**Assignment I**

**Problem Bank 31**

**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
* **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 **30th December 2024 through Taxila only**. File submitted by any means outside TAXILA will not be accepted and marked. In case of any issues, please drop an email to the course lead TA Vaibhav Jain (email: vaibhav.jain@wilp.bits-pilani.ac.in).

**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. Remember that any kind of group changes after the announcement of the assignment is not allowed.

**Evaluation:**

* The assignment carries 10 marks
* Grading will depend on
  + Contribution of each student in the implementation of the assignment
  + **Plagiarism or copying will result in -10 marks**

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

**Assignment Set Number:**

**Group Name:**

**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 Taxila)** | **ID NO** | **Contribution** |
| **1** |  | **2024da04202** | **100%** |
| **2** | **MOHAMMED MUDASSIRULLAH SHERIFF** | **2024da04200** | **100%** |
|  |  |  |  |
|  |  |  |  |

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

* Use following link to login to “eLearn” portal.
  + <https://elearn.bits-pilani.ac.in>
* Click on “Cloud Virtual Lab - CSIS - Relevant to CSIS faculties and CSIS Students only” to login to lab portal using elearn credentials.
* Use Cloud virtual lab access manual to use the lab platform and access the Cloud virtual machine.
* In cloud lab machine, click on “Lab Resources” -> Click on “Computer Organization and software systems” course folder.
  + Use resources within “LabCapsule3: Cache Memory”

**Code to be used:**

The following code written in STL Language:

program <PB31\_ID Number>

var a array(5) byte

a(0) = 4

a(1) = 5

a(2) = 3

a(3) = 1

a(4) = 2

var len byte

var temp byte

var l1 byte

var l2 byte

var x1 byte

var x2 byte

var j byte

var j1 byte

var k byte

var i byte

len = 3

l1 = len - 1

for k =0 to len

write(a(k)," ")

next

writeln("")

writeln("Bubble Sort Starts")

for i = 0 to l1

l2 = len - i - 1

for j=0 to l2

j1 = j + 1

x1 = a(j)

x2 = a(j1)

if x1 > x2 then

temp = a(j1)

a(j1) = a(j)

a(j) = temp

end if

next

for k =0 to len

write(a(k)," ")

next

writeln("")

next

end

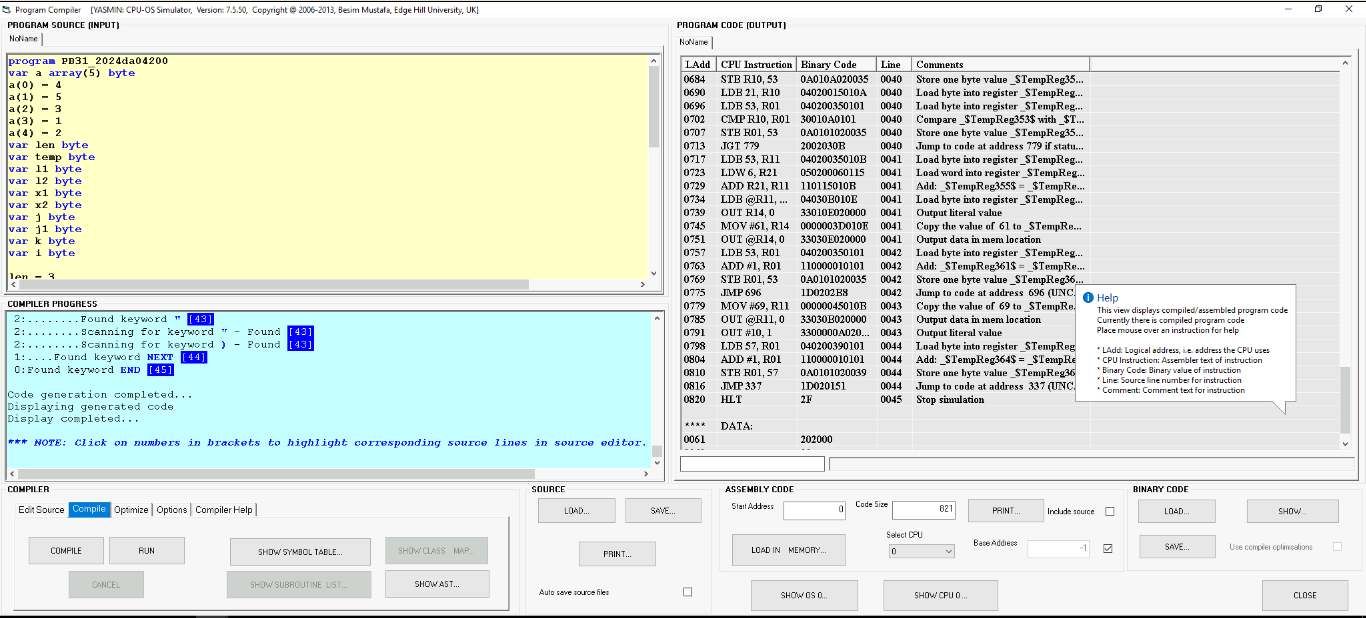
**General procedure to convert the given STL program into ALP:**

* Open CPU OS Simulator. Go to **advanced tab** and press **compiler** button
* Copy the above program in **Program Source** window
* Replace <PB31\_ID Number> by “**PB31\_ followed by your bits ID**”. For example, if mail id is 2024fc04666@wilp.bits-pilani.ac.in then PB31\_2024fc04666
* 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 “Data”. Press “SHOW CACHE” button.
* In the newly opened cache window, choose appropriate cache Type, cache size, set blocks, replacement algorithm according to the instructions as given in each part.
* The write policy is by-default used as write back policy.

**At first, present the snapshot of your PROGRAM SOURCE (INPUT) Window of the compiler simulator.**

**(Clear picture that shows the program name followed by your ID number as stated above)**

**Part I: Direct Mapped Cache**

1. Execute the above program by setting block size to 4, 8, 16, 32 and 64 for cache size = 16, 32 and 64. Record the observation in the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Block Size | Cache size | # Hits | # Misses | % Miss Ratio | %Hit Ratio |
| 4 | 16 | 353 | 210 | 37.3 | 62.69 |
| 8 | 330 | 233 | 41.3 | 58.92 |
| 16 | 366 | 197 | 34.9 | 65.00 |
| 4 | 32 | 429 | 134 | 23.8 | 76.19 |
| 8 | 412 | 151 | 26.8 | 73.17 |
| 16 | 433 | 130 | 23.0 | 76.90 |
| 32 | 408 | 155 | 27.5 | 72.46 |
| 4 | 64 | 517 | 46 | 8.17 | 91.8 |
| 8 | 513 | 50 | 8.88 | 91.11 |
| 16 | 521 | 42 | 7.46 | 92.53 |
| 32 | 529 | 34 | 6.04 | 93.96 |
| 64 | 521 | 42 | 7.46 | 92.53 |

1. Plot a single graph of Cache hit ratio Vs Block size with respect to cache size = 16, 32 and 64. Comment on the graph that is obtained.

**(Use any platform to obtain single graph, multiple graphs will fetch 0 marks)**

A graph with numbers and lines

Description automatically generated

* From the graph obtained, the increase in block size by keeping cache size constant does not have much change in the HIT ratio.
* Also higher the cache size higher is the hit ratio and similarly lower is its miss ratio which can improve performance and reduce power consumption.
* Cache performance is unpredictable in direct mapping. Handling of spatial locality is poor. Use of cache space is inefficient. Conflict misses are high.

**Part II: Associative Mapped Cache**

a) Use the given program, 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 |
| 4 | 4 | 327 | 236 | 41.92 |
| 4 | 8 | 295 | 268 | 47.60 |
| 4 | 16 | 223 | 340 | 60.39 |
| 4 | 32 | 126 | 437 | 77.62 |
| 4 | 64 | 56 | 507 | 90.05 |
| Replacement Algorithm: FIFO | | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 4 | 4 | 327 | 236 | 41.92 |
| 4 | 8 | 306 | 257 | 45.65 |
| 4 | 16 | 246 | 317 | 56.31 |
| 4 | 32 | 103 | 460 | 87.71 |
| 4 | 64 | 46 | 517 | 91.83 |
| Replacement Algorithm: LRU | | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 4 | 4 | 327 | 236 | 41.92 |
| 4 | 8 | 316 | 247 | 43.87 |
| 4 | 16 | 263 | 300 | 53.29 |
| 4 | 32 | 94 | 469 | 83.30 |
| 4 | 64 | 45 | 518 | 92.01 |

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

**(Use any platform to obtain single graph, multiple graphs will fetch 0 marks)**

**Observations and Inference:**

1. From the Hit-Ratio vs cache size graph,

* We can see that for smaller cache sizes upto 16Bytes, the Hit-Ratio for Random replacement is better than the other algorithms. This may be because the number of cache lines required is maximum at 4 lines for a block size of 4
* For Cache Size greater than 16b and above, the LRU Algorithm performs best for the given block size. This is because of the Temporal Locality of reference.

1. For a given block size=4, upon **increasing the cache size, there is an increase in Hit-Raitos** for all the 3 replacement algorithms.

There is a considerable jump in the performance (hit-ratio) from 16b to 32b.

Beyond 32b, the difference in the change of increase is minimal.

**Part III: Set Associative Mapped Cache**

Execute the above program by setting the following Parameters:

* Number of sets (Set Blocks): 4 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 |
| 4 | 16 |  |  |  |
| 4 | 32 |  |  |  |
| 4 | 64 |  |  |  |
| 4 | 128 |  |  |  |
|  | Replacement Algorithm: FIFO | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 4 | 16 |  |  |  |
| 4 | 32 |  |  |  |
| 4 | 64 |  |  |  |
| 4 | 128 |  |  |  |
|  | Replacement Algorithm: LRU | | | |
| Block Size | Cache size | Miss | Hit | Hit ratio |
| 4 | 16 |  |  |  |
| 4 | 32 |  |  |  |
| 4 | 64 |  |  |  |
| 4 | 128 |  |  |  |

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

**(Use any platform to obtain single graph, multiple graphs will fetch 0 marks)**

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 |
| 4, 64 | 2 – Way |  |  |  |
| 4, 64 | 4 – Way |  |  |  |
| 4, 64 | 8 – Way |  |  |  |