

Appendix D Error Detection Sample

The error handler should contain the following two functions.

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

Even if there are no errors, still create an empty “error_dump.rpt” file.

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
Overwrite HI-LO registers	Continue	Overwrite HI-LO registers	N/N
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=0x00000000**) 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 in 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 below:

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 in the file “*error_dump.rpt*”, but still execute the instruction at this cycle and continue to simulate the next instruction with the truncated result.

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, bgtz** you may print out error messages using the following code:

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

3. Overwrite Hi-Lo registers

A modern processor must avoid accidentally overwriting HI or LO registers before moving them to other general-purpose registers. As for our specification, only *mult* and *multu* write to these two special purpose registers.

The error occurs when there is no *mfhi* or *mflo* instruction in between two subsequent “multiply” (*mult* or *multu*) instructions. We assume that only partial multiply result is needed and hence the execution of either *mfhi* or *mflo* implies intended operation.

When the above error occurs, the error handler shall print out the “**Overwrite HI-LO registers**” message to the file “*error_dump.rpt*”, and continue the simulation.

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

```
fprintf(file_ptr, "In cycle %d: Overwrite HI-LO registers\n", cycle);
```

4. 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);
```

5. 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 **blocks** (i.e. **words/half words/bytes** in our case). *Aligned Data* is the data located at a **memory offset in multiples of 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 in 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);
```

Order of Error Print in the Projects

- **In project 1**, if multiple error occurs, detect errors in the following order.

- (1) Write To Register \$0
- (2) Number Overflow
- (3) Overwrite HI-LO registers
- (4) D-Memory Address Overflow
- (5) 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) Overwrite HI-LO registers
- (5) Number Overflow