Звіт по лабораторній роботі №6

З архітектури обчислювальних систем

Студента групи К-22

Ламзіна Олега

**Мій варіант:**

1. Адресність процесора – **2-адресна**

2. Бітність регістру – **28-бітні**

3. Команди: **№15.**

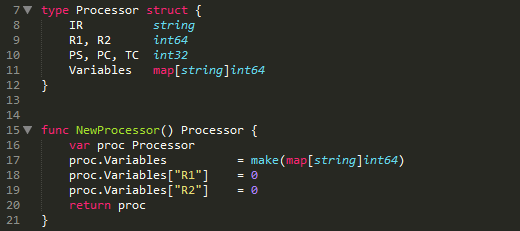
Перестановка значень пари бітів у 1-му операнді. Номери бітів задаються 2-м операндом у вигляді AABB, де AA та BB є номерами бітів (з ведучими нулями для доповнення до двозначного числа). 2-й операнд представляється у:

· команді безпосередньо чи регістрі для безстекової реалізації;

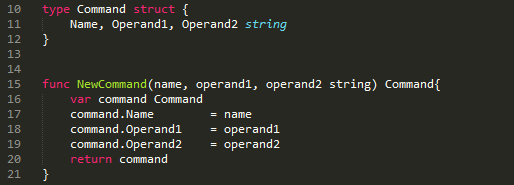
· верхівці стека в стековій реалізації розміщення операндів.

Для реалізації я використав мову програмування Go. Регістри зберігаю в доповнюючомі коді в змінній типу int64. Молодші 27 бітів виділяється для зберігання інформації про число, а також старший біт int64 використовую для зберігання знаку. Тобто всього корисних бітів буде 28, що власне і потрібно. Таким чином діапазон значень над якими можемо працювати: **[-2^27, 2^27 - 1].**

Структура **Processor** та відповідний конструктор. Тут ми зберігаємо інформацію, яка необхідна з умов лабораторної роботи а також значення регістрів **R1, R2** та значення усіх змінних – **map[string]int64**.

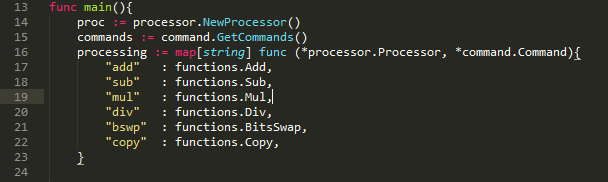


Структура **Command** та конструктор:

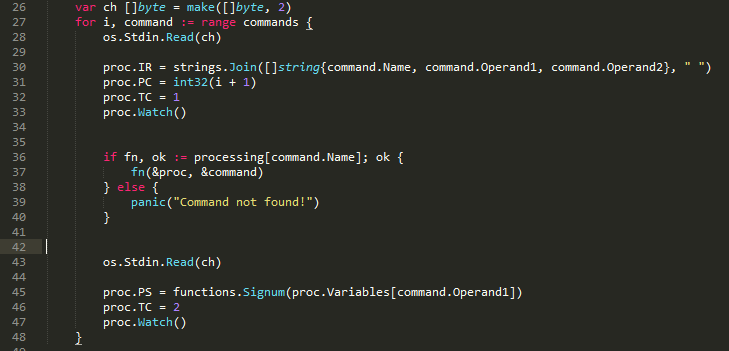


Цих структур нам буде достатньо.

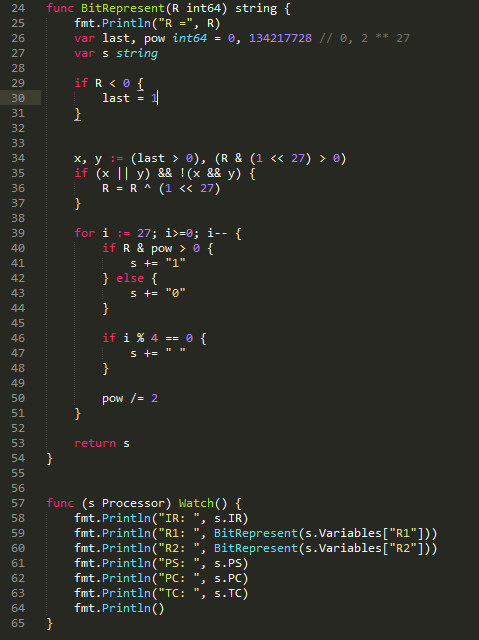
Далі створюємо об‘єкт процессор та парсимо вхідний файл, після чого отримуємо набір операцій. Крім того створюємо map функцій які будуть викликатися для обробки операції по ключу назви команди(щоб не робити величезний switch оператор). Доступні команди: додавання, віднімання, множення, ділення і т.д.



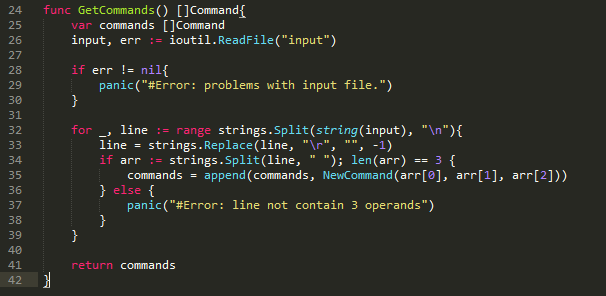
Далі по черзі опрацьовуємо всі вхідні операції. Watch() – ф-я що виводить на екран стан процесора. Виводиться лише після натискання клавіші юзером.



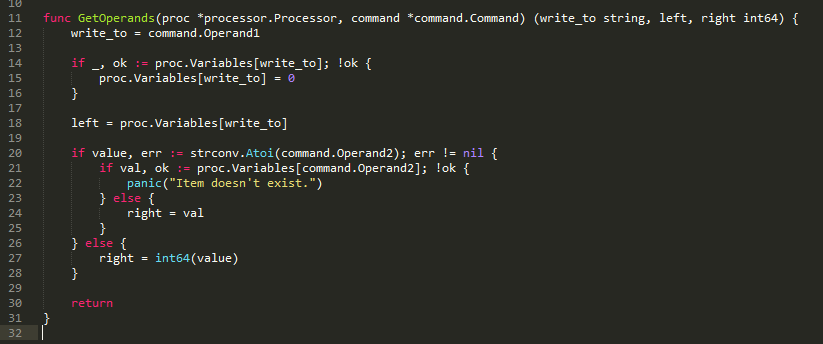
Лістинг ф-ї **Watch**() & **BitRepresent**() – виводить побітово **int64**.



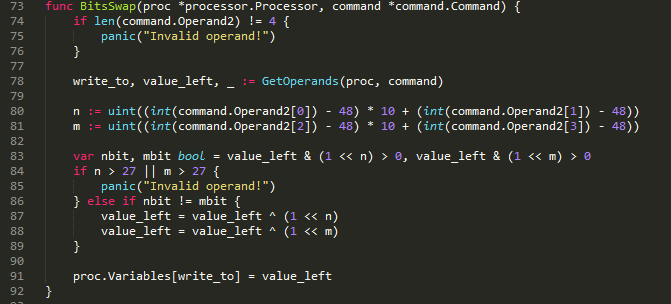
Парсинг файлу:



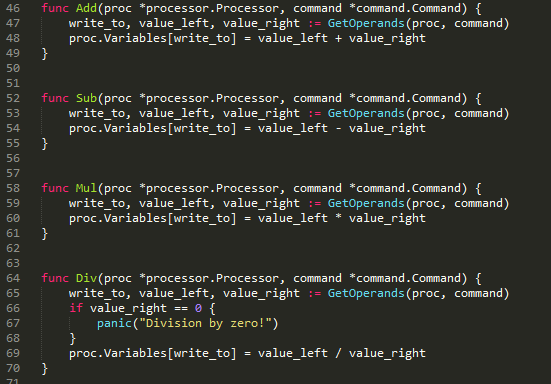
Парсинг операції – отримання назви змінної куди записувати та лівого, правого значення операнда:



Тепер ф-я обробки операції, що вказано в моєму варіанті – свопінг двох бітів:

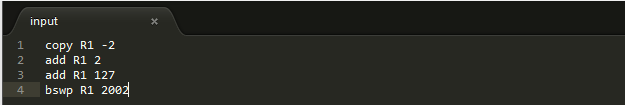


Інші ф-ї:



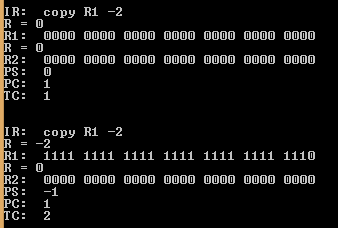
Скріншоти роботи програми:

Вхідний файл:

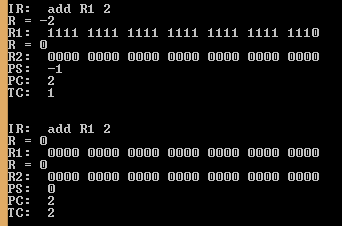


Запуск: **go run main.go**

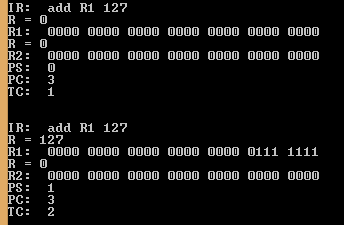
1 операція, 1 та 2 такти // R – значення регістру, а потім його побітове представлення.



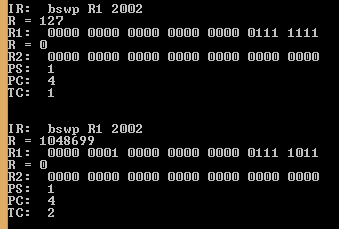
2 операція, 1 та 2 такти



3 операція, 1 та 2 такти



4 операція, 1 та 2 такти



Програма іспішно виконала усі операції.