Tomasz Ligęza  
  
Programowanie równoległe. Przetwarzanie równoległe i rozproszone.

Sprawozdanie z laboratorium 1.

Celem laboratorium było nabycie (odświeżenie, pogłębienie) umiejętności realizacji pomiaru czasu wykonania fragmentów kodu oraz umiejętności posługiwania się narzędziem make do kompilacji i konsolidacji kodu.

W ramach zajęć zrealizowałem następujące kroki:

* Utworzenie wskazanych katalogów, oraz skopiowanie do nich pobranych uprzednio plików,
* Przeanalizowałem procedury pomiaru czasu w pliku pomiar\_czasu.c,
* Zmodyfikowałem plik moj\_program.c, w sposób opisany w poleceniu,
* Zmodyfikowałem plik Makefile, umożliwiający skompilowanie napisanego programu,
* Skompilowałem program poleceniem make,
* Dodałem kod źródłowy wypisujący czas realizacji drugiej pętli zawartej w pliku moj\_program.c,
* Przeprowadziłem po 5 pomiarów czasu realizacji programu w wersji do debugowania oraz zoptymalizowanej,
* Przeanalizowałem wyniki,

Plik źródłowy moj\_program.c (z pominiętymi fragmentami):

**#include "pomiar\_czasu.h"**

**...**

**inicjuj\_czas();**

**for (i = 0; i < liczba; i++) {**

**printf("%d ", k + i);**

**}**

**printf("Czas wykonania %d operacji wejscia/wyjscia: \n", liczba);**

**drukuj\_czas();**

**liczba \*= 100;**

**double t1 = czas\_zegara();**

**double t2 = czas\_CPU();**

**a = 1.000001;**

**for (i = 0; i < liczba; i++) {**

**a = 1.000001 \* a + 0.000001;**

**}**

**t1 = czas\_zegara() - t1;**

**t2 = czas\_CPU() - t2;**

**printf("Czas wykonania %d operacji arytmetycznych: \n", liczba);**

**printf("czas CPU = %lf\n", t2);**

**printf("czas zegarowy = %lf\n", t1);**

Zmiany w pliku Makefile:

**# jak uzyskac plik moj\_program.o ?**

**moj\_program.o: moj\_program.c pomiar\_czasu.h**

**$(CCOMP) -c $(OPT) moj\_program.c**

Analiza wyników:

Po przeprowadzeniu 5 pomiarów dla wersji do debugowania i zoptymalizowanej wyliczyłem średnią z pomiarów (zoptymalizowanych) i jej użyłem do dalszych obliczeń:

Czas wykonania pojedynczej operacji:

* Wejścia/wyjścia:
  + Czas CPU: 4.901E-08
  + Czas zegara: 5.23365E-06
* Arytmetycznej:
  + Czas CPU: 2.0091E-09
  + Czas zegara: 2.00896E-09

Optymalizacja przeprowadzona przez kompilator zmieniła czas drukowania, lecz w sposób bardzo nieznaczny ~10%. Inaczej ma się sytuacja przy operacjach arytmetycznych, gdyż tutaj można zaobserwować zysk w wysokości 39%.

W ramach zadań dodatkowych zrealizowałem:

* Przeniosłem pliki związane z pomiarem czasu poza katalog lab\_1,
* Utworzyłem bibliotekę z przeniesionych plików,
* Zmodyfikowałem plik Makefile, aby do pomiaru czasu program moj\_program wykorzystywał utworzoną bibliotekę.
* Skompilowałem ponownie pliki źródłowe i sprawdziłem poprawność działania.

Plik Makefile (najważniejsze fragmenty):

**moj\_program: moj\_program.o**

**$(LINKER) $(OPT) moj\_program.o -o moj\_program $(LIB)**

**# jak uzyskac plik moj\_program.o ?**

**moj\_program.o: moj\_program.c ../pomiar\_czasu/pomiar\_czasu.h**

**$(CCOMP) -c $(OPT) moj\_program.c $(INC)**

Wnioski:

* Użycie narzędzia make pozwala na zarządzania procesem budowania plików binarnych ze źródłowy w bardzo czytelny sposób,
* Pomiar czasu wykazał, że
  + Realizacja procedur wejścia i wyjścia zajmuje więcej czasu niż procedury arytmetyczne, oraz nie da się ich znacząco zoptymalizować,
  + Czas wykorzystania procesora powinien być niższy od czasu zewnętrznego, lecz w moim przypadku był on nieznacznie wyższy (np: cpu = 0.020348, zew = 0.020347). Być może miało to związek z kolejnością wywoływania funkcji czas\_zegara() oraz czasu\_CPU().