**10120 410001 – Computer Architecture 2013**

**Appendix D Error Detection Sample**

The error handler should contain the following two functions.

1. Show an error message

A. Open the “Error\_dump.rpt” file and write out the error message as specified in Table 1

1. 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\_report.txt” 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 (Note that there is one space after “in cycle: %d\n”):**fprintf( file\_ptr , "**Write $0 error in cycle: %d\n**", cycle);

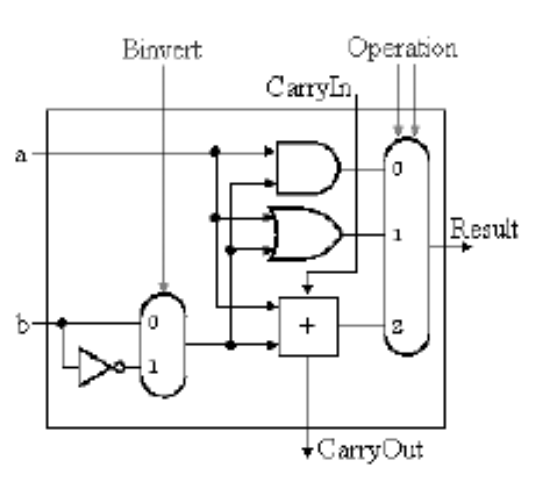
1. **Number overflow**

The error is a condition that occurs when a calculation produces a result that is **greater than** what a given **register can store or represent.** When this error occurs, the error handler shall print out the “**Number overflow**” message to the file “Error\_report.txt”, and then execute the instruction at this cycle and continue to simulate the next instruction.

**You may print out the error message using the following code (Note that there is one space after “in cycle: %d\n”):**fprintf( file\_ptr , "**Number overflow in cycle: %d\n**", cycle);

Note that the way to detecting overflow is different for signed and unsigned operations. Here we illustrate how unsigned operations detect overflows. Please refer to the textbook for signed overflow detection.

The following is a 1-bit ALU with four operations: and, or, add, sub. The full instruction operation “a-b” can be done by inverting the input b and set CarryIn of the least significant bit of the ALU to 1.



Assume there are two unsigned number A and B:

(A)2=an-1an-2an-3……a1a0

(B)2=bn-1bn-2bn-3……b1b0

and Ci is the carry out of ai-1 + bi-1:

(C)2=cncn-1cn-2……c1

For an addition of two unsigned numbers A+B , if Cn=1 , it indicates an overflow.

For a subtraction of two unsigned numbers A-B, If A<B and Cn=0, it indicates an overflow.

1. **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\_report.txt”, and it should halt simulation.

**You may print out the error message using the following code (Note that there is one space after “in cycle: %d\n”):**fprintf( file\_ptr , "**Address overflow in cycle: %d\n**", cycle);

1. **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 error occurs, the error handler shall print out the “**Misalignment error**” message to the “Error\_report.txt” file, and it should halt simulation.

**You may print out the error message using the following code (Note that there is one space after “in cycle: %d\n”):**fprintf( file\_ptr , "**Misalignment error in cycle: %d\n", cycle**);

**Priority**

* 1. Number Overflow
  2. Address Overflow
  3. Miss Align Error
  4. Write To Register $0