ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ _____

ПРЕПОДАВАТЕЛЬ

ассистент		И.Д. Свеженин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

ОРГАНИЗАЦИЯ РАБОТЫ С ПОДПРОГРАММАМИ

по курсу: Архитектура ЭВМ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4329		Д.С. Шаповалова
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

1. Цель работы:

Изучение организации вызова подпрограмм в языке ассемблера, передачи параметров через стек и возврат значений из функции.

2. Задание:

1. Сформировать исходные данные в соответствии с вариантом.
2. Составить алгоритм подпрограммы и основной программы для решения поставленной задачи.
3. Провести трассировку заданного алгоритма с использованием заданных исходных данных.
4. Составить программу заданного алгоритма в мнемокодах.
5. Оформить отчет по лабораторной работе.
6. В учебной лаборатории проверить результаты выполнения программы в программе-отладчике, сравнивая их с результатами ручной трассировки алгоритма.

Для всех заданий входные данные передаются в подпрограмму через стек, а результат возвращается через регистр AL. Для массивов входными данными являются адрес массива и число элементов в нем.

Вариант 4:

Найти сумму четных элементов массива.

3. Ход работы:

1. Определение исходных данных:

Были выбраны 10 чисел в размерности байт, в пределах от -128 до +127:

10, 15, 22, 33, 44, 51, 60, 77, 88, 99

2. Ручная трассировка алгоритма с использованием исходных данных:

Чётные элементы:

10 + 22 + 44 + 60 + 88 = 224

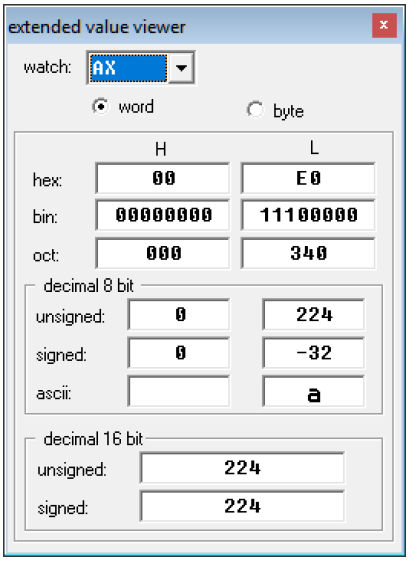

Рисунок 1 – Результат работы алгоритма

3. Описание алгоритма:

1. Загружаем в память массив из N байтов.
2. Загружаем в регистр CX количество элементов массива (N).
3. Обнуляем регистр AL — он будет накапливать сумму чётных элементов.
4. Инициализируем указатель (например, SI) адресом начала массива.
5. Переходим к обработке очередного элемента:
    a. Загружаем текущий элемент из памяти по адресу [SI] в регистр (например, DL).
    b. Выполняем команду TEST DL, 1 — это логическое И с 1, проверяющее младший бит числа.
    c. Если результат равен 0 (флаг ZF = 1), число чётное:
        1) Добавляем значение DL к AL.
    d. Если результат не равен 0 (флаг ZF = 0), число нечётное — пропускаем его.
6. Увеличиваем указатель SI на 1, чтобы перейти к следующему элементу массива.
7. Уменьшаем счётчик циклов CX на 1 (команда LOOP делает это автоматически).
8. Если CX ≠ 0, возвращаемся к шагу 5.
9. После завершения цикла результат (сумма чётных элементов) находится в регистре AL.
10. Записываем значение AL в ячейку памяти (если требуется) или оставляем его в регистре как результат работы подпрограммы.

4. Алгоритм в мнемокодах

```
.MODEL SMALL

; Stack
Sseg SEGMENT STACK 'stack'
    DB 256 DUP(?)
Sseg ENDS

; Data
Dseg SEGMENT 'data'
    Arr  DB 10, 15, 22, 33, 44, 51, 60, 77, 88, 99
    N    DW 10
    Result DB ?
Dseg ENDS

; Code
Cseg SEGMENT 'code'
    ASSUME CS:Cseg, DS:Dseg, SS:Sseg

; SubProgramm
; Input = BP+4 = address of array, BP+6 = hm elements
; Output = AL, %2 el-ts
SumEven PROC NEAR
    PUSH BP
    MOV BP, SP

    PUSH SI        ; save SI
    PUSH CX        ; save CX
    PUSH DX

    MOV SI, [BP+4] ; SI = address of array
    MOV CX, [BP+6] ; CX = hm el-ts
    XOR AX, AX     ; AX = 0 (start sum)

next_elem:
```

```
    MOV DL, [SI]    ; next el-t to DL
    TEST DL, 1      ; logic DL AND 1=== DL[SI} are %2?
    JNZ skip_add    ; jump if Not Zero = skip this
    ADD AL, DL      ; this %2! add this!

skip_add:
    INC SI          ; SI++ t onext elt
    LOOP next_elem

    ; result to AL

    POP DX
    POP CX
    POP SI
    POP BP
    RET
SumEven ENDP

; Main Program
Main PROC FAR
    PUSH DS
    MOV AX, 0
    PUSH AX

    MOV AX, Dseg
    MOV DS, AX

    ; Prepare parametrs: lengh, address
    PUSH N          ; hm elt-s (word)
    PUSH OFFSET Arr ; address start of array

    CALL SumEven    ; SubProgram

    ; save result
    MOV Result, AL
```

```
    ; exit to DOS
    RET
Main ENDP

Cseg ENDS

END Main
```

4. Вывод:

В ходе выполнения лабораторной работы №4 была изучена организация вызова подпрограмм на языке ассемблера, а также методы передачи параметров через стек и возврата результата через регистр. Была разработана подпрограмма SumEven, которая получает через стек адрес массива и количество его элементов, вычисляет сумму чётных элементов и возвращает результат в регистре AL.

Алгоритм корректно реализует проверку чётности элементов с помощью логической операции TEST, использует регистр CX как счётчик цикла и обеспечивает безопасную работу с регистрами путём их сохранения и восстановления в подпрограмме. Программа была протестирована вручную и с использованием эмулятора (например, Emu8086), что подтвердило правильность её работы.

Таким образом, цель работы достигнута: получены практические навыки работы с подпрограммами, стеком и условными операциями в ассемблере, а также закреплено понимание механизма передачи параметров и возврата результата, принятого в языках высокого уровня.