**Assignment I**

**Problem Bank 18**

**Assignment Description:**

The assignment aims to provide deeper understanding of cache by analysing its behaviour using cache implementation of CPU- OS Simulator. The assignment has three parts.

* Part I deals with Cache Memory Management with Direct Mapping
* Part II deals with Cache Memory Management with Associative Mapping
* Part III deals with Cache Memory Management with Set Associative Mapping
* Part IV – Case Study

**Submission:** You will have to submit this documentation file and the name of the file should be GROUP-NUMBER.pdf. For Example, if your group number is 1, then the file name should be GROUP-1.pdf.

* Submit the assignment by **6th August 2023 through canvas only**. File submitted by any means outside CANVAS will not be accepted and marked.
* In case of any issues, please drop an email to the instructor.

**Caution!!!**

Assignments are designed for individual groups which may look similar and you may not notice minor changes in the assignments. Hence, refrain from copying or sharing documents with others. Any evidence of such practice will attract severe penalty.

**Evaluation:**

* The assignment carries 20 marks
* Grading will depend on
  + Contribution of each student in the implementation of the assignment
  + **Plagiarism or copying will result in -20 marks**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FILL IN THE DETAILS GIVEN BELOW\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Assignment Set Number:**

**Group Name: GROUP SECTION B 18**

**Contribution Table:**

**Contribution** (This table should contain the list of all the students in the group. Clearly mention each student’s contribution towards the assignment. Mention “No Contribution” in cases applicable.)

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Name (as appears in Canvas)** | **ID NO** | **Contribution** |
| **1.** | **ARCHANA BANAKAR** | **2022MT70188** | **100%** |
| **2.** | **SIJY PAUL** | **2022MT70135** | **100%** |

**Resource for Part I, II and III:**

* Use following link to login to “eLearn” portal.
  + <https://elearn.bits-pilani.ac.in>
* Click on “My Virtual Lab – CSIS”
* Using your canvas credentials login in to Virtual lab
* In “BITS Pilani” Virtual lab click on “Resources”. Click on “Computer Organization and software systems” course.
  + Use resources within “LabCapsule3: Cache Memory”

**Code to be used:**

The following code written in STL Language, implements Sorting of elements in an array using Selection Sort technique.

program SelectionSort

VAR a array(10) INTEGER

VAR len byte

VAR i byte

VAR j byte

VAR p byte

VAR q byte

VAR x byte

VAR n byte

a(1)=15

a(2)=20

a(3)=19

a(4)=80

a(5)=30

a(6)=35

len = 6

for n = 1 to len

write(a(n), " ")

next

writeln("")

for i = 1 to len

for j = i+1 to len

p = a(i)

q = a(j)

if p > q then

x = p

a(i) = q

a(j) = x

end if

next

next

writeln("Sorted Array in ascending order")

for n = 1 to len

write(a(n), " ")

next

writeln("")

end

**General procedure to convert the given STL program into ALP:**

* Open CPU OS Simulator. Go to **advanced tab** and press **compiler** button
* Copy the above program in **Program Source** window
* Open **Compile** tab and press **compile** button
* In **Assembly Code,** enter **start address** and press **Load in Memory** button
* Now the assembly language program is available in CPU simulator.
* Set speed of execution to **FAST.**
* Open I/O console
* To run the program press **RUN** button.

**General Procedure to use Cache set up in CPU-OS simulator**

* After compiling and loading the assembly language code in CPU simulator, press “Cache-Pipeline” tab and select cache type as “both”. Press “SHOW CACHE” button.
* In the newly opened cache window, choose appropriate cache Type, cache size, set blocks, replacement algorithm and write policy.

**Part I: Direct Mapped Cache**

1. Execute the above program by setting block size to 2, 4, 8, 16 and 32 for cache size = 8, 16 and 32. Record the observation in the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Block Size | Cache size | # Hits | # Misses | % Miss Ratio | %Hit Ratio |
| 2 | 8 | 356 | 363 | 50.4% | 49.6% |
| 4 | 352 | 367 | 51.0% | 49% |
| 8 | 420 | 299 | 41.5% | 58.5% |
| 2 | 16 | 462 | 257 | 35.7% | 64.3% |
| 4 |  | 388 | 331 | 46% | 54% |
| 8 |  | 441 | 278 | 38.6% | 61.4% |
| 16 |  | 519 | 200 | 27.8% | 72.2% |
| 2 | 32 | 579 | 140 | 19.4% | 80.6% |
| 4 |  | 543 | 176 | 24.4% | 75.6% |
| 8 |  | 550 | 169 | 23.5% | 76.5% |
| 16 |  | 581 | 138 | 19.1% | 80.9% |
| 32 |  | 601 | 118 | 16.4% | 83.6% |

1. Plot a single graph of Cache hit ratio Vs Block size with respect to cache size = 8, 16 and 32. Comment on the graph that is obtained.

**Part II: Associative Mapped Cache**

1. Execute the above program by setting block size to 2, 4, 8, 16 and 32 for cache size = 8, 16 and 32. Record the observation in the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LRU Replacement Algorithm | | | | | |
| Block Size | Cache size | # Hits | # Misses | % Miss Ratio | %Hit Ratio |
| 2 | 8 | 332 | 387 | 53.8% | 46.2% |
| 4 | 325 | 394 | 54.8% | 45.2% |
| 8 | 384 | 335 | 46.5% | 53.5% |
| 2 | 16 | 502 | 217 | 30.1% | 69.9% |
| 4 |  | 364 | 355 | 49.3% | 50.7% |
| 8 |  | 404 | 315 | 43.8% | 56.2% |
| 16 |  | 475 | 244 | 33.9% | 66.1% |
| 2 | 32 | 620 | 99 | 13.7% | 86.3% |
| 4 |  | 546 | 173 | 24.0% | 76% |
| 8 |  | 475 | 244 | 33.9% | 66.1% |
| 16 |  | 491 | 228 | 31.7% | 68.3% |
| 32 |  | 576 | 143 | 19.8% | 80.2% |

1. Plot a single graph of Cache hit ratio Vs Block size with respect to cache size = 8, 16 and 32. Comment on the graph that is obtained.

c) Fill up the following table for three different replacement algorithms and state which replacement algorithm is better and why?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Replacement Algorithm: Random | | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 2 | 4 | 411 | 308 | 42.9% |
| 2 | 8 | 347 | 372 | 51.8% |
| 2 | 16 | 215 | 504 | 70.1% |
| 2 | 32 | 106 | 613 | 85.3% |
| 2 | 64 | 65 | 654 | 90.96% |
| Replacement Algorithm: FIFO | | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 2 | 4 | 417 | 302 | 42% |
| 2 | 8 | 387 | 332 | 46.2% |
| 2 | 16 | 279 | 440 | 61.2% |
| 2 | 32 | 99 | 620 | 86.3% |
| 2 | 64 | 82 | 637 | 88.6% |
| Replacement Algorithm: LRU | | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 2 | 4 | 416 | 303 | 42.2% |
| 2 | 8 | 387 | 332 | 46.2% |
| 2 | 16 | 217 | 502 | 69.9% |
| 2 | 32 | 99 | 620 | 86.3% |
| 2 | 64 | 79 | 640 | 89.1% |

d) Plot the graph of Cache Hit Ratio Vs Cache size with respect to different replacement algorithms. Comment on the graph that is obtained.

**Part III: Set Associative Mapped Cache**

Execute the above program by setting the following Parameters:

* Number of sets (Set Blocks): 2 way
* Cache Type: Set Associative
* Replacement: LRU/FIFO/Random

a) Fill up the following table for three different replacement algorithms and state which replacement algorithm is better and why?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Replacement Algorithm: Random | | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 2 | 4 | 410 | 309 | 43% |
| 2 | 8 | 382 | 337 | 46.9% |
| 2 | 16 | 265 | 454 | 63.2% |
| 2 | 32 | 136 | 583 | 81.1% |
| 2 | 64 | 82 | 637 | 88.6% |
| Replacement Algorithm: FIFO | | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 2 | 4 | 417 | 302 | 42% |
| 2 | 8 | 388 | 331 | 46.1% |
| 2 | 16 | 272 | 447 | 62.2% |
| 2 | 32 | 130 | 589 | 82% |
| 2 | 64 | 74 | 645 | 89.8% |
| Replacement Algorithm: LRU | | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 2 | 4 | 416 | 303 | 42.2% |
| 2 | 8 | 373 | 346 | 48.2% |
| 2 | 16 | 253 | 466 | 64.9% |
| 2 | 32 | 125 | 594 | 82.7% |
| 2 | 64 | 73 | 646 | 89.9% |

b) Plot the graph of Cache Hit Ratio Vs Cache size with respect to different replacement algorithms. Comment on the graph that is obtained.

c) Fill in the following table and analyse the behaviour of Set Associate Cache. Which one is better and why?

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Replacement Algorithm: LRU | | | | |
| Block Size, Cache size | Set Blocks | Miss | Hit | Hit ratio |
| 2, 64 | 2 – Way | 73 | 646 | 89.9% |
| 2, 64 | 4 – Way | 75 | 644 | 89.6% |
| 2, 64 | 8 – Way | 77 | 642 | 89.3% |

**Part IV: Case Study**

You are asked to design a Data Analytics Server for healthcare applications. What would be the challenges as a hardware designer you would need to address? Also mention the solutions you would propose.

[Address this case study with reference to the COA syllabus - Computer Organisation, Architecture perspective) we studied till date] (8M)