Федеральное государственное бюджетное образовательное учреждение   
высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Факультет ИУ – «Информатика и управление»

Кафедра ИУ-3 – «Информационные системы и телекоммуникации»

Отчёт по лабораторной работе № 1

«Исследование архитектуры и организации микропроцессора»

по дисциплине «Микропроцессорные устройства обработки сигналов»

Студент группы ИУ3-22М                                            М.К. Быковский

Преподаватель кафедры ИУ-3                                   А.И. Германчук

Москва, 2018

СОДЕРЖАНИЕ

Введение 3

1 Описание индивидуального задания 4

2 Реализация индивидуального задания 5

3 Результаты 6

4 Выводы 7

Приложение 1. Исходный код 8

# Введение

Целью лабораторной работы является знакомство с интегрированной средой разработки программ Code Composer Studio (CCS) компании Texas Instruments. Среда CCS предназначена для разработки программного обеспечения для микропроцессоров и микроконтроллеров, выпускаемых компанией Texas Instruments, и включает такие инструменты, как редактор исходных текстов, компилятор, компоновщик, отладчик и симуляторы.

Лабораторная работа заключается в написании в соответствие с выданным индивидуальным заданием текстов несложных программ на языке C и ассемблера, создании нового проекта в CCS, компиляции и отладке разработанных программ с помощью симулятора микропроцессора TMS320C5515 или оценочной платы TMS320C5515 DSP Evaluation Module (TMDXEVM5515).

# 1 Описание индивидуального задания

Темой индивидуального задания является разработка низкоуровневой функции, реализующей вычитание с заемом. Данная функция определяет разность двух операндов и, кроме того, уменьшает полученный результат на содержимое влага переноса.

AX = a – b – CF,

где AX – результат операции, a, b – значения операндов, CF – содержимое флага переноса.

Прототип низкоуровневой функции вычитания целых чисел с заемом на языке Си имеет следующий вид:

int \_subb(int src1, int src2),

где src1 и src2 – операнды, а возвращаемое значение функции – результат вычитания с заемом из src1 значения src2.

# 2 Реализация индивидуального задания

В ходе лабораторной работы был разработан код на языке ассемблер, реализующий требуемую функцию. В ходе выполнения требовалось:

* Используя отладчик, протрассируйте выполнение разработанной программы. Убедитесь в корректности работы разработанной программы и ее соответствии заданию.
* В случае необходимости остановите отладку, внесите изменение в исходные тексты программ и повторите процесс отладки. При внесении изменений воспользуйтесь справочными данными.
* Сохраните результаты работы программы (вывод в консоль среды разработки припомощи функции printf).

Ниже приведен спроектированный код:

.def \_subb

.ref \_d

**.data**

w **.word** 0FFFFh, 0FFA9h, 8765h, 4320h

**.text**

.asg \*SP(#0), var

**\_subb:**

PSH mmap(ST1\_55)

BSET SATA

**SUB** T1, T0

**MOV** mmap(ST0\_55), T2

**AND** 0x400, T2

**MOV** T2, HI(AC0)

SFTL AC0, #-11

**MOV** HI(AC0), T2

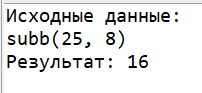
**SUB** T2, T0

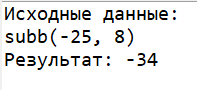
**POP** mmap(ST1\_55)

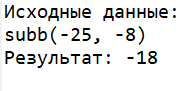
RET

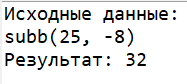
# 3 Результаты

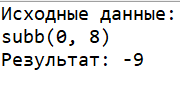
В ходе тестирования разработанной функции была выявлена корректность ее работы с различными исходными данными:

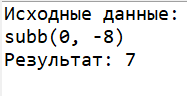


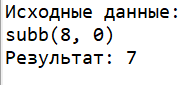


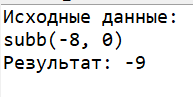


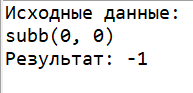












# 4 Выводы

В ходе лабораторной работы была реализована требуемая функция на языке ассемблер. Была изучена среда разработки Code Composer Studio (CCS) и отработаны методы и особенности написания программ на низкоуровневых языках, а также интеграция модулей низкоуровневых языков с высокоуровневыми.

# Приложение 1. Исходный код

Файл *main.c*

**#include** "stdio.h"

**int** **subb**(**int**, **int**);

**void** **main**(**void**)

{

**int** a = 0;

**int** b = 0;

**int** c;

c = subb(a, b);

**printf**( "Исходные данные:\nsubb(%d, %d)\n", a, b);

**printf**( "Результат: %d\n", c);

}

Файл *subb.asm*

.def \_subb

.ref \_d

**.data**

w **.word** 0FFFFh, 0FFA9h, 8765h, 4320h

**.text**

.asg \*SP(#0), var

**\_subb:**

PSH mmap(ST1\_55)

BSET SATA

**SUB** T1, T0

**MOV** mmap(ST0\_55), T2

**AND** 0x400, T2

**MOV** T2, HI(AC0)

SFTL AC0, #-11

**MOV** HI(AC0), T2

**SUB** T2, T0

**POP** mmap(ST1\_55)

RET