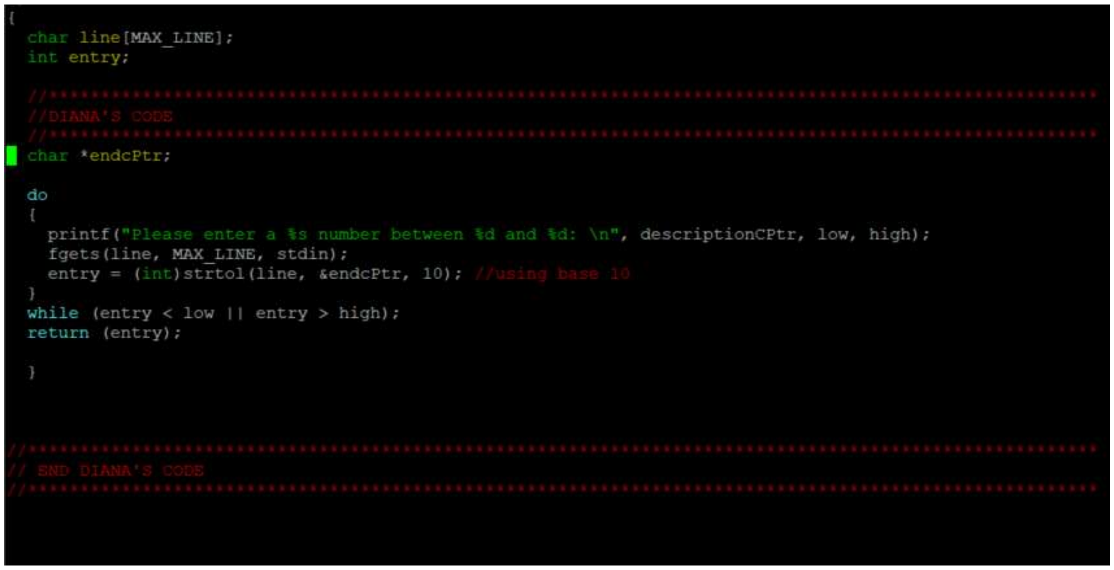
Diana Marczuk

CSC 407 Systems 2 – Homework 1

4/18/2021

2. My code in main.c:



```
{
  char line[MAX_LINE];
  int entry;

  //***************************************************************************************
  //DIANA'S CODE
  //***************************************************************************************
  char *endcPtr;

  do
  {
    printf("Please enter a %s number between %d and %d: \n", descriptionCPtr, low, high);
    fgets(line, MAX_LINE, stdin);
    entry = (int)strtol(line, &endcPtr, 10); //using base 10
  }
  while (entry < low || entry > high);
  return (entry);

  }



//****************************************************************************************
// END DIANA'S CODE
//****************************************************************************************
```

3. Answers

   a. 45.99 seconds

   b. 0.15 seconds

4. Answers:

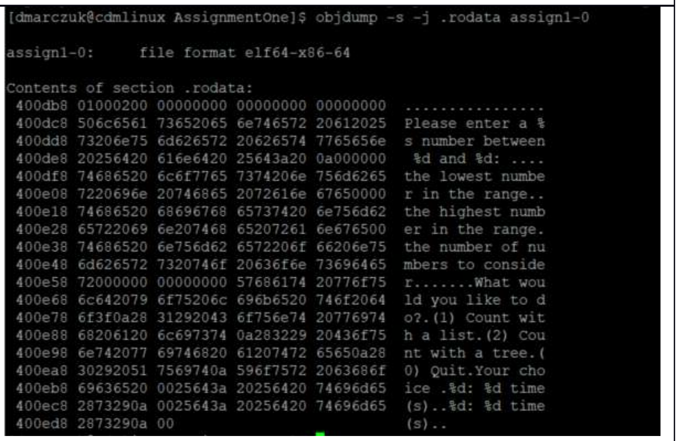
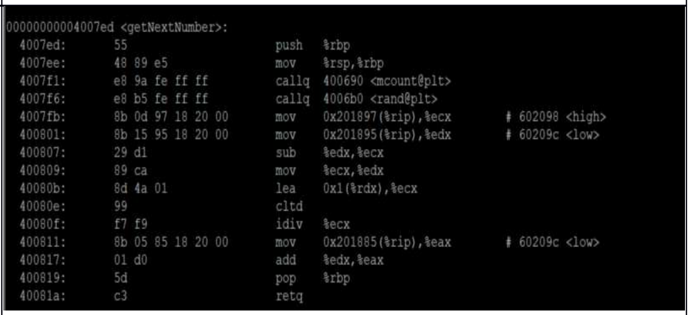
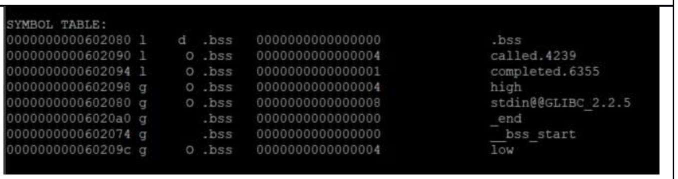
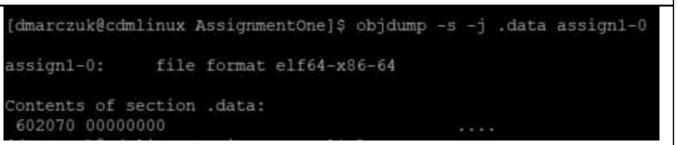
   a. 23.57 seconds

   b. 0.07 seconds

5. **Option -O0 with a good algorithm will be faster.** The bad algorithm that can be referenced in this questions is generating a list instead of a binary tree. Whether doing the compiler optimization or not, a list will be over 300 times slower than attempting to generate a binary tree. Compiler optimization is not enough to account for the bad algorithm (should you choose to generate a linear list), as the optimization only makes the bad algorithm (and the good algorithm for that matter), twice as fast as when the programmer uses no compiler optimization, which is not enough to beat the time to generate a tree with the same data provided.

   Should the algorithm be good, optimization can be added to make the process run even faster, however a programmer must watch out for unintentional changes the optimatization makes to an algorithm.

6.

Question	**Command**	**Result**
(A) The string "%d: %d time(s)\n" in printList( )	objdump -s -j .rodata assign1-0	
(B) The code for getNext Number()	Objdump -d -j .text assign1-0	
(C) The global variable hi gh	Objdump -t -j .bss assign1-0	
(D) treePtr in countWith Tree()	objdump -s -j .data assign1-0	 TreePtr is a variable that has been declared but not defined yet, which is why it is zero. The compiler does not know where to actually put the variable yet because the end user did not put in a number since the program did not run yet, there it will put in zero for the time being.

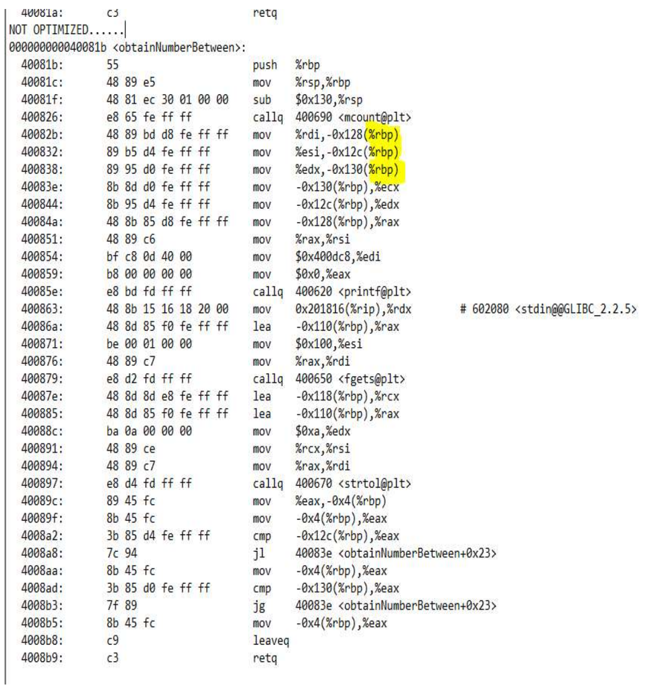
```
[dmarczuk@cdmlinux AssignmentOne]$ objdump -s -j .rodata assign1-0

assign1-0:     file format elf64-x86-64

Contents of section .rodata:
 400db8 01000200 00000000 00000000 00000000  ................
 400dc8 506c6561 73652065 6e746572 20612025  Please enter a %
 400dd8 73206e75 6d626572 20626574 7765656e  s number between
 400de8 20256420 616e6420 25643a20 0a000000   %d and %d: ....
 400df8 74686520 6c6f7765 7374206e 756d6265  the lowest numbe
 400e08 7220696e 20746865 2072616e 67650000  r in the range..
 400e18 74686520 68696768 65737420 6e756d62  the highest numb
 400e28 65722069 6e207468 65207261 6e676500  er in the range.
 400e38 74686520 6e756d62 6572206f 66206e75  the number of nu
 400e48 6d626572 7320746f 20636f6e 73696465  mbers to conside
 400e58 72000000 00000000 57686174 20776f75  r.......What wou
 400e68 6c642079 6f75206c 696b6520 746f2064  ld you like to d
 400e78 6f3f0a28 31292043 6f756e74 20776974  o?.(1) Count wit
 400e88 68206120 6c697374 0a283229 20436f75  h a list.(2) Cou
 400e98 6e742077 69746820 61207472 65650a28  nt with a tree.(
 400ea8 30292051 7569740a 596f7572 2063686f  0) Quit.Your cho
 400eb8 69636520 0025643a 20256420 74696d65  ice .%d: %d time
 400ec8 2873290a 0025643a 20256420 74696d65  (s)..%d: %d time
 400ed8 2873290a 00                           (s)..
```

```
00000000004007ed <getNextNumber>:
  4007ed:       55                      push   %rbp
  4007ee:       48 89 e5                mov    %rsp,%rbp
  4007f1:       e8 9a fe ff ff          callq  400690 <mcount@plt>
  4007f6:       e8 b5 fe ff ff          callq  4006b0 <rand@plt>
  4007fb:       8b 0d 97 18 20 00       mov    0x201897(%rip),%ecx        # 602098 <high>
  400801:       8b 15 95 18 20 00       mov    0x201895(%rip),%edx        # 60209c <low>
  400807:       29 d1                   sub    %edx,%ecx
  400809:       89 ca                   mov    %ecx,%edx
  40080b:       8d 4a 01                lea    0x1(%rdx),%ecx
  40080e:       99                      cltd
  40080f:       f7 f9                   idiv   %ecx
  400811:       8b 05 85 18 20 00       mov    0x201885(%rip),%eax        # 60209c <low>
  400817:       01 d0                   add    %edx,%eax
  400819:       5d                      pop    %rbp
  40081a:       c3                      retq
```

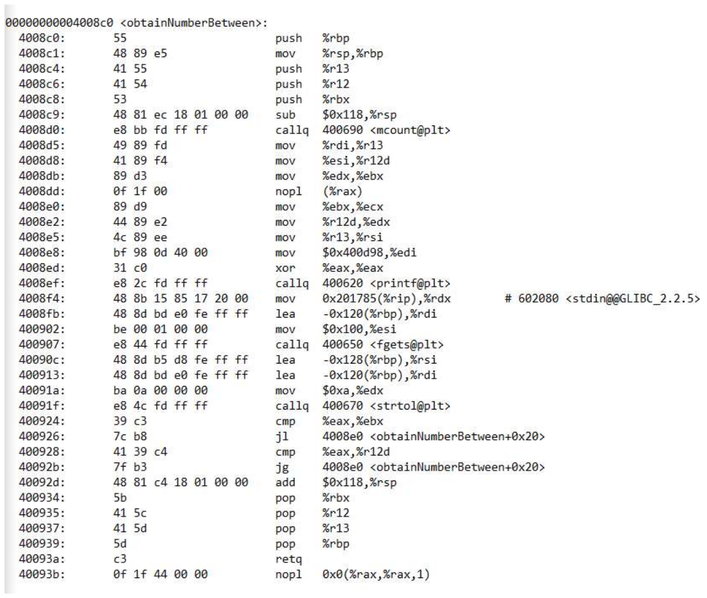
```
SYMBOL TABLE:
0000000000602080 l    d  .bss   0000000000000000              .bss
0000000000602090 l     O .bss   0000000000000004              called.4239
0000000000602094 l     O .bss   0000000000000001              completed.6355
0000000000602098 g     O .bss   0000000000000004              high
0000000000602080 g     O .bss   0000000000000008              stdin@@GLIBC_2.2.5
00000000006020a0 g       .bss   0000000000000000              _end
0000000000602074 g       .bss   0000000000000000              __bss_start
000000000060209c g     O .bss   0000000000000004              low
```

```
[dmarczuk@cdmlinux AssignmentOne]$ objdump -s -j .data assign1-0

assign1-0:     file format elf64-x86-64

Contents of section .data:
 602070 00000000                             ....
```

7. **Optimization 1**: Variables are stored in rbp or ram in the not optimized code versus the optimized code (the second screenshot), which keeps many variables in their own registers.



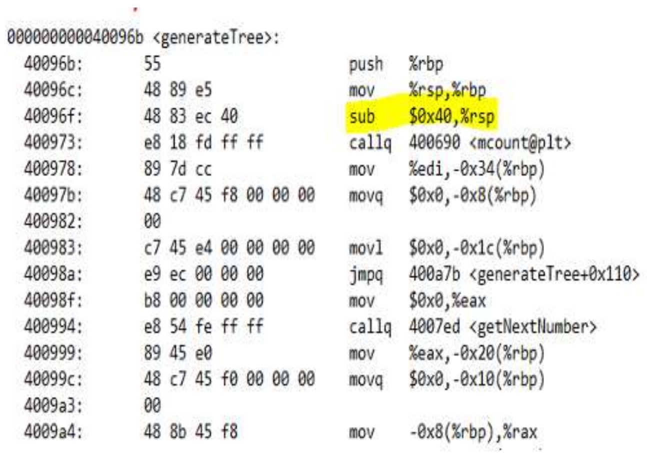
```
  40081a:       c3                      retq
NOT OPTIMIZED......
000000000040081b <obtainNumberBetween>:
  40081b:       55                      push   %rbp
  40081c:       48 89 e5                mov    %rsp,%rbp
  40081f:       48 81 ec 30 01 00 00    sub    $0x130,%rsp
  400826:       e8 65 fe ff ff          callq  400690 <mcount@plt>
  40082b:       48 89 bd d8 fe ff ff    mov    %rdi,-0x128(%rbp)
  400832:       89 b5 d4 fe ff ff       mov    %esi,-0x12c(%rbp)
  400838:       89 95 d0 fe ff ff       mov    %edx,-0x130(%rbp)
  40083e:       8b 8d d0 fe ff ff       mov    -0x130(%rbp),%ecx
  400844:       8b 95 d4 fe ff ff       mov    -0x12c(%rbp),%edx
  40084a:       48 8b 85 d8 fe ff ff    mov    -0x128(%rbp),%rax
  400851:       48 89 c6                mov    %rax,%rsi
  400854:       bf c8 0d 40 00          mov    $0x400dc8,%edi
  400859:       b8 00 00 00 00          mov    $0x0,%eax
  40085e:       e8 bd fd ff ff          callq  400620 <printf@plt>
  400863:       48 8b 15 16 18 20 00    mov    0x201816(%rip),%rdx        # 602080 <stdin@@GLIBC_2.2.5>
  40086a:       48 8d 85 f0 fe ff ff    lea    -0x110(%rbp),%rax
  400871:       be 00 01 00 00          mov    $0x100,%esi
  400876:       48 89 c7                mov    %rax,%rdi
  400879:       e8 d2 fd ff ff          callq  400650 <fgets@plt>
  40087e:       48 8d 8d e8 fe ff ff    lea    -0x118(%rbp),%rcx
  400885:       48 8d 85 f0 fe ff ff    lea    -0x110(%rbp),%rax
  40088c:       ba 0a 00 00 00          mov    $0xa,%edx
  400891:       48 89 ce                mov    %rcx,%rsi
  400894:       48 89 c7                mov    %rax,%rdi
  400897:       e8 d4 fd ff ff          callq  400670 <strtol@plt>
  40089c:       89 45 fc                mov    %eax,-0x4(%rbp)
  40089f:       8b 45 fc                mov    -0x4(%rbp),%eax
  4008a2:       3b 85 d4 fe ff ff       cmp    -0x12c(%rbp),%eax
  4008a8:       7c 94                   jl     40083e <obtainNumberBetween+0x23>
  4008aa:       8b 45 fc                mov    -0x4(%rbp),%eax
  4008ad:       3b 85 d0 fe ff ff       cmp    -0x130(%rbp),%eax
  4008b3:       7f 89                   jg     40083e <obtainNumberBetween+0x23>
  4008b5:       8b 45 fc                mov    -0x4(%rbp),%eax
  4008b8:       c9                      leaveq
  4008b9:       c3                      retq
```



```
00000000004008c0 <obtainNumberBetween>:
  4008c0:       55                      push   %rbp
  4008c1:       48 89 e5                mov    %rsp,%rbp
  4008c4:       41 55                   push   %r13
  4008c6:       41 54                   push   %r12
  4008c8:       53                      push   %rbx
  4008c9:       48 81 ec 18 01 00 00    sub    $0x118,%rsp
  4008d0:       e8 bb fd ff ff          callq  400690 <mcount@plt>
  4008d5:       49 89 fd                mov    %rdi,%r13
  4008d8:       41 89 f4                mov    %esi,%r12d
  4008db:       89 d3                   mov    %edx,%ebx
  4008dd:       0f 1f 00                nopl   (%rax)
  4008e0:       89 d9                   mov    %ebx,%ecx
  4008e2:       44 89 e2                mov    %r12d,%edx
  4008e5:       4c 89 ee                mov    %r13,%rsi
  4008e8:       bf 98 0d 40 00          mov    $0x400d98,%edi
  4008ed:       31 c0                   xor    %eax,%eax
  4008ef:       e8 2c fd ff ff          callq  400620 <printf@plt>
  4008f4:       48 8b 15 85 17 20 00    mov    0x201785(%rip),%rdx        # 602080 <stdin@@GLIBC_2.2.5>
  4008fb:       48 8d bd e0 fe ff ff    lea    -0x120(%rbp),%rdi
  400902:       be 00 01 00 00          mov    $0x100,%esi
  400907:       e8 44 fd ff ff          callq  400650 <fgets@plt>
  40090c:       48 8d b5 d8 fe ff ff    lea    -0x128(%rbp),%rsi
  400913:       48 8d bd e0 fe ff ff    lea    -0x120(%rbp),%rdi
  40091a:       ba 0a 00 00 00          mov    $0xa,%edx
  40091f:       e8 4c fd ff ff          callq  400670 <strtol@plt>
  400924:       39 c3                   cmp    %eax,%ebx
  400926:       7c b8                   jl     4008e0 <obtainNumberBetween+0x20>
  400928:       41 39 c4                cmp    %eax,%r12d
  40092b:       7f b3                   jg     4008e0 <obtainNumberBetween+0x20>
  40092d:       48 81 c4 18 01 00 00    add    $0x118,%rsp
  400934:       5b                      pop    %rbx
  400935:       41 5c                   pop    %r12
  400937:       41 5d                   pop    %r13
  400939:       5d                      pop    %rbp
  40093a:       c3                      retq
  40093b:       0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)
```

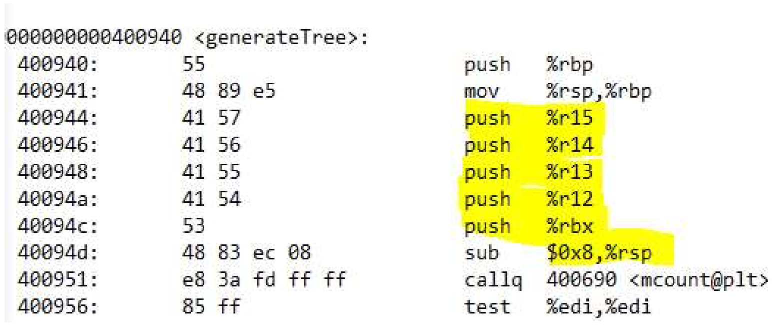
Optimization 2: In tree.c, the generateTree() function, the non optimized version is allocating space on the stack for local variables instead of using registers r15 to r12 and rbx. The difference between these two sub commands between non optimized and optimized version is 0x4 in the sub command. Sub command allocates less space on the stack for the optimization and instead uses registers to do operations, whereas the non optimized version uses more space on the stack and does not take advantage of registers.

No optimization-



```
000000000040096b <generateTree>:
  40096b:       55                      push   %rbp
  40096c:       48 89 e5                mov    %rsp,%rbp
  40096f:       48 83 ec 40             sub    $0x40,%rsp
  400973:       e8 18 fd ff ff          callq  400690 <mcount@plt>
  400978:       89 7d cc                mov    %edi,-0x34(%rbp)
  40097b:       48 c7 45 f8 00 00 00    movq   $0x0,-0x8(%rbp)
  400982:       00
  400983:       c7 45 e4 00 00 00 00    movl   $0x0,-0x1c(%rbp)
  40098a:       e9 ec 00 00 00          jmpq   400a7b <generateTree+0x110>
  40098f:       b8 00 00 00 00          mov    $0x0,%eax
  400994:       e8 54 fe ff ff          callq  4007ed <getNextNumber>
  400999:       89 45 e0                mov    %eax,-0x20(%rbp)
  40099c:       48 c7 45 f0 00 00 00    movq   $0x0,-0x10(%rbp)
  4009a3:       00
  4009a4:       48 8b 45 f8             mov    -0x8(%rbp),%rax
```

With optimization-



```
0000000000400940 <generateTree>:
  400940:       55                      push   %rbp
  400941:       48 89 e5                mov    %rsp,%rbp
  400944:       41 57                   push   %r15
  400946:       41 56                   push   %r14
  400948:       41 55                   push   %r13
  40094a:       41 54                   push   %r12
  40094c:       53                      push   %rbx
  40094d:       48 83 ec 08             sub    $0x8,%rsp
  400951:       e8 3a fd ff ff          callq  400690 <mcount@plt>
  400956:       85 ff                   test   %edi,%edi
```