MPCA CODING ASSIGNMENT

TEAM MEMBERS:

1) M NIRANJAN-PES2UG23CS308
2) KISHORE H N – PES2UG23CS278
3) KEERTHAN P V – PES2UG23CS272

QUESTION NO.3

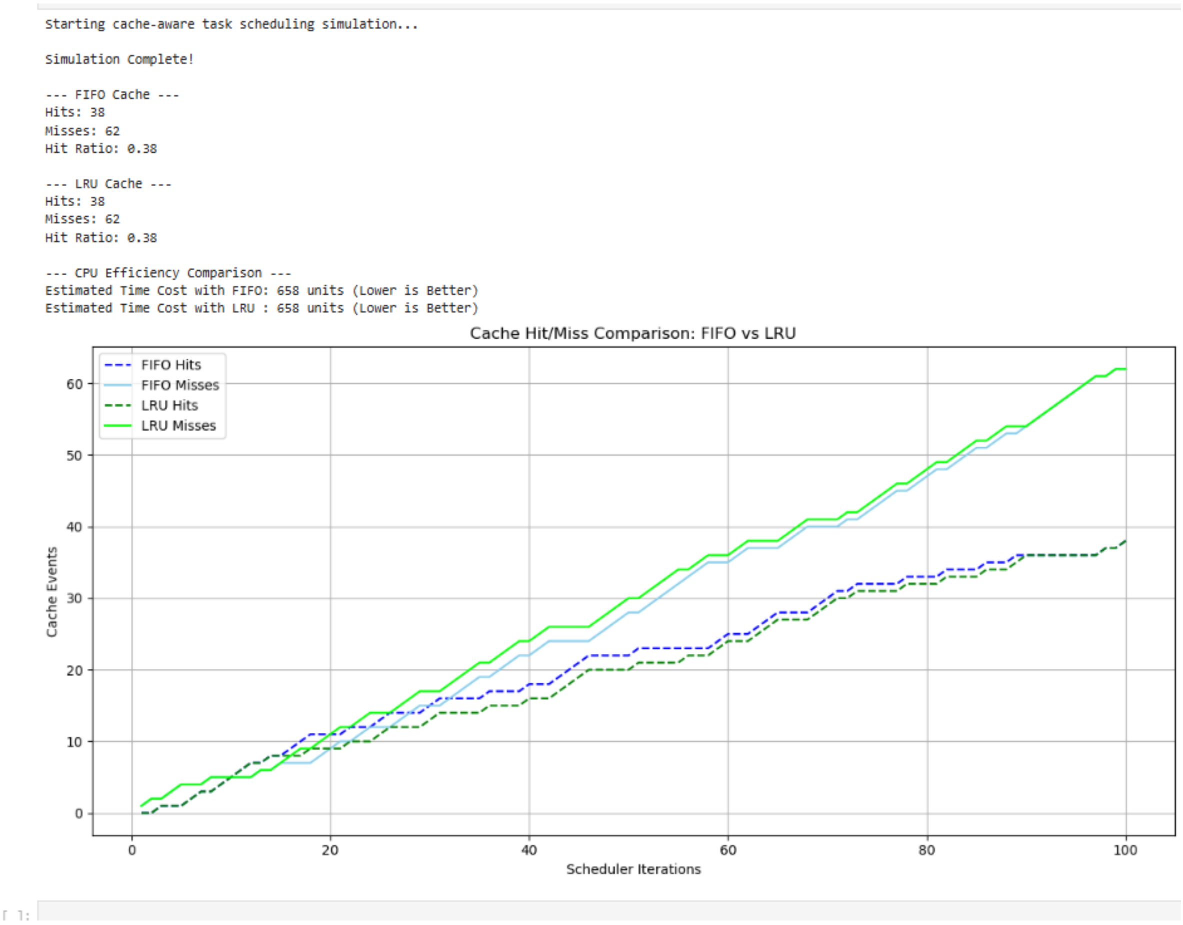
CPU task scheduling affects cache locality (keeping frequently used data in cache). Implement a task scheduler that optimizes CPU cache utilization. Use a priority queue to schedule tasks while maximizing cache hits. Expected Outcome: A program that simulates task scheduling with cache-aware strategies. Improved CPU efficiency by reducing cache misses. Comparison of FIFO and LRU scheduling policies

SOLUTION FOR THE GIVEN CODE

1) MAIN CODE:

```
import heapq

import random

import matplotlib.pyplot as plt

from collections import deque, OrderedDict


class Task:

    def __init__(self, task_id, access_frequency):

        self.task_id = task_id
        self.access_frequency = access_frequency

    def __lt__(self, other):
        return self.access_frequency > other.access_frequency  # max-heap behavior

class CacheFIFO:
    def __init__(self, capacity):
        self.cache = deque()
        self.capacity = capacity
        self.cache_set = set()
        self.hits = []
        self.misses = []
        self.total_hits = 0
        self.total_misses = 0

    def access(self, task_id):
        if task_id in self.cache_set:
            self.total_hits += 1
        else:
            self.total_misses += 1
            if len(self.cache) >= self.capacity:
                removed = self.cache.popleft()
                self.cache_set.remove(removed)
            self.cache.append(task_id)
            self.cache_set.add(task_id)
        self.hits.append(self.total_hits)
        self.misses.append(self.total_misses)

class CacheLRU:
    def __init__(self, capacity):
        self.cache = OrderedDict()
        self.capacity = capacity
        self.hits = []
        self.misses = []
        self.total_hits = 0
        self.total_misses = 0

    def access(self, task_id):
        if task_id in self.cache:
            self.total_hits += 1
            self.cache.move_to_end(task_id)
        else:
            self.total_misses += 1
            if len(self.cache) >= self.capacity:
                self.cache.popitem(last=False)
            self.cache[task_id] = True
        self.hits.append(self.total_hits)
        self.misses.append(self.total_misses)


def simulate(task_count=20, cache_size=5, scheduler_iterations=100):
    tasks = [Task(task_id=i, access_frequency=random.randint(1, 10)) for i in range(task_count)]
    pq = []
    for task in tasks:
        heapq.heappush(pq, task)

    fifo_cache = CacheFIFO(cache_size)
    lru_cache = CacheLRU(cache_size)

    print("Starting cache-aware task scheduling simulation...\n")
    for _ in range(scheduler_iterations):
        task = heapq.heappop(pq)
        task_id = task.task_id

        fifo_cache.access(task_id)
        lru_cache.access(task_id)

        task.access_frequency = max(1, task.access_frequency - random.randint(0, 2))
        heapq.heappush(pq, task)
    print(f"Simulation Complete!\n")
    print(f"--- FIFO Cache ---")
    print(f"Hits: {fifo_cache.total_hits}")
    print(f"Misses: {fifo_cache.total_misses}")
    print(f"Hit Ratio: {fifo_cache.total_hits / (fifo_cache.total_hits + fifo_cache.total_misses):.2f}")
    fifo_efficiency = fifo_cache.total_hits * 1 + fifo_cache.total_misses * 10

    print(f"\n--- LRU Cache ---")
    print(f"Hits: {lru_cache.total_hits}")
    print(f"Misses: {lru_cache.total_misses}")
    print(f"Hit Ratio: {lru_cache.total_hits / (lru_cache.total_hits + lru_cache.total_misses):.2f}")
    lru_efficiency = lru_cache.total_hits * 1 + lru_cache.total_misses * 10

    print("\n--- CPU Efficiency Comparison ---")
    print(f"Estimated Time Cost with FIFO: {fifo_efficiency} units (Lower is Better)")
    print(f"Estimated Time Cost with LRU : {lru_efficiency} units (Lower is Better)")

    steps = list(range(1, scheduler_iterations + 1))
    plt.figure(figsize=(12, 6))
    plt.plot(steps, fifo_cache.hits, label='FIFO Hits', linestyle='--', color='blue')
    plt.plot(steps, fifo_cache.misses, label='FIFO Misses', linestyle='-', color='skyblue')
    plt.plot(steps, lru_cache.hits, label='LRU Hits', linestyle='--', color='green')
    plt.plot(steps, lru_cache.misses, label='LRU Misses', linestyle='-', color='lime')
    plt.xlabel('Scheduler Iterations')
    plt.ylabel('Cache Events')
    plt.title('Cache Hit/Miss Comparison: FIFO vs LRU')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

# Run the simulation
simulate()
```

2) SCREENSHOT OF THE EXPECTED OUTCOME



```
Starting cache-aware task scheduling simulation...

Simulation Complete!

--- FIFO Cache ---
Hits: 38
Misses: 62
Hit Ratio: 0.38

--- LRU Cache ---
Hits: 38
Misses: 62
Hit Ratio: 0.38

--- CPU Efficiency Comparison ---
Estimated Time Cost with FIFO: 658 units (Lower is Better)
Estimated Time Cost with LRU : 658 units (Lower is Better)
```