CPE 315 Lab #4 Report – Performance Measurement and Optimization

Justin Nguyen

In this lab, I ran various profiling tools to record the performance of the matadd program I wrote at various compiler optimization levels, loop unrolling levels, and utilizing the gcc '-pg' flag.

gcc Optimization Raw Performance Data

Optimization Level	Run Number	Cycles	Instructions	Clock Speed (GHz)	Branch Misses	Execution time (s)
0	1	1160639909	973328043	1.102	3968894	1.06008626
0	2	1159010858	973327826	1.143	3986043	1.023985977
0	3	1153828514	973317372	1.1	3966682	1.05528075
0	4	1160705829	973327201	1.157	3968638	1.006978984
0	5	1176877827	973327284	1.157	3968788	1.020666193
1	1	1161457505	973326930	1.157	3968615	1.009570019
1	2	1155744380	973310580	1.162	3957211	0.998102963
1	3	1197315625	973327742	1.156	3969403	1.039951254
1	4	1170660241	973327403	1.15	3984363	1.029232163
1	5	1160732577	973326963	1.157	3967754	1.006400808
2	1	1155962704	973310594	1.162	3957287	0.999550355
2	2	1155960826	973310690	1.162	3957037	0.998723117
2	3	1205037626	973310479	1.161	3957860	1.041864687
2	4	1163439132	973310758	1.162	3957206	1.037600425
2	5	1160881470	973327250	1.157	3968386	1.006416746
3	1	1161174806	973327870	1.102	3969193	1.059137304
3	2	1156007916	973310656	1.162	3957131	0.998394733
3	3	1171237291	973310843	1.163	3957508	1.010783142
3	4	1155927188	973310867	1.162	3957287	0.998357024
3	5	1176045216	973326907	1.158	3968286	1.019422341

Table 1: Raw data obtained from running 'perf stat' on gcc optimized matadd

Optimization Level	Run Number	Cycles	Instructions	Clock Speed (GHz)	Branch Misses	Execution time (s)	matadd % runtime utilized
1	1	1177855162	973337388	1.156	3971807	1.051370187	84.41
1	2	1162329122	973337256	1.155	3971167	1.024441237	71.6
1	3	1174782293	973320786	1.16	3959758	1.042016771	83.54
1	4	1167012073	973320966	1.105	3959837	1.090423329	57.83
1	5	1156902380	973320730	1.16	3960160	1.011457204	71.6
Averages		1167776206	973327425.2	1.1472	3964545.8	1.043941746	73.796

Table 2: Raw data from gprof profiling of gcc –O1 optimized matadd

Loop Unrolling Raw Performance Data

Loop Unroll Level	Run Number	Cycles	Instructions	Clock Speed (GHz)	Branch Misses	Execution time(s)
1	1	1157752515	973310705	1.106	3957534	1.052190133
1	2	1155900570	973310585	1.161	3959078	0.998589576
1	3	1172572906	973310626	1.162	3960274	1.012336993
1	4	1155629427	973310645	1.162	3957456	0.997827964
1	5	1156713662	973310853	1.105	3958257	1.054886377
2	1	1082491866	873697975	1.1	3957856	0.989554805
2	2	1083549958	873697876	1.159	3957623	0.938293045
2	3	1100904633	873697673	1.16	3957924	0.952159942
2	4	1083186047	873697638	1.159	3957444	0.938157785
2	5	1098722094	873697695	1.16	3960453	0.950458122
4	1	1036668804	835701403	1.148	3981627	0.911096023
4	2	1038437380	835686726	1.158	3960093	0.900075788
4	3	1038248772	835686942	1.158	3957488	0.899688705
4	4	1039558319	835686849	1.158	3957477	0.905002234
4	5	1059409556	835701381	1.154	3968039	0.921584281
8	1	1030533210	819644784	1.148	3981500	0.903255312
8	2	1016570999	819630588	1.157	3957198	0.882028693
8	3	1015843158	819630584	1.157	3957371	0.880965466
8	4	1020817000	819644720	1.153	3967954	0.888879927
8	5	1020681710	819644564	1.153	3967711	0.888177325

Table 3: Raw data obtained from running 'perf stat' on inner loop unrolled matadd. Note 'Loop Unroll Level' indicates the number of times the inner loop logic of matadd is repeated.

Results

gcc Optimization Results

matadd – Pi	-O0	-O1	-O2	-O3
Average CPI	1.194063582	1.20122606	1.200287273	1.195990588
Instructions	973325545.2	973323924	973313954.2	973317428.6
Branch-misses	3971809	3969469.2	3959555.2	3961881
Clock Speed (Hz)	1131800000	1156400000	1160800000	1149400000
Runtime (measured)	1.033399633	1.01665144	1.016831066	1.017218909
Runtime (equation)	1.026870991	1.01105333	1.006423459	1.012770562
Speedup		*[1]1.01647388	1.016294316	1.015906826

Table 4: Results for gcc optimization performance

Loop Unrolling Results

matadd - Pi	No Unrolling	Unrolled 2	Unrolled 4	Unrolled 8
Average CPI	1.191514525	1.2473088	1.247425777	1.245535114

Instructions	973310682.8	873697771	835692660.2	819639048
Branch-misses	3958519.8	3958260	3964944.8	3966346.8
Clock Speed (Hz)	1139200000	1147600000	1155200000	1153600000
Runtime (measured)	1.023166209	0.95372474	0.907489406	0.888661345
Runtime (equation)	*1.018007212	0.94960868	0.902410462	0.884959445
Speedup		*[2]1.07281081	1.127469039	1.151356717
Accel. Factor		*[3]0.13573839	0.150743579	0.150239368

Table 5: Results for unrolling performance

Example Calculations

(*) Denotes example calculation for that line

$$*Runtime = \frac{\sum instructions * \overline{CPI}}{\overline{clock\ frequency}} = \frac{973310682.8\ instr * 1.191514525 \frac{cycle}{instr}}{1139200000 \frac{cycle}{s}} = 1.018007212\ s$$

$$*^{1}Speedup = \frac{unoptimized\ runtime}{optimized\ runtime} = \frac{1.033399633\ s}{1.01665144\ s} = 1.01647388$$

$$*^{2}Speedup = \frac{original\ inner\ loop}{unrolled\ inner\ loop} = \frac{1.023166209\ s}{0.95372474\ s} = 1.07281081$$

$$*^{3}Accel.Factor = \frac{enhancement(1-speed\ up)}{speed\ up(1-enhancement)} = \frac{2(1-1.07281081)}{1.07281081(1-2)} = 0.13573839$$

Discussion

In all of the averaged cases, the runtime equation slightly underestimated the runtime compared to the runtime that I measured (by a few milliseconds every trial). I also noticed that from trial to trial the instruction count always changed by a few thousand instructions around the average. In most cases, I feel that the compiler is optimizing specific instructions with an instruction/cycle count trade off that is very close and the result is an oscillation that occurs in these values.

When the gcc optimization is run on my implementation of matadd, over several trials on different Pi's, the CPI of the optimized program always fluctuates, but the runtime rarely improves. It is clear that the gcc compiler is optimizing off of a set of rules and conditions that in general optimize certain code structures, but that is unclear without taking a closer look at the compiler.