Санкт-Петербургский политехнический университет Петра Великого Институт компьютерных наук и технологий Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчёт по лабораторной работе №1

Дисциплина: Низкоуровневое программирование

Тема: Программирование EDSAC

Вариант: 2

Выполнил студент группы 3530901/00002 _____ А.Г. Антонов
(подпись)

Принял преподаватель _____ Д.С. Степанов
(подпись)

«__» _____ 2021г.

Санкт-Петербург

2021

Постановка задачи

1. Разработать программу для EDSAC, реализующую определенную вариантом задания функциональность, и предполагающую загрузчик Initial Orders 1. Массив (массивы) данных и другие параметры (преобразуемое число, длина массива, параметр статистики и пр.) располагаются в памяти по фиксированным адресам.

2. Выделить определенную вариантом задания функциональность в замкнутую (closed) подпрограмму, разработать вызывающую ее тестовую программу. Использовать возможности загрузчика Initial Orders 2. Адрес обрабатываемого массива данных и другие параметры передавать через ячейки памяти с фиксированными адресами.

Задание

Формирование в памяти десятичного представления дробного числа (-1;1).

Алгоритм

Для реализации задания было принято решение сохранять число в его ieee754 представлении в 2/10 коде. Знак и экспонента не нуждаются в переводе в 2/10 код, т.к занимают в памяти =< 1 байта. А для перевода мантиссы был выбран алгоритм, представленный на рис. 1.

Алгоритм 2.4. Перевод дробного двоичного числа в 2/10 последовательным умножением двоичной дроби на 10 в двоичной арифметике. Умножение осуществляется как 10A = 2(4A + A) с использованием двоичного сумматора. За n_2 циклов двоичная дробь будет точно переведена в 2/10 . Пример: $A_2 = 0{,}1011$

Получение		0000,1011	записана исходная двоичная дробь
первой 2/10	←2 разряда	0010,1100	
цифры	∑	0011,0111	
	←1 разряд	**0110**,1110	умн. на 10 получили целую и дробную части
Получение		0000,1110	оставлена дробная часть
второй 2/10	←2 разряда	0011,1000	
цифры	∑	0100,0110	
	←1 разряд	**1000**,1100	умн. на 10 получили целую и дробную части
Получение		0000,1100	оставлена дробная часть
третьей 2/10	←2 разряда	0011,0000	
цифры	∑	0011,1100	
	←1 разряд	**0111**,1000	умн. на 10 получили целую и дробную части
Получение		0000,1000	оставлена дробная часть
четвертой 2/10	←2 разряда	0010,0000	
цифры	∑	0010,1000	
	←1 разряд	**0101**,0000	умн. на 10 получили целую и дробную части

После четырех циклов умножения на 10 в дробной части двоичного числа остался 0. Значит перевод выполнен точно. Получен результат:
$A_{2/10} = 0,\ 0110\ 1000\ 0111\ 0101$

Рис 1.

Число 0,36 в ieee754 равно 0111010101110001, что в десятичной системе счисления будет равно 30065

Initial Orders 1

Initial Orders 1 – простая программа, которая решает следующую задачу: чтение символов (т.е., разумеется, кодов символов) с перфоленты, формирование кодов инструкций (и констант) и запись их в память. **Initial Orders 1** состоит инструкций, которые загружаются в ячейки 0 – 30 и начинают исполняться после нажатия кнопки «Start». Программа, записанная на ленте, загружается в следующие друг за другом ячейки памяти, начиная с адреса 31.

```
T 92 S [Программа заканчивается на 99 строке]
E 36 S [Начало программы]
P 15032 L [Число]
P 0 L [1]
P 15 L  [31]
A 33 S [Записываем число в аккумулятор]
U 148 S [Записываем число в ячейку]
R 1024 S [Сдвигаем число вправо]
R 2 S [Сдвигаем вправо и остаётся знак числа]
T 184 S [Записываем число в ячейку знак числа]
A 184 S  [Добавляем число из ячейки]
L 1024 S [Сдвигаем в лево]
L 2 S [Сдвигаем влево и остаётся знак с нулевой мантиссой и нулевым порядком]
T 144 S  [Записываем в ячейку]
A 148 S  [Берём из ячейки изначальное число]
S 144 S  [Вычитаем знак]
U 142 S  [Записываем число без знака в ячейку]
R 256 S  [Сдвигаем вправо, чтобы получить порядок]
T 140 S  [Записываем порядок]
A 140 S  [Добавляем порядок]
L 256 S  [Сдвигам порядок влево, чтобы получить порядок с нулевой мантиссой]
T 134 S  [Записываем в ячейку]
A 142 S  [Добавляем число без знака]
S 134 S  [Вычитаем порядок]
T 138 S  [Записываем мантиссу]
A 38 S   [Добавляем максимальное значение порядка]
S 140 S  [Вычитаем порядок]
U 140 S  [Записываем порядок в положительной записи]
T 100 S   [Записываем порядок]
A 138 S   [Берем мантиссу]
T 128 S  [Записываем мантиссу в рабочую ячейку]
A 128 S
L 1 S   [Выполняем сдвиг на два разряда]
T 130 S [Сохраняем число]
A 130 S  [прибавляем слагаемое 1]
A 128 S    [Прибавляем слагаемое 2]
```

```
T 130 S  [Сохраняем результат]
A 130S
L 1 L    [Сдвигаем на один разряд]
T 132 S  [Сохраняем результат]
A 132 S
R 256 S  [Убираем дробную часть]
T 182 S  [Сохраняем результат]
A 182 S
L 256 S  [Сдвигаем на 10 бит влево]
T 154 S
A 132 S
S 154 S [Убираем целую часть]
T 128 S  [Сохраняем результат]
A 128 S [Повторяем алгоритм]
L 1 S
T 130 S
A 130 S
A 128 S
T 130 S
A 130S
L 1 L
T 132 S
A 132 S
R 256 S
T 180 S
```

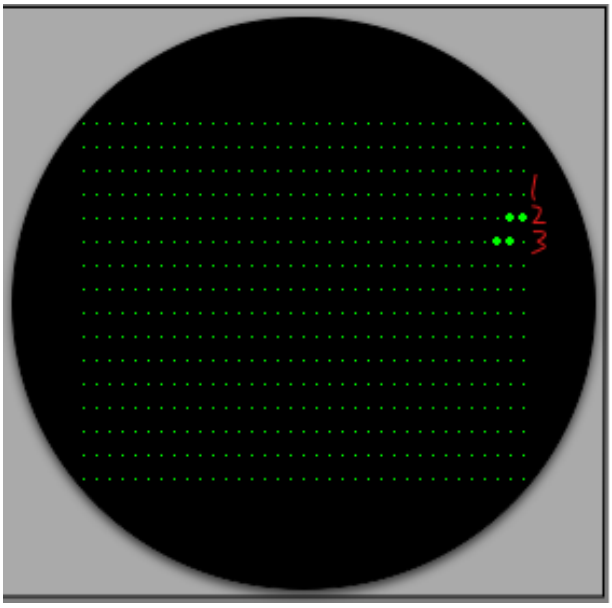

Рис 2. Итог выполнения программы

Как видно из рисунка 2 в строчке 1(WORD 184) хранится знак числа. В строчке 2(WORD 182) хранится 1 цифра после запятой. В строчке 3(WORD 180) 2 цифра после запятой.

Initial Orders 2

Initial Orders 2 – значительно более сложная (семантически) программа, чем Initial Orders 1, имеющая, тем не менее, немногим больше инструкций (41 вместо 31), и решающая схожую задачу.

Форма записи инструкции при использовании Initial Orders 2 незначительно отличается от формы записи инструкций в случае Initial Orders 1. Однако, в отличие от Initial Orders 1, при использовании Initial Orders 2 на ленте находятся не только инструкции (или константы) загружаемой программы, но и управляющие последовательности (“control combinations” в терминах Initial Order 2, современный термин – “директивы”), определяющее поведение самой программы Initial Order 2 в процессе загрузки.

Директивы Initial Order 2 записываются в той же форме, что и инструкции загружаемой программы. Initial Orders 2 состоит из 41 инструкции

Код программы:

```
         T 56 K
      G K
      A 3 F [пролог: формирование кода инструкции возврата в аккумуляторе]
      T 53 @ [пролог запись инструкции возврата]
      A 33 F [Записываем число в аккумулятор]
      U 148 F [Записываем число в ячейку]
      R 1024 F [Сдвигаем число вправо]
      R 2 F [Сдвигаем вправо и остаётся знак числа]
      T 184 F [Записываем число в ячейку знак числа]
      A 184 F  [Добавляем число из ячейки]
      L 1024 F [Сдвигаем в лево]
      L 2 F [Сдвигаем влево и остаётся знак с нулевой мантиссой и нулевым
порядком]
      T 144 F  [Записываем в ячейку]
      A 148 F  [Берём из ячейки изначальное число]
      S 144 F  [Вычитаем знак]
      U 142 F  [Записываем число без знака в ячейку]
      R 256 F  [Сдвигаем вправо, чтобы получить порядок]
      T 140 F  [Записываем порядок]
      A 140 F  [Добавляем порядок]
      L 256 F  [Сдвигам порядок влево, чтобы получить порядок с нулевой
мантиссой]
      T 134 F  [Записываем в ячейку]
      A 142 F  [Добавляем число без знака]
      S 134 F  [Вычитаем порядок]
      T 138 F  [Записываем мантиссу]
      A 115 F  [Добавляем максимальное значение порядка]
      S 140 F  [Вычитаем порядок]
      U 140 F  [Записываем порядок в положительной записи]
      T 100 F   [Записываем порядок]
      A 138 F   [Берем мантиссу]
      T 128 F  [Записываем мантиссу в рабочую ячейку]
      A 128 F
      L 1 F  [Выполняем сдвиг на два разряда]
      T 130 F [Сохраняем число]
      A 130 F  [прибавляем слагаемое 1]
```

```
A 128 @    [Прибавляем слагаемое 2]
T 130 F  [Сохраняем результат]
A 130F
L 1 D    [Сдвигаем на один разряд]
T 132 F  [Сохраняем результат]
A 132 F
R 256 F  [Убираем дробную часть]
T 182 F  [Сохраняем результат]
A 182 F
L 256 F  [Сдвигаем на 10 бит влево]
T 154 F
A 132 F
S 154 F [Убираем целую часть]
T 128 F[Сохраняем результат]
A 128 F [Повторяем алгоритм]
L 1 F
T 130 F
A 130 F
A 128 @
T 130 F
A 130F
L 1 D
T 132 F
A 132 F
R 256 F
T 180 F
E 0 F
P 15 D
P 0 L

G K [-директива IO2, фиксация начального адреса программы]
A 5 @ [загрузка исходного числа]
T 33 F [запись исходного числа]
A 2 @  [\вызов]
G 56 F [/подпрограммы]
Z F [остановка]
```

```
P 15032 D
EZ PF
```

Вывод

Являясь одной из первых ЭВМ, EDSAC может выполнять широкий спектр задач, несмотря на ограничения, вызванные неудобством программирования и малой вычислительной мощностью.