Benchmark: “Selection Sort”

Author: Abdalrahman Alshannaq *Reviewed by: “Hassan TaqiEddin”*

Description & Notes

- Selection Sort Algorithm
- Scalable by changing size of array ($13) and changing elements in data memory
- Support both Word and Byte addressing modes
- Benchmark Complexity is $O(N^2)$
- Current Array Size is 20 Element
- Array must Be sorted on first memory locations (starting from 0x00), or use Arr as offset

Algorithm (Pseudo or C)

given array consists of 10 numbers, sort the array using selection sort algorithm, pseudo code as follow:

```
for i from 0 to 10
        do for j from 0 to 10
                do if Arr[j] > Arr[i]
                        do swap (Arr[i], Arr[j])
```

Registers and memory used in implementation

$13: Array Size

$11: counter i

$12: counter j

$21: byte-addressable memory location for i

$22: byte-addressable memory location for j

$10: temp register for branch and SLT instructions

$8: Arr [i]

$9: Arr [j]

Code (.data and .text)

```
        .data:
                Arr: .word 0x5, 0x7, 0x2, 0xF, 0xA, 0x10, 0x30, 0x1, 0xFF, 0x55,
0x0, 0x6, 0xAB, 0xAD, 0x99, 0x33, 0x1, 0x16, 0x22, 0x79

        .text:
                ADDI $13, $0, 20
                ADD  $11, $0, $0

LOOP1:          XOR $21, $11, $0
                # this line is commented, use it for byte addressing memory
                # SLL $21, $11, 2
                ADD  $12, $0, $0

LOOP2:          XOR $22, $12, $0
                # this line is commented, use it for byte addressing memory
                # SLL $22, $12, 2
                LW $8, 0x0($21)
                LW $9, 0x0($22)

IF:             SLT $10, $8, $9
                BEQ $10, $0, ENDIF
                ADD $3, $8, $0
                ADD $8, $9, $0
                ADD $9, $3, $0
                SW $8, 0x0($21)
                SW $9, 0x0($22)

ENDIF:          ADDI $12, $12, 1
                SLT $10, $12, $13
                BNE $10, $0, LOOP2
                ADDI $11, $11, 1
                SLT $10, $11, $13
                BNE $10, $0, LOOP1
```

Expected Output

First 20 locations in memory contains the array sorted in ascending order.