GROUP 19

E/20/100

E/20/280

In the second part of Lab 06, the data memory hierarchy for the CPU was modified by adding a The modification of the data memory hierarchy by adding a data cache module significantly impacts the performance of the CPU in terms of data access delays. Here's an analysis of the performance implications based on the given details:

## ***STALLING TIME***

### Cache-less Data Memory Setup

- **Delay for every single access**: 40 clock cycles
- **CPU stalls for each access**: 5 clock cycles

### Data Cache Setup

- **Cache Hit**: Data served within the same clock cycle, no CPU stall.
- **Cache Miss (Non-dirty)**: 23 clock cycles delay.
- **Cache Miss (Dirty)**: 43 clock cycles delay.

## ***Performance***

### Cache-less Setup

- The consistent delay of 40 clock cycles per access and an additional 5 clock cycle stall per access means that the performance is predictable but consistently slow.

### Cache Setup

- **Cache Hits**: Extremely efficient, as the data is served immediately without any delay or CPU stall.
- **Cache Misses**:
    - **Non-dirty Miss**: A delay of 23 clock cycles, which is better than the cache-less delay (40 clock cycles) but still significant.
    - **Dirty Miss**: A delay of 43 clock cycles, which is worse than the cache-less delay

\
***ADDITIONALLLY***

Strategies to Optimize Performance

1. **Adjusting Block Size**: By choosing an optimal block size that matches the CPU's access patterns, the hit rate can be improved, leading to better overall performance.
2. **Cache-friendly Code**: Writing code that takes advantage of the cache's loading mechanisms can reduce miss rates. This involves:
    1. Spatial Locality: Accessing data locations that are close to each other within a short period.
    2. Temporal Locality: Reusing data that has been accessed recently.

## OVERALL

The inclusion of a data cache introduces a balance between high efficiency during cache hits and potential performance degradation during cache misses. The key to leveraging the benefits of the cache lies in managing the hit rate through careful block size selection and writing cache-friendly code. By doing so, the overall performance can surpass that of the cache-less setup, provided the miss rate is kept low.