

Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

КУРСОВАЯ РАБОТА

Программирование на языке Ассемблера
по дисциплине: «Архитектура компьютера»

Выполнила
студентка гр. в5130904/30022

Г.М.Феллер

Руководитель
проф, д.т.н.

С.А.Молодяков

«_____» _____ 2024 г.

Содержание

Введение	3
Блок-схема	4
Список использованных прерываний BIOS	6
Листинг кода	7
Пример выполнения программы	12

Введение

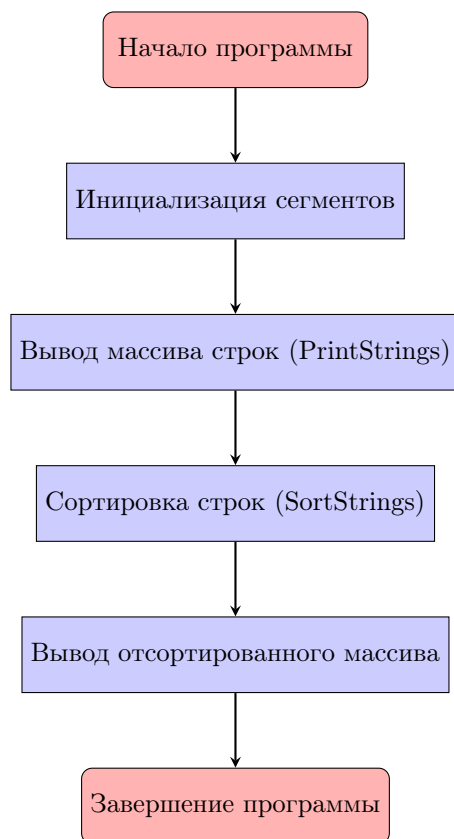
Язык программирования ассемблер относится к категории низкоуровневых и предназначен для работы с машинным кодом. Его структура и функции определяются архитектурой процессора и особенностями вычислительной системы. Ассемблер дает возможность взаимодействовать с аппаратным обеспечением компьютера на прямом уровне. Программы, написанные на ассемблере, состоят из инструкций, которые в процессе трансляции преобразуются в машинные коды.

Задание

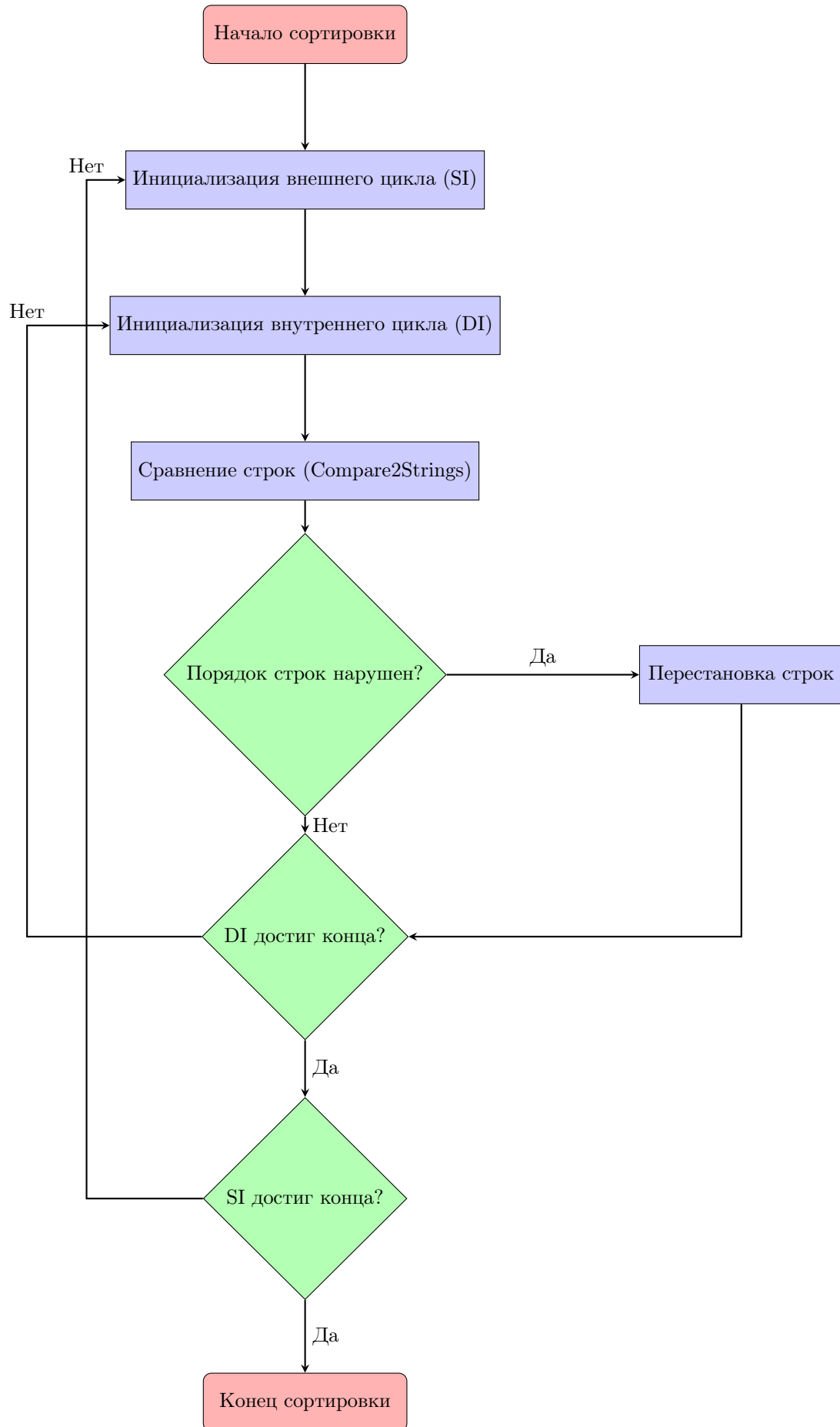
Дан список из 20 слов по 10 символов в каждом. Напечатать его в обратном алфавитном порядке, предварительно удалив из него повторяющиеся слова. При сортировке игнорировать высоту букв (Например, A = a).

Блок-схема

Структура программы



Алгоритм сортировки



Список использованных прерываний BIOS

INT 21h – Main DOS API

AH	Description
09h	Display string
20h	Terminate the current program
4C00h	Terminate with return code 0

Листинг кода

```
1 Name words
2 ; Сегмент данных: строки, указатели и вспомогательные данные
3 data segment
4 db "Scepticism",9,"$"
5 db "acceptance",9,"$"
6 db "yellowwood",9,"$"
7 db "Conversion",9,"$"
8 db "management",9,"$"
9 db "Corruption",9,"$"
10 db "depression",9,"$"
11 db "collection",9,"$"
12 db "investment",9,"$"
13 db "assembling",9,"$"
14 db "Management",9,"$"
15 db "conversion",9,"$"
16 db "obligation",9,"$"
17 db "prospectus",9,"$"
18 db "Collection",9,"$"
19 db "securities",9,"$"
20 db "Investment",9,"$"
21 db "typewriter",9,"$"
22 db "Validation",9,"$"
23 db "xenogenies",9,"$"
24
25 ; Смещения начала каждой строки
26 stringarr dw
    ↪ 0,12,24,36,48,60,72,84,96,108,120,132,144,156,168,180,192,204,216,228
    ↪
27 ; Количество строк в массиве
28 stringnum dw 20
29 ; Вывод заголовков
30 array db 'Array:',13,10,'$'
31 result db 13,10,'Result:',13,10,'$'
32 data ends
33
34 ; Сегмент стека: используется для временного хранения данных
35 stacks segment word stack 'stack'
36 ; Выделение 200 слов для стека
37 dw 200 dup(?)
38 ; Указатель на вершину стека
39 StackTop:
40 stacks ends
41
42 ; Сегмент кода программы
43 code segment
44 ; Указываем, что CS, DS, ES, SS указывают на свои сегменты
45 assume CS: code, DS: data, ES: data, SS: stacks
46
47 ; Процедура PrintStrings: выводит на экран строки из массива
48 PrintStrings proc
49     push bp                ; Сохраняем базовый указатель
50     mov bp,sp              ; Устанавливаем стек-кадр
51     push si                ; Сохраняем регистр SI
52     push di                ; Сохраняем регистр DI
53     push dx                ; Сохраняем регистр DX
54     push cx                ; Сохраняем регистр CX
55
56     mov si,[bp+4]          ; Указатель на массив строк (stringarr)
```

```

57     mov di,[bp+6]      ; Указатель на количество строк (stringnum)
58     mov cx,[di]        ; Количество строк
59
60     print:
61     mov dx,[si]         ; Адрес текущей строки
62     mov ah,09h          ; Функция DOS (09h - вывод строки)
63     int 21h             ; Вызов DOS для вывода строки
64     add si,2            ; Переход к следующему указателю в stringarr
65     loop print          ; Уменьшаем CX и повторяем цикл, если CX > 0
66
67     pop cx              ; Восстанавливаем регистр CX
68     pop dx              ; Восстанавливаем регистр DX
69     pop di              ; Восстанавливаем регистр DI
70     pop si              ; Восстанавливаем регистр SI
71     pop bp              ; Восстанавливаем базовый указатель
72     ret 4               ; Выход из процедуры
73 PrintStrings endp
74
75 ; Процедура Compare2Strings: сравнивает две строки
76 Compare2Strings proc
77     push bp              ; Сохраняем базовый указатель
78     mov bp,sp            ; Устанавливаем стек-кадр
79     push si              ; Сохраняем регистр SI
80     push di              ; Сохраняем регистр DI
81     push cx              ; Сохраняем регистр CX
82     push dx              ; Сохраняем регистр DX
83
84     mov si,[bp+4]        ; SI = указатель на первую строку
85     mov di,[bp+6]        ; DI = указатель на вторую строку
86     mov cx,10            ; CX = количество символов сравнения
87
88     startcom:
89     lodsb                ; Загружаем текущий символ первой строки в AL
90     cmp al,'a'           ; Проверяем, меньше ли символ 'a'
91     jb readsymstr2       ; Если меньше, пропускаем преобразование
92     cmp al,'z'           ; Проверяем, больше ли символ 'z'
93     ja readsymstr2       ; Если больше, пропускаем преобразование
94     sub al,20h           ; Преобразуем символ в верхний регистр
95
96     readsymstr2:
97     mov ah,[di]          ; Загружаем символ второй строки в AH
98     inc di               ; Увеличиваем DI для перехода к следующему символу
99     cmp ah,'a'           ; Проверяем, меньше ли символ 'a'
100    jb comparesym        ; Если меньше, пропускаем преобразование
101    cmp ah,'z'           ; Проверяем, больше ли символ 'z'
102    ja comparesym        ; Если больше, пропускаем преобразование
103    sub ah,20h           ; Преобразуем символ в верхний регистр
104
105    comparesym:
106    cmp al,ah             ; Сравниваем символы
107    jz nextsym            ; Если равны, переходим к следующему
108    ja above              ; Если первый символ больше, переходим к "above"
109    mov al,1              ; Устанавливаем результат = 1 (меньше)
110    jmp exit
111
112    above:
113    mov al,2              ; Устанавливаем результат = 2 (больше)
114    jmp exit
115
116    nextsym:
117    loop startcom         ; Повторяем цикл для всех символов

```



```

118     mov al,0           ; Если строки равны, результат = 0
119
120     exit:
121     pop dx             ; Восстанавливаем регистр DX
122     pop cx             ; Восстанавливаем регистр CX
123     pop di             ; Восстанавливаем регистр DI
124     pop si             ; Восстанавливаем регистр SI
125     pop bp             ; Восстанавливаем базовый указатель
126     ret 4              ; Выход из процедуры
127 Compare2Strings endp
128
129 ; Процедура RemoveDuplicate: удаляет дублирующую строку
130 RemoveDuplicate proc
131     push bp            ; Сохраняем базовый указатель
132     mov bp,sp          ; Устанавливаем стек-кадр
133     push si            ; Сохраняем регистр SI
134     push di            ; Сохраняем регистр DI
135
136     mov di,[bp+8]      ; DI = указатель на stringnum
137     mov ax,[di]        ; AX = количество строк
138     dec ax             ; Уменьшаем количество строк
139     mov [di],ax        ; Обновляем stringnum
140
141     mov si,[bp+6]      ; SI = конец массива строк
142     mov di,[bp+4]      ; DI = текущая строка
143
144     cmp si,di          ; Проверка, достигнут ли конец массива
145     jz exitdup         ; Если достигнут, выходим
146
147     moving:
148     mov ax,[di+2]      ; Копируем следующую строку в текущую
149     mov [di],ax
150     add di,2           ; Переходим к следующей строке
151     cmp di,si          ; Проверяем, достигнут ли конец массива
152     jb moving          ; Если нет, продолжаем
153
154     exitdup:
155     pop di             ; Восстанавливаем регистр DI
156     pop si             ; Восстанавливаем регистр SI
157     pop bp             ; Восстанавливаем базовый указатель
158     ret 6              ; Выход из процедуры
159 RemoveDuplicate endp
160
161 ; Процедура SortStrings: сортирует строки в массиве stringarr в алфавитном
    ↪ порядке
162 SortStrings proc
163     push bp            ; Сохраняем базовый указатель
164     mov bp,sp          ; Устанавливаем стек-кадр
165     push si            ; Сохраняем регистр SI
166     push di            ; Сохраняем регистр DI
167     push cx            ; Сохраняем регистр CX
168     push bx            ; Сохраняем регистр BX
169     push dx            ; Сохраняем регистр DX
170
171     mov si,[bp+4]      ; SI = указатель на начало массива строк (stringarr)
172     mov di,[bp+6]      ; DI = указатель на количество строк (stringnum)
173     mov ax,[di]        ; AX = количество строк
174     dec ax             ; AX = количество строк - 1 (для границ сортировки)
175     shl ax,1           ; Умножаем AX на 2 (каждый указатель занимает 2 байта)
176     mov dx,si          ; DX = начало массива строк
177     add dx,ax           ; DX = конец массива строк

```

```

178
179      sort:
180      mov di,si          ; DI = указатель на текущую строку (начало текущего пр
      ↪ охода)
181      add di,2          ; DI = указатель на следующую строку
182
183      sort2:
184      push [di]         ; Сохраняем указатель на строку 2
185      push [si]         ; Сохраняем указатель на строку 1
186
187      ; Вызов процедуры сравнения строк
188      call Compare2Strings
189
190      cmp al,0          ; Проверяем результат сравнения
191      jne check1        ; Если строки не равны, переходим к следующей проверке
192
193      ; Удаление дублирующейся строки
194      push [bp+6]       ; Передаем указатель на stringnum
195      push dx           ; Передаем конец массива строк
196      push di           ; Передаем указатель на дублирующуюся строку
197
198      ; Вызов процедуры удаления дублирующейся строки
199      call RemoveDuplicate
200
201      sub dx,2          ; Сдвигаем конец массива строк
202      cmp di,dx         ; Проверяем, достиг ли DI конца массива
203      jbe sort2        ; Если нет, продолжаем сортировку
204
205      jmp exitremove    ; Переходим к завершению процедуры
206
207      check1:
208      cmp al,1          ; Проверяем, первая строка меньше второй?
209      jne next          ; Если нет, переходим к следующей строке
210
211      ; Обмен строк местами
212      mov bx,[di]       ; BX = указатель на строку 2
213      mov cx,[si]       ; CX = указатель на строку 1
214      mov [di],cx       ; Меняем местами указатели строк
215      mov [si],bx       ; Завершаем обмен
216
217      next:
218      add di,2          ; DI = следующая строка
219      cmp di,dx         ; Проверка, достиг ли DI конца массива
220      jbe sort2        ; Если нет, продолжаем сортировку текущего прохода
221
222      add si,2          ; SI = следующая строка для нового прохода
223      cmp si,dx         ; Проверка, достиг ли SI конца массива
224      jb sort          ; Если нет, продолжаем сортировку
225
226      exitremove:
227      pop dx            ; Восстанавливаем регистр DX
228      pop bx            ; Восстанавливаем регистр BX
229      pop cx            ; Восстанавливаем регистр CX
230      pop di            ; Восстанавливаем регистр DI
231      pop si            ; Восстанавливаем регистр SI
232      pop bp            ; Восстанавливаем базовый указатель
233      ret 4             ; Выход из процедуры
234      SortStrings endp
235
236      ; Начало выполнения программы
237      start:

```

```

238     mov sp,offset StackTop ; Инициализация указателя стека (SP) на вершин
    ↪ у стека
239
240     ; Инициализация сегментов данных
241     mov ax,data
242     mov ds,ax
243     mov es,ax
244
245     ; Вывод заголовка "Array:"
246     mov dx,offset array      ; DX = адрес строки
247     mov ah,09h              ; Функция DOS (09h - вывод строки)
248     int 21h                 ; Вызов DOS для вывода строки
249
250     ; Вывод массива строк перед сортировкой
251     lea ax,stringnum         ; AX = адрес количества строк
252     push ax                  ; Передаем адрес количества строк в стек
253     lea ax,stringarr        ; AX = адрес массива строк
254     push ax                  ; Передаем адрес массива строк в стек
255     call PrintStrings        ; Вызов процедуры PrintStrings
256
257     ; Вывод заголовка "Result:"
258     mov dx,offset result     ; DX = адрес строки
259     mov ah,09h              ; Функция DOS (09h - вывод строки)
260     int 21h                 ; Вызов DOS для вывода строки
261
262     ; Сортировка массива строк
263     lea ax,stringnum         ; AX = адрес количества строк
264     push ax                  ; Передаем адрес количества строк в стек
265     lea ax,stringarr        ; AX = адрес массива строк
266     push ax                  ; Передаем адрес массива строк в стек
267     call SortStrings        ; Вызов процедуры SortStrings
268
269     ; Вывод массива строк после сортировки
270     lea ax,stringnum         ; AX = адрес количества строк
271     push ax                  ; Передаем адрес количества строк в стек
272     lea ax,stringarr        ; AX = адрес массива строк
273     push ax                  ; Передаем адрес массива в стек
274     call PrintStrings        ; Вызываем процедуру PrintStrings
275
276     ; Завершение программы
277     mov ax,4c00h             ; Функция DOS (4Ch - завершение программы)
278     int 21h                 ; Вызов DOS для завершения программы
279
280     code ends
281     end start

```

Пример выполнения программы

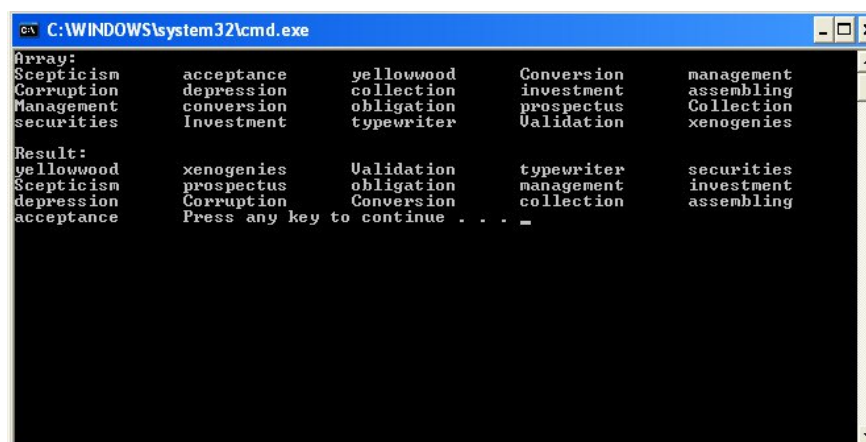


```
Окно компиляции
C:\ASMT00L>TASM.EXE temp.asm...
Turbo Assembler Version 4.1 Copyright (c) 1988, 1996 Borland International

Assembling file: temp.asm
Error messages: None
Warning messages: None
Passes: 1
Remaining memory: 443k

C:\ASMT00L>TLINK.EXE temp.OBJ
Turbo Link Version 7.1.30.1 Copyright (c) 1987, 1996 Borland International
Press any key to continue . . .
```

Рис. 1: Окно компиляции программы



```
C:\WINDOWS\system32\cmd.exe
Array:
Scepticism      acceptance      yellowwood     Conversion      management
Corruption      depression     collection      investment      assembling
Management      conversion     obligation      prospectus     Collection
Securities      Investment     typewriter     Validation      xenogenies

Result:
yellowwood      xenogenies     Validation      typewriter      securities
Scepticism      prospectus     obligation      management      investment
Depression      Corruption     Conversion      collection      assembling
acceptance      Press any key to continue . . .
```

Рис. 2: Окно выполнения программы