***Computer Architecture***

***Performance Report***



***Group Members:***

Abid Waqar (i160229)

M Nauraiz Mushtaq (i160106)

Khizer Shabir (i160294)

***Section CS\_B***

**Submitted To:**

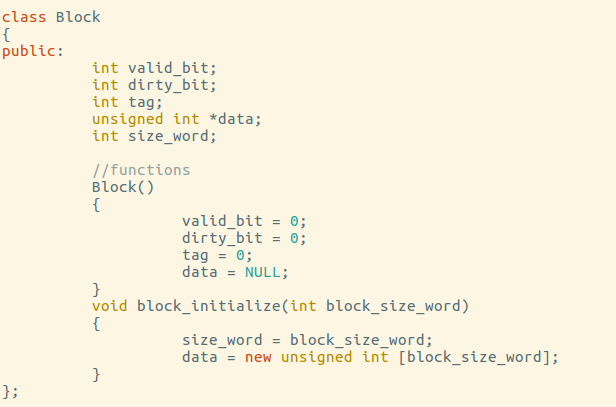
Mrs. Noor-ul-Ain

***Cache Simulator***

***Trace Driven Simulator***

The simulator is known as trace driven simulator because it inputs the traces of events. On depending on those traces cache simulate the data inside in and out.

In order to make cache simulator and become familiar with cache. We use some Internet tools like on line cache simulators to understand it’s working. After words we developed a structure for cache simulator. Which consist of fallowing classes

* **Block**
* **Set**
* **Cache**

***Block:***

*Block is a unit that holds the data of cache inside itself.*

*Valid bit signifies that data is placed in this block. Tag represents the middle address of memory location from where data had been placed in to the cache.*

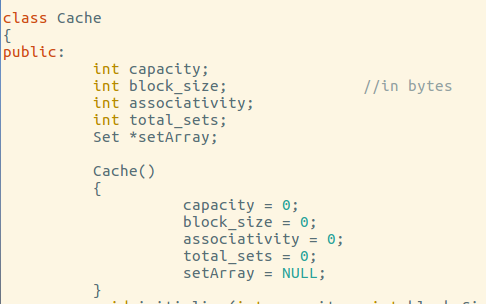
*Dirty Bit is use for verification*

*That weather data change occur in cache also persists in memory.*

***Set:***

*A cache has Number of sets and a set has number of Blocks depending upon the type of block that each set can hold. An array of block represents numbers of blocks and lru is used for holding recently used instruction, which is use at replacement time to replace instruction. Number of block being stored for iteration.*

***Cache***

*A cache has number of sets and each set has number of block and block can hold number of words. Cache has its own capacity, block size, associativity and total number of set that it can hold. By using such block size, associativity, and capacity we can calculate the number of tag, index, and offset bits. By using these formulas.*

***Formulas:***

* int temp = decimal address;
* temp = temp / 4;
* **Offset** = temp % (block size/4);
* temp = temp / (block size/4);
* **index** = temp % total sets;
* temp = temp / total sets;
* **tag** = temp;

By using fallowing structures we build our cache simulator. And make it N-way associative cache. Also implement with split and unified compatibility.

**Working:**

Cache simulator reads the trace file given by user. And break each address into tag, index, offset and on the basis of index and tag place the data inside cache.

Store policies are write back, write through, write allocate, and write around. These policies are used by store in the case of hit and miss.

LRU is being used by each set in order to place the recently used instruction. If cache is full then it will automatically replace the unused instruction from cache and place the new one. Memory is word address able as it data type is (int) which is bytes. Cache place the number of word inside its data depending upon its block size. If block size is larger than many number of words will be written on the cache. Cache reads data from main memory which is 16MB. And on the other hand capacity also defied the cache size larger the capacity larger will be the cache. Some experiments had been done on trace driven simulator given below.

**Experiments:**

*Performance Evaluation:*

***1) Working Set Characterization:***

Block Size = 4KB

Split Cache

Write Back, Write Allocate

Cache Size = 2, 4, 6, 8, 10, 12……

Associativity =1

**Experiment working and significance.**

As the cache size increase so there is less chance of conflict. Means that Miss rate decreases and also number of memory stalls also decrease. This work as we increase the size of cache so the number of sets also increase as capacity is directly proportional to the no of sets. And cache can store more data so there will be less chance of miss, and memory stalls also decreases. But it become so expensive to increase the size of capacity. On the other hand hit rate increase gradually as the capacity of cache increases. Cause Miss Rate decreases and hit rate increases. If capacity increased then also the set increase and if block size also increase then cache become faster, larger, and costly.

Some test runs gives fallowing results for fallowing three test files.

**Total Instruction Set Size and Data Size for Three sample files.**

**CC.trace**

*if c==4*

Data Set = 511

Instruction Set =511

**if c==8**

Data Set = 511

Instruction Set =511

**Formula for calculating Set Size of Data Cache and instruction cache**

No of Sets=Capacity / (2b \* Associativity).

***Effect of Capacity On the Fully Associated:***

***Spice.trace***

***2)-Impact of Block Size***

**Explain why the hit rate vs. block size plot has the shape that it does. In particular, explain the**

**Relevance of spatial locality to the shape of this curve.**

Smaller Blocks don’t take maximum advantage of special locality. If Blocks are too large then fewer blocks will be available for holding data then more conflict misses takes place. At some instances of graph the data hit rate increase and instruction hit rate also increases but at middle it decrease and at the edge it also increase means that the hit rate of both Data and instruction oscillates between upper hit rate and lower hit rate.

**Optimal Block Size**

For Tex.trace the optimal block size = 128 for data, 512 for instruction

For cc.trace the optimal block size = 32 for Data, 512 for instruction

For spice. Trace the optimal block size = 32 for data, 512 for instruction

**Difference between Graphs of Data and Instructions**

The plots for data and instruction varies as booth's hits are dependent on the block size but the degree of dependencies are different. The instructions hit rates vary to limited extend with block size while data instructions not.

Data Hits depends on the smaller block size but if block size is larger than the chances of conflict increases. And it can’t take maximum advantage of special locality.

***3-Impact of Associativity***

**Shape of Graph**

Higher Associativity requires more hard work but it decrease the number of misses done on cache meanwhile it also decrease the the memory stalls. As associativity increases the number of blocks in the set also increase and the data block also remain same. As the Associativity increases the cure falls as representing the miss rate decreases as the Associativity increases.

**Difference between Data and Instruction Caches Graphs**

Although there is no such difference between degree of their plotting. But the Data read and data write graph get sudden peak at increasing associativity while instruction rises slowly up. The degree of change in the ploting of Data and Instructions is a little bit differ.

***4-Write Allocate, around vs Write Back, Write Through***

***1-Write Back Policy vs. Write Through policy***

|  |  |  |
| --- | --- | --- |
| ./cache –s –c8 –b32 a1 | 69526 | 107054 |
| ./cache –s –c16 –b32 a1 | 36683 | 89063 |
| ./cache –s –c16 –b16 a1 | 45155 | 98364 |
| ./cache –s –c16 –b32 a2 | 16803 | 77019 |
| ./cache –s –c8 –b16 a4 | 33613 | 92956 |

As from given table we conclude that write back policy is better then write through and its performance is better than all other policies.

**2-No s**such Scenario had been detected by our simulator

It is possible in real world but may not in our simulator.

***3-Write Around Policy vs. Write Allocate policy***

|  |  |  |
| --- | --- | --- |
| ./cache –s –c8 –b32 a1 | 69526 | 60692 |
| ./cache –s –c16 –b32 a1 | 36683 | 30256 |
| ./cache –s –c16 –b16 a1 | 45155 | 40203 |
| ./cache –s –c16 –b32 a2 | 16803 | 12355 |

**4**\_NO such Scenario had been detected by our simulator

It is possible in real world but may not in our simulator.

***5- Cache Configuration***

***System Enclosure***

Manufacturer: Alien ware

Case Type: Portable

Version: A11

Serial Number: HVQRDV1

Asset Tag Number: Unknown

***Processor***

Processor Manufacturer: Intel(R) Corporation

Processor Version: Intel(R) Core(TM) i7-3610QM CPU @ 2.30GHz

External Clock: 100 MHz

Maximum Clock Supported: 4000 MHz

Current Clock: 2300 MHz

CPU Socket: Populated

CPU Status: Enabled

Processor Type: Central Processor

Processor Voltage: 0.9 V

Processor Upgrade: Socket rPGA988B

Socket Designation: U3E1

***L1 Cache***

Socket Designation: L1 Cache

Cache State: Enabled

Cache Type: Internal, Data

Cache Scheme: Write-Through

Supported SRAM Type:

Current SRAM Type:

Cache Speed: Unknown

Error Correction Type: Parity

Maximum Cache Size: 32 Kbytes

Installed Cache Size: 32 Kbytes

Cache Associativity: 8-way Set-Associative

***L1 Cache***

Socket Designation: L1 Cache

Cache State: Enabled

Cache Type: Internal, Instruction

Cache Scheme: Write-Through

Supported SRAM Type:

Current SRAM Type:

Cache Speed: Unknown

Error Correction Type: Parity

Maximum Cache Size: 32 Kbyte’s

Installed Cache Size: 32 Kbytes

Cache Associativity: 8-way Set-Associative

***Tests***

*Spice. Trace*

Hit Rate of data read 0.998806

Hit Rate of data write 0.998647

Instruction Hit rate is: 0.998376

*CC.trace*

Hit Rate of data read 0.982591

Hit Rate of data write 0.993846

Instruction hit rate is: 0.990851

*Tex. trace*

Hit Rate of data read 0.999449

Hit Rate of data write 0.998861

Instruction hit rate is: 0.999983

***Capacity Tests on Direct Mapped Cache:***

*Test runs for Direct Mapped and with block Size of 4 and variable capacity.*

Spice.txt

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **4** | **1** | **4** | 0.872866 | 0.8438 | 0.79251 |
| **8** | **1** | **4** | 0.902 | 0.882 | 0.859 |
| **16** | **1** | **4** | 0.953 | 0.928 | 0.909 |
| **32** | **1** | **4** | 0.9675 | 0.954 | 0.956 |
| **64** | **1** | **4** | 0.985 | 0.9622 | 0.976 |

CC.trace

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **4** | **1** | **4** | 0.850 | 0.752 | 0.563 |
| **8** | **1** | **4** | 0.907 | 0.830 | 0.666 |
| **16** | **1** | **4** | 0.938 | 0.870 | 0.776 |
| **32** | **1** | **4** | 0.961 | 0.899 | 0.868 |
| **64** | **1** | **4** | 0.976 | 0.914 | 0.918 |

Tex.trace

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **4** | **1** | **4** | 0.937 | 0.928 | 0.997 |
| **8** | **1** | **4** | 0.940 | 0.928 | 0.99972 |
| **16** | **1** | **4** | 0.958 | 0.928 | 0.99973 |
| **32** | **1** | **4** | 0.968 | 0.928 | 0.99973 |
| **64** | **1** | **4** | 0.969 | 0.9284 | 0.99972 |

***Block Size Test***

*Block Size tests for2\_way associative and variable block size.*

Spice. Trace

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **8** | **2** | **4** | 0.966 | 0.929 | 0.902 |
| **8** | **2** | **8** | 0.968 | 0.9595 | 0.944 |
| **8** | **2** | **32** | 0.975 | 0.975 | 0.975 |
| **8** | **2** | **128** | 0.947 | 0.967 | 0.986 |
| **8** | **2** | **256** | 0.944 | 0.965 | 0.986 |

CC.trace

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **8** | **2** | **4** | 0.925 | 0.847 | 0.710 |
| **8** | **2** | **8** | 0.933 | 0.909 | 0.836 |
| **8** | **2** | **32** | 0.9366 | 0.959 | 0.936 |
| **8** | **2** | **128** | 0.907 | 0.965 | 0.9677 |
| **8** | **2** | **256** | 0.875 | 0.950 | 0.975 |

Tex.trace

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **8** | **2** | **4** | 0.942 | 0.928 | 0.9997 |
| **8** | **2** | **8** | 0.971 | 0.964 | 0.9999 |
| **8** | **2** | **32** | 0.992805 | 0.990 | 0.999951 |
| **8** | **2** | **128** | 0.997 | 0.997 | 0.999978 |
| **8** | **2** | **256** | 0.970 | 0.985 | 0.999983 |

**Associativity Test:**

*Associativity test of cache using different associativity.*

Spice. Trace

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **8** | **1** | **128** | 0.890 | 0.931 | 0.982 |
| **8** | **2** | **128** | 0.947 | 0.967 | 0.986 |
| **8** | **4** | **128** | 0.9811 | 0.9891 | 0.9905 |
| **8** | **8** | **128** | 0.982 | 0.9894 | 0.9905 |
| **8** | **16** | **128** | 0.983 | 0.989 | 0.990927 |

CC. Trace

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **8** | **1** | **128** | 0.867 | 0.939 | 0.964 |
| **8** | **2** | **128** | 0.907 | 0.965 | 0.967 |
| **8** | **4** | **128** | 0.923 | 0.974 | 0.968 |
| **8** | **8** | **128** | 0.930 | 0.9750 | 0.968 |
| **8** | **16** | **128** | 0.9322 | 0.976 | 0.9682 |

Tex.Trace

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **8** | **1** | **128** | 0.972 | 0.981 | 0.999 |
| **8** | **2** | **128** | 0.997 | 0.997 | 0.99978 |
| **8** | **4** | **128** | 0.998 | 0.9977 | 0.999978 |
| **8** | **8** | **128** | 0.998 | 0.997 | 0.999978 |
| **8** | **16** | **128** | 0.9981 | 0.999 | 0.999978 |

**Q1: Fully asssociative**

Spice.trace SPLIT:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **4** | **1** | **2048** | 0.490362 | 0.596261 | 0.961972 |
| **8** | **1** | **4096** | 0.498782 | 0.599357 | 0.9757 |
| **16** | **1** | **8192** | 0.503082 | 0.600739 | 0.981412 |
| **32** | **1** | **16384** | 0.553527 | 0.604948 | 0.985384 |
| **64** | **1** | **32768** | 0.574609 | 0.626574 | 0.987181 |

Cc.trace SPLIT:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **4** | **1** | **2048** | 0.411919 | 0.648404 | 0.968177 |
| **8** | **1** | **4096** | 0.425851 | 0.662881 | 0.975237 |
| **16** | **1** | **8192** | 0.430386 | 0.671805 | 0.980043 |
| **32** | **1** | **16384** | 0.472283 | 0.696845 | 0.984852 |
| **64** | **1** | **32768** | 0.482306 | 0.699314 | 0.988038 |

Tex.trace SPLIT:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** | **Data Write**  **Hit Rate** | **Inst. Read**  **Hit Rate** |
| **4** | **1** | **2048** | 0.542964 | 0.642906 | 0.937496 |
| **8** | **1** | **4096** | 0.542964 | 0.642906 | 0.949994 |
| **16** | **1** | **8192** | 0.542964 | 0.642906 | 0.949994 |
| **32** | **1** | **16384** | 0.542964 | 0.642906 | 0.974995 |
| **64** | **1** | **32768** | 0.542964 | 0.642906 | 0.975005 |

Tex.trace unified:

|  |  |  |  |
| --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** |
| **4** | **1** | **4096** | 0.399139 |
| **32** | **1** | **32768** | 0.417084 |

Spice.trace unified:

|  |  |  |  |
| --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** |
| **4** | **1** | **4096** | 0.550563 |
| **32** | **1** | **32768** | 0.556714 |

Cc.trace unified:

|  |  |  |  |
| --- | --- | --- | --- |
| **Capacity** | **Associativity** | **Block Size** | **Data Reads**  **Hit Rate** |
| **4** | **1** | **4096** | 0.499694 |
| **32** | **1** | **32768** | 0.506926 |