ACA-HW1-PART-1

Arush Nandankar [2022101116]

Note: Due to the limitations of the ARM processor, power calculations could not be accurately performed. The perf energy package was unavailable, and alternative tools like PAPI were non-functional. When trying to use battery-based tools like powerstat, the results were inconsistent for the matrix multiplication program and negligible for the faster-executing programs.

CPU Specs

```
Architecture:             aarch64
  CPU op-mode(s):         64-bit
  Byte Order:             Little Endian
CPU(s):                   8
  On-line CPU(s) list:    0-7
Vendor ID:                Apple
  Model name:             Icestorm-M1
    Model:                1
    Thread(s) per core:   1
    Core(s) per socket:   4
    Socket(s):            1
    Stepping:             0x1
    Frequency boost:      enabled
    CPU(s) scaling MHz:   29%
    CPU max MHz:          2064.0000
    CPU min MHz:          600.0000
    BogoMIPS:             48.00
    Flags:                fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cp
                          it uscat ilrcpc flagm ssbs sb paca pacg dcpodp flagm2 frint
  Model name:             Firestorm-M1
    Model:                1
    Thread(s) per core:   1
    Core(s) per socket:   4
    Socket(s):            1
    Stepping:             0x1
    CPU(s) scaling MHz:   68%
    CPU max MHz:          3204.0000
    CPU min MHz:          600.0000
    BogoMIPS:             48.00
    Flags:                fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cp
                          it uscat ilrcpc flagm ssbs sb paca pacg dcpodp flagm2 frint
Caches (sum of all):
  L1d:                    768 KiB (8 instances)
  L1i:                    1.3 MiB (8 instances)
  L2:                     16 MiB (2 instances)
```

```
NUMA:
  NUMA node(s):           1
  NUMA node0 CPU(s):      0-7
Vulnerabilities:
  Gather data sampling:   Not affected
  Itlb multihit:          Not affected
  L1tf:                   Not affected
  Mds:                    Not affected
  Meltdown:               Not affected
  Mmio stale data:        Not affected
  Reg file data sampling: Not affected
  Retbleed:               Not affected
  Spec rstack overflow:   Not affected
  Spec store bypass:      Mitigation; Speculative Store Bypass disabled via prctl
  Spectre v1:             Mitigation; __user pointer sanitization
  Spectre v2:             Not affected
  Srbds:                  Not affected
  Tsx async abort:        Not affected
```

System Information

```
             .',;::::;,'.                 arushnandankar@fedora
         .';:cccccccccccc:;,.             ---------------------
      .;cccccccccccccccccccccc;.          OS: Fedora Linux Asahi Remix 40 (Workstation Editio
    .:cccccccccccccccccccccccccc:.        Host: Apple MacBook Air (M1, 2020)
  .;ccccccccccccc;.:dddl:.;ccccccc;.      Kernel: 6.11.0-400.asahi.fc40.aarch64+16k
 .:ccccccccccccc;OWMKOOXMWd;ccccccc:.     Uptime: 12 mins
.:ccccccccccccc;KMMc;cc;xMMc:ccccccc:.    Packages: 2175 (rpm)
,cccccccccccccc;MMM.;cc;;WW::ccccccccc,   Shell: bash 5.2.26
:cccccccccccccc;MMM.;ccccccccccccccccc:   Resolution: 2560x1600
:ccccccc;oxOOOo;MMM000k.;cccccccccccc:    DE: GNOME 46.5
cccccc:0MMKxdd:;MMMkddc.;cccccccccccc;    WM: Mutter
ccccc:XM0';cccc;MMM.;cccccccccccccccc'    WM Theme: Adwaita
ccccc;MMo;ccccc;MMW.;ccccccccccccccc;     Theme: Adwaita [GTK2/3]
ccccc;0MNc.ccc.xMMd:ccccccccccccccc;      Icons: Adwaita [GTK2/3]
cccccc;dNMWXXXWM0::ccccccccccccccc:,      Terminal: gnome-terminal
ccccccccc;.:odl:.;ccccccccccccccc:,.      CPU: (8) @ 2.064GHz
:cccccccccccccccccccccccccccccc:'.        Memory: 5322MiB / 7516MiB
.:ccccccccccccccccccccccc:;,..
  '::cccccccccccccc::;,.
```

Matrix Multiplication `matmul.cpp`

Without Optimization

```
Tiled matrix multiplication completed in 5.42768 seconds.
```

```
 Performance counter stats for './a.out':

          5,478.92 msec task-clock                       #    1.000 CPUs utilized
                77      context-switches                 #   14.054 /sec
                 1      cpu-migrations                   #    0.183 /sec
             1,635      page-faults                      #  298.416 /sec
    17,184,507,425      cycles                           #    3.136 GHz
   110,320,339,117      instructions                     #    6.42  insn per cycle
    13,438,315,544      branches                         #    2.453 G/sec
        34,878,217      branch-misses                    #    0.26% of all branches

       5.480068402 seconds time elapsed

       5.466267000 seconds user
       0.003972000 seconds sys
```

Metric	Value
Cycles per Instruction (CPI)	0.16
Branch Prediction Accuracy	99.74%
Instructions per Cycle (IPC)	6.42
Retired Instructions	110,320,339,117

With O2 Optimization

```
Tiled matrix multiplication completed in 0.501185 seconds.

 Performance counter stats for './matmul':

            544.42 msec task-clock                       #    0.999 CPUs utilized
                11      context-switches                 #   20.205 /sec
                 1      cpu-migrations                   #    1.837 /sec
             1,636      page-faults                      #    3.005 K/sec
     1,684,631,104      cycles                           #    3.094 GHz
     8,611,499,611      instructions                     #    5.11  insn per cycle
     1,231,658,684      branches                         #    2.262 G/sec
         1,184,032      branch-misses                    #    0.10% of all branches

       0.545141853 seconds time elapsed

       0.540795000 seconds user
       0.002973000 seconds sys
```

Metric	Value
Cycles per Instruction (CPI)	0.19
Branch Prediction Accuracy	99.9%
Instructions per Cycle (IPC)	5.11
Retired Instructions	8,611,499,611

Performance Comparison

Metric	Without Optimization	With O2 Optimization	Improvement
Execution Time	5.42768 s	0.501185 s	90.8% faster
Instructions	110,320,339,117	8,611,499,611	92.2% reduction
IPC	6.42	5.11	20.4% decrease
Branch Prediction Accuracy	99.74%	99.9%	0.16% improvement

Insights

- The O2 optimization dramatically improved performance, reducing execution time by over 90%.
- Instruction count was significantly reduced, indicating effective code optimization.
- Despite lower IPC with optimization, the overall performance improved due to the massive reduction in total instructions.
- Branch prediction accuracy was already high but slightly improved with optimization.

Branch Predication `bpred.cpp`

No optimization

```
Sum: 501330

 Performance counter stats for './bpred':

             82.14 msec task-clock                       #    0.994 CPUs utilized
                 1      context-switches                 #   12.175 /sec
                 0      cpu-migrations                   #    0.000 /sec
               336      page-faults                      #    4.091 K/sec
        49,279,905      cycles                           #    0.600 GHz
       135,440,855      instructions                     #    2.75  insn per cycle
        24,667,110      branches                         #  300.312 M/sec
           552,651      branch-misses                    #    2.24% of all branches

       0.082666696 seconds time elapsed
```

```
0.076161000 seconds user
0.005990000 seconds sys
```

Metric	Value
Cycles per Instruction (CPI)	0.36
Branch Prediction Accuracy	97.76%
Instructions per Cycle (IPC)	2.74
Retired Instructions	135,440,855

With O2 Optimization

```
Sum: 499928

 Performance counter stats for './bpred':

             53.54 msec task-clock                       #    0.991 CPUs utilized
                 0      context-switches                 #    0.000 /sec
                 0      cpu-migrations                   #    0.000 /sec
               333      page-faults                      #    6.220 K/sec
        32,121,062      cycles                           #    0.600 GHz
       107,648,032      instructions                     #    3.35  insn per cycle
        23,110,947      branches                         #  431.673 M/sec
            52,322      branch-misses                    #    0.23% of all branches

       0.054046488 seconds time elapsed

       0.051719000 seconds user
       0.001994000 seconds sys
```

Metric	Value
Cycles per Instruction (CPI)	0.29
Branch Prediction Accuracy	99.77%
Instructions per Cycle (IPC)	3.35
Retired Instructions	107,648,032

Performance Comparison

Metric	Without Optimization	With O2 Optimization	Improvement
Instructions	135,440,855	107,648,032	20.5% reduction
IPC	2.74	3.35	22.3% increase
Branch Prediction Accuracy	97.76%	99.77%	2.01% improvement

Insights

- O2 optimization led to a modest reduction in instruction count.
- IPC increased significantly with optimization, indicating better utilization of CPU resources.
- Branch prediction accuracy improved notably, which is crucial for this branch-heavy program.
- The optimization seems to have restructured the code to be more branch predictor friendly.

Binary Search Tree `bst.cpp`

No optimization

```
Inserting 50000 keys into the BST...
Insertion completed in 0.040273 seconds.
Performing 5000 random searches...
Search completed in 0.004487 seconds.
1987 keys were found out of 5000 searches.

 Performance counter stats for './bst':

             47.80 msec task-clock                       #    0.984 CPUs utilized
                17      context-switches                 #  355.619 /sec
                 1      cpu-migrations                   #   20.919 /sec
               169      page-faults                      #    3.535 K/sec
        28,671,121      cycles                           #    0.600 GHz
        54,144,372      instructions                     #    1.89  insn per cycle
         9,698,393      branches                         #  202.878 M/sec
           433,725      branch-misses                    #    4.47% of all branches

       0.048591054 seconds time elapsed

       0.045963000 seconds user
       0.001965000 seconds sys
```

Metric	Value
Cycles per Instruction (CPI)	0.53
Branch Prediction Accuracy	95.53%
Instructions per Cycle (IPC)	1.89
Retired Instructions	54,144,372

With O2 Optimization

```
Inserting 50000 keys into the BST...
Insertion completed in 0.026755 seconds.
```

```
Performing 5000 random searches...
Search completed in 0.001989 seconds.
1904 keys were found out of 5000 searches.

 Performance counter stats for './bst':

            31.67 msec task-clock                       #    0.985 CPUs utilized
                0      context-switches                 #    0.000 /sec
                0      cpu-migrations                   #    0.000 /sec
              169      page-faults                      #    5.336 K/sec
       19,000,613      cycles                           #    0.600 GHz
       29,807,266      instructions                     #    1.57  insn per cycle
        7,870,033      branches                         #  248.504 M/sec
          366,571      branch-misses                    #    4.66% of all branches

       0.032168205 seconds time elapsed

       0.029044000 seconds user
       0.002916000 seconds sys
```

Metric	Value
Cycles per Instruction (CPI)	0.63
Branch Prediction Accuracy	95.34%
Instructions per Cycle (IPC)	1.57
Retired Instructions	29,807,266

Performance Comparison

Metric	Without Optimization	With O2 Optimization	Improvement
Insertion Time	0.040273 s	0.026755 s	33.6% faster
Search Time	0.004487 s	0.001989 s	55.7% faster
Instructions	54,144,372	29,807,266	44.9% reduction
IPC	1.89	1.57	16.9% decrease
Branch Prediction Accuracy	95.53%	95.34%	0.19% decrease

Insights

- O2 optimization improved both insertion and search times significantly.
- Instruction count was nearly halved with optimization.
- IPC decreased with optimization, but the overall performance improved due to reduced instruction count.
- Branch prediction accuracy remained relatively unchanged, suggesting that the BST structure inherently has less predictable branching patterns.

Overall Observations

1. **Optimization Impact**: O2 optimization consistently improved performance across all three programs, with matrix multiplication seeing the most dramatic improvement.
2. **Instruction Reduction**: All programs saw a significant reduction in instruction count with optimization, ranging from 20.5% to 92.2%.
3. **IPC Variability**: Interestingly, IPC increased for the branch prediction program but decreased for matrix multiplication and BST with optimization. This suggests that the optimization's effect on IPC is highly dependent on the nature of the program.
4. **Branch Prediction**: The branch prediction program, as expected, showed the most significant improvement in branch prediction accuracy with optimization. The other programs had minimal changes in this metric.
5. **Optimization Strategies**: The dramatic improvements in matrix multiplication suggest that the O2 optimization is particularly effective at optimizing loop structures and memory access patterns.
6. **Program Characteristics**: Each program responded differently to optimization, highlighting the importance of profiling and optimizing based on the specific characteristics of each application.

These insights demonstrate the significant impact of compiler optimizations on program performance and the importance of choosing the right optimization strategies based on the nature of the program and the target architecture.