МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С. П. КОРОЛЕВА»

(САМАРСКИЙ УНИВЕРСИТЕТ)

Институт информатики и кибернетики

Факультет информатики

Кафедра информационных систем и технологий

Отчет по лабораторной работе № 2

по курсу «Микропроцессорные средства и системы»

Выполнил: Алеев И.И.

Группа 6304

Проверил: Иоффе В.Г.

# ЗАДАНИЕ

Выполнить процедуру нормализации числа с плавающей точкой, находящегося в ОЗУ. Число состоит из 3 байт: первый байт – знак числа + характеристика, второй – старший байт мантиссы, третий младший байт мантиссы. Мантисса представлена в дополнительном коде.

Нормализация состоит в сдвиге мантиссы влево до появления первого значащего бита с одновременной коррекцией характеристики. Для положительного числа значение значащего бита - 1, для отрицательного - 0. В целях повышения точности после этого может выполнятся дополнительный сдвиг с коррекцией характеристики ( формирование «скрытого» разряда).

# БЛОК-СХЕМА АЛГОРИТМА

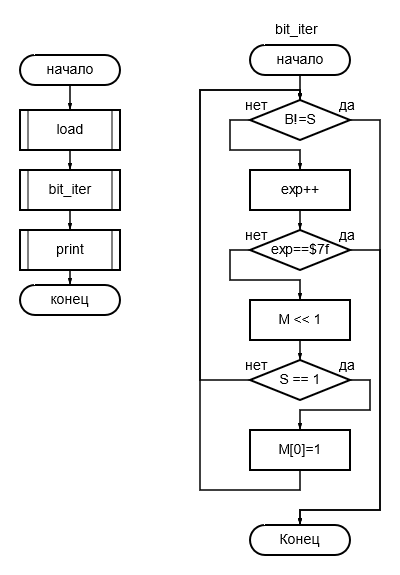


Рисунок 1 – Блок-схема алгоритма

bit\_iter – процедура нормализации

Print – процедура вывода на 7 сегментные дисплеи результата

M – байты мантиссы

EXP – порядок

S – знак

B – старший бит мантиссы

M=MH:ML

# РАЗРЯДНОСТЬ ДАННЫХ

# ML [0:7] MH [0:7] EXP [0:6] S [7] SE [0:7]

# ОПИСАНИЕ АЛГОРИТМА

1. С помощью битовой маски, вычисляется бит знака
2. Проверяется старший бит мантиссы, Если бит является обратным к биту знака, то алгоритм завершается
3. Над старшим байтом совершается логический сдвиг влево
4. Проверяется старший бит младшего майта мантиссы. Если бит установлен, то инкрементируется старший байт мантиссы.
5. Над младшим байтом совершается логический сдвиг влево
6. Проверяется бит знака. Если бит поднят, то инкрментируется младший байт мантиссы, иначе ничего не делаем
7. Инкрементируется порядок
8. Переход к пункту 2

# РАСПРЕДЕЛЕНИЕ РЕГИСТРОВ

r10 хранит бит знака

r9 порядок

r8 старший байт мантиссы

r7 младший байт мантиссы

# ОБЪЕМ

Объем сегмента кода .cseg, вычисленный средствами ПО Proteus, равен 154 байт:

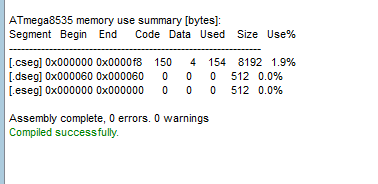


Рисунок 2 – Результат компиляции

# ВРЕМЯ ВЫПОЛНЕНИЯ

Время обработки: t = 13 + N , при частоте работы процессора в 1 МГц , где N – количество итераций в цикле нормализации.

# ПРИНЦИПИАЛЬНАЯ СХЕМА

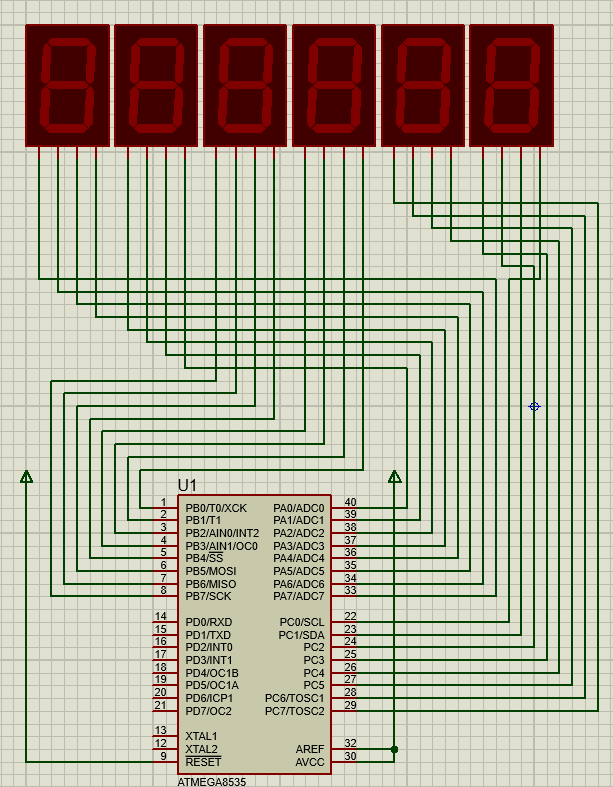


Рисунок - Принципиальная схема

# ЛИСТИНГ ПРОГРАММЫ

.def sign=r10

.def exp=r9

.def mh=r8

.def ml=r7

; макрос для удобной загрузки значения в младший банк регистров

.macro lldi

ldi @0, @2

mov @1, @0

.endmacro

; макрос для удобного применения маски на регистры младшего банка

.macro andli

ldi @0, @2

and @1, @0

.endmacro

; макрос для удобной загрузки адреса в пару регистров из младшего банка

.macro lldiw

lldi @0, @1, high(@3)

lldi @0, @2, low(@3)

.endmacro

; вектор прерываний

rjmp start

.org $30

; 3 байтовое число с плавающей точкой

arr: .db 0xC0, 0xFF, 0xFC, 0

; in : sign, exp, mh, ml

; out: -

bit\_iter:

push r16 ; начало пролога, сохранение значений используемых регистров

in r16, sreg

push r16

push r17 ; конец пролога

bit\_it:

mov r17, mh ; применение маски для вычисление старшего бита мантисы

andi r17, $80

cp r17, sign ; если бит знака противоположен биту мантиссы, то выходим

brne bit\_it\_end

inc exp ; увеличиваем порядок во время смещения

mov r17, exp ; наложение маски на порядок во избежание переполнения

cpi r17, $7F ; при достижении специального значения - выходим

breq bit\_it\_end

lsl mh ; смещаем старший байт мантиссы

sbrc ml, 7 ; перемещаем биты из младшего байта мантиссы в старший

inc mh

lsl ml ; смещаем младший байт мантиссы

sbrc sign, 7 ; заполняем освободившиеся биты младшего байта мантиссы в соответствии биту знака

inc ml

rjmp bit\_it ; зацикливаемся

bit\_it\_end:

pop r17 ; начало эпилога, восстановление значения

pop r16

out sreg, r16

pop r16 ; конец эпилога и возврат

ret

; in : sign, exp, mantissa

; out: -

print: ; процедура вывода на индикаторы

push sign ; начало пролога

push r16

in r16, sreg

push r16 ; конец пролога

ser r16

OUT DDRA, r16 ; настройка портов

out ddrb, r16

out ddrc, r16

mov r16, sign ; формирование байта со знаком и порядком

or r16, exp

out porta, r16 ; вывод байт на индикаторы

out portb, mh

out portc, ml

pop r16 ; начало эпилога

out sreg, r16

pop r16

pop sign ; конец эпилога и возврат

ret

; in : r0:r1 - PM float addr

; out: loaded sign, exp, mantissa

load: ; функция загрузки

push r16 ; начало пролога, сохранение состояния

in r16, sreg

push r16

push zh

push zl ; конец пролога

movw zh:zl, r0:r1 ; задание регистра косвенной адресации

lpm exp, Z+ ; загрузка данных в регистры

lpm mh, Z+

lpm ml, Z

mov sign, exp ; применение маски для получение отдельно бита знака

andli r16, sign, $80

andli r16, exp, $7F

pop zl ; начало эпилога

pop zh

pop r16

out sreg, r16

pop r16 ; конец эпилога и возврат

ret

start: ; точка входа

ldi r16, high(ramend) ; инициализация стека

out sph, r16

ldi r16, high(ramend)

out spl, r16

lldiw r16, r1, r0, arr << 1 ; передача аргументов в процедуру

rcall load ; загрузка

rcall bit\_iter ; нормализация

rcall print ; вывод

end:

rjmp end ; окончание работы программы

# РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Результат выполнения программы для $C0FFFC приведен ниже на рисунках 4 и 5.

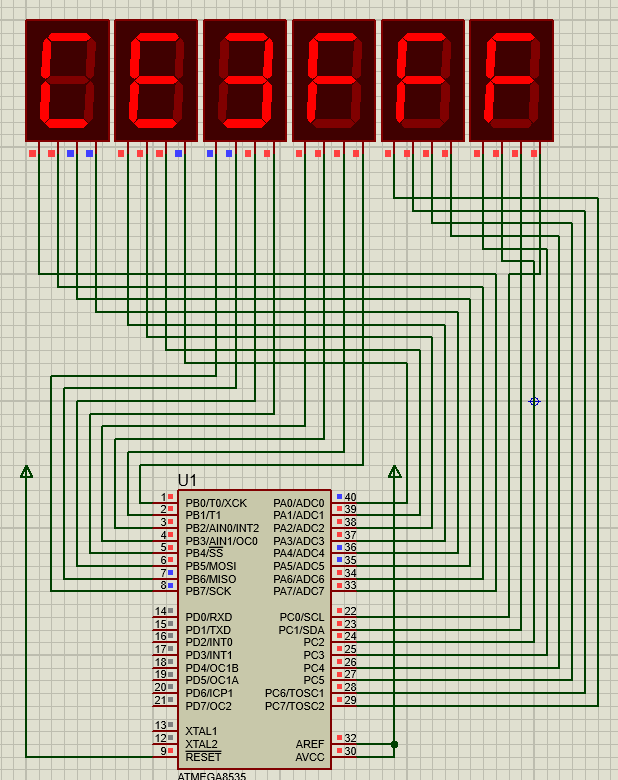


Рисунок - Результат выполнения выведенный на индикаторы

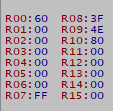


Рисунок - Результат выполнения в регистрах

В результате работы программы было нормализовано число с плавающей точкой со знаком.