МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «МИРЭА – Российский технологический университет»

Институт информационных технологий

Кафедра вычислительной техники

Отчет

По лабораторному практикуму

дисциплины:

«Архитектура вычислительных машин и систем»

Исполнитель: Желудков Данила Михайлович

студент группы ИКБО-07-19

Руководитель: Скрябин Юрий Михайлович

ассистент кафедры ВТ

Москва 2020

**Лабораторная работа №1**

**ПРОЦЕССОР С РЕГИСТРОВОЙ ПАМЯТЬЮ**

*Цель работы*: изучение аппаратной реализации операций уможения и деления со знаковыми и беззнаковыми числами

*Задание к работе*: а) Найти свёртку двух массивов без учета возможности выхода результата за пределы 32 бит. б) Сосчитать количество битов необходимых под результат, т.е. количество значащих бит результата, если исходные данные состоят целиком из чисел, максимальных для указанной разрядности. Найти свёртку двух массивов, записав результат в два 32-битных регистра или в 8 байт оперативной памяти. 64 бит однозначно хватит любому варианту. (12 вариант)

Работу выполнил:

Практическая часть

Часть а)

section .data

Vec1: Times 60 dd 99, 99, 99, 99, 99

Vec2: Times 60 dd 99, 99, 99, 99, 99

sum1 dd 0; sum1 хранит сумму младших битов

sum2 dd 0; sum2 хранит сумму старших битов

section .text

global CMAIN

CMAIN:

mov rbp, rsp; for correct debugging

xor rax, rax

xor rbx, rbx

xor rcx, rcx

xor rdx, rdx

mov ecx, 300

for1:

mov eax, [Vec1 + (ecx - 1)\*4]

mov edx, [Vec2 + (ecx - 1)\*4]

mul edx

loop for1

PRINT\_DEC 4, [sum1]

NEWLINE

PRINT\_DEC 4, [sum2]

ret

Часть б)

Количество битов: (2^21 - 1) \* (2^21 - 1) \* 2^9 = 51 bit

%include "io64.inc"

section .data

Vec1: Times 60 dd 999999, 999999, 999999, 999999, 999999

Vec2: Times 60 dd 999999, 999999,999999, 999999, 999999

sum1 dd 0; sum1 хранит сумму младших битов

sum2 dd 0; sum2 хранит сумму старших битов

section .text

global CMAIN

CMAIN:

mov rbp, rsp; for correct debugging

xor rax, rax

xor rbx, rbx

xor rcx, rcx

xor rdx, rdx

mov ecx, 300

for1:

mov eax, [Vec1 + (ecx - 1)\*4]

mov edx, [Vec2 + (ecx - 1)\*4]

mul edx

mov ebx, [sum1]

add ebx, eax;так как eax хранит младшие биты

mov [sum1], ebx

mov ebx, [sum2]

adc ebx, edx;так как edx хранит старшие биты

mov [sum2], ebx

no\_adc:

loop for1

PRINT\_DEC 4, [sum1]

NEWLINE

PRINT\_DEC 4, [sum2]

ret