La Trobe University

Department of Computer Science and Computer Engineering

CSE2ARC/CS3 ARCHITECTURE

ASSIGNMENT 1

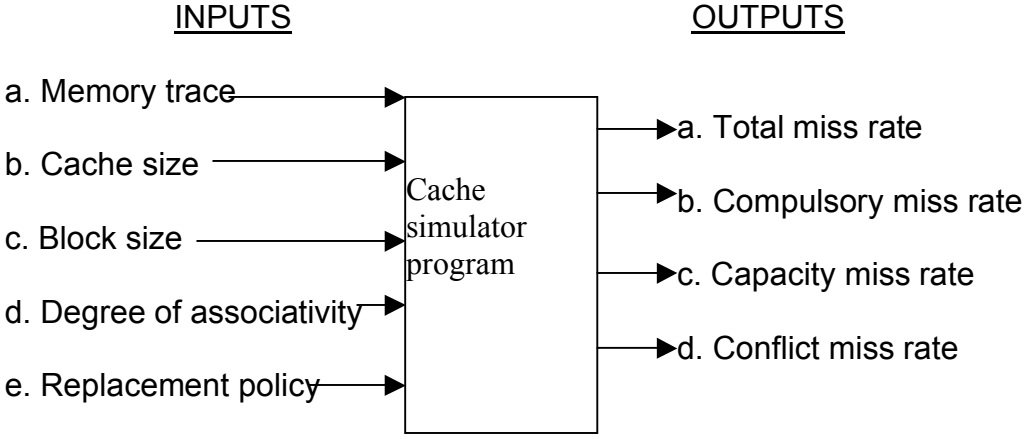
Due Date Monday, 9.30am, 10 October 2005

Objective To write a cache simulator and examine various cache performances given a trace file.

Marks This assignment is worth 30% of the subject.

Requirements

You are required to program and implement a cache simulator in C, C++, or Java, which will have the following inputs and outputs:-



Memory trace □ smalltex.din
Cache size □ size of cache in bytes
Block size □ size of blocks in bytes
Degree of associativity □ direct (1-way), 2-way, 4-way, 8-way, fully associative
Replacement policy □ Random and LRU
Total miss rate = compulsory + capacity + conflict miss rates

* The memory trace file (smalltex.din) is located at
/home/csiii/csiiilib/arc/smalltex.din
Copy this file to your arc directory where you will be running your assignment from.

Submission Requirements

Using your cache simulator and using smalltex.din as your memory trace determine the total miss rate, compulsory miss rate, capacity miss rate, and conflict miss rate for the following cache configurations by varying the inputs as indicated. Examine and comment on your results/observations in each case and briefly compare them to a small selection of runs using dineroiv (commercial cache simulator):-

1. Keeping block size constant (say 64 bytes) compare the different replacement policy's with several cache sizes (say 16, 64, 256, 512KB) and associativities (direct, 2-, 4-, 8-way, fully associative). {See Fig5.6, p400 of

prescribed text for a guide to tabulating your data} Comment on your results/observations

2. Keeping replacement policy constant (either random or LRU) and block size constant (say 64 bytes) collect total, compulsory, capacity, and conflict miss rates for each of the following cache organizations:-

   a. Cache sizes 4, 16, 64, 128, 256, 512 KB,
   b. Degree of associativity 1-way, 2-way, 4-way, 8-way, fully.

   {See Fig 5.14, p424 as a guide to tabulating your data}
   Produce a graph of Total miss rate per type versus cache size {Fig 5.15 top graph only is required}. Comment on your results/observations.

3. For:-
   a. Direct mapped cache, and
   b. 4-way set associative cache (using either random or LRU replacement policy)

   Using the data collected in 2. above produce a graph giving total miss rate versus block size and comment on your observations. {See Fig 5.16, 5.17 p427 as a guide.}
4. From your results make a recommendation as to which cache organization you would select for a single level cache. Justify your recommendation.

Report

Your **written** submission/report should include clear documentation of:

a. cache simulator – both design and code,
b. all your results – tables, graphs, comments and recommendations,
c. **several script runs only** of both your simulator and dineroIV.

Electronic Submission

You are **to electronically submit** your cache simulator code:

>submit Arc "filename.c" {to be confirmed}

OR

http://students.cs.latrobe.edu.au/submit/

Submission Date

Monday, 10 October 2005, 9.30am in the designated assignment "Arc" submission box.

Late Submissions, Penalties, Plagiarism

The normal penalties for late submissions and procedures for occurrences of plagiarism apply as outlined in the student handbook.

Outline Marking Scheme

Code = 50%
Results / correctness = 30%
Report = 10%
Overall presentation = 10%

Important notes

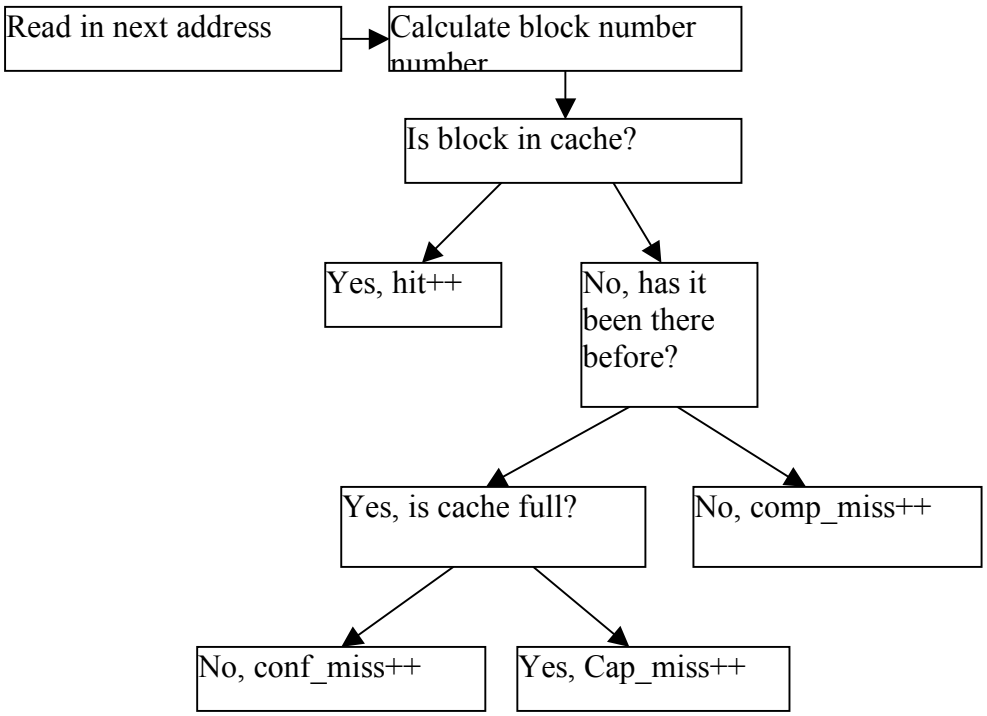
1. Your program must be able to be compiled and run on latcs"n" machines running Linux,

2. Use microtex.din when developing your simulator code, however all reported/final results are to be on smalltex.din
3. I will be available for assignment consultation from Mon 6 Sep at most times when I'm not either lecturing or at prac classes.

Assignment hints

1. You will notice smalltex.din has only two columns – you can ignore the first column, the second column is the byte address in **hexadecimal** of the byte address issued by the CPU. You can assume that:
   a. All the addresses are in RAM (no page faults)
   b. The addresses start at 0 and go through to some large number - say ffffff = 16MB (check the trace file to make sure that no address is present which is larger than 16MB)
   c. Dynamically calculate which block_number the address is in immediately after reading in each address, then, once you have the block_number you can see if its in cache (according to your cache mapping function).

1.



When determining whether cache is full – it means over the whole cache regardless of the mapping (due to its associativity), thus any free space even if the block can't go there due to its mapping/associativity - means that the cache **is not yet full.**

When we refer to the least recently used block in cache, this means the block that was least recently used in time. Whenever a block is used (either called into cache (a miss), or hit while in the cache), you will need to note that it then becomes the most recently used block. How you implement this is entirely up to you, however I would suggest that you keep it simple. This does not require any advanced algorithms or data structures.

Smalltex.din is a very large file. While developing your code, you can just make a file (eg. Microtex.din) which only has the first 100 entries. However, make sure that when you obtain your final results you use the unchanged smalltex.din