Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 **Информатика и вычислительная техника**

ОТЧЕТ

по домашнему заданию № 1

Название: Обработка символьной информации
Дисциплина: Машинно-зависимые языки и основы компиляции

Студент ИУ6-42Б _____ Э.Э. Джафаров
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ С.С. Данилюк
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Цель работы: изучение команд обработки цепочек и приемов обработки символьной информации.

Задание: Дан текст 8 слов по 6 символов. В словах с четным номером изменить порядок букв на обратный

1. **Схема**

Схема алгоритма представлена на рисунке 1

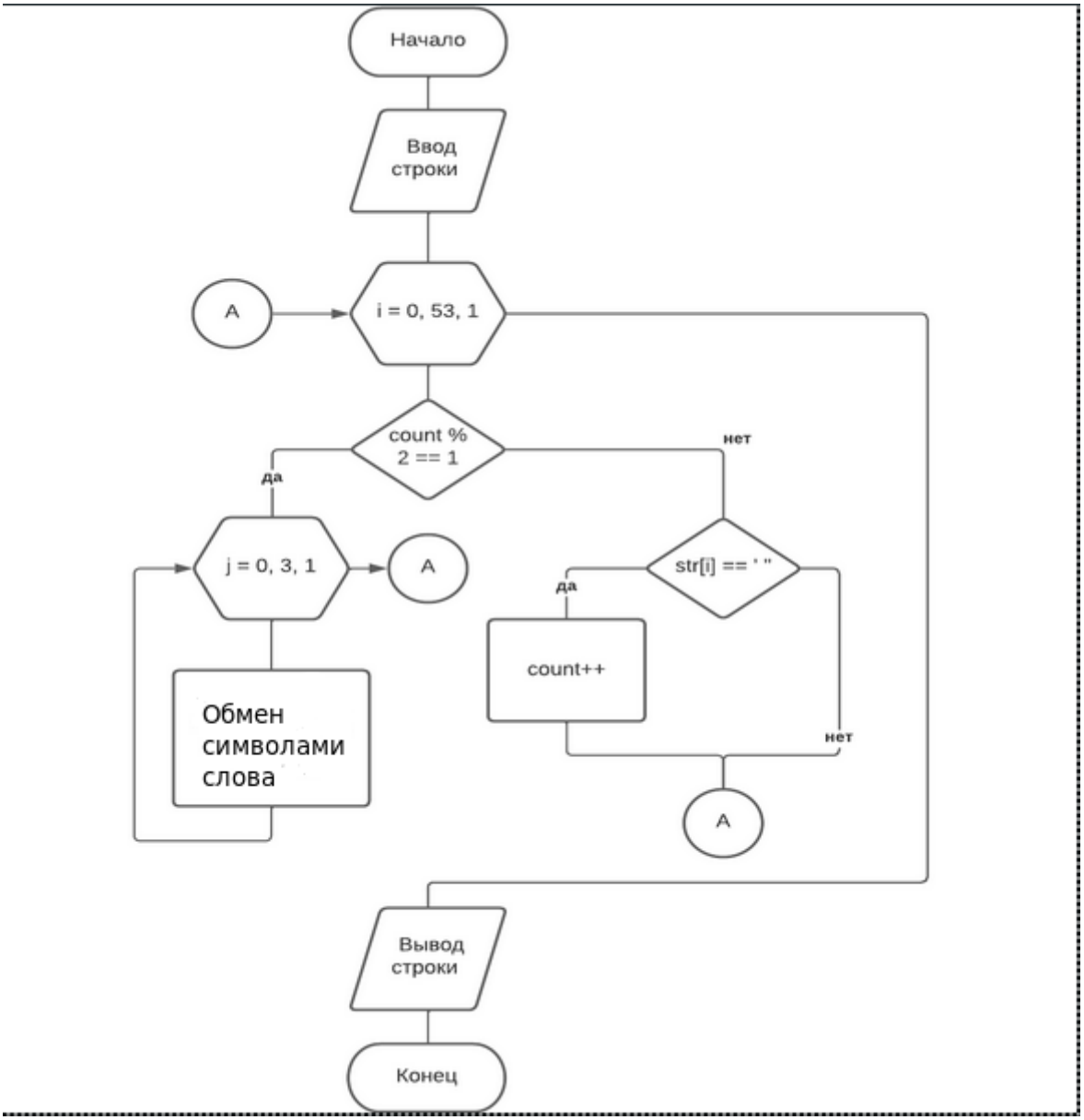

Рисунок 1 - Схема алгоритма

2. **Код программы:**

```
section .data
; сегмент инициализированных переменных
ExitMsg db "Result: ",10
lenExit equ $-ExitMsg
EnterMsg db "Enter string:",10
lenM equ $-EnterMsg
; сегмент неинициализированных переменных
section .bss
; буфер для вводимой строки
InBuf resb 100
lenIn equ $-InBuf
section .text ; сегмент кода
global _start
_start:
   ; write
   mov rax, 1 ; системная функция 1 (write)
   mov rdi, 1 ; дескриптор файла stdout=1
   mov rsi, EnterMsg ; адрес выводимой строки
   mov rdx, lenM ; длина выводимой строки
   syscall; вызов системной функции
   ; read
   mov rax, 0 ; системная функция 0 (read)
   mov rdi, 0 ; дескриптор файла stdin=0
   mov rsi, InBuf ; адрес буфера ввода
   mov rdx, lenIn ; размер буфера
   syscall ; вызов системной функции
   ; подсчет длины введенной строки до кода Enter
   lea rdi, [InBuf] ; загружаем адрес строки в edi
   mov rcx, 8 ; загружаем размер буфера ввода
   mov al,' ' ; загружаем в al пробел для поиска
   mov ebx,0 ; обнуляем счетчик пробелов
```

```
    cld

cic1:
    push rcx
    AND BX, 1; выполнить операцию AND с маской 1
    JNZ pod_swap ; перейти к метке pod_swap (если
результат не равен 0)
    mov rcx, 9
    repne scasb ; проверяем очередной символ на
пробел
    pop rcx
    je consl ; если пробел - меняем местами символы
    jmp exit

pod_swap:
    mov r9, 5 ; длина слова - 1
    push rcx
    mov rcx, 3 ; загружаем количество повторений -
длина слова / 2
    jmp swap

swap:
    push rcx
    mov al, BYTE[rdi+r9]
    mov cl, BYTE[rdi]
    mov [rdi], al
    mov [rdi+r9], cl
    sub r9, 2
    inc rdi
    pop rcx
    loop swap
    mov ebx, 0; обнуляем счетчик пробелов
```

```
    add rdi, 4; переходим к следующему слову
    mov al, ' '
    pop rcx
    loop cic1

consl: inc ebx ; иначе - увеличиваем счетчик слов
       loop cic1

exit:
    push rsi
    ;write text
    mov rax, 1 ; системная функция 1 (write)
    mov rdi, 1 ; дескриптор файла stdout=1
    mov rsi, ExitMsg
    mov rdx, lenExit ; длина строки
    syscall ; вызов системной функции

    ;write modified string
    mov rax, 1 ; системная функция 1 (write)
    mov rdi, 1 ; дескриптор файла stdout=1
    pop rsi
    mov rdx, lenIn ; длина строки
    syscall ; вызов системной функции

    ; exit
    mov rax, 60 ; системная функция 60 (exit)
    xor rdi, rdi ; код возврата 0
    syscall ; вызов системной функции завершения
```

Результат трансляции и компоновки программы представлен на рисунке 2.

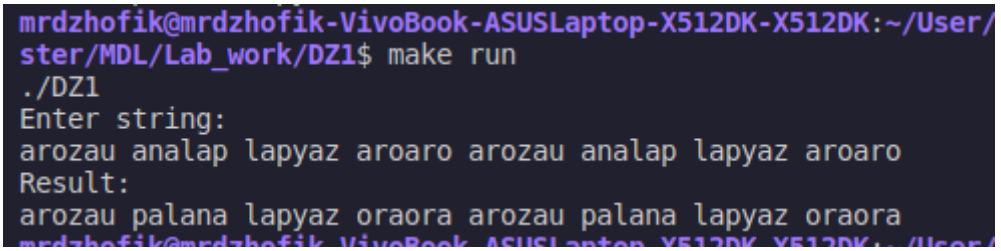

Рисунок 2 - Результат трансляции и компоновки программы

3. Результат работы программы

Результат работы программы на 3-х примерах представлен в таблице 1.

Таблица 1 - Результат работы программы

№	Исходные данные	Ожидаемый результат	Полученный результат
1	123456 123456 789123 789123 456789 456789	123456 654321 789123 321987 456789 987654	123456 654321 789123 321987 456789 987654
2	arozau analap lapyaz aroaro arozau analap lapyaz aroaro	arozau palana lapyaz oraora arozau palana lapyaz oraora	arozau palana lapyaz oraora arozau palana lapyaz oraora
3	apolon messui messii ronald ronald popsoc popsoc apolon	apolon iussem messii dlanor ronald cospop popsoc nolopa	apolon iussem messii dlanor ronald cospop popsoc nolopa

Контрольные вопросы

1. Дайте определение символьной строки.

Под строкой символов понимается последовательность байт, а цепочка — это более общее название для случаев, когда элементы последовательности имеют размер больше байта — слово или двойное слово.

2. Назовите основные команды обработки цепочек?

1) Пересылка цепочки:
   movs Адрес_приемника,Адрес_источника
2) сравнение цепочек:
   cmps Адрес_приемника,Адрес_источника
3) сканирование цепочки:
   scas Адрес_приемника
4) загрузка элемента из цепочки:
   lods Адрес_источника
5) сохранение элемента в цепочке:
   stos Адрес_приемника

3. Какие операции выполняют строковые команды MOVS? Какие особенности характерны для этих команд?

Команда копирует байт, слово или двойное слово из цепочки, адресуемой операндом Адрес_источника, в цепочку, адресуемую операндом Адрес_приемника.

Размер пересылаемых элементов ассемблер определяет, исходя из атрибутов идентификаторов, указывающих на области памяти приемника и источника. Для цепочечных команд с операндами, к которым относится и команда пересылки

movs Адрес_приемника,Адрес_источника, не существует машинного аналога.

При трансляции в зависимости от типа операндов транслятор преобразует ее в одну из трех машинных команд:

movsb, movsw или movsd.

4. Какие операции выполняют строковые команды CMPS, SCAS? Какие особенности характерны для этих команд?

Команды, реализующие операцию сканирования, выполняют поиск некоторого значения в области памяти. Логически эта область памяти рассматривается как последовательность (цепочка) элементов фиксированной длины размером 8, 16 или 32 бит.
Условие поиска для каждой из этих трех команд находится в строго определенном месте. Так:

- если цепочка описана с помощью директивы db (resb), то искомый элемент должен быть байтом и находиться в al, а сканирование цепочки осуществляется командой scasb;
- если цепочка описана с помощью директивы dw (resb), то это — слово в ax, и поиск ведется командой scasw;
- если цепочка описана с помощью директивы dd, то это — двойное слово в eax, и поиск ведется командой scasd.

Принцип поиска тот же, что и в команде сравнения cmps, то есть последовательное выполнение вычитания (содержимое регистра аккумулятора – содержимое очередного элемента цепочки)

Команды, реализующие операцию сравнения, выполняют сравнение элементов цепочки-источника с элементами цепочки-приемника.
Алгоритм выполнении работы команды cmps заключается вычитания (элемент цепочки-источника в последовательном – элемент цепочки-приемника) над очередными элементами обеих цепочек.
Принцип выполнения вычитания командой cmps аналогичен команде сравнения cmp. Она, так же, как и cmp, производит вычитание элементов, не записывая при этом результата, и устанавливает флаги ZF, SF и OF.

5. Как обеспечить циклическую обработку строк?

Префиксы повторения имеют свои мнемонические обозначения:

- rep
- repe или repz
- repne или repnz

Эти префиксы повторения указывают перед нужной цепочечной командой. Цепочечная команда без префикса выполняется один раз. Размещение префикса перед цепочечной командой заставляет ее выполняться в цикле.
Отличия приведенных префиксов в том, на каком основании принимается решение о циклическом выполнении цепочечной команды:

- по состоянию регистра ecx/cx

- по флагу нуля ZF.

1) Префикс rep заставляет данные команды выполняться, пока содержимое ecx/cx не станет равным 0. При этом цепочечная команда, перед которой стоит префикс, автоматически уменьшает содержимое ecx/cx на единицу.
2) Префиксы повторения repe или repz (REPeat while Equal or Zero). Эти префиксы являются абсолютными синонимами.
   Они заставляют цепочечную команду выполняться до тех пор, пока содержимое ecx/cx не равно нулю или флаг ZF равен 1
3) Префиксы повторения repne или repnz (REPeat while Not Equal or Zero). Эти префиксы также являются абсолютными синонимами.
   Префиксы repne/repnz заставляют цепочечную команду циклически выполняться до тех пор, пока содержимое ecx/cx не равно нулю или флаг ZF равен нулю.

6. Какова роль флага DF во флажковом регистре при выполнении команд обработки строк?

Следует отметить, что цепочечные команды сами выполняют модификацию регистров, адресующих операнды, обеспечивая тем самым автоматическое продвижение по цепочке. Количество байтов, на которые эта модификация осуществляется, определяется кодом команды. А вот знак этой модификации определяется значением флага направления DF (Direction Flag) в регистре eflags/flags:

- если DF = 0, то значение индексных регистров esi/si и edi/di будет автоматически увеличиваться (операция инкремента) цепочечными командами, то есть обработка будет осуществляться в направлении возрастания адресов;
- если DF = 1, то значение индексных регистров esi/si и edi/di будет автоматически уменьшаться (операция декремента) цепочечными командами, то есть обработка будет идти в направлении убывания адресов.

Состоянием флага DF можно управлять с помощью двух команд, не имеющих операндов:

- cld (Clear Direction Flag) — очистить флаг направления. Команда сбрасывает флаг направления DF в 0.
- std (Set Direction Flag) — установить флаг направления. Команда устанавливает флаг направления DF в 1.

7. Как правильно выбрать тестовые данные для проверки алгоритма обработки строки?

Для проверки алгоритма обработки строки необходимо проверить корректность работы всех частей программы, поэтому тестовые данные должны содержать такие исходные данные, чтобы все части программы были протестированы.

Вывод: Были изучены команды обработки цепочек и приемов обработки символьной информации. Программа работает корректно на заданных тестовых данных.