МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

**НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №1

по курсу «Архитектура современных микропроцессоров и мультипроцессоров»

**ОЦЕНКА ПРОИЗВОДИТЕЛЬНОСТИ МИКРОПРОЦЕССОРА НА ЗАДАННЫХ ОПЕРАЦИЯХ**

**Выполнил:** студент 3-го курса гр. 17208

Гафиятуллин А.Р

Новосибирск, 2020

1. **ЦЕЛИ РАБОТЫ:**
   1. научиться оценивать производительность микропроцессора на заданных операциях.
2. **ХОД РАБОТЫ:**
   1. Написана программа, выполняющая многократно (в цикле) операцию целочисленного деления:
      1. с использованием последовательности зависимых операций (оценка латентности);
      2. с использованием последовательности независимых операций (оценка темпа выдачи результатов).

Листинг программы:

1. #include <stdio.h>
2. #include <time.h>
3. #include <x86intrin.h>
4. #ifndef WARMING\_UP\_SECONDS
5. #define WARMING\_UP\_SECONDS 10
6. #endif
7. #ifndef CYCLE\_NUM
8. #define CYCLE\_NUM 100000000
9. #endif
10. int main**()** **{**
11. unsigned long long start **=** 0**;**
12. unsigned long long end **=** 0**;**
13. unsigned x **=** 1 **<<** 31**;**
14. unsigned y **=** 2**;**
15. unsigned result **=** 0**;**
16. double clocks\_per\_empty\_iter **=** 0**;**
17. // warming up a processor
18. time\_t start\_time **=** time**(NULL);**
19. **while** **(**time**(NULL)** **-** start\_time **<** WARMING\_UP\_SECONDS**)** **{**
20. result **=** x **/** y**;**
21. **}**
22. printf**(**"Processor is warmed up! result = %d\n"**,** result**);**
23. // count cycle clocks
24. int i**;**
25. start **=** \_\_rdtsc**();**
26. **for(**i **=** 0**;** i **<** CYCLE\_NUM**;** i**++)** **{}**
27. end **=** \_\_rdtsc**();**
28. clocks\_per\_empty\_iter **=** **(**double**)(**end **-** start**)** **/** i**;**
29. printf**(**"Amount of clocks per empty iteration: %lf\n"**,** clocks\_per\_empty\_iter**);**
30. //---------------------------independent operations-------------------
31. // count clocks for cycle with divisions
32. start **=** \_\_rdtsc**();**
33. **for(**i **=** 0**;** i **<** CYCLE\_NUM**;** i**++)** **{**
34. result **=** x **/** y**;**
35. **}**
36. end **=** \_\_rdtsc**();**
37. printf**(**"Amount of clocks per iteration with division: %lf, result = %d\n\n"**,**
38. **(**double**)(**end **-** start**)** **/** i**,** result**);**
39. // real division clocks
40. printf**(**"Division clocks for independent operations: %lf\n\n"**,**
41. **(**double**)(**end **-** start**)** **/** i **-** clocks\_per\_empty\_iter**);**
42. //---------------------------Dependent operations---------------------
43. x **=** 1 **<<** 31**;**
44. start **=** \_\_rdtsc**();**
45. **for(**i **=** 0**;** i **<** CYCLE\_NUM**;** i**++)** **{**
46. x **=** x **/** y**;**
47. **}**
48. end **=** \_\_rdtsc**();**
49. printf**(**"Amount of clocks per iteration with division: %lf, result = %d\n\n"**,**
50. **(**double**)(**end **-** start**)** **/** i**,** x**);**
51. // real division clocks
52. printf**(**"Division clocks for dependent operations: %lf\n\n"**,**
53. **(**double**)(**end **-** start**)** **/** i **-** clocks\_per\_empty\_iter**);**
54. **return** 0**;**
55. **}**

Исходный код был скомпилирован с ключом -O0, далее был проанализирован ассемблерный код, генерируемый компилятором. Во всех циклах, которые были важны для оценки производительности, обращения к памяти на стеке были заменены на обращения к регистрам. Была так же произведена раскрутка циклов, чтобы уменьшить влияние накладных расходов на организацию цикла на конечные результаты оценки производительности операции целочисленного деления. Перед началом тестов был произведен «прогрев» процессора, чтобы достичь стабильной частоты.

Ассемблерные листинги данных циклов:

* цикл для оценки числа тактов на итерацию холостого цикла:

#**-------------------------------** empty cycle **-------------------------------**

movl $0**,** %ecx

**jmp** .L5

.L6**:**

addl $1**,** %ecx

.L5**:**

cmpl $99999999**,** %ecx

**jle** .L6

movl %ecx**,** **-**40**(%rbp)**

#**-------------------------------** empty cycle **-------------------------------**

* цикл для оценки темпа выдачи:

#**-------------------------------** independent operations **---------------------**

movl $0**,** %ecx

movl **-**48**(%rbp),** %edi

movl **-**36**(%rbp),** %ebx

**jmp** .L11

.L12**:**

movl %edi**,** %eax

movl $0**,** %edx

divl %ebx

movl %edi**,** %eax

movl $0**,** %edx

divl %ebx

movl %edi**,** %eax

movl $0**,** %edx

divl %ebx

movl %edi**,** %eax

movl $0**,** %edx

divl %ebx

addl $4**,** %ecx

.L11**:**

cmpl $2499999**,** %ecx

**jle** .L12

movl %eax**,** **-**44**(%rbp)**

movl %ecx**,** **-**40**(%rbp)**

#**-------------------------------** independent operations **---------------------**

* цикл для оценки латентности:

#**-------------------------------** dependent operations **-----------------------**

movl $0**,** %ecx

movl **-**48**(%rbp),** %eax

movl **-**36**(%rbp),** %ebx

**jmp** .L19

.L20**:**

movl $0**,** %edx

divl %ebx

movl $0**,** %edx

divl %ebx

movl $0**,** %edx

divl %ebx

movl $0**,** %edx

divl %ebx

addl $4**,** %ecx

.L19**:**

cmpl $2499999**,** %ecx

**jle** .L20

movl %eax**,** **-**48**(%rbp)**

movl %ecx**,** **-**40**(%rbp)**

#**-------------------------------** dependent operations **-----------------------**

* 1. Оценки производительности (в тактах):

|  |  |  |  |
| --- | --- | --- | --- |
|  | Intel Core i5 7200U  Skylake-X (2.5 GHz, 3.1 GHz Turbo Boost) | Intel Core i7 9700F  Coffee Lake (3.0 GHz, 4.7 GHz Turbo Boost) | Agner Fog (Skylake-X) |
| Пустой цикл | 1.757505 | 1.314142 |  |
| Темп выдачи | 5.484572 | 4.091744 | 6 |
| Темп выдачи с поправкой на Turbo Boost | **6.800869** | **6.410398** |
| Латентность | 20.271018 | 15.632926 | 23 |
| Латентность с поправкой на Turbo Boost | **25.136062** | **24.491584** |

Можно заметить, что при учитывании режима Turbo Boost полученные результаты очень близки к тем, что получил Агнер Фог.

Документация Intel не дает своих оценок для операции целочисленного деления, указывая лишь то, что при различном количестве значащих битов в операндах количество тактов, как для латентности, так и для темпа выдачи может существенно различаться.

1. **ВЫВОДЫ:**
   1. научились оценивать производительность микропроцессора на заданных операциях;
   2. узнали про отличия латентности и темпа выдачи, как характеристик микропроцессорных инструкций;
   3. написали программу для оценки латентности и темпа выдачи для операции целочисленного деления на x86-64 процессоре;
   4. поработали с ассемблерными листингами программы;
   5. получили оценки латентности и темпа выдачи для операции целочисленного деления на двух компьютерах и сравнили эти результаты с другими оценками;
   6. нужно стараться писать программы с небольшими информационными зависимостями, чтобы операции работали как можно быстрее.