Execution Unit Simulation Report

1. Loading Values into Registers

a) Load 141 to R4:

b) Load 208 to R6:

c) Load 32 to R8:

Values were loaded into specific registers using load operations. Values such as 141, 208, and 32 were loaded into registers R4, R6, and R8, respectively, to initialize their contents for further computation.

Code snippet to Load:

```
data_in = xx;   address = xx;  write_en = 1; #10; write_en = 0;  #10;
```

2. Moving Values Across Registers

a) Move R4 to R5:

b) Move R8 to R9:

c) Move R6 to R7:

Values from registers R4, R8, and R6 were moved to registers R5, R9, and R7, respectively, using data transfer operations with the help of data_in and data_out lines to the registers.

Code snippet to Move:

```
address = 4; read_en = 1; #10;  address = 5; write_en = 1; data_in =
data_out;   read_en = 0;  write_en = 1; 10;
```

Arithmetic Operations on Registers:

2'b00: begin // ALU Addition

2'b01: begin // ALU Subtraction

2'b10: begin // ALU Comparison

2'b11: begin // ALU Increment

3. R4 + R6 => R10:

Registers R0 and R1 are mapped as inputs for addition operation, enabling the module to perform calculations. Register R2 is mapped as the output register, where the result of ALU operations is stored. The value of R10 and carry were checked. An additional operation was performed between registers R4 and R6, and the result was stored in register R2 then it is moved R10.

4. R6 - R8 => R11:

Now, R0 and R1 are mapped as inputs for Subtraction operation, enabling the module to perform calculations. Register R2 is mapped as the output register, where the result of ALU operations is stored. Subtraction operation was performed between registers R6 and R8, and the result was stored in register R11. The value of R11 and borrow were checked.

5. R8 - R4 => R12:

Again, R0 and R1 are mapped as inputs for Subtraction operation, enabling the module to perform calculations. Register R2 is mapped as the output register, where the result of ALU operations is stored. Subtraction operation was performed between registers R8 and R4, and the result was stored in register R12. The value of R12 and borrow were checked.

6. INC R12:

Increment operation was performed on register R12, then the value is stored in the same R12 address, and the result was checked.

7. R4 ≥ R6?:

Registers R4 and R5 are mapped as inputs for comparison operations, specifically evaluating whether the value in R4 is greater than that in R5. Comparison operation was performed between registers R4 and R6, and the flag was checked to determine if R4 is greater than or equal to R6.

8. R4 - R4 => R13:

R0 and R1 are mapped as inputs for Subtraction operation, enabling the module to perform calculations. Register R2 is mapped as the output register, where the result of ALU operations is stored. A subtraction operation was performed between registers R4 and R4 (zero), and the result was stored in register R13. The zero flag was checked.

9. R4 + R5 + R6 + R7 + R8 + R9 => R13, check R13 with carry:

A loop was executed to add the numbers stored in registers R4, R5, R6, R7, R8, and R9, values are moved to the respective address R0 and R1 are mapped as inputs for Subtraction operation, enabling the module to perform calculations. Register R2 is mapped as the output register, with the result stored in register R13. The carry was checked.

10. Find Largest Number:

Registers R4 and R5 are mapped as inputs for comparison operations, specifically evaluating whether the value in R4 is greater than that in R5. The values are moved to the respective address R0 and R1 are mapped as inputs for Subtraction operation, enabling the module to perform calculations. Register R2 is mapped as the output register The largest number among registers R4, R5, R6, R8, R9, and R10 was found and stored in register R13.

Simulation Outputs: