Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №3
по дисциплине: **КСиС**
Тема: ОБРАБОТКА СИМВОЛЬНЫХ ДАННЫХ

Выполнил
студент 2 курса
Корнасевич И. Д.

Проверил
Савицкий Ю. В.

Брест 2021

Задание:

Первое правило преобразования: Количество строчных латинских букв в тексте равно сумме цифр в нем.

Первое правило преобразования: Заменить каждую ненулевую цифру соответствующей ей строчной буквой латинского алфавита (1 → a, 2 → b и т.д.).

Второе правило преобразования: Удалить символы в каждой второй позиции.

string.asm

```
1  STDIN equ 0
2  STDOUT equ 1
3  STDERR equ 2
4
5  SYS_READ equ 0
6  SYS_WRITE equ 1
7  SYS_EXIT equ 60
8
9  NEW_LINE equ 10
10 ZERO equ 0
11 ASCII_ZERO equ '0'
12 ASCII_NINE equ '9'
13
14 ASCII_A equ 'a'
15 ASCII_Z equ 'z'
16
17 LINE_SIZE equ 128
18 DIGIT_SIZE equ 128
19 POS_SIZE equ 8
20
21 section .data
22   prompt db "Enter a line: ", ZERO
23   lineMarker db "You have entered: ", ZERO
24   newLine db NEW_LINE, ZERO
25   digitsSumMarker db "Sum of digits in this line: ", ZERO
26   charactersCountMarker db "Number of characters in this line: ",
      ZERO
27   positiveMarker db "Running positive condition process...", ZERO
28   negativeMarker db "Running negative condition process...", ZERO
29   resultMarker db "Result: ", ZERO
30
31 section .bss
32   line resb LINE_SIZE
33   digit resb DIGIT_SIZE
34   digitPos resb POS_SIZE
35
36 section .text
37   global _start
38
39 %macro exit 0
40   mov rax, SYS_EXIT
```

```
41    mov rdi, ZERO
42    syscall
43  %endmacro
44
45  _start:
46    mov rax, prompt
47    call _print
48
49    call _getLine
50
51    mov rax, lineMarker
52    call _print
53
54    mov rax, line
55    call _print
56
57    mov rax, newLine
58    call _print
59
60    mov rax, digitsSumMarker
61    call _print
62
63    mov rax, line
64    call _digitsSumCount
65    mov r8, rax
66    call _printRAX
67
68    mov rax, charactersCountMarker
69    call _print
70
71    mov rax, line
72    call _charactersCount
73
74    cmp rax, r8
75    je _positiveCondition
76
77    jmp _negativeCondition
78
79  _positiveCondition:
80    call _printRAX
81    mov rax, positiveMarker
82    call _print
83
84    mov rax, line
85    call _positiveConditionProcess
86    jmp _end
87
88  _negativeCondition:
89    call _printRAX
90    mov rax, negativeMarker
91    call _print
92
93    mov rax, line
94    call _negativeConditionProcess
95    jmp _end
96
```

```
97  _end:
98    mov rax, newLine
99    call _print
100
101   mov rax, newLine
102   call _print
103
104   mov rax, resultMarker
105   call _print
106
107   mov rax, line
108   call _print
109
110   mov rax, newLine
111   call _print
112
113   exit
114
115 _print:
116   push rax
117   mov rbx, ZERO
118
119 _printLoop:
120   mov cl, [rax]
121
122   cmp cl, ZERO
123   je _printExit
124
125   inc rax
126   inc rbx
127   jmp _printLoop
128
129 _printExit:
130   mov rax, SYS_WRITE
131   mov rdi, STDOUT
132   pop rsi
133   mov rdx, rbx
134   syscall
135   ret
136
137 _getLine:
138   mov rax, SYS_READ
139   mov rdi, STDIN
140   mov rsi, line
141   mov rdx, LINE_SIZE
142   syscall
143   ret
144
145 _digitsSumCount:
146   mov rbx, rax
147   mov rax, ZERO
148
149 _digitsSumCountLoop:
150   mov rcx, ZERO
151   mov cl, [rbx]
152
```

```
153   cmp cl, ZERO
154   je _digitsSumCountExit
155
156   cmp cl, NEW_LINE
157   je _digitsSumCountExit
158
159   inc rbx
160
161   cmp cl, ASCII_ZERO
162   jl _digitsSumCountLoop
163
164   cmp cl, ASCII_NINE
165   jg _digitsSumCountLoop
166
167   add rax, rcx
168   sub rax, ASCII_ZERO
169
170   jmp _digitsSumCountLoop
171
172 _digitsSumCountExit:
173   ret
174
175 _printRAX:
176   mov rcx, digit
177   mov rbx, NEW_LINE
178   mov [rcx], rbx
179   inc rcx
180   mov [digitPos], rcx
181
182 _printRAXMakingLoop:
183   mov rdx, ZERO
184   mov rbx, 10
185   div rbx
186   push rax
187   add rdx, ASCII_ZERO
188
189   mov rcx, [digitPos]
190   mov [rcx], dl
191   inc rcx
192   mov [digitPos], rcx
193
194   pop rax
195   cmp rax, ZERO
196   jne _printRAXMakingLoop
197
198 _printRAXPrintingLoop:
199   mov rcx, [digitPos]
200
201   mov rax, SYS_WRITE
202   mov rdi, STDOUT
203   mov rsi, rcx
204   mov rdx, 1
205   syscall
206
207   mov rcx, [digitPos]
208   dec rcx
```

```
209   mov [digitPos], rcx
210
211   cmp rcx, digit
212   jge _printRAXPrintingLoop
213
214   ret
215
216 _charactersCount:
217   mov rbx, rax
218   mov rax, ZERO
219
220 _charactersCountLoop:
221   mov rcx, ZERO
222   mov cl, [rbx]
223
224   cmp cl, ZERO
225   je _charactersCountExit
226
227   cmp cl, NEW_LINE
228   je _charactersCountExit
229
230   inc rbx
231
232   cmp cl, ASCII_A
233   jl _charactersCountLoop
234
235   cmp cl, ASCII_Z
236   jg _charactersCountLoop
237
238   jmp _characterCountRAXIncrement
239
240 _characterCountRAXIncrement:
241   inc rax
242   jmp _charactersCountLoop
243
244 _charactersCountExit:
245   ret
246
247 _positiveConditionProcess:
248   mov rbx, ZERO
249   mov bl, [rax]
250
251   cmp bl, ZERO
252   je _positiveExit
253
254   cmp bl, NEW_LINE
255   je _positiveExit
256
257   cmp bl, ASCII_ZERO
258   jle _positiveRAXIncrement
259
260   cmp bl, ASCII_NINE
261   jg _positiveRAXIncrement
262
263   sub bl, 1
264   sub bl, ASCII_ZERO
```

```
265   add bl, ASCII_A
266   mov [rax], bl
267
268   jmp _positiveRAXIncrement
269
270 _positiveRAXIncrement:
271   inc rax
272   jmp _positiveConditionProcess
273
274 _positiveExit:
275   ret
276
277 _negativeConditionProcess:
278   mov rbx, ZERO
279   mov bl, [rax]
280
281   cmp bl, ZERO
282   je _negativeExit
283
284   cmp bl, NEW_LINE
285   je _negativeExit
286
287   inc rax
288   mov bl, [rax]
289
290   cmp bl, ZERO
291   je _negativeExit
292
293   cmp bl, NEW_LINE
294   je _negativeExit
295
296   push rax
297   call _moveStringToTheLeft
298   pop rax
299
300   jmp _negativeConditionProcess
301
302 _negativeExit:
303   ret
304
305 _moveStringToTheLeft:
306   mov rbx, ZERO
307   push rax
308
309   inc rax
310   mov bl, [rax]
311
312   pop rax
313   mov [rax], bl
314
315   inc rax
316
317   cmp bl, ZERO
318   je _moveStringToTheLeftExit
319
320   cmp bl, NEW_LINE
```

```
321   je _moveStringToTheLeftExit
322
323   jmp _moveStringToTheLeft
324
325 _moveStringToTheLeftExit:
326   mov bl, ZERO
327   mov [rax], rbx
328   ret
```

results.txt

```
1  Enter a line: Hello, world 123.
2  You have entered: Hello, world 123.
3
4  Sum of digits in this line: 6
5  Number of characters in this line: 9
6  Running negative condition process...
7
8  Result: Hlo ol 2.
9
10
11 Enter a line: hhh3.
12 You have entered: hhh3.
13
14 Sum of digits in this line: 3
15 Number of characters in this line: 3
16 Running positive condition process...
17
18 Result: hhhc.
```

Вывод: Я написал программу для обработки текста на ассемблере NASM. Были использованы команды