Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Wprowadzenie do wysokowydajnych komputerów
Sprawozdanie laboratoryjne

Laboratorium nr 2

Hubert Cioroch

Prowadzący - mgr inż. Przemysław Świercz

26 marca 2025

1 Wstęp

Celem laboratorium było zaimplementowanie algorytmu Quick Sort w języku C oraz przeprowadzenie pomiarów czasu jego wykonania za pomocą instrukcji RDTSC w języku asemblera. Algorytm QuickSort działa na zasadzie dziel i rządz. Główna idea polega na podziale tablicy na dwie części względem wybranego elementu, zwanego pivotem, a następnie na osobnym sortowaniu każdej z tych części. Proces ten przebiega rekurencyjnie, aż do uzyskania posortowanej tablicy.
Algorytm składa się z następujących etapów:

1. **Wybór pivotu** – Wybierany jest element, względem którego będzie dokonywany podział tablicy. Może to być pierwszy, ostatni lub środkowy element tablicy.

2. **Podział tablicy** – Tablica jest reorganizowana w taki sposób, że wszystkie elementy mniejsze od pivotu znajdują się po jego lewej stronie, a większe po prawej.

3. **Rekurencyjne sortowanie** – Dwie podtablice (lewa i prawa) są sortowane niezależnie przy użyciu tej samej procedury.

Proces powtarza się, aż każda część tablicy zostanie posortowana.
Złożoność QuickSort zależy od sposobu wyboru pivotu oraz struktury wejściowej tablicy:

1. **Średnia i optymalna złożoność: O(n log n)** – gdy pivot dzieli tablicę na dwie części o podobnej wielkości.

2. **Najgorszy przypadek: $\mathbf{O(n^2)}$** – gdy pivot dzieli tablicę na bardzo nierówne części (np. zawsze wybierany jest skrajny element, a tablica jest już posortowana).

Mimo pesymistycznej złożoności $O(n^2)$, QuickSort w praktyce działa bardzo szybko i często przewyższa inne algorytmy sortowania.

2 Przebieg zadania

Najpierw zaimplementowano algorytm sortowania szybkiego w języku C, który rekurencyjnie dzieli tablicę na mniejsze części i sortuje je osobno.
Zadeklarowano trzy różne zestawy danych w postaci tablic:

- arr – tablica z losowymi wartościami,
- arrSorted – tablica uporządkowana rosnąco,
- arrReversed – tablica uporządkowana malejąco.

W celu zmierzenia wydajności algorytmu zaimplementowano kod w języku Assembly. Pomiar realizowano przy użyciu instrukcji RDTSC, która odczytuje licznik cykli procesora. Proces pomiaru wyglądał następująco:

1. Zapisanie wartości początkowej licznika przed uruchomieniem algorytmu.
2. Wywołanie funkcji quickSort.
3. Odczytanie wartości licznika po zakończeniu sortowania.
4. Obliczenie różnicy między wartościami przed i po wykonaniu algorytmu.
5. Wyświetlenie wyniku w postaci liczby cykli procesora potrzebnych do posortowania tablicy.

3 Wyniki

Nr	**Losowa**	**Posortowana**	**Odwrotnie posortowana**
1	13264	46556	34948
2	13570	43946	38888
3	14092	43792	35994
4	13786	41236	38650
5	15174	40382	40718
6	12996	41404	37070
7	12952	40498	34930
8	12928	49656	35880
9	12746	44556	38538
10	14342	41662	35508

Tabela 1: Liczba cykli procesora dla prawego pivota

Nr	Losowa	Posortowana	Odwrotnie posortowana
1	11452	6028	7324
2	11722	6094	7374
3	14766	7476	12038
4	11952	6570	8020
5	12236	6486	7466
6	11906	6186	7060
7	11642	6014	7864
8	19766	6210	7510
9	11834	6090	7388
10	12514	6552	7830

Tabela 2: Liczba cykli procesora dla środkowego pivota

4 Wnioski

Wybór prawego pivota prowadzi do znacznie większej liczby cykli procesora w przypadku tablic posortowanych. Dzieje się tak, ponieważ tablica jest dzielona w nieoptymalny sposób, co prowadzi do głębokości rekurencji O(n) i całkowitej złożoności $O(n^2)$. Wybór środkowego pivota zapewnia lepszy podział tablicy, co skutkuje znacznie mniejszą liczbą operacji dla danych posortowanych i odwrotnie posortowanych.