

**Міністерство освіти і науки України**  
**Національний технічний університет України «КПІ» імені Ігоря**  
**Сікорського**  
**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**  
**з розрахунково-графічної роботи**  
**з навчальної дисципліни «Архітектура програмного забезпечення»**

**Тема:**  
**Архітектурні діаграми та бенчмарки**

**Виконав**  
Студенти 3 курсу ІІІ-94  
Рудик М.  
Сумський П.  
Литвишко Н.

**Перевірів**  
Мазур Р. Ф.

**Київ 2021**

**Мета:** Закріплення навичок ілюстрації організації програмних систем та оцінки часу виконання алгоритмів.

**Завдання**

- Для 2-гої роботи, підтвердьте лінійний час виконання вашої функції перетворення чи обчислення вхідного виразу.
- Для 3-тої роботи, побудуйте діаграму взаємодії компонентів у вашій імплементації.
- Для 4-ої роботи, побудуйте діаграму взаємодії для вашої реалізації (на ній, скоріш за все, мають опинитися компоненти парсера, черги команд, ядра цикла) та підтвердьте лінійний час роботи вашого парсера команд.

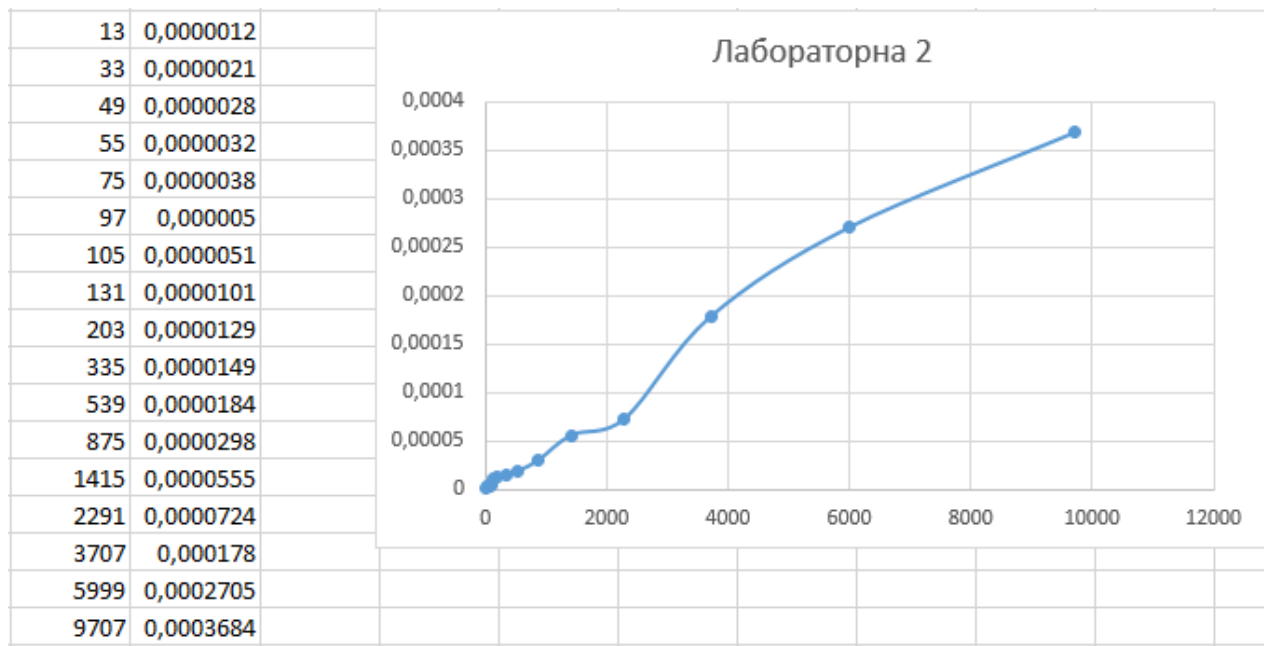
## Частина перша

Для 2-гої роботи функція для підтвердження лінійного часу виконання нашої функції:

```
8 + func BenchmarkCalculatePostfix(b *testing.B) {
9 +     postfixes := []string{
10 +         "4 2 - 3 * 5 +",
11 +         "9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + *",
12 +         "4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / +",
13 +         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + *",
14 +         "1 2 + 4 * 3 + 4 2 - 3 * 6 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 +",
15 +         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2
16 +         "4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2 * 1 3 - 2 ^ / + 3 4 2 * 1 3 - 2 ^ / + 9 3 4 + *
17 +         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6
18 +         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2
19 +         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6
20 +         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2
21 +         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6
22 +         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2
23 +         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6
24 +         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2
25 +         "3 4 2 * 1 3 - 2 ^ / + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 1 2 + 4 * 3 + 4 2 - 3 * 6
26 +         "4 2 - 3 * 5 + 9 3 4 + * 6 / 1 - 8 2 ^ 5 1 - + * 4 2 - 3 * 5 + 4 2 - 3 * 5 + 3 4 2
27 +     }
28 +     l := len(postfixes);
29 +     for i := 0; i < l; i++ {
30 +         b.Run(fmt.Sprintf("len=%d", len(postfixes[i])), func(b *testing.B) {
31 +             calculatePostfix(postfixes[i])
32 +         })
33 +     }
34 + }
```

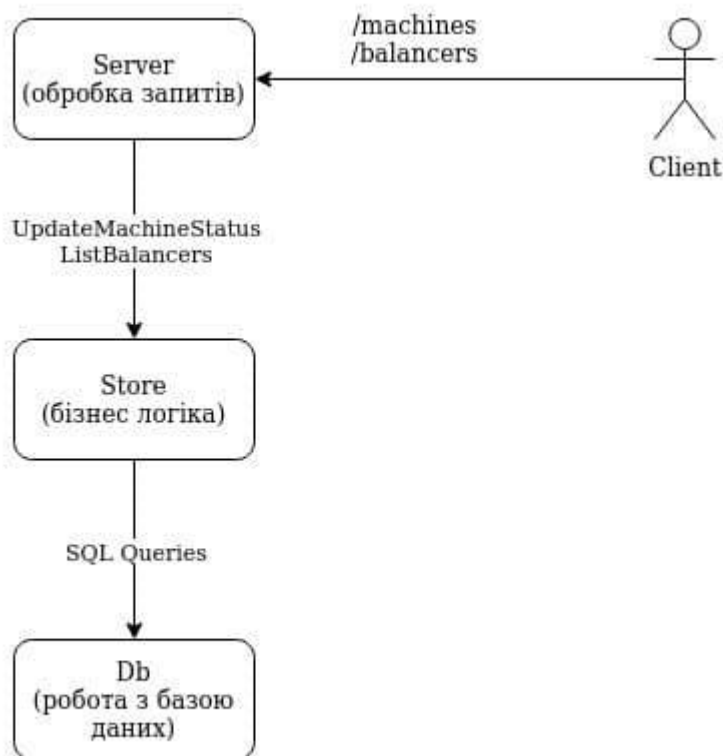
Результати виконання:

```
goos: linux
goarch: amd64
pkg: github.com/Destaby/architecture-lab-2
cpu: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
BenchmarkCalculatePostfix/len=13      1000000000      0.0000012 ns/op
BenchmarkCalculatePostfix/len=33      1000000000      0.0000021 ns/op
BenchmarkCalculatePostfix/len=49      1000000000      0.0000028 ns/op
BenchmarkCalculatePostfix/len=55      1000000000      0.0000032 ns/op
BenchmarkCalculatePostfix/len=75      1000000000      0.0000038 ns/op
BenchmarkCalculatePostfix/len=97      1000000000      0.0000050 ns/op
BenchmarkCalculatePostfix/len=105     1000000000      0.0000051 ns/op
BenchmarkCalculatePostfix/len=131     1000000000      0.0000101 ns/op
BenchmarkCalculatePostfix/len=203     1000000000      0.0000129 ns/op
BenchmarkCalculatePostfix/len=335     1000000000      0.0000149 ns/op
BenchmarkCalculatePostfix/len=539     1000000000      0.0000184 ns/op
BenchmarkCalculatePostfix/len=875     1000000000      0.0000298 ns/op
BenchmarkCalculatePostfix/len=1415    1000000000      0.0000555 ns/op
BenchmarkCalculatePostfix/len=2291    1000000000      0.0000724 ns/op
BenchmarkCalculatePostfix/len=3707    1000000000      0.0001780 ns/op
BenchmarkCalculatePostfix/len=5999    1000000000      0.0002705 ns/op
BenchmarkCalculatePostfix/len=9707    1000000000      0.0003687 ns/op
PASS
ok      github.com/Destaby/architecture-lab-2  0.036s
```



### Частина друга

Для 3-тої роботи ми побудували наступну діаграму взаємодії компонентів у нашій імплементації:



Система підтримки управління віртуальними машинами та балансерами навантаження. Користувач може ініціалізувати кілька машин, які виконують однакову функцію та підключити їх до балансера, який рівномірно розподілятиме запити до цих машин. Машина може бути помічена як

неробоча. У такому випадку, балансер повинен перестати відправляти на неї запити.

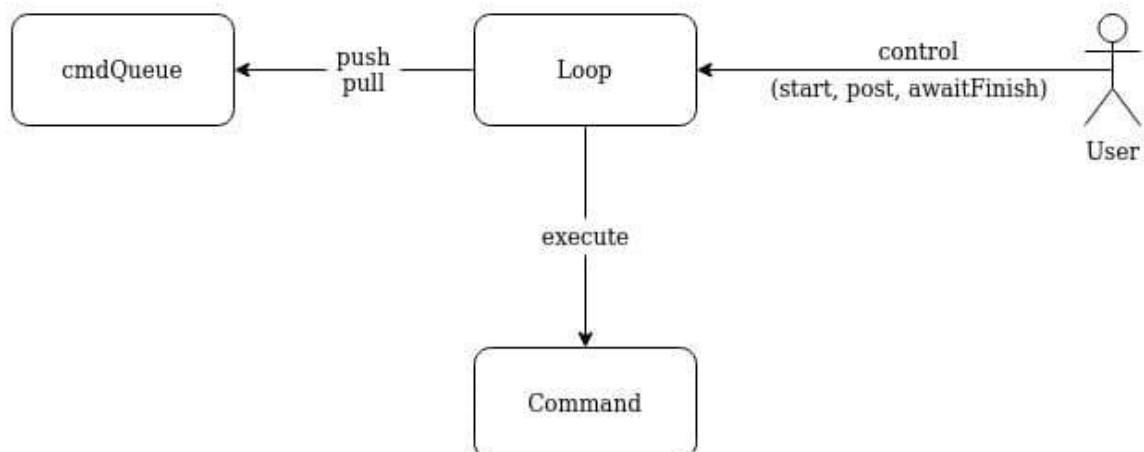
Клієнт надсилає запити до серверу (machines/balancers). У цей час сервер обробляє його та надсилає сигнал для оновлення статусу машини та список балансерів. Store зв'язується з базою даних та надсилає SQL запити.

### Частина третя

Для 4-ої роботи функція для підтвердження лінійного часу роботи парсера команд:

```
1  package main
2
3  import (
4      "testing"
5      "fmt"
6      "strings"
7  )
8
9  func BenchmarkParse(b *testing.B) {
10     commands := []string{
11         "print",
12         "palindrom",
13     }
14     const baseLen = 5
15
16     for i := 0; i < 20; i++ {
17         l := baseLen * 1<<i
18         in := make([]string, l)
19         in[0] = commands[i % 2]
20         for k := 1; k < l; k++ {
21             in[k] = "a"
22         }
23         b.Run(fmt.Sprintf("len=%d", l), func(b *testing.B) {
24             parse(strings.Join(in, ""))
25         })
26     }
27 }
```

Побудована діаграма взаємодії для нашої реалізації:

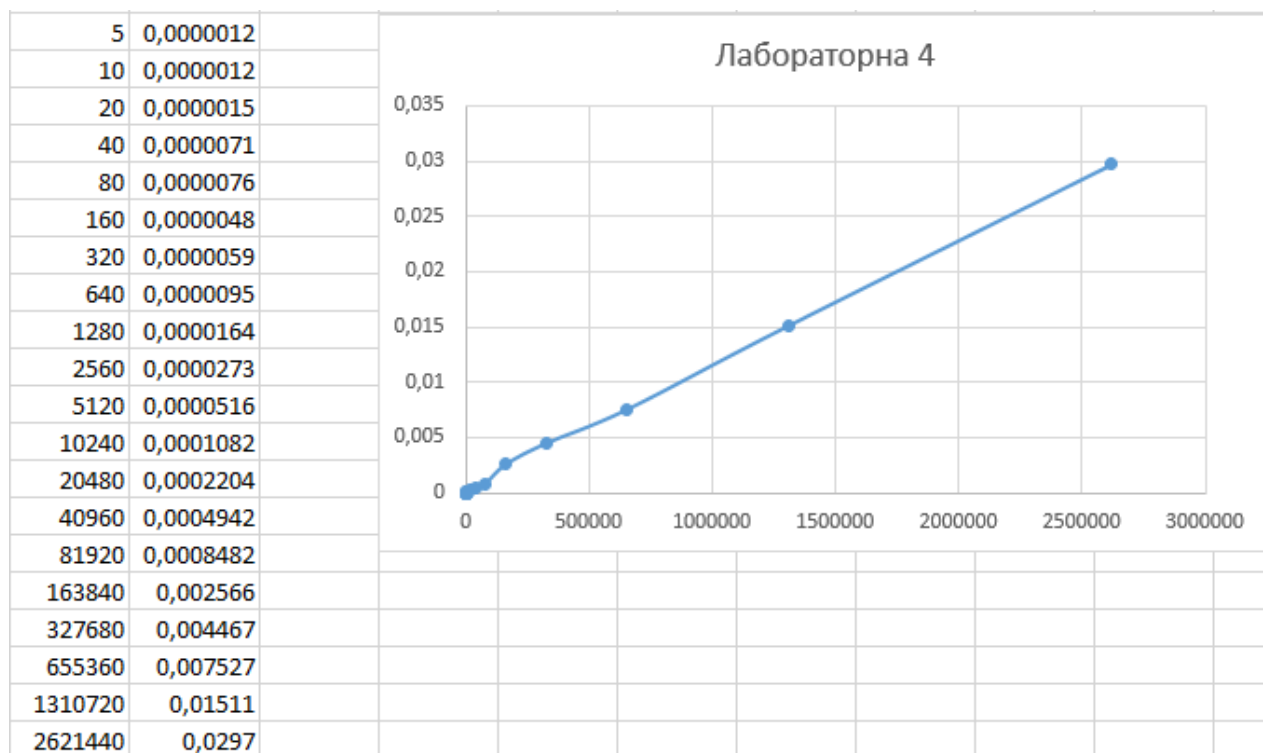


Інтерпретатор читає вхідний файл рядок за рядком, парсить команду в прочитаному рядку та додає команду в чергу обробки циклу подій. На

початку програми створюється екземпляр EventLoop у пакеті engine. Він має 3 публічних метода: Start, який створює чергу повідомлень - messageQueue та запускає цикл зчитування команд з даної черги, Post(Command) (який додає нову команду до черги для обробки) та AwaitFinish (який очікує, поки всі заплановані команди, що було додано до черги повідомлень, завершаться). У даному інтерфейсі є лише один метод Execute, який виконується всередині циклу подій. Його аргумент надає імплементації команди можливість планувати виконання додаткових команд. Далі EventLoop виводить послідовний результат (push/pull).

Результати виконання функції виміру часу:

```
goos: linux
goarch: amd64
pkg: github.com/Destaby/architecture-lab4
cpu: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
BenchmarkParse/len=5      10000000000      0.0000012 ns/op
BenchmarkParse/len=10     10000000000      0.0000012 ns/op
BenchmarkParse/len=20     10000000000      0.0000015 ns/op
BenchmarkParse/len=40     10000000000      0.0000071 ns/op
BenchmarkParse/len=80     10000000000      0.0000076 ns/op
BenchmarkParse/len=160    10000000000      0.0000048 ns/op
BenchmarkParse/len=320    10000000000      0.0000056 ns/op
BenchmarkParse/len=640    10000000000      0.0000095 ns/op
BenchmarkParse/len=1280   10000000000      0.0000164 ns/op
BenchmarkParse/len=2560   10000000000      0.0000273 ns/op
BenchmarkParse/len=5120   10000000000      0.0000516 ns/op
BenchmarkParse/len=10240  10000000000      0.0001082 ns/op
BenchmarkParse/len=20480  10000000000      0.0002204 ns/op
BenchmarkParse/len=40960  10000000000      0.0004942 ns/op
BenchmarkParse/len=81920  10000000000      0.0008482 ns/op
BenchmarkParse/len=163840 10000000000      0.002566 ns/op
BenchmarkParse/len=327680 10000000000      0.004467 ns/op
BenchmarkParse/len=655360 10000000000      0.007527 ns/op
BenchmarkParse/len=1310720 10000000000      0.01511 ns/op
BenchmarkParse/len=2621440 10000000000      0.02970 ns/op
PASS
ok      github.com/Destaby/architecture-lab4    0.792s
```



### Висновки:

При виконанні розрахунково-графічної роботи ми змогли закріпити навички ілюстрації програмних систем та оцінки часу виконання алгоритмів.

Розробили бенчмарки для двох робіт, які виконували раніше, та діаграми взаємодії.