Національний Технічний Університет України
"Київський Політехнічний Інститут"
Фізико-Технічний Інститут

СПЕЦІАЛЬНІ РОЗДІЛИ ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ

КОМП'ЮТЕРНИЙ ПРАКТИКУМ №1

Багаторозрядна арифметика

Виконав студент 3-го курсу
групи ФІ-13
Мельник Євгеній

Київ 2023

1 Вступ

Звіт з першого комп'ютерного практикума по спеціальним розділам обчислюванної математики. Увесь код можна знайти за посиланням на GitHub.

1.1 Мета

Отримання практичних навичок програмної реалізації багаторозрядної арифметики; ознайомлення з прийомами ефективної реалізації критичних по часу ділянок програмного коду та методами оцінки їх ефективності.

1.2 Завдання

1) Повинні бути реалізовані такі операції:

    1. переведення малих констант у формат великого числа (зокрема, 0 та 1);
    2. додавання чисел;
    3. віднімання чисел;
    4. множення чисел, піднесення чисел до квадрату;
    5. ділення чисел, знаходження остачі від ділення;
    6. піднесення числа до багаторозрядного степеня;
    7. конвертування (переведення) числа в символьну строку та обернене перетворення символьної строки у число; обов'язкова підтримка шістнадцяткового представлення, бажана десяткового та двійкового.
    8. визначення номеру старшого ненульового біта числа;
    9. бітові зсуви (вправо та вліво), які відповідають діленню та множенню на степені двійки.

2) Проконтролювати коректність реалізації алгоритмів; наприклад, для декількох багаторозрядних a, b, c, n перевірити тотожності:

    1. $(a + b) \cdot c = c \cdot (a + b) = a \cdot c + b \cdot c$;
    2. $n \cdot a = \underbrace{a + a + \ldots + a}_{n}$, де n повинно бути не менш за 100 ;

    і таке інше.

    Продумати та реалізувати свої тести на коректність.

3) Обчислити середній час виконання реалізованих арифметичних операцій. Підрахувати кількість тактів процесора (або інших одиниць виміру часу) на кожну операцію. Результати подати у вигляді таблиць або діаграм.

2 Реалізація завдання

2.1 Структура проекту

Проект складається з файлів BigInt.*, та системи збірки - CMake.
BigInt - бібліотека довгих чисел. У бібліотеці є клас BigInt - самі довгі числа, в якому реалізовані усі допоможні функції, арифметика, конвертори з різних систем числення, та інше.

2.2 Середній час виконання

Розглянемо середній час виконання арифметичних операцій. Заміри для всіх операцій, окрім піднесення до степіню проводились на різних розмірах великого числа(на розмірах вбудованного масива від 64 до 8192) у тактах процесора. Як можна буде побачити, час виконання суттєво залежить від того, яку ми поставили можливу довжину числа (тобто довжини вбудованого масива).
Кратке уточнення: для майже всіх подальших таблиць Small numbers та Big numbers = числа довжини (Array Size)/7 та (ArraySize)*6 у шістнацятирічній системі числення. При чому варто зазначити, що судячи з вимірів, які були зроблені на випадкових числах по 1000 разів на довжинах до 1024, і по 100 разів у подальшому, можна зробити висновок що заповненність самого масиву не грає ролі у кількості тактів процесора.

1) Додавання: як можна побачити з таблиці 1, кількість тактів процесора для додавання росте з збільшенням масиву. Чомусь для маленьких чисел додавання працює довше.

Array size	64	128	256	512	1024	2048	4096	8192
Small numbers	28	53	99	149	321	565	1133	2898
Big numbers	22	41	89	143	362	677	1269	2456

Табл. 1: Кількість тактів для додавання

2) Віднімання: все так само як і з додаванням, числа виходять майже однаковими. Але немає дивної поведінки, тому що для менших чисел віднімання працює швидше.

Array size	64	128	256	512	1024	2048	4096	8192
Small numbers	25	47	96	155	299	600	1160	2452
Big numbers	27	47	100	175	340	701	1440	2996

Табл. 2: Кількість тактів для віднімання

3) Множення: як можемо бачити, кількість тактів суттєво більша ніж у додавання і множення. Можливо із-за того що у множенні використовується додавання, але для деяких менших чисел алгоритм працює довше.

Array size	64	128	256	512	1024	2048	4096	8192
Small numbers	2540	9638	42947	151549	607967	2328355	9307792	38913218
Big numbers	2461	9486	44913	145122	587425	2312513	9437713	40168797

Табл. 3: Кількість тактів для множення

4) Ділення: побачивши результати, я думав що я щось переплутав, але ні. З довжинни масиву 2048 ділення починає працювати ***швидше*** ніж множення.

Array size	64	128	256	512	1024	2048	4096	8192
Small numbers	60315	122052	262816	496024	966243	1895101	3772466	7639897
Big numbers	51360	101949	235052	397795	792811	1582316	3223783	6939508

Табл. 4: Кількість тактів для ділення

5) Модуль числа: Працює трошки швидше ніж ділення - але не набагато, тому що використовується одна й та ж сама функція, але не обчислюється дільник, тільки остача.

6) Піднесення в степінь: Бралась окрема вибірка з числом розміру (Array size)$*4$, і підносилась до степіню 1000. Числа з розміром масиву 8192 не брались, тому що занадто довго обраховується. :)

Array size	64	128	256	512	1024	2048	4096	8192
Small numbers	41833	86172	187056	342929	681078	1329635	2651094	5353054
Big numbers	33228	66216	148376	255733	515606	1022484	2073757	4490010

Табл. 5: Кількість тактів для модуля

Array size	64	128	256	512	1024	2048	4096
Small numbers	53422	203852	634750	2271761	10142121	36631209	174713669

Табл. 6: Кількість тактів для піднесення у степінь

2.3 Декілька прикладів обчислення великих чисел

Усі числа будуть у hex, тому що це саме зручне представлення. (Відображається трошки криво, тому що латех не справляється з довгими словами)

1) Перше число: A := 77f612aec34ab3905fdca712aecde12394834; Друге число: B := b159b7283e51f8bec6a01189c6735334

   $A + B$ = 77f61dc45ebd37757f68937cafe67d8ac9b68

   $A - B$ = 77f6079927d82fab4050baa8adb544bc5f500

   $A * B$ = 77f6079927d82fab4050baa8adb544bc5f500531b
   2d4b16ca65fa2de787cb8cc04874b8e57a7d5ad57c7796396f851b26 bed678690

   A/B = ad290

   $A\%B$ = a8a3bbb53569d14ab95c4a129711d2f4

   A^{14} = 19cfbc97fb4428f247570b0cc7a51171e3005c51a3b8e84bdd1def76c81daf58311
   0cd113fd4ef4a6c98f68f487b4cf0dee4289b88491de7e00d32852b4ed8a648c9cc69336f680ad
   f9c212a821737f01af799a4218a5680e9aef7c9162c4fe38ef532b9054340541e8250a05848b4c
   500a8d4cbfeca439b0f78471e46052a59f8e0a3444a40601fad7fbf9c50cc7a330dfad6e11f73b09
   4218d929276395c1a959a6c9ba6e4da7af1109e8ac66d3f41e84b2058bde32728625649ebdf90
   de20afc52d129c86b795a9ba156c93b304dc90ba78a39609bef50f2e6b773d37d2e9189243942
   add65db6db92c6c1ccabbaeb363535d70a782063683adcb99d90000000

2) Перше число: A := d8096aa33c5e6; Друге число: B := ff135f6c6

   $A + B$ = d80a69b69bcac

   $A - B$ = d8086b8fdcf20

   $A * B$ = d8086b8fdcf20d741ba7276ce94e2d413e4

   A/B = 165

   $A\%B$ = a3f643a46263c2e

   A^{13} = 1c2ebcb56b2114bda171e1f8139302063d1defed32f591601df211
   4e7442aaa4757d75638b61eadb8129d07ecfb3275ab7fb5413545d06c79663897
   2ecd08c3b317981d0a54b5df8172986c3ba9cd41e952206000

3 Висновок

У підсумку цього комп'ютерного практикума, я зрозумів як працювати з багаторозрядними числами з основою 2^n, як реалізовуються операції над такими числами. Також було зробленно такий висновок - додавання, віднімання працюють найшвидше, а піднесення до степіню (доволі великого) - саме найдовше.