**Homework Wan Huzaifah bin Wan Azhar**

**Answer:**



**FIFO s0 -n 10:**

ARG addresses -1

ARG addressfile

ARG numaddrs 10

ARG policy FIFO

ARG clockbits 2

ARG cachesize 3

ARG maxpage 10

ARG seed 0

ARG notrace False

Access: 8 MISS State of Memory = [8]

Access: 7 MISS State of Memory = [7,8]

Access: 4 MISS State of Memory = [4,7,8]

Access: 2 MISS State of Memory = [2,4,7], Replace: 8

Access: 5 MISS State of Memory = [5,2,4], Replace: 7

Access: 4 HIT State of Memory = [5,2,4]

Access: 7 MISS State of Memory = [7,5,2], Replace: 4

Access: 3 MISS State of Memory = [3,7,5], Replace: 2

Access: 4 MISS State of Memory = [4,3,7], Replace: 5

Access: 5 MISS State of Memory = [5,4,3], Replace: 7

**FIFO s1 -n 10:**

Access: 1 MISS State of Memory = [1]

Access: 8 MISS State of Memory = [8, 1]

Access: 7 MISS State of Memory = [7, 8, 1]

Access: 2 MISS State of Memory = [2, 7, 8]

Access: 4 MISS State of Memory = [4, 2, 7]

Access: 4 HIT State of Memory = [4, 2, 7]

Access: 6 MISS State of Memory = [6, 4, 2]

Access: 7 MISS State of Memory = [7, 6, 4]

Access: 0 MISS State of Memory = [0, 7, 6]

Access: 0 HIT State of Memory = [0, 7, 6]

**LRU -s 0 -n 10**

ARG addresses -1

ARG addressfile

ARG numaddrs 10

ARG policy LRU

ARG clockbits 2

ARG cachesize 3

ARG maxpage 10

ARG seed 0

ARG notrace False

Access: 8 MISS State of Memory = [8]

Access: 7 MISS State of Memory = [7, 8]

Access: 4 MISS State of Memory = [4, 7, 8]

Access: 2 MISS State of Memory = [2, 4, 7], Replace: 8

Access: 5 MISS State of Memory = [5, 2, 4], Replace: 7

Access: 4 HIT State of Memory = [4, 5, 2]

Access: 7 MISS State of Memory = [7, 4, 5], Replace: 2

Access: 3 MISS State of Memory = [3, 7, 4], Replace: 5

Access: 4 HIT State of Memory = [4, 3, 7]

Access: 5 MISS State of Memory = [5, 4, 3], Replace: 7

**LRU -s 1 -n 10**

Access: 1 MISS State of Memory = [1]

Access: 8 MISS State of Memory = [8, 1]

Access: 7 MISS State of Memory = [7, 8, 1]

Access: 2 MISS State of Memory = [2, 7, 8], Replace: 1

Access: 4 MISS State of Memory = [4, 2, 7], Replace: 8

Access: 4 HIT State of Memory = [4, 2, 7]

Access: 6 MISS State of Memory = [6, 4, 2], Replace: 7

Access: 7 MISS State of Memory = [7, 6, 4], Replace: 2

Access: 0 MISS State of Memory = [0, 7, 6], Replace: 4

Access: 0 HIT State of Memory = [0, 7, 6]

**OPT -s 0 -n 10**

ARG addresses -1

ARG addressfile

ARG numaddrs 10

ARG policy OPT

ARG clockbits 2

ARG cachesize 3

ARG maxpage 10

ARG seed 0

ARG notrace False

Access: 8 MISS State of Memory = [8]

Access: 7 MISS State of Memory = [7, 8]

Access: 4 MISS State of Memory = [4, 7, 8]

Access: 2 MISS State of Memory = [4, 7, 2], Replace: 8

Access: 5 MISS State of Memory = [4, 7, 5], Replace: 2

Access: 4 HIT State of Memory = [4, 7, 5]

Access: 7 HIT State of Memory = [4, 7, 5]

Access: 3 MISS State of Memory = [4, 3, 5], Replace: 7

Access: 4 HIT State of Memory = [4, 3, 5]

Access: 5 HIT State of Memory = [4, 3, 5]

**OPT -s 1 -n 10**

Access: 1 MISS State of Memory = [1]

Access: 8 MISS State of Memory = [1, 8]

Access: 7 MISS State of Memory = [1, 8, 7]

Access: 2 MISS State of Memory = [2, 8, 7], Replace: 1

Access: 4 MISS State of Memory = [2, 4, 7], Replace: 8

Access: 4 HIT State of Memory = [2, 4, 7]

Access: 6 MISS State of Memory = [2, 6, 7], Replace: 4

Access: 7 HIT State of Memory = [2, 6, 7]

Access: 0 MISS State of Memory = [0, 6, 7], Replace: 2

Access: 0 HIT State of Memory = [0, 6, 7]



Worst address reference stream for:

* LRU: python .\paging-policy.py -C 5 -p LRU -c -a 1,2,3,4,5,6,1,2,3,4,5,6
  + As cache size Is 5, when number 6 is inputted, it replace 1 as it is the least recently used since it was the first one to be inserted and yet not used.
  + By repeating the same number, it achieved the worst case as the old one is replaced before it can be used.
* FIFO: python .\paging-policy.py -C 5 -p LRU -c -a 1,2,3,4,5,6,1,2,3,4,5,6
  + The reasoning is the same for LRU but it is to be noted that if there is a hit in LRU, the cache will change to most recently used but for FIFO, the cache will not change if there is a hit.
* MRU: python .\paging-policy.py -C 5 -p MRU -c -a 1,2,3,4,5,6,5,6,5,6,5,6
  + It achieved the worst misses if there is a repeating miss between two different number.
  + As it can be seen, the first number of 6 replace 5 as it is the most recently used, but then, there is an access to 5, so it replace 6 and so on and on..



Random trace:

python .\paging-policy.py -s 13

Access: 2 Hit/Miss? State of Memory?

Access: 6 Hit/Miss? State of Memory?

Access: 6 Hit/Miss? State of Memory?

Access: 8 Hit/Miss? State of Memory?

Access: 1 Hit/Miss? State of Memory?

Access: 2 Hit/Miss? State of Memory?

Access: 1 Hit/Miss? State of Memory?

Access: 2 Hit/Miss? State of Memory?

Access: 7 Hit/Miss? State of Memory?

Access: 1 Hit/Miss? State of Memory?

* FIFO: Will be moderately great at swapping. Approximately 40% hitrate
* OPT: The best policy to swap. Around 50% hitrate.
* LRU: Slightly better than FIFO. Around 40% hitrate.
* RAND: Just random. For this trace, it is expected to hit 50%



python .\paging-policy.py -c -a 1,2,3,3,4,1,2,1,2,3 -C 4

This generates trace with temporal locality. All the address are in the range of 1 to 4.

* LRU performs the worst on this trace. Hitrate are 30%.
  + As it can be seen from the trace, LRU perform badly because LRU replace the old address with new address.
  + But as locality insert old address, LRU does not have the cache for old address when it has been replaced by newer address.
* Hitrate for random are 50%, which much better than LRU.
  + However RAND are just random.
* Hitrate for CLOCK are 40% which is less than RAND.
  + CLOCK perform better than LRU because it chooses the best LRU to be replaced rather than the oldest one in the queue.
  + Changing clock bits does not change anything.