10420 CS410001 – Computer Architecture 2016

Appendix D Error Detection Sample

The error handler should contain the following two functions.

1. Output error message if detect an error
   A. Open the "error_dump.rpt" file and write out the error message as specified in Table 1
2. Decide whether it should continue simulation.

Table 1

Error type	Continue or halt	Error message	Remark
Write to register $0	Continue	Write $0 Error	$0 is fixed to be 0
Number overflow	Continue	Number Overflow	The overflow is a number, too
Memory address overflow	Halt	Address Overflow	N/N
Data misaligned	Halt	Misalignment Error	N/N

- **Error Definitions**
  1. **Write to register $0**
     Register 0 is a **hard-wired constant 0**; any attempt to write to register 0 takes no effect.
     The error occurs when an instruction try to **write to the register $0.** Note that **NOP** instruction **(sll $0, $0, 0)** is the only exception for which no error is reported. When this error occurs, the error handler shall print out the "**Write $0 Error**" message to the file "error_dump.rpt" and do nothing at this cycle and then continue to simulate the next instruction.
     You may print out the error message using the following code:
     fprintf(file_ptr , "In cycle %d: Write $0 Error\n", cycle);

  2. **Number overflow**
     The error is a condition that occurs when a calculation produces a result that meet the situation described as follow:
     An addition overflow occurs if two same sign addends produce a sum of different sign.
     When this error occurs, the error handler shall print out the "**Number Overflow**" message to the file "error_dump.rpt", and then execute the instruction at this cycle and continue to simulate the next instruction.

     Notes:
     (1) The subtraction **a - b** is done as addition **a + (-b)**.
     (2) For the set of instructions that include signed addition/subtraction, ***add, sub, addi, lw, lh, lhu, lb, lbu, sw, sh, sb, beq, bne,*** you may print out the error message using the following code:
     fprintf(file_ptr , "In cycle %d: Number Overflow\n", cycle);

3. **Memory address overflow**

   The error occurs when **a D-memory access beyond the memory address bound.** When this error occurs, the error handler shall print out the "**Address Overflow**" message to the file "error_dump.rpt", and it should halt simulation.

   You may print out the error message using the following code:

   fprintf(file_ptr , "In cycle %d: Address Overflow\n", cycle);

4. **Data misaligned**

   The error occurs when the instruction try to **access misaligned data location in D-memory.** A modern computer reads from or writes to a memory address which is in multiples of **basic blocks** (i.e. **words/half words/bytes** in our case). *Aligned Data* is the data located at a **memory offset** in **multiples of basic blocks (words);** otherwise, it is a misaligned data.

   For example:

   lw $5 4($0) is aligned because the memory offset is 4 bytes (0+4) and is in multiples of **words.**

   lw $5 2($0) is misaligned because the memory offset is 2 bytes (0+2) and is **not** in multiples of **words**.

   lh $5 2($0) is aligned because the memory offset is 2 bytes (0+2) and is in multiples of **half words.**

   lh $5 1($0) is misaligned because the memory offset is 1 bytes (0+1) and is not in multiples of **half words.**

   When this type of error occurs, the error handler shall print out the "***Misalignment Error***" message to the "error_dump.rpt" file, and it should halt simulation.

   You may print out the error message using the following code:

   fprintf(file_ptr , "In cycle %d: Misalignment Error\n", cycle);

- **In project 1**, if multiple error occurs, detect errors in the following order.
  (1) Write To Register $0
  (2) Number Overflow
  (3) D-Memory Address Overflow
  (4) D-Memory Miss Align Error

- **In project 2**, if multiple error occurs, detect errors in the following order.
  (1) Write To Register $0
  (2) D-Memory Address Overflow
  (3) D-Memory Miss Align Error
  (4) Number Overflow