**EE/CE/CS 6304: Computer Architecture**

**Project # 1**



Project description

Cache design choices (i.e. # of levels, size, associativity, replacement policy etc.) affect the performance of a microprocessor. In this project, you are asked to fine-tune the cache hierarchy of an Alpha microprocessor for 4 individual benchmarks. The cache design parameters you can modify are:

- **Cache levels**: One or two levels, for data and instruction caches.

- **Unified caches**: Selection of separate vs. unified instruction/data caches. For example, you can have separate L1 caches and a unified L2 cache.

- **Size**: Cache size, one of the most important choices.

- **Associativity**: Selection of cache associativity (e.g. direct mapped, 2-way set associative, etc.).

- **Block size**: Block size of the cache, usually 64 or 32 bytes.

- **Block replacement policy**: Selection between FIFO, LRU and Random.

While larger caches generally mean better performance, they also come at a greater cost. Thus, sensible design choices and trade-offs are required. To this end, in this project you will also be asked to define a cost function and to use it in order to identify the optimal configuration.

Part 1: Find CPI

In this part, we will calculate the CPI for the four individual benchmarks. Our baseline configuration will be the Alpha 21264 EV6 configuration:

- **Cache levels**: Two levels.

- **Unified caches**: Separate L1 data and instruction cache, unified L2 cache.

- **Size**: 64K Separate L1 data and instruction caches, 1MB unified L2 cache.

- **Associativity**: Two-way set-associative L1 caches, Direct-mapped L2 cache.

- **Block size**: 64 bytes.

- **Block replacement policy**: FIFO.

**Deliverables:** Given an L1 miss penalty of 5 cycles, L2 miss penalty of 40 cycles, and one cycle cache hit/instruction execution, calculate the CPI for each benchmark.

# CPI Formulae:

 1. L1 separate and L2 separate  
  
CPI = CPI ideal + 5 \* ( L1InsMissRate \* %No of instructions + L1DataMissRate \* %No of load/store instruction) + 40 \* ( L2InsMissRate \* L1InsMissRate \* %No of instructions + L2DataMissRate \* L1DataMissRate \* %No of load/store instruction )  
  
2. L1 separate and L2 unified  
  
CPI = CPI ideal + 5 \* ( L1InsMissRate \* % No of instructions + L1DataMissRate \* % No of load/store instruction) + 40 \* ( L2MissRate \* ( L1InsMissRate \* % No of instructions + L1DataMissRate \* % No of load/store instruction ) )  
  
3. L1 unified and L2 unified  
  
CPI = CPI ideal + 5 \* ( L1MissRate \* ( % No of instructions + % No of load/store instruction) ) + 40 \* ( L2InsMissRate \* ( L1InsMissRate\*(%No of instructions + % No of load/store instruction) ) )

## Solution:

CPI = CPI ideal + 5 \* ( L1InsMissRate \* % No of instructions + L1DataMissRate \* % No of load/store instruction) + 40 \* ( L2MissRate \* ( L1InsMissRate \* % No of instructions + L1DataMissRate \* % No of load/store instruction ) )

***GCC Benchmark :***

|  |  |
| --- | --- |
| Ideal CPI | 1 |
| L1 Instruction MissRate | 0.0047 |
| %Number of instruction | 1 |
| L1 Datamiss Instructions | 0.0106 |
| LoadStore Instuction | 121892639 |
| Total Instructions | 337326984 |
| L2MissRate | 0.1311 |
|  |  |
| LoadStore Instruction/Total Instruction [Data] | 0.3613486 |
| L1 Instruction Missrate \* %ins | 0.0047 |
| L1 Datamiss Instruction \* Data | 0.0038303 |
|  |  |
| Final CPI | 1.0873843 |

## *Anagram Benchmark:*

|  |  |
| --- | --- |
| Ideal CPI | 1 |
| L1 Instruction MissRate | 0.0536 |
| %Number of Instructions | 1 |
| L1 Datamiss Instructions | 0.0763 |
| LoadStore | 1846 |
| Total Instructions | 4865 |
| L2MissRate | 0.09 |
|  |  |
| LoadStore/Total Instruction | 0.379445 |
| L1 Instruction Missrate \* %ins | 0.0536 |
| L1 Datamiss Instruction \* Data | 0.0289517 |
|  |  |
| Final CPI | 1.7099442 |

## *GO Benchmark:*

|  |  |
| --- | --- |
| Ideal CPI | 1 |
| L1 Instruction MissRate | 0.0013 |
| %Number of Instructions | 1 |
| L1 Datamiss Instructions | 0.001 |
| LoadStore | 211693195 |
| Total Instructions | 545823547 |
| L2MissRate | 0.0907 |
|  |  |
| LoadStore/Total Instruction | 0.3878418 |
| L1 Instruction Missrate \* %ins | 0.0013 |
| L1 Datamiss Instruction \* Data | 0.0003878 |
|  |  |
| Final CPI | 1.0145627 |

Part 2: Optimize CPI for each benchmark

**Deliverables**: Given a two-level cache hierarchy, 128KB available for L1 cache and 1MB available for L2 cache, identify the optimal configuration (in terms of achieved CPI) for each benchmark. You should decide between unified/separate caches, associativity, replacement policy etc. Explain the reasoning behind your design choices for each configuration. Present graphs showing the trade-offs between design choices. Note that writing scripts (python, shell) to automate the process will save you a lot of time.

## Configuration Choices and Reasoning:

1. For ease of analysis, from 3 replacement policies, FIFO, Random LRU L1 cache and L2 cache both are defined with same replacement policy.
2. Increasing associativity reduces conflict misses. Greater associativity can come at the cost of increased hit time. Therefore set associativity - Direct mapped is not used and 2 way, 4 way and 8 way associativity is taken into consideration.
3. The simplest way to reduce the miss rate is to take advantage of spatial locality and increase the block size. Larger blocks reduce compulsory misses, but they also increase the miss penalty. Because larger blocks lower the number of tags, they can slightly reduce static power. Larger block sizes can also increase capacity or conflict misses, especially in smaller caches. Therefore analysis is more focused for block sizes of values 32, 64, 128.
4. Number of sets are calculated accordingly using cache size, block size and associativity.

Number of sets = Cache size / (Block size \* Associativity)

## Number of configurations used for each of 4 benchmark:

**Cache Configuration \* size of Block \* Replacement Policy \* Set associativity for L1 cache \* Set Associativity for L2 cache**

* 3 - Cache Configuration : L1 separate, L2 separate
  + - * L1 separate, L2 unified
      * L1 unified, L2 unified
* 4 - Size of Block : 16, 32, 64, 128
* 3 - Replacement Policy : FIFO, Random, LRU
* 3 - Set Associativity : 2, 4, 8

Cache Configuration \* size of Block \* Replacement Policy \* Set associativity for L1 cache \* Set Associativity for L2 cache = 3\*4\*3\*3\*3

**= 324**

L1 separate and L2 separate:

# GCC

Figure 1:Using GCC benchmark

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Config | il1\_miss\_rate | il2\_miss\_rate | dl1\_miss\_rate | dl2\_miss\_rate | CPI |
| gcc.ll1:128:64:8:l.dl1:128:64:8:l dl2:1024:64:8:l | 0.030 | 0.0815 | 0.0091 | 0.07177 | 1.263 |
| gcc.ll1:256:64:4:l.dl1:256:64:4:l dl2:2048:32:8:l | 0.036 | 0.0852 | 0.0091 | 0.0777 | 1.329 |
| gcc.g gcc.ll1:128:32:4:l.dl1:128:32:4:l d dl2:1024:64:8:l | 0.038 | 0.0923 | 0.009 | 0.0778 | 1.346 |

Table 1 : Miss Rates and CPI Data

### Optimum Configuration for GCC in terms of Performance (CPI) :       L1  64 byte block size, 8-way associativity, LRU replacement policy

### L2 64 byte block size, 8-way associativity, LRU replacement policy

From Figure1 it can also be concluded that cache with block size of 64 bytes has better performance than block size of 32 bytes. The rings in the Figure 1 show the cluster of CPI with their block sizes.

# GO

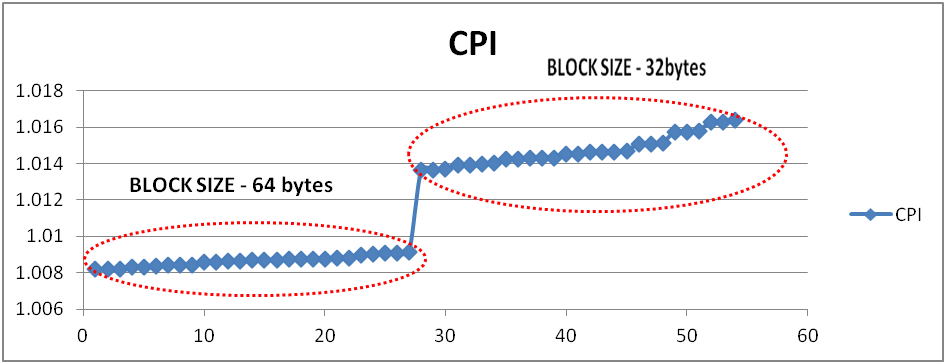


Figure 2:Using GO benchmark

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Config** | **il1\_missrate** | **il2\_miss\_rate** | **dl1\_miss\_rate** | **dl2\_miss\_rate** | **CPI** |
| go.ll1:128:64:8:r.dl1:128:64:8:r | 0.0006 | 0.9111 | 0.0250 | 0.4756 | 1.258 |
| go.ll1:128:64:8:r.dl1:128:64:8:r | 0.0006 | 0.9914 | 0.0270 | 0.5269 | 1.2233 |
| go.ll1:128:64:8:r.dl1:128:64:8:r | 0.0007 | 0.9914 | 0.0270 | 0.5668 | 1.321 |

Table 2 : Miss Rates and CPI Data

**Optimum Configuration for Go in terms of Performance (CPI) :**  
      L1 64 byte block size, 8-way associativity, random replacement policy

L2 64 byte block size, 4-way associativity, random replacement policy

# Anagram

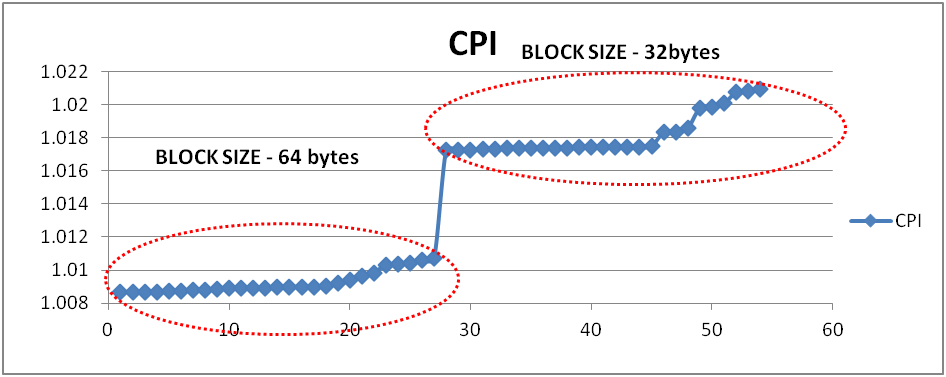


Figure 2:Using Anagram benchmark

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Config | il1\_missrate | il2\_missrate | dl1\_missrate | dl2\_missrate | CPI |
| anagram.ll1:256:64:4:f | 0 | 1 | 0.0019 | 0.2004 | 1.008656 |
| anagram.ll1:256:64:4:f. | 0 | 1 | 0.0019 | 0.2004 | 1.008656 |
| anagram.ll1:128:64:8:f. | 0 | 1 | 0.0019 | 0.2011 | 1.008674 |

Table 3 : Miss rates and CPI data

**Optimum Configuration for Anagram in terms of Performance (CPI) :**  
      L1 64 byte block size, 4-way associativity, FIFO replacement policy

L2 64 byte block size, 8-way associativity, FIFO replacement policy

L1 separate and L2 unified:

# GCC

Figure 4:Using GCC benchmark

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Config** | **il1\_miss\_rate** | **dl1\_miss\_rate** | **ul2\_miss\_rate** | **CPI** |
| gcc.ll1:128:64:8:l.dl1: | 0.0027 | 0.001 | 0.0104 | 1.016 |
| gcc.ll1:128:64:8:l.dl1: | 0.004 | 0.0016 | 0.0104 | 1.0247 |
| gcc.ll1:128:64:8:l.dl1: | 0.0052 | 0.002 | 0.0106 | 1.0321 |

Table 4 : Miss rates and CPI data

**Optimum Configuration for GCC for in terms of Performance (CPI) :**  
        L1 64 byte block size, 8-way associativity, LRU replacement policy

        L2 64 byte block size, 8-way associativity, LRU replacement policy

# ANAGRAM

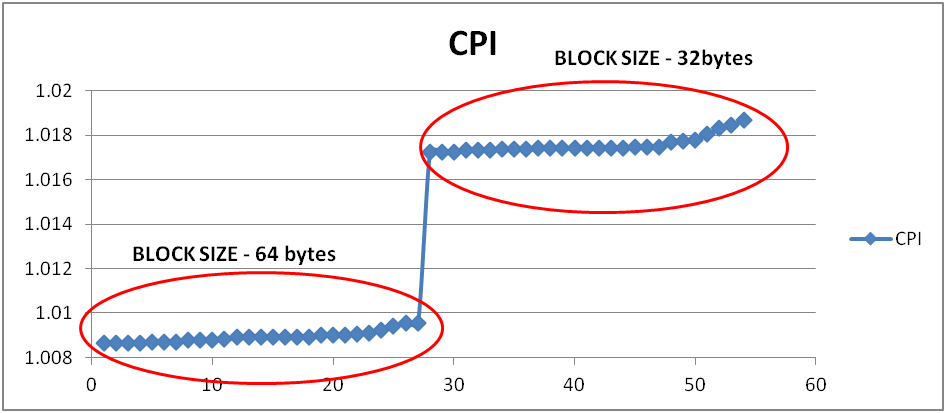


Figure 5:Using ANAGRAM benchmark

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Config** | **il1\_miss\_rate** | **dl1\_miss\_rate** | **ul2\_miss\_rate** | **CPI** |
| anagram.ll1:128:64:8:f. | 0 | 0.0019 | 0.1997 | 1.008637 |
| anagram.ll1:256:64:4:f. | 0 | 0.0019 | 0.2004 | 1.008656 |
| anagram.ll1:256:64:4:f. | 0 | 0.0019 | 0.2004 | 1.008656 |
| anagram.ll1:256:64:4:f | 0 | 0.0019 | 0.2004 | 1.008656 |

Table 5 : Miss rates and CPI data

**Optimum Configuration for Anagram in terms of Performance (CPI) :**  
      L1 64 byte block size, 8-way associativity, FIFO replacement policy

L2 64 byte block size, 2-way associativity, FIFO replacement policy

# GO

Figure 6: Using GO benchmark

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Config** | **il1\_miss\_rate** | **dl1\_miss\_rate** | **ul2\_miss\_rate** | **CPI** |
| go.ll1:128:64:8:l.dl1: | 0.0007 | 0.0008 | 0.0314 | 1.0063 |
| go.ll1:128:64:8:l.dl1: | 0.0013 | 0.001 | 0.0907 | 1.0145 |
| go.ll1:128:64:8:l.dl1: | 0.0015 | 0.001 | 0.0987 | 1.0137 |

Table 6: Miss rates and CPI data

**Optimum Configuration for Go in terms of Performance (CPI) :**  
      L1 64 byte block size, 2-way associativity, FIFO replacement policy

L2 64 byte block size, 8-way associativity, FIFO replacement policy

L1 unified and L2 unified:

# GCC

Figure 6: Using GCC benchmark

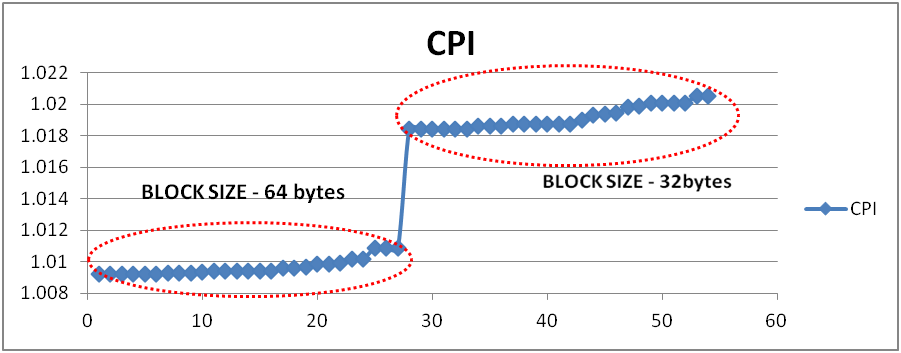
|  |  |  |  |
| --- | --- | --- | --- |
| **Config** | **ul1\_miss\_rate** | **ul2\_miss\_rate** | **CPI** |
| gcc.ul1:256:64:8:l.ul2:2048:64:8:l | 0.0137 | 0.9887 | 1.737 |
| gcc.ul1:256:64:8:l.ul2:4096:64:4:l | 0.0148 | 0.9887 | 1.796 |
| gcc.ul1:512:64:4:l.ul2:2048:64:8:l | 0.0149 | 0.9997 | 1.8096 |

Table 7 : Miss rates and CPI data

## Optimum Configuration for GCC  in terms of Performance (CPI) :       L1 64 byte block size, 8-way associativity, LRU replacement policy

## L2 64 byte block size, 8-way associativity, LRU replacement policy

# ANAGRAM

 Figure Figure 8: Using ANAGRAM benchmark

|  |  |  |  |
| --- | --- | --- | --- |
| **Config** | **ul1\_miss\_rate** | **ul2\_miss\_rate** | **CPI** |
| anagram.ul1:512:64:4:f.ul2:2048:64:8:f | 0.0005 | 0.2165 | 1.009221 |
| anagram.ul1:512:64:4:f.ul2:4096:64:4:f | 0.0005 | 0.2165 | 1.009221 |
| anagram.ul1:512:64:4:f.ul2:8192:64:2:f | 0.0005 | 0.2165 | 1.009221 |

Table 8: Miss rates and CPI data

**Optimum Configuration for Anagram in terms of Performance (CPI) :**  
      L1 64 byte block size, 4-way associativity, FIFO replacement policy

L2 64 byte block size, 8-way associativity, FIFO replacement policy

# GO

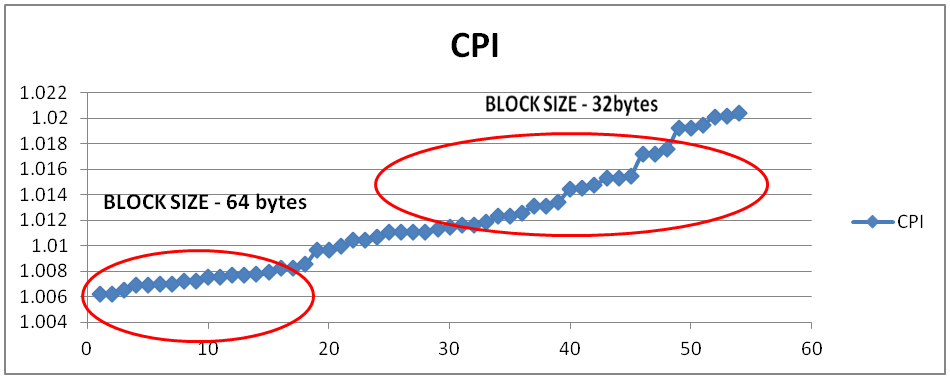


Figure 9: Using GO benchmark

|  |  |  |  |
| --- | --- | --- | --- |
| **Config** | **ul1\_miss\_rate** | **ul2\_miss\_rate** | **CPI** |
| go.ul1:256:64:8:l.ul2:2048:64:8:l | 0.0008 | 0.0357 | 1.0071 |
| go.ul1:256:64:8:l.ul2:4096:64:4:l | 0.0009 | 0.0491 | 1.008 |
| go.ul1:256:64:8:l.ul2:8192:64:2:l | 0.0011 | 0.0517 | 1.0108 |

Table 9 : Miss rates and CPI data

**Optimum Configuration for Anagram unified in terms of Performance (CPI) :**  
      L1 64 byte block size, 8-way associativity, LRU replacement policy

L2 64 byte block size, 8-way associativity, LRU replacement policy

**FINAL OPTIMAL CONFIGURATIONS:**

GO Benchmark:

* Optimal CPI value is 1.0063.
* Optimal Configuration:
  + L1 unified 64 block size,8 way set associative and LRU replacement Policy.
  + L2 unified 64 block size,8 way set associative and LRU replacement Policy.

ANAGRAM Benchmark:

* Optimal CPI value is 1.009.
* Optimal Configuration:
  + L1 separate 64 block size, 8-way set associative and random replacement policy
  + L2 unified 64 block size, 2-way set associative and random replacement policy

GCC Benchmark:

* Optimal CPI value is 1.016.
* Optimal Configuration:
  + L1 unified 64 block size, 8-way set associative and LRU replacement Policy
  + L2 unified 64 block size, 8-way set associative and LRU replacement Policy.

Conclusion:

1. Increasing the block size reduces the miss rate and thereby resulted better CPI. All three benchmarks got better result in 64 bytes block size compared to 32 byte block size. This is shown in Figures above.
2. Increasing associativity reduces the miss rate and hence better CPI. All benchmarks got optimized CPI using the higher associativity (8-way set associative) used for the analysis.
3. The LRU replacement Policy is better than FIFO and random as 2/3 benchmarks optimized configuration has LRU replacement policy.

Part 3: Define Cost function

**Deliverables**: Define a cost function, in arbitrary cost units, using any parameters you see fit. The cost function should accurately reflect the design choices. For example, change of replacement policy from FIFO to LRU could increase the cost by 5%, while doubling the cache sizes would double the cost.

## Solution:

The cost function can be defined as below.

Cost Function=0.45\*(L1 cache size) + 0.25\* (L2 Cache size)+ 0.15 \*(Unified/Separate)+ 0.1\* (L1 associativity)+ 0.1\* (L2 associativity)+ 0.025\*(L1 Replacement policy)+0.025\*(L2 Replacement policy)+ 0\*(block size)

Explanation of the cost function can be found in the below table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Cost Function** | **Weight** | **Overall Weight** | **Comment** |
| L1 Cache Size | 0.45 | f(x) = x | If size doubles, cost also doubles |
| L2 Cache Size | 0.25 | f(x) = x | If size doubles, cost also doubles |
| L1 Associativity | 0.1 | Weight \* LOG (LOG(associtivity,2)+1,LOG(Number of Blocks,2)+1) | eg. For 2048 blocks and 4 way associativity , LOG (2,12) |
| L2 Associativity | 0.1 | Weight \* LOG (LOG(associtivity,2)+1,LOG(Number of Blocks,2)+1) | eg. For 16384 blocks and 4 way associativity , LOG (2,14) |
| Unified/Separate | 0.05 | L1 Separate, L2 Separate - 1 \* Weight L1 Separate, L2 Unified - 0.5 \* Weight L1 Unified, L2 Unified - 0 | Separate data and instruction cache will involve some additional hardware cost compared to unified model. |
| L1 Repl Policy | 0.025 | LRU -1 \* Weight FIFO -0.5 \* Weight Random - 0 | Random, no extra hardware needed |
| L2 Repl Policy | 0.025 | LRU -1 \* Weight FIFO -0.5 \* Weight Random - 0 | Random, no extra hardware needed |
| Block Size | 0 | 32 Bytes - 0 64 Bytes -0 | With respect to hardware there will not be any additional cost for change in block size |

Table: Cost Function

For a given cost of a configuration say 80% if the L1Cache Size doubles, the new cost function will become 116%. Say if L1 and L2 replacement policy changes from random to LRU, the new cost function will be 84%. So assumed cost function is reasonable.

Part 4: Optimize caches for performance/cost

Given the cost function you defined in the previous part, you can now accurately select an optimal cache configuration for each benchmark, as well as all benchmarks combined.

**Deliverables**: Identify the optimal cache configuration for each benchmark, and the optimal configuration for all benchmarks (in terms of the average CPI). Present graphs showing the trade-off between CPI and cost for different design choices.

1. Reviewing from part-3 infers that the block size of 64 bytes has better performance over 32 bytes, so ignoring the 32 bytes block size for the analysis. Thus the number of iteration for each benchmark is now reduced for fixed value of L1 and L2 cache size.
2. The optimum configuration for all the benchmarks (in terms of average CPI from part 3 is L1 unified, Cache Size 128 KB, 64 byte block size, 8-way associativity and LRU policy L2 unified, Cache Size 1024 KB, 64 byte block size, 8-way associativity and LRU policy.

The below figure show the CPI graph for the overall (in terms of average CPI).

**TO OPTIMIZE THE CACHES FOR PERFORMANCE/COST:**

1. L1 and L2 Cache Sizes are optimizing
   1. For optimizing L1 Cache Size - The different sample values of L2 Cache Size, replacement policy, L1 and L2 associativity , unified/separate are taken from the optimal configuration of all the benchmarks (Average CPI) done from Part 3. And accordingly L1 cache Optimize configuration is figured out.
   2. For optimizing L2 Cache Size - The values of L1 Cache Size, replacement policy, L1 and L2 associativity , unified/separate are taken from the optimal configuration of all the benchmarks (Average CPI) done from Part 3. And accordingly L1 cache Optimize configuration is figured out.
2. After obtaining optimized values of L1 and L2 Cache size, the simulations are run to get the optimal configuration for each benchmarks, and the optimal configuration for all benchmarks (in terms of the average CPI).

**TO DETERMINE THE OPTIMUM CONFIGURATION:**

1. The CPI and Cost values are normalized using the below formula,

**C\_normalized = (C - Cmin) / (Cmax-Cmin)**

Cmin - refers to number with minimum value eg. Max CPI, Max Cost

Cmax - refers to value with maximum value eg. Min Cost, Min CPI

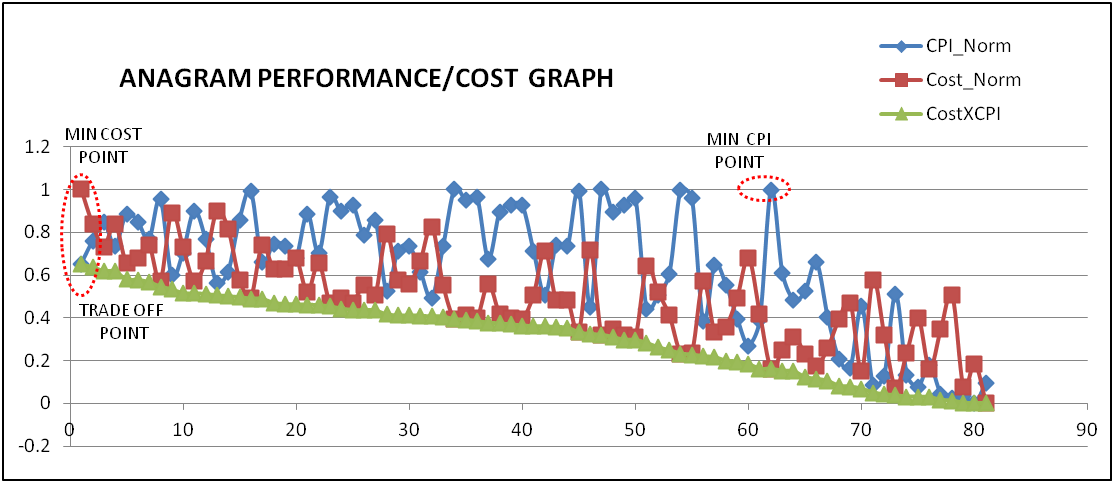
The values are normalized so that the individual range would not affect other.

1. The product of the normalized CPI and normalized cost (i.e CPI\_norm X Cost\_norm) is calculated.
2. The maximum value of the product of the normalized CPI and normalized cost (i.e CPI\_normalized X Cost\_normalized) will determine the optimum configuration.

**ANAGRAM Benchmark:**

***Trade Off Between Performance And Cost:***

The trade off point in terms of performance and cost, optimal CPI point, optimal cost point for the anagram configuration is shown in the below graph.



Optimum Configuration for Anagram in terms of Performance (CPI) is

L1 separate, 32 KB, 64 byte block size, 8-way associativity, FIFO replacement policy

L2 unified, 512 KB, 64 byte block size, 4-way associativity, FIFO replacement policy

Optimum Configuration for Anagram in terms of Cost is gcc.ul1:1024:64:2:r.ul2:8192:64:2:r

L1 unified, 64 KB, 64 byte block size, 2-way associativity, random replacement policy

L2 unified, 512 KB, 64 byte block size, 2-way associativity, random replacement policy

Optimum Configuration for Anagram in terms of Performance and Cost is

L1 unified, 64 KB, 64 byte block size, 2-way associativity, random replacement policy

L2 unified, 512 KB, 64 byte block size, 2-way associativity, random replacement policy

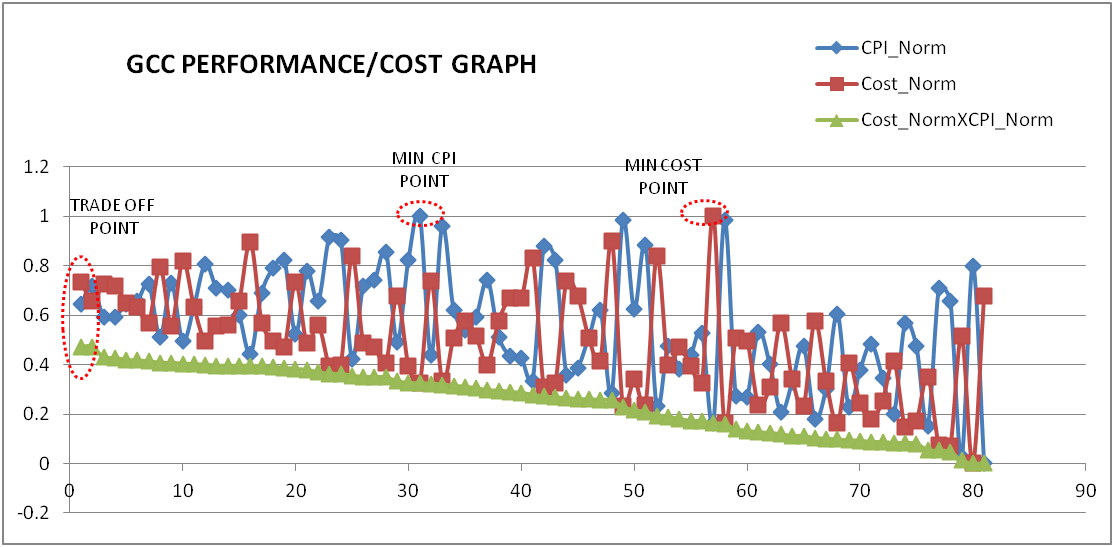
***Conclusion:***

As expected the optimum configuration based on cost is unified, 2-way associativity and random policy. The trade off is made between the performance and cost, the resulted configuration is also both L1 and L2 unified, 2-way associativity and random policy.

**GCC BENCHMARK:**

***Trade Off Between Performance And Cost:***

The trade off point in terms of performance and cost, optimal CPI point, optimal cost point for the GCC configuration is shown in the below graph.



Optimum Configuration for GCC in terms of Performance (CPI) is

L1 unified, 128 KB, 64 byte block size, 8-way associativity, LRU replacement policy

L2 unified, 1024 KB, 64 byte block size, 8-way associativity, LRU replacement policy

Optimum Configuration for GCC in terms of Cost is

L1 unified, 128 KB, 64 byte block size, 2-way associativity, random replacement policy

L2 unified, 1024 KB, 64 byte block size, 2-way associativity, random replacement policy

Optimum Configuration for GCC in terms of Performance and Cost is

L1 separate, 64 KB, 64 byte block size, 4-way associativity, random replacement policy

L2 unified, 1024 KB, 64 byte block size, 2-way associativity, random replacement policy

***Conclusion:***

As expected the optimum configuration based on cost is unified, 2-way associativity and random policy. A trade off is made between the performance and cost, the resulted configuration is L1 separate, 4-way associativity,random policy and L2 unified, 2-way associativity, random policy.

**GO Benchmark:**

***Trade Off Between Performance And Cost:***

The trade off point in terms of performance and cost, optimal CPI point, optimal cost point for the GO configuration is shown in the below graph.

Optimum Configuration for GO in terms of Performance (CPI) is

L1 separate, 32 KB, 64 byte block size, 8-way associativity, LRU replacement policy

L2 separate, 256 KB, 64 byte block size, 8-way associativity, LRU replacement policy

Optimum Configuration for GO in terms of Cost is

L1 unified, 32 KB, 64 byte block size, 2-way associativity, random replacement policy

L2 unified, 256 KB, 64 byte block size, 2-way associativity, random replacement policy

Optimum Configuration for GO in terms of Performance and Cost is

L1 separate, 64 KB, 64 byte block size, 8-way associativity, random replacement policy

L2 unified, 1024 KB, 64 byte block size, 2-way associativity, random replacement policy

***Conclusion:***

As expected the optimum configuration based on cost is unified, 2-way associativity and random policy. A trade off is made between the performance and cost, the resulted configuration is L1 separate 4-way associativity, random policy, L2 unified,2-way associativity, random policy.

**For All Benchmarks:**

Optimum Configuration for all benchmarks in terms of performance (CPI) is

L1 unified, 128 KB, 64 byte block size, 8-way associativity, LRU replacement policy

L2 unified, 1024 KB, 64 byte block size, 8-way associativity, LRU replacement policy

Optimum Configuration for all benchmarks in terms of Cost is

L1 unified, 128 KB, 64 byte block size, 2-way associativity, random replacement policy

L2 unified, 1024 KB, 64 byte block size, 2-way associativity, random replacement policy

Optimum Configuration for all benchmarks in terms of Performance and Cost is

L1 separate, 64 KB, 64 byte block size, 8-way associativity, random replacement policy

L2 unified, 1024 KB, 64 byte block size, 2-way associativity, random replacement policy

***Conclusion:***

As expected the optimum configuration based on cost is unified, 2-way associativity and random policy. A trade off is made between the performance and cost, the resulted configuration **is L1 separate, 8-way associativity, random policy and L2 unified, 2-way associativity, random policy**.