Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности электронновычислительных систем (КИБЭВС)

ПРОГРАММИРОВАНИЕ НА ЯЗЫКЕ АССЕМБЛЕР

Отчёт по лабораторной работе №2
по дисциплине «Системное программирование»
Вариант 12

Студенты гр. 738-1
_____ С.А Литовкин
_____

Принял
Преподаватель каф. КИБЭВС
_____ Е. Ю. Калинин
__. __2022

Томск 2022

Введение

Целью работы является: ознакомление со структурой программы на языке Ассемблер, средствами создания программ на Ассемблере для ОС Linux.

Все программы, написанные во время выполнения работы, были загружены на github репозиторий по ссылке:

https://github.com/Trapka/Lab2

Вариант 12.

Задача: установить 0 в 5-ом бите всех байтов массива с четным индексом и 1 в 4-ом бите всех нечетных байтов массива. Определить сумму элементов полученного массива.

1 Ход работы

Был воспроизведён пример программы на GAS из презентации (рисунок 1.1). Была создана программа по индивидуальному заданию для GAS (рисунок 1.2) и C++ (рисунок 1.3).

```
abcd text
serg@ubuntu:~/lab2$ gcc -no-pie -nostdlib qwe.s -o qwe
qwe.s: Сообщения ассемблера:
qwe.s: Предупреждение: конец файла не в конце строки; вставлен символ новой строки
serg@ubuntu:~/lab2$ ./qwe
abcd text
```

Рисунок 1.1 – Результат работы программы из примера

```
serg@ubuntu:~/lab2$ gcc -m32 -fno-pie -no-pie zadan.s -o zadan -g
serg@ubuntu:~/lab2$ ./zadan
20 -> 4 - even number
134 -> 142 - odd number
6 -> 6 - even number
28 -> 28 - odd number
126 -> 110 - even number
16 -> 24 - odd number
31 -> 15 - even number
100 -> 108 - odd number
240 -> 224 - even number
66 -> 74 - odd number
20 -> 4 - even number
55 -> 63 - odd number
sum = 802
```

Рисунок 1.2 – Результат работы программы на GAS

```
serg@ubuntu:~/lab2$ g++ zadanie.cpp  -g
serg@ubuntu:~/lab2$ ./a.out
20->    4- even number
134->   142 - odd number
6->     6- even number
28->    28 - odd number
126->   110- even number
16->    24 - odd number
31->    15- even number
100->   108 - odd number
240->   224- even number
66->    74 - odd number
20->    4- even number
55->    63 - odd number
sum = 802
```

Рисунок 1.3 – Результат работы программы на C++

Был запущен отладчик GDB (рисунок 1.4), были выполнены базовые команды отладчика и просмотрено состояние регистров в определённый момент

выполнения программы (рисунок 1.5). Была просмотрена информация о размерах программы на C++ (рисунок 1.6) и размерах программы на GAS (рисунок 1.7).

```
serg@ubuntu:~/lab2$ gdb test1
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from test1...
```

Рисунок 1.4 – Отладчик GDB

```
Breakpoint 1 at 0x8049176: file test1.s, line 17.
(gdb) run
Starting program: /home/serg/lab2/test1

Breakpoint 1, main () at test1.s:17
17      test1.s: Нет такого файла или каталога.
(gdb) i reg
eax            0xf7fb5088          -134524792
ecx            0xacc59081          -1396338559
edx            0xffffcf34          -12492
ebx            0x0                 0
esp            0xffffcf0c          0xffffcf0c
ebp            0x0                 0x0
esi            0xf7fb3000          -134533120
edi            0xf7fb3000          -134533120
eip            0x8049176           0x8049176 <main>
eflags         0x246               [ PF ZF IF ]
cs             0x23                35
ss             0x2b                43
ds             0x2b                43
es             0x2b                43
```

Рисунок 1.5 – Информация о состоянии реестров

```
serg@ubuntu:~/lab2$ g++ zadanie.cpp -static && ls -s -h a.out
2,3M a.out
```

Рисунок 1.6 – Размер программы на C++

```
serg@ubuntu:~/lab2$ cc -m32 -fno-pie -no-pie zadan.s -o zadan -g && du -sb zadan
16212   zadan
```

Рисунок 1.7 – Размер программы на GAS

Программа на C++ была дизассемблирована (рисунок 1.8). Была просмотрена информация о времени выполнения программы на C++ (рисунок 1.9) и программы на GAS (рисунок 1.10). Как можно заметить время выполнения программ одинаковое.

```
0000000005ce710 <_ZTVSt8numpunctIcE>:
        ...
 5ce718:       e8 e3 5c 00 00          callq  5d4400 <_ZSt4cerr+0x60>
 5ce71d:       00 00                   add    %al,(%rax)
 5ce71f:       00 40 c2                add    %al,-0x3e(%rax)
 5ce722:       40 00 00                add    %al,(%rax)
 5ce725:       00 00                   add    %al,(%rax)
 5ce727:       00 b0 c2 40 00 00       add    %dh,0x40c2(%rax)
 5ce72d:       00 00                   add    %al,(%rax)
 5ce72f:       00 e0                   add    %ah,%al
 5ce731:       69 45 00 00 00 00 00    imul   $0x0,0x0(%rbp),%eax
 5ce738:       f0 69 45 00 00 00 00    lock imul $0x0,0x0(%rbp),%eax
 5ce73f:       00
 5ce740:       30 70 45                xor    %dh,0x45(%rax)
 5ce743:       00 00                   add    %al,(%rax)
 5ce745:       00 00                   add    %al,(%rax)
 5ce747:       00 80 70 45 00 00       add    %al,0x4570(%rax)
 5ce74d:       00 00                   add    %al,(%rax)
 5ce74f:       00 d0                   add    %dl,%al
 5ce751:       70 45                   jo     5ce798 <_ZTVSt15numpunct_bynameI
```

Рисунок 1.8 – Дизассемблированная программа

```
serg@ubuntu:~/lab2$ time ./a.out
20->    4- even number
134->   142 - odd number
6->     6- even number
28->    28 - odd number
126->   110- even number
16->    24 - odd number
31->    15- even number
100->   108 - odd number
240->   224- even number
66->    74 - odd number
20->    4- even number
55->    63 - odd number
sum = 802

real    0m0,002s
user    0m0,001s
sys     0m0,000s
```

Рисунок 1.9 – Время выполнения программы на C++

```
serg@ubuntu:~/lab2$ time ./zadan
20 -> 4 - even number
134 -> 142 - odd number
6 -> 6 - even number
28 -> 28 - odd number
126 -> 110 - even number
16 -> 24 - odd number
31 -> 15 - even number
100 -> 108 - odd number
240 -> 224 - even number
66 -> 74 - odd number
20 -> 4 - even number
55 -> 63 - odd number
sum = 802

real    0m0,002s
user    0m0,001s
sys     0m0,000s
```

Рисунок 1.10 – Время выполнения программы на GAS

Был создан Docker на Debian для работы и компилированием программ на C++ и GAS (рисунок 1.11).

```
root@1fab55682c64:/# g++ zadanie.cpp
root@1fab55682c64:/# ./a.out
20->    4- even number
134->   142 - odd number
6->     6- even number
28->    28 - odd number
126->   110- even number
16->    24 - odd number
31->    15- even number
100->   108 - odd number
240->   224- even number
66->    74 - odd number
20->    4- even number
55->    63 - odd number
sum = 802
root@1fab55682c64:/# gcc -m32 -fno-pie -no-pie zadan.s -o zadan -g
root@1fab55682c64:/# ./zadan
20 -> 4 - even number
134 -> 142 - odd number
6 -> 6 - even number
28 -> 28 - odd number
126 -> 110 - even number
16 -> 24 - odd number
```

Рисунок 1.11 – Docker с работой программы на GAS и программы на C++

Заключение

В ходе выполнения данной работы было выполнено ознакомление со структурой программ на языке Ассемблер. Рассмотрены средства создания программ на Ассемблере для ОС Linux.