JoSDC'24 Final Benchmarks

Benchmark 1

Matrix Multiplication

Author: Abdalrahman Alshannaq

1. Problem Description

Matrix multiplication is a fundamental operation in linear algebra, computer science, and engineering. It involves computing the product of two matrices, resulting in a new matrix that combines the information from both input matrices in a structured way. This operation is widely used in areas such as graphics processing, machine learning, scientific computing, and optimization problems.

Understanding the Problem

Given two matrices:

- Matrix A with dimensions $(m \times n)$
- Matrix B with dimensions $(n \times p)$

The goal is to compute their product Matrix C, where:

$$C = A \times B$$

- The resulting matrix C will have dimensions $(m \times p)$, where each element $C[i][j]$ is obtained by computing the dot product of the ith row of A and the jth column of B.
- For matrix multiplication to be valid, the number of columns in A must be equal to the number of rows in B (i.e., n must be the same in both matrices).

Steps to Solve the Problem

1. Verify the Dimensions:
    - Ensure that the number of columns in A matches the number of rows in B. If not, matrix multiplication is not possible.
2. Initialize the Result Matrix:
    - Create an empty matrix C of size $(m \times p)$ to store the computed values.
3. Compute Each Element of C:
    - For each element $C[i][j]$, compute the dot product of the ith row of A and the jth column of B.
    - This involves multiplying corresponding elements and summing the results:

$$C[i][j] = \sum_{k=1}^{n} A[i][k] \times B[k][j]$$

2. Matrix Multiplication Algorithm

To compute the product of two matrices, we iterate through each row of the first matrix and each column of the second matrix, calculating the dot product for each element in the result matrix. The following pseudocode and C++ implementation illustrate this process.

Pseudocode

```
Function MatrixMultiply(A, B):
    m = rows in A, n = columns in A (rows in B), p = columns in B
    Initialize result matrix C of size (m × p) with zeros

    for i from 0 to m-1:         // Iterate over rows of A
        for j from 0 to p-1:     // Iterate over columns of B
            sum = 0
            for k from 0 to n-1:  // Compute dot product
                sum = sum + (A[i][k] * B[k][j])
            C[i][j] = sum         // Store result in C

    return C
```

C++ Code

```
// Iterate over each row of matrix A
for (int i = 0; i < rowsA; ++i) {
    // Iterate over each column of matrix B
    for (int j = 0; j < colsB; ++j) {
        // Compute dot product of row A[i] and column B[j]
        for (int k = 0; k < colsA; ++k) {
            C[i][j] += A[i][k] * B[k][j] // multiply and add to result
        }
    }
}
```

3. Specifications and Complexity Analysis

3.1 Storage Complexity

Data Memory

This benchmark processes test data where both matrices, A (36×36) and B (36×36), contain 1296 elements each, with each element stored as a single word in data memory. The resulting matrix C (36×36) also requires the same storage, leading to the following memory requirements:

- Matrix A: 1296 words
- Matrix B: 1296 words
- Matrix C: 1296 words
- Total Required Storage: 3888 words

To ensure simplicity in memory allocation, the data memory specification is set to 4KW (4096 words), which provides sufficient space for matrix storage while allowing for potential additional data if needed.

Memory Layout

For easier access and structured storage, matrices should be stored in the following order:

Matrix	Start Address	End Address
A	0	1295
B	1300	2595
C	2600	3895

Alternatively, if your assembler supports automatic variable allocation, ensure that proper offset handling is implemented based on your memory allocation strategy.

Test Data Used

The values for matrices A and B are randomly generated within a specified range (1, 50). These values must remain unchanged, as modifying them will affect the execution time and fairness of evaluation.

A is:

```
  5  18   3   1  23  30  29  50  38  41  14  16  42   1  19  20  39  12  18  37   7  31  10   5  33  31  20  49  49  27  14   3  20  42  37  50
 37  42  45  24  16  22  20  50  14   1  48  36  25  47  48  15   6  43   2  38   7   7  47   8   1  26  49  13  34   7  41  32  37  33  23  35
 10  47  28  25  21  11  40  19  49  39   8  26   2  19  40  28  47   8  32   2  31  15  20  27  11  49   4  19  43  41  32  30  16   5  44  33
  9  33  26  22  28   2  44   7  11  18  41   3  39  16  33  48   2  32  17   7  10  27  43   6   3  43  49  36  10   7  13  47   6   8  28  24
 34  23  37  48  50  34  43  24   3  50  35  22   9  37  39  33   7  23  10   6  33  23   9  28  20  15  22  14  22  30   5  16  16  40  17  12
 47   1   9  24  39  34  26  13  10   8  11  10  30  48   3   2  28  43  19  34   4  29  47  12  18  46   9   1  40  28  48   8  33   2  29   2
 13  44   5   2  50   9  44   5   1   8  43  26  20   8  30  37  37  19   6  14  34  38  17  25  24  17  38   9  38  27  26  14   7   8  41   1
 35  25  11  11  44  39  22  41  19   8  29   6   7  38   8   2  45  10  11  16  48  35  34  11  25  16  39  34   2   9  31  34  39  43   8   4
 49  14  25  16  20  19  34  11   3  48  19   7   8   1  32  45  41  19  24  26  50  12   7  26  28  48  28   2  18  39  18  48  36  47  11   1
 49   2  10  47  48  41  26  34  27  35  29   9  17  39  37  44  41  47  31  30  49  13   5  16  10  39   3  44   9  18  13  40  16  24  24  20
 10  50  12   1  24  44  38  35  37  20   9  29  41  39  35   4   5  48  17  33  17  40  29  29  39  26   2  40  43   5   7   1   9  45  44  39
 18  23  21  11  27  38   1  29   3  23  16  32  28  13  36   6  45  10   3  23  24  43  10  50   5  47  38  37  14  43  21   3  10  28   1  21
 16  25  12  47   5  40  29  20  41  37   9  15  10  35   8   7  18  25   8  50  40  28  32  50  38   5   8  47  22  11  34  25  42  28   1   4
 32  35   1  28  33   7  29  16  15   4  31   6  19  33  40  47  37   7  48  44  17   8  34  33  33  31  25  13  31  21  33   9  38  35   3  37
 16  43  37  14   6  13  21   6  41  47  10  40  14  18  32   2  32  32  13  40  27  31  11  35  21   7  47   6  27  23  10  44  43   3  24  24
 50  22  28  22  32  48   7  46  42   9  25  15  45  25  38  35  20  29  27  21  20  48  24  12  34   4   9  22  20  29  33   2  46  30  14  31
 24   8  20  37  38  35   4  22  38  29   4  35   6  39  12  27  13   4  34   7  44  15  37   5  33   1  31  39   5   7  48   3  26  47  18  33
 20  22  22  38  37  30  34  31  33  33  45   5  20  26  36  34  22  31  29  23   2  36  46  32   6   8   4   5  32  26  20  32  38  45  40  21
 38  24  25  45  13   1  13  50  27  16  35  28  33  16   7  17  13   5  41  23  27  27  33  30  37  39  12  24  21  17  27  29  31   5  12  43
 22  36  29  41  17  46  38   5  27  34  48  40  32  12   7  21  29  22  22  32  33  35  22  15  35  41  24  27  44  40  22  36  37  19  11  43
 42   8  44   8  32   3  27   7  28  35   9   5   1  44  20  47  43  50  41   2  41  35  12  28  17  46   6  13  32  16  18  22  17  13  12  39
 13  12  33  41  29  17  18  39  39  43  16   4  25   4   9  14   3  37  49   3  25  31  39  40  48  29  30   4   7   8  21  17  28  34  39  30
 18  10  11  27  46   9  36  22  36  40  13  36  14   9  48  20   2  37  21  46  26  49  43   8  40  13  11  49  39  31  18  40  26  46  25   4
 30  45   1  40  28  34  28  20  18  28  29  43  24  25  23  38  36  16  19  19  31  14  43   9  22  33  19  30  29  16   3  33  45  32  35  18
  4  50  17   8   5  11  38  12   7  43  26  50  34  46  10  44   6   3  19  48  35  42  17   3   9  44  30   5  45   1  23  30  36   7  39   9
 34  27  26  43  23  27  17  29  23  50  15   5  26   5  18  41  50   9  10  45  38  23  12  20  17  15  23  48   9  25  42  20  29  30  38   2
 38  46  25  32   2  49  37  20   1   3  22  30  43  10  33  30  12   5  48   9   9  15  16  48   3  26  27  24   9  45  38  49  20  28  17  11
 27  32   1  33   6  13   7  16  33  21  50  33  22  36  43  13   8  38  41  42  22   5  40   5  32  18  35  16  36  31  20  31  29  39   9  32
 14  21  19  27  26  21  15  38  40   7  44  36  15  29  28  50  28   8  37  21  35  13   9   3  48  36  18  17  40  30  18   1  16   1  35  40
  6  41  19   5  39   6   4  41  24  31  12  50   3  18   5  32  34  47  33   3  38  43  16  21  42  39  11  22  15  10  31   2  43   5  47   8
  7  45  50   4  19  30  21  38  10   5  31  44  17  26  35  21  18  39  47  25  31  14  11  31  33  14  11  31  39  15   7  32  17  37  19  14
 12  44   8  18  42  23   4  47  14   6  20  38  34  16  32  24  21  11  13  32  41  27  38  18  42  30  12  42  19  32   1  34   1  45  13  20
 47  49   9  37  45  47  47  46  12  14  34  37  40  36   6   4  23  50  46  14  50  30   4  24  16  32  29  26  26  40   3  42  50  11  30  38
  5  12  33  19  30  32  22  15  48  47   1  24  29  25  12  19  47  31  10  19  42  44  27  10  49  42  35   3  25  42  33  22  43  20  39  24
 28  34  39  50  38  12  50  41  11  23  33  41  28  49  20  22  46  43  42  37  47  49  38  38  48  44   3  20  21  21  45   4  48   8  39  50
 18  28  10  23  33  10   3  20  39  39  43  25  28   9  28  37  26  31  17  39  25   3  50   9   2   9   8  49  31  39  43  47  45  32  50   8
```

B is:

```
 11  46  33   2   7  26  14  31  19  24  15  28  28  22  17  35  46  28  27  44  46   6  24  10  13   6  35  19  12  45  43  21  41  36  11  17
 27  38  27  40  13  40  21  48  23  49  24   5  31  15  43  25  13   9  50  18  50   3   4  24  19  27  20   9  35  50  34   1  23  39  45  45
 24  11  23  40  35  38  47  47   3  26  27  10  27  17  44  43  28  44  14  30  24  23  34  49  21  27   6  22   5  29  10  33  14  13  37  47
 48  37  38  20  26  28  22  10  30  33  27  42  35  17  36  47  46  35  50  43  12  47  34  42  30  11   3  20  25   8  21  35  18  29   3  40
 23  11  24  20  46  10  42  40  17  40  15  37  36  43  20  11  30  38   3   1  19  27  28  19  33  25  25  36  11  50   7   6  43  13   4   7
 14  43   5  49  30   4  35  50  48   6  37  31   3  33  22  13  46   1  48  33  48  17   2  24  30  25  23  43   2  45  38   7  20  41   5  20
 21   4  23  32  29  14  19  21   3  38   6  12  14  32  48  30   7  19  19   1  39  45  46  37  32  19  16  40  35  35  24   2  36   4  43  27
 13  11  41   3  22  24  44  19  27  17  21  38  18   7  41  17  20   9  47  21  46  34  45   4  30  44  37  37   9   6  22  38  48  13   1  39
 28  35  49   5  17  28  18  32  19  41  33  46   5  13  20  36  41  12  42  21  19  22  20  13  50  28   8  14  46  35  39  49  45  17  41  33
 38   1  30  45  14  50  27  26  40   9  50  20  29  23  30  29   6   9  26  11  39  43  20  45   5  48  17  21  10  17  15  43  22  47   6   8
  5  17  33  39   3  23  44  40  21  28  10   8   6  48  34  10  18  28   5  24  19  44   8  18   5  23   8   5   1  12  36  46  28  30  34  45
 22  28  35  38  27  21  18  48  16  39  27  47  49  47  34  28  32  33  22  47  50  39  41  20   8  42  47   9  17  11  20  32   8   1  14  39
  5   1  50  23  18  28  14  21   6  48  16  37  21  11   9  13  45   7  35  27  19  23  46  29   4   2  20  18  18  43  44  40  47   4  11  12
  9   4  27  46  29  35  35   8  42   9  30  10  38  38  27  50  49  26  40  49   6   1  41  12  30  36  14  21  25  50  45  12  15  23  39  42
 11  45  50  21   4   1  19  32  43  15  32  36  45  10  34  38  35   3  10   5  47  19  17   5  13  14  35  40  48  26  30  30  36  38  25  41
 30  39  15  34  26  22  40  46  11   2   1  44  41  11  14  48  27  16  40  26  15  40  24  48  34  11   5   3  30  26  20   7   8  50  15  47
 29  45  12  36  13  15  26  12  32  33  11   1  24   5  43  15  19  28  43  46  49  26  42  19  41  22  44   3  44  13  43  32   1  44  32  39
 37  18  49  13  40   7  20  40  43  49  41  43  26   7  33  25  33  40  43  47  27  29  20  26  23  28  10   7  50  26  47  16  27  30  15  11
 37  31  47  18  18  32  11  33   4  37  24  50  44   7  50  50  18  19  41  33  30  45  34  44  35  24  39  23  38  24  30  35  37  32   7  12
 18  24  16  11   5   4  21  18  37  21  30   2  50  48  28  44  17   3  44  33  49  22  15  29   8  11  26   8  13  32   8  31  11  48  13  39
 43  35  34  17  32   7  41  41  22  45  23  37   1  26  24  48  48  38  42  13   6  37  50  28  36  15  11  29  30  30  13  37  20  44  32  33
 34  33  13  16  32  22  24   7  14  10  31  23  22  14  44  31  22   2  15  44  32   6  19  36   5  50   3  33  28  14  27  48  12  31  19  17
 26  15   7  23  38  19  30  22  10  39  30  19  39  23  39   8   4  48  46  46  20  41  29  23  19  15  34  50  50  40   5  32  33  26  19  50
 19   2  12  31  49  47  40  26  41   9  44   9   8  39   4  27   1  21   6  10  18   1  10  43  32  15  40  27   2  12  48  31   3  14  40  40
  9  34   1  25  31   1  13   8  44  11  26  31  46  11  29  45  21  31   5   9   6  18  19  47   6  48  24  36  18  13  17  42  41  19  15   3
 19   9  20  13  32  19  22  16  49  39  11  19  35  37  40   4  24  29  41  44   6   1  17  15  42  25   4  46  31   8  41   6  40  17   8  43
 41  33  35  36  36  15  45   6  22  30   9  18  13  50  49  21  10  32  18  13  18  30  39  23  45  26  12  15   6  43  49  46  17  15  28  11
 34  38  24  23  39  16   1  14  37  47   9  21  10  46  31  35   4  43  11   1  47  35  18  13  30  41  23  13  11  25   6  16  49  42   7  30
 42  45  26  37  24  28  10  35  29  50  29  31   7   4  21  40  21  49   6  50  40  19  11  33   4  28  48   8  23  15   3   1   4  12   1  24
  2  21  16  11  39  17  46  46  34  31  10  29  45  37  32  21  46  39  42   2   8  29  21  22  44  48  15  22  29  27   3   7  31  18  31  47
  9  42  19  21  22   3  41  15  49   1   7  48  22  30  46  31  36  48  20  34  32  35  19  20   4  38  47  16  35   5  32  35  46  25  13   6
 44  47  21  43  12  34  35  30  46  41  47   7  17  43  46   4  11  38  50   5  49  44  22  22  24  14  40   7  17  50  49  14  36  47  43  15
  8  30   9  12   6  14  44  14  26  42  39   1  47  44  47  43  33  17  42  42  36   2  12  45   5  20   4   4  48  19  12  33  16  17  46  16
 44  19  41  29  22  10  14  36  10  20  43  33  20  13  28  39   9  28  17  25  30  28  47  26   2  36  37  25  27  14  19  12  13  33  15  41
 15  48  10   1  35   8  25  50  12  43  42  35  18  36  11  13   7  37  39  47  45  24   4  45  32  23  47  29  17  36   5  18  10   8  22   1
 36  50  42  30  23  19  19   1  13   6   8  11  50   5  44  40  43   9  36  29   2  33  33   6  10  44  18   3  47  34  23  15  13  41  11  38
```

Instruction Memory

The benchmark consists of a total of 54 MIPS instructions. However, since two pseudo-instructions are used, and each is expanded into two hardware instructions, the final count becomes:

- Total Hardware Instructions: 56
- Storage per Instruction: 1 word

Instruction Memory Requirement

Since the benchmark is relatively small in size, a 256-word instruction memory is more than sufficient for execution.

Notes for Assembly Processing

- If your assembler or compiler modifies the benchmark code (e.g., adding NOP instructions, reordering, or optimizing), you must clearly document the changes and their impact.
- You are NOT allowed to remove SLL (Shift Logical Left) instructions used for address adjustments, even if they appear unnecessary in your implementation. This ensures a fair comparison across all teams in the competition.

3.2 Time Complexity Analysis

Benchmark Loop Structure

The core of the benchmark is a triple nested loop, where the innermost operation performs element-wise multiplication and accumulation. However, since 'mul' instruction is not supported in the competition instruction set, the multiplication is implemented using addition-based multiplication function (Mul_Fun), thus, additional loops are introduced inside the third loop.

Breakdown of Steps in the Innermost Loop

Since MIPS handles 2D arrays as 1D arrays, indexing requires computation based on:

$$Index = offset + (row \times numOfCols + col)$$

Where:

- row is the row index in the 2D array
- col is the column index in the 2D array
- numOfCols is the total number of columns in the 2D array.

Each iteration of the third loop involves:

1. Finding $A[i][k]$ → Computing its index using Mul_Fun, and retrieving the value.
2. Finding $B[k][j]$ → Computing its index using Mul_Fun, and retrieving the value.
3. Finding $C[i][j]$ → Computing its index using Mul_Fun, and retrieving the value.
4. Performing multiplication → Using Mul_Fun, which consists of an internal loop.

Since Mul_Fun, performs multiplication using addition, it introduces 4 additional linear loops, each iterating based on the values being multiplied, making the total number of iterations for each of these 4 loops unpredictable.

Hierarchy of Loops in Execution

```
LOOP1 {   // Iterates over rows of A (36)
    LOOP2 {   // Iterates over columns of B (36)
        LOOP3 {   // Iterates over columns of A (36)
            LOOP4 {}   // Iterates based on number X being multiplied
            LOOP5 {}   // Iterates based on number Y being multiplied
            LOOP6 {}   // Iterates based on number Z being multiplied
            LOOP7 {}   // Iterates based on number W being multiplied
        }
    }
}
```

Final Time Complexity

Since the number of iterations in LOOP4 to LOOP7 is data-dependent (X, Y, Z and W are random values), the overall complexity is:

$$O\ (rows\ A \times cols\ B\ \times cols\ A\ \times (random\ X + random\ Y + random\ Z + random\ W))$$

This makes the exact execution time unpredictable, as it depends on the random values used in multiplication.

4. Related Attached Files

The benchmark directory includes the following files, each serving a specific purpose in the project:

File Name	Functionality
MatrixMultiplication.asm	Contains the MIPS assembly code, structured with guiding comments to ensure clarity and ease of understanding
definition.txt	Stores the first three lines of MIPS code, responsible for data memory allocation and initialization of arrays A, B, and C
Matrix.txt	Provides the values of matrices A and B, arranged in a clear 2D format for traceability and visualization
dataMemoryWORD.mif	A memory initialization file for those using word addressing mode, well-organized for easy use
dataMemoryWORD.hex	A memory initialization file using hex format based on word addressing mode
expectedResults.txt	Provides expected result of matrix C, arranged in clear 2D format

Important Notes

- If you are using a different addressing mode, adjust your memory initialization file accordingly.
- Ensure you provide your machine code file with execution results, along with any other required files for benchmark processing and project evaluation.