МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)

Институт информационных технологий, математики и механики
Кафедра прикладной математики

Направление подготовки: «Математика и компьютерные науки»

ОТЧЁТ

по учебной практике
на тему
Пример программирования в кодах процессора MYP128M

Выполнила: студентка группы 381605
__________________ А. А. Сарыкова
Подпись

Научный руководитель:
кандидат физико-математических наук, доцент
__________________ В. С. Гаврилов
Подпись

Нижний Новгород
2018 г.

Оглавление

Введение

Данная работа посвящена реализации в машинных кодах процессора MYP128M некоторых численных методов одномерной оптимизации. А именно написана программа, реализующая метод дихотомии. Описание методов взято из [1], описание процессора MYP128M - из [2].

Работа состоит из введения и двух глав. В первой главе находятся необходимые сведения из теории оптимизации. А вторая - посвящена реализации методов дихотомии.

Первая глава состоит из двух разделов. В первом разделе упоминается определение унимодальной функции и её основное свойство. Во втором разделе приводится краткое описание метода дихотомии.

Вторая глава состоит из шести разделов. В первом из разделов приводятся необходимые определения, заменяющие имена регистров. Во втором представлены ассемблерные макросы, упрощающие написание программы. В третьем приводится текст подпрограммы, реализующий метод дихотомии. Эта подпрограмма вызывается из основной программы. В четвёртом разделе представлена подпрограмма, реализующая минимизируемую функцию. В пятом приведён текст основной программы. В последнем разделе представлен шеснадцатиричный дамп исполняемого файла программы.

Глава 1. Сведения из теории оптимизации

1.1. Унимодальные функции

Материал данного раздела взят из [1, гл.1, опр.7].

Определение 1.1.1. *[1, гл.1, опр.7] Функция $f(x)$ непрерывная на $[a, b]$ называется унимодальной, если найдутся числа α, β из R такие, что*

1) $f(x)$ строго убывает на $[a, \alpha]$;

2) $f(x)$ минимизируема на $[\alpha, \beta]$;

3) $f(x)$ строго возрастает на $[\beta, b]$.

Один или два из этих отрезков могут иметь нулевую длину.

$$f(x) \to min, x \in [a, b] \tag{1.1.1}$$

Определение 1.1.2. *Отрезком неопределённости называется всякий отрезок $[c, d] \subset [a, b]$ такой, что $[c, d] \cap [\alpha, \beta] \neq \varnothing$.*

Определение 1.1.3. *Решением задачи (1.1.1) с точностью $\varepsilon > 0$ называется отрезок неопределённости $[c, d]$, такой, что $|d - c| < 2\varepsilon$*

Лемма 1.1.1. ***(Основное свойство унимодальных функций):*** *Пусть $f(x) : [a, b] \to \mathbb{R}$ - унимод. Если $z_1, z_2 \in (a, b), z_1 < z_2$, то, вычислив $f(z_1)$ и $f(z_2)$, можно найти отрезок неопределённости $[c, d] \subset [a, b]$ такой, что $|d - c| < |b - a|$. Именно:*

1) если $f(z_1) < f(z_2)$, то $[a, z_2]$ - отрезок неопределённости;

2) если $f(z_1) > f(z_2)$, то $[z_1, b]$ - отрезок неопределённости;

3) если $f(z_1) = f(z_2)$, то $[z_1, z_2]$ - отрезок неопределённости.

1.2. Метод дихотомии

В данном разделе напомним метод дихотомии [1, гл.1, пар.3] В данном методе имеется параметр $\lambda > 0$, причём $0 < \lambda < \varepsilon$, где ε требуемая точность вычислений. Метод дихотомии следует схеме, сформулированной в лемме 1.1.1, при этом точки z_1, z_2 выбираются следующим образом:

$$z_1 = \frac{a + b}{2} - \lambda, \qquad z_2 = \frac{a + b}{2} + \lambda$$

Замечание 1.2.1. ***(оно касается всех методов, основанных на лемме):*** *можно сказать, что из-за приближённого характера вычислений случай $f(z_1) = f(z_2)$ (см. лемму) никогда не реализуется. Т.е. бывает либо 1) $f(z_1) < f(z_2)$, либо 2) $f(z_1) > f(z_2)$. Поэтому от отрезка неопределённости $[a, b]$ мы переходим либо к отрезку неопределённости $[a, z_2]$, либо - к $[z_1, b]$.*

Глава 2. Реализация метода дихотомии

2.1. Определения

Чтобы не использовать имена регистров введём следующие определения на языке программирования ассемблер:

```
func    equ r1
a       equ f3
b       equ f4
lambda  equ f5
epsilon equ f6
c       equ f0
d       equ f2
m       equ f7
z1      equ f8
z2      equ f9
F1      equ f10
F2      equ f11
len     equ f12
temp    equ r2
```

2.2. Нужные макросы

Для упрощения написания программы нам потребуется ряд ассемблерных макросов. Это макросы для вывода на экран строки, чтения и печати вещественного числа, выхода из программы.

```
macro printStr строка, длина
{
push r1-r2
movu r1, строка
movu r2, длина
movu r0, 1
trap 0x80
pop r1-r2
}
```

Сначала приведён макрос printStr для вывода на экран строки, у которого указываются два аргумента: адрес выводимой строки и её длина. При вызове макроса все регистры сохраняются.

```
macro readF
{
push r0-r0
```

```
movu r0, 2
trap 0x80
pop r0-r0
}
```

Второй макрос readF предназначен для чтения вещественного числа. На этом шаге программа приостанавливается и ожидает ввода пользователем вещественного числа, которое заносится в регистр f0.

```
macro printF
{
push r0-r0
movu r0, 3
trap 0x80
pop r0-r0
}
```

Следующий макрос printF для печати вещественного числа. Программа выводит то число, которое было введено пользователем и содержится в регистре f0.

```
macro exit0
{
movu r0, 4
movu r1, 0
trap 0x80
}
```

Последний макрос exit0 для выхода из программы. Программа успешно завершается с кодом ошибки 0, то есть без ошибок.

2.3. Подпрограмма метода дихотомии

Ниже приведён текст подпрограммы реализации метода дихотомии, при этом аргументы a, b, λ, ε заносятся, согласно определениям в пункте 2.1, в соответствующие регистры f3, f4, f5, f6. Ссылка на функцию заносится в регистр r1. Результат работы подпрограммы дихотомии будет в регистре r0.

```
дихотомия:
push f3-f12
push r1-r2
add epsilon, epsilon, epsilon
цикл:
subf len, b, a
absf len, len
cmpf temp, len, epsilon
jmpsr temp, выход
addf m, a, b
```

```
mul2f m, m, -1
subf z1, m, lambda
addf z2, m, lambda
mov f0, z1
call func
mov F1, f0
mov f0, z2
call func
mov F2, f0
cmpt temp, F1, F2
jmpger temp, больше_или_равно
mov b, z2
jmpr цикл
больше_или_равно:
jmpzr temp, равно
mov a, z1
jmpr цикл
равно:
mov a, z1
mov b, z2
jmpr цикл
выход:
mov c, a
mov d, b
pop r1-r2
pop f3-f12
ret
```

2.4. Минимизируемая функция

В качестве минимизируемой функции возьмём функцию $f(x) = x^3 - x = x * (x^2 - 1)$. Приведён текст подпрограммы. Аргумент функции заносится в регистр f0. После выполнения функции её значение находится также в f0.

```
func1:
push f1-f2
mulf f1, f0, f0
fld1 f2
subf f1, f1, f2
mulf f0, f1, f0
pop f1-f2
ret
```

2.5. Основная программа

Приведён текст основной программы.

```
printStr 0x4140,0x1A
readF
mov a, f0
readF
mov b, f0
readF
mov lambda, f0
readF
mov epsilon, f0
mov func, func1
callr дихотомия
printStr 0x415A,0x14
printF
exit0
```

2.6. Дамп программы

Ниже представлен шестнадцатеричный дамп исполняемого файла программы.

```
0000:0000  CC D3 D0 31  32 38 CC 00  00 40 00 00  00 00 00 00
0000:0010  00 00 00 00  00 00 00 00  00 04 00 00  00 00 00 00
0000:0020  00 00 00 00  00 00 00 00  00 44 00 00  00 00 00 00
0000:0030  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
0000:0040  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
0000:0050  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
0000:0060  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
0000:0070  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00
0000:0080  00 00 00 00  00 00 00 00  C1 B0 88 00  C0 10 C1 40
0000:0090  C0 11 00 1A  C0 10 00 01  84 00 00 80  C1 C0 88 00
0000:00A0  C1 B0 00 00  C0 10 00 02  84 00 00 80  C1 C0 00 00
0000:00B0  C1 31 80 00  C1 B0 00 00  C0 10 00 02  84 00 00 80
0000:00C0  C1 C0 00 00  C1 32 00 00  C1 B0 00 00  C0 10 00 02
0000:00D0  84 00 00 80  C1 C0 00 00  C1 32 80 00  C1 B0 00 00
0000:00E0  C0 10 00 02  84 00 00 80  C1 C0 00 00  C1 33 00 00
0000:00F0  C0 10 81 24  82 30 00 0E  C1 B0 88 00  C0 10 C1 5A
0000:0100  C0 11 00 14  C0 10 00 01  84 00 00 80  C1 C0 88 00
0000:0110  C1 B0 00 00  C0 10 00 03  84 00 00 80  C1 C0 00 00
0000:0120  C0 10 00 04  C0 10 80 00  84 00 00 80  C1 B1 B0 00
0000:0130  C1 10 88 00  40 03 18 C0  40 16 10 60  40 A6 30 00
0000:0140  40 41 30 C0  80 F1 00 15  40 03 8C 80  40 B3 9F E0
0000:0150  40 14 1C A0  40 A4 9C A0  C1 30 20 00  81 50 80 00
```

```
0000:0160  C1 35 00 00  C1 30 24 00  81 50 80 00  C1 35 80 00
0000:0170  40 41 29 60  81 31 00 03  C1 32 24 00  80 EF FF EF
0000:0180  81 01 80 03  C1 31 A0 00  80 EF FF EC  C1 31 A0 00
0000:0190  C1 32 24 00  80 EF FF E9  C1 30 0C 00  C1 30 90 00
0000:01A0  C1 20 88 00  C1 C1 B0 00  82 A0 00 00  C1 B0 88 00
0000:01B0  40 20 80 00  C1 D1 00 00  40 10 84 40  40 20 04 00
0000:01C0  C1 C0 88 00  82 A0 00 00  00 00 00 00  00 00 00 00
0000:01D0  D0 92 D0 B2  D0 B5 D0 B4  D0 B8 D1 82  D0 B5 20 61
0000:01E0  2C 62 2C CE  BB 2C CE B5  3A 0A D0 A0  D0 B5 D0 B7
0000:01F0  D1 83 D0 BB  D1 8C D1 82  D0 B0 D1 82  3A 0A
```

Заключение

В данной работе был изучен один из методов одномерной оптимизации, а именно метод дихотомии. Разработанный алгоритм поиска экстремума заданной функции перенесён сначала на язык ассемблера, а затем на машинный код процессора МУР128М. Эта программа, написанная для реализации метода половинного деления, в дальнейшем послужит тестом для эмулятора МУР128М.

Список литературы

[1] Васильев Ф.П. Численные методы решения экстремальных задач / Васильев Ф.П. —2-е изд. — М.: Наука, 1988. — 552с.

[2] Гаврилов В.С. Процессор МУР128М (машина учебная регистровая, 128–разрядная) – Режим доступа: `http://github.com/gavr-vlad-s/mur128m`