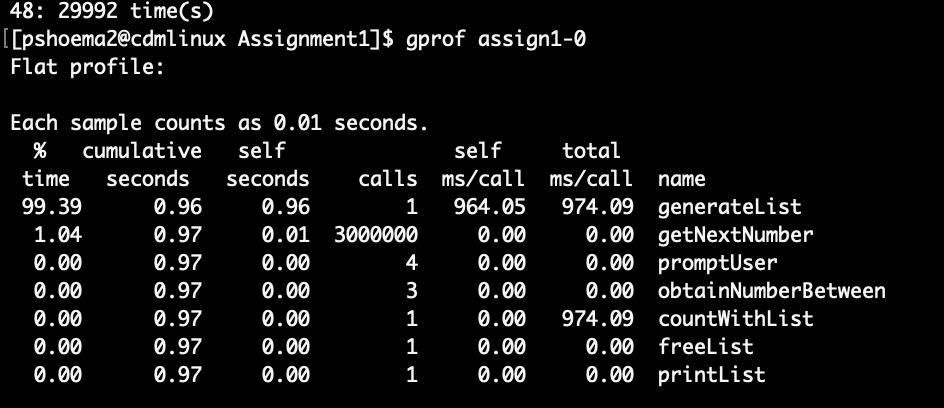
**Assignment 1 Answers**

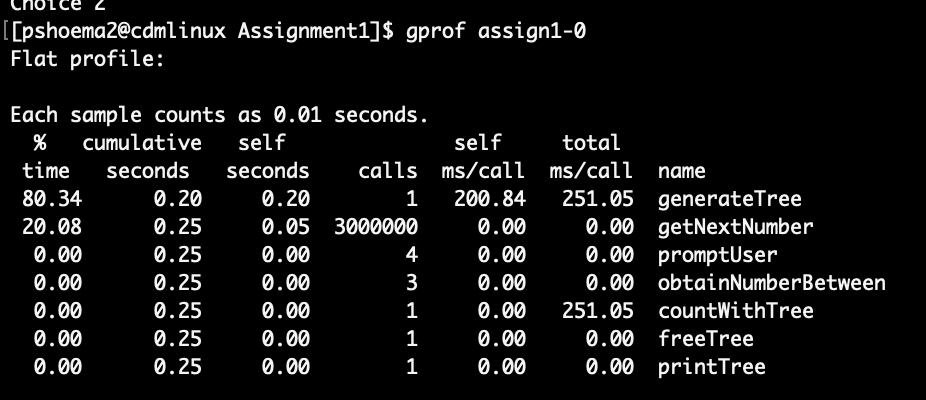
3) a. How for 3 million numbers how many self seconds did generateList() take?

**ANSWER: 0.96 Seconds**



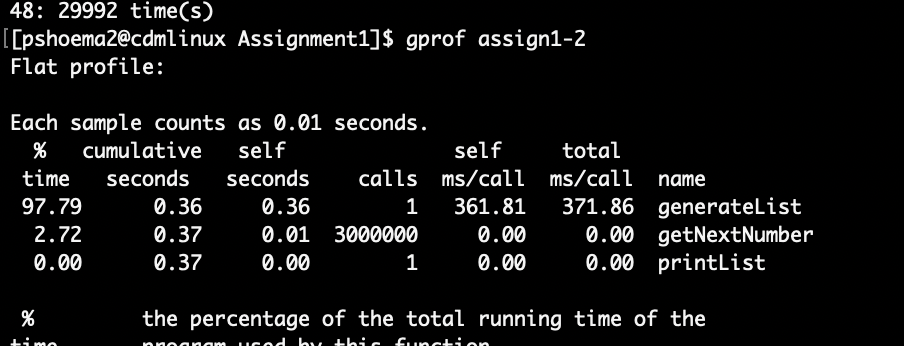
3) b. How for 3 million numbers how many self seconds did generateTree() take?

**ANSWER: 0.20 Seconds**

****

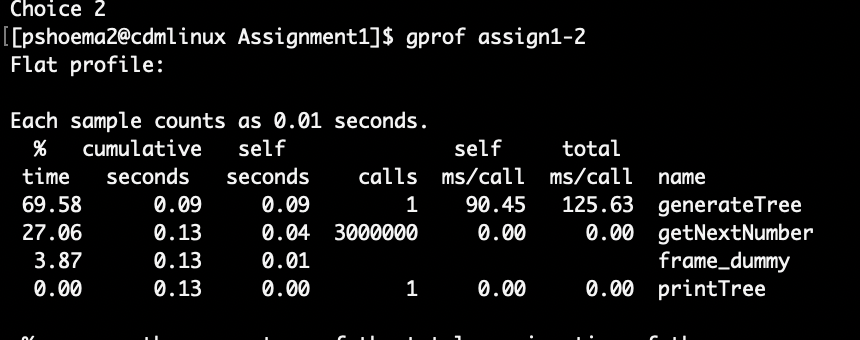
4) a. How for 3 million numbers how many self seconds did generateList() take?

**ANSWER: 0.36 Seconds**



4) b. How for 3 million numbers how many self seconds did generateTree() take?

**ANSWER: 0.09 Seconds**



5)

Which is faster:

* A bad algorithm and data-structure optimized with -O2
* A good algorithm and data-structure optimized with -O0

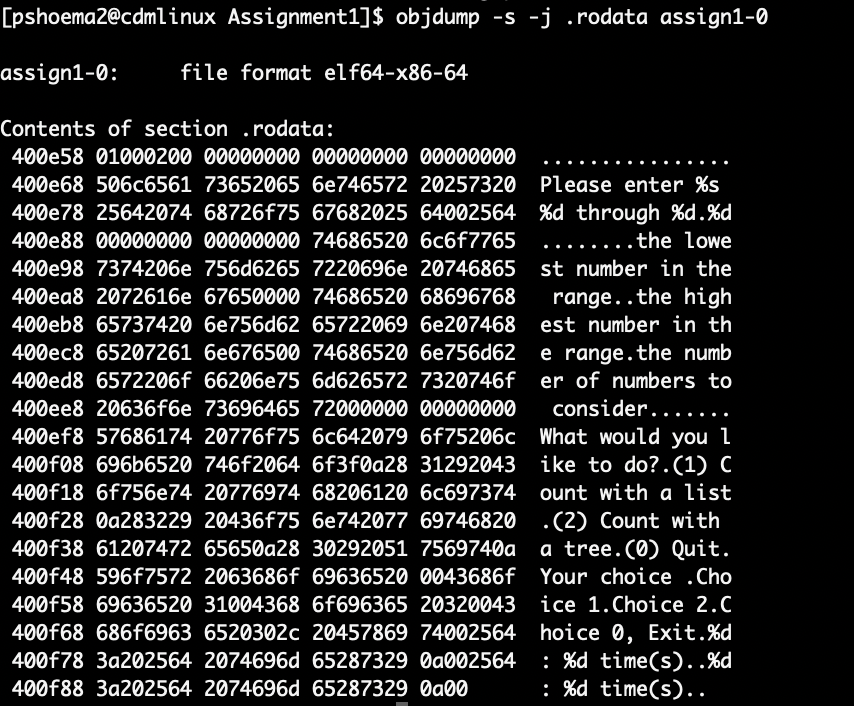
**ANSWER: A good algorithm and data-structure optimized with -O0**

6) a. The string "%d: %d time(s)\n" in printList()

**ANSWER:**

**Command:** objdump -s -j .rodata assign1-0

**Result:**

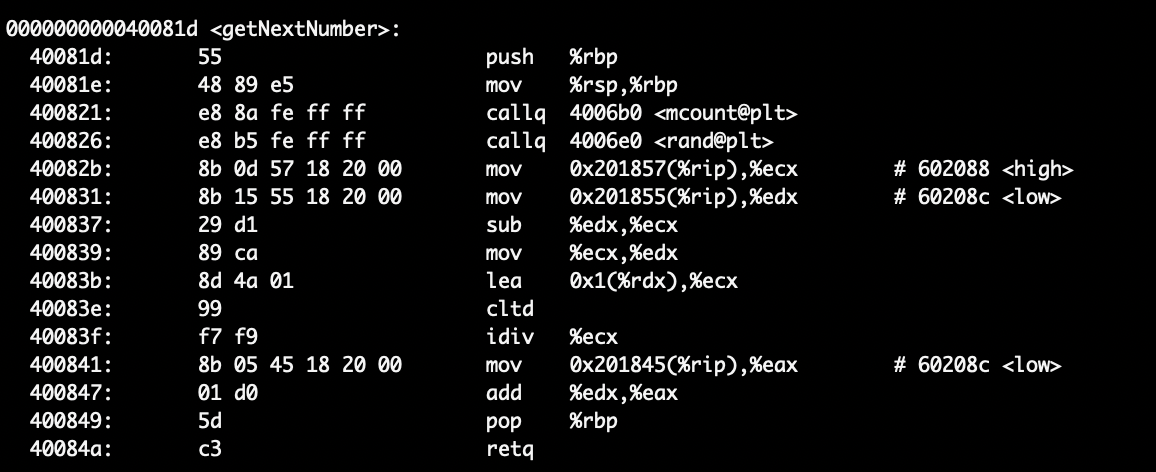
****

6) b. The code for getNextNumber()

**ANSWER:**

**Command:** objdump -d -j .text assign1-0

**Result:**

****

6) c. The global variable high

**ANSWER:**

**Command:** objdump -s -t assign1-0

**Result:**

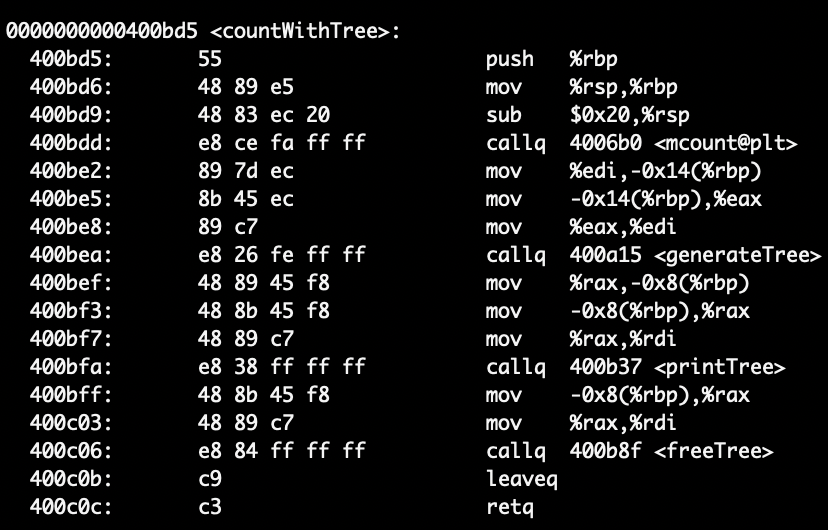
****

6) d. treePtr in countWithTree()

**ANSWER:**

**Command:** objdump -d -j .text assign1-0; although treePtr specifically cannot be identified as it is a pointer and will be created on the stack at run-time

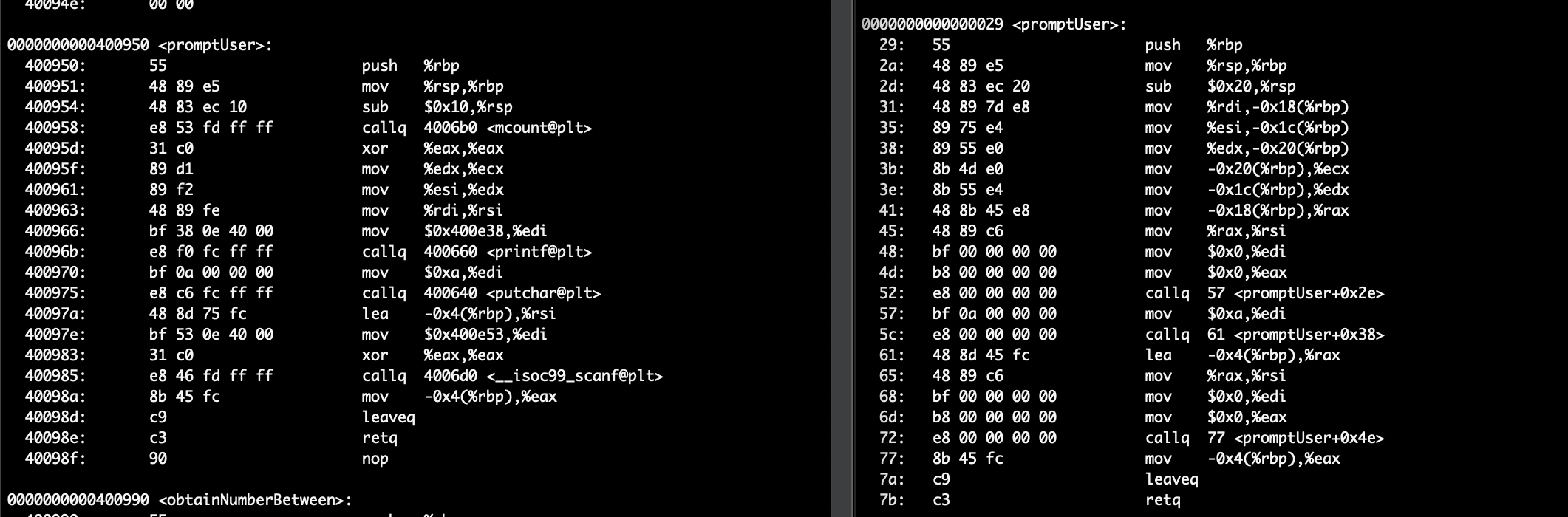
**Result:**



7) Optimizations:

1: Using registers to hold variables. The left screen shows the optimized code while the right reflects not optimized code.

The optimized “promptUser” function stores variables more effectively using registers as shown below.



2: Using registers to hold variables. The left screen shows the optimized code while the right reflects not optimized code.

The optimized “getNextNumber” function stores variables more effectively using registers as shown below.

