Benchmark для процессора на архитектуре RISC-V

Оглавление

Технические характеристики тестируемых плат

Sipeed LicheePi 4A

CPU: TH1520 с четырьмя ядрами RISC-V C910

```
isa             : rv64imafdcvsu
mmu             : sv39
cpu-freq        : 1.848Ghz
cpu-icache      : 64KB
cpu-dcache      : 64KB
cpu-l2cache     : 1MB
cpu-tlb         : 1024 4-ways
cpu-cacheline   : 64Bytes
cpu-vector      : 0.7.1
```

Banana Pi BPI-F3

```
8 cores
CPU(s) scaling MHz:  100%
  CPU max MHz:         1600.0000
  CPU min MHz:         614.4000
Caches (sum of all):
  L1d:                 256 KiB (8 instances)
  L1i:                 256 KiB (8 instances)
  L2:                  1 MiB (2 instances)
LEVEL1_ICACHE_SIZE                 32768
LEVEL1_ICACHE_ASSOC                4
LEVEL1_ICACHE_LINESIZE             64
LEVEL1_DCACHE_SIZE                 32768
LEVEL1_DCACHE_ASSOC                4
LEVEL1_DCACHE_LINESIZE             64
LEVEL2_CACHE_SIZE                  524288
LEVEL2_CACHE_ASSOC                 16
LEVEL2_CACHE_LINESIZE              64
LEVEL3_CACHE_SIZE                  0
LEVEL3_CACHE_ASSOC                 0
LEVEL3_CACHE_LINESIZE              0
LEVEL4_CACHE_SIZE                  0
LEVEL4_CACHE_ASSOC                 0
LEVEL4_CACHE_LINESIZE              0
```

Анализируемый алгоритм

Сортировка слиянием (Merge Sort) — это алгоритм сортировки, использующий метод **"разделяй и властвуй"**. Он рекурсивно делит массив на подмассивы, сортирует их и затем объединяет в один отсортированный массив.

Реализация

Функция mergeSort (Рекурсивное деление массива)

```
static inline void mergeSort(int32_t array[], size_t left, size_t right) {
    if (left < right) {
        size_t middle = left + (right - left) / 2;
        mergeSort(array, left, middle);
        mergeSort(array, middle + 1, right);
        merge(array, left, middle, right);
    }
}
```

- Разделяет массив на две половины (left → middle и middle+1 → right).
- Рекурсивно вызывает mergeSort для каждой из половин.
- После разделения выполняется слияние подмассивов с помощью merge().

Функция merge (Слияние двух отсортированных подмассивов)

```
static inline void merge(int32_t array[], size_t left, size_t middle,
                         size_t right) {
    size_t leftSize = middle - left + 1;
    size_t rightSize = right - middle;

    int32_t* leftArray = malloc(leftSize * sizeof(int32_t));
    int32_t* rightArray = malloc(rightSize * sizeof(int32_t));

    if (!leftArray || !rightArray) {
        printf("Ошибка: недостаточно памяти при слиянии\n");
        free(leftArray);
        free(rightArray);
        return;
```

```
    }

    memcpy(leftArray, &array[left], leftSize * sizeof(int32_t));
    memcpy(rightArray, &array[middle + 1], rightSize * sizeof(int32_t));

    size_t leftIndex = 0, rightIndex = 0, mergedIndex = left;

    while (leftIndex < leftSize && rightIndex < rightSize) {
        if (leftArray[leftIndex] <= rightArray[rightIndex]) {
            array[mergedIndex++] = leftArray[leftIndex++];
        } else {
            array[mergedIndex++] = rightArray[rightIndex++];
        }
    }

    while (leftIndex < leftSize) {
        array[mergedIndex++] = leftArray[leftIndex++];
    }
    while (rightIndex < rightSize) {
        array[mergedIndex++] = rightArray[rightIndex++];
    }

    free(leftArray);
    free(rightArray);
}
```

- Создает **два временных массива**, в которые копирует значения из array[left..middle] и array[middle+1..right].
- Проходит по этим массивам, сравнивает элементы и копирует их **в правильном порядке** в array.
- Освобождает память после завершения слияния.

Горячие инструкции

Наибольшую нагрузку создают следующие операции:

Загрузка и сохранение данных:

- lw (load word)
- sw (store word).

Сравнение элементов:

- bge (branch if greater or equal)
- blt (branch if less than).

Смещение указателей:

- addi (add immediate)

Overhead	Command	Shared Object	Symbol
79.77%	mergesort_bench	mergesort_bench	[.] merge
7.86%	mergesort_bench	libc.so.6	[.] _mcount
4.60%	mergesort_bench	libc.so.6	[.] __random
3.09%	mergesort_bench	mergesort_bench	[.] mergeSort
2.17%	mergesort_bench	mergesort_bench	[.] main

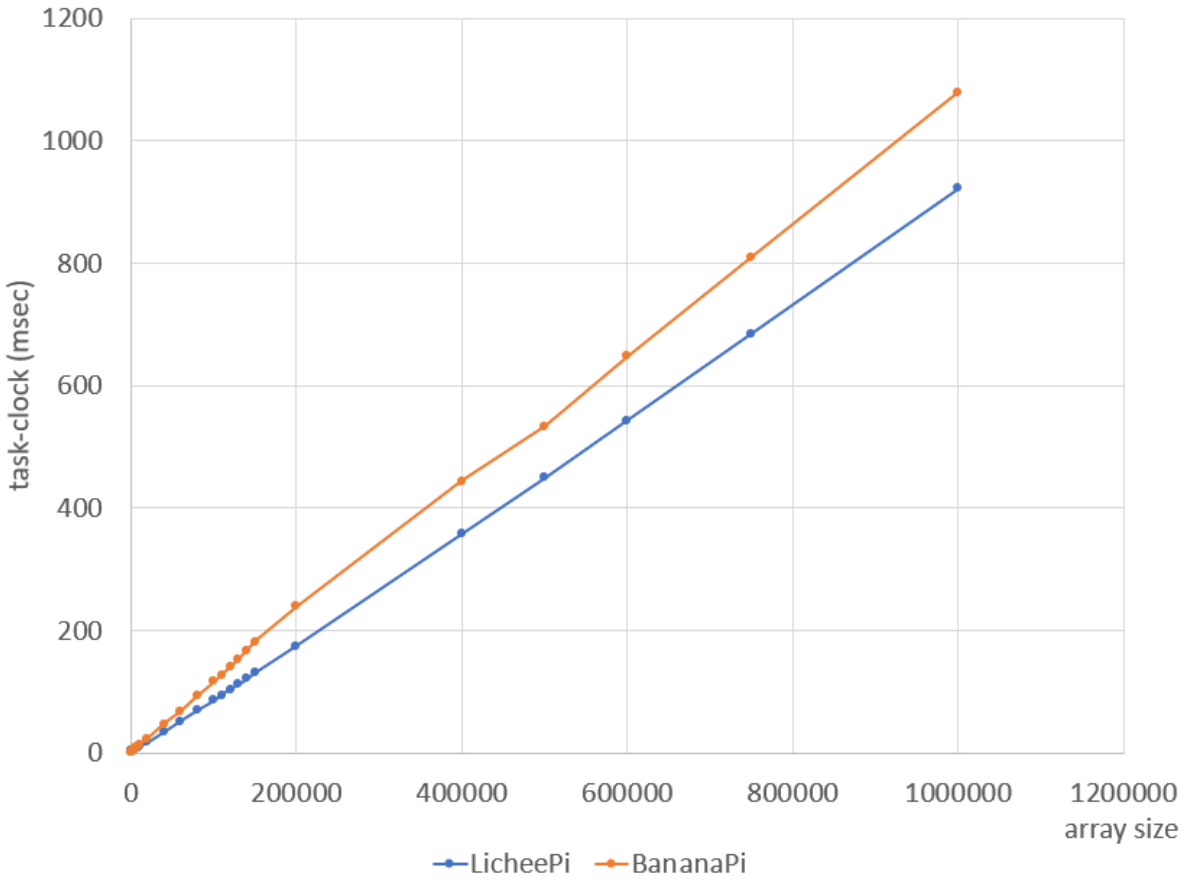
Профилирование алгоритма сортировки слиянием

Результаты тестирования:

Тестирование проводилось путем многократного запуска программы сортировки с различными размерами массивов и фиксированными начальными условиями, а так же с псевдослучайными (seed).

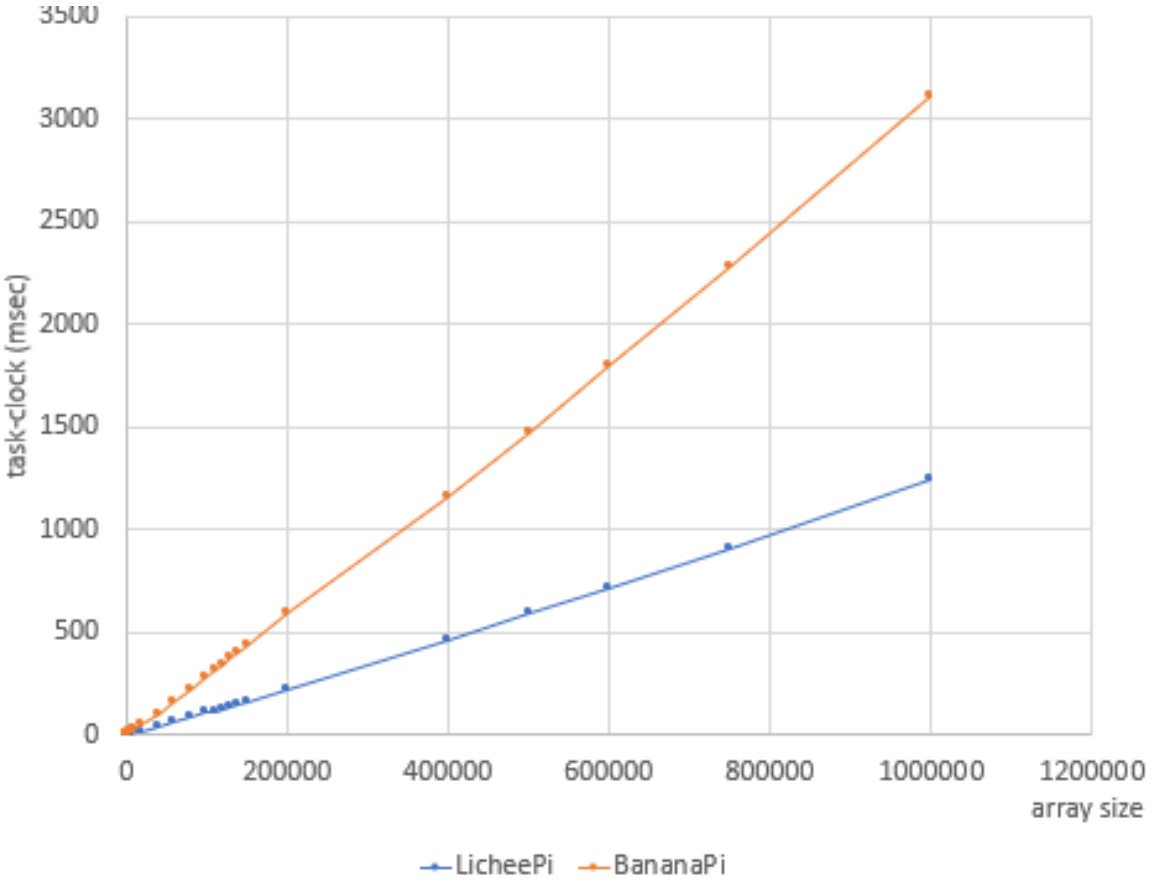
Параметры тестов:

- **Диапазон размеров массива**: от **10 до 1 000 000 элементов**.
- **Количество прогонов (num_runs): 15 запусков** для каждого размера массива.



Зависимость времени выполнения сортировки слиянием от размера массива (merge sort)

Так же было дополнительно проведено тестирование при помощи qsort (stdlib.h):



Зависимость времени выполнения сортировки слиянием от размера массива (qsort)

Вывод

В ходе тестирования производительности алгоритмов сортировки **(Merge Sort и qsort)** на двух различных платах **Sipeed LicheePi 4A** и **Banana Pi BPI-F3** были получены следующие результаты:

1. **Средняя разница в производительности** между платформами составила **≈10% в пользу Sipeed LicheePi 4A**.
2. **Merge Sort показал стабильные результаты на обоих устройствах**, однако задержки доступа к памяти на BPI-F3 оказались выше, что негативно сказалось на времени выполнения.
3. **qsort (стандартный stdlib.h) оказался быстрее Merge Sort** на обоих процессорах, что ожидаемо, поскольку он оптимизирован для работы с кеш-памятью и использует эффективные методы разбиения данных.
4. **Профилирование с perf выявило, что горячим кодом являются инструкции загрузки (LD), записи (SD), а также сравнений (BGE, BLT)**, которые чаще всего выполняются в процессе сортировки.

Основные причины отставания Banana Pi BPI-F3

- **Процессор SpacemiT K1 на BPI-F3 менее производительный**, чем T-Head TH1520 на LicheePi 4A.
- **Меньшая пропускная способность памяти на BPI-F3** приводит к увеличению времени доступа при операциях LD и SD.
- **Различия в архитектуре кеш-памяти** могут также вносить вклад в снижение производительности.