REPORT – Assignment 3

IMT2019524 - N.Srijan kumar

Code written in C++

Run using g++ IMT2019524_DMCache.c
g++ IMT2019524_SACache.c

Program reads the input line by line from the files “gcc.trace”, “gzip.trace”, “mcf.trace”,“swim.trace”,“twolf.trace” finds the addresses, prints the Total input adress count,Hit count,Miss count,Hit Rate,Miss Rate and Hit/Miss rate of the respective files.

Hit Rate(per 1) :

Cache	gcc	gzip	mcf	swim	twolf
DM	0.937632	0.667055	0.010322	0.925945	0.987476
SA	0.938332	0.667055	0.010326	0.926222	0.987625

Miss Rate(per 1) :

Cache	gcc	gzip	mcf	swim	twolf
DM	0.062367	0.332945	0.989677	0.740551	0.012524
SA	0.061667	0.332945	0.989673	0.737781	0.012751

Hit/Miss Rates :

Cache	gcc	gzip	mcf	swim	twolf
DM	15.0339	2.0035	0.010430	12.5035	78.8452
SA	15.2159	2.0035	0.010434	12.5542	79.8074

1) Direct Mapped 256 KB Cache

Block size : 4bytes , 32 bit adress

Index = 16 bits , byte offset = 2 bits , Tag = 14 bits

1 2 3 4 5 6 7 8 .....15 16 17 18 19 20 21 .....31 32
0 1 1 0 1 1 1 1 1 .....1 0 1 1 1 1 1 .....0 1

(1 – 14) Tag , (14-30) Index , byte offset – (31-32)

Index	Tag – 14 bit	Valid	Data - 4Byte
0			
1			
2			
.			
.			
.			
.			
65534			
65535			

2) Set Associative 256 KB Cache

Block size : 4bytes , 32 bit adress

Index = 14 bits , byte offset = 2 , Tag = 16 bits

1 2 3 4 5 6 7 8 .....15 16 17 18 19 20 21 32
0 1 1 0 1 1 1 1 1 .....1 0 1 1 1 1 1 1

(1 – 16) Tag , (16-30) Index , (31-32) byte offset

Random replacement used in the case 4 ways are filled.

4 Way Blocks of Type :

Index	Tag – 16 bit	Valid	Data - 4Byte
0			
1			
2			
.			
.			
.			
.			
16382			
16383			

Observations :

We can observe 90% hit rates in some of the cases this shows us how useful the cache is in accessing the data fastly.

Set Associative hit rate is better than Direct Mapped cache irrespective of replacement algorithm used.

We can see that the hit rate of Set Associative cache is never less than that of Direct Mapped.