

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ІВАНА ФРАНКА

Факультет прикладної математики та інформатики

**Теорія алгоритмів**

**ЛАБОРАТОРНА РОБОТА №8**

**Тема: «РАМ. Програмування та оцінювання»**

Виконав:

*Ст. Лук'янчук Денис*

Група ПМІ-23

2025

**Тема:** «РАМ. Програмування та оцінювання»

**Мета роботи:** Ознайомитися з принципами роботи регістрової арифметичної машини (РАМ) та її розширеної версії (РАСП), вивчити реалізацію базових операцій для сортування чисел, перевірки послідовностей виду  $0\ 1^n\ 2^n\ 0$ , обчислення  $n^n$ , введення масивів з непрямою адресацією та підрахунку від'ємних чисел. Зрозуміти відмінності між РАМ і машиною Тюрінга, оцінити якість програмування коду, його відповідність завданням і ефективність виконання. Навчитися застосовувати РАМ для вирішення алгоритмічних задач і оцінювати програмні реалізації.

### **Хід роботи**

Для виконання завдань використано наданий код класів RAM і RASP, які реалізують регістрову арифметичну машину з підтримкою операцій READ, WRITE, LOAD, LOADM, STOREM, ADD, SUB, MUL, JMP, JGT, JEQ (для РАМ) та додаткової операції JLT (для РАСП).

### **Завдання 1: Сортування чисел**

#### **Алгоритм виконання:**

1. Зчитування кількості елементів ( $n$ ) у регістр R0.
2. Перевірка, чи  $n > 0$ ; якщо ні, завершення програми.
3. Зчитування  $n$  чисел і збереження їх у пам'яті з використанням непрямої адресації.
4. Сортування масиву методом бульбашки шляхом порівняння та обміну сусідніх елементів.
5. Виведення відсортованого масиву.

#### **Реалізація:**

- Використовуються інструкції READ, JGT, JMP, LOAD, SUB, STOREM, LOADM, ADD, WRITE.
- Зчитування  $n$  (R0), перевірка умовою JGT.
- Збереження чисел у пам'ять через STOREM з індексами в регістрах.
- Два вкладені цикли (JGT, JMP) для порівняння та обміну елементів.
- Виведення результату через LOADM і WRITE.

```
Завдання 1: Сортування чисел
Вхідні дані: [3, 0, 4, 1, 2]
Відсортовані числа: [0, 1, 2, 3, 4]
Вхідні дані: [3, -1, 4, 1, 2]
Число має бути невід'ємним!
Вхідні дані: [9, 130, 48, 10, 2]
Відсортовані числа: [2, 9, 10, 48, 130]
```

## Завдання 2: Перевірка послідовності $1^n 2^n 0$

### Алгоритм виконання:

1. Перетворення вхідного рядка на числові коди ( $1 \rightarrow 1, 2 \rightarrow 2, 0 \rightarrow 0$ ).
2. Підрахунок кількості одиниць і двійок, перевірка їх рівності.
3. Перевірка наявності завершального нуля.
4. Виведення "Valid" або "Invalid".

### Реалізація:

- Використовуються інструкції LOAD, SUB, JGT, JMP, LOADM, JEQ, ADD, WRITE.
- Вхідний рядок записується в пам'ять як масив чисел.
- Цикли (JGT, JEQ) підраховують одиниці (R1) і двійки (R2).
- Перевірка рівності ( $R1 == R2$ ) та наявності нуля в кінці.

```
Завдання 2: Перевірка  $1^n 2^n 0$ 
Вхідна послідовність: 1122220
Послідовність відповідає формату  $1^n 2^{(n^2)} 0$ 
Вхідна послідовність: 112220
Послідовність НЕ відповідає формату  $1^n 2^{(n^2)} 0$ 
Вхідна послідовність: 110
Послідовність НЕ відповідає формату  $1^n 2^{(n^2)} 0$ 
Вхідна послідовність: 111222220
Послідовність НЕ відповідає формату  $1^n 2^{(n^2)} 0$ 
Вхідна послідовність: 1112220
Послідовність НЕ відповідає формату  $1^n 2^{(n^2)} 0$ 
```

### Завдання 3: Обчислення $n^n$

#### Алгоритм виконання:

1. Зчитування числа  $n$  у регістр R0.
2. Ініціалізація регістрів для обчислень.
3. Обчислення  $n^n$  через ітеративне множення.
4. Виведення результату.

#### Реалізація:

- Використовуються інструкції READ, LOAD, JGT, JMP, SUB, MUL, JEQ, WRITE.
- Зчитування  $n$  через READ у R0.
- Цикл (JGT) виконує множення ( $R1 = R1 \cdot R2$ ) з оновленням R2.
- Результат (R9) виводиться через WRITE.

Завдання 3: Обчислення  $n^n$

$n = 5$

$5^5 = 3125$

$n = 0$

$0^0 = 1$

$n = 3$

$3^3 = 27$

$n = -1$

$-1^{-1} = -1.0$

### Завдання 4: Введення масиву з непрямою адресацією

#### Алгоритм виконання:

1. Зчитування кількості елементів ( $n$ ) у регістр R0.
2. Зчитування  $n$  чисел і збереження їх у пам'яті.
3. Виведення збережених чисел.

### Реалізація:

- Використовуються інструкції READ, LOAD, SUB, JGT, JMP, STOREM, ADD, LOADM, WRITE.
- Зчитування n (R0), цикл (JGT) для введення чисел.
- Збереження через STOREM з індексами в R1.
- Виведення через LOADM і WRITE.

```
Завдання 4: Введення масиву з непрямою адресацією  
Вхідні значення: [10, 20, 30, 40]  
Введений масив: [10, 20, 30, 40]  
Вхідні значення: [11, 22, 33]  
Введений масив: [11, 22, 33]  
Вхідні значення: []  
Введений масив: []  
Вхідні значення: [17, 89, 70]  
Введений масив: [17, 89, 70]
```

### Завдання 5: Підрахунок від'ємних чисел (РАСП)

#### Алгоритм виконання:

1. Зчитування кількості елементів (n) у регістр R0.
2. Зчитування n чисел.
3. Перевірка кожного числа на від'ємність за допомогою JLT.
4. Підрахунок і виведення кількості від'ємних чисел.

### Реалізація:

- Використовуються інструкції READ, LOAD, SUB, JGT, JMP, JLT, ADD, WRITE.
- Зчитування n (R0), цикл (JGT) для обробки чисел.
- Перевірка від'ємності через JLT, інкремент лічильника (R1).
- Виведення результату через WRITE.

```
Завдання 5: Підрахунок від'ємних чисел (РАСП)
Вхідні числа: [1, -2, 3, -4, -5, 6]
Кількість від'ємних чисел: 3
Вхідні числа: [0, 7, 89, 6, 1, 4]
Кількість від'ємних чисел: 0
Вхідні числа: [0, -2, -8, 6, 1, 4]
Кількість від'ємних чисел: 2
Вхідні числа: []
Кількість від'ємних чисел: 0
```

**Висновок:** Реалізована віртуальна машина RAM та її розширення RASP успішно виконують поставлені завдання: сортування чисел, перевірку послідовностей  $1^n$ ,  $2^n$ ,  $0$ , обчислення  $n^3$ , введення масивів і підрахунок від'ємних чисел. Код є модульним, із чітким поділом на класи та функції, що полегшує його аналіз. Обробка помилок забезпечує стійкість до некоректних даних.

Порівняно з машиною Тюрінга, RAM є більш абстрактною і зручною для програмування завдяки регістрам і операціям над пам'яттю, тоді як машина Тюрінга працює з посимвольною обробкою стрічки, що ускладнює реалізацію складних алгоритмів. Однак RAM менш універсальна, оскільки не моделює обчислення з довільною пам'яттю так ефективно, як машина Тюрінга.