```
import multiprocessing

def matrix_multiply_mapper(row, col):
    result = 0
    for i in range(len(row)):
        result += row[i] * col[i]
    return result

def matrix_multiply_worker(args):
    row_index, row, columns = args
    return [(row_index, col_index, matrix_multiply_mapper(row, col))
            for col_index, col in enumerate(columns)]

def matrix_multiply_reduce(results):
    final_result = {}
    for row_index, col_index, value in results:
        if row_index not in final_result:
            final_result[row_index] = {}
        final_result[row_index][col_index] = value

    return final_result

def map_reduce_matrix_multiply(matrix1, matrix2):
    num_workers = multiprocessing.cpu_count()
    pool = multiprocessing.Pool(processes=num_workers)

    args = [(i, matrix1[i], matrix2) for i in range(len(matrix1))]
    intermediate_results = pool.map(matrix_multiply_worker, args)

    pool.close()
    pool.join()

    final_result = matrix_multiply_reduce(
        [item for sublist in intermediate_results for item in sublist])

    return final_result

if __name__ == "__main__":
    matrix1 = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ]

    matrix2 = [
        [9, 8, 7],
        [6, 5, 4],
        [3, 2, 1]
    ]

    result = map_reduce_matrix_multiply(matrix1, matrix2)
    for row_index, row in result.items():
        print(row)
```

```
{0: 46, 1: 28, 2: 10}
{0: 118, 1: 73, 2: 28}
{0: 190, 1: 118, 2: 46}
```