2.

extern void bubbleSort(int arrayLen, int\*array);

extern void insertionSort(int arrayLen, int\*array);

extern void exchange(int\*array, int i, int j);

3.

a) 18.25

19.01

b) 115.18

117.6

4.

a) 3.4

3.35

b) 61.52

61

5.

A good algorithm and data-structure optimized with -O0

6.

A)

Command:

A screenshot of a cell phone

Description automatically generated

Result:

A close up of text on a black background

Description automatically generated

B)

It’s a local variable so it’s in stack and cannot be shown.

C)

Command:

A screenshot of a cell phone

Description automatically generated

Result:

The global variable array[] in sortProg.c lives in stack so cannot be shown.

D)

Command:

A screenshot of a cell phone

Description automatically generated

Result:

A close up of text on a black background

Description automatically generated

A close up of text on a black background

Description automatically generated

7.

**Use insertionSort as an example to compare the differences between sortO0 and sortO2.**

**From the comparison of assembly code we can tell, sortO2 has more registers are being pushed, and use some call functions instead of mov instructions. Also it uses more xor and jne to make code run faster.**

// my code for sortO0

gcc -c -pg -O0 bubbleSort.c

gcc -c -pg -O0 insertionSort.c

gcc -c -pg -O0 sortProg.c

gcc bubbleSort.o insertionSort.o sortProg.o -pg -O0 -o sortO0

./sortO0

1

objdump -j -d .text insertionSort.c

objdump -d insertionSort.o

// my code or sortO2:

gcc -c -pg -O2 bubbleSort.c

gcc -c -pg -O2 insertionSort.c

gcc -c -pg -O2 sortProg.c

gcc bubbleSort.o insertionSort.o sortProg.o -pg -O2 -o sortO2

./sortO2

1

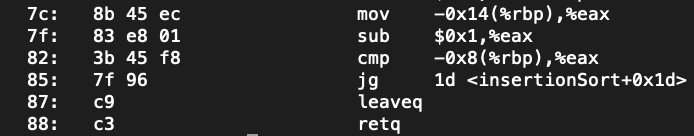
objdump -j -d .text insertionSort.c

objdump -d insertionSort.o

For sortO0:

A close up of text on a black background

Description automatically generated



For sortO2:

A close up of text on a black background

Description automatically generated

A picture containing electronics

Description automatically generated