Lab Report

ECPE 170 – Computer Systems and Networks – Spring 2016

Name: Nicolas Ahn

Lab Topic: Performance Measurement (Lab #: 4)

I am booting Linux Directly. (Skip questions 1-5)

Question #6 Boot Linux. With no applications running in Linux, how much RAM is available inside the virtual machine? The "System Monitor" program should report that information. This is the space that is actually available for our test application.

Answer:

```
> grep MemTotal /proc/meminfo
MemTotal:       15170152 kB
```

Question #7 What is the code doing? (Describe the algorithm in a paragraph, focusing on the combine1() function.)

Answer:

The code is either getting the product or the sum of the input vector, which could contain Floats or Integers as elements.

Question #8 What is the largest number of elements that the vector can hold WITHOUT using swap storage (virtual memory), and how much memory does it take? Be sure to leave enough memory for Firefox and LibreOffice, since you'll need those when running this lab as well.

Answer:

2 055 000 000

Question #9 What vector size are you using for all experiments in this lab?

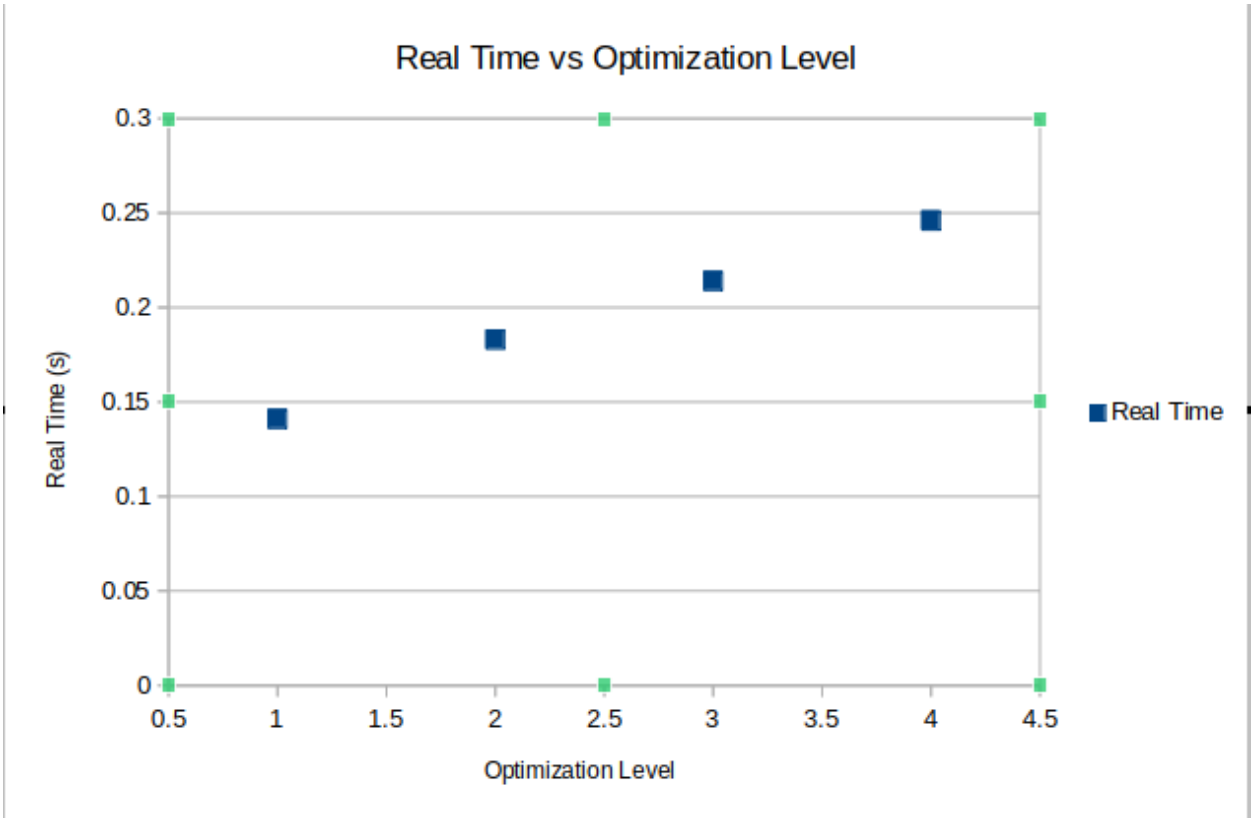
Answer:

200000000

Question #10 How much time does the compiler take to finish with (a) no optimization, (b) with -O1 optimization, (c) with -O2 optimization, and (d) with -O3 optimization? Report the Real time, which is the "wall clock" time. Create both a table and a graph in LibreOffice Calc.

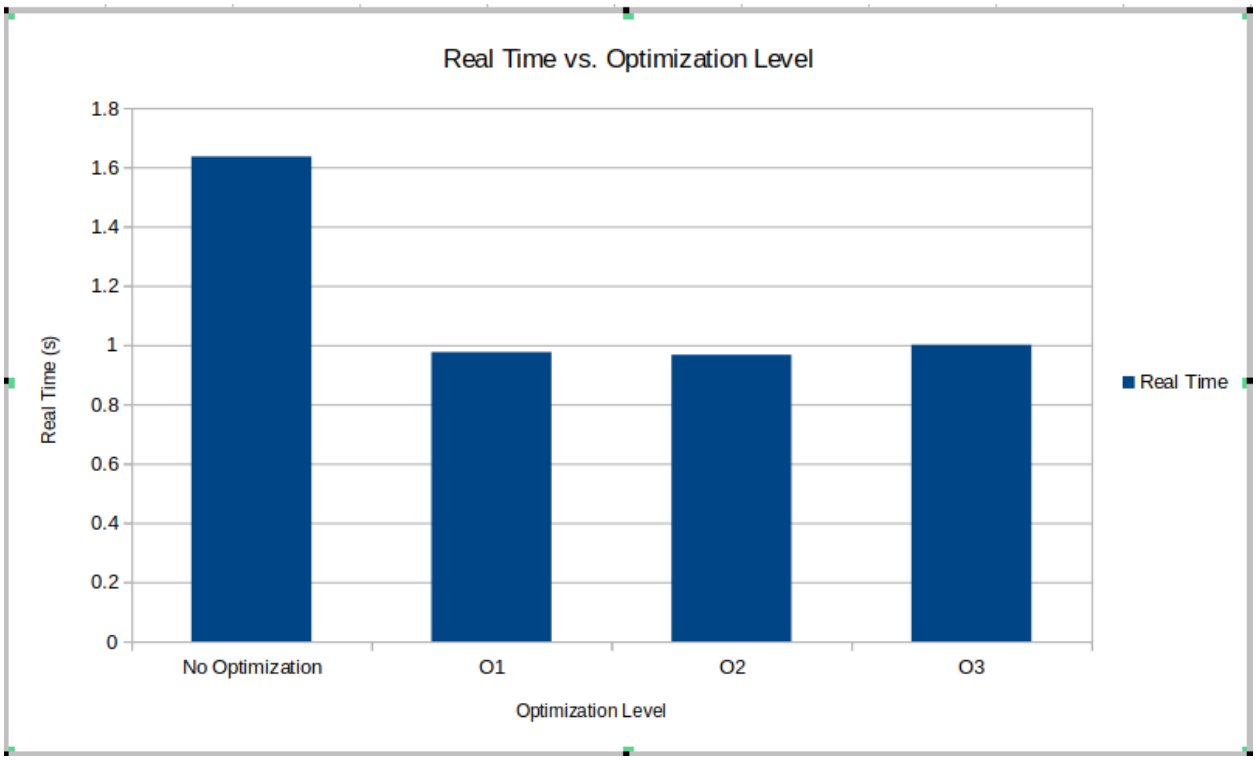
Answer:

	Real Time
No Optimization	0.141
O1	0.183
O2	0.214
O3	0.246



Question #11 How much time does the **program** take to finish with (a) no optimization, (b) with -O1 optimization, (c) with -O2 optimization, and (d) with -O3 optimization? Report the Real time, which is the "wall clock" time. Create both a table and a graph in LibreOffice Calc.
Answer:

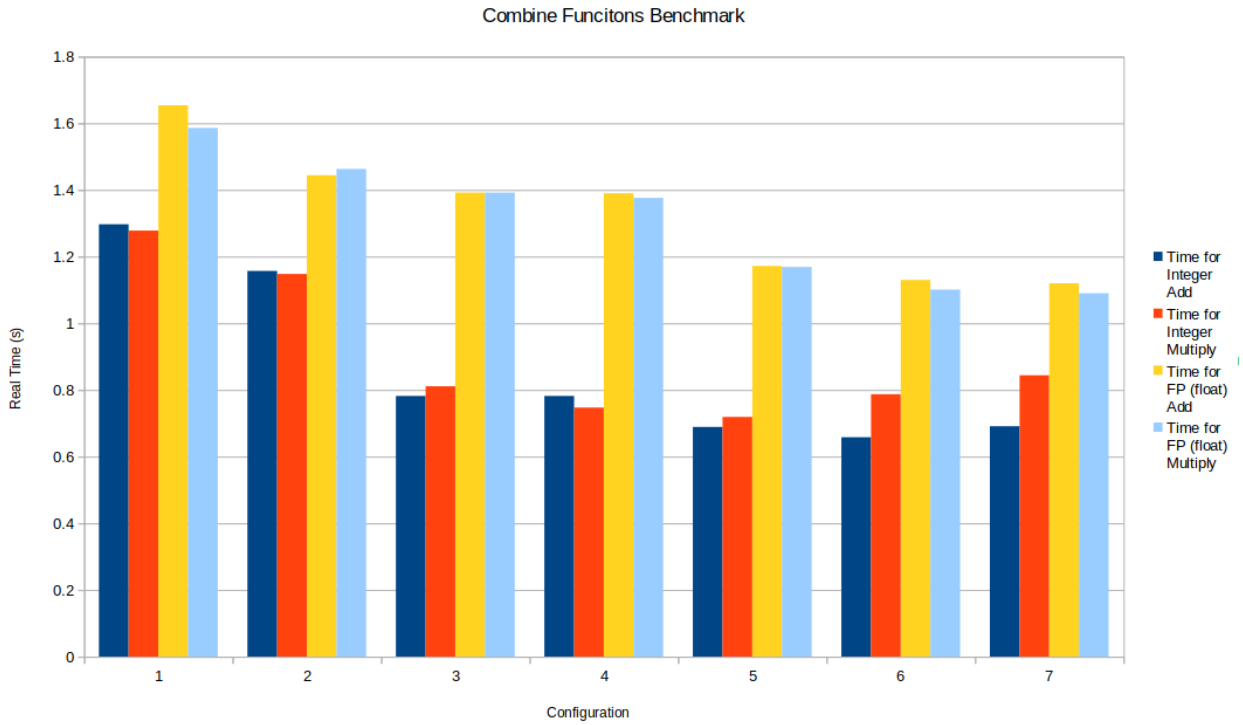
	Real Time
No Optimization	1.636
O1	0.976
O2	0.967
O3	1.001

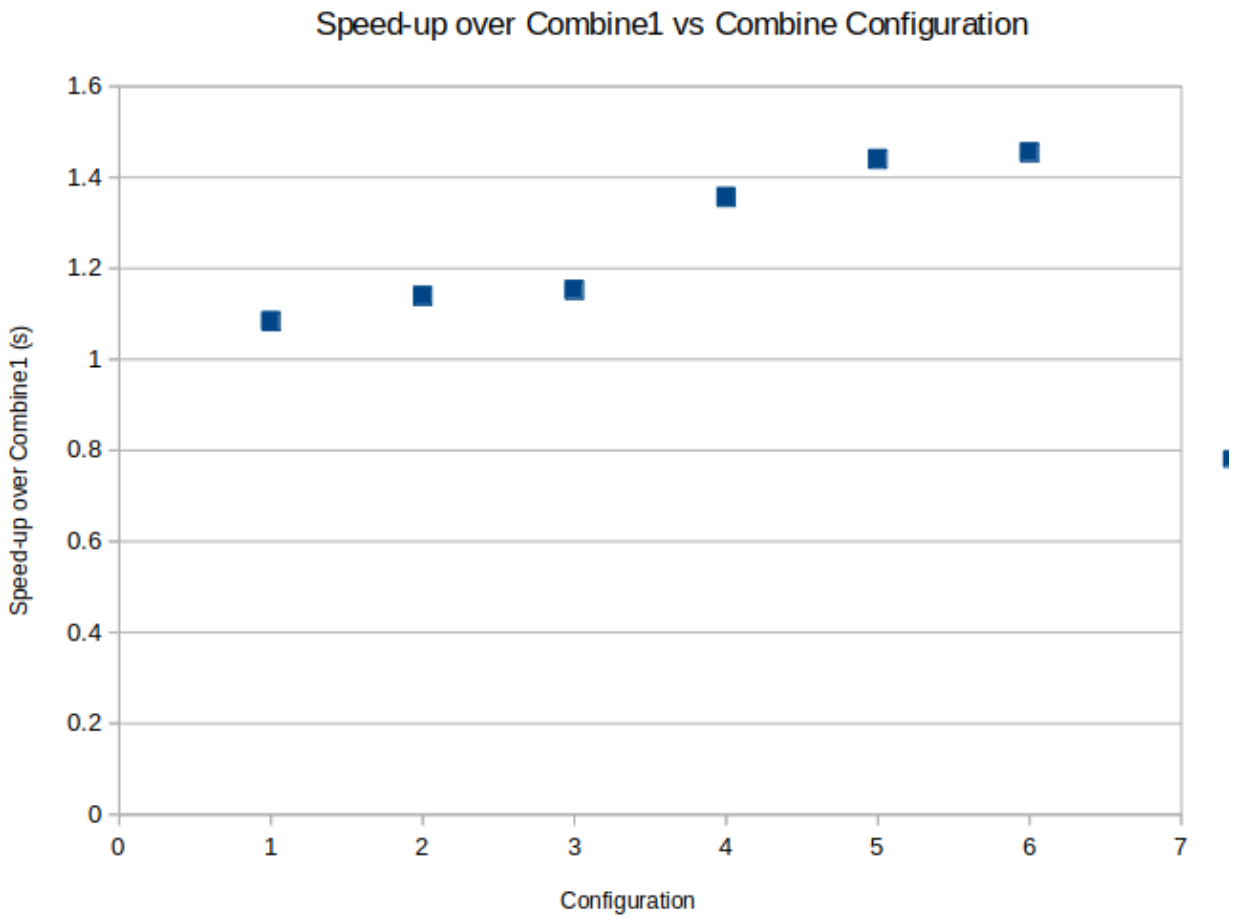


Question #12 After implementing each function, benchmark it for a variety of data types and mathematical operations. Fill in the table below as you write each function.
Answer:

Configuration	Vector Size (elements)	Vector Size (MB)	Time for Integer Add	Time for Integer Multiply	Time for FP (float) Add	Time for FP (float) Multiply
combine1()	200000000	2 055 000 000	1.297	1.278	1.654	1.586
combine2()	200000000	2 055 000 000	1.157	1.148	1.444	1.463
combine3()	200000000	2 055 000 000	0.782	0.811	1.392	1.392
combine4()	200000000	2 055 000 000	0.782	0.747	1.39	1.376
combine5x2()	200000000	2 055 000 000	0.689	0.719	1.172	1.169
combine5x3()	200000000	2 055 000 000	0.658	0.787	1.13	1.101
combine6()	200000000	2 055 000 000	0.691	0.844	1.12	1.13

Question #13 Using LibreOffice Calc, make two graphs:
Graph 1: Create a single graph that shows the data in the table created, specifically the four time columns. You don't need to plot vector size.
Graph 2: For FP (float) multiply only, plot a line graph that shows the speed-up of combine2(), combine3(), combine4(), combine5x2(), combine5x3(), and combine6() over combine1() for the vector size tested in Question 12. Plot speed-up on the y axis and function names on the x-axis. Note that the speed-up of program A over program B is defined as (TB/TA) where TB is the execution time for program B and TA is the execution time for program A.
Answer:
Note: Labels 5 = combine_program 5x2, 6 = combine_program 5x3, and combine_program 7 = 6





#14:

- **combine1()**

```
11 void combine1(vec_ptr v, data_t *dest)
12 {
13   printf("Running combine1() - No code-level optimizations\n");
14
15   long int i;
16
17   *dest = IDENT;
18
19   for(i=0; i < vec_length(v); i++)
20     {
21     ¦ data_t val;
22     ¦ get_vec_element(v, i, &val);
23     ¦ *dest = *dest OP val;
24     }
25 }
```

- **combine2()**

```
28 // CODE MOTION OPTIMIZATION:
29
30 // Move the call to vec_length() out of the loop
31 // because we (the programmer) know that the vector length will
32 // not change in the middle of the combine() function.
33 //  The compiler, though, doesn't know that!
34 void combine2(vec_ptr v, data_t *dest)
35 {
36   printf("Running combine2()\n");
37   printf("Added optimization: Code motion\n");
38
39   // XXX - STUDENT CODE GOES HERE - XXX
40   long int i;
41   long int size = vec_length(v);
42   *dest = IDENT;
43
44   for(i=0; i < size; i++)
45   {
46     ¦ data_t val;
47     ¦ get_vec_element(v, i, &val);
48     ¦ *dest = *dest OP val;
49   }
50
51 }
```

- **combine3()**

```
60 void combine3(vec_ptr v, data_t *dest)
61 {
62   printf("Running combine3()\n");
63   printf("Added optimization: Reducing procedure calls\n");
64
65   // XXX - STUDENT CODE GOES HERE - XXX
66   long int i;
67   long int size = vec_length(v);
68   *dest = IDENT;
69   data_t * VData = get_vec_start(v);
70
71   for(i=0; i < size; i++)
72   {
73     ¦ *dest = *dest OP VData[i];
74   }
75 }
```

- **combine4()**

```
85 void combine4(vec_ptr v, data_t *dest)
86 {
87   printf("Running combine4()\n");
88   printf("Added optimization: Eliminating unneeded memory accesses\n");
89
90   // XXX - STUDENT CODE GOES HERE - XXX
91   long int i;
92   long int size = vec_length(v); //variable to hold vector length
93   *dest = IDENT;
94   data_t  accumulate = IDENT;
95   data_t * VData = get_vec_start(v);
96   for(i=0; i < size; i++)
97   {
98     ¦ accumulate =  accumulate OP VData[i];
99   }
100  *dest = accumulate;
101 }
```

- **Combine5x2()**

```
106 void combine5x2(vec_ptr v, data_t *dest)
107 {
108   printf("Running combine5x2()\n");
109   printf("Added optimization: Loop unrolling x2\n");
110
111   // XXX - STUDENT CODE GOES HERE - XXX
112   long int i;
113   long int size = vec_length(v); //variable to hold vector length
114   *dest = IDENT;
115   data_t  accumulate = IDENT;
116   data_t * VData = get_vec_start(v);
117
118   if(size % 2 == 1){
119     accumulate = accumulate OP VData[size - 1];
120     size -= 1;
121   }
122   for(i=0; i < size; i+=2)
123   {
124     ¦ accumulate =  (accumulate OP VData[i]) OP VData[i+1];
125   }
126
127   *dest = accumulate;
128 }
```

- **combine5x3()**

```
132 void combine5x3(vec_ptr v, data_t *dest)
133 {
134   printf("Running combine5x3()\n");
135   printf("Added optimization: Loop unrolling x3\n");
136 
137   // XXX - STUDENT CODE GOES HERE - XXX
138   long int i;
139   long int size = vec_length(v);
140   *dest = IDENT;
141   data_t  accumulate = IDENT;
142   data_t * VData = get_vec_start(v);
143 
144   if(size % 3 == 1){
145     accumulate =  accumulate OP VData[size - 1];
146     size -= 1;
147   }
148   else if(size % 3 == 2){
149     accumulate = (accumulate OP VData[size - 1]) OP VData[size - 2];
150     size -= 2;
151   }
152   for(i=0; i < size; i+=3)
153   {
154     ¦ accumulate =  (accumulate OP VData[i]) OP VData[i+1] OP VData[i + 2];
155   }
156 
157   *dest = accumulate;
158 }
```

- **combine6()**

```
161 // LOOP UNROLLING x2 + 2-way parallelism
162 void combine6(vec_ptr v, data_t *dest)
163 {
164   printf("Running combine6()\n");
165   printf("Added optimization: Loop unrolling x2, Parallelism x2\n");
166 
167   // XXX - STUDENT CODE GOES HERE - XXX
168   long int i;
169   long int size = vec_length(v); //variable to hold vector length
170   *dest = IDENT;
171   data_t  accumulate0 = IDENT;
172   data_t accumulate1 = IDENT;
173   data_t * VData = get_vec_start(v);
174 
175   for( i = 0; i < size; i+=2){
176     accumulate0 =  accumulate0 OP VData[i];
177     accumulate1 = accumulate1 OP VData[i+1];
178   }
179 
180   *dest = accumulate0 OP accumulate1;
181 }
```