

ГУАП

КАФЕДРА № 43

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

---

доцент, канд.тех.наук\_

---

подпись, дата

---

Попов А.А.

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

Использование циклов

по дисциплине: АРХИТЕКТУРА ЭВМ И СИСТЕМ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТГР. №

4936

---

подпись, дата

---

Назаров М.Р.  
фамилия, инициалы

Санкт-Петербург 2021

## Лабораторная работа 2. Использование циклов

### Цель работы:

Освоение принципов построения приложений на языке ассемблера для системы Texas Instruments, ознакомление с командами и правилами построения программ в соответствии с особенностями организации циклов.

### Задание на лабораторную работу:

освоение принципов построения приложений на языке ассемблера для системы Texas Instruments, ознакомление с командами и правилами построения программ в соответствии с особенностями организации циклов.

### Варианты заданий:

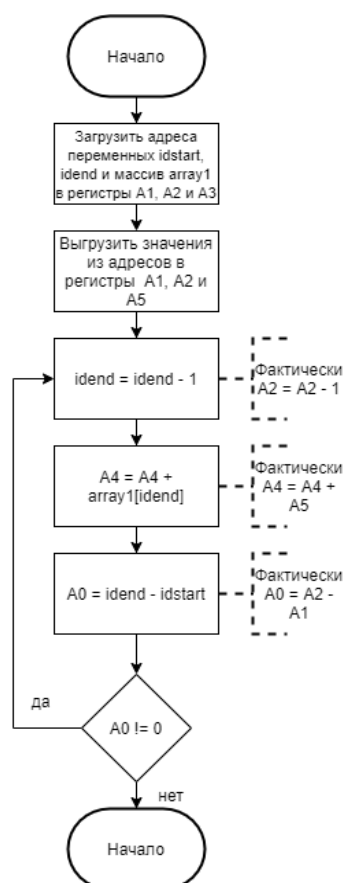
Вариант №13:

unsigned int

13. Разработать программу, вычисляющую сумму элементов массива между двумя заданными. Задаются индексы массива.

### Ход работы:

Создадим граф схему алгоритма:



Напишем код программы с комментариями:

```
.global _c_int00      ;точка входа
_c_int00:

.data                ;секция данных

idstart      .uint 1      ;индекс массива, от которого насчитается суммирование
idend .uint 5;индекс массива, до которого идет суммирование
array1:      .uint 1,2,3,4,5,6,7,8      ;создаем массив 32 разрядных чисел

.text            ;секция кода

MVK .S1 idstart,A1 ;Загрузить адрес регистра idstart в A1
MVK .S1 idend,A2   ;Загрузить адрес регистра idend в A2

LDW .D1 *A1,A1     ;Выгрузить значение регистра idstart в A1
LDW .D1 *A2,A2     ;Выгрузить значение регистра idend в A2

MVK .S1 array1,A3   ;загружаем адрес массива1 в A3
MVKH .S1 array1,A3
NOP 2 ;2x тактовая задержка загрузки






MVK .S1 0,A4        ;сумма элементов
MVK .S1 0,A5        ;тек. элемент выбираемый из массива 1

LOOP:
SUB .L1 A2,1,A2      ;A2 := A2 - 1






LDW .D1 *A3[A2], A5  ;загружаем текущий элемент в A5
NOP 4                ;4x тактовая задержка загрузки
ADD .L1 A4, A5,A4     ;сумма результатов умножения в A4
SUB .L1 A2, A1,A0     ;разница между индексом текущего элемента и индексом
начального элемента в A0
[A0] B .S1 LOOP      ;переход если A0 <> 0
NOP 5
```

Проверим работу программы:






Начальное состояние программы (все регистры пустые).

Name	Value	Description
 Core Registers		
 A0	0x00000000	Core
 A1	0x00000000	Core
 A2	0x00000000	Core
 A3	0x00000000	Core





Записанные адреса ячеек памяти в первые регистры.

Name	Value	Description
▼  Core Registers		
 A0	0x00000000	Core
 A1	0x00000040	Core
 A2	0x00000044	Core
 A3	0x00000048	Core








После выгрузки значений в соответствующие регистры и ожидания двух тактов

Name	Value
▼  Core Registers	
 A0	0x00000000
 A1	0x00000001
 A2	0x00000005
 A3	0x00000048







В регистр A2 записываем результат вычисления значения регистров A2 - 1

Name	Value
▼  Core Registers	
 A0	0x00000000
 A1	0x00000001
 A2	0x00000004



В регистр A5 запишем текущий элемент array1[A2] = 5

Name	Value
▼  Core Registers	
 A0	0x00000000
 A1	0x00000001
 A2	0x00000004
 A3	0x00000048
 A4	0x00000000
 A5	0x00000005








В регистр A4 запишем результат суммы значений A4 + A5.

Name	Value
▼  Core Registers	
 A0	0x00000000
 A1	0x00000001
 A2	0x00000004
 A3	0x00000048
 A4	0x00000005

В регистр A0 запишем разницу между idend и idstart

Name	Value
▼  Core Registers	
 A0	0x00000003

Пройдем по циклу еще несколько раз, пока idend-idstart != 0

name	value
▼  Core Registers	
 A0	0x00000000
 A1	0x00000001
 A2	0x00000001
 A3	0x00000048
 A4	0x0000000E
 A5	0x00000002

Результатом работы программы будет  $5 + 4 + 3 + 2 = 14 = 0E$

**Вывод:** в ходе лабораторной работы были получены основные навыки работы с таким низкоуровневым языком программирования как assembler, так же были изучены и применены на практике основные команды и директивы языка.