Exercise 1

Alberto Martin Lionardi , 7001812

November 2023

1 Task 1

a)

```
0000120d <secret>:
    120d:       f3 0f 1e fb             endbr32
    1211:       55                      push   %ebp
    1212:       89 e5                   mov    %esp,%ebp
    1214:       8b 45 08                mov    0x8(%ebp),%eax
    1217:       83 e8 01                sub    $0x1,%eax
    121a:       01 c0                   add    %eax,%eax
    121c:       5d                      pop    %ebp
    121d:       c3                      ret
```

b) the value in eax is firstly 5, after some si it changes to 4 and the return is 8
c) the function is f(n) = (n-1)*2
d)

```
There are 36 section headers, starting at offset 0x4138:
Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .interp           PROGBITS        000001b4 0001b4 000013 00   A  0   0  1
  [ 2] .note.gnu.build-i NOTE            000001c8 0001c8 000024 00   A  0   0  4
  [ 3] .note.gnu.propert NOTE            000001ec 0001ec 00001c 00   A  0   0  4
  [ 4] .note.ABI-tag     NOTE            00000208 000208 000020 00   A  0   0  4
  [ 5] .gnu.hash         GNU_HASH        00000228 000228 000020 04   A  6   0  4
  [ 6] .dynsym           DYNSYM          00000248 000248 0000a0 10   A  7   1  4
  [ 7] .dynstr           STRTAB          000002e8 0002e8 0000a7 00   A  0   0  1
  [ 8] .gnu.version      VERSYM          00000390 000390 000014 02   A  6   0  2
  [ 9] .gnu.version_r    VERNEED         000003a4 0003a4 000030 00   A  7   1  4
  [10] .rel.dyn          REL             000003d4 0003d4 000060 08   A  6   0  4
  [11] .rel.plt          REL             00000434 000434 000020 08  AI  6  24  4
  [12] .init             PROGBITS        00001000 001000 000024 00  AX  0   0  4
  [13] .plt              PROGBITS        00001030 001030 000050 04  AX  0   0 16
  [14] .plt.got          PROGBITS        00001080 001080 000010 10  AX  0   0 16
```

```
  [15] .plt.sec          PROGBITS        00001090 001090 000040 10  AX  0   0 16
  [16] .text             PROGBITS        000010d0 0010d0 000239 00  AX  0   0 16
  [17] .fini             PROGBITS        0000130c 00130c 000018 00  AX  0   0  4
  [18] .rodata           PROGBITS        00002000 002000 00001c 00   A  0   0  4
  [19] .eh_frame_hdr     PROGBITS        0000201c 00201c 000054 00   A  0   0  4
  [20] .eh_frame         PROGBITS        00002070 002070 00013c 00   A  0   0  4
  [21] .init_array       INIT_ARRAY      00003ec8 002ec8 000004 04  WA  0   0  4
  [22] .fini_array       FINI_ARRAY      00003ecc 002ecc 000004 04  WA  0   0  4
  [23] .dynamic          DYNAMIC         00003ed0 002ed0 000100 08  WA  7   0  4
  [24] .got              PROGBITS        00003fd0 002fd0 000030 04  WA  0   0  4
  [25] .data             PROGBITS        00004000 003000 000008 00  WA  0   0  4
  [26] .bss              NOBITS          00004008 003008 000004 00  WA  0   0  1
  [27] .comment          PROGBITS        00000000 003008 00002b 01  MS  0   0  1
  [28] .debug_aranges    PROGBITS        00000000 003033 000020 00      0   0  1
  [29] .debug_info       PROGBITS        00000000 003053 000369 00      0   0  1
  [30] .debug_abbrev     PROGBITS        00000000 0033bc 00011f 00      0   0  1
  [31] .debug_line       PROGBITS        00000000 0034db 000107 00      0   0  1
  [32] .debug_str        PROGBITS        00000000 0035e2 0002b9 01  MS  0   0  1
  [33] .symtab           SYMTAB          00000000 00389c 0004d0 10     34  51  4
  [34] .strtab           STRTAB          00000000 003d6c 000271 00      0   0  1
  [35] .shstrtab         STRTAB          00000000 003fdd 000158 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings), I (info),
  L (link order), O (extra OS processing required), G (group), T (TLS),
  C (compressed), x (unknown), o (OS specific), E (exclude),
  p (processor specific)
```

e) `<A>` is stored in data section because this is a `<C>` code and usually global and static variables are stored in data section, `<B>` is the same as `<A>`, `<C>` is a function, therefore its stored in code section, `<D>` is a local variable and is stored in the stack.

2 Task 3

a) it is used to store system settings and configuration which then are used by processes and applications.
b) there are a lot of aspects that could make arr's differs. But since this is about Environment Variable, Environment Variable could influence the program's behavior and the memory layout. When debugging using GDB the environment variable could change and affecting the memory allocation. The environment variables are usually stored in before stack or at the first part of the stack. By using

3 Task 4

a) argc is argument count, it counts how many arguments are passed into the program. argv is "argument vector" and is an array of string. argc and argv is in the stack.
b) argv[0] contains the string representation of the program name
c) whitespace indicates that it is a different argument, quotation marks is used to enclosed the string as a single argument, and (are used to include the literal characters.