а не її машинне відображення.

```
№    Команда        Коментарі
001 ВВЦ 100 001 000 Read(n)
002 ВВД 015 001 000 Введення дійсн. 1 в <015>
003 ВДД 101 101 101 y = 0.0
004 ПЕР 102 000 014 i = 1
005 ВДЦ 000 102 100 i = i-n; формування w
006 УМВ 007 007 012 If i>n then goto 012
007 ДСН 016 000 102 Переведення i в дійсн. Та запис в <016>
008 ДІД 017 015 016 <017> = 1.0/<016>
009 ДОД 101 101 017 y = y+<017>
010 ДОЦ 102 102 014 i = i+1
011 БЕЗ 000 005 000 Наступна ітерация циклу
012 ВИД 101 001 000 Write(y)
013 ЗУП 000 000 000 Стоп
014 00  000 000 001
                    Ціла константа 1
015 ??? 480 000 000 Тут записуємо дійсну 1.0
```



Завдання для домашньої підготовки.

Вказівки до змісту звіту

1. Текст програми. Кожен рядок програми супроводити коментарями.
2. Результат роботи програми (скріншот виконаної програми).

Варіанти для лабораторної роботи №3

Кількість ітерацій циклу n=10

Варіанти

1. $\sum_{n=1}^{10} 5(n+2)$	6. $\sum_{n=1}^{10} 8(12-n)$	11. $\sum_{n=1}^{10} 3n^2$
2. $\sum_{n=1}^{10} 2n^2$	7. $\sum_{n=1}^{10} 4(2n+1)$	12. $\sum_{n=1}^{10} 7(n+1)$
3. $\sum_{n=1}^{10} 4n^2$	8. $\sum_{n=1}^{10} (8n-1)$	13. $\sum_{n=1}^{10} (110-n^2)$
4. $\sum_{n=1}^{10} (12-n)^2$	9. $\sum_{n=1}^{10} (6n^2-5)$	14. $\sum_{n=1}^{10} (n^2-1)$
5. $\sum_{n=1}^{10} (2+n)^2$	10. $\sum_{n=1}^{10} (11-n)^2$	15. $\sum_{n=1}^{10} 5(21-n)$

Лабораторна робота № 4.
Обробка масивів.

Мета роботи: Ознайомитися з методами роботи з масивами в 3-х адресній навчальній машині

Обладнання: персональний комп'ютер.

Програмне забезпечення: Windows 7/8/10, DOSBox Portable, емулятор навчальної машини ЕМ-3.

Завдання:

Нехай потрібно написати програму для введення масиву x з 100 дійсних чисел і обчислення суми всіх елементів цього масиву:

$$S = \sum_{i=1}^{n} x[i]$$

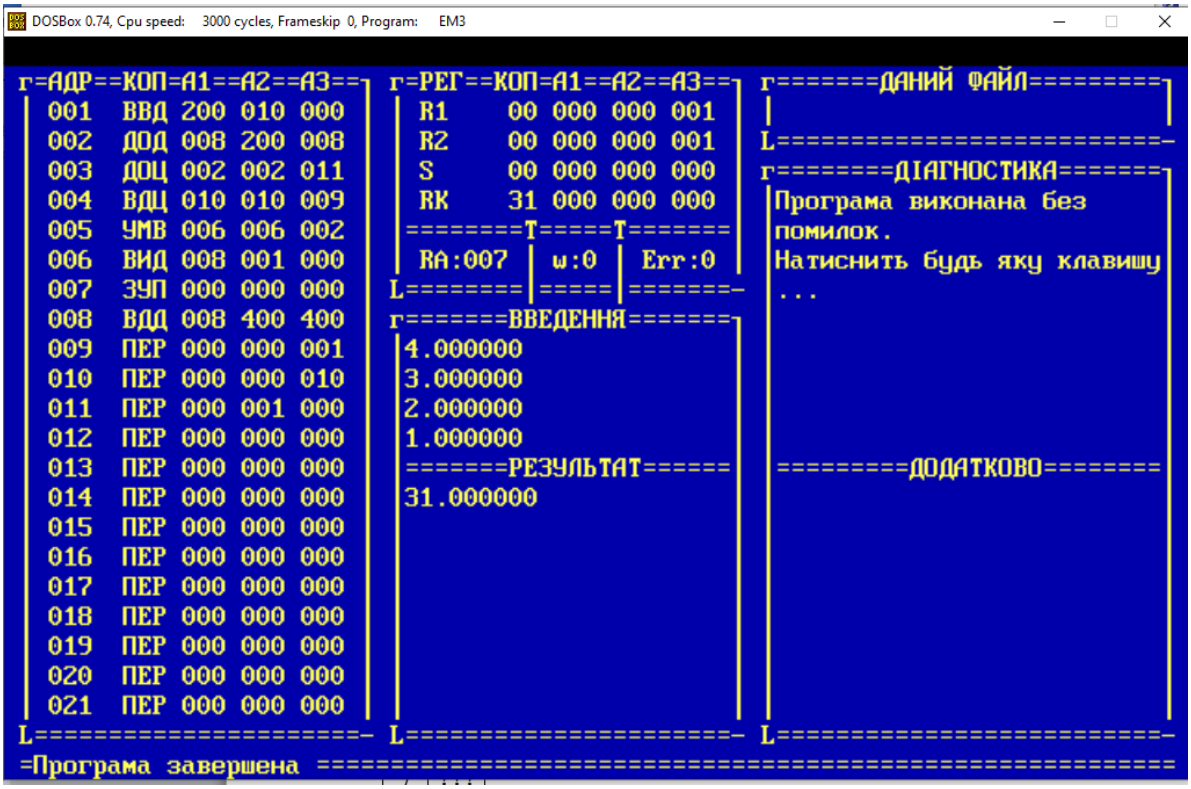
Будемо припускати, що довжина програми не перевищує 200 осередків, і помістимо масив x, починаючи з 200-ої комірки пам'яті. Дійсну змінну S з початковим значенням 0.0 і цілу змінну i з початковим значенням 100 розмістимо в кінці тексту програми. На рис. наведено текст цієї програми.

```
№       Команда                  Коментарі
001 ВВД 20  01  00  Введення елементів масиву xᵢ в
002 ДОД 00  20  00  S = S+x[i]
003 ДОЦ 00  00  01  Модифікація команди в осередці 2
004 ВДЦ 01  01  00  n := n-1
005 УМВ 00  00  00  Наступна ітерация цикла
006 ВИД 00  00  00  Write(S)
007 ЗУП 00  00  00  Стоп
008 ВДД  00  40  40 Дійсна змінна S = 0.0
009 00   00  00  00  Ціла константа 1
010 00   00  00  01  Змінна n з початковим значенням
011 00   00  00  00  Константа переадресування
```

Вже згадана програма виділяється своїм новим прийомом програмування і може бути названа самомодіфіцірующейся програмою. Звернемо увагу на третій рядок програми. Міститься в ній команда змінює вихідний код програми (команду в осередку 2) для організації циклу перебору елементів масиву. Модифікована команда розглядається як ціле число, яке складається зі спеціально підібране константою переадресації. Відповідно до одного з принципів фон Неймана, числа і команди в навчальній машині не відрізняються один від одного, а, значить, змінюючи числове уявлення команди, ми можемо змінювати і її суть.

У такого методу програмування є один істотний недолік: модифікація коду програми всередині її самої може привести до плутанини і викликати появу помилок. Крім того, самомодіфіцірующуюся програму важко розуміти і вносити в неї зміни. У нашій навчальній машині це, однак, єдиний спосіб

обробки масивів. В інших архітектурах ЕОМ є й інші, більш ефективні способи роботи з масивами, тому метод з модифікацією команд не використовується.



Завдання для домашньої підготовки.

Вказівки до змісту звіту

1. Текст програми. Кожен рядок програми супроводити коментарями.
2. Результат роботи програми (скріншот виконаної програми).

Варіанти для лабораторної роботи №4

n= номеру варіанту, якщо номер варіанту > 8 або n = (номер варіанту + 5) в іншому випадку.

Варіант 1

В одновимірному масиві, що складається з n дійсних елементів, обчислити суму від'ємних елементів масиву.

Варіант 2

В одновимірному масиві, що складається з n дійсних елементів, обчислити суму додатних елементів масиву.

Варіант 3

В одновимірному масиві, що складається з n цілих елементів, обчислити добуток елементів масиву.

Варіант 4

В одновимірному масиві, що складається з n цілих елементів, обчислити добуток додатних елементів масиву.

Варіант 5

В одновимірному масиві, що складається з n цілих елементів, обчислити добуток від'ємних елементів масиву.

Варіант 6

В одновимірному масиві, що складається з n дійсних елементів, обчислити кількість елементів масиву, рівних 0.

Варіант 7

В одновимірному масиві, що складається з n дійсних елементів, обчислити кількість елементів масиву, великих з (вводиться з клавіатури).

Варіант 8

В одновимірному масиві, що складається з n дійсних елементів, обчислити кількість негативних елементів масиву.

Варіант 9

В одновимірному масиві, що складається з n цілих елементів, обчислити кількість позитивних елементів масиву.

Варіант 10

В одновимірному масиві, що складається з n дійсних елементів, обчислити кількість елементів масиву, менших з (вводиться з клавіатури).

Варіант 11

В одновимірному масиві, що складається з n дійсних елементів, обчислити суму елементів масиву, менших з (вводиться з клавіатури).

Варіант 12

В одновимірному масиві, що складається з n дійсних елементів, обчислити добуток елементів масиву, менших з (вводиться з клавіатури).

Варіант 13

В одновимірному масиві, що складається з n дійсних елементів, обчислити суму елементів масиву, великих з (вводиться з клавіатури).

Варіант 14

В одновимірному масиві, що складається з n дійсних елементів, обчислити добуток елементів масиву, великих з (вводиться з клавіатури).

Варіант 15

В одновимірному масиві, що складається з n дійсних елементів, обчислити суму перших K (вводиться з клавіатури) елементів масиву.

Варіант 16

В одновимірному масиві, що складається з n дійсних елементів, обчислити суму останніх K (вводиться з клавіатури) елементів масиву.

ЗМІСТ

Навчальні машини. Короткі теоретичні відомості. 3

Лабораторна робота № 1. Створення лінійних програм. 15

Лабораторна робота № 2. Реалізація умовного переходу. 17

Лабораторна робота № 3. Організація циклів. 21

Лабораторна робота № 4. Обробка масивів. 23