Przetwarzanie współbieżne

Laboratorium 1

Jan Bartula

# Implementacja funkcji

## Funkcja operacja arytmetyczna

1. double calc\_test(int count) {
2. double test1 = 32190325454.0;
3. double test2 = 2.0;
4. double wynik;
5. for (int i = 0; i < count; i++) {
6. wynik = test1 - 1.2314 / test2 + 123.2131 - 1231.23 \* 12312 - test1 / 2321.32342;
7. }
8. return wynik;
9. }

## Funkcja operacja I/O

1. void io\_test(int count) {
2. for (int i = 0; i < count - 1; i++) {
3. printf("H");
4. }
5. printf("\n");
6. }

## Main

1. int main() {
2. int count = 9999999;
4. inicjuj\_czas();
5. io\_test(count);
6. printf("Test Function I/O\n");
7. drukuj\_czas();
9. inicjuj\_czas();
10. calc\_test(count);
11. printf("Test Function Calculations\n");
12. drukuj\_czas();
13. }

# Zestawienie czasów

1. // Test Function I/O
2. czas standardowy = 0.307044
3. czas CPU = 0.092422
4. czas zegarowy = 2.733612
5. //Test Function Calculations
6. czas standardowy = 0.019048
7. czas CPU = 0.019048
8. czas zegarowy = 0.019049

## Analiza

Jak możemy zauważyć w czasie wykonywania obliczeń, czas użycia procesora jest praktycznie równy z czasem rzeczywistym wykonywania programu.

Jednak w przypadku funkcji, w której korzystamy z operacji I/O zauważamy, że czas zegarowy jest różny od czasu wykonywania obliczeń na procesorze. Oznacza to, że w czasie 2.74s, tylko 0.09s obliczenia wykonywał procesor. Procesor więc jest odpowiedzialny tylko za przekazanie informacji jakie mają zostać wyświetlone na ekranie lub np. plikach, jednak to w jakim czasie zostanie to wykonane nie zależy już od niego. Procesor czeka na wykonanie operacji przez układy wejścia/wyjścia.

## Wnioski

Dlatego trzeba pamiętać, że np. operując na plikach, nie należy wielokrotnie ich otwierać, czytać i zamykać. Najlepiej odczytać raz, przeprowadzić obliczenia w pamięci RAM i dopiero później zrobić zapis, a nie używać plików na dysku jako pamięć podręczna programu, ponieważ takie programy będą nieefektywne, niepotrzebnie wydłużając czas rzeczywisty wykonywania programu.