Cache design tradeoffs

Simulation Project report

Advanced architecture

University of Jordan
Computer engineering department

Submitted by: Abeer Hyari

Jan-2010

Design options

Cache memories remain one of the hot topics in the computer architecture research, since the ever-increasing speed gap between processor and memory only emphasizes the need for more efficient memory hierarchy. As modern processors include multiple levels of caches, and as cache associatively increases, it is important to revisit the effectiveness of these factors on access time and dissipated power. [1]

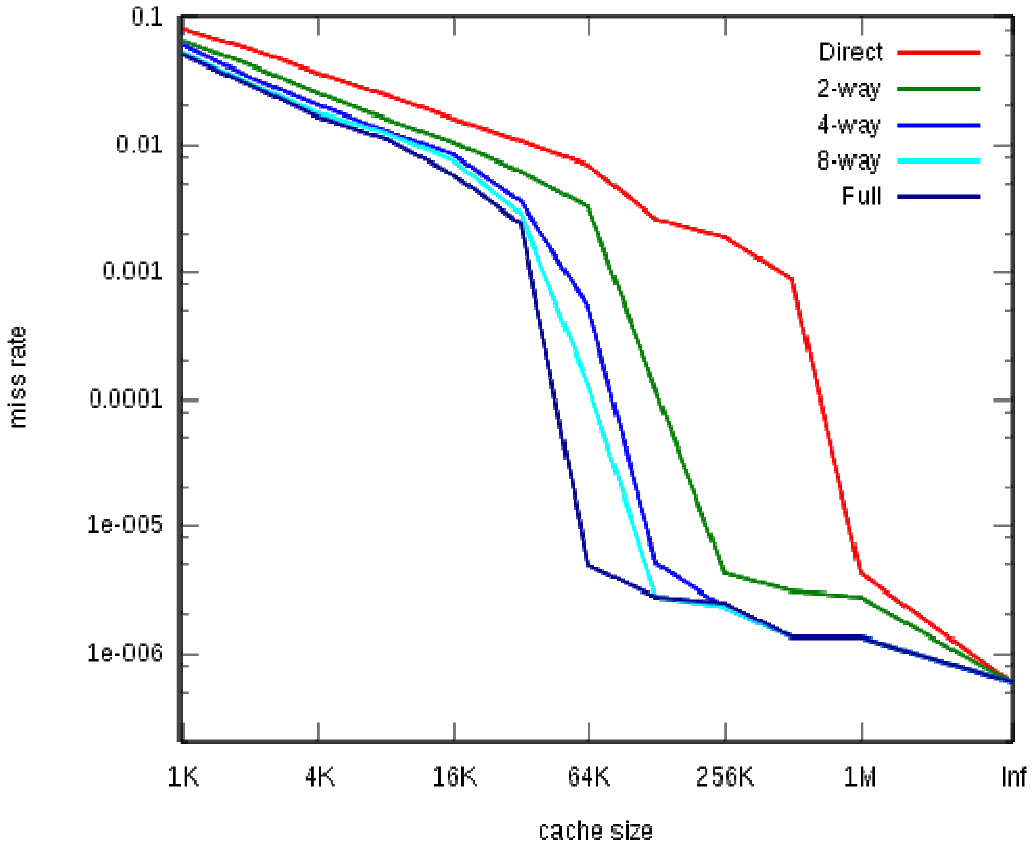


Figure 1: the relation between the miss rate and the cache size [2]

Memory hierarchy design is based on three important principles:

- Make the common case.
- Principle of locality.
- Smaller is faster.

Moving farther away from the CPU, the memory in the level becomes larger and slower. The above principles suggest that to keep recently accessed items in the fastest memory. Because the smaller memories are more expensive and faster, we want to use smaller memories to try to hold the most recently accessed items close to the CPU and successively larger (and slower, and less expensive) memories as we move away from the CPU. This type of organization is called a memory hierarchy.

To evaluate the effectiveness of the memory hierarchy we can use the formula:

Memory_stall_cycles = IC * Mem_Refs * Miss_Rate * Miss_Penalty

where IC = Instruction count
Mem_Refs = Memory References per Instruction
Miss_Rate = the fraction of accesses that are not in the cache
Miss_Penalty = the additional time to service the miss

Simulator

Many simulation tools for computer architecture researches and education purposes have been developed previously. Each tool has its own advantages and disadvantages in complexity, efficiency, and value as an educational aid.

- Simplescalar 3.0
  The SimpleScalar simulator, is efficient and includes support for cache and superscalar simulation, and uses an appropriate instruction set derived from the MIPS ISA. The SimpleScalar toolset provides an infrastructure for simulation and architectural modeling. The toolset can model a variety of platforms ranging from simple unpipelined processors to detailed dynamically scheduled microarchitectures with multiple-level memory hierarchies. For users with more individual needs, SimpleScalar offers a documented and well-structured design, which simplifies extending the toolset to accomplish most architectural modeling tasks.
  SimpleScalar simulators reproduce computing device operations by executing all program instructions using an interpreter. The toolset's instruction interpreters support several popular instruction sets, including Alpha, Power PC, x86, and ARM. [3][4]
- CACTI 4.1
  CACTI is an integrated cache and memory access time, cycle time, area, leakage, and dynamic power model. By integrating all these models together, users can have confidence that tradeoffs between time, power, and area are all based on the same assumptions and, hence, are mutually consistent. CACTI is intended for use by computer architects to better understand the performance tradeoffs inherent in memory system organizations. The name CACTI was derived from the phrase "Cache Access and Cycle Timing Information". Many cache evaluations employ the CACTI cache access modeling tool to estimate delay, power, and area for a given cache size. [7]
  Version 4 was developed by David Tarjan, Shyamkumar Thoziyoor and Norm Jouppi in 2006 and its primary enhancement was the addition of a leakage power model and a web interface. Version 4 also added support for modeling sequential and fast cache access modes amongst other enhancements. [6]

Workloads

The subset of SPEC CPU2000 benchmarks that I used in my study were compiled to produce PISA binaries which I downloaded them from [5], and the inputs for these binaries downloaded from[8].

I used two floating SPEC CPU2000: ammp and equake. Another two as intger SPEC CPU2000:bzip2 and parser. The details about these SPEC are listed in table 1.

Table 1: benchmarks details

SPEC CPU2000	Description
ammp	The benchmark runs molecular dynamics (i.e. solves the ODE defined by Newton's equations for the motions of the atoms in the system) on a protein-inhibitor complex which is embedded in water.[9]
parser	The Link Grammar Parser is a syntactic parser of English, based on link grammar, an original theory of English syntax. Given a sentence, the system assigns to it a syntactic structure, which consists of set of labeled links connecting pairs of words. The parser has a dictionary of about 60000 word forms. It has coverage of a wide variety of syntactic constructions, including many rare and idiomatic ones. The parser is robust; it is able to skip over portions of the sentence that it cannot understand, and assign some structure to the rest of the sentence. It is able to handle unknown vocabulary, and make intelligent guesses from context about the syntactic categories of unknown words. [10]
--Bzip2	256.bzip2 is based on Julian Seward's bzip2 version 0.1. The only difference between bzip2 0.1 and 256.bzip2 is that SPEC's version of bzip2 performs no file I/O other than reading the input. All compression and decompression happens entirely in memory. This is to help isolate the work done to only the CPU and memory subsystem.[11]
equake	The program simulates the propagation of elastic waves in large, highly heterogeneous valleys, such as California's San Fernando Valley, or the Greater Los Angeles Basin. The goal is to recover the time history of the ground motion everywhere within the valley due to a specific seismic event. Computations are performed on an unstructured mesh that locally resolves wavelengths, using a finite element method. [12]

Results

I divided my expermints into two parts, in the first I examined the effect of changing the L1 data cache over the misses rate, access time and power consumption using one of my benchmarks which is (ammp).

In the second part, I examined the effect of changing the associtivity of the L1 data cache over the misses rate, access time, and power consumption using all the four benchmarks listed in workload section.

Excaminiation procedure of the first part

To caclculate the misses rate I used the sim-cache tool which is a part of the SimpleScalar, changing the cache parameters and re-run sim-cache on ammp SPEC 2000 benchmarks, then I ran the CACTI 4.1 to measure the access time and the power.

1. In simplescalar(sim-cache), I changed cache configuration at command line

The cache config parameter <config> has the following format:

<name>:<nsets>:<bsize>:<assoc>:<repl>

<name> - name of the cache being defined

<nsets> - number of sets in the cache

<bsize> - block size of the cache

<assoc> - associativity of the cache

<repl> - block replacement strategy, 'l'-LRU, 'f'-FIFO, 'r'-random

-cache:il1 il1:1024:32:1:l to configure L1 I-cache to 32K size, with 32 byte block size and direct-mapped cache.

-cache:dl1 dl1:512:32:2:l to configure L1 I-cache to 32K size, with 32 byte block size and 2-way set-associative cache.

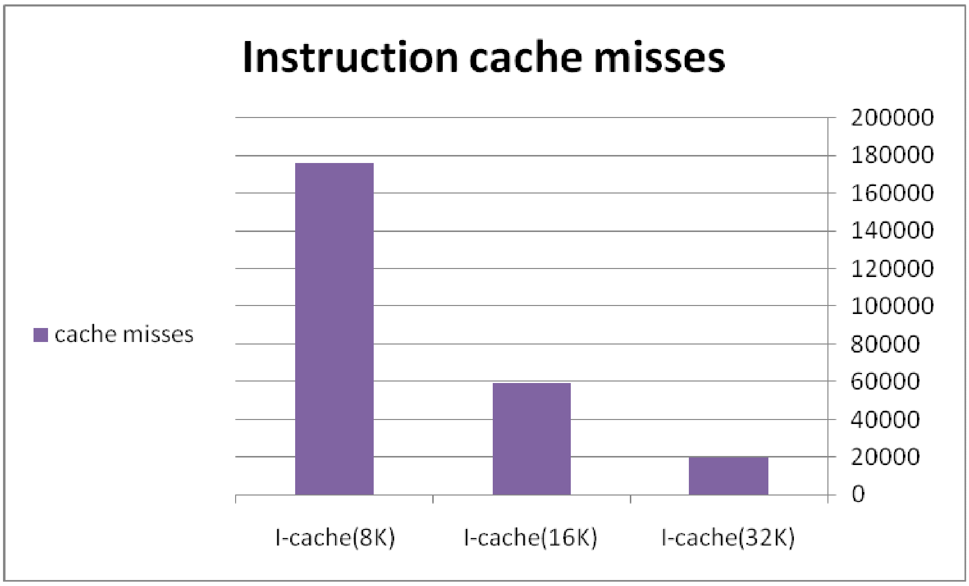
The following results was obtained:

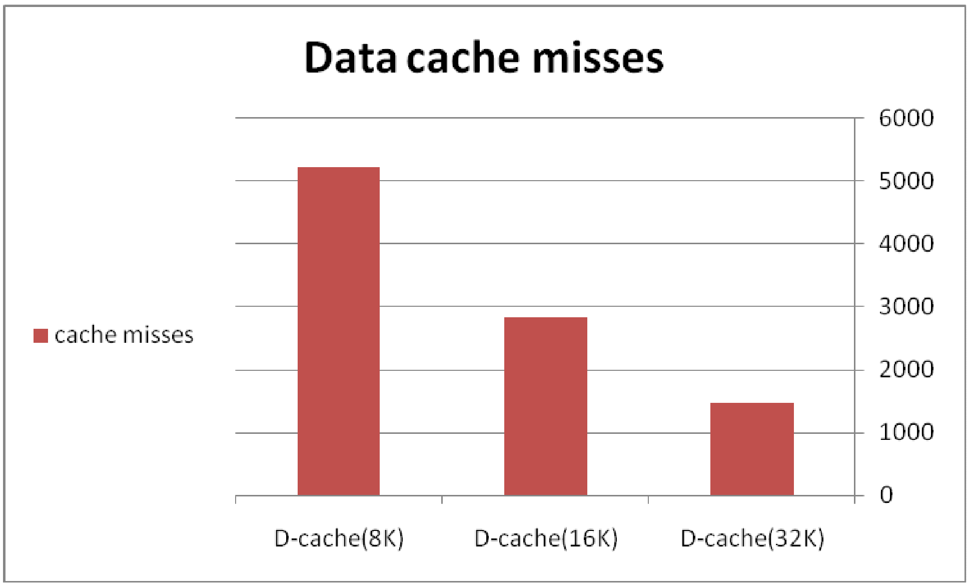
benchmarks/ammp	cache accesses	cache hits	cache misses
I-cache(32K)	800000	779825	20175
I-cache(16K)	800000	740511	59489
I-cache(8K)	800000	623981	176019
benchmarks/ammp	cache accesses	cache hits	cache misses
D-cache(32K)	262846	261377	1468
D-cache(16K)	262846	260014	2832
D-cache(8K)	262846	257630	5216

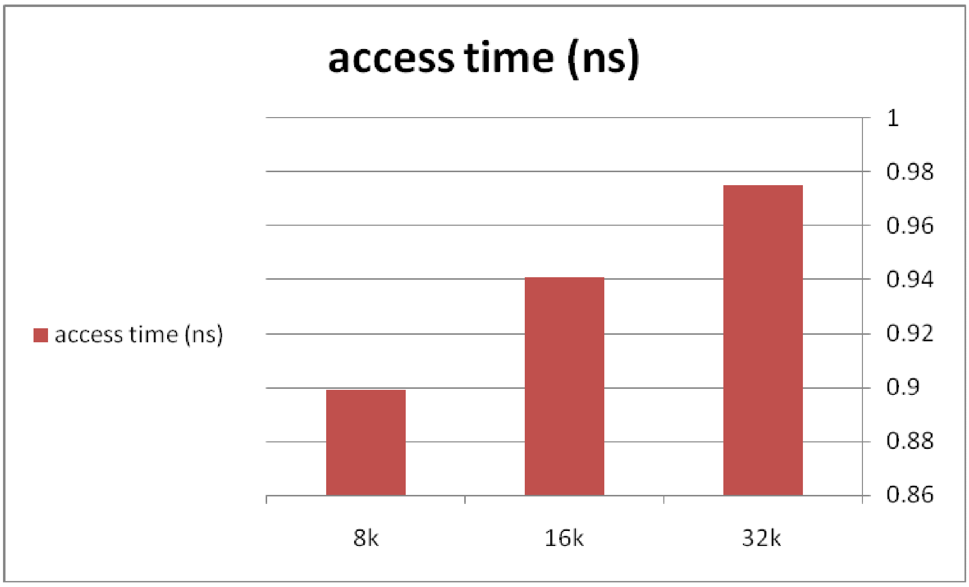
To test the access time, the total dynamic read power and the leakage read/write power I used CACTI 4.1.

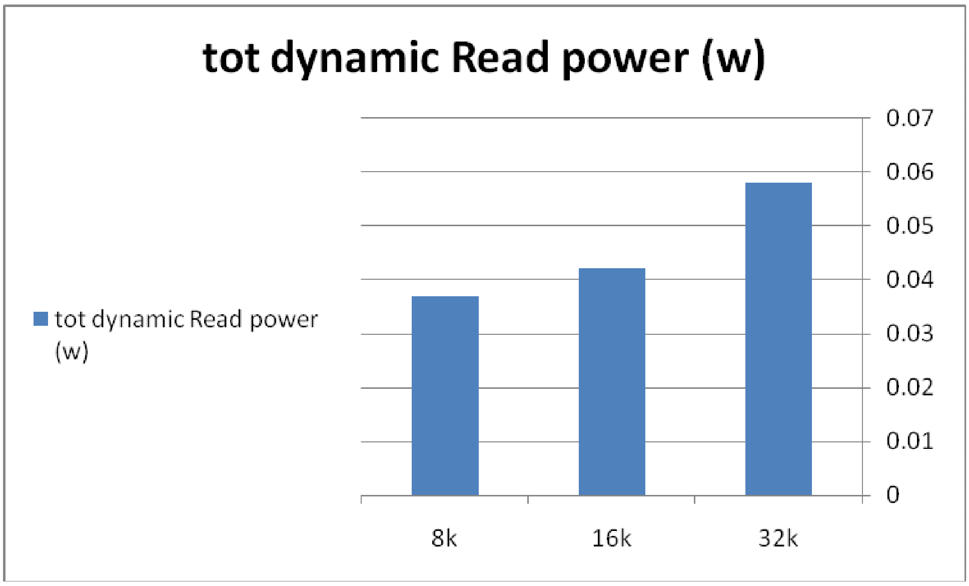
The command line cacti configuration is: cacti C B A TECH Nsubbanks, where C for cache size, B for cache block size, A for associativity, TECH for the transistor size, and Nsubbanks for the number of memory banks.

For example: type ./cacti 32768 32 1 0.13 1 to simulate 32K cache with 32 byte, direct-mapped cache.



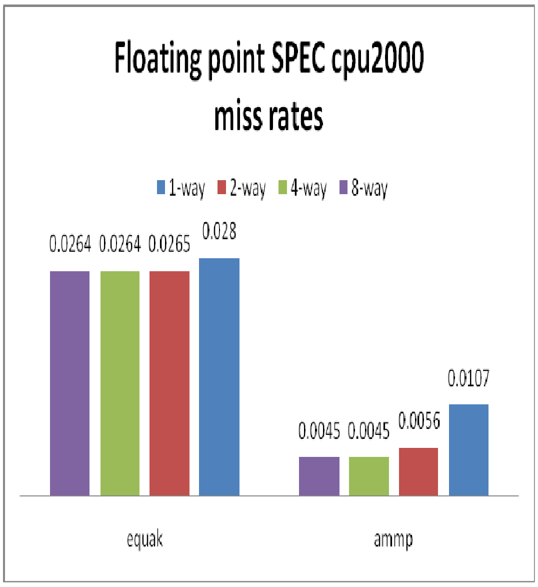


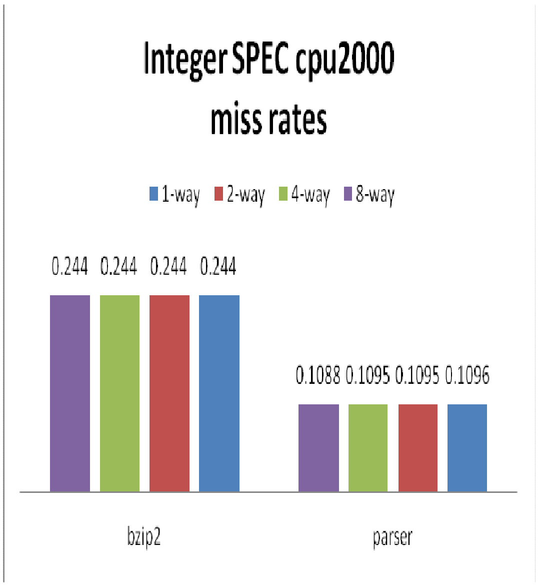


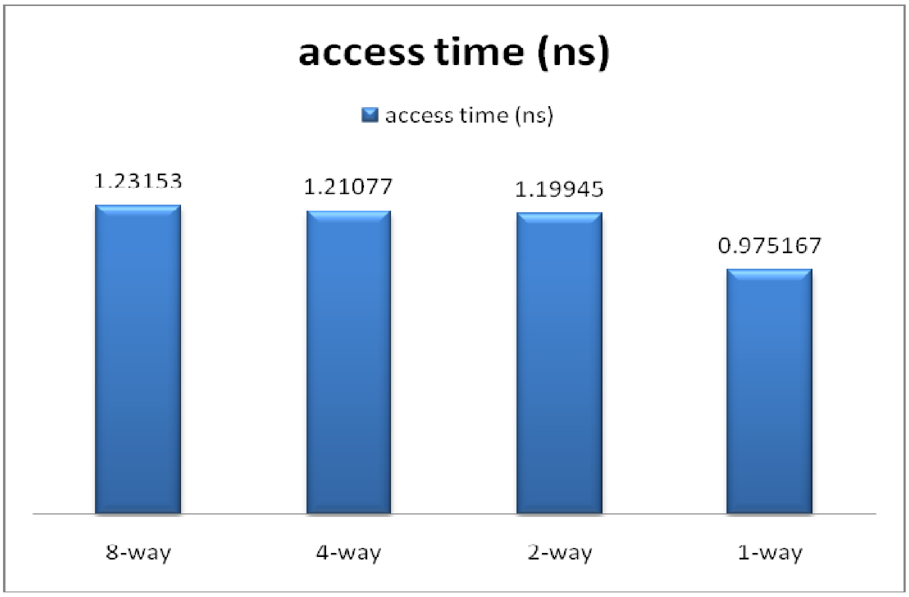


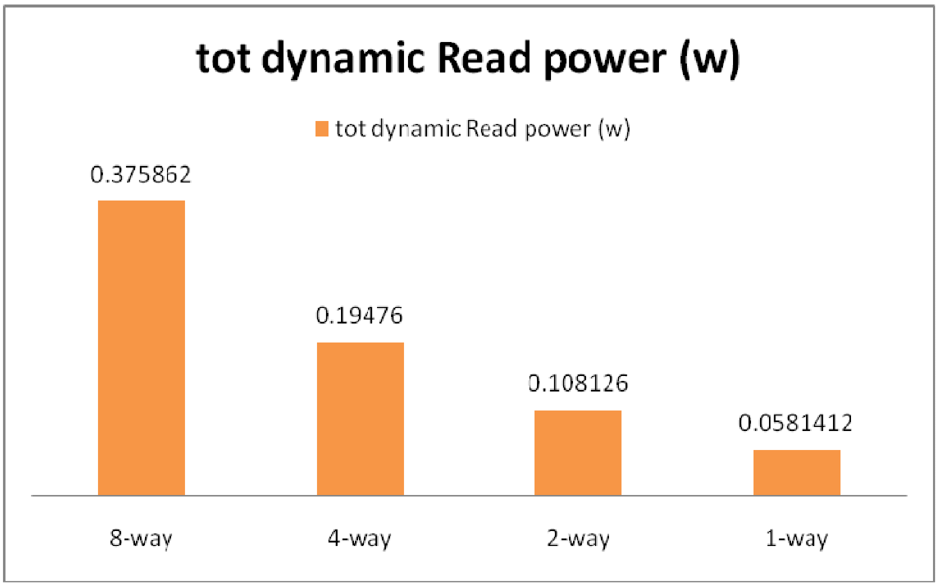
Excaminiation procedure of the second part

To caclculate the misses rate I used the sim-cache tool which is a part of the SimpleScalar, changing the cache parameters to configure it as direct mapped(one-way), 2-way, 4-way, and 8-way set associtive. Then re-run sim-cache on my four SPEC 2000 benchmarks, then I ran the CACTI 4.1 to measure the access time and the power for each configration.


Floating point SPEC cpu2000 miss rates
1-way
2-way
4-way
8-way
0.0264
0.0264
0.0265
0.028
0.0045
0.0045
0.0056
0.0107
equak
ammp


Integer SPEC cpu2000 miss rates
1-way
2-way
4-way
8-way
0.244
0.244
0.244
0.244
0.1088
0.1095
0.1095
0.1096
bzip2
parser


access time (ns)
access time (ns)
1.23153
1.21077
1.19945
0.975167
8-way
4-way
2-way
1-way


tot dynamic Read power (w)
tot dynamic Read power (w)
0.375862
0.19476
0.108126
0.0581412
8-way
4-way
2-way
1-way

Conclusions

Nowadays, it is really difficult to build a memory system to keep on track with faster CPUs. It is the principle of locality that gives a chance to overcome the long latency of memory access. As the gap between processor speed and memory speed is expanding, it is very important to consider tradeoffs for power and performance of cache memories. The simulation in this report demonstrating cache design tradeoff. Data examples calculated using SimpleScalar and CACTI demonstrate these tradeoff. For future work, more accurate performance prediction will be possible by combining multi-level cache simulator with power consideration to calculate miss penalty effectively. In addition to performance and cost estimation, it is important to estimate the complete design including other factors such as power consumption in cache memory design.

References

[1] John L. Hennessy, David A. Patterson. "Computer Architecture: A Quantitative Approach", 4th Edition .

[2] This link from where I copy a figure about the the relation between the miss rate and the cache size (date: 18-Jan-2010)
http://en.wikipedia.org/wiki/CPU_cache .

[3] Todd Austen, Eric Larson, Dan Ernst. SimpleScalar: an infrastructure for computer system modeling. February 2002 IEEE.

[4] This link from where I retrieved some information about SimpleScalar simulator (date: 18-Jan-2010)
http://www.simplescalar.com.

[5] This link from where I downloaded some PISA binaries of SPEC cpu2000 benchmark (date: 18-Jan-2010)
http://www.simplescalar.com/benchmarks.html.

[6] This link from where I retrieved some information about the HP CACTI 4.1 simulator (date: 18-Jan-2010)
http://www.hpl.hp.com/research/cacti/

[7] David Tarjan, Shyamkumar Thoziyoor, Norman P. Jouppi. CACTI 4.0 . HP Laboratories Palo Alto. HPL-2006-86.

[8] This link from where I downloaded the required inputs for my SPEC cpu2000 benchmark binaries(date: 18-Jan-2010)
http://students.cs.tamu.edu/baiksong/teaching/cpsc614/spec2000args.tgz

[9] This link from where I retrieved some information about the floating point ammp SPEC cpu2000 benchmark (date: 18-Jan-2010)
http://www.spec.org/cpu2000/CFP2000/188.ammp/docs/188.ammp.html

[10] This link from where I retrieved some information about the integer parser SPEC cpu2000 benchmark (date: 18-Jan-2010)
http://www.spec.org/cpu2000/CINT2000/197.parser/docs/197.parser.html

[11] This link from where I retrieved some information about the integer bzip2 SPEC cpu2000 benchmark (date: 18-Jan-2010)
http://www.spec.org/cpu2000/CINT2000/256.bzip2/docs/256.bzip2.html

[12] This link from where I retrieved some information about the floating point equake SPEC cpu2000 benchmark (date: 18-Jan-2010)
http://www.spec.org/cpu2000/CFP2000/183.equake/docs/183.equake.html