Benchmark: “Sparse Matrix Count”

Author: Issa Qandah *Reviewed by: “Hassan TaqiEddin”*

Description & Notes

- The benchmark counts the number of zeros in a matrix by iterating through each element.
- calculating the address dynamically for each element.
- The zero counter is incremented whenever a zero is encountered in the matrix.
- Time Complexity O(N^2).

Algorithm (Pseudo or C)

```
Initialize zero_counter = 0
For each row_index from 0 to num_rows - 1:
    row_base_address = matrix_base_address + (row_index * num_cols * 4)
    For each col_index from 0 to num_cols - 1:
        element = MEM [row_base_address + (col_index * 4)]
        If element == 0:
            zero_counter += 1
Return zero_counter
```

Registers and memory used in implementation

$8 : zeros counter

$16 :number of Rows

$17 : number of Cols

$20 : Base address of the matrix

$9 : rowIndex

$10 : colIndex

$11 : tmp reg for SLT output

$13 : loop counter for multiplication

$14 : product accumulator (rowIndex * Cols)

$15 : address of matrix[rowIndex][colIndex]

$24 : matrix[rowIndex][colIndex]

Code (.data and .text)

```
.data
matrix:   .word 1, 0, 0, 0, 3, 5, 0, 0, 0, 0, 22, 0   # 4x3 matrix (row-major order)

.text
        # Initializations
        ADDI $8  , $0, 0            # Zero counter
        ADDI $16 , $0, 4             # Rows
        ADDI $17 , $0, 3             # Columns
        ADDI $20 , $0, 0             # Base address of the matrix
        ADDI $9  , $0, 0            # rowIndex

outerLoop:
        SLT  $11, $9, $16            # Check if rowIndex < Rows
        BEQ  $11, $0, exitOuter

        # Precompute row base offset: rowIndex * Cols
        ADDI $14, $0, 0              # Reset base offset accumulator
        ADDI $15, $0, 0              # Initialize loop counter

Multiply:
        # Multiply rowIndex * Cols by repeated addition
        SLT  $11, $15, $9
        BEQ  $11, $0, endMultiply
        ADD  $14, $14, $17           # Accumulate: base offset += Cols
        ADDI $15, $15, 1             # Increment loop counter
        JAL  Multiply

endMultiply:
        # Choose one of these Insertion based on your memory
        # For Word addressable             # For byte addressable
        # ADD $14, $14, $0                   SLL  $14, $14, 2
        ADD  $14, $14, $20           # Base address for current row
        ADDI $10, $0, 0             # colIndex

innerLoop:
        SLT  $11, $10, $17           # Check if colIndex < Cols
        BEQ  $11, $0, exitInner
        # Choose one of these Insertion based on your memory
        # For Word addressable             # For byte addressable
        # ADD $15, $10, $0                 SLL  $15, $10, 2

        ADD  $24, $14, $15           # Base row address + column offset
        LW   $24, matrix($24)            # Load element: matrix[rowIndex][colIndex]
        BNE  $24, $0, notZero        # Check if element != 0
        ADDI $8, $8, 1              # Increment zero counter if element == 0

notZero:
        ADDI $10, $10, 1             # Increment colIndex
        JAL  innerLoop

exitInner:
        ADDI $9, $9, 1              # Increment rowIndex
        JAL  outerLoop

exitOuter:
        NOP
```

Expected Output

```
$8 = 0x0008     # number of zero elements in the matrix
```