Київський національний університет
імені Тараса Шевченка

Звіт

до лабораторної роботи №3
на тему:

«Імітаційна модель математичного співпроцесора»

Студента другого курсу
Групи К-22
Факультету комп'ютерних
наук та кібернетики
Коваля Максима

Київ

2020

Мета

Метою даної лабораторної роботи є розробка програмної моделі співпроцесора та реалізування його імітаційної (тобто комп'ютерну) моделі.

Основні принципи виконання роботи

Для виконання даної лабораторної роботи було обрано мову програмування Java (SE 8). Також було використано основні принципи ООП, що можна побачити у Додатку.

Згідно з варіантом мені було запропоновано розробити імітаційну модель математичного співпроцесора з форматом представлення числа з плаваючою крапкою IEEE 754 з наступними обмеженнями: 3 біти для характеристики, 9 бітів для мантиси (без урахування неявного біта)), та з заданими обмеженнями провести розрахунок наступної формули: $ln\ y$*cos $x\ +\ 3tgx$.

Перед тим як реалізовувати безпосередньо саму модель співпроцесора, було розроблено конвертер між стандартним десятковим представленням числа та його двійковим представленням у форматі IEEE 754. При розробці конвертера було враховано усі рекомендації відповідного формату, а також можливість опрацювання наступних значень:

- мінімальне за абсолютною величиною ненульове представлення;
- максимальне додатне представлення;
- мінімальне від'ємне представлення;
- число +1,0E0;
- значення +∞;
- значення -∞;
- будь-який варіант для ненормалізованого ЧПТ;
- будь-який варіант для NaN-значення.

В програмній реалізації математичного співпроцесора було використано стекову архітектуру з 4 регістрами (сам стек представлений у вигляді окремого класу, котрий агрегує у собі регістри даних), оскільки при розрахунку формули, заданої варіантом, використовується лише 3 регістра (було додано ще 1, щоб візуально бачити його “пустоту”). Також до програмної реалізації було додано регістр стану, що містить знак останнього результату та ознаку переповнення. Логіка переповнення схожа до аналогічної в сучасних мовах програмування (Java, C++): тип “зациклений” (наприклад, при додаванні 1 до максимального значення результатом буде значення, на 1 більше від мінімального).

Для візуального представлення виконаних обрахунків на екран виводиться стек з його 4 регістрами, регістр стану, а також команду (операцію), що виконується на відповідному етапі обрахунку. Було використано стандартну асемблерівську мнемоніку команд (операцій). Для того, щоб “спровокувати” виконання наступної за чергою команди, користувачу необхідно натиснути на клавішу Enter.

Також слід зазначити, що дотримання принципів ООП дозволяють легко модифікувати програму. Для того, щоб змінити кількість бітів, відведених для характеристики, чи мантиси, достатньо всього лиш змінити відповідну число у одному місці програми. Також доволі легко добавити нові команди: для цього необхідно виконати наслідування класу ValueCommand (для команд, котрі приймають один, чи декілька операндів) чи класу OperandCommad (для команд, котрі не приймають операндів).

Результати роботи програми

При запуску програми з заданими змінними *x* та *y*, які відповідно дорівнюють *0.123* та *1.23*, отримали наступні результати:

```
Ця лабораторна робота симулює роботу математичного сопроцесора та демонструє обрахунок наступного виразу: ln(y)*cos(x) + 3*tg(x)

Введіть x, y для того, щоб почати обрахунок: 0.123 1.23

command = FLD 0.123     ### Встановлюємо 0.123 на вершину стеку
name   s ch  man
P1  =  0 000 111101111
P2  =  0 000 000000000
P3  =  0 000 000000000
P4  =  0 000 000000000
PS  =  0 | 0

command = FDUPL     ### Дублюємо вершину стеку
name   s ch  man
P1  =  0 000 111101111
P2  =  0 000 111101111
P3  =  0 000 000000000
P4  =  0 000 000000000
PS  =  0 | 0

command = FTAN      ### Обчислюємо тангенс від значення, котре зберігається на вершині стеку та заміняємо вершину стеку на отримане значення
name   s ch  man
P1  =  0 000 111110100
P2  =  0 000 111101111
P3  =  0 000 000000000
P4  =  0 000 000000000
PS  =  0 | 0

command = FLD 3.0       ### Встановлюємо 3.0 на вершину стеку
name   s ch  man
P1  =  0 100 100000000
P2  =  0 000 111110100
P3  =  0 000 111101111
P4  =  0 000 000000000
PS  =  0 | 0

command = FMULP ST(0) ST(1)     ### Множимо два верхні значення зі стеку та викидуємо їх; результат залишається на вершині
name   s ch  man
P1  =  0 001 011110111
P2  =  0 000 111101111
P3  =  0 000 000000000
P4  =  0 000 000000000
PS  =  0 | 0

command = FXCH ST(1)        ### Міняємо місцями два верхні значення зі стеку
name   s ch  man
P1  =  0 000 111101111
P2  =  0 001 011110111
P3  =  0 000 000000000
P4  =  0 000 000000000
PS  =  0 | 0
command = FCOS      ### Обчислюємо косинус від значення, котре зберігається на вершині стеку та заміняємо вершину стеку на отримане значення
name   s ch  man
P1  =  0 010 111111000
P2  =  0 001 011110111
P3  =  0 000 000000000
P4  =  0 000 000000000
PS  =  0 | 0
```

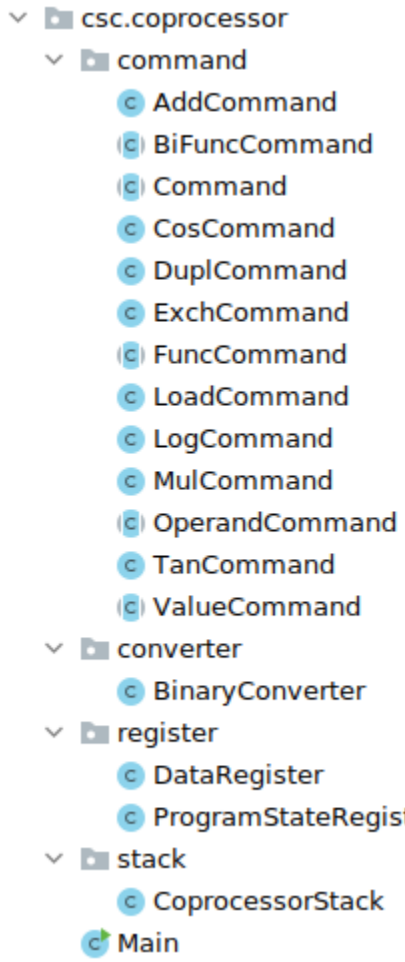
```
command = FLD 1.23      ### Встановлюємо 1.23 на вершину стеку
name   s ch  man
P1  =  0 011 001110101
P2  =  0 010 111111000
P3  =  0 001 011110111
P4  =  0 000 000000000
PS  =  0 | 0

command = FLOGE     ### Обчислюємо натуральний логарифм від значення, котре зберігається на вершині стеку та заміняємо вершину стеку на отримане значення
name   s ch  man
P1  =  0 000 101001111
P2  =  0 010 111111000
P3  =  0 001 011110111
P4  =  0 000 000000000
PS  =  0 | 0

command = FMULP ST(0) ST(1)     ### Множимо два верхні значення зі стеку та викидуємо їх; результат залишається на вершині
name   s ch  man
P1  =  0 000 101001001
P2  =  0 001 011110111
P3  =  0 000 000000000
P4  =  0 000 000000000
PS  =  0 | 0

command = FADDP ST(0) ST(1)     ### Додаємо два верхні значення зі стеку та викидуємо їх; результат залишається на вершині
name   s ch  man
P1  =  0 010 001001110
P2  =  0 000 000000000
P3  =  0 000 000000000
P4  =  0 000 000000000
PS  =  0 | 0
Результат обчислення: 0.5763223815948912
```

Висновок

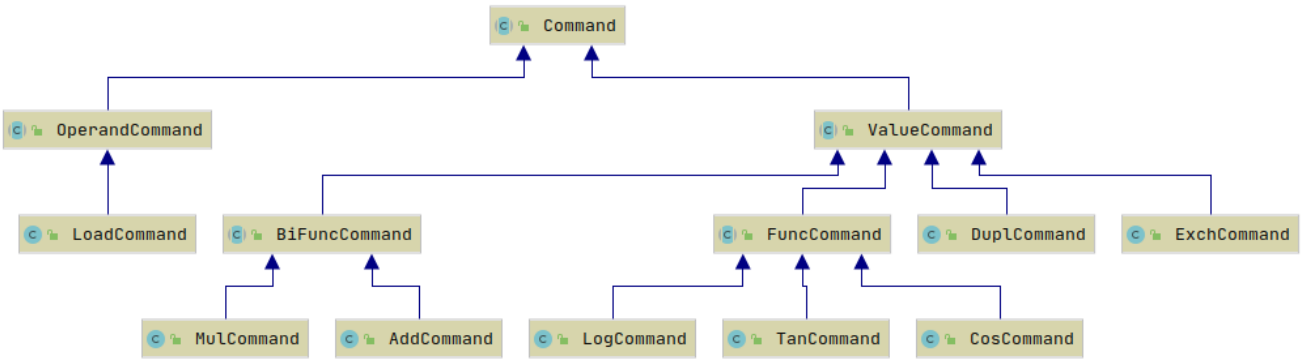
Під час виконання даної лабораторної роботи були виконані всі вимоги до її виконання, а саме: було реалізовано модель математичного співпроцесора з форматом представлення числа з плаваючою крапкою IEEE 754 з наступними обмеженнями: 3 біти для характеристики, 9 бітів для мантиси (без урахування неявного біта)), та з заданими обмеженнями провести розрахунок наступної формули: $ln\ y$*cos $x\ +\ 3tgx$. Також було реалізована логіка регістру стану, а саме логіку при переповненні під час виконання команди.

Додаток

Програма містить наступні класи:



Детальна структура структура класів команд (UML Diagram):



Посилання на репозиторій GitHub, на котрому розміщено вихідні коди програми - https://github.com/maxym-ko/cyb_java