Politechnika Wrocławska

Wstęp do wysokowydajnych komputerów

Laboratorium nr. 4

Wykonawca:	
Imię i Nazwisko, nr indeksu, wydział	**Katarzyna Idasz** Wydział Informatyki i Telekomunikacji (W4N)
Termin zajęć: dzień tygodnia, godzina	Wtorek (11:15 – 13:00)
Numer grupy	1
Data wykonania ćwiczenia	14.05.2024 r.
Data oddania sprawozdania	20.05.2024 r.
Ocena końcowa	

Spis treści

1. Wstęp

1.1. Cel zadania

Należało napisać kod z wykorzystaniem x87, w którym obliczana jest całka oznaczona (1) metodą prostokątów.

$$\int_a^b \frac{x^4 - x}{1 - 3x} dx \tag{1}$$

Granicę należało wybrać tak, aby miała sens matematyczny (uwzględnienie dziedziny).

Następnie należało określić:

A. intensywność arytmetyczną
B. wydajność w GFLOPS oraz w IPC - *instructions per cycle*
C. zbadać dokładność dla:
    a. różnej precyzji obliczeń
    b. różnych trybów zaokrąglania

1.2. Metoda prostokątów

Jest jedną z metod całkowania numerycznego - polega na przybliżonym obliczaniu całek oznaczonych. Proste metody całkowania numerycznego polegają na przybliżeniu całki za pomocą odpowiedniej sumy ważonej wartości całkowanej funkcji w kilku punktach. Aby uzyskać dokładniejsze przybliżenie dzieli się przedział całkowania na niewielkie fragmenty. Ostateczny wynik jest sumą oszacowań całek w poszczególnych podprzedziałach.[1]

Metoda prostokątów polega na zastosowaniu wzoru (2)

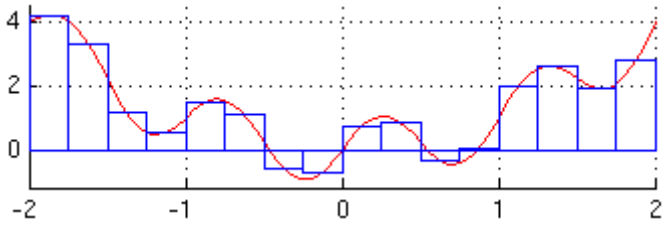
$$\int_{x_0}^{x_n} f(x) \approx h \sum_{i=0}^{n-1} f(x_i + \alpha h),\ h = \frac{x_n - x_0}{n} \tag{2}$$

w którym n jest liczbą przedziałów o długości h.

Metoda ta ma trzy warianty:

- lewych prostokątów, gdy $\alpha = 0$,
- średnich prostokątów, gdy $\alpha = \frac{1}{2}$ – ten wariant daje najlepsze przybliżenie,
- prawych prostokątów, gdy $\alpha = 1$.[2]

[1] Całkowanie numeryczne – Wikipedia
[2] Całkowanie numeryczne – Metoda prostokątów - Wikipedia



Obraz 1. Ilustracja metody prostokątów

2. Kody

2.1. Kod w C

Zaimplementowano metodę lewych prostokątów, czyli $\alpha = 0$.
Długość przedziału ustawiono na dx = 0.0001.
Liczona jest całka $\int_1^5 f(x)\,dx$.

```c
#include <stdio.h>

double dx = 0.0001;
double result = 0.0;

int main() {
    for(double x = 1.0; x <= 5.0; x += dx) {
       result += (x*x*x*x - x)/(1.0 - 3.0*x) * dx;
    }

    return 0;
}
```

Kod 1. lab4.c

2.2. Kod asemblerowy

Aby wygenerować kod z wykorzystaniem x87 użyto
$ gcc -S -mno-sse lab4.c

```
      .file   "lab4.c"
    .text
    .globl   dx
    .data
    .align 8
    .type    dx, @object
    .size    dx, 8
dx:
    .long    3944497965
    .long    1058682594
```

```
    .globl    result
    .bss
    .align 8
    .type    result, @object
    .size    result, 8
result:
    .zero    8
    .text
    .globl    main
    .type    main, @function
main:
.LFB0:
    .cfi_startproc
    pushq    %rbp
    .cfi_def_cfa_offset 16
    .cfi_offset 6, -16
    movq    %rsp, %rbp
    .cfi_def_cfa_register 6
    fld1
    fstpl    -8(%rbp)
    jmp    .L2
.L3:
    fldl    -8(%rbp)
    fmull    -8(%rbp)
    fmull    -8(%rbp)
    fmull    -8(%rbp)
    fsubl    -8(%rbp)
    fldl    -8(%rbp)
    fldl    .LC1(%rip)
    fmulp    %st, %st(1)
    fld1
    fsubp    %st, %st(1)
    fdivrp    %st, %st(1)
    fldl    dx(%rip)
    fmulp    %st, %st(1)
    fldl    result(%rip)
    faddp    %st, %st(1)
    fstpl    result(%rip)
    fldl    dx(%rip)
    fldl    -8(%rbp)
    faddp    %st, %st(1)
    fstpl    -8(%rbp)
.L2:
    fldl    -8(%rbp)
    fldl    .LC2(%rip)
    fucomip    %st(1), %st
```

```
    fstp    %st(0)
    jnb     .L3
    movl    $0, %eax
    popq    %rbp
    .cfi_def_cfa 7, 8
    ret
    .cfi_endproc
.LFE0:
    .size   main, .-main
    .section    .rodata
    .align 8
.LC1:
    .long   0
    .long   1074266112
    .align 8
.LC2:
    .long   0
    .long   1075052544
    .ident  "GCC: (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0"
    .section    .note.GNU-stack,"",@progbits
```

Kod 2. lab4.s

3. Intensywność arytmetyczna

$$I = \frac{N_A}{S_{MEM}} \tag{3}$$

$N_A -$ *liczba wykonanych operacji (arytmetycznych)*

$S_{MEM} -$ *rozmiar przesyłanych danych (ang. memory traffic)*

Do obliczeń wykorzystywane są tylko operacje zmiennoprzecinkowe, od linijki *.L3:* do linijki *jnb .L3* - jest to pętla *for*.

$$I = \frac{10}{15 \cdot 8} = \frac{1}{12} = 0,08(3)\ FLOP/B \tag{4}$$

4. Wydajność

4.1. GFLOPS

$$n = \frac{end - start}{precision} = \frac{5-1}{0,0001} = 40\,000 \tag{5}$$

$$P = \frac{n \cdot N_A}{t} = \frac{40\,000 \cdot 10}{0,002557632} \approx 156\,394\,665 \approx 0,156\ GFLOPS \tag{6}$$

$N_A -$ $liczba\ wykonanych\ operacji\ (arytmetycznych)$

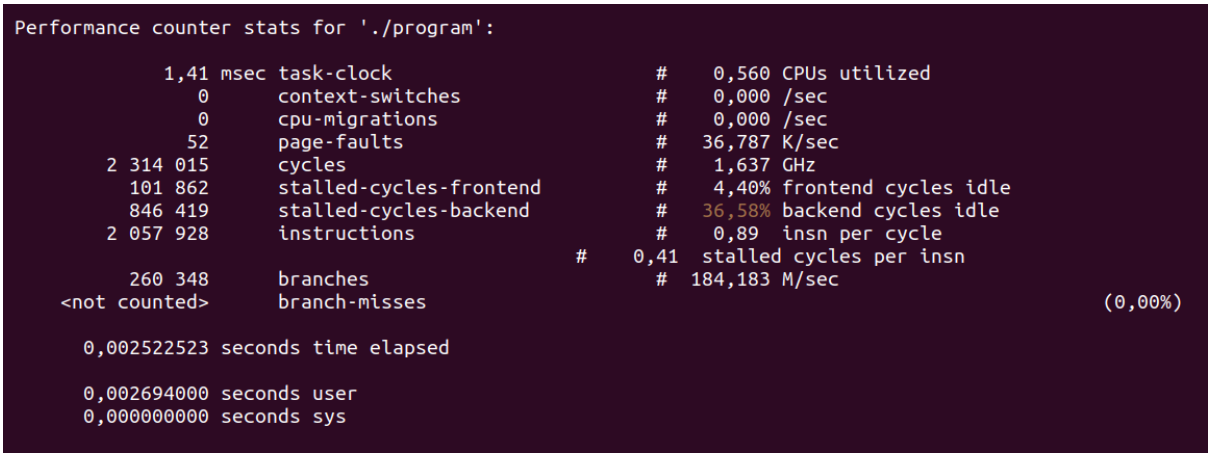
$n -$ $ilość\ przejść\ pętli$

$t -$ $czas\ w\ [s]$

Do wzoru (5) podstawiono wartości, które zostały wykorzystane w kodzie (1). Do obliczenia czasu w równaniu (6) wykorzystano *perf stat*. Wykonano 20 pomiarów. Wyniki były zbliżone - brak odstających wartości - dlatego do wzoru podstawiono ich średnią.

Tabela 1. Czas w [s] wykoanania programu

Nr pomiaru	Czas [s]
1	0,002522523
2	0,002410703
3	0,002676247
4	0,002601520
5	0,002349736
6	0,002621429
7	0,002577239
8	0,002615718
9	0,002661511
10	0,002595498
11	0,002487453
12	0,002413548
13	0,002547856
14	0,002648735
15	0,002434499
16	0,002539692
17	0,002595020
18	0,002632327
19	0,002643008
20	0,002578385
Średnia	0,002557632



Obraz 2. Przykład perf stat

4.2. IPC

$$P = \frac{n \cdot N_F}{C} = \frac{40\,000 \cdot 24}{612836} \approx 1,57 \text{ IPC} \tag{7}$$

$N_F -$ *liczba wykonanych operacji zmiennoprzecinkowych*

$n -$ *ilość przejść pętli*

$C -$ *ilość cykli*

Do obliczenia ilości cykli w równaniu (7) wykorzystano *rdtsc* - kod z poprzednich zajęć - time.s. Odczytano liczbę cykli przed wejściem do pętli oraz po wyjściu z niej, a następnie odjęto te wartości od siebie (analogicznie jak na poprzednich laboratoriach).
Wykonano 20 pomiarów. Wyniki były zbliżone - brak odstających wartości - dlatego do wzoru podstawiono ich średnią.

```
.global readTime
.type readTime, @function

readTime:
    push %ebx
    xor %eax, %eax
    cpuid
    rdtsc
    pop %ebx
    ret
```

Kod 3. time.s

Tabela 2. Ilość cykli

Nr pomiaru	Ilość cykli
1	608640
2	608640
3	620100
4	614880
5	608670
6	608640
7	608670
8	622470
9	608610
10	608610
11	608640
12	620130
13	615660
14	608640
15	608640
16	608640
17	608640
18	624060
19	613890
20	611850
Średnia	612836