Objective:

The main aim of this project is to fine-tune the cache hierarchy of an Alpha microprocessor for 4 individual benchmarks namely GCC, ANAGRAM, GO. The associativity, block size and replacement policies are varied for all possible combinations. A cost function is formulated and is plotted against the CPI from which an optimum CPI is determined.

PART 1 :

This stage involves setting up environment for running the benchmarks. The simplescalar tool is installed and the various benchmarks are installed into the virtual box and tested.

PART 2:

In this part, we will calculate the CPI for the four individual benchmarks. Our baseline configuration is 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.

Formula Used :

Separate L1 Cache & Unified L2 cache:

CPI = CPI ideal + 5\* (L1InsMissRate \* (L1Ins Access/Total Ins) + L1DataMissRate \* (L1

Data access/total Ins)) + 40 \* (L2MissRate \* (L2 access/Total Ins) )

The following are the results obtained:

GCC Benchmark:

{cs6304-32:~/Project\_1/simplesim-3.0} ./sim-cache -cache:dl1 dl1:512:64:2:f -cache:il1 il1:512:64:2:f -cache:il2 dl2 -cache:dl2 ul2:16384:64:1:f -tlb:itlb none -tlb:dtlb none benchmarks/cc1.alpha -O benchmarks/1stmt.i

sim: \*\* starting functional simulation w/ caches \*\*

warning: partially supported sigaction() call...

label\_rtx emit\_jump expand\_label expand\_goto expand\_goto\_internal expand\_fixup fixup\_gotos expand\_asm expand\_asm\_operands expand\_expr\_stmt clear\_last\_expr expand\_start\_stmt\_expr expand\_end\_stmt\_expr expand\_start\_cond expand\_end\_cond expand\_start\_else expand\_end\_else expand\_start\_loop expand\_start\_loop\_continue\_elsewhere expand\_loop\_continue\_here expand\_end\_loop expand\_continue\_loop expand\_exit\_loop expand\_exit\_loop\_if\_false expand\_exit\_something expand\_null\_return expand\_null\_return\_1 expand\_return drop\_through\_at\_end\_p tail\_recursion\_args expand\_start\_bindings use\_variable use\_variable\_after expand\_end\_bindings expand\_decl expand\_decl\_init expand\_anon\_union\_decl expand\_cleanups fixup\_cleanups move\_cleanups\_up expand\_start\_case expand\_start\_case\_dummy expand\_end\_case\_dummy pushcase pushcase\_range check\_for\_full\_enumeration\_handling expand\_end\_case do\_jump\_if\_equal group\_case\_nodes balance\_case\_nodes node\_has\_low\_bound node\_has\_high\_bound node\_is\_bounded emit\_jump\_if\_reachable emit\_case\_nodes get\_frame\_size assign\_stack\_local put\_var\_into\_stack fixup\_var\_refs fixup\_var\_refs\_insns fixup\_var\_refs\_1 fixup\_memory\_subreg walk\_fixup\_memory\_subreg fixup\_stack\_1 optimize\_bit\_field max\_parm\_reg\_num get\_first\_nonparm\_insn parm\_stack\_loc assign\_parms get\_structure\_value\_addr uninitialized\_vars\_warning setjmp\_protect expand\_function\_start expand\_function\_end

time in parse: 0.000000

time in integration: 0.000000

time in jump: 0.000000

time in cse: 0.000000

time in loop: 0.000000

time in cse2: 0.000000

time in flow: 0.000000

time in combine: 0.000000

time in sched: 0.000000

time in local-alloc: 0.000000

time in global-alloc: 0.000000

time in sched2: 0.000000

time in dbranch: 0.000000

time in shorten-branch: 0.000000

time in stack-reg: 0.000000

time in final: 0.000000

time in varconst: 0.000000

time in symout: 0.000000

time in dump: 0.000000

warning: partially supported sigprocmask() call...

sim: \*\* simulation statistics \*\*

sim\_num\_insn 337330187 # total number of instructions executed

sim\_num\_refs 121894242 # total number of loads and stores executed

sim\_elapsed\_time 36 # total simulation time in seconds

sim\_inst\_rate 9370282.9722 # simulation speed (in insts/sec)

il1.accesses 337330187 # total number of accesses

il1.hits 335742191 # total number of hits

il1.misses 1587996 # total number of misses

il1.replacements 1586972 # total number of replacements

il1.writebacks 0 # total number of writebacks

il1.invalidations 0 # total number of invalidations

il1.miss\_rate 0.0047 # miss rate (i.e., misses/ref)

il1.repl\_rate 0.0047 # replacement rate (i.e., repls/ref)

il1.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl1.accesses 124104359 # total number of accesses

dl1.hits 122789983 # total number of hits

dl1.misses 1314376 # total number of misses

dl1.replacements 1313352 # total number of replacements

dl1.writebacks 416880 # total number of writebacks

dl1.invalidations 0 # total number of invalidations

dl1.miss\_rate 0.0106 # miss rate (i.e., misses/ref)

dl1.repl\_rate 0.0106 # replacement rate (i.e., repls/ref)

dl1.wb\_rate 0.0034 # writeback rate (i.e., wrbks/ref)

dl1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ul2.accesses 3319252 # total number of accesses

ul2.hits 2892370 # total number of hits

ul2.misses 426882 # total number of misses

ul2.replacements 410498 # total number of replacements

ul2.writebacks 138069 # total number of writebacks

ul2.invalidations 0 # total number of invalidations

ul2.miss\_rate 0.1286 # miss rate (i.e., misses/ref)

ul2.repl\_rate 0.1237 # replacement rate (i.e., repls/ref)

ul2.wb\_rate 0.0416 # writeback rate (i.e., wrbks/ref)

ul2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ld\_text\_base 0x0120000000 # program text (code) segment base

ld\_text\_size 1564672 # program text (code) size in bytes

ld\_data\_base 0x0140000000 # program initialized data segment base

ld\_data\_size 277104 # program init'ed `.data' and uninit'ed `.bss' size in bytes

ld\_stack\_base 0x011ff9b000 # program stack segment base (highest address in stack)

ld\_stack\_size 16384 # program initial stack size

ld\_prog\_entry 0x0120025f70 # program entry point (initial PC)

ld\_environ\_base 0x011ff97000 # program environment base address address

ld\_target\_big\_endian 0 # target executable endian-ness, non-zero if big endian

mem.page\_count 785 # total number of pages allocated

mem.page\_mem 6280k # total size of memory pages allocated

mem.ptab\_misses 613823 # total first level page table misses

mem.ptab\_accesses 926309129 # total page table accesses

mem.ptab\_miss\_rate 0.0007 # first level page table miss rate

CPI = 1.0936145

ANAGRAM Benchmark:

{cs6304-32:~/Project\_1/simplesim-3.0} ./sim-cache -cache:dl1 dl1:512:64:2:f -cache:il1 il1:512:64:2:f -cache:il2 dl2 -cache:dl2 ul2:16384:64:1:f -tlb:itlb none -tlb:dtlb none benchmarks/anagram.alpha benchmarks/words < benchmarks/anagram.in > OUT

sim: \*\* simulation statistics \*\*

sim\_num\_insn 25593186 # total number of instructions executed

sim\_num\_refs 9031728 # total number of loads and stores executed

sim\_elapsed\_time 3 # total simulation time in seconds

sim\_inst\_rate 8531062.0000 # simulation speed (in insts/sec)

il1.accesses 25593186 # total number of accesses

il1.hits 25592691 # total number of hits

il1.misses 495 # total number of misses

il1.replacements 16 # total number of replacements

il1.writebacks 0 # total number of writebacks

il1.invalidations 0 # total number of invalidations

il1.miss\_rate 0.0000 # miss rate (i.e., misses/ref)

il1.repl\_rate 0.0000 # replacement rate (i.e., repls/ref)

il1.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl1.accesses 11153897 # total number of accesses

dl1.hits 11099651 # total number of hits

dl1.misses 54246 # total number of misses

dl1.replacements 53222 # total number of replacements

dl1.writebacks 37897 # total number of writebacks

dl1.invalidations 0 # total number of invalidations

dl1.miss\_rate 0.0049 # miss rate (i.e., misses/ref)

dl1.repl\_rate 0.0048 # replacement rate (i.e., repls/ref)

dl1.wb\_rate 0.0034 # writeback rate (i.e., wrbks/ref)

dl1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ul2.accesses 92638 # total number of accesses

ul2.hits 63100 # total number of hits

ul2.misses 29538 # total number of misses

ul2.replacements 13154 # total number of replacements

ul2.writebacks 12741 # total number of writebacks

ul2.invalidations 0 # total number of invalidations

ul2.miss\_rate 0.3189 # miss rate (i.e., misses/ref)

ul2.repl\_rate 0.1420 # replacement rate (i.e., repls/ref)

ul2.wb\_rate 0.1375 # writeback rate (i.e., wrbks/ref)

ul2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ld\_text\_base 0x0120000000 # program text (code) segment base

ld\_text\_size 106496 # program text (code) size in bytes

ld\_data\_base 0x0140000000 # program initialized data segment base

ld\_data\_size 71264 # program init'ed `.data' and uninit'ed `.bss' size in bytes

ld\_stack\_base 0x011ff9b000 # program stack segment base (highest address in stack)

ld\_stack\_size 16384 # program initial stack size

ld\_prog\_entry 0x01200059c0 # program entry point (initial PC)

ld\_environ\_base 0x011ff97000 # program environment base address address

ld\_target\_big\_endian 0 # target executable endian-ness, non-zero if big endian

mem.page\_count 182 # total number of pages allocated

mem.page\_mem 1456k # total size of memory pages allocated

mem.ptab\_misses 454294 # total first level page table misses

mem.ptab\_accesses 73719151 # total page table accesses

mem.ptab\_miss\_rate 0.0062 # first level page table miss rate

CPI = 1.05684953

GO Benchmark:

{cs6304-32:~/Project\_1/simplesim-3.0} ./sim-cache -cache:dl1 dl1:512:64:2:f -cache:il1 il1:512:64:2:f -cache:il2 dl2 -cache:dl2 ul2:16384:64:1:f -tlb:itlb none -tlb:dtlb none benchmarks/go.alpha 50 9 benchmarks/2stone9.in > OUT

sim: \*\* simulation statistics \*\*

sim\_num\_insn 545812708 # total number of instructions executed

sim\_num\_refs 211690635 # total number of loads and stores executed

sim\_elapsed\_time 57 # total simulation time in seconds

sim\_inst\_rate 9575661.5439 # simulation speed (in insts/sec)

il1.accesses 545812708 # total number of accesses

il1.hits 545098009 # total number of hits

il1.misses 714699 # total number of misses

il1.replacements 713675 # total number of replacements

il1.writebacks 0 # total number of writebacks

il1.invalidations 0 # total number of invalidations

il1.miss\_rate 0.0013 # miss rate (i.e., misses/ref)

il1.repl\_rate 0.0013 # replacement rate (i.e., repls/ref)

il1.wb\_rate 0.0000 # writeback rate (i.e., wrbks/ref)

il1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

dl1.accesses 213788508 # total number of accesses

dl1.hits 213579212 # total number of hits

dl1.misses 209296 # total number of misses

dl1.replacements 208272 # total number of replacements

dl1.writebacks 95533 # total number of writebacks

dl1.invalidations 0 # total number of invalidations

dl1.miss\_rate 0.0010 # miss rate (i.e., misses/ref)

dl1.repl\_rate 0.0010 # replacement rate (i.e., repls/ref)

dl1.wb\_rate 0.0004 # writeback rate (i.e., wrbks/ref)

dl1.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ul2.accesses 1019528 # total number of accesses

ul2.hits 927360 # total number of hits

ul2.misses 92168 # total number of misses

ul2.replacements 75784 # total number of replacements

ul2.writebacks 25726 # total number of writebacks

ul2.invalidations 0 # total number of invalidations

ul2.miss\_rate 0.0904 # miss rate (i.e., misses/ref)

ul2.repl\_rate 0.0743 # replacement rate (i.e., repls/ref)

ul2.wb\_rate 0.0252 # writeback rate (i.e., wrbks/ref)

ul2.inv\_rate 0.0000 # invalidation rate (i.e., invs/ref)

ld\_text\_base 0x0120000000 # program text (code) segment base

ld\_text\_size 376832 # program text (code) size in bytes

ld\_data\_base 0x0140000000 # program initialized data segment base

ld\_data\_size 612032 # program init'ed `.data' and uninit'ed `.bss' size in bytes

ld\_stack\_base 0x011ff9b000 # program stack segment base (highest address in stack)

ld\_stack\_size 16384 # program initial stack size

ld\_prog\_entry 0x0120007bb0 # program entry point (initial PC)

ld\_environ\_base 0x011ff97000 # program environment base address address

ld\_target\_big\_endian 0 # target executable endian-ness, non-zero if big endian

mem.page\_count 246 # total number of pages allocated

mem.page\_mem 1968k # total size of memory pages allocated

mem.ptab\_misses 1656511 # total first level page table misses

mem.ptab\_accesses 1520170656 # total page table accesses

mem.ptab\_miss\_rate 0.0011 # first level page table miss rate

CPI = 1.01521279

PART 3 :

Design Choices:

All types of associativity is taken into consideration namely Direct, fully, 2, 4 , 8

Replacement Policies are l, f, r

Block sizes are 64 KB and 32 KB

Formulae :

a) Separate L1 cache & Separate L2 Cache:

CPI = CPI = CPI ideal + 5\* ( L1InsMissRate \* (L1 Ins Access/Total Ins) +

L1DataMissRate \* (L1 Data access/Total Ins)) + 40 \* ( L2InsMissRate \* ( L2 Ins

access/Total Ins) +L2data MissRate\*(L2 Data Access/Total Ins))

b) Separate L1 Cache & Unified L2 cache:

CPI = CPI ideal + 5\* (L1InsMissRate \* (L1 Ins Access/Total Ins) + L1DataMissRate \* (L1

Data access/total Ins)) + 40 \* (L2MissRate \* (L2 access/Total Ins) )

c) Unified L1 Cache & Unified L2 Cache:

CPI = CPI = CPI ideal + 5\* (L1MissRate \* (L1 Access/Total Ins) ) + 40 \* ( L2MissRate \* ( L2 access/Total Ins)

GCC Benchmark :

Separate L1 cache and Separate L2 cache

Separate L1 cache and Unified L2 cache

Unified L1 and Unified L2 cache

ANAGRAM Benchmark:

Separate L1 cache and Separate L2 cache

Separate L1 cache and Unified L2 cache:

Unified L1 and Unified L2 cache :

GO Benchmark:

Separate L1 cache and Separate L2 cache:

Separate L1 cache and Unified L2 cache:

Unified L1 cache and Unified L2 cache:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ***BENCHMARK*** | ***TYPE*** | ***REP POLICY*** | ***BLOCK SIZE*** | ***ASSOCIATIVITY OF L1*** | ***ASSOCIATIVITY OF L2*** | ***CPI*** |
| GCC | Sep L1 & Sep L2 | F | 32 | Full(4096) | Full(32768) | 1.070019525 |
| Sep L1 & Unified L2 | F | 32 | Full(4096) | Full(32768) | 1.030414411 |
| Unified L1 & L2 | F | 32 | Full(4096) | Full(32768) | 1.02370471 |
| ANAGRAM | Sep L1 & Sep L2 | F | 32 | 8 | 1 | 1.05649295 |
| Sep L1 & Unified L2 | F | 32 | 4 | 1 | 1.056531875 |
| Unified L1 & L2 | F | 32 | 4 | 1 | 1.057093017 |
| GO | Sep L1 & Sep L2 | F | 32 | Full(4096) | 2 | 1.00337697 |
| Sep L1 & Unified L2 | F | 32 | Full(4096) | 1 | 1.003491394 |
| Unified L1 & L2 | F | 64 | Full(2048) | 2 | 1.00294521 |

PART 4: DEFINING COST FUNCTION

Cost of a cache is an important factor while designing a memory hierarchy. Though lower CPI is a desired criterion, cost plays a vital role in determining the optimal CPI. A tradeoff of CPI should be made to acquire lower cost which is the ultimate aim of manufacturers. The following are the parameters considered while determining the cost function:

* Cache size
* Associativity
* Cache splitting
* Replacement policy

Cache Size: This accounts for the major percentage of cost. As the size of the cost increases the cost also increases. We assume a that the size of the cache accounts for 60% of the total 100% cost. L1 caches are of higher cost than L2 caches because they are faster and operate at the CPU clock cycle. L2 caches are larger than L1 caches but are slower than L2 caches. But comparatively cost of L1 caches is higher than L2 caches.

L1 – 35

L2 – 25

Associativity : The next important factor which plays a vital role in determining the cost of a cache is associativity. As the associativity increases the cost also increases because of hardware complexity. Lets assume a weightage of 20% for associativity

1 – 1.25

2 – 2.5

4 – 5

8 – 10

Fully – 20

Cache Splitting: There are three main types of cache hierarchy

L1 separate- L2 separate

L1 separate-L2 unified

L1 unified-L2 unified

Separate caches for instruction and data require more hardware which eventually increases the cost. L1 caches are expensive than L2 because they are much faster and operate at CPU clock cycles.

L1 separate – 3.75

L2 separate –3.25

L1 unified – 2.5

L2 unified – 2

Replacement Policy:

There are three types of replacement policies namely, LRU , FIFO and Random. The hardware required for LRU is much complex and costlier than the other two replacement policies with random policy having the lowest cost. An overall weightage of 8.5 % is assumed.

LRU – 2.95

FIFO –2.85

Random –2.7

Thus the cost function can be defined as follows

Cost= w(CachesizeL1) +w(CachesizeL2) + w(L1unified+L1separate+L2unified+L2separate)+

w(Asociativity\_1 or Asociativity\_2 or Asociativity\_4 or Asociativity\_8 or fully associativity)+

w(replacement\_LRU or replacement\_FIFO or replacement\_random)