МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №2

по курсу «Архитектура современных микропроцессоров и мультипроцессоров»

ОПРЕДЕЛЕНИЕ РАЗМЕРА БУФЕРА ПЕРЕУПОРЯДОЧИВАНИЯ КОМАНД ПРОЦЕССОРА

Выполнил: студент 3-го курса гр. 17208

Гафиятуллин А.Р

Новосибирск, 2020

1. **ЦЕЛИ РАБОТЫ:**
   1.1. научиться определять размер буфера переупорядочивания команд процессора.
2. **ХОД РАБОТЫ:**
   2.1. Для достижения цели написано две программы:
   - программа на Си, выполняющая операции и замеряющая время в тактах процессора;
   - программа на Python, генерирующая и компилирующая программу на Си, а также получающая и обрабатывающая результаты (отрисовка графики).

Программа на Си содержит в себе директивы условной компиляции для удобной сборки на разных процессорах.

Листинг программы на Си:

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#ifdef __x86_64__
#include <x86intrin.h>
#endif

#ifndef SLEEP_TIME
#define SLEEP_TIME 1
#endif

#ifndef CYCLE_NUM

#ifdef __arm__
#define CYCLE_NUM 10000       // а иначе долго ждать, компьютер слабенький
#else
#define CYCLE_NUM 10000000
#endif

#endif

#ifndef ARRAY_SIZE
#define ARRAY_SIZE (12 * 1024 * 1024 * 10)  // 10 x LLC
#endif

#ifdef __arm__
static inline unsigned long long __ccnt() {
  unsigned long long cycles = 0;
  __asm__ volatile ("mrc p15, 0, %0, c9, c13, 0":"=r" (cycles));
  return cycles;
}
#endif

int main() {
```

```
    // prepare
    unsigned long long start = 0;
    unsigned long long end = 0;

    int* array = malloc(ARRAY_SIZE * sizeof(int));
    for(size_t i = 0; i < ARRAY_SIZE; i++) {
        array[i] = rand() % ARRAY_SIZE;
    }

    // test
    int k = 0;
    sleep(SLEEP_TIME);
#ifndef __arm__
    start = __rdtsc();  // x86
#else
    start = __ccnt();   // arm
#endif
    for(size_t i = 0; i < CYCLE_NUM; i++) {
        k = array[k];
#include "nops" // file with asm("nop");
    }
#ifndef __arm__
    end = __rdtsc();    // x86
#else
    end = __ccnt();     // arm
#endif

    // save results
    printf("%llu", (end - start) / CYCLE_NUM);
    return 0;
}
```

Листинг программы на Python:

```
import subprocess
from subprocess import Popen, PIPE
import signal
import matplotlib.pyplot as plt
import platform

nops_amount = 1
max_nops_amount = 500
results = []
stop = False


def stop_benchmark(signal, frame):
    global stop
    print("Benchmark is stopped!")
    stop = True


signal.signal(signal.SIGINT, stop_benchmark)
log = open("compute.log", "w")
while not stop and nops_amount <= max_nops_amount:
```

```
    nops = open("nops", "w")
    for i in range(nops_amount):
        nops.write("asm(\"nop\");\n")
    nops.close()
    subprocess.call(["gcc", "main.c", "-O0"])

    with Popen(["./a.out"], stdout=PIPE) as test:
        try:
            cycles = int(test.stdout.read().decode("ascii"))
        except:
            continue
        print("nops command amount: " + str(nops_amount) + ", clocks: " +
str(cycles))
        log.write(str(nops_amount) + " " + str(cycles) + "\n")
        results.append(cycles)
    nops_amount = nops_amount + 1

log.close()

plt.plot(range(nops_amount)[1:], results)
plt.xlabel("nops")
plt.ylabel("cycles")
plt.title("nops-cycles dependency " + platform.machine())
plt.show()
```

2.2. Описание тестируемых архитектур:

2.2.1. **Intel Coffee Lake,** реализующий набор инструкций **x86_64**:

2.2.1.1. теоретический размер буфера переупорядочивания - **224 инструкции**;

2.2.1.2. в качестве представителя этой микроархитектуры был взят **Intel Core i7-9700F (12M кэш, до 4.70 GHz).**

2.2.2. **ARM Cortex-A72** в качестве реализации набора инструкций **ARMv8-A**:

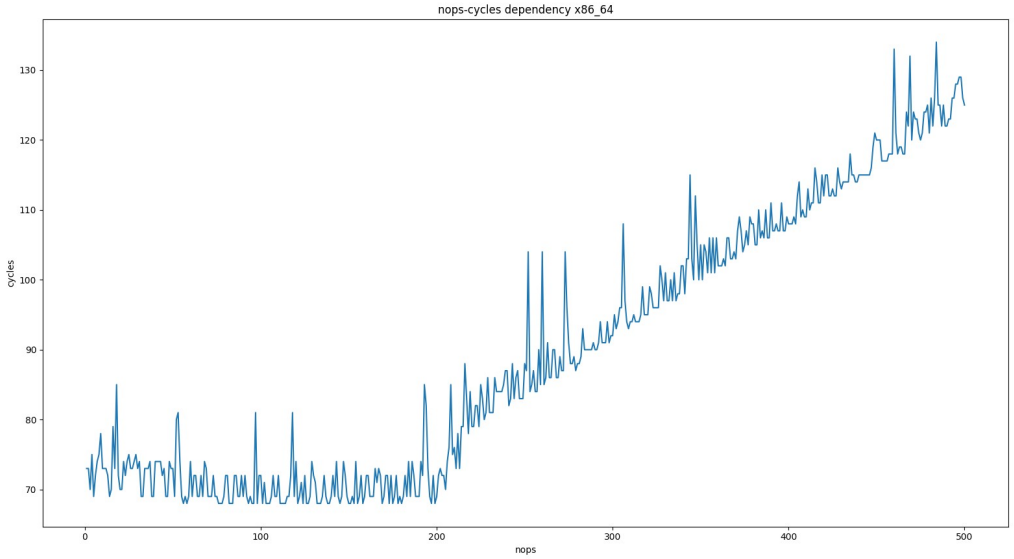
2.2.2.1. теоретический размер буфера переупорядочивания – **128 инструкций**;

2.2.2.2. в качестве представителя этой микроархитектуры был взят **Broadcom BCM2711 (кэша 3-го уровня нет, до 1.5 GHz).**

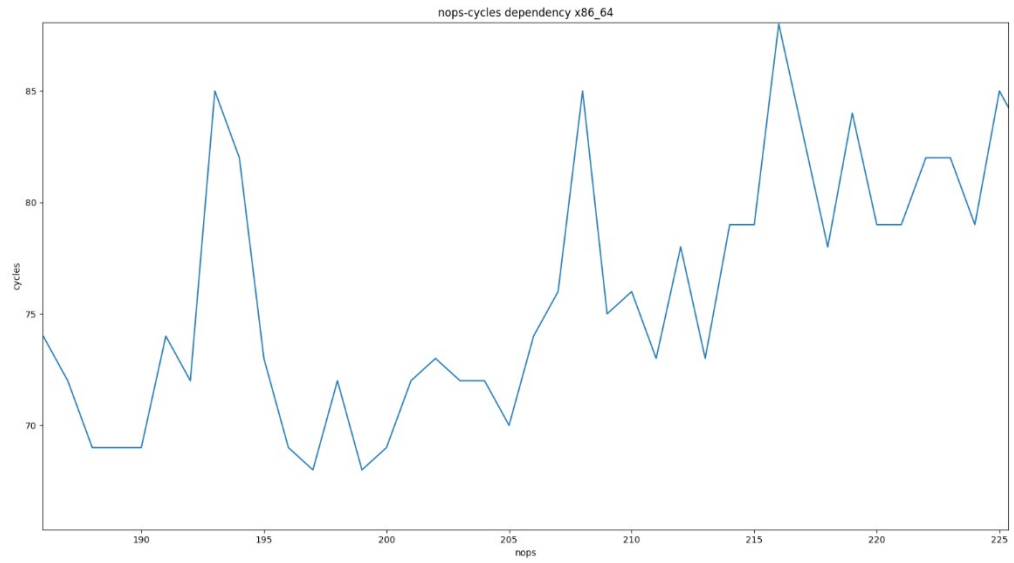
2.3. Результаты:

2.3.1. **Intel Coffee Lake:**

График зависимости количества тактов от количества «нопов»:



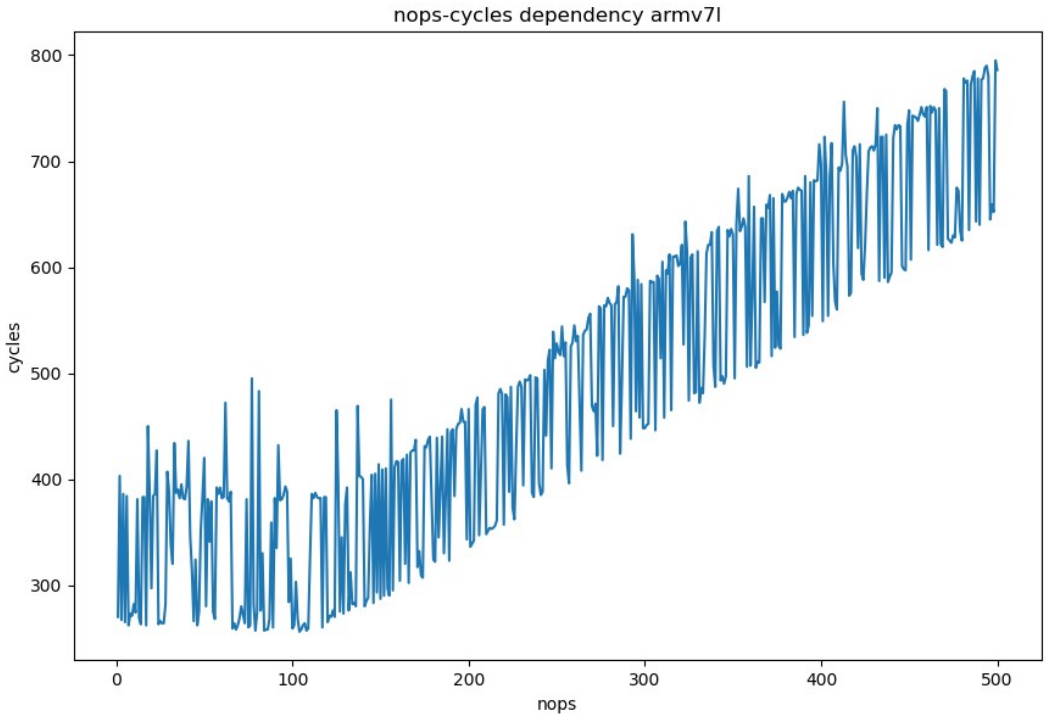
Характерная точка на этом графике (место начала роста):



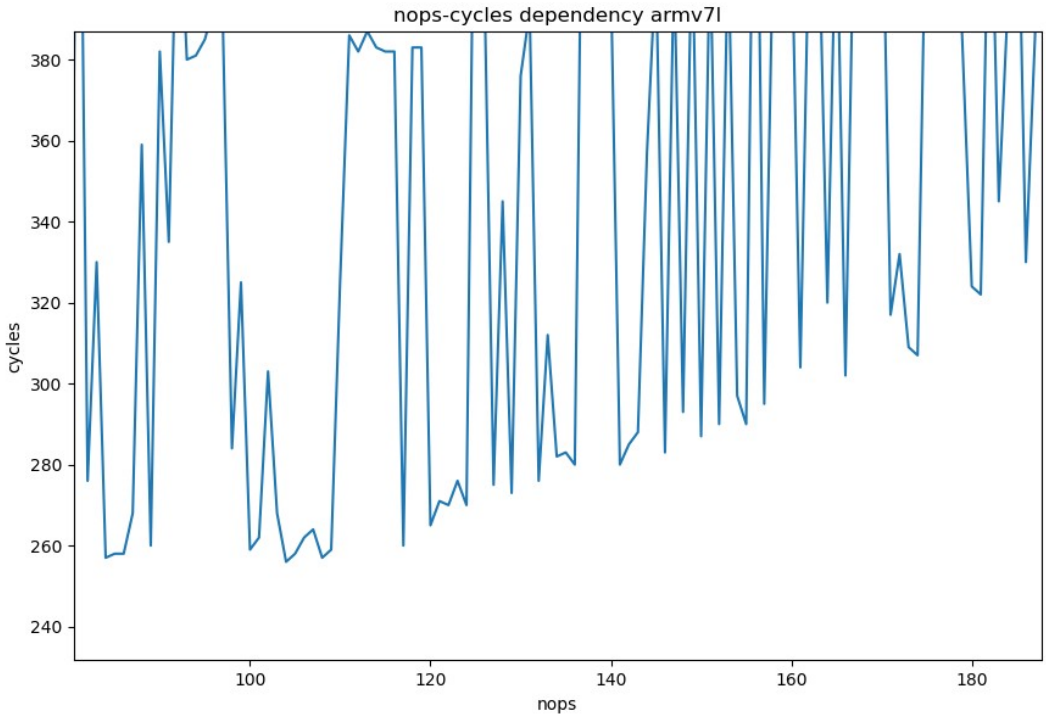
Постоянный рост начинается примерно в диапазоне 205 – 210 «нопов», что немного ниже 224, но довольно близко. Возможно, можно списать на погрешность измерений.

2.3.2. **ARM Cortex-A72:**

График зависимости количества тактов от количества «нопов»:



Характерная точка на этом графике (место начала роста):



Постоянный рост начинается в области 120-140 «нопов», что очень близко к теоретическому значению 128, можно сказать, что совпадает.

3. ВЫВОДЫ:

3.1. научились определять размер буфера переупорядочивания команд процессора;

3.2. теоретические и расчетные показатели совпадают на обеих архитектурах;

3.3. энергоэффективные мобильные ARM-процессоры при цене на порядок ниже проигрывают в этом показателе своим старшим x86-собратьям **всего** в 2 раза.

4. ПРИЛОЖЕНИЕ:

4.1. Графики в исходном разрешении:

nops-cycles-dependency-x86-64.png

nops-cycles-dependency-x86-64-key-moment.png

nops-cycles-dependency-armv7l.png

nops-cycles-dependency-armv7l-key-moment.png

4.2. Логи вычислений, по которым можно отстроить эти графики:

compute-x86-64.log

compute-arm.log

4.3. Скрипт для отстройки графиков по логам (передать нужный файл в качестве аргумента):

```
import matplotlib.pyplot as plt
import sys
import platform

if len(sys.argv) < 2:
    raise Exception("usage: python3 build_plot.py compute.log")

log = open(sys.argv[1], "r")
lines = log.readlines()

x_axis = []
y_axis = []

for line in lines:
    data = line.split()
```

```
    x_axis.append(int(data[0]))
    y_axis.append(int(data[1]))

plt.plot(x_axis, y_axis)
plt.xlabel("nops")
plt.ylabel("cycles")
plt.title("nops-cycles dependency " + platform.machine())
plt.show()
```