Национальный исследовательский университет
«Высшая школа экономики»

# ОТЧЕТ ПО ЗАДАНИЮ №2

## «Ассемблерная программа обработки строк символов»

**Вариант 7**

Выполнил:
студент 213 группы
Фам Ч. Х.

Преподаватель:
Виденин С. А.

Москва
2022

## 1. Условие задачи

Разработать программу, заменяющую все гласные буквы в задан- ной ASCII-строке их ASCII кодами в шестнадцатиричной системе счисления. Код каждого символа задавать в формате «0xDD», где D — шестнадцатиричная цифра от 0 до F.

## 2. Код на языке Си

```c
#include <stdio.h>
#include <stdbool.h>

bool compare(char a, char b) {
    return (a == b) ? true : false;
}

int main(void) {
    char a[50];
    fgets( Buf: a,  MaxCount: 50,  File: stdin);
    for (int i = 0; a[i] != '\0'; ++i) {
        if (compare( a: a[i],  b: 'e') || compare( a: a[i],  b: 'y') || compare( a: a[i],  b: 'u') || compare( a: a[i],  b: 'i') ||
            compare( a: a[i],  b: 'o') || compare( a: a[i],  b: 'a')) {
            printf( format: "0x%x", a[i]);
        } else {
            printf( format: "%c", a[i]);
        }
    }
    return 0;
}
```

## 3. Используемые опции

**-O0** убирает любую оптимизацию

**-Wall** задает режим предупреждения сомнительных конструкциях в исходном коде

**-masm=intel** устанавливает синтаксиса Intel

**-fno-asynchronous-unwind-tables** одна из опций откладки

**-fcf-protection=none** убирает технологию CET из кода программы

```
rel@rel-VirtualBox:~$ gcc -O0 -Wall -masm=intel -fno-asynchronous-unwind-tables
-fcf-protection=none string.c -o string -save-temps
```

## 4. Результаты тестов

```
rel@rel-VirtualBox:~$ ./string_non_opt
qwertyuiop
qw0x65rt0x790x750x690x6fp
rel@rel-VirtualBox:~$ ./string
qwertyuiop
qw0x65rt0x790x750x690x6fp
rel@rel-VirtualBox:~$ ./string_non_opt
ghsldfjsaldfbnlldfwdt
ghsldfjs0x61ldfbnlldfwdt
rel@rel-VirtualBox:~$ ./string
ghsldfjsaldfbnlldfwdt
ghsldfjs0x61ldfbnlldfwdt
rel@rel-VirtualBox:~$ ./string_non_opt
aaapooooheeeee
0x610x610x61p0x6f0x6f0x6f0x6fh0x650x650x650x650x65
rel@rel-VirtualBox:~$ ./string
aaapooooheeeee
0x610x610x61p0x6f0x6f0x6f0x6fh0x650x650x650x650x65
```

```
rel@rel-VirtualBox:~$ ./string
aaapooooheeeee
0x610x610x61p0x6f0x6f0x6f0x6fh0x650x650x650x650x65
rel@rel-VirtualBox:~$ ./string_non_opt
a
0x61
rel@rel-VirtualBox:~$ ./string
a
0x61
rel@rel-VirtualBox:~$ ./string_non_opt
eyuioa
0x650x790x750x690x6f0x61
rel@rel-VirtualBox:~$ ./string
eyuioa
0x650x790x750x690x6f0x61
```

Обе программы выдают одинаковый результат на одних и тех же входных данных.

## 5. Ассемблерный код с комментариями

```
	.file "string.c"
	.intel_syntax noprefix
	.text
	.globl	compare
	.type compare, @function
compare:
	push  rbp
	mov   rbp, rsp
	mov   edx, edi #a (1st parameter)
	mov   eax, esi #b (2nd parameter)
	mov   BYTE PTR -4[rbp], dl #get byte from 1st parameter
	mov   BYTE PTR -8[rbp], al #get byte from 2nd parameter
	movzx eax, BYTE PTR -4[rbp]
	cmp   al, BYTE PTR -8[rbp]
	sete  al  #if (a == b) al = true
	pop   rbp
	ret
	.size compare, .-compare
	.section   .rodata
.LC0:
	.string    "0x%x"
	.text
	.globl	main
	.type main, @function
main:
	push  rbp
	mov   rbp, rsp
	sub   rsp, 80
	mov   rax, QWORD PTR fs:40
	mov   QWORD PTR -8[rbp], rax
	xor   eax, eax
	mov   rdx, QWORD PTR stdin[rip]
	lea   rax, -64[rbp] #char a[50]
	mov   esi, 50
	mov   rdi, rax
	call  fgets@PLT
	mov   DWORD PTR -68[rbp], 0 #int i
	jmp   .L4
```

```
.L8:
      mov   eax, DWORD PTR -68[rbp]
      cdqe
      movzx eax, BYTE PTR -64[rbp+rax]
      movsx eax, al
      mov   esi, 101 #esi = 'e'
      mov   edi, eax #edi = 'a[i]'
      call  compare
      test  al, al
      jne   .L5
      mov   eax, DWORD PTR -68[rbp]
      cdqe
      movzx eax, BYTE PTR -64[rbp+rax]
      movsx eax, al
      mov   esi, 121 #esi = 'y'
      mov   edi, eax #edi = 'a[i]'
      call  compare
      test  al, al
      jne   .L5
      mov   eax, DWORD PTR -68[rbp]
      cdqe
      movzx eax, BYTE PTR -64[rbp+rax]
      movsx eax, al
      mov   esi, 117 #esi = 'y'
      mov   edi, eax #edi = 'a[i]'
      call  compare
      test  al, al
      jne   .L5
      mov   eax, DWORD PTR -68[rbp]
      cdqe
      movzx eax, BYTE PTR -64[rbp+rax]
      movsx eax, al
      mov   esi, 105 #esi = 'i'
      mov   edi, eax #edi = 'a[i]'
      call  compare
      test  al, al
      jne   .L5
      mov   eax, DWORD PTR -68[rbp]
      cdqe
      movzx eax, BYTE PTR -64[rbp+rax]
      movsx eax, al
      mov   esi, 111 #esi = 'o'
      mov   edi, eax #edi = 'a[i]'
      call  compare
      test  al, al
      jne   .L5
      mov   eax, DWORD PTR -68[rbp]
      cdqe
      movzx eax, BYTE PTR -64[rbp+rax]
      movsx eax, al
      mov   esi, 97  #esi = 'a'
      mov   edi, eax #edi = 'a[i]'
      call  compare
      test  al, al
      je    .L6
.L5:
```

```
    mov   eax, DWORD PTR -68[rbp]
    cdqe
    movzx eax, BYTE PTR -64[rbp+rax]
    movsx eax, al
    mov   esi, eax
    lea   rax, .LC0[rip]
    mov   rdi, rax
    mov   eax, 0
    call  printf@PLT #print vowel symbol
    jmp   .L7
.L6:
    mov   eax, DWORD PTR -68[rbp]
    cdqe
    movzx eax, BYTE PTR -64[rbp+rax]
    movsx eax, al
    mov   edi, eax
    call  putchar@PLT #print consonant symbol
.L7:
    add   DWORD PTR -68[rbp], 1 #++i
.L4:
    mov   eax, DWORD PTR -68[rbp]
    cdqe
    movzx eax, BYTE PTR -64[rbp+rax]
    test  al, al
    jne   .L8
    mov   eax, 0
    mov   rdx, QWORD PTR -8[rbp]
    sub   rdx, QWORD PTR fs:40
    je    .L10
    call  __stack_chk_fail@PLT
.L10:
    leave
    ret
    .size main, .-main
    .ident     "GCC: (Ubuntu 11.3.0-1ubuntu1~22.04) 11.3.0"
    .section   .note.GNU-stack,"",@progbits
```

**6. Дополнение на 5 баллов**

В реализованной программе используется функция **`compare`** с передачей параметров через параметры.

Используется локальная переменная **`int i`** в 11 строке кода на языке Си.

В ассемблерный код добавлены комментарии для формальных параметров, описывающие связь между параметрами языка Си и регистрами.

**7. Дополнение на 6 баллов**

Была проведена модификация ассемблерного кода, целью которой было уменьшить количество обращений к памяти за счет использования регистров.

Скриншоты изменений кода:

```
6  compare:
7      push   rbp
8      mov    rbp, rsp
9      mov    edx, edi #a (1st parameter)
10     mov    eax, esi #b (2nd parameter)
11     cmp al, dl #cmp a and b // dl = BYTE PTR -8[rbp] from prev program
12     sete   al #if (a == b) al = true
33     call   fgets@PLT
34     mov    ebx, 0 #int i = DWORD PTR -68[rbp] from previos program
35     jmp    .L4
36 .L8:
37     mov    eax, ebx
38     cdqe
39     movzx  eax, BYTE PTR -64[rbp+rax]
45     jne    .L5
46     mov    eax, ebx
47     cdqe
48     mov    esi, 121 #esi = 'y'
49     call   compare
50     test   al, al
51     jne    .L5
52     mov    eax, ebx
53     cdqe
54     mov    esi, 117 #esi = 'u'
55     call   compare
56     test   al, al
57     jne    .L5
58     mov    eax, ebx
59     cdqe
59     cdqe
60     mov    esi, 105 #esi = 'i'
61     call   compare
62     test   al, al
63     jne    .L5
64     mov    eax, ebx
65     cdqe
66     mov    esi, 111 #esi = 'o'
67     call   compare
68     test   al, al
69     jne    .L5
70     mov    eax, ebx
71     cdqe
72     mov    esi, 97 #esi = 'a'
73     call   compare
77     mov    eax, ebx
78     cdqe
88     mov    eax, ebx
89     cdqe
90     movzx  eax, BYTE PTR -64[rbp+rax]
91     movsx  eax, al
92     mov    edi, eax
93     call   putchar@PLT #print consonant symbol
94 .L7:
95     add    ebx, 1 #++i
96 .L4:
97     mov    eax, ebx
98     cdqe
```

```
6  compare:
7      push   rbp
8      mov    rbp, rsp
9      mov    edx, edi #a (1st parameter)
10     mov    eax, esi #b (2nd parameter)
11     mov    BYTE PTR -4[rbp], dl #get byte from 1st parameter
12     mov    BYTE PTR -8[rbp], al #get byte from 2nd parameter
13     movzx  eax, BYTE PTR -4[rbp]
14     cmp    al, BYTE PTR -8[rbp]
15     sete   al #if (a == b) al = true
36     call   fgets@PLT
37     mov    DWORD PTR -68[rbp], 0 #int i
38     jmp    .L4
39 .L8:
40     mov    eax, DWORD PTR -68[rbp]
41     cdqe
42     movzx  eax, BYTE PTR -64[rbp+rax]
48     jne    .L5
49     mov    eax, DWORD PTR -68[rbp]
50     cdqe
51     movzx  eax, BYTE PTR -64[rbp+rax]
52     movsx  eax, al
53     mov    esi, 121 #esi = 'y'
54     mov    edi, eax #edi = 'a[i]'
55     call   compare
56     test   al, al
57     jne    .L5
58     mov    eax, DWORD PTR -68[rbp]
59     cdqe
60     movzx  eax, BYTE PTR -64[rbp+rax]
61     movsx  eax, al
62     mov    esi, 117 #esi = 'y'
63     mov    edi, eax #edi = 'a[i]'
64     call   compare
65     test   al, al
66     jne    .L5
67     mov    eax, DWORD PTR -68[rbp]
68     cdqe
68     cdqe
69     movzx  eax, BYTE PTR -64[rbp+rax]
70     movsx  eax, al
71     mov    esi, 105 #esi = 'i'
72     mov    edi, eax #edi = 'a[i]'
73     call   compare
74     test   al, al
75     jne    .L5
76     mov    eax, DWORD PTR -68[rbp]
77     cdqe
78     movzx  eax, BYTE PTR -64[rbp+rax]
79     movsx  eax, al
80     mov    esi, 111 #esi = 'o'
81     mov    edi, eax #edi = 'a[i]'
82     call   compare
83     test   al, al
84     jne    .L5
85     mov    eax, DWORD PTR -68[rbp]
86     cdqe
87     movzx  eax, BYTE PTR -64[rbp+rax]
88     movsx  eax, al
89     mov    esi, 97 #esi = 'a'
90     mov    edi, eax #edi = 'a[i]'
91     call   compare
95     mov    eax, DWORD PTR -68[rbp]
96     cdqe
106    mov    eax, DWORD PTR -68[rbp]
107    cdqe
108    movzx  eax, BYTE PTR -64[rbp+rax]
109    movsx  eax, al
110    mov    edi, eax
111    call   putchar@PLT #print consonant symbol
112 .L7:
113    add    DWORD PTR -68[rbp], 1 #++i
114 .L4:
115    mov    eax, DWORD PTR -68[rbp]
116    cdqe
```

В функции **compare** были полностью убраны обращение к памяти и лишние действия. Итератор цикла **for int i** был заменен на регистр **ebx**. В строках [54; 91] были убраны лишние действия с памятью (перенос значения **a[i]** в регистр **eax**). Соответствующие комментарии были добавлены в ассемблерный код.

```
rel@rel-VirtualBox:~$ ./string
qwertyuiop
qw0x65rt0x790x750x690x6fp
rel@rel-VirtualBox:~$ ./string_reg
qwertyuiop
qw0x65rt0x790x750x690x6fp
rel@rel-VirtualBox:~$ ./string
ghsldfjsaldfbnlldfwdt
ghsldfjs0x61ldfbnlldfwdt
rel@rel-VirtualBox:~$ ./string_reg
ghsldfjsaldfbnlldfwdt
ghsldfjs0x61ldfbnlldfwdt
rel@rel-VirtualBox:~$ ./string
aaapooooheeeee
0x610x610x61p0x6f0x6f0x6f0x6fh0x650x650x650x650x65
rel@rel-VirtualBox:~$ ./string_reg
aaapooooheeeee
0x610x610x61p0x6f0x6f0x6f0x6fh0x650x650x650x650x65
rel@rel-VirtualBox:~$ ^C
rel@rel-VirtualBox:~$ ./string
asnkdsvfslkbnefuiuidsfkdsfnksjdndsioi
0x61snkdsvfslkbn0x65f0x750x690x750x69dsfkdsfnksjdnds0x690x6f0x69
rel@rel-VirtualBox:~$ ./string_reg
asnkdsvfslkbnefuiuidsfkdsfnksjdndsioi
0x61snkdsvfslkbn0x65f0x750x690x750x69dsfkdsfnksjdnds0x690x6f0x69
```

Результаты выполнения программы после рефакторинга ассемблерного кода не изменились.