

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**



**КАФЕДРА ЕЛЕКТРОННИХ ОБЧИСЛЮВАЛЬНИХ МАШИН**

## **МЕТОДИЧНІ ВКАЗІВКИ**

**до циклу лабораторних робіт  
з дисципліни  
“Архітектура комп’ютерів”  
для студентів базового напрямку 6.050102  
“Комп’ютерна інженерія”.**

*Затверджено  
на засідання кафедри  
“Електронні обчислювальні машини”.*

*Протокол №\_\_\_\_\_від.....2024 р.*

Львів – 2024

Методичні вказівки до циклу лабораторних робіт з дисципліни "Архітектура комп'ютера" для студентів базового напрямку 6.050102 "Комп'ютерна інженерія" /Укл.: Клименко В.А., Бойко Г.В. Кушнір Д.О.- Львів: Видавництво Національного університету "Львівська політехніка", 2024 – 44 с.

Укладачі: Мельник А.О., док. техн. наук., проф.

Клименко В.А., асистент  
Бойко Г.В., асистент  
Кушнір Д.О., д.ф., асистент

Відповідальний за випуск Кушнір Д.О., д.ф., асистент.

Рецензент: Дунець Р.Б., проф. док. техн. наук., завідувач кафедри ЕОМ,

# ЗМІСТ

<b>ЛР № 1. Робота з симулятором машини Ноймана.Дослідження виконання машинного коду в автоматичному режимі. ....</b>	<b>4</b>
Методика виконання лабораторної роботи.....	4
Варіанти завдань на лабораторну роботу №2 .....	11
<b>ЛР № 2. Дослідження макроалгоритмів та.....</b>	<b>12</b>
<b>мікроалгоритмів виконання машинних інструкцій. ....</b>	<b>12</b>
Методика виконання лабораторної роботи.....	12
Варіанти завдань на лабораторну роботу №2 .....	18
<b>ЛР № 3. Робота з симулятором машини Ноймана.Дослідження виконання асемблерної програми симулятора.....</b>	<b>23</b>
Теоретичні відомості.....	23
Хід виконання роботи: .....	31
assemble prog1.as prog1.mc.....	31
simulate.exe prog1.mc>result.txt.....	31
Завдання .....	31
<b>ЛР № 4. Робота з симулятором машини Ноймана.Дослідження архітектури системи команд. ....</b>	<b>32</b>
Теоретичні відомості.....	32
Хід виконання роботи: .....	34
Варіанти завдань.....	34
<b>ЛР № 05. Робота з симулятором MARIE.....</b>	<b>34</b>
Методика виконання лабораторної роботи.....	35
Варіанти завдань.....	39
<b>ЛР № 6. Дослідження виконання інструкцій симулятораMARIE за допомогою DataPath.....</b>	<b>40</b>
Методика виконання лабораторної роботи.....	40
Завдання .....	41
<b>Додаток до лабораторних робіт 4-5. ....</b>	<b>42</b>
Директиви асемблера.....	42
Операнди .....	42
<b>ЛР № 7. Дослідження виконання циклів на конвеєрі інструкцій.....</b>	<b>43</b>
Методика виконання лабораторної роботи.....	43
Варіанти завдань.....	58
<b>ЛР № 8. Дослідження виконання циклів на конвеєрі інструкцій.....</b>	<b>59</b>
Методика виконання лабораторної роботи.....	59
Варіанти завдань.....	65

# ЛР № 1. Робота з симулятором машини Ноймана. Дослідження виконання машинного коду в автоматичному режимі.

**Мета:** опанувати роботу на симуляторі машини Ноймана, зрозуміти і дослідити принцип виконання програми машиною Ноймана.

**Завдання:** запустити симулятор, увести до нього коди машинних інструкцій і коди чисел, навчитися утворювати і змінювати ці коди, дослідити і пояснити принципи трактування машиною Ноймана бінарних кодів. Розробити тестову програму, завантажити програму і відповідні дані до симулятора, виконати програму в автоматичному режимі, проаналізувати і пояснити отримані результати, скласти звіт з виконання лабораторних досліджень та захистити його.

## Методика виконання лабораторної роботи

*CISC – complex instruction set computing (обчислення зі складною системою машинних інструкцій)*

Аби дослідити дію машини Ноймана використовують симулятори цієї машини, що, в свою чергу, є готовими до використання комп'ютерними програмами. При роботі з такою програмою складається враження роботи з комп'ютером першої генерації, що приймає дані і подає результати і двійковий (бінарний) системі числення, має обмежений обсяг пам'яті, обмежене число регістрів і відсутню операційну систему. Коли в комп'ютері (в нас – в симуляторі комп'ютера) ОС нема, тоді кожен програму треба завершувати виконанням машинної інструкції СТОП. В автоматичному режимі одним натиском на клавішу комп'ютер змушують виконати уведено до нього програму - повністю, від першої до останньої машинної інструкції програми. Розглянемо конкретний симулятор машини Ноймана, якому автор надав назву «Кроха».

«Кроха» є DOS програмою, яку запускають у вікні DOS. Отже, миша з симулятором не працює так само, як і в ті давні часи, коли реалізували машину Ноймана. Треба користуватися курсором (рискою підкреслення) і клавішами пересування курсора. Але курсор діє лише у вікні пам'яті. Залишаються можливості керування симулятором натисканням вибраних клавіш апаратури. Таке керування є примітивним але таким, що дозволяє правдиво і чітко зрозуміти, як працює комп'ютер Ноймана, як з ним колись працювали, а також які в нього є принципові недоліки. Наступний рисунок 1 містить копію вікна симулятора Кроха. Симулятор є DOS програмою, що викликається в режимі командного рядка.

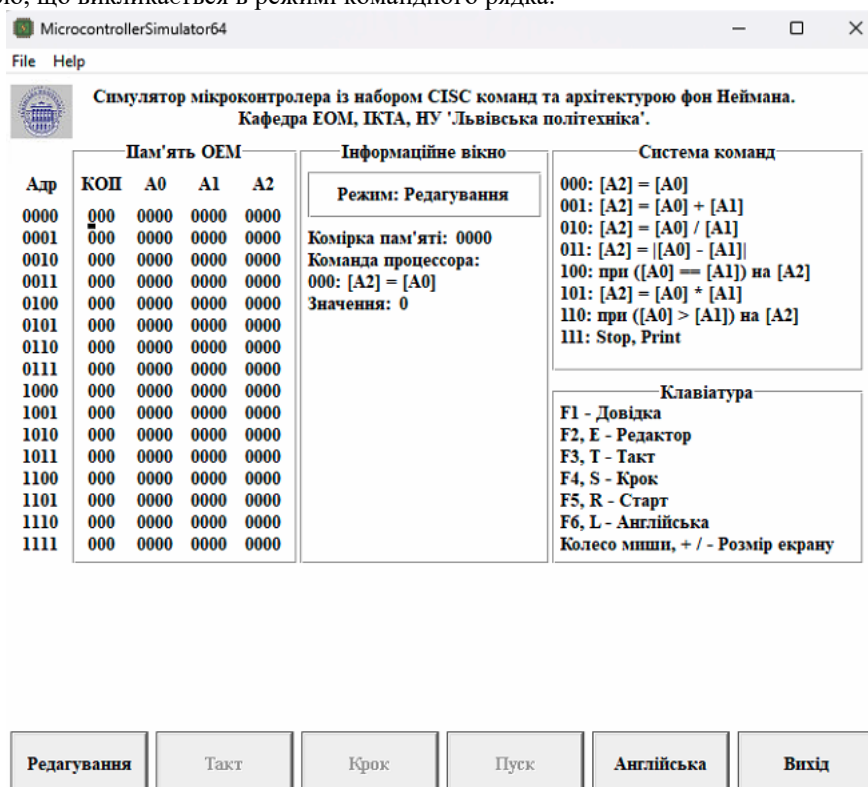


Рис. 1.1. Вікно навчального комп'ютера «Кроха-М»

Зліва розташоване вікно пам'яті. Тут позначені адреси в бінарному коді, від  $0000_2$  до  $1111_2$  (всього 16 комірок). Вмістимо кожної комірки – це  $3+4+4+4=15$  бітова структура, що може бути або бінарним кодом цілого чила, або бінарним кодом машинної команди (так званої інструкції). Отже маємо не зовсім звичний 15-бітовий комп'ютер, що не підтримує парадигму байта.

#### Формат машинної інструкції машини Ноймана

В комп'ютері використаний наступний класичний формат 3-адресної машинної інструкції:

Поле коду операції. Довжина 3 біти	Поле адреси 1-го операнда. Довжина 4 біти	Поле адреси 2-го операнда. Довжина 4 біти	Поле адреси результату. Довжина 4 біти
<b>КОп</b>	<b>A1</b>	<b>A2</b>	<b>A3</b>

Приклад кодування інструкції (A1) + (A2) → A3 :			
000	0100	0101	0110

Рис. 1.2 – Формат машинної інструкції

### Формат числа машини Ноймана

Симулятор використовує наступний єдиний формат даних:

Старший розряд	13 середніх розрядів бінарного коду	Молодший розряд
1	00000000001	1
Приклад кодування позитивного десяткового числа : $16384_{10} + 3_{10} = 16387_{10}$		

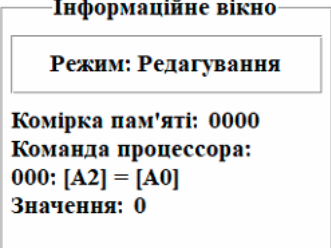
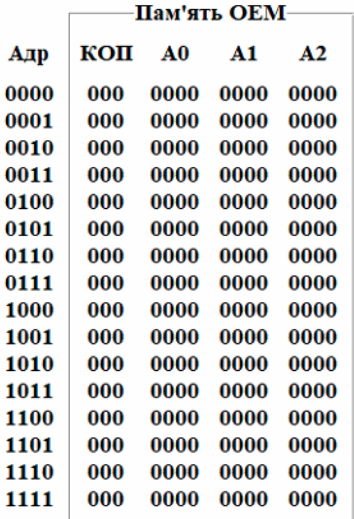
Рис. 1.3 – Формат машинного числа

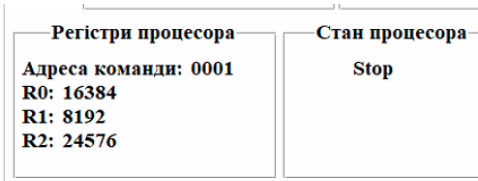
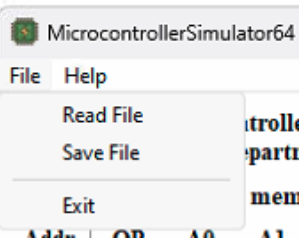
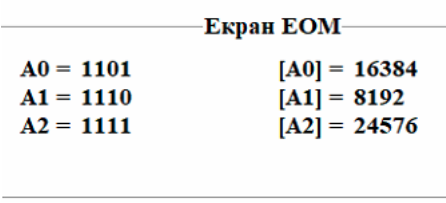
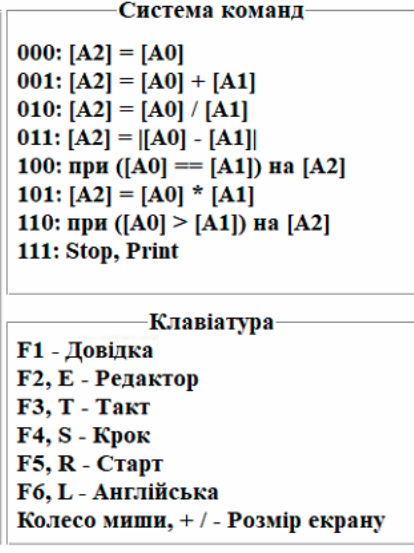
Негативні числа симулятор комп'ютера не опрацьовує, а всі кодові комбінації трактує як коди натуральних чисел і нуля. Максимальним є число з кодом  $111\ 1111\ 1111\ 1111_2 = 32768_{10} - 1 = 32767_{10}$ , а мінімальним числом є число з бінарним кодом  $000\ 0000\ 0000\ 0000_2 =$  тобто нуль.

### Керування роботою з симулятором

Далі розглянемо вікно симулятора комп'ютера (табл. 1) та правила роботи з ним. Зауважимо, що симулятор запускають в командному рядку, отже, з мишою він не працює. Але є Windows симулятор цього комп'ютера, що сприймає мишу.

Табл. 1 – Управління симулятором

Клавіші керування комп'ютером	Дія клавіш	Ілюстрація
Інформаційне вікно  Негайно задати АВТО: <b>F1, A</b> Негайно задати РЕДАКТОР: <b>F2, E</b> Негайно задати ШАГ: <b>F3, S</b> Негайно задати ТАКТ: <b>F4, T</b> Вибирати режим безпосередньо курсором: <b>F5, R</b>	Вибір і встановлення режиму роботи. В режимі редактора можна змінювати вміст комірок пам'яті, суматора (СМ), лічильника інструкцій (СК), регістра інструкції (РК). В режимі АВТО одним натиском на клавішу ENTER змушують комп'ютер виконати всю програму. В режимі ШАГ одним натиском на клавішу ENTER змушують комп'ютер виконати чергову інструкцію програми. В режимі ТАКТ одним натиском на клавішу ENTER змушують комп'ютер виконати чергову мікродію виконання чергової інструкції програми. Зауважимо, що виконання кожної інструкції розкладають на виконання певної послідовності мікрокроків, кожний з яких називають мікродією (або мікрокомандою).	
Вікно пам'яті	Бінарні адреси пам'яті не змінюються. Вміст комірок пам'яті (бінарні коди) змінюються. Для цього в режимі редагування потрібно підвести курсор під біт, що змінюється та натиснути 0 або 1. Показує:	

Вікно арифметичного та логічного пристрою (АЛП)	<ul style="list-style-type: none"> <li>код виконуваної інструкції, що зберігає регістр інструкції (R0, R1);</li> <li>вмістиме суматора (R2);</li> <li>Адреса команди, яка буде йти на пристрій керування</li> </ul>	 <p>Регістри процесора Адреса команди: 0001 R0: 16384 R1: 8192 R2: 24576</p> <p>Стан процесора Stop</p>
Вікно збереження чи завантаження результатів програми	Дозволяє швидко зберігати результати програми, та завантажувати їх при наступному відкритті програми. Це дозволяє оминати проблеми з повторним вводом даних.	 <p>Microcontroller Simulator64 File Help Read File Save File Exit</p>
Екранне вікно для візуалізації значень операндів і результатів обчислень	Показує десяткові значення вмістимого комірок пам'яті за адресами A1, A2 та A3 інструкції СТОП (Виведення), бінарний код операції якої є 111 <sub>2</sub> . Отже, коли за одною з трьох вказаних адрес розміщено інструкцію ми побачимо її як якесь число. Це на перший погляд є дивною, але принципово важливою рисою машини Ноймана, якою є ПК та багато інших комп'ютерів.	 <p>Екран ЕОМ</p> <p>A0 = 1101 [A0] = 16384 A1 = 1110 [A1] = 8192 A2 = 1111 [A2] = 24576</p>
Довідкове вікно	Вікно містить всі дозволені коди операцій, а також правила маніпулювання з режимами симулятора за допомогою клавіатури. Всього дозволено виконання вісьми операцій (пересилання вмісту комірки пам'яті з адресою A1 до комірки пам'яті з адресою A3 (код 000 <sub>2</sub> ), додавання (001 <sub>2</sub> ), ділення вмісту A1 на вміст A2 (код 010 <sub>2</sub> ), обчислення модуля різниці вмісту A1 і A2 з записом результату до A3 (код 011 <sub>2</sub> ), умовного переходу (коли вміст A1 і A2 є рівними, тоді наступною виконують інструкцію не з наступної за чергою адресою, а з адреси A3 (код 100 <sub>2</sub> ), множення вмісту A1 на вміст A2 з записом добутку до A3 (код 101 <sub>2</sub> , ще одного умовного переходу [коли вміст A1 є більшим від вмісту A2, тоді наступно виконуваною є інструкція з комірки з адресою A3] (код 110 <sub>2</sub> ) та завершення обчислень і виведення до дисплейного вікна вмістимого комірок пам'яті з адресами A1, A2, A3.	 <p>Система команд</p> <p>000: [A2] = [A0] 001: [A2] = [A0] + [A1] 010: [A2] = [A0] / [A1] 011: [A2] =  [A0] - [A1]  100: при ([A0] == [A1]) на [A2] 101: [A2] = [A0] * [A1] 110: при ([A0] &gt; [A1]) на [A2] 111: Stop, Print</p> <p>Клавіатура</p> <p>F1 - Довідка F2, E - Редактор F3, T - Такт F4, S - Крок F5, R - Старт F6, L - Англійська Колесо миши, + / - Розмір екрану</p>

Далі подамо стан комп'ютера за умови, що не усі комірки пам'яті містять нульовий бінарний код (рис. 1.4).

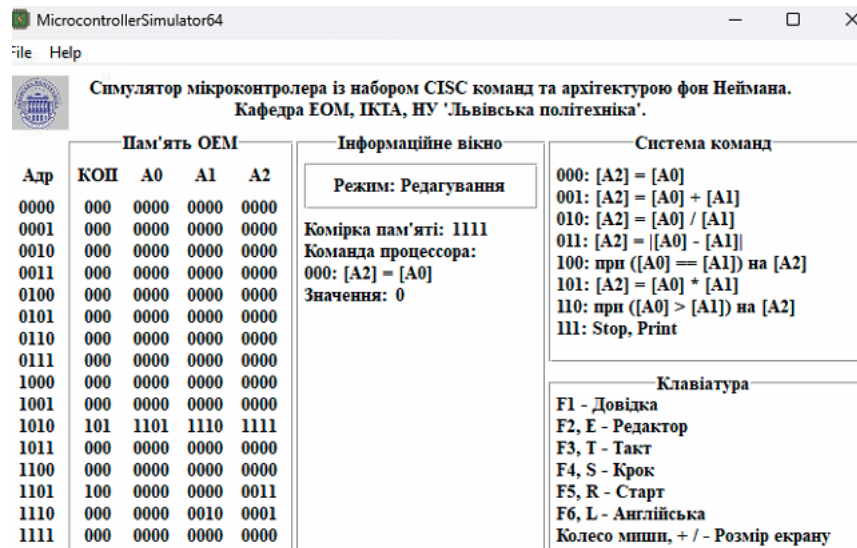


Рис. 1.2. Стан симулятора з ненульовою пам'яттю

Запишемо до комірки пам'яті з адресою 1010<sub>2</sub> код 101 1101 1110 1111<sub>2</sub>, а до комірок те, що подане наступною таблицею 2:

Таблиця 2. Бінарний код

1010 <sub>2</sub>	101 1101 1110 1111 <sub>2</sub> = інструкція A1 * A2 => A3
1101 <sub>2</sub>	100 0000 0000 0011 <sub>2</sub> = 16384 <sub>10</sub> + 3 <sub>10</sub> = 16387 <sub>10</sub>
1110 <sub>2</sub>	000 0000 0010 0001 <sub>2</sub> = 32 <sub>10</sub> + 1 <sub>10</sub> = 33 <sub>10</sub>
1111 <sub>2</sub>	000 0000 0000 0000 <sub>2</sub> = 0 <sub>10</sub>

Далі, коли виставити курсор на лінію вмістимого комірки з адресою 1010<sub>2</sub> (курсор не можна побачити на поданому рисунку, але його розташування легко побачити у вікні реальної програми), тоді в інформаційному вікні можна побачити, як ці записи трактує комп'ютер.

*По-перше*, бінарний код комірки, на якій знаходиться курсор, він сприймає як код інструкції та розшифровує його як інструкцію множення вмістимого комірки за першою адресою A1=1101<sub>2</sub> на вмістиме комірки за другою адресою A2=1110<sub>2</sub>. Така інструкція має поміщати добуток до комарки з третьою адресою A3=1111<sub>2</sub>. Але тут знаходиться не добуток, а нульовий бінарний код, який комп'ютер сприймає як десятковий нуль.

*По-друге*, коли бінарний код за адресою 1010<sub>2</sub> протрактовано як інструкцію, тоді комп'ютер вже мусить трактувати вмістиме адрес 1101<sub>2</sub> - 1111<sub>2</sub> як бінарні коди цілих позитивних чисел, адже ці адреси належать щойно протрактованій інструкції. Нульовий код добутку пояснити нескладно: ця інструкція ще не виконувалася і комірка результату містить сміття (в нас це є нуль).

*По-третьє*, вмістиме кожної комірки пам'яті, на якій стоїть курсор в режимі редагування, комп'ютер завжди трактує як машинну інструкцію. Поглянемо на рис. 1.5.

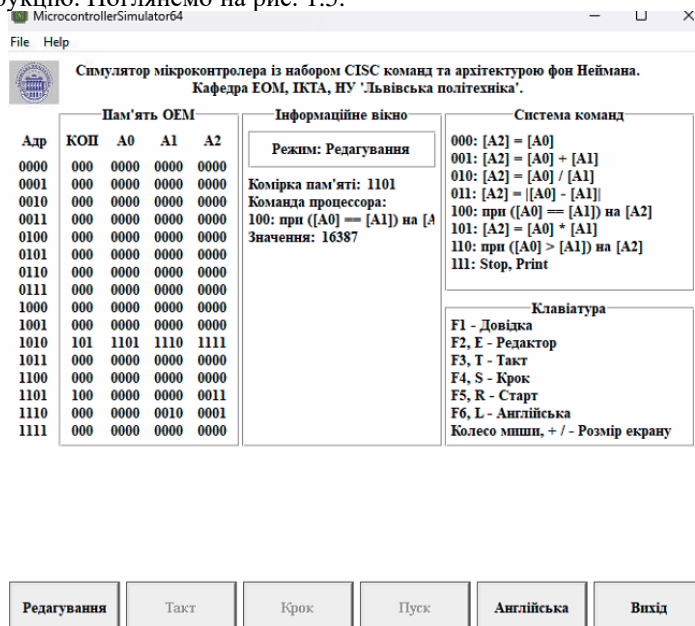


Рис. 1.3. Вікно для стану комп'ютера, коли курсор в режимі редагування розташований на вмістимому комірки пам'яті



з адресою 1101<sub>2</sub>.

Нічого не мінялося. Ми лише змусили комп'ютер перейти від трактування вмістимого комірки 1010<sub>2</sub> до трактування вмістимого комірки 1101<sub>2</sub>. А результат побачили разючий! Те вмістиме, що трактувалося числом, почало трактуватися як машинна інструкція.

Але все коректно. Ось такими і є сучасні комп'ютери. В межах пам'яті неможливо відрізнити код числа від коду інструкції. А розрізнити потрібно. Так ось, розрізнення відбувається вже поза пам'яттю. Бінарний код, що з пам'яті пристрій керування надсилає до суматора стає числом, а той код, що той самий пристрій надсилає до регістра інструкції (РК), стає інструкцією. Пристрій керування приймає рішення щодо напрямку надсилання, керуючись і уведеною до пам'яті програмою, і вказівкою, з якої комірки пам'яті розпочинається програмний код, за умови, що інструкції програми розташовані в пам'яті в комірках з послідовними адресами ( $n$ ,  $n+1$ ,  $n+2$ ,  $n+3$ , ...). В цьому комп'ютері прийнято, що перша інструкція програми завжди знаходиться в комірці з адресою 0000<sub>2</sub>. Якщо програміст цього обмеження не дотримується, тоді уведена ним до пам'яті програма не надає коректні результати.

### Про машину Ноймана

Структуру машини Ноймана [Johann von Neumann (Будапешт. 1903 р.н. і Німеччина), John von Neumann (USA), Джон Нейман (Росія)] містить рис. 1.6.

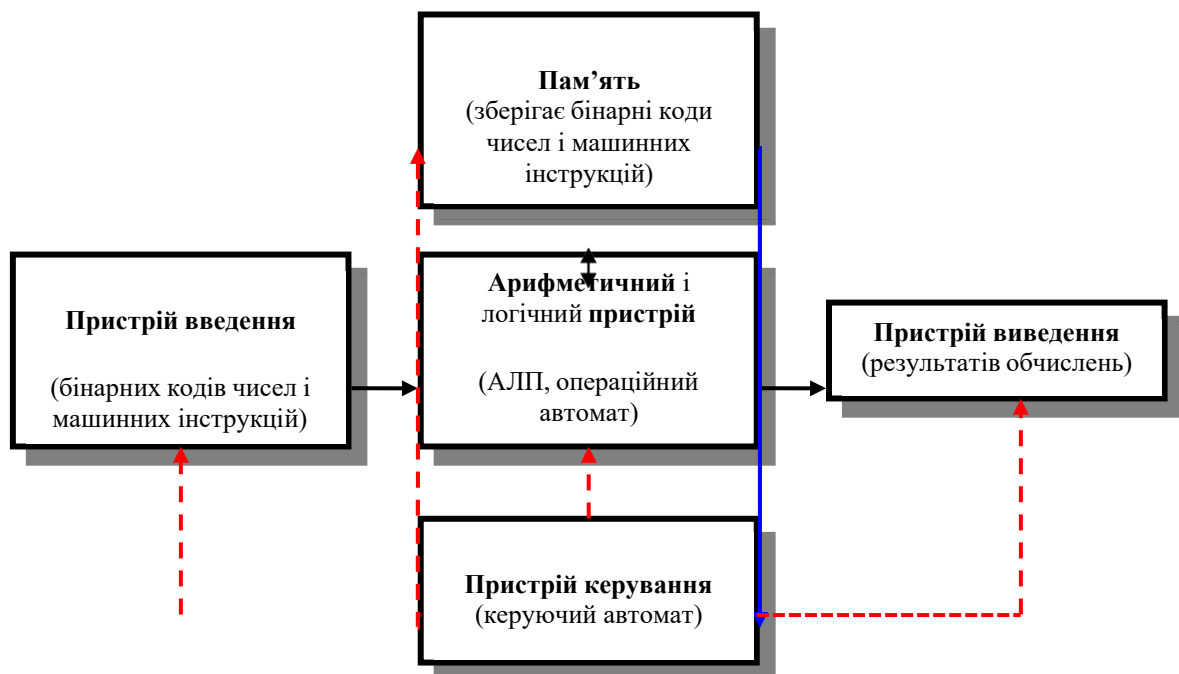


Рис. 1.4. Структура машини Ноймана, запропонована в 1945 році в Гарвардському університеті США. Саме цю (приховану від наших очей) структуру має комп'ютер «Кроха».

За допомогою пристрою введення бінарні коди чисел і машинних інструкцій потрапляють (чорні лінії) транзитом через АЛП до пам'яті, де і зберігаються під час автоматичного виконання обчислень. Отже, під час введення АЛП зайнято транзитним пересиланням і комп'ютер не може обчислювати. Коли введення завершено, тоді по команді старту, що надає людина, комп'ютер починає виконувати заведену людиною до пам'яті програму автоматично (без участі людини). При цьому бінарні коди з пам'яті, що надсилаються (синя лінія) до пристрою керування, сприймаються як машинні інструкції. На основі прийнятих з пам'яті кодів машинних інструкцій пристрій керування керує роботою всіх інших пристроїв комп'ютера (червоні штрихові лінії).

Бінарні коди, що надсилаються з пам'яті до АЛП (чорні лінії), сприймаються як коди чисел, що приймають участь в формуванні результату. Можливе збереження проміжних результатів обчислень в тій самій пам'яті чорна двохнаправлена лінія). Остаточні результати також спочатку надсилають до пам'яті, а вже потім виводять для користування людиною через пристрій виводу (чорна лінія).

До впровадження багатопроцесорності всі персональні комп'ютери (навіть з процесором Пентіум 4) були класичними машинами Ноймана. Отже, з 1945 року до 2000 року ПК незмінну ідею, але змінювалися при цьому технологічно.

Розглянемо стан симулятора, що поданий рисунком 2.1.

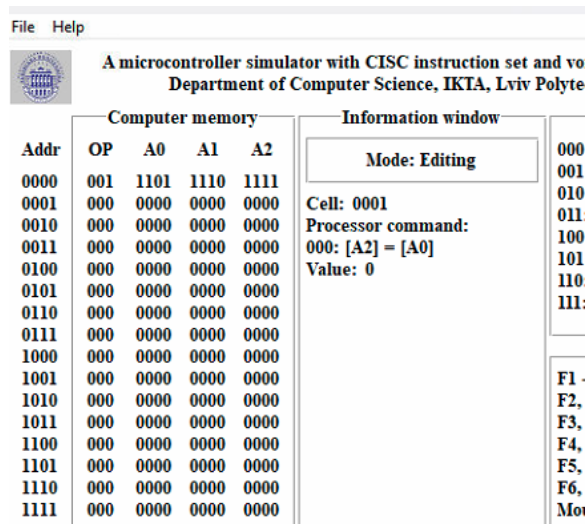


Рис. 1.5. – Стан симулятора після введення першої інструкції додавання [ + A1 A2 A3 чи 001 1101 1110 1111, тобто, додати вміст комір пам'яті A1 і A2, а результат записати до комірки A3] до комірки пам'яті з нульовою адресою, де має розташовуватися перша команда програми.

Інформаційне вікно повідомляє, що уведений до комірки з нульовою адресою код можна трактувати як десяткове число 7663 або як команду A1 + A2 =Ю A3. Проте вміст лічильника інструкцій СК є нульовим (0000<sub>2</sub>). З цього випливає, що симулятор трактує вміст комірки з нульовою адресою як команду, а не як число. Курсор розташований на першій лінії пам'яті, але його не видно на рисунку. Ясно, що саме позиція курсора наказує симулятору вибирати конкретний рядок (вміст відповідної комірки пам'яті) для інтерпретації

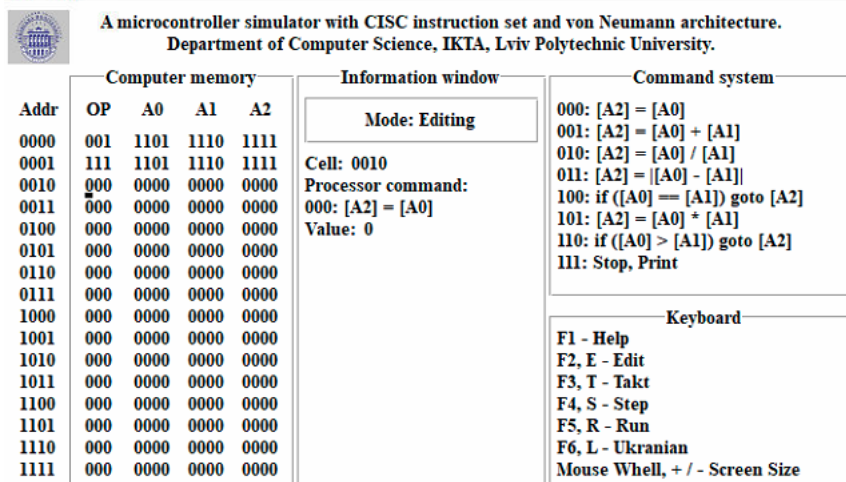


Рис. 1.6 – Стан симулятора після введення другої інструкції СТОП, ВИВЕДЕННЯ [ СТОП/ВИБ A1 A2 A3 чи 111 1101 1110 1111] до комірки пам'яті з першою (0001<sub>2</sub>) адресою, де розташовували другу команду програми.

Інформаційне вікно повідомляє, що уведений до комірки з першою адресою код можна трактувати як десяткове число 32239 або як команду СТОП, ВИВЕДЕННЯ. Але після виконання першої інструкції додавання вміст лічильника інструкцій (СК), що було нуль, збільшиться на одиницю (кажуть – інкрементується) і стане дорівнювати одиниці. В цей часовий момент СК змусить симулятор трактувати вміст (код) першої комірки пам'яті як інструкцію, а не як число. Отже, після виконання першої інструкції з нульової комірки пам'яті, виконається друга (і остання) інструкція програми, а саме – СТОП/ВИВЕДЕННЯ. В результаті в екранному вікні ми побачимо подані в десятковій формі доданки і суму. Тут курсор розташований на другій лінії пам'яті, але його не видно на рисунку.

Отже, до симулятора вручну уведена коротка програма, що складена двома інструкціями (рис. 2.3):

Адреса пам'яті	Машинна інструкція		Функція інструкції
	Мнемонічний код	Бінарний код	
0000 <sub>2</sub>	+ A1 A2 A3	001 1101 1110 1111 <sub>2</sub>	Додати
0001 <sub>2</sub>	СТОП A1 A2 A3	111 1101 1110 1111 <sub>2</sub>	Зупинитися і показати

Рис. 2.3 – Перша програма симулятора машини Ноймана, що складена двома машинними інструкціями


Проте уведена програма ще не має даних. Якщо казати точно, то ці дані мають розміщатися в комірках пам'яті з адресами 1101<sub>2</sub>= 13<sub>10</sub> та 1110<sub>2</sub>=14<sub>10</sub>. Ці комірки мають нульове наповнення, отже містять коди нуля. Ясно, що і сума повинна бути нульовою. Так і є, але суто випадково. До введення даних і до виконання програми вміст трьох останніх комірок пам'яті треба розглядати як «сміття», бо вона ані нами, ані виконуваною програмою не визначено. Просто в комірках не може бути «нічого». Там завжди «живуть» одиниці і нулі.

Уведено доданки, наприклад  $17_{10}$  і  $23_{10}$  до комірок з адресами  $13_{10}$  і  $14_{10}$  відповідно. Отримаємо стан симулятора, як він поданий на рис. 2.3.

Computer memory					Information window	Command system
Addr	OP	A0	A1	A2	Mode: Editing	000: [A2] = [A0] 001: [A2] = [A0] + [A1] 010: [A2] = [A0] / [A1] 011: [A2] =  [A0] - [A1]  100: if ([A0] == [A1]) goto [A2] 101: [A2] = [A0] * [A1] 110: if ([A0] > [A1]) goto [A2] 111: Stop, Print
0000	001	1101	1110	1111	Cell: 1111 Processor command: 000: [A2] = [A0] Value: 0	
0001	111	1101	1110	1111		
0010	000	0000	0000	0000		
0011	000	0000	0000	0000		
0100	000	0000	0000	0000		
0101	000	0000	0000	0000		
0110	000	0000	0000	0000		
0111	000	0000	0000	0000		
1000	000	0000	0000	0000		
1001	000	0000	0000	0000		
1010	000	0000	0000	0000		
1011	000	0000	0000	0000		
1100	000	0000	0000	0000		
1101	000	0000	0001	0001		
1110	000	0000	0001	0111		
1111	000	0000	0000	0000		
						Keyboard
						F1 - Help F2, E - Edit F3, T - Takt F4, S - Step F5, R - Run F6, L - Ukrainian Mouse Whell, + / - Screen Size

Рис. 1.7. Стан симулятора після уведення доданків. В інформаційному вікні трактується вміст комірки з адресою  $1110_2 = 14_{10}$ , де міститься другий доданок  $23_{10}$ . Ясно, що невидимий на рисунку курсор розташований на лінії  $1110_2$  пам'яті.

На поточний момент часу до симулятора введено програму і дані. Всього нам потрібно  $2+3=5$  комірок пам'яті. Пам'ять має 16 комірок. Отже, 11 комірок ми не використовуємо. Саме в цих комітках і залишається сміття (в нас нулі, але реально – довільна суміш нулів і одиниць). Змінимо режим роботи симулятора з РЕДАКТОР на АВТО і цим виконаємо всю програму. Отримуємо наступне вікно симулятора (рис. 2.4).



Симулятор мікроконтролера із набором CISC команд та архітектурою фон Неймана.

Кафедра ЕОМ, ІКТА, НУ 'Львівська політехніка'.

Пам'ять OEM					Інформаційне вікно		Система команд	
Адр	КОП	A0	A1	A2	Режим: Виконання		<div> <div>000: [A2] = [A0]</div> <div>001: [A2] = [A0] + [A1]</div> <div>010: [A2] = [A0] / [A1]</div> <div>011: [A2] =  [A0] - [A1] </div> <div>100: при ([A0] == [A1]) на [A2]</div> <div>101: [A2] = [A0] * [A1]</div> <div>110: при ([A0] &gt; [A1]) на [A2]</div> <div>111: Stop, Print</div> </div> <div>Клавіатура</div> <div> <div>F1 - Довідка</div> <div>F2, E - Редактор</div> <div>F3, T - Такт</div> <div>F4, S - Крок</div> <div>F5, R - Старт</div> <div>F6, L - Англійська</div> <div>Колесо миши, + / - Розмір екрану</div> </div>	
0000	001	1101	1110	1111	Адреса команди: 0001			
0001	111	1101	1110	1111	Команда процесора:			
0010	000	0000	0000	0000	111: Stop, Print			
0011	000	0000	0000	0000	Такт процесора:			
0100	000	0000	0000	0000	Вичитування R2			
0101	000	0000	0000	0000	Операнди, значення:			
0110	000	0000	0000	0000	A0 = 1101 [A0] = 17			
0111	000	0000	0000	0000	A1 = 1110 [A1] = 23			
1000	000	0000	0000	0000	A2 = 1111 [A2] = 40			
1001	000	0000	0000	0000				
1010	000	0000	0000	0000				
1011	000	0000	0000	0000				
1100	000	0000	0000	0000				
1101	000	0000	0001	0001				
1110	000	0000	0001	0111				
1111	000	0000	0010	1000				

Регістри процесора		Стан процесора		Екран ЕОМ	
Адреса команди: 0001		Stop		A0 = 1101 [A0] = 17	
R0: 17				A1 = 1110 [A1] = 23	
R1: 23				A2 = 1111 [A2] = 40	
R2: 40					

Редагування	Такт	Крок	Пуск	Англійська	Вихід
-------------	------	------	------	------------	-------