10220 CS410001 – Computer Architecture 2014

Appendix D-2 Error Detection Sample for Project 2

2014/04/07

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 overflowed 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

The error occurs when the instruction try to **write to the register $0**. 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 , "Write $0 error in cycle: %d\n", cycle);

(Note that there is a space after "in cycle: %d\n"):

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 overflows if the signs of the addends are the same and the sign of their sum is different from the one of the addends.

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) Each subtraction like ***a - b*** is done by an addition like ***a + (-b).***
In the case that b is 0x80000000 (-2^{31}), where –b cannot be represented, we report this error and still do ***a + 0x80000000*** anyway.

(2) 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 , "Number overflow in cycle: %d\n", cycle);

(Note that there is one space after "in cycle: %d\n")

3. **Memory address overflow**

   The error occurs when a memory **access (both I-memory and D-memory) 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 , "Address overflow in cycle: %d\n", cycle);

4. **Data *misaligned***

   The error occurs when the instruction try to **access misaligned data location.** 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 , "*Misalignment error *in cycle: %d\n", cycle);*

   (Note that there is one space after "in cycle: %d\n")

- When multiple error occurs, detect the set of error in the following order.
  (1) Write To Register $0
  (2) D-Memory Address Overflow
  (3) D-Memory Miss Align Error
  (4) Number Overflow
  (5) I-Memory Address Overflow
  (6) I-Memory Miss Align Error