Математические функции с использованием целочисленной арифметики для RISCV

АО Микрон 2026

Оглавление

Автор: Жораев Тимур Юлдашевич

Каждый параграф проиндексирован соответствующим номером. Для замечаний указывать данное число.

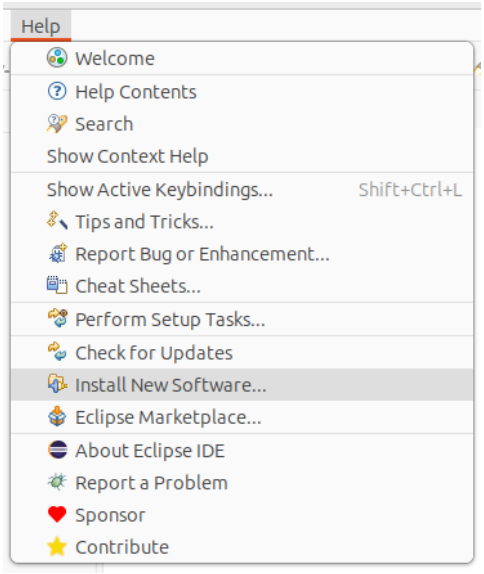
UUID документа 019b0cdc-e816-78b8-b89a-34fbf6d578ff

1 Использование симулятора

Для изучения работы команд целевой платформы RISC-V, произвести отладку математических функций, других алгоритмов, не требующих какой-либо связи с периферией и не жёстко привязанных к реальному времени, можно использовать симулятор с кросс-компиляцией и согласующей библиотекой для исполняемой платформы.

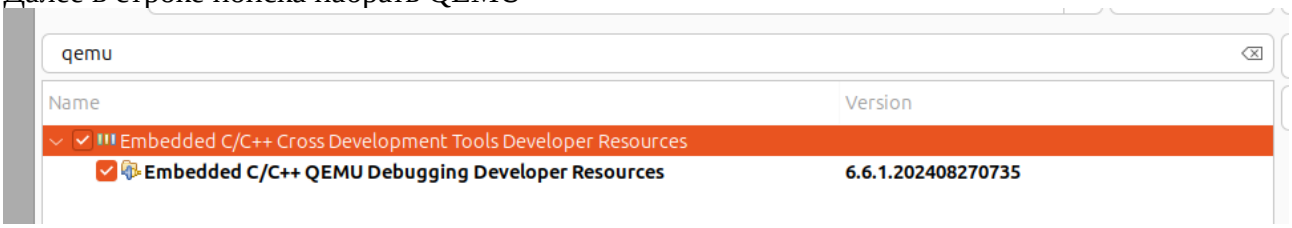
Для установки симулятора потребуется следующее.

1. Вначале установить необходимое расширение

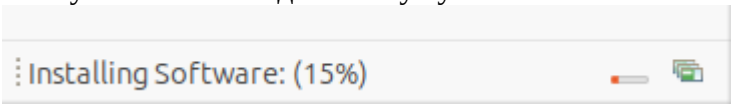


Далее выбрать все доступные сайты для поиска

Далее в строке поиска набрать QEMU

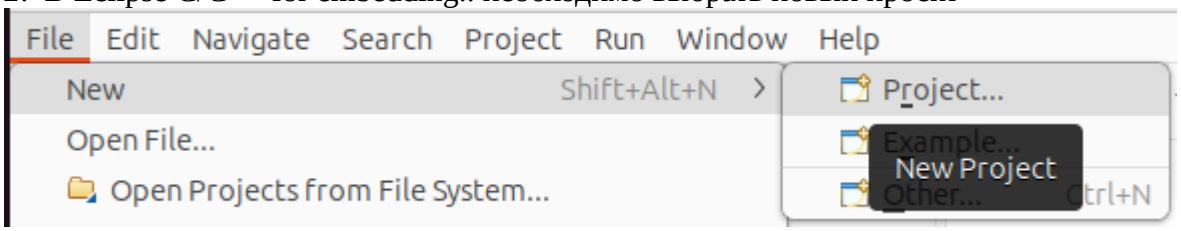


Выбрать установить средство для отладки
Внизу можно наблюдать статус установки

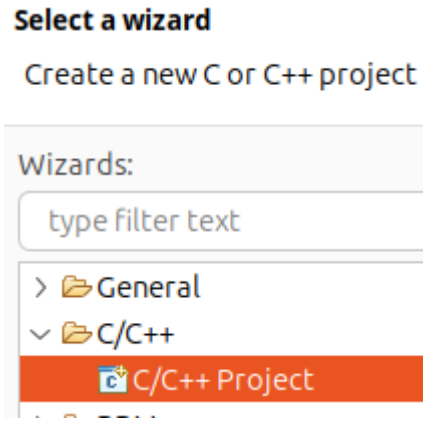


После завершения программа предложит перезапуск среды разработки. Перезапустить.

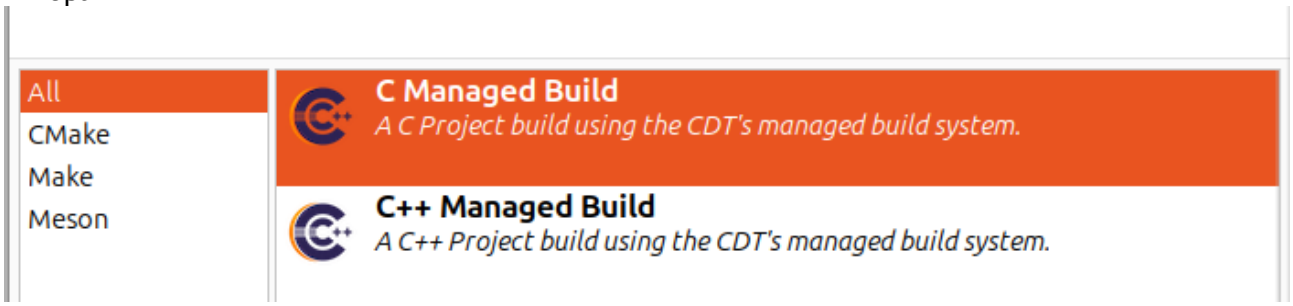
2. В Eclipse C/C++ for embedding.. необходимо выбрать новый проект



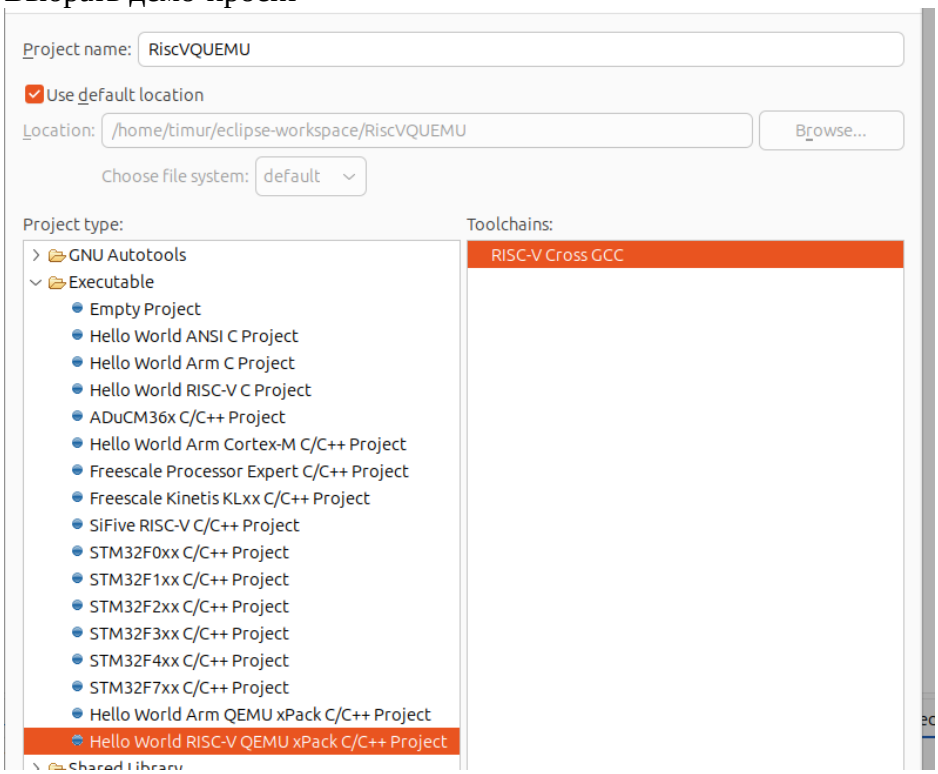
Мастер проекта:



Выбрать

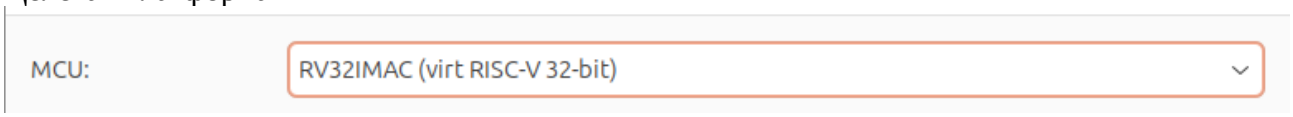


Выбрать демо-проект

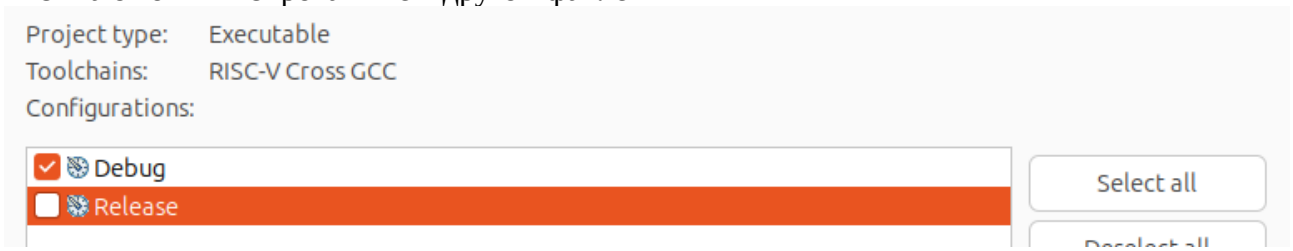


Примечание. Дополнительно этот и другие проекты располагаются по адресу https://gitflic.ru/project/mikron-mik32/mik32-examples.git

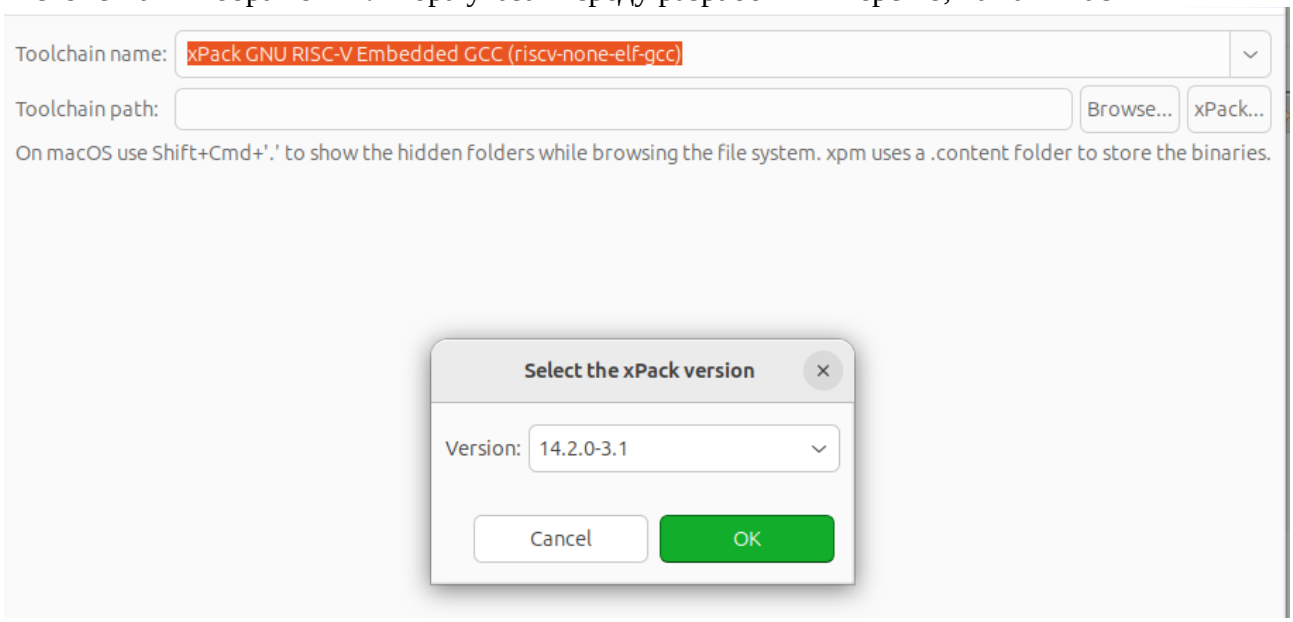
Целевая платформа



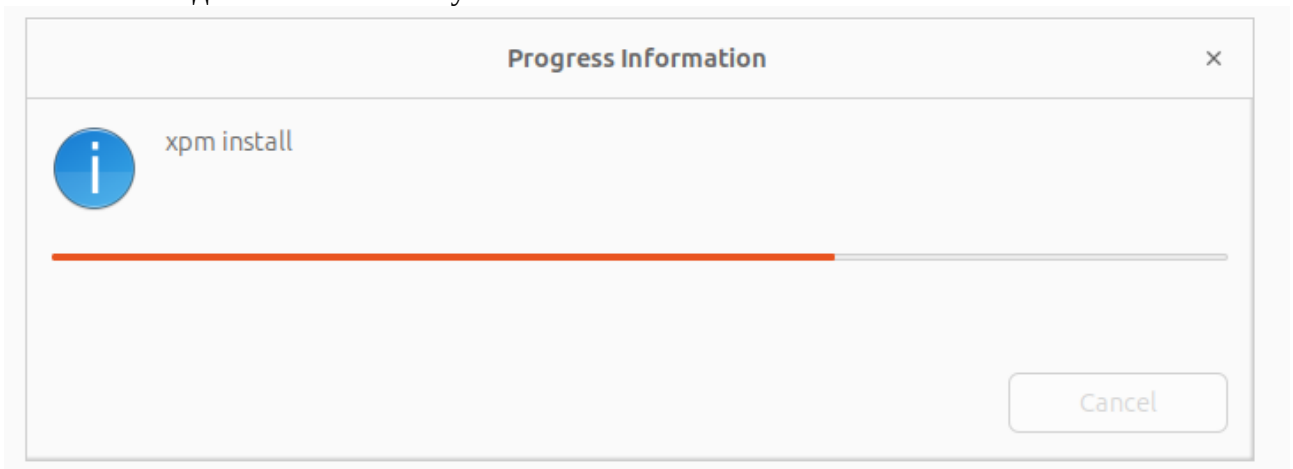
Можно указать только отладку, оптимизацию включать отдельно, например, чтобы были видны глобальные символы или локальные в не оптимизированных функциях, которые вызывают оптимизированные в другом файле



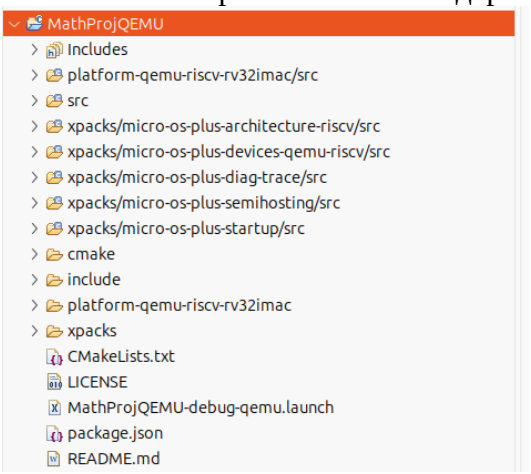
В свойствах выбора компилятора указать среду разработки и версию, нажав xPack



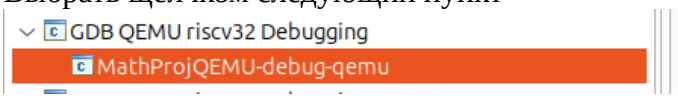
Возможно потребуется подождать, так как происходит загрузка дополнительных пакетов, это может занять до нескольких минут и более



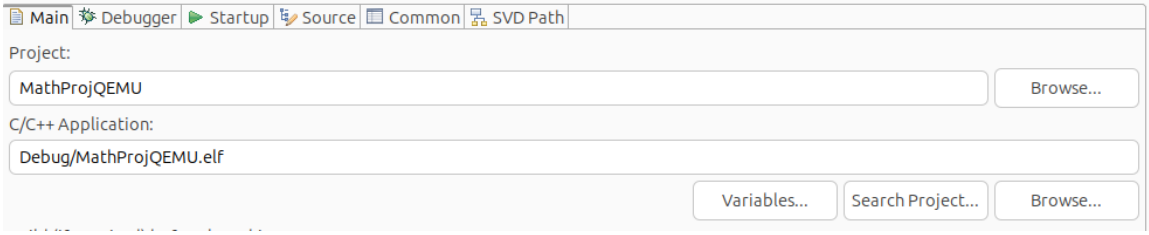
3. После чего проект появится в дереве:



Необходимо создать создать среду отладки в режиме эмулятора. Нажать  Выбрать щелчком следующий пункт



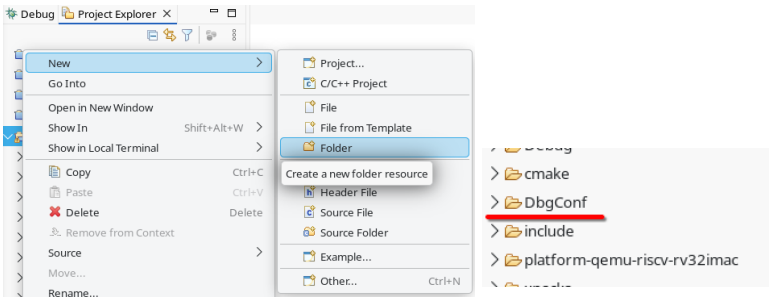
Появится соответствующее меню настройки, убедиться, что имеется выбор файла приложения elf для отладки



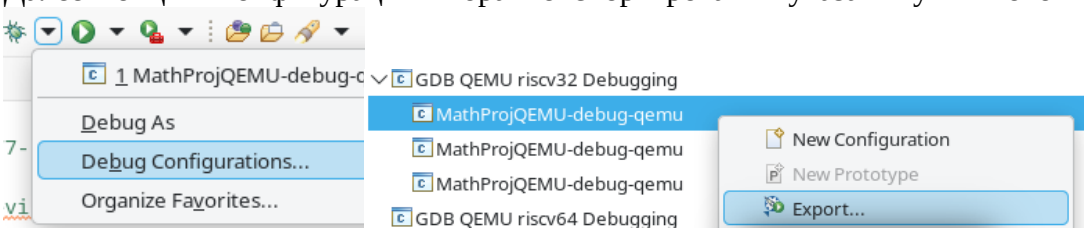
Остальные опции принимаются по умолчанию

4. Чтобы в дальнейшем опции отладки копировались в новый проект, желательно создать папку внутри проекта, наименование произвольное, например, DbgConf. Далее экспортировать туда файл отвечающий за конфигурацию отладки.

Создать папку



Далее в опциях конфигурации выбрать экспортировать и указать путь к новой папке



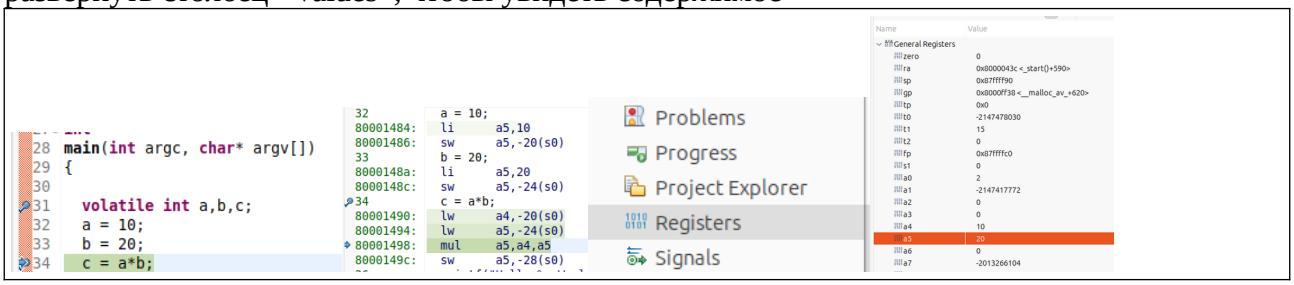
5. Далее можно вносить код на своё усмотрение. Например, можно исследовать умножение, написав следующее в теле функции main:

```
volatile int a,b,c;
a = 10;
b = 20;
c = a*b;
```

```
27 int
28 main(int argc, char* argv[])
29 {
30
31   volatile int a,b,c;
32   a = 10;
33   b = 20;
34   c = a*b;
35
36   printf("Hello %s World!" "\n", argc > 1 ? argv[1] : "Arm");
37
```

Поставить точку остановки на выражении c = a*b.

Запустить отладку. Также включить режим  пошагового следования в дизассемблере, а также посмотреть содержимое эмулируемых регистров. Следует отметить что необходимо развернуть столбец «Values», чтобы увидеть содержимое



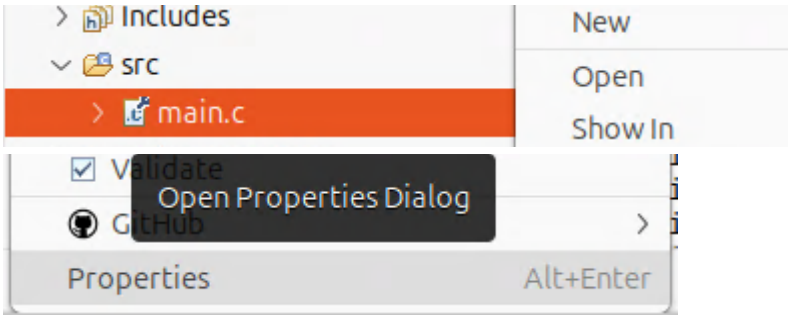
2 Основные математические операции

Для работы потребуется проект math-project-1. Основной файл — main.c, в нём приведён пример работы и тестирование арифметики.
Переменные объявлены как volatile чтобы компилятор их не оптимизировал, не рассчитывал оконечное выражение и не подставлял его в качестве ответа.

Предварительная настройка проекта. Для того, чтобы сформировать листинг ассемблера и дополнительные символы, необходимые для отладки отдельного файла, не задевая остальные библиотеки, средства Eclipse позволяют сформировать настраиваемые этапы построения для отдельных файлов и папок. В данном случае необходим отдельный выбор настроек для файла main.c. Несмотря на то что проект изначально настроен, необходимо проверить соответствующие параметры настроек.

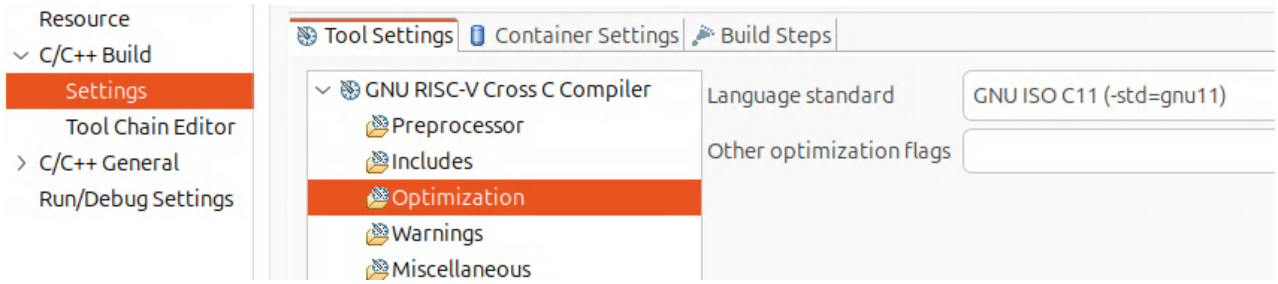
6. Установка необходимых флагов для сборки отдельного файла.

Выбрать свойства файла main.c правой кнопкой



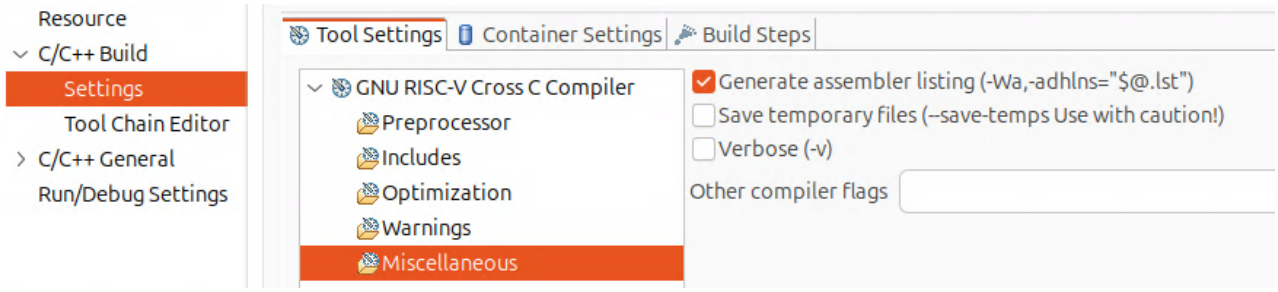
опция Properties:

Для оптимизации можно назначить отдельное значение флагов



например, -O2, по размеру, по скорости и др. В данном случае это поле оставляется по умолчанию, наследуя флаги из свойств проекта.

Чтобы просмотреть листинг на языке Ассемблера, который получается после кодогенерации, необходимо включить опцию



Generate assembler listing. Следует отметить, что данный файл остаётся в папке при использовании «Очистить проект». Поэтому, необходим дополнительный скрипт или удаление вручную lst файлов, чтобы они не загромождали проект.

2.1 Общие сведения о режиме отладки с дизассемблером при работе с аппаратурой

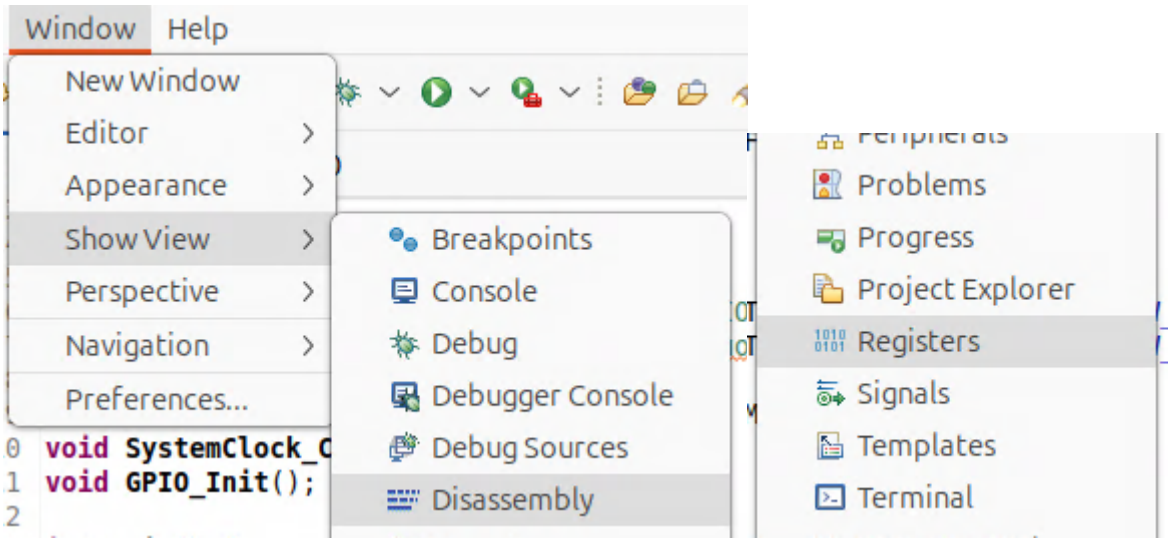
В качестве примера используется плата с МК «Амур», наименование проекта «math-project-1». UUID проекта 019c2777-2ed0-7ab9-8469-85fe89e31e63.

7. Поставить точку остановки на выражении c = a+b. Запустить. Открыть меню просмотра регистров и листинга дизассемблера.

Поставить точку остановки и запустить:

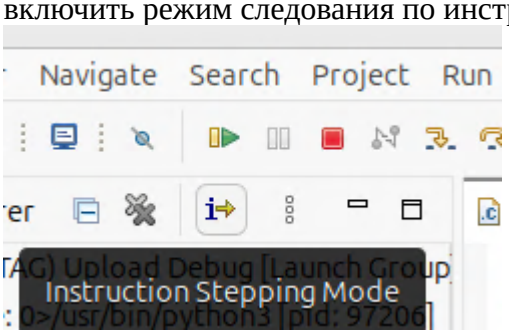
```
13 int main() {
14     SystemClock_Config();
15
16     GPIO_Init();
17
18     volatile int a = 0x10000;
19     volatile int b = 0x200;
20     volatile int c;
21     //тестирование простого сложения
22     c = a + b;
23     asm(" nop");
```

Выбрать меню просмотра регистров и дизассемблирования

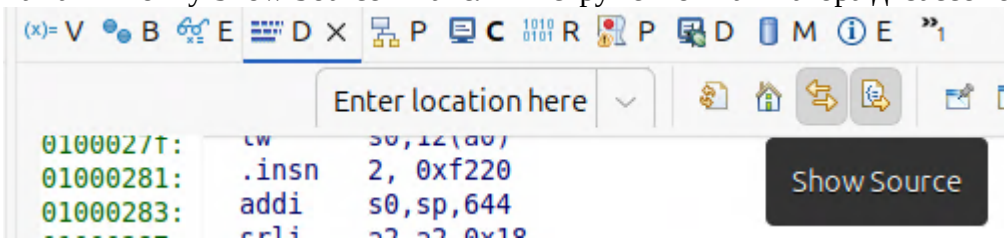


8. Анализ результата по шагам.

Прежде всего открыть окно дизассемблера. В нём также будет показана строка остановки, соответствующая с-файлу.
Чтобы иметь возможность анализа результата по шагам в окне дизассемблера необходимо включить режим следования по инструкциям на панели инструментов



Тогда при нажатии клавиши «F6» - сделать шаг, выполнение будет идти по листингу дизассемблера, в противном случае — по строкам исходного кода «C» с пропуском.
Также, чтобы видеть одновременно листинг ассемблера и исходного кода, необходимо нажать кнопку Show Source в панели инструментов навигатора дизассемблера.



2.2 Сложение

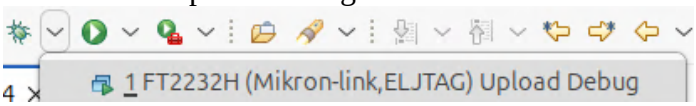
9. Как было сказано ранее, поставить точку остановки на сложении:

```
22      c = a + b;
23      asm(" nop");
```

Можно также открыть вкладку точек остановки и выявить какие из них активны

Иногда, точку остановки скрывают различного рода предупреждения и чтобы её дезактивировать можно также использовать это меню а также горячую клавишу.

Выполнить Upload-Debug



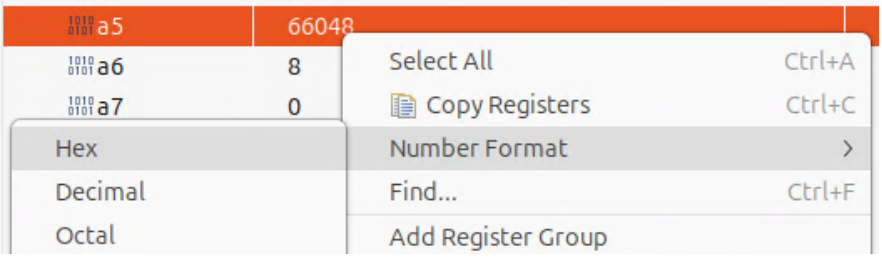
Не забыть также включить режим нажатием кнопки  следования по инструкциям ассемблера

10. После чего произвести пошаговую отладку нажатием F6:

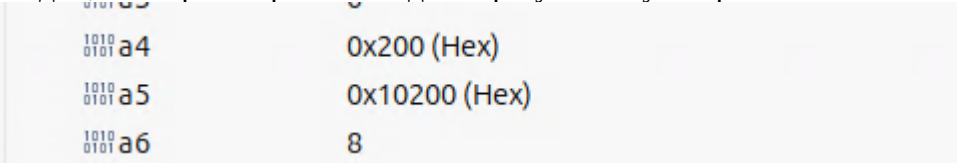
```
22          c = a + b;
 01000290:  lw      a5,4(sp)
 01000292:  lw      a4,8(sp)
 01000294:  add     a5,a5,a4
 01000296:  sw      a5,12(sp)
23            asm(" nop");
 01000298:  nop
```

Инструкция на ассемблере nop введена специально в исходный код на «С» чтобы идентифицировать разделение между примерами в коде и завершения заданной операции.

11. В окне регистров можно преобразовать формат в шестнадцатеричный



Видно, что в регистре a5 находится результат суммирования



12. Произвести отладку переполнения. Вновь выбрать формат «Десятичный» для регистров a4,a5 (возможны другие).

```
0100029a:   lui     a5,0x80000
0100029e:   addi    a5,a5,-1 # 0x7fffffff
010002a0:   sw      a5,4(sp)
26            b = 1;
010002a2:   li      a5,1
010002a4:   sw      a5,8(sp)
27            c = a + b;
010002a6:   lw      a5,4(sp)
010002a8:   lw      a4,8(sp)
010002aa:   add     a5,a5,a4
010002ac:   sw      a5,12(sp)
28            asm(" nop");
010002ae:   nop
```

```
24    //тестирование переполнения
25    a = 0x7FFFFFFF;
26    b = 1;
27    c = a + b;
28    asm(" nop");
```

В окне дизассемблера видно, что идёт подготовка к сложению. Данный листинг зависит от флагов сборки проекта, связанных с оптимизацией компилятора. Следует обратить на них внимание.

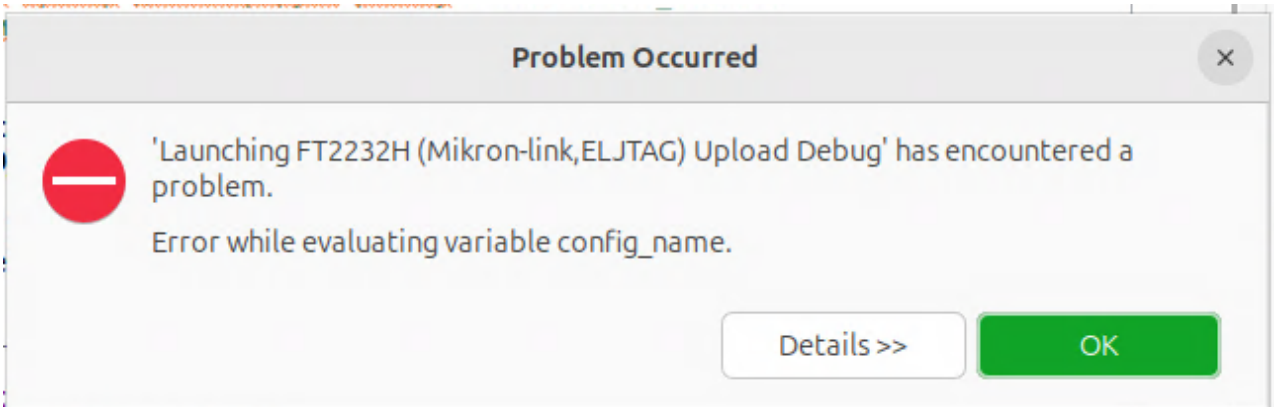
Значение регистра результата до и после выполнения команды суммирования

a4	1 (Decimal)
a5	2147483647 (Decimal)

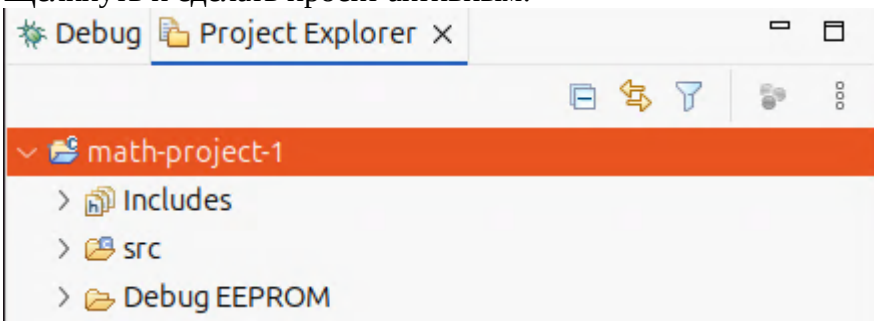
a4	1 (Decimal)
a5	-2147483648 (Decimal)

Изменение регистра/памяти в активной области окна просмотра подсвечивается жёлтым

13. При возникновении следующей ошибки, установить активный проект в менеджере проектов Project Explorer



Щёлкнуть и сделать проект активным:



2.3 Умножение 16 бит

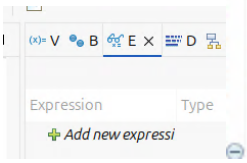
Для тестирования умножения применяются глобальные ячейки памяти, в которых можно изменить значение, и, с использованием команды установки шага, можно без остановки отладки реализовать проход инструкций заново.

14. Поставить точку остановки на «тестирование умножения 16 бит со знаком»

```
32    //тестирование умножения 16 бит со знаком (формат q15)
33    register int ra, rb, rc;
34    volatile int rd;
35    ra = ma;
36    rb = mb;
37    rc = ra * rb;
38    rd = rc >> 16;
39    asm(" nop");
```

15. Окно Watch для просмотра и редактирования глобальных и локальных символов (в области видимости {}).

Окно просмотра переменных:




```
volatile int ma = 0;
volatile int mb = 0;

int main() {
```

Добавить 2 глобальные переменные ma, mb, в поле Value записать значения, например 400 и 500

Expression	Type	Value
ma	volatile int	0
mb	volatile int	0

Expression	Type	Value
ma	volatile int	400
mb	volatile int	500

16. Произвести запуск до точки остановки и просмотреть результат работы команды умножения и арифметического сдвига в регистре

Содержимое регистров до mul:

```
35        ra = ma;
010002b0: lw      a5,-2012(gp)
36        rb = mb;
010002b4: lw      a4,-2016(gp)
37        rc = ra * rb;
010002b8: mul     a5,a5,a4
```

a4	500 (Decimal)
a5	400 (Decimal)

После:

a4	500 (Decimal)
a5	200000 (Decimal)

Далее следует команда арифметического сдвига вправо, также представлен результат инструкции:

```
010002b8: mul     a5,a5,a4
38          rd = rc >> 16;
010002bc: srai    a5,a5,0x10
010002be: sw      a5,12(sp)
39          asm(" nop");
010002c0: nop
```

a4	500 (Decimal)
a5	3 (Decimal)

Для проверки можно использовать умножение в шестнадцатеричной форме:

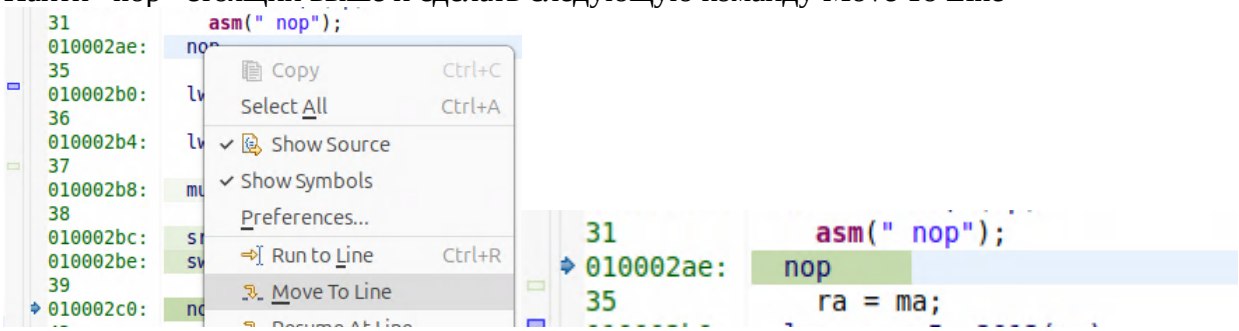
Множитель 1	Множитель 2	Результат
400	500	200000
190	1F4	**3**`0D40

Видно, что старшая тетрада (разделена апострофом `) в ответе «30D40» имеет значение 3 и после сдвига на 16 разрядов (4 значащих цифры в шестнадцатеричном представлении) получается искомый результат 3.

17. Задание значений заново с изменением точки отладки

Для того, чтобы вновь пройти без перезагрузки с новыми числами, необходимо заново встать на исходную загрузку операндов из памяти в регистры. Для этого необходимо изменить счётчик программ.

Найти «nop» стоящий выше и сделать следующую команду Move To Line

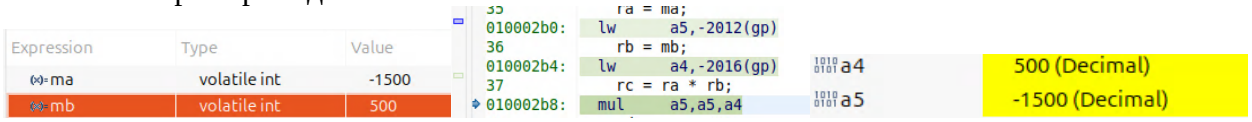


После чего курсор указателя программ переместится на точку выше.

Важно! Данный приём следует использовать при гарантии отсутствия реакции команд с учётом флагов статуса. В основном к ним чувствительны команды условного перехода. Также следует исключить какие-либо прерывания при данном способе отладки или влияние аппаратуры.

18. Применение значений и повторный ход вычислений

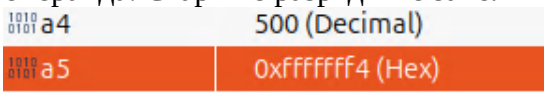
В качестве примера задать значения



Перед командой умножения операнды загружены в регистр.

В ходе умножения и арифметического сдвига получается следующий результат

Обратить внимание. Арифметический сдвиг в отличие от двоичного, сохраняет знак операнда. Старшие разряды не заполняются нулями:



19. Арифметические действия

Иллюстрация работы команды умножения

Множитель 1	Множитель 2	Результат
-1500	500	-750000
FFFF`FA24	01F4	**FFF4**`8E50

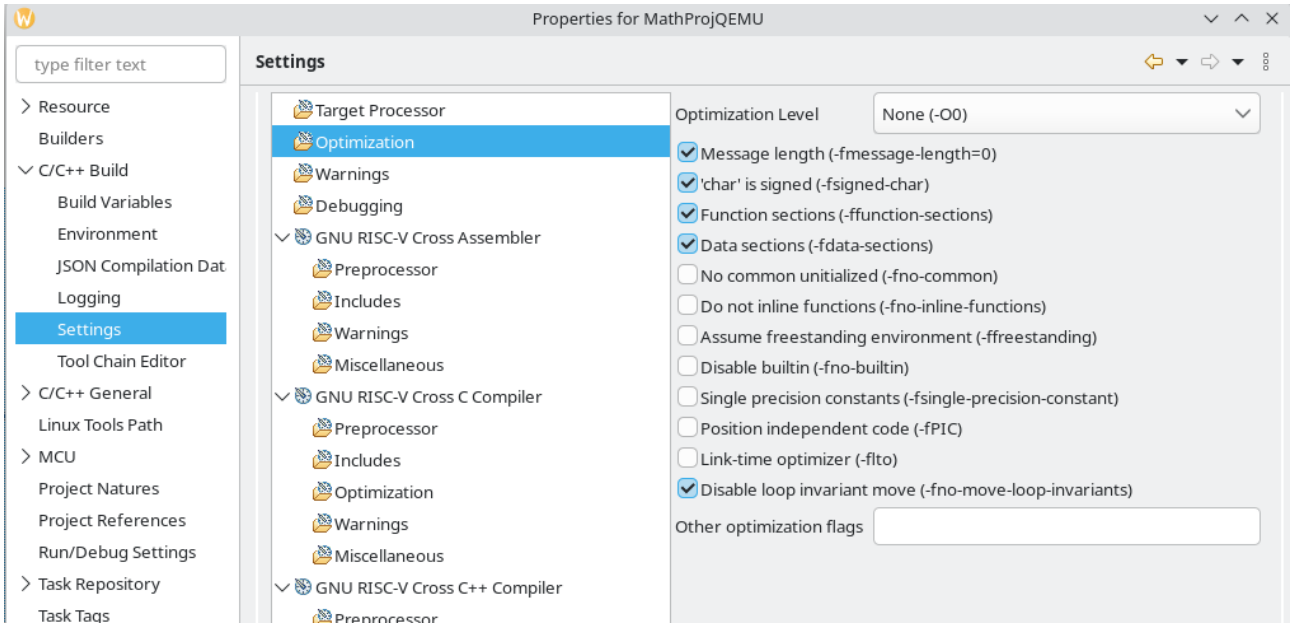
Иллюстрация работы команды арифметического сдвига вправо

Операнд	Сдвиг	Результат
-750000	16 бит	-12
FFF4`8E50	16 бит	FFF4

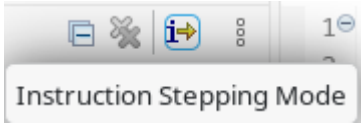
2.4 Умножение 32 бит в формате с фиксированной точкой в заданном Q-формате.

В качестве примера проекта необходимо использовать проект MathProjQUEMU с UUID 019b0cdd-5747-76f9-9c77-557d92c081ad

В этом примере рассматриваются основные математические операции. Оптимизация в виде опций компилятора существенно влияет на конечный результат, поэтому, используются переменные с ключевым словом volatile, чтобы избежать автоматических вычислений и исключения тестируемых инструкций. В этом случае переменная может быть размещена как в памяти, так и в стеке или регистре. Для размещения в памяти можно использовать ключевое слово static или размещение в глобальной области, для регистра — register, стековые переменные обычно создаются автоматически внутри функции.
20. Опции оптимизации — их можно проверить или установить в соответствующем меню. Достаточно вызывать свойства проекта.



21. Аналогично примеру выше, приводится пример «простого умножения». Следует отметить, что пошаговая отладка ассемблерного кода производится с включённой кнопкой




```
volatile int a,b,c;
asm(" nop");
a = 10;
b = 20;
c = a*b;
```

->

```
20          asm(" nop");
80001480:   nop
21          a = 10;
80001482:   li      a1,10
80001484:   sw      a1,-20(s0)
22          b = 20;
80001488:   li      a1,20
8000148a:   sw      a1,-24(s0)
23          c = a*b;
8000148e:   lw      a0,-20(s0)
80001492:   lw      a1,-24(s0)
80001496:   mul     a1,a0,a1
8000149a:   sw      a1,-28(s0)
24          asm(" nop");
```

22. Использование окна просмотра переменных в различных форматах

Как уже было указано, чтобы просмотреть значение одной и той же переменной в различных форматах необходимо сделать формальную добавку к имени, например, приведение к типу самой переменной «из int в int», в противном случае к одинаковому имени применяется один и тот же формат, например, шестнадцатеричный или десятичный. Пример приведён на рисунке.

(x)= a	10 (Decimal)
(x)= b	20 (Decimal)
(x)= c	200 (Decimal)
(x)= (long)a	0xa (Hex)
(x)= (long)b	0x14 (Hex)
(x)= (long)c	0xc8 (Hex)

23. Использование q-формата представления чисел и их просмотр.
Для преобразования форматов используются следующие макросы

#define Float2Q26(x) ((long)(((x)*1.0)*(1<<26)))	Преобразование из плавающей запятой в q-формат с 26 дробными разрядами
#define Q262Float(x) ((float)(x)/(float)(1<<26))	Преобразование числа в q-формата с 26 дробными разрядами в формат представления с плавающей запятой

Математически преобразование чисел осуществляется умножением/делением на два в соответствующей степени с отсечением дробной части. При этом, степень двух заменяется арифметическим сдвигом влево на заданное количество разрядов. В данном случае, формат Q26, поэтому смещение осуществляется на 26 разрядов.

Можно записать что преобразование числа x из формата с плавающей запятой в Q-формат осуществляется как $q=\lfloor x\cdot 2^{26}\rfloor$, где скобки $\lfloor\ \rfloor$ - отсечение дробной части числа, в данном случае, в десятичном представлении (в общем виде — для числа с произвольным основанием системы счисления). Обратное преобразование осуществляется как $x=q/2^{26}$. Число q всегда является целым. При этом, положение запятой определяется формально, она может быть в любом месте. Её положение по факту закрепляется правилами перевода чисел из одного формата в другой и операциями умножения/деления. Линейная операция сложения или вычитания работает с данными числами без изменения. Базовая схема реализация числа в формате представлена на рис. 1.

Рис. 1. Представление формата q26

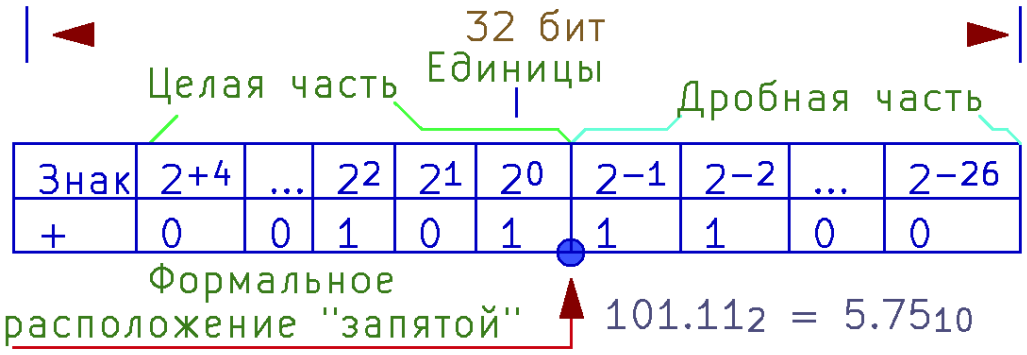


Рис. 2. Пример числа в формате q26

На рис. 2 представлено число в заданном формате, его общий вид
$0001\ 0.111\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = 1700\ 0000_{16} = 385875968_{10} = 5.75_{q26}$
Следует отметить, что $385875968/2^{26} = 23/4 = 5.75$

24. Отладка чисел в q-формате

В примере для отладчика можно увидеть результат преобразования. Также, необходимо использовать конструкцию вида `(float)переменная/(1<<26)` в окне просмотра переменных, чтобы в ячейке сразу показывался результат преобразования из целого в формат q26 или иной другой.

```
c = 0b00010111000000000000000000000000;
c = 0x17000000;
c = 385875968;
c = Float2Q26(5.75);
```

(x)= c	385875968 (Decimal)
(x)= (int)c	10111000000000000000000000000000
(x)= (float)c/(1<<26)	5.75
(x)= (int) c	0x17000000 (Hex)
(x)= (int)(5.75*(1<<26))	385875968

```
c = 0b00010111000000000
lui     a1,0x17000
sw      a1,-28(s0)
c = 0x17000000;
lui     a1,0x17000
sw      a1,-28(s0)
c = 385875968;
lui     a1,0x17000
sw      a1,-28(s0)
c = Float2Q26(5.75);
lui     a1,0x17000
sw      a1,-28(s0)
```

Обратить внимание на ассемблерную загрузку одного и того же числа в различных системах счисления.

25. Для ввода числа в q-формате при изменении переменной необходимо использовать конструкцию вида `(int)(число*(1<<26))`. Возможно, использование макроса или плагина для реализации подобного рода действий с автоматизацией ввода-вывода в необходимом q- формате.

Для других чисел в примере можно видеть следующие фрагменты окна переменных

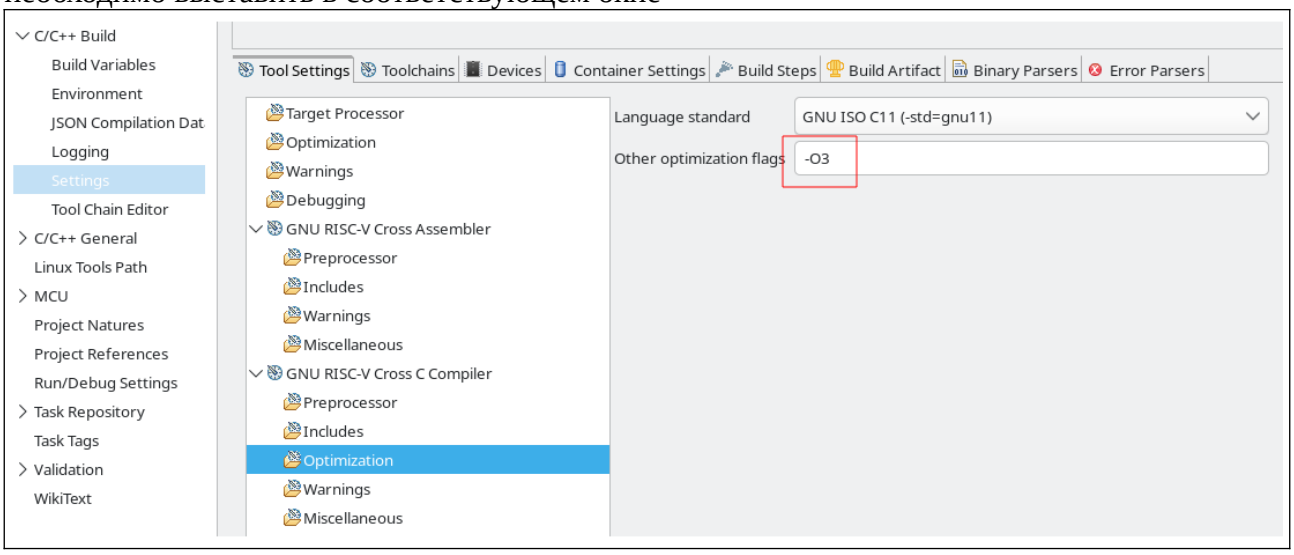
```
a = Float2Q26(PI_f/2);
b =  Float2Q26(0.45);
```

(x)= (float)a/(1<<26)	1.57079637
(x)= (float)b/(1<<26)	0.449999988
(x)= (float)c/(1<<26)	5.75
(x)= a	105414357 (Decimal)
(x)= b	30198988 (Decimal)
(x)= c	385875968 (Decimal)

2.5 Умножение чисел в q-формате

Рис. 3. Умножение двух числе в q-формате

Для уменьшения кода используем опцию оптимизации компилятора -O3. Для этого её необходимо выставить в соответствующем окне



26. Для загрузки чисел в регистры используются сдвоенные инструкции, как показано на примере ниже. В листинге дизассемблера можно видеть соответствующие числа

```
a = Float2Q26(PI_f/4);
lui     a5,0x3244
addi    a5,a5,-150 # 0x3243f6a
sw      a5,4(sp)
b =  Float2Q26(0.8);
lui     a5,0x3333
addi    a5,a5,819 # 0x3333333
sw      a5,8(sp)
```

```
a = Float2Q26(PI_f/4);
b =  Float2Q26(0.8);
```

(float)a/(1<<26)	0.785398126
(float)b/(1<<26)	0.800000012
a	52707178 (Decimal)
b	53687091 (Decimal)
(long)a	0x3243f6a (Hex)
(long)b	0x3333333 (Hex)

$\frac{\pi}{4} \approx 0.7853981633974483, \lfloor \frac{\pi}{4} \cdot 2^{26} \rfloor = 52707178_{10} = 3243F6A_{16}$

$\lfloor 0.8 \cdot 2^{26} \rfloor = 53687091_{10} = 03333333_{16}$

27. Для умножения необходимо использовать следующие макросы

```
#define MULch26(x,y) (((long long)(x)*(long)(y))>>26)
#define MULch24(x,y) (((long long)(x)*(long)(y))>>24)
```

Структурная схема умножения представлена на рис. 3. Правило достаточно простое — осуществляется умножение с получением числа с двойной разрядностью, осуществляется арифметический сдвиг влево для коррекции разрядов и выбор только старшей части числа. В этом случае происходит некоторая потеря точности. Процессоры DSP при операции умножения с накоплением осуществляют сохранение значащих разрядов с расширением, чтобы избежать потери точности, для этого существуют «теневые» биты аккумулятора. Таким образом, для промежуточных вычислений, требующих улучшенную точность рационально складывать результат умножения с учётом правых разрядов.
В RISC-V существует пара команд умножения — получения младшей mul и старшей части результата mulh. Возможно использование только mulh, при этом, увеличивается быстродействие, однако, происходит потеря точности. В настоящей реализации используется пара команд для получения результата 64 бита, при коррекции результата осуществляется арифметический сдвиг влево и разряды в младшей части сохраняются.
Общая структурная схема умножения представлена на рис. 4.

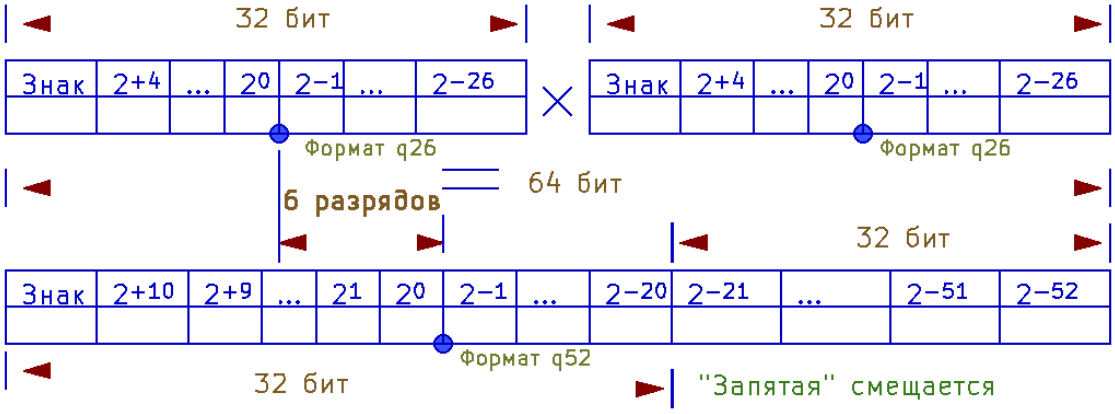


Рис. 4. Пример умножения чисел в формате q26

28. Далее производится коррекция результата с применением смещения старшей части младшей части на количество разрядов, определяемое как 32 (всего) - 26 (формат) = 6 разрядов. Пример представлен на рис. 5.

Рис. 5. Коррекция результата для выделения старшей части результата совместно с остаточными битами

На рис. 6 показана реализация

Рис. 6. Реализация коррекции с использованием смещения старшей и младшей частей в отдельных регистрах

29. На рис. 7 представлен листинг дизассемблера и описание основных шагов по инструкциям.

В соответствующие переменные в стеке загружаются необходимые значения

```
a = Float2Q26(PI_f/4);
lui     a5,0x3244
addi    a5,a5,-150 # 0x3243f6a
sw      a5,4(sp)          1
b =  Float2Q26(0.8);
lui     a5,0x3333
addi    a5,a5,819 # 0x3333333
sw      a5,8(sp)          2
c = MULch26(a,b);
lw      a5,4(sp)
lw      a3,8(sp)          3
mul     a4,a5,a3
mulh    a5,a5,a3          4
srli    a4,a4,0x1a
slli    a5,a5,0x6         5
or      a4,a4,a5          6
sw      a4,12(sp)         7
```

Шаг 1 и 2 — загрузка регистров перед первым каскадом умножений
Шаг 3 — выгрузка из стека значений. В регистре a3 значение 0.8, в регистре a5 — значение $\pi/4$ a3 53687091 a4 0 a5 52707178
Шаг 4 — умножение для получения старшей и младшей частей результата После команды mul в регистре a4 значение 0xb8f8c01e После команды mulh в регистре a5 значение 0xa0d97.
Шаг 5 — арифметический свдиг регистровой пары после команды srli в регистре a4 значение 0x2e после slli в регистре a5 0x28365c0
Шаг 6 — коррекция младших разрядов с использованием дополнения от сдвинутого значения

Рис. 7. Ассемблерный листинг инструкций умножения

30. На шаге 4 производится умножение командой mul для младшей части и mulh для старшей. Потенциально для большего упрощения можно использовать только mulh с определённой потерей точности.

$52707178 \cdot 53687091 = 2829695061639198_{10} = 0A0D97B8F8C01E_{16}$

Разбиение полученного результата на старшую и младшую часть по 32 бита, младшая часть подчёркнута линией:

0A 0D97 B8F8 C01E

Видно, что в окне отладчика имеется отрицательное значение для регистра a4, однако, необходимо обратить внимание на шестнадцатеричное. Для этого необходимо навести курсор на регистр и посмотреть дополнительную информацию внизу


```
Name : a4
    Hex:0xb8f8c01e
    Decimal:-1191657442
    Octal:027076140036
    Binary:10111000111110001100000000011110
    Default:-1191657442
```

31. В результирующем регистре младшая часть результата умножения. Так как бит знака может быть при этом любым, оно должно быть интерпретировано именно как целое беззнаковое значение, относящееся к общему результату.

Далее производится умножение командой mulh которая выделяет старшую часть результата


```
Name : a5
    Hex:0xa0d97
    Decimal:658839
    Octal:02406627
    Binary:10100000110110010111
    Default:658839
```

32. Далее производится коррекция результата и приведение его вновь к формату q26.

Теоретически значение в формате q52 будет равно для значения 64 бита:

$2829695061639198_{10}/2^{52} \approx 0.6283185220199878$

При этом

$\pi/4 \cdot 0.8 \approx 0.6283185307179586$

Если взять только старшую часть и перевести её согласно правилу преобразования можно получить

$a0d97_{16} = 658839_{10} = 0.009817466139793396_{q26}$

однако, $a0d97_{16} = 0.6283178329467773_{q20} = \left(658839_{10} \cdot \frac{2^6}{2^{26}}\right)_{q26}$

33. Коррекция осуществляется арифметическим сдвигом вправо (!) младшей части на значение 0x1a=26 разрядов инструкцией srli (srli a4,a4,0x1a):

$0xb8f8c01e_{16} >> 26 = 3103309854_{10} >> 26 = 0x2e_{16} = 46_{10}$.

В этом случае в регистре остаются старшие значащие биты младшей части.

Коррекция старшей части осуществляется с использованием сдвига влево на 6 разрядов командой slli (slli a5,a5,0x6):

$0x28365c0_{16} = 42165696_{10} \approx 0.6283178329467773_{q26}$, разница с идеальным значением $7 \cdot 10^{-7}$

В этом случае младшие разряда старшей части заполняются нулями.

34. Производится коррекция младших разрядов старшей части путём выполнения команды or — двоичное «или», при этом осуществляется дополнение младших разрядов (фактически, сложение).

$0x28365ee_{16} = 42165742_{10} = 0.6283185184001923_{q26}$, разница с идеальным значением скорректированного путём добавления младших разрядов $0.12 \cdot 10^{-7}$.

```
0x28365c0 10100000110110010111000000 | двоичное «или»
                          0x2e     00000000000000000000101110     =
0x28365ee 10100000110110010111101110
```

В случае если можно пренебречь младшей частью то можно обойтись без её расчёта и коррекции сдвигом. В этом случае берётся только старшая часть результата.

2.6 Деление

35. Деление осуществляется следующим образом.
Имеются два числа, представленные в формате Q26, например,

a = Float2Q26(1.21);

b = Float2Q26(0.43);

Известно, что деление может быть осуществлено по формуле (1), где $\lfloor x \rfloor$ - целая часть числа.

$$y = \frac{a}{b} = \frac{\mathrm{mod}(a,b)}{b} + \lfloor \frac{a}{b} \rfloor \qquad (1)$$

36. Осуществляется обычное целочисленное деление для извлечения целой части результата командой div:

c = a / b;

После чего находится остаток от деления командой rem:

r = a % b;

Производится децимация делителя до значения разрядностью вдвое меньшей, то есть в 16 бит:

d16 = (b >> 16);

Находится частичный остаток от деления, при этом остаток от деления в формате q26:

p = r / d16;

Осуществляется коррекция результата частичного остатка от деления до q26:

p = p << 10;

Приведение целой части от деления к формату q26:

cq = c << 26;

Сложение с частичным остатком:

c = cq + p;

Листинг дизассемблера для арифметического действия деления с описанием инструкцией представлен на рис. 8.

37. Вычисление деления, листинг дизассемблера

`a = Float2Q26(1.21);`
`lui a5,0x4d71`
`addi a5,a5,-1475 # 0x4d70a3d`
`sw a5,12(sp)`
`b = Float2Q26(0.43);`
`lui a5,0x1b85`
`addi a5,a5,491 # 0x1b851eb`
`sw a5,16(sp)`
	1. Производится загрузка в регистры a5 — делимое и a4 — делитель. Значения 1.21 и 0.43.
	Исходные значения
	$a5 = 81201725_{10} = 1.209999993443489_{q26}$
	$a4 = 28856811_{10} = 0.4299999922513962_{q26}$

`c = a / b;`
`lw a5,12(sp)`
`lw a4,16(sp)` 1
`div a5,a5,a4` 2
`sw a5,20(sp)`
	2. После выполнения инструкции деления получается целая часть результата в регистре a5 в формате q0 (без дробной)
	a5 = 2

`d = a % b;`
`lw a5,12(sp)`
`lw a4,16(sp)` 3
`rem a5,a5,a4` 4
`sw a5,24(sp)`
	3. Загружаются операнды для нахождения остатка от деления как в п. 1.

`e = d / (b >> 16);`
`lw a5,24(sp)`
`lw a4,16(sp)` 5
`srai a4,a4,0x10` 6
`div a5,a5,a4` 7
`sw a5,28(sp)`
	4. Рассчитывается остаток от деления, ответ в регистре a5 (теоретически mod(1.21,0.43)=0.35)
	$a5 = 23488103_{10} = 0.3500000089406967_{q26}$

`e = e << 10;`
`lw a5,28(sp)` 8
`slli a5,a5,0xa` 9
`sw a5,28(sp)`
	5. Загрузка в регистр a4 делителя по п. 1
	$a4 = 28856811_{10} = 0.4299999922513962_{q26}$

`c = (c<<26) + e;`
`lw a5,20(sp)`
`lw a4,28(sp)` 10
`slli a5,a5,0x1a` 11
`add a5,a5,a4` 12
`sw a5,20(sp)`
	6. Децимация делителя путём его арифметического сдвига вправо на 16 разрядов (0x10) $a4 = 440_{10} = 0x1b8_{16}$

	7. Производится деление остатка от деления на прореженный делитель, получается частичный остаток от деления $a5 = 53382_{10} = 0xd086_{16}$
	8. Производится загрузка частичного остатка от деления по п. 7. $a5 = 53382_{10} = 0xd086_{16}$
	9. Осуществляется коррекция к формату q26 частичного остатка от деления путём арифметического сдвига влево на 26 разрядов (0x1a) $a5 = 54663168_{10} = 0x3421800_{16} = 0.814544677734375_{q26}$
	10. Загружаются регистры a5,a4, содержащие частичный остаток от деления и целую часть в формате q0 $a4 = 54663168_{10} = 0.814544677734375_{q26}$ a5 = 2
	11. Производится приведение целой части в формат q26: $a5 = 0x8000000_{16} = 134217728_{10} = 2.0_{q26}$
	12. Осуществляется суммирование целой части и частичного остатка от деления с формированием результата $a5 = 188880896_{10} = 2.814544677734375_{q26}$ Теоретическое значение ≈ 2.813953488, разница $6 \cdot 10^{-4}$

Рис. 8. Схема вычисления действия деления

3 Тригонометрические функции

38. Чтобы создать отдельный проект, необходимо произвести его копирование. В Eclipse это можно сделать открыв проект на основе которого создаётся новый, правой клавишей копировать затем вставить с новым именем в рабочую область. Убедиться что символические ссылки также скопированы. Возможно также потребуется редактирование файлов проекта, которые содержат имя предыдущего с соответствующим переименованием.

3.1 Синус

39. В общем виде используется функция синуса от аргумента $\sin(x)$. Для вычисления синуса используется таблица его значений от 0 до 90º = $\frac{\pi}{2}$. Количество точек равно 65, включая 0 и значение равное 1 при угле 90º. Аргумент в зависимости от индекса вычисляется по (2):

$$x=\frac{2\cdot\pi\cdot f\cdot i}{fd}, i=\frac{fd\cdot x}{2\cdot\pi\cdot f} \tag{2}$$

При этом задаётся соответственно количество точек и производится нормировка к четверти периода $f=\frac{1}{4}, fd=64$. При значении i равном 64 аргумент в точности равен $\frac{\pi}{2}$, то есть четверти периода.

График функции $\sin\left(\frac{2\ \pi\ f\ i}{fd}\right)$ в точках $i=i0$ (зелёным цветом) а также в виде непрерывной линии (промежуточные значения) представлен на рис. 9.

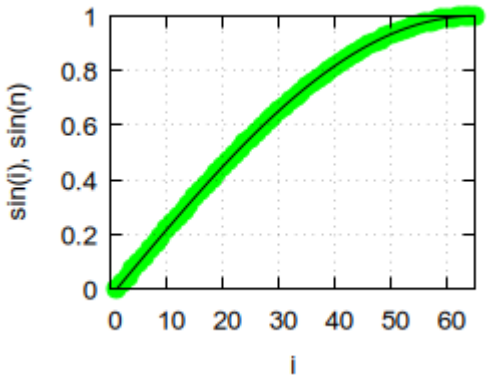


Рис. 9. Синусоидальная функция на четверти периода

40. Разложение в ряд Тейлора второго порядка в окрестности точки $i0$. Введём необходимые обозначения переменных

$$\sin\left(\frac{2\cdot\pi\cdot f\cdot i0}{fd}\right)=ST, \cos\left(\frac{2\cdot\pi\cdot f\cdot i0}{fd}\right)=CT \tag{3}$$

41. Запишем разложение синуса в ряд Тейлора второго порядка в окрестности точки $i0$ как (4):

$$S(i)=ST\cdot\left(1-2\left(\frac{\pi f}{fd}\right)^2(i-i0)^2\right)+CT\cdot\left(2\frac{\pi f}{fd}(i-i0)\right) \tag{4}$$

42. Аналогично, разложение косинуса в ряд Тейлора 2 порядка в окрестности точки $i0$ будет представлено в виде (5):

$$C(i)=CT\cdot\left(1-2\left(\frac{\pi f}{fd}\right)^2(i-i0)^2\right)-ST\cdot\left(2\frac{\pi f}{fd}(i-i0)\right) \tag{5}$$

При подстановке $i=i0$ можно убедиться, что значение в точности равно по таблице, для (4) это ST, для (5) - CT соответственно.

43. Построим график функции ряда Тейлора, например, в окрестности точки, соответствующей восьмой части периода, в этом случае $i0=N/2=64/2=32$.

В этом случае ряд будет иметь вид, с учётом того, что синус и косинус определены для значений $\pi/4$ в замкнутом виде. В данной точке разложение будет иметь вид (6)

$$-\left(\frac{\frac{\pi^2 i^2}{\sqrt{2}}-2^{\frac{11}{2}}\pi^2 i-2^{\frac{15}{2}}\pi i+2^{\frac{19}{2}}\pi^2+2^{\frac{25}{2}}\pi-2^{\frac{29}{2}}}{32768}\right) \tag{6}$$

График функции (6) совместно с исходным синусом (чёрная линия) представлен на рис. 10.

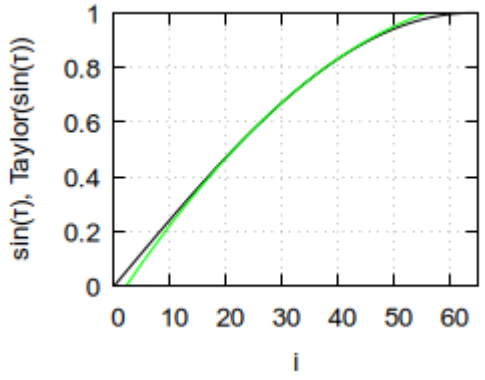


Рис. 10. Разложение в ряд Тейлора в окрестности точки i = 32 и идеальная синусоида

44. Для нахождения погрешности, можно воспользоваться подстановкой $i=i0+1$; и $i=i0-1$в выражение для разложения синуса в ряд Тейлора. Затем, взять разницу между идеальным значением и аппроксимируемым. Аппроксимируемые выражения представлены в виде примера в (7).

$$S(i0+1)=-\left(\frac{32768\ \sin\left(\frac{\pi\ (i0+1)}{128}\right)+\pi^2\ \sin\left(\frac{\pi\ i0}{128}\right)-32768\ \sin\left(\frac{\pi\ i0}{128}\right)-256\ \pi\ \cos\left(\frac{\pi\ i0}{128}\right)}{32768}\right)$$
$$S(i0-1)=\left(\frac{\pi^2\ \sin\left(\frac{\pi\ i0}{128}\right)-32768\ \sin\left(\frac{\pi\ i0}{128}\right)+256\ \pi\ \cos\left(\frac{\pi\ i0}{128}\right)+32768\ \sin\left(\frac{\pi\ (i0-1)}{128}\right)}{32768}\right) \quad (7)$$

График выражений $S(i0)-S(i0\pm1)$ из формул (7) представлен на рис. 11.

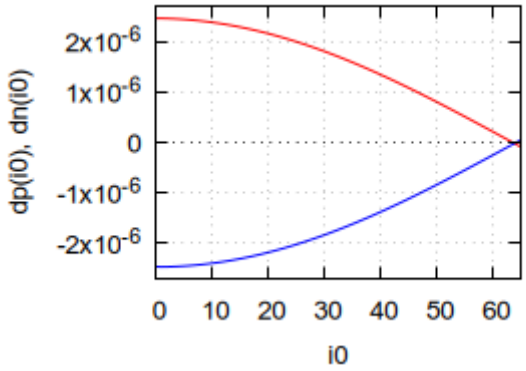


Рис. 11. Разница между идеальным значением в точке границы диапазона индекса и аппроксимируемым значением рядом Тейлора

45. Видно, что максимальная ошибка в этом случае определяется в точке i0=0, поэтому, можно записать, что максимальная ошибка в этом случае для полинома второго порядка определяется как (8).

$$E=\frac{2\ \pi\ f}{fd}-\sin\left(\frac{2\ \pi\ f}{fd}\right) \quad (8)$$

При подстановке численных параметров можно получить $2.464083257970595\ 10^{-6}$, при этом, ошибка квантования по уровню с учётом разрядности 32 бит равна $1/2^{32}=1.4901161193847656\ 10^{-8}$.

3.2 Косинус

Определяется аналогично синусу, с той разницей что используется разложение (6).

4 Функции для нахождения амплитуды и фазы

В данном разделе рассматриваются функции, которые необходимы для определения амплитуды и фазы комплексного числа — это корень квадратный и четырёхквадрантный арктангенс.

4.1 Корень квадратный

46. Функция корня квадратного описана в файле Maxima, UUID 019b5979-02ce-7ebf-80e5-9a86a456fc58.

47. Корень вычисляется с использованием алгоритма Ньютора-Рафсона с табличной выборкой значений равными $1/\sqrt{x0}$. Сам квадратный корень в конечном итоге вычисляется согласно тождества $\sqrt{x}=x/\sqrt{x}$, путём умножения входной величины на величину обратную её корню квадратному. Определим функцию f как (9):

$$f=\frac{1}{y^2}-x0 \tag{9}$$

Корень квадратный в точке $x0$ есть решение уравнения (10) относительно y или $f=0$.

$$f=\frac{1}{y^2}-x0=0 \tag{10}$$

Производная от левой части выражения (10) равна (11):

$$\frac{d}{dy}f=\frac{d}{dy}\left(\frac{1}{y^2}-x0\right)=-\frac{2}{y^3} \tag{11}$$

48. Далее осуществляется итеративный процесс нахождения значения с учётом начального условия $y_0=1/\sqrt{x0}$. Метод Ньютона заключается в нахождении последовательных итераций согласно выражению (12)

$$y_n=y_{n-1}-\frac{y_{n-1}}{\frac{d}{dy}y_{n-1}} \tag{12}$$

С учётом (11), (12) можно записать (13):

$$y_n=\frac{y_{n-1}^3\left(\frac{1}{y_{n-1}^2}-x0\right)}{2}+y_{n-1}=-\left(\frac{y_{n-1}\left(y_{n-1}^2\ x0-3\right)}{2}\right) \tag{13}$$

Одна и две итерации вычисляются раскрытыми выражениями, получаемыми последовательными подстановками и представленными в (14):

$$y_1=-\left(\frac{y_0\left(y_0^2\ x0-3\right)}{2}\right),\ y_2=\frac{y_0\left(y_0^2\ x0-3\right)\left(y_0^6\ x0^3-6\ y_0^4\ x0^2+9\ y_0^2\ x0-12\right)}{16} \tag{14}$$

49. Для составления таблицы используем индексацию. Предположим, что при значении индекса равным половине точек $idx = N/2$ значение функции равно 1 $\frac{1}{\sqrt{b + 32\ a}} = 1$. Для удобства составления таблицы потребуем $y = 1/\sqrt{1/m}$, при этом $m = 2^8 + 1$, получаем $\frac{1}{\sqrt{b}} = \sqrt{257} = 16.0312195418814$.

Исходные уравнения для индексации (15):

$$x0 = a\ idx + b,\ idx = \frac{x0 - b}{a},\ y_0 = \frac{1}{\sqrt{a\ idx + b}} \tag{15}$$

Решением уравнений (15) для коэффициентов прямой является (16):

$$\left[a = \frac{8}{257}, b = \frac{1}{257} \right] x0 = \frac{8\ idx + 1}{257} \tag{16}$$

Также, исходя из (16) можно записать (17):

$$\left[idx = \frac{257\ x0 - 1}{8} \right] [idx = 32.125\ x0 - 0.125] \tag{17}$$

50. С учётом того, что значения корректируются методом последовательных итераций, можно выбрать приближённое значение индекса как, которое полностью определяется только операцией арифметического сдвига влево, как представлено в (18).

$$idx = 32\,x0 = x0 \ll 5 \tag{18}$$

Умножение на 32 есть сдвиг на 5 разрядов. При этом, исходная функция для составления таблицы выглядит как, представлена функция и первые три её значения при изменении индекса от 0 . Таблица составляется для значения индекса от 0 до 64, область значений от 0 до 2 по аргументу корня. $\frac{\sqrt{257}}{\sqrt{8\ idx + 1}} \left[\sqrt{257}, \frac{\sqrt{257}}{3}, \frac{\sqrt{257}}{\sqrt{17}}, ... \right]$

51. Для реализации значений свыше 2, а именно, от 2 до 32 используется аппроксимация с использованием полинома $p(x)$, который требует только операции умножения и сложения.
Для синтеза этого полинома потребуем следующее. Общий вид представлен в выражении (19).

$$p(x) = c - a\ x\ (1 - b\ x\ (1 - d\ x)) \tag{19}$$

52. Воспользуемся методом наименьших квадратов (НК).

Ошибка вычисляется как $\int_{x1}^{x2} (p(x) - 1/\sqrt{x})^2\, dx$, который равен (20) в общем виде с учётом $[x2 > x1, x2 > 0, x1 > 0]$ можно получить (20):

$$(210\log(x2)+30a^2b^2d^2x2^7-70a^2b^2dx2^6+84a^2bdx2^5+42a^2b^2x2^5-105abcdx2^4-105a^2bx2^4+120abdx2^{7/2}+140abcx2^3+70a^2x2^3-168abx2^{5/2}-210acx2^2+280ax2^{3/2}+210c^2x2-840c\sqrt{x2}-210\log(x1)-30a^2b^2d^2x1^7+70a^2b^2dx1^6-84a^2bdx1^5-42a^2b^2x1^5+105abcdx1^4+105a^2bx1^4-120abdx1^{7/2}-140abcx1^3-70a^2x1^3+168abx1^{5/2}+210acx1^2-280ax1^{3/2}-210c^2x1+840c\sqrt{x1})/210 \quad (20)$$

С учётом заданных параметров пределов интегрирования $x1=1, x2=33$ можно записать (21):

$$(1278553289280a^2b^2d^2-124521600abcd-90402757760a^2b^2d+3287372928a^2bd+120\,33^{7/2}abd-120abd+6720c^2+5031040ab-228480ac-840\sqrt{33}c+840c+1643686464a^2b^2-124521600a^2b-1abd+168abd+2515520a^2+280\,33^{3/2}a-280a+210\log(33))/210 \quad (21)$$

Далее находятся частные производные от (21), из которых можно составить систему уравнений (22):

$$\frac{\partial}{\partial a}(Q1)=0, \frac{\partial}{\partial b}(Q1)=0, \frac{\partial}{\partial c}(Q1)=0, \frac{\partial}{\partial d}(Q1)=0$$

Получаемая система уравнений:

$$[(4(319638322320ab^2d^2-15565200bcd-22600689440ab^2d+821843232abd+15\,33^{7/2}bd-15bd+628880bc-28560c+410921616ab^2-31130400ab-21\,33^{5/2}b+21b+628880a+35\,33^{3/2}-35))/105, (4a(319638322320abd^2-15565200cd-22600689440abd+410921616ad+15\,33^{7/2}d-15d+628880c+410921616ab-15565200a-21\,33^{5/2}+21))/105, -\left(\frac{4(444720abd-48c-17968abb+816a+3\sqrt{33}-3)}{3}\right), (4ab(319638322320abd-15565200c-11300344720ab+410921616a+15\,33^{7/2}-15))/105] \quad (22)$$

53. Решений данной СЛАУ будет несколько, из них выбирается такое, которое не содержит ноль или свободную переменную.

Решение 1

$$\left[d=-\left(\frac{\left(8554\ \sqrt{33}-1603644\right)\ \%r1-1190\ \sqrt{33}+59535}{42828795\ \%r1}\right)c=\frac{\sqrt{33}-1}{16},b=\%r1,a=0\right]$$

Решение 2

$$\left[d=-\left(\frac{7\ \sqrt{33}-714}{31113}\right),c=\frac{281\ \sqrt{33}-553}{2048},b=0,a=\frac{9\ \sqrt{33}-25}{2048}\right] \quad (23)$$

Решение 3

$$d=-\left(\frac{5\ \sqrt{33}-510}{31113}\right),c=\frac{26952253\ \sqrt{33}-98570125}{67108864},$$
$$b=-\left(\frac{6837\ \sqrt{33}-360468}{6877865}\right),a=\frac{4799175\ \sqrt{33}-21715895}{67108864}$$

В данном случае, выбирается решение 3:
$d=0.01546868469023591$, $c=0.8383211616254229$,
$b=0.04669943727968192$, $a=0.08722046672104511$

54. Полином, с учётом численных параметров (23), можно записать как (24).

$$0.838321-0.08722\ x\left(1.0-0.0466994\left(1.0-0.01546868469\ x\right)x\right) \quad (24)$$

Используя этот полином, при начальном значении $x0=4$ можно получить с использованием итерационной формулы приближённое значение функции $y_2=0.4998122042390029$ (теоретическое значение 0.5)

Также воспользуемся другим способом, когда полином определяется по трём точкам
Зададим точки по x как 2,16,32, соответствующие началу диапазона, середине и завершению, при этом, точки по y будут равны $\left[\frac{1}{\sqrt{2}},\frac{1}{4},\frac{1}{2^{\frac{5}{2}}}\right]$ соответственно
Составляя систему уравнения с полиномом в общем виде можно получить (25).

$$\frac{1}{\sqrt{2}}=c-2\ a\left(1-2\ b\left(1-2\ d\right)\right),\frac{1}{4}=c-16\ a\left(1-16\ b\left(1-16\ d\right)\right)$$
$$\frac{1}{2^{\frac{5}{2}}}=c-32\ a\left(1-32\ b\left(1-32\ d\right)\right) \quad (25)$$

55. Выбираем значение $c=1$, чтобы исключить решение со свободной переменной, так как имеются три уравнения и четыре неизвестных. В этом случае решением будет являться (26):

$$a=0.16684758192347446,b=0.06345021421215845,d=0.01823208875818476 \quad (26)$$

Полином соответствующий решению (26) представляется как (27):

$$1.0 - 0.16684758\,x\,\left(1.0 - 0.06345\left(1.0 - 0.018232\,x\right)\,x\right) \qquad (27)$$

56. График основной функции представлен на рис. 12 красным, полинома (24) построенного по методу наименьших квадратов — синий, полинома (27), определяемым прямым решением системы уравнений при прохождении через точки — зелёным.

Видно, что решение уравнения по точкам довольно чувствительно к изменению свободного параметра, также, имеются существенные «выбросы». Поэтому, необходимо использовать коэффициенты, получаемые по методу НК.

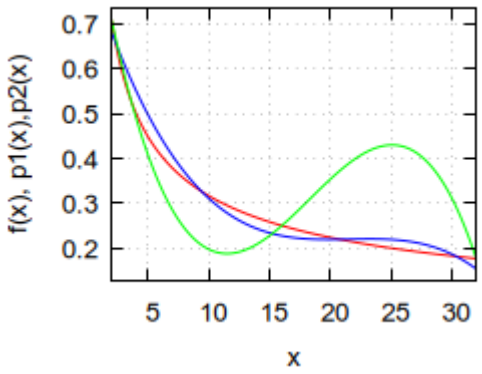


Рис. 12. Полиномы, построенные различными методами

57. Рассмотрим полином (24), получаемый по методу наименьших квадратов и его коэффициенты. Исходные значения коэффициентов в порядке c, a, b, d:
$[0.8383211616254229, 0.08722046672104511, 0.04669943727968192, 0.01546868469023591]$
Домножим эти коэффициенты на степень двойки, а именно на 2^6:
$53.65255434402707, 5.582109870146887, 2.988763985899643, 0.9899958201750982$
Видно, что коэффициент d близок к 1, b близок к 3, а коэффициент a к 6 или к 3·2. Таким образом, можно записать следующую аппроксимацию (28):

$$c - \frac{3\,x\left(1 - \frac{3\left(1 - \frac{x}{64}\right)x}{64}\right)}{32} \qquad (28)$$

58. Интегрируя как было иллюстрировано выше, можно получить следующую функцию ошибки (29):

$$\begin{aligned}&(601295421440\log(x2) + 405\,x2^7 - 60480\,x2^6 + 3870720\,x2^5 - 20643840\,c\,x2^4 - \\ &123863040\,x2^4 + 23592960\,x2^{7/2} + 1761607680\,c\,x2^3 + 1761607680\,x2^3 - \\ &2113929216\,x2^{5/2} - 56371445760\,c\,x2^2 + 75161927680\,x2^{3/2} + 601295421440 \\ &c^2\,x2 - 2405181685760\,c\sqrt{x2} - 601295421440\log(x1) - 405\,x1^7 + 60480\,x1^6 - \\ &3870720\,x1^5 + 20643840\,c\,x1^4 + 123863040\,x1^4 - 23592960\,x1^{7/2} - 1761607680 \\ &c\,x1^3 - 1761607680\,x1^3 + 2113929216\,x1^{5/2} + 56371445760\,c\,x1^2 - 75161927680 \\ &x1^{3/2} - 601295421440\,c^2\,x1 + 2405181685760\,c\sqrt{x1})/601295421440\end{aligned} \qquad (29)$$

Подставляя граничные значения в пределы интегрирования $x1=1, x2=33$ можно получить (30):

$$(19241453486080\ c^2 - 2405181685760\sqrt{33}\ c - 20103760445440\ c + 601295421440\log(33) + 31094996992\ 33^{3/2} + 6975036823136)/601295421440 \tag{30}$$

Производная по единственному коэффициенту $\frac{d}{dc}$ от будет равна (31):

$$\left[\frac{131072\ c - 8192\ \sqrt{33} - 68473}{2048}\right] \tag{31}$$

Откуда численное значение будет равно (32):

$$c = \frac{8192\ \sqrt{33} + 68473}{131072} \approx 0.881442697146908 \tag{32}$$

59. Общий полином для нахождения начальных условий представится как (33):

$$0.881442697146908 - \frac{3\ x\left(1 - \frac{3\left(1 - \frac{x}{64}\right)x}{64}\right)}{32} \tag{33}$$

На рис. 13 представлен исходный график и график полинома, получаемого по оптимизированному соотношению (33) (синяя линия).

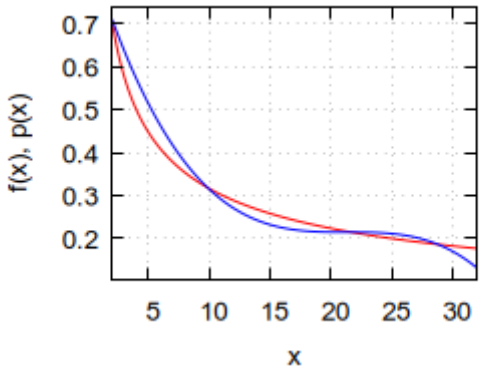


Рис. 13. Аппроксимируемая функция и аппроксимирующий полином (33).

60. Таким образом, полином, задающий начальные условия может быть сформирован с использованием только операций двоичного сдвига и сложения. Также, следует учесть что $3x = 2x + x$ или $3x = (x \ll 1) + x$. Код вычислений (непосредственно сам алгоритм):

```
a = (val >> 6);                 //x / 64
s = Float2Q26(1.0) - a;         // (1 - x/64)
c = MULch26(val, s);            //(1-x/64) * x
c = c >> 6;                     //((1-x/64)*x) / 64
d = c;
c = (c << 1) + d;           //умножение на 3: c*2+c = 3c;  3 * ((1-x/64)*x)/64
s = Float2Q26(1.0) - c;         //1 - (3*(1-x/64)*x)/64
s = s >> 5;                     // ((1-(3*(1-x/64)*x)/64)) / 32
c = MULch26(val, s);            // (x * (1-(3*(1-x/64)*x)/64))/32
d = c;
c = (c << 1) + d;               //3 * (x*(1-(3*(1-x/64)*x)/64))/32
s = cc - c;                     //c - (3*x*(1-(3*(1-x/64)*x)/64))/32
y0 = s;
```

4.2 Арктангенс

61. Для нахождения арктангенса используем два диапазона. От 0 до 1, где значение находится с использованием табличной выборки и разложения Тейлора в окрестности этой точки, при этом, коэффициент ряда Тейлора заменяется аппроксимирующим полиномом, который также представляет собой ряд Тейлора но уже от выражения для коэффициента.

А также диапазон от 1 до 32, где значение арктангенса вычисляется с аргументом, представляющим собой двоичный логарифм от исходного. При этом, имеются 5 аппроксимирующих полиномов второго порядка как решение, проходящее через три точки. Коэффициенты этих полиномов находятся из таблицы.

62. Для диапазона от 0 до 1 составляется следующая таблица.

Задаём общее количество точек равное 64. Аргумент арктангенса записывается в общем виде (34):

$$atan(x0),\ x0 = a\ idx + b\left[idx = \frac{x0 - b}{a}\right]\ atan(a\ idx + b) \tag{34}$$

Решением системы (34) будет (35):

$$atan(b) = 0,\ atan(b + 64\ a) = \frac{\pi}{4}\ \text{решение}\ [b = 0],\left[a = -\left(\frac{b-1}{64}\right)\right] \text{->} \left[a = \frac{1}{64}, b = 0\right] \tag{35}$$

63. Удобство нахождения индекса из аргумента - арифметический сдвиг, $[idx = 64\ x0, idx = x0 \ll 6]$, в этом случае непосредственно находится индекс, по которому далее из таблицы извлекается нужное значение функции арктангенса. Первые значения для таблицы представлены в (36):

$$\left[0, atan\left(\frac{1}{64}\right), atan\left(\frac{1}{32}\right), atan\left(\frac{3}{64}\right), atan\left(\frac{1}{16}\right), atan\left(\frac{5}{64}\right), atan\left(\frac{3}{32}\right), ...\right] \tag{36}$$

64. Следующая таблица составляется для значений арктангенса свыше 1. Используется функция от аргумента $atan(2^x)$. Пользователь задаёт значение y, откуда $y = 2^x$, затем находится x для вычисления положения в таблице. Аргумент y принимает значения $[1, 2, 4, 8, 16, 32]$ для значений x от 0 до 5 можно записать систему (37):

$$x0 = a\ idx + b,\left[idx = \frac{x0 - b}{a}\right],\ atan(2^{x0}),\ atan(2^{a\ idx + b}),\ [b > 0, a > 0] \tag{37}$$

Для точек $atan(2^{idx})$, где idx от 0 до 5 будут следующие точки:

$\left[\frac{\pi}{4}, atan(2), atan(4), atan(8), atan(16), atan(32)\right]$ в градусах это будут следующие значения:

$[45.0, 63.43, 75.96, 82.87, 86.42, 88.21]$

Видно, что для значений справа угол изменяется в относительно небольшом диапазоне.

65. Для формирования аппроксимирующего полинома используются значения индекса, соответствующие промежуточным точкам. $idx = i/2$, для iот 0 до 10, в этом случае значения примут вид (38):

$$\begin{gathered}\left[1, \sqrt{2}, 2, \ldots, 2^{\frac{7}{2}}, 16, 2^{\frac{9}{2}}, 32\right] \\ \left[\frac{\pi}{4}, atan(\sqrt{2}), atan(2), \ldots atan\left(2^{\frac{7}{2}}\right), atan(16), atan\left(2^{\frac{9}{2}}\right), atan(32)\right]\end{gathered} \tag{38}$$

66. В качестве базового аппроксимирующего полинома выбираем форму (39):

$$a\ x\ (1 - b\ x) + c \tag{39}$$

Для значений на i-м интервале можно записать $y_i = a\ x_i\ (1 - b\ x_i) + c$. Для каждого интервала составляются системы уравнений для 3х неизвестных a, b, c (40):

$$\begin{gathered}\left[\frac{\pi}{4} = c + a\ (1 - b), atan(\sqrt{2}) = c + \sqrt{2}\ a\ (1 - \sqrt{2}\ b), atan(2) = c + 2\ a\ (1 - 2\ b)\right], \\ \left[atan(2) = c + 2\ a\ (1 - 2\ b), atan\left(2^{\frac{3}{2}}\right) = c + 2^{\frac{3}{2}}\ a\ \left(1 - 2^{\frac{3}{2}}\ b\right), atan(4) = c + 4\ a\ (1 - 4\ b)\right], \\ \left[atan(4) = c + 4\ a\ (1 - 4\ b), atan\left(2^{\frac{5}{2}}\right) = c + 2^{\frac{5}{2}}\ a\ \left(1 - 2^{\frac{5}{2}}\ b\right), atan(8) = c + 8\ a\ (1 - 8\ b)\right], \\ \left[atan(8) = c + 8\ a\ (1 - 8\ b), atan\left(2^{\frac{7}{2}}\right) = c + 2^{\frac{7}{2}}\ a\ \left(1 - 2^{\frac{7}{2}}\ b\right), atan(16) = c + 16\ a\ (1 - 16\ b)\right], \\ \left[atan(16) = c + 16\ a\ (1 - 16\ b), atan\left(2^{\frac{9}{2}}\right) = c + 2^{\frac{9}{2}}\ a\ \left(1 - 2^{\frac{9}{2}}\ b\right), atan(32) = c + 32\ a\ (1 - 32\ b)\right]\end{gathered} \tag{40}$$

67. Решая каждую систему уравнений можно получить следующие значения коэффициентов (41) полиномов на соответствующих интервалах

$$\begin{gathered}[a = 0.7748280472520914, b = 0.19491528667221494, c = 0.16159594709717506], \\ [a = 0.3147929446206428, b = 0.10877968899592544, c = 0.6145351430085881], \\ [a = 0.09211685070718545, b = 0.05605284039282908, c = 1.0399648389221738], \\ [a = 0.024033001091188666, b = 0.028244111018645383, c = 1.2976199315781933], \\ [a = 0.006074091479639554, b = 0.014149597029285421, c = 1.4331942154942237]\end{gathered} \tag{41}$$

Коэффициентам (41) соответствуют следующие полиномы (42)

$$\begin{gathered}0.7748280472520914\ (1 - 0.19491528667221494\ x)\ x + 0.16159594709717506, \\ 0.3147929446206428\ (1 - 0.10877968899592544\ x)\ x + 0.6145351430085881, \\ 0.09211685070718545\ (1 - 0.05605284039282908\ x)\ x + 1.0399648389221738, \\ 0.024033001091188666\ (1 - 0.028244111018645383\ x)\ x + 1.2976199315781933, \\ 0.006074091479639554\ (1 - 0.014149597029285421\ x)\ x + 1.4331942154942237\end{gathered} \tag{42}$$

68. Графики аппроксимирующих полиномов на заданных интервалах для значения аргумента свыше 1 по формулам (42) представлены на рис. 14.

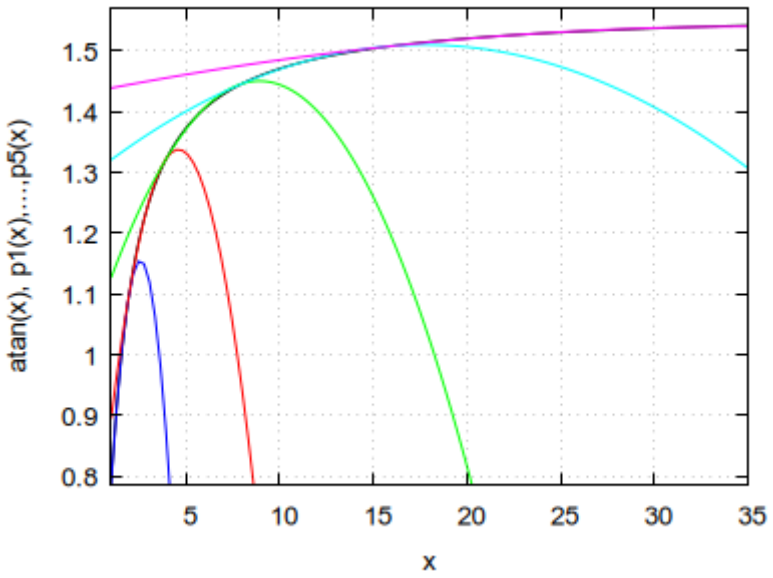


Рис. 14. Графики аппроксимирующих полиномов, соответствующих заданным интервалам

На рис. 15 представлены графики ошибки, определяемой как разница между аппроксимирующими полиномами и исходной функцией.

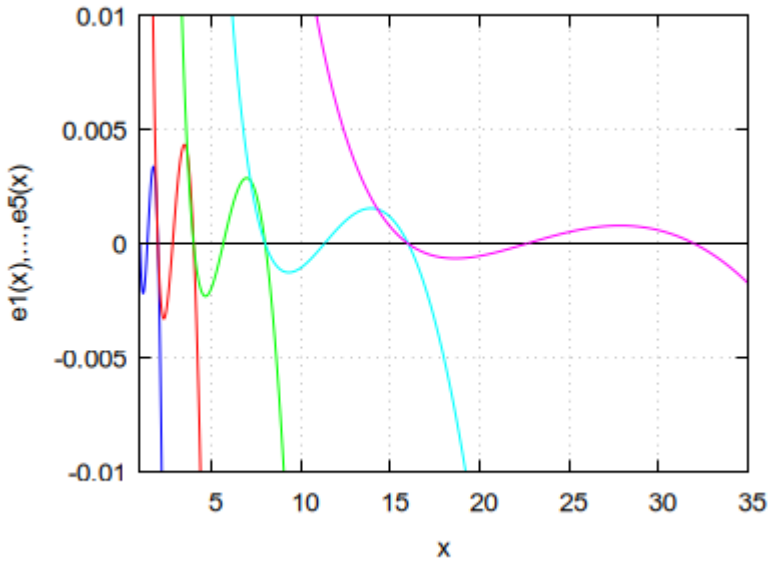


Рис. 15. аппроксимирующие полиномы на интервалах

69. Разложение Тейлора функции арктангенса в заданной точке содержит нелинейную зависимость коэффициентов полинома от самой точки. Обозначение: $\mathrm{Tay}(atan(x), x0)$ - разложение функции арктангенса в точке x0, представленное в (43).

$$\mathrm{Tay}(atan(x), x0) = atan(x0) + \frac{x - x0}{x0^2 + 1} - \frac{x0\,(x - x0)^2}{x0^4 + 2\,x0^2 + 1} + \ldots \tag{43}$$

70. Модифицированный ряд Тейлора с аппроксимированными коэффициентами, не содержащих деление, представлен в (44). Более подробно про выбор коэффициентов см. далее.

$$\mathrm{TayA}(atan(x), x0) = atan(x0) + (x - x0)\left(1 - \frac{x0\,(x0 + 1)}{4}\right) + (x - x0)^2\left(\frac{3\,x0}{4} - 1\right)x0 \tag{44}$$

В качестве примера можно использовать разложение в точке $x0 = 1/2$ при этом ряды (43) и (44) можно записать как (45).

$$\mathrm{Tay}(atan(x), \frac{1}{2}) = -\left(\frac{8\,x^2 - 28\,x - 25\,atan\left(\frac{1}{2}\right) + 12}{25}\right)$$
$$\mathrm{TayA}(atan(x), \frac{1}{2}) = -\left(\frac{20\,x^2 - 72\,x - 64\,atan\left(\frac{1}{2}\right) + 31}{64}\right) \tag{45}$$

Для сравнения график аппроксимируемой функции и для различного представления ряда Тейлора для заданной точки представлены на рис. 16.

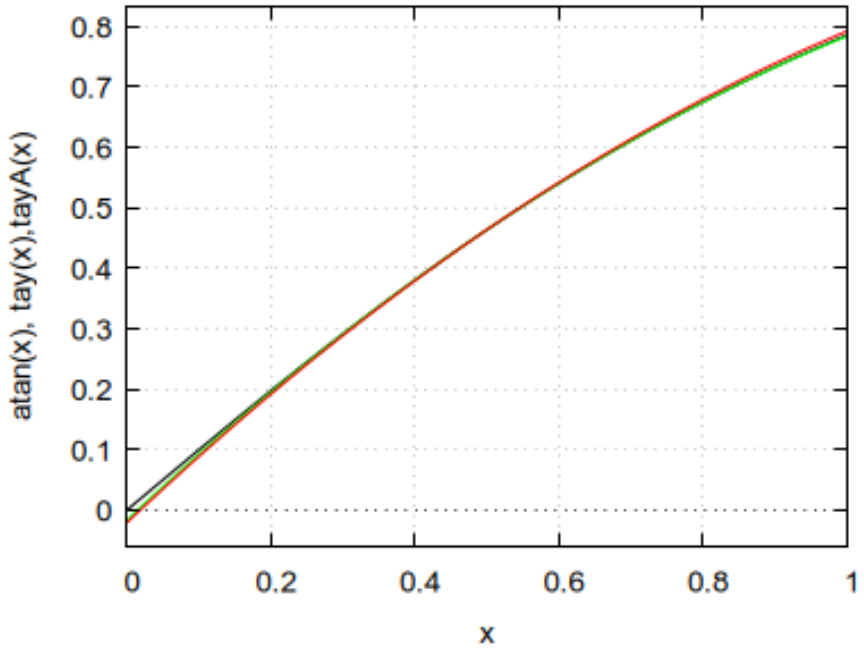


Рис. 16. Разложение в ряд Тейлора для ряда с теоретическими коэффициентами и аппроксимированными

71. Для того чтобы оценить разницу между идеальным значением арктангенса и его аппроксимацией в зависимости от индекса можно использовать замену переменных $\left[x=\frac{i}{64}, x0=\frac{i0}{64}\right]$. Значение в точке $atan\left(\frac{i0}{64}\right)$, далее определяется в виде (46):

$$\begin{gathered}\mathrm{TayA}(atan(i), i0+1)= \\ =-\left(\frac{67108864\ atan\left(\frac{i0+1}{64}\right)+61\ i0^2+4352\ i0-67108864\ atan\left(\frac{i0}{64}\right)-1048576}{67108864}\right) \\ \mathrm{TayA}(atan(i), i0-1)= \\ \frac{67\ i0^2+3840\ i0+67108864\ atan\left(\frac{i0}{64}\right)-67108864\ atan\left(\frac{i0-1}{64}\right)-1048576}{67108864}\end{gathered} \tag{46}$$

Далее можно построить график ошибки для предыдущего и следующего индекса, например, для $ep(i0)=atan\left(\frac{i0}{64}\right)-\mathrm{TayA}(atan(i), i0+1), en(i0)=atan\left(\frac{i0}{64}\right)-\mathrm{TayA}(atan(i), i0-1)$, он представлен на рис. 17.

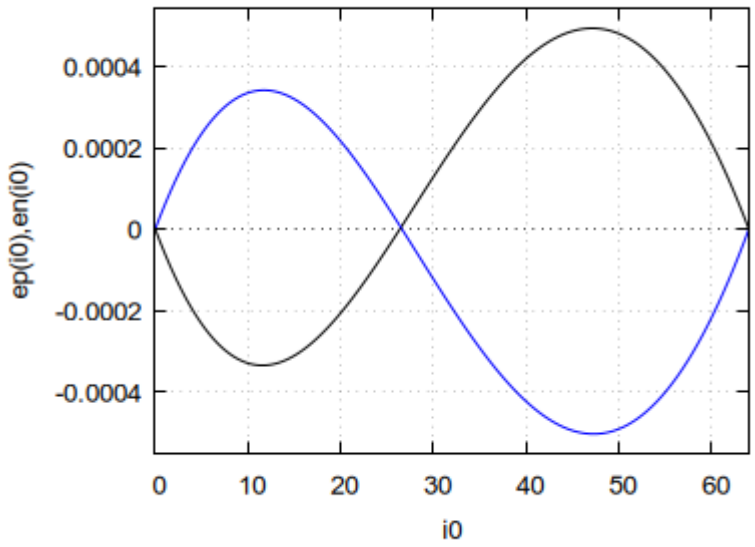


Рис. 17. График ошибки при нахождении арктангенса в зависимости от индекса

72. Далее рассматриваются коэффициенты разложения в ряде Тейлора и их аппроксимации в зависимости от x0. Коэффициент при первой степени $(x-x0)$ даётся выражением (47):

$$p1(x0)=coeff\left(\mathrm{Tay}(atan(x), x0), (x-x0)\right)=\frac{1}{x0^2+1} \tag{47}$$

Определяем граничные точки коэффициента: при $x0=0$, $p1(x0)=1$, $x0=1$, $p1(x0)=1/2$.

Рассмотрим линейную аппроксимацию коэффициента $a\,x0+b$, в этом случае имеем решение $\left[b=1, b+a=\frac{1}{2}\right]$ и $\left[a=-\left(\frac{1}{2}\right), b=1\right]$, прямая будет определяться как $p2(x0)=1-\frac{x0}{2}$.

73. Рассмотрим аппроксимацию вида $a\,x0^2+b\,x0+c$, экстремум этой функции в точке $x0=-\left(\frac{b}{2\,a}\right)$. Рассмотрим вариант параболы, экстремум которой располагается в точке x0=0:

в этом случае $\left[c=1, c+b+a=\frac{1}{2}, c-\frac{b^2}{4\,a}=1\right]$, решением будет $\left[a=-\left(\frac{1}{2}\right), b=0, c=1\right]$, полином $p3(x0)=1-\frac{x0^2}{2}$. Рассмотрим среднее значение между прямой и этим полиномом, определяемое согласно (48):

$$p4(x0)=(p1(x0)+p2(x0))/2=-\left(\frac{x0^2}{4}\right)-\frac{x0}{4}+1 \qquad (48)$$

Графики полиномов $p1(x0), p2(x0), p3(x0), p4(x0)$ представлены на рис. 18. Соответственно чёрным, синим, красным и зелёным.

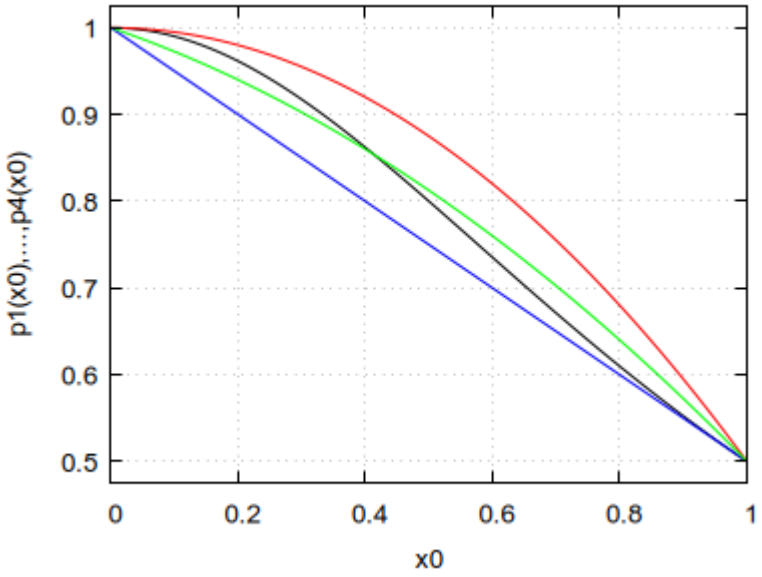


Рис. 18. Графики аппроксимирующих полиномов к выражению для линейного члена разложения функции арктангенса в ряд Тейлора

74. Рассмотрим коэффициент при второй степени разницы между аргументом и точкой в разложении в ряд Тейлора. Коэффициентом будет являться (49):

$$p1_2(x0)=coeff\left(\mathrm{Tay}\left(atan(x), x0\right), (x-x0)^2\right)=-\frac{x0}{x0^4+2\,x0^2+1} \qquad (49)$$

Зададим граничные точки в (50):

$$p1_2(0)=0,\ p1_2(1)=-1/4 \tag{50}$$

Производная запишется в виде (51):

$$\frac{d}{dx0}p1_2(x0)=\frac{3\ x0^2-1}{\left(x0^2+1\right)^3} \tag{51}$$

Решением уравнения для производной (51), приравниваемой к нулю $\frac{d}{dx0}p1_2(x0_{эр1})=0$ с целью поиска экстремума, будет $x0_{эр1}=\pm\frac{1}{\sqrt{3}}$, при этом, выбирается решение с положительным знаком. Значение функции в этой точке $p1_2(x0_{эр1})=-\left(\frac{3^{\frac{3}{2}}}{16}\right)$ или -0.32475952641916445.

Зададим аппроксимирующий полином (52) :

$$p_2(x0)=a\ x0^2+b\ x0+c \tag{52}$$

Экстремум полинома (52) будет в точке $x0_{эр2}=-\left(\frac{b}{2\ a}\right)$. Подставляя значение этой точки в полином можно получить $p_2(x0_{эр2})=c-\frac{b^2}{4\ a}$. Таким образом можно сформулировать первое условие — равенство функций на границах заданным значениям (50) и равенство экстремума значению $p1_2(x0')$ в соответствующей точке, обозначим этот полином как $p2_2(x0)\leftarrow p_2(x0)$, в этом случае $p2_2(x0_{эр2})=p1_2(x0')$ можно записать и решить систему уравнений (53):

уравнения

$$\left[c=0,\ c+b+a=-\left(\frac{1}{4}\right),\ c-\frac{b^2}{4\ a}=-\left(\frac{3^{\frac{3}{2}}}{16}\right)\right]$$

решение

$$a=\frac{\sqrt{27-4\ 3^{\frac{3}{2}}}+3^{\frac{3}{2}}-2}{8},\ b=-\left(\frac{\sqrt{27-4\ 3^{\frac{3}{2}}}+3^{\frac{3}{2}}}{8}\right),\ c=0 \tag{53}$$

полином

$$p2_2(x0)=0.7111526094089196\ x0^2-0.9611526094089196\ x0\approx x0\cdot\left(\frac{3}{4}x0-1\right)$$

75. Следующим условием является как и ранее определение границ (50) и непосредственное значение аппроксимирующей функции в точке экстремума исходной. Введём новый полином $p3_2(x0) \leftarrow p_2(x0)$ и запишем условие $p3_2(x0_{эр1}) = p1_2(x0_{эр1})$, система уравнений и решение представлено в (54):

уравнения

$$\left[c = 0, c + b + a = -\left(\frac{1}{4}\right), c + \frac{b}{\sqrt{3}} + \frac{a}{3} = -\left(\frac{3^{\frac{3}{2}}}{16}\right)\right]$$

решение

$$a = \frac{5\sqrt{3}+15}{32}, b = -\left(\frac{5\sqrt{3}+23}{32}\right), c = 0 \qquad (54)$$

полином

$$p3_2(x0) = 0.739382938682637\ x0^2 - 0.989382938682637\ x0 \approx x0 \cdot \left(\frac{3}{4} x0 - 1\right)$$

Видно, что полиномы (53) и (54) могут быть аппроксимированы с использованием полинома, содержащего умножение на 3, которое может быть представлено как $2x0 + x0 = (x0 \ll 1) + x0$, с использованием арифметического сдвига влево, а также, деление на 4, которое представляется как арифметический сдвиг вправо. Данный метод представлен в (55):

$$p4_2(x0) = \frac{3\ x0^2}{4} - x0 = 0.75\ x0^2 - 1.0\ x0 =$$
$$= x0 \cdot \left(\frac{1}{4}(2 \cdot x0 + x0) - 1\right) = x0 \cdot \left(((x0 \ll 1 + x0) \gg 2) - 1\right) \qquad (55)$$

Графики аппроксимирующих полиномов для коэффициента второго порядка ряда Тейлора представлены на рис. 19.

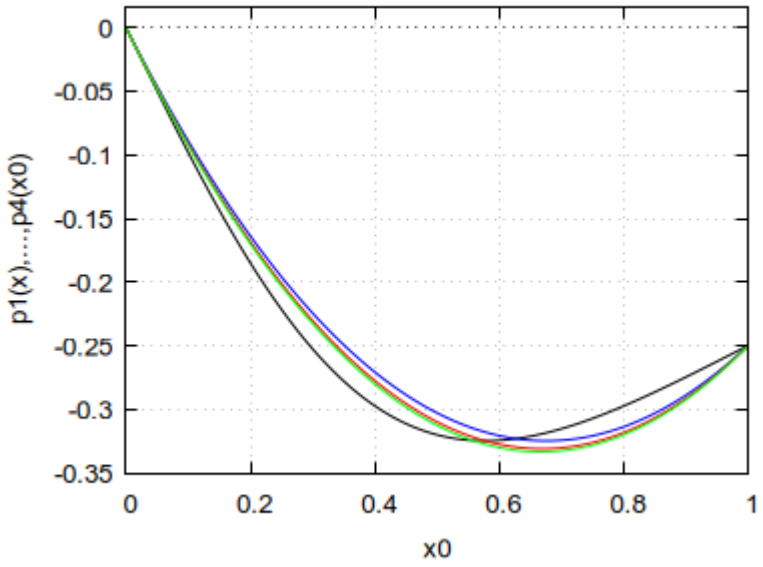


Рис. 19. Графики аппроксимирующих полиномов к выражению для квадратичного члена разложения функции арктангенса в ряд Тейлора

76. Таким образом, окончательный вид функции вычисления арктангенса будет следующим:

```
volatile long a, b, ka,kb,kc, s, i0, dx, dx2, y0, x0, yn, res;
char sgn = 0;

volatile register long vals; //ключевое слово volatile обязательно
if (val < 0) {
	vals = val - 1; //проблема модуля val = 0x8000'0000.
			//Инструкция neg val не меняет знак. Проверяется сложением с +-1
	if (vals > 0)
	{
		val = vals;
	}
	else
	{
		val = -val; //взятие модуля
	}
	sgn = 1;
}
//Выборка значения из таблицы
//Таблица дана до значения равным 1 в соответствующем q-формате
//Крайний индекс равен 64, поэтому приводим к максимальному значению индекса в таблице
//При входном значении равным q26(1.0) на выходе имеем 64
i0 = val >> 20;
if (i0 <= 63) {
	//значение по индексу - начальные условия y0
	y0 = AtanPi4Tables[i0]; //табличное значение
	//восстановление x0 по индексу без дробной части - "табличный аргумент"
	x0 = i0 << 20;
	//Вычисление с использованием ряда Тейлора
	//atan(x0)+(x-x0)*(1-(x0*(x0+1))/4)+(x-x0)^2*((3*x0)/4-1)*x0
	dx = (val - x0); // разница между актуальным и "табличным аргументом"
	dx2 = MULch26(dx, dx); // возведение в квадрат, умножение числа само на себя
	a = y0; //"Аккумулятор" начальное значение - atan(x0) по таблице в заданной точке
	//домножение на коэффициент по первой степени разницы
	b = MULch26(x0, (x0+Float2Q26(1.0)));
	b = b >> 2;		//делим на 4
	b = (Float2Q26(1.0) - b);
	b = MULch26(dx, b);
	a = a + b;		//закончили с первой степенью
	//по второй степени разницы
	b = MULch26(Float2Q26(3.0), x0);
	b = b >> 2; //делим на 4
	b = b - Float2Q26(1.0);
	b = MULch26(b, x0);
	b = MULch26(dx2, x0);
	a = a + b;		//завершили вторую степень
	res = a;		//итого
	} else {
	//вычисляем с использованием полинома на интервале, определяемым
	//с использованием двоичного логарифма от аргумента
	a = __builtin_clz(val);//подсчёт количества старших нулей до первой попавшейся единицы
	//проверяются возможные границы не больше 5 и не менее 0
	if (a > 5) {
		a = 5;
	}
	if (a < 0) {
		a = 0;
	}
	i0 = 5 - a;
	//для проверки "табличного аргумента"
	x0 = 1<<(i0+26); //приведение к формату Q26
	//вычисление непосредственное
	//P1:a*x*(1-b*x)+c; - полином для вычислений
	ka = AtanPi42Pow2Tables_a[i0];
	kb = AtanPi42Pow2Tables_b[i0];
	kc = AtanPi42Pow2Tables_c[i0];
	s = MULch26(kb,val);
	s = Float2Q26(1.0) - s;
	s = MULch26(s,ka);
	s = MULch26(s,val);
	s = s + kc;
	res = s;
}
//восстановление знака
asm(" nop");
```

```
if (sgn == 1)
{
        res = -res;
}
return res;
```

Следует отметить, что инструкция `clz`, соответствующая вставке `__builtin_clz` является расширением, необходимо уточнять её наличие в контроллере. При её отсутствии — формируется алгоритм подсчёта нулей за счёт применения других команд.