# CS622

# Assignment 3

# Group 22

NAME: ANKITA DEY
ROLL NO: 20111013

NAME: SHILPA CHATTERJEE
ROLL NO: 20111057

## 1.1 REPORT ON COLLECTION OF TRACES

| **Program Name** | **Total Number of Memory Accesses** |
|---|---|
| Program 1 | 140525386 |
| Program 2 | 2531784 |
| Program 3 | 9715608 |
| Program 4 | 1064762 |

## 1.2 REPORT ON CACHE SIMULATOR

Explanation of the messages obtained by both L1 and L2 banks for all the programs

a) PUTE : This message is sent by the home node of to the requester node when there is no sharer in the directory for the block and the block is asked in read mode(S State) by the requester. Directory thinks block is given in M state and requester stores block in E state (Has both read and write permission but isn't dirty yet).

b) PUT : This message is sent by the home node of to the requester node when there are sharers in the directory for the block and the block is asked in read mode by the requester. This message can also be sent by the owner of the block to the requester , when the block requested is in M state in the directory.

c) PUTX : This message is sent by home node to the requester node when the block is requested in M state. If the home node finds the requested block in M state , then the home node forwards the requester's request to the owner and the owner sends this PUTX message then to the requester.

d) GET : This message is received by the home node when a requester node sends read request for a block. If the home node on checking the directory finds out that the block is in M state , then it forwards the request to the owner of the block.

e) GETX : This message is received by the home node when a requester node sends write request for a block. If the home node on checking the directory finds out that the block is in M state , then it forwards the request to the owner of the block.

f) UPGRADE : This message is received by the home node when the requester needs write permission to a block which it already has in S state.
g) UPGRADE_ACK : This message is sent by the home node to the requester on receiving an UPGRADE message from the requester. This indicates requester can now change the state of the block to M from S.
h) INV : This is the message sent by the home node to the sharers of a block on receiving a write request for the same block. This indicates home node is requesting Sharers to invalidate the block.
i) INV_ACK : This message sent by the nodes on receiving INV message. On receiving INV_ACK from all sharers. This indicates sharers have invalidated the block.
j) SWB : This message is sent by the owner of a block to the home node on receiving a forwarded GET request from home node so that home node can provide block next time any core sends Read request for it. After SWB is served there is no owner of the block. Also, if a block is in Pending Shared state, it changes to S state upon receiving SWB.
k) ACK : This message is sent by the owner of a block to the home node on receiving a forwarded GETX request from home node. L2 directory node changes state from PDEX (Pending Dirty Exclusive) to M upon receiving ACK.
l) WB_Req: When the shared L2 cache evicts a dirty block , it checks the directory to find if the block is in M state in any of the L1 caches of the cores, then it request for the writeback so that it can write the updated data back to next level of memory.
m) WB : A node on receiving a WB_Req , sends WB to the home node. Also if a core has to be evicted from L2 cache and is in M state in L1, owner sends WB message to the home node so that the most updated version can be written back to the memory.
n) NAK (Negative Acknowledgement) : NAK is received by a core when it sends GET/GETX/UPGRADE requests to a directory node that has the block currently in Pending state. The requester on receiving NAK stores these requests in a buffer with a timer (we considered the timer to be 5 cycles) that is decremented every cycle and later the requester resends the request when NAK timer goes to 0.

## 1.2.1 REPORT ON CACHE ACCESSES ON PROGRAM 1

| **Count Name** | **Total Counts** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Number of simulated cycles | 140078085 | | | | | | | |
| Number of L1 cache hits | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 36570383 | 13750750 | 13688278 | 13356431 | 13556061 | 13492469 | 13425216 | 13603029 |
| Number of L1 cache read misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 426143 | 721122 | 753471 | 916038 | 818733 | 851584 | 883351 | 781224 |
| Number of L1 cache write misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 767332 | 208287 | 237980 | 407656 | 305331 | 335992 | 371497 | 295624 |
| Number of L1 cache upgrade misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 25 | 57 | 479 | 83 | 73 | 173 | 154 | 361 |
| Number of L1 cache accesses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 37763883 | 14680216 | 14680208 | 14680208 | 14680198 | 14680218 | 14680218 | 14680238 |
| Number of L2 cache misses | 6499504 | | | | | | | |

Messages received by L1 Caches and their respective counts

| **Message Name** | **Message Count Per Core** | | | | | | | | **Total Message Count** |
|---|---|---|---|---|---|---|---|---|---|
| | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** | **L1 Cache (All 8 cores)** |
| PUTE | 424347 | 718674 | 750195 | 913419 | 815843 | 848800 | 880279 | 776964 | 6128521 |
| PUTX | 721234 | 4 | 5 | 4 | 4 | 4 | 4 | 4 | 721263 |
| PUT | 3437 | 2668 | 3307 | 2184 | 1532 | 2944 | 2919 | 3467 | 22458 |
| GET | 3435 | 2599 | 2038 | 1992 | 1349 | 2629 | 2408 | 2862 | 19312 |
| GETX | 21 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 22 |
| UPGRADE_ACK | 25 | 75 | 1227 | 195 | 175 | 313 | 502 | 605 | 3117 |
| INV | 221 | 473 | 187 | 258 | 1369 | 13 | 490 | 161 | 3172 |
| INV_ACK | 28 | 75 | 1247 | 195 | 178 | 314 | 530 | 605 | 3172 |
| WB_Req | 204 | 409 | 333 | 430 | 458 | 298 | 332 | 380 | 2844 |

Messages received by L2 Cache Banks and their respective counts

| **Message Name** | **Message Count** |
|---|---|
| GET | 6150979 |
| GETX | 721263 |
| UPGRADE | 3117 |
| SWB | 19312 |
| WB | 6572944 |
| ACK | 22 |

We can see here that the following are followed when prog1 trace is given as input to our simulator (These explains the frequency of message types):

- L2 GET = L1 PUT + L1 PUTE [For every GET request sent to home node, PUT is received as reply from home in case there are other sharers or PUTE if there are no sharers]
- L2 GETX = L1 PUTX [For every GETX request sent to home node, PUTX is received as reply either from current owner or from home node in case block is not present in L1 cache]
- L2 UPGRADE = L1 UPGRADE_ACK [For every Upgrade request sent to home node, home node replies with an Upgrade Acknowledgement to requesting core]
- L2 SWB = L1 GET [For every GET request received by owner of a block, it sends Sharing Write Back to home node (apart from sending PUT to core that request for the block for Read)]

- L2 ACK = L1 GETX [For every GETX request received by owner of a block, owner replies with acknowledgement to home node (apart from sending PUTX to core that request for the block for Write)]
- L1 INV = L1 INV_ACK [For every invalidation request sent from home node to sharers, every sharers reply with invalidation acknowledgement]
- We didn’t get any NAK

## 1.2.2 REPORT ON CACHE ACCESSES ON PROGRAM 2

| **Count Name** | **Total Counts** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Number of simulated cycles | 2525517 | | | | | | | |
| Number of L1 cache hits | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 653355 | 239296 | 240741 | 240309 | 247372 | 237672 | 237668 | 188563 |
| Number of L1 cache read misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 16720 | 24605 | 24606 | 24607 | 24606 | 24607 | 24607 | 16416 |
| Number of L1 cache write misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 65871 | 24 | 18 | 18 | 26 | 18 | 24 | 24 |
| Number of L1 cache upgrade misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 7 | 0 | 1 | 1 | 0 | 2 | 0 | 0 |
| Number of L1 cache access | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 735953 | 263925 | 265366 | 264935 | 272004 | 262299 | 262299 | 205003 |
| Number of L2 cache misses | 67322 | | | | | | | |

<u>Messages received by L1 Caches and their respective counts</u>

| **Message Name** | **Message Count Per Core** | | | | | | | | **Total Message Count** |
|---|---|---|---|---|---|---|---|---|---|
| | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** | **L1 Cache (All 8 cores)** |
| PUTE | 16694 | 24587 | 24099 | 24582 | 24567 | 24583 | 24583 | 12141 | 175836 |
| PUTX | 65860 | 6 | 6 | 5 | 6 | 6 | 6 | 6 | 65901 |
| PUT | 64 | 11 | 9 | 17 | 13 | 19 | 12 | 15 | 160 |
| GET | 60 | 5 | 4 | 5 | 4 | 4 | 1 | 8 | 91 |
| GETX | 20 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 25 |
| UPGRADE_ACK | 7 | 1 | 2 | 4 | 0 | 5 | 1 | 1 | 21 |
| INV | 3 | 5 | 6 | 8 | 4 | 3 | 4 | 6 | 39 |
| INV_ACK | 14 | 2 | 4 | 7 | 0 | 9 | 1 | 2 | 39 |

Messages received by L2 Cache Banks and their respective counts

| **Message Name** | **Message Count** |
|---|---|
| GET | 175996 |
| GETX | 65901 |
| UPGRADE | 21 |
| SWB | 91 |
| WB | 131386 |
| ACK | 25 |

We can see here that the following are followed when prog2 trace is given as input to our simulator (These explains the frequency of message types):

- L2 GET = L1 PUT + L1 PUTE [For every GET request sent to home node, PUT is received as reply from home in case there are other sharers or PUTE if there are no sharers]
- L2 GETX = L1 PUTX [For every GETX request sent to home node, PUTX is received as reply either from current owner or from home node in case block is not present in L1 cache]
- L2 UPGRADE = L1 UPGRADE_ACK [For every Upgrade request sent to home node, home node replies with a Upgrade Acknowledgement to requesting core]
- L2 SWB = L1 GET [For every GET request received by owner of a block, it sends Sharing Write Back to home node (apart from sending PUT to core that request for the block for Read)]
- L2 ACK = L1 GETX [For every GETX request received by owner of a block, owner replies with acknowledgement to home node (apart from sending PUTX to core that request for the block for Write)]
- L1 INV = L1 INV_ACK [For every invalidation request sent from home node to sharers, every sharers reply with invalidation acknowledgement]
- We didn’t get any NAK

### 1.2.3 REPORT ON CACHE ACCESSES ON PROGRAM 3

| **Count Name** | **Total Counts** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Number of simulated cycles | 9693055 | | | | | | | |
| Number of L1 cache hits | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 1933020 | 1006985 | 1026359 | 1029690 | 1035052 | 1004389 | 1027455 | 996241 |
| Number of L1 cache read misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 131414 | 65563 | 65567 | 65573 | 65567 | 65564 | 65565 | 65564 |
| Number of L1 cache write misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 65881 | 23 | 21 | 15 | 20 | 23 | 22 | 23 |
| Number of L1 cache upgrade misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 3 | 0 | 2 | 4 | 2 | 0 | 1 | 0 |
| Number of L1 cache access | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 2130318 | 1072571 | 1091949 | 1095282 | 1100641 | 1069976 | 1093043 | 1061828 |
| Number of L2 cache misses | 67417 | | | | | | | |

<u>Messages received by L1 Caches and their respective counts</u>

| **Message Name** | **Message Count Per Core** | | | | | | | | **Total Message Count** |
|---|---|---|---|---|---|---|---|---|---|
| | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** | **L1 Cache (All 8 cores)** |
| PUTE | 127902 | 65554 | 65554 | 65548 | 65545 | 65544 | 65544 | 65544 | 586725 |
| PUTX | 65867 | 4 | 4 | 4 | 5 | 4 | 4 | 4 | 65896 |
| PUT | 57 | 502 | 511 | 518 | 512 | 510 | 512 | 510 | 3632 |
| GET | 56 | 502 | 506 | 503 | 500 | 501 | 501 | 501 | 3570 |
| GETX | 21 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 22 |
| UPGRADE_ACK | 3 | 0 | 2 | 4 | 3 | 0 | 2 | 0 | 14 |
| INV | 3 | 2 | 4 | 4 | 3 | 3 | 2 | 1 | 22 |
| INV_ACK | 7 | 0 | 3 | 2 | 6 | 0 | 4 | 0 | 22 |
| WB_Req | 0 | 4 | 4 | 5 | 4 | 4 | 4 | 4 | 29 |

Messages received by L2 Cache Banks and their respective counts

| Message Name | Message Count |
|---|---|
| GET | 590357 |
| GETX | 65896 |
| UPGRADE | 14 |
| SWB | 3570 |
| WB | 586708 |
| ACK | 22 |

We can see here that the following are followed when prog3 trace is given as input to our simulator (These explains the frequency of message types):

- L2 GET = L1 PUT + L1 PUTE [For every GET request sent to home node, PUT is received as reply from home in case there are other sharers or PUTE if there are no sharers]
- L2 GETX = L1 PUTX [For every GETX request sent to home node, PUTX is received as reply either from current owner or from home node in case block is not present in L1 cache]
- L2 UPGRADE = L1 UPGRADE_ACK [For every Upgrade request sent to home node, home node replies with a Upgrade Acknowledgement to requesting core]
- L2 SWB = L1 GET [For every GET request received by owner of a block, it sends Sharing Write Back to home node (apart from sending PUT to core that request for the block for Read)]
- L2 ACK = L1 GETX [For every GETX request received by owner of a block, owner replies with acknowledgement to home node (apart from sending PUTX to core that request for the block for Write)]
- L1 INV = L1 INV_ACK [For every invalidation request sent from home node to sharers, every sharers reply with invalidation acknowledgement]
- We didn’t get any NAK

## 1.2.4 REPORT ON CACHE ACCESSES ON PROGRAM 4

| **Count Name** | **Total Counts** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Number of simulated cycles | 1063490 | | | | | | | |
| Number of L1 cache hits | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 530465 | 57451 | 57450 | 57450 | 57445 | 57449 | 57445 | 57445 |
| Number of L1 cache read misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 8525 | 8220 | 8221 | 8221 | 8221 | 8221 | 8221 | 8221 |
| Number of L1 cache write misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 65937 | 19 | 18 | 18 | 24 | 19 | 24 | 24 |
| Number of L1 cache upgrade misses | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 5 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| Number of L1 cache access | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** |
| | 604932 | 65690 | 65690 | 65690 | 65690 | 65690 | 65690 | 65690 |
| Number of L2 cache misses | 66887 | | | | | | | |

<u>Messages received by L1 Caches and their respective counts</u>

| **Message Name** | **Message Count Per Core** | | | | | | | | **Total Message Count** |
|---|---|---|---|---|---|---|---|---|---|
| | **Core 0** | **Core 1** | **Core 2** | **Core 3** | **Core 4** | **Core 5** | **Core 6** | **Core 7** | **L1 Cache (All 8 cores)** |
| PUTE | 8496 | 8210 | 8198 | 8199 | 8199 | 8199 | 8199 | 8203 | 65903 |
| PUTX | 65860 | 11 | 5 | 6 | 6 | 7 | 7 | 7 | 65909 |
| PUT | 56 | 12 | 13 | 15 | 10 | 16 | 12 | 8 | 142 |
| GET | 52 | 12 | 4 | 4 | 1 | 5 | 3 | 5 | 86 |
| GETX | 21 | 4 | 0 | 1 | 0 | 2 | 2 | 2 | 32 |
| UPGRADE_ACK | 5 | 0 | 3 | 2 | 0 | 4 | 1 | 0 | 15 |
| INV | 3 | 5 | 2 | 4 | 2 | 3 | 4 | 3 | 26 |
| INV_ACK | 4 | 0 | 6 | 4 | 0 | 10 | 2 | 0 | 26 |

Messages received by L2 Cache Banks and their respective counts

| Message Name | Message Count |
|---|---|
| GET | 66045 |
| GETX | 65909 |
| UPGRADE | 15 |
| SWB | 86 |
| WB | 65845 |
| ACK | 32 |

We can see here that the following are followed when prog4 trace is given as input to our simulator (These explains the frequency of message types):

- L2 GET = L1 PUT + L1 PUTE [For every GET request sent to home node, PUT is received as reply from home in case there are other sharers or PUTE if there are no sharers]
- L2 GETX = L1 PUTX [For every GETX request sent to home node, PUTX is received as reply either from current owner or from home node in case block is not present in L1 cache]
- L2 UPGRADE = L1 UPGRADE_ACK [For every Upgrade request sent to home node, home node replies with a Upgrade Acknowledgement to requesting core]
- L2 SWB = L1 GET [For every GET request received by owner of a block, it sends Sharing Write Back to home node (apart from sending PUT to core that request for the block for Read)]
- L2 ACK = L1 GETX [For every GETX request received by owner of a block, owner replies with acknowledgement to home node (apart from sending PUTX to core that request for the block for Write)]
- L1 INV = L1 INV_ACK [For every invalidation request sent from home node to sharers, every sharers reply with invalidation acknowledgement]
- We didn’t get any NAK