МРКП

Лабораторна робота №1

Огляд бібліотеки GNU GMP

Булигін Олексій, ФІ-52мн

1 Вступ

Бібліотека **GNU GMP (GNU Multiple Precision Arithmetic Library)** використовується для виконання арифметичних операцій довільної точності в наукових, криптографічних та аналітичних застосуваннях. GMP дозволяє оперувати числами практично необмеженої довжини — наприклад, на 64-бітних системах максимальна довжина операндів сягає 2^{37}.

Основні переваги:

- висока швидкість завдяки оптимізованим алгоритмам;
- підтримка багатьох архітектур (x86, ARM, RISC-V);
- автоматичний вибір алгоритму залежно від розміру операндів;

2 Основні типи даних

GMP надає три головні числові типи:

Тип даних	Призначення	Приклади функцій
`mpz_t`	Цілі числа довільної точності	`mpz_add`, `mpz_mul`, `mpz_powm`
`mpq_t`	Раціональні числа	`mpq_add`, `mpq_div`
`mpf_t`	Дійсні числа	`mpf_add`, `mpf_sqrt`

Додаткові типи:

- `mp_bitcnt_t` — для зберігання кількості бітів;
- `gmp_randstate_t` — стан генератора випадкових чисел.

3 Внутрішнє представлення чисел

Тип `mpz_t` є псевдонімом для структури `__mpz_struct`:

```
typedef struct {
    int _mp_alloc;     // allocated limbs
    int _mp_size;      // assigned limbs
    mp_limb_t *_mp_d; // limbs array
} __mpz_struct;
```

Кожен лімб (`mp_limb_t`) — це машинне слово в залежності від архітектури. Наймолодший лімб зберігається першим, знак визначається знаком `_mp_size`.

Переваги такого підходу

- Використання апаратної арифметики без проміжних копій.
- Оптимізація під ієрархію памяті (послідовне розташування лімбів).
- Можливість асемблерних оптимізацій і SIMD.
- Просте масштабування.

4 Типи функцій

GMP має декілька класів функцій:

1. Цілі числа `mpz_t` — ∼150 функцій
2. Раціональні `mpq_t` — ∼30 функцій
3. Float `mpf_t` — ∼70 функцій
4. Натуральні числа `mpn_` – низькорівневі функції над масивами лімбів. Зазвичай не використовуються окремо
5. Допоміжні функції – генератори випадкових чисел, округлення, тощо

5 Алгоритми множення

GMP автоматично вибирає алгоритм множення залежно від кількості лімбів.

Алгоритм	**Діапазон**	**Складність**	**Ідея**
Класичний	до ∼32 лімбів	$O(n^2)$	Подвійний цикл
Карацуба	до ∼300 лімбів	$O(n^{1.585})$	3 множення замість 4
Toom–Cook	до $\sim 10^4$ лімбів	$O(n^{1.46})$	Інтерполяція частин
FFT (Шёнхаге–Штрассен)	$>10^4$ лімбів	$O(n \log n \log\log n)$	Перетворення Фур'є

Фактичні границі обрання алгоритмів обираються під час компіляції і зазвичай підбираються під конкретні платформи або задачі:

```
#define MUL_TOOM22_THRESHOLD 26
#define MUL_TOOM33_THRESHOLD 73
#define MUL_TOOM44_THRESHOLD 208
```

Параметр означає кількість лімбів, тобто машинних слів. В цьому прикладі для процесора skylake, до 25 слів включно (25*64=1600біт) використовуватиметься звичайне множення.

Коротко про кожен метод

- **Класичний** — реалізований на асемблері для коротких чисел.
- **Карацуба** — розбиття на половини, зменшення кількості множень.
- **Toom–Cook** — поліноміальна інтерполяція для великих чисел.
- **FFT** — використання швидкого перетворення Фур'є.

6 Ключові функції

6.1 Функція `mpz_mul`

- **Призначення:** множення двох цілих чисел довільної точності.
- **Синтаксис:**

```
mpz_mul(product, a, b);
```

- **Складність:** від $O(n^2)$ до $O(n \log n \log \log n)$.

6.2 Функція `mpz_powm`

- **Призначення:** обчислення $rop = base^{exp} \bmod mod$.
- **Синтаксис:**

```
mpz_powm(result, base, exp, mod);
```

6.3 Функція `mpz_powm_sec`

- **Призначення:** Те ж саме, тільки з однаковим часом використання і очищенням кешу для ускладнення side-channel атак

7 Приклад використання (Diffie–Hellman)

```
#include <gmp.h>
#include <stdio.h>

int main(void) {
    mpz_t p, g, a, b, A, B, s1, s2;
    mpz_inits(p, g, a, b, A, B, s1, s2, NULL);

    mpz_set_str(p, "FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF61", 16);
    mpz_set_ui(g, 5);

    mpz_set_ui(a, 12345);
    mpz_set_ui(b, 67890);

    mpz_powm(A, g, a, p);
    mpz_powm(B, g, b, p);

    mpz_powm(s1, B, a, p);
    mpz_powm(s2, A, b, p);

    gmp_printf("Shared secret: 0x%Zx\n", s1);
    mpz_clears(p, g, a, b, A, B, s1, s2, NULL);
}
```

8 Низькорівневі функції mpn_

Призначення

Функції сімейства mpn_ — це внутрішній рівень бібліотеки GMP, який безпосередньо працює з масивами лімбів (mp_limb_t).

Загальні принципи

Усі функції mpn_ мають вигляд:

```
mp_limb_t mpn_operation(mp_limb_t *rp,
                        const mp_limb_t *xp,
                        const mp_limb_t *yp,
                        mp_size_t n);
```

де:

- xp, yp — вхідні масиви лімбів;
- rp — вихідний масив (результат);
- n — кількість лімбів;
- функція зазвичай повертає перенесення (carry) або залишок.

Функції цього рівня не перевіряють коректність розмірів або виділеної пам'яті — їх використовують виключно всередині бібліотеки

9 Паралелизація обчислень

GNU GMP призначена для швидких однопоточних обчислень. Навіть алгоритм множення через FFT, який теоретично легко паралелизується, всередині використовує послідовні обчислення. Проте стандартна збірка GMP передбачає thread-safety в рамках функцій. Тобто доволі легко розділити обчислення з використанням функцій GMP.

10 Висновки

GNU GMP — це високопродуктивна бібліотека для арифметики довільної точності, що:

- поєднує теоретично ефективні алгоритми;
- має глибоку оптимізацію на рівні машинних інструкцій;
- може бути фундаментальним блоком для реалізації розділених обчислень

11 Список джерел

1. **GNU GMP Manual** — https://gmplib.org/manual