THE UNIVERSITY OF
MELBOURNE

SWEN90010 High Integrity Systems Engineering
Assignment 3
Fault Report

Margareta Hardiyanti | Ivan Ken Weng Chee
852105 | 736901

Faults and Diagnosis

SPARK faults in VM ExecuteProgram function

SPARK Warnings and Errors	Causes	Consequences
24:40 medium: range check might fail, in call inlined at machine.adb:114 (e.g. when PC = 65536) 24:40 medium: range check might fail, in call inlined at machine.adb:117 (e.g. when PC = 65536) 24:40 medium: range check might fail, in call inlined at machine.adb:120 (e.g. when PC = 65536) 24:40 medium: range check might fail, in call inlined at machine.adb:123 (e.g. when PC = 65536) 24:40 medium: range check might fail, in call inlined at machine.adb:126 (e.g. when PC = 65536) 24:40 medium: range check might fail, in call inlined at machine.adb:129 (e.g. when PC = 65536) 24:40 medium: range check might fail, in call inlined at machine.adb:132 (e.g. when PC = 65536) 24:40 medium: range check might fail, in call inlined at	The proof is failing on procedure of IncPC. This procedure is called from some other functions that increment the PC value which could produce out of range value of PC (ProgramCounter). In this case, the ProgramCounter type has a boundary value from 1 to 65536. For example, when the current ProgramCounter holds the first boundary value, and then incremented by negative values. Another error could also occur when the incremented PC result is more than the last boundary value of Program Counter, which is 65536.	Exception error (range checked failed).

machine.adb:138 (e.g. when PC = 1) 24:40 medium: range check might fail, in call inlined at machine.adb:141 (e.g. when PC = 1) 24:40 medium: range check might fail, in call inlined at machine.adb:143 (e.g. when PC = 65536) 24:40 medium: range check might fail, in call inlined at machine.adb:146 (e.g. when PC = 65536)		
33:29 medium: overflow check might fail, in call inlined at machine.adb:113 (e.g. when Regs = (others => -1073741825))	The proof is failing on procedure DoAdd. An error could be triggered when the addition computation of two integer values produces an integer overflow. For example, when both Regs(Rs1) and Regs(Rs2) hold -1073741825, then the addition computation value should give -2147483650 as a result.	When this fault occurs, the computation will result in an unexpected value.
34:11 warning: unused assignment, in call inlined at machine.adb:113	The proof is failing on procedure DoAdd. The warning emerges since the value of Ret is assigned but it does not get used.	It does not give any effect.
42:29 medium: overflow check might fail, in call inlined at machine.adb:116 (e.g. when Regs = (1 => 1, others => -2147483648))	The proof is failing on procedure DoSub. An error could be triggered when the subtraction computation of two integer values produces an integer overflow. For example, when Regs(Rs1) holds 1 and Regs(Rs2) hold -2147483648, then the subtraction computation should produce 2147483649 as a result.	When this fault occurs, the computation will result in an unexpected value.

43:11 warning: unused assignment, in call inlined at machine.adb:116	The proof is failing on procedure DoSub. The warning emerges since the value of Ret is assigned but it does not get used.	It does not give any effect.
51:29 medium: overflow check might fail, in call inlined at machine.adb:119 (e.g. when Regs = (1 => 2, others => -2))	The proof is failing on procedure DoMul. An error could be triggered when the multiplication computation of two integer values produces an integer overflow. For example, when a Regs(Rs1) holds 2 and Regs(Rs2) holds 2147483647, then the multiplication computation should produce 4294967294 as a result.	When this fault occurs, the computation will result in an unexpected value.
52:11 warning: unused assignment, in call inlined at machine.adb:119	The proof is failing on procedure DoMul. The warning emerges since the value of Ret is assigned but it does not get used.	It does not give any effect.
60:29 medium: overflow check might fail, in call inlined at machine.adb:122 (e.g. when Regs = (0 => -1, others => 0))	The proof is failing on procedure DoDiv. An error could be triggered when the division computation of two integer values produces an integer overflow. For example, when Regs(Rs2) as the denominator holds -1 and Regs(Rs1) hold -2147483648, then the division value should produce 2147483648 as a result.	When this fault occurs, the computation will result in an unexpected value.
60:29 medium: divide by zero might fail, in call inlined at machine.adb:122 (e.g. when Regs = (others => 0))	The proof is failing on procedure DoDiv. An error could be triggered when the denominator holds 0 value.	When this fault occurs, an exception error will be thrown (divide by zero exception).
61:11 warning: unused assignment, in call inlined at machine.adb:122	The proof is failing on procedure DoDiv. The warning emerges since the value of Ret is assigned but it does not get used.	It does not give any effect.
68:33 medium: range check might fail, in call inlined at machine.adb:125 (e.g. when A = 0)	The proof is failing on procedure DoLdr. An error could be triggered when the addition computation of Regs(Rs) and offset produces the value which is out of bounds of the memory index. In this case, the computed address index must hold the value within 0-65535 range. I	An exception error will be thrown (range checked failed)..

68:33 medium: overflow check might fail, in call inlined at machine.adb:125 (e.g. when A = 0 and Regs = (others => -2147483648))	The proof is failing on procedure DoLdr. An error could be triggered when the addition computation of Regs(Rs) and offset produces an integer overflow.	An unexpected value will be return.
71:11 warning: unused assignment, in call inlined at machine.adb:125	The proof is failing on procedure DoLdr. The warning emerges since the value of Ret is assigned but it does not get used.	It does not give any effect.
78:33 medium: range check might fail, in call inlined at machine.adb:128 (e.g. when A = 0)	The proof is failing on procedure DoStr. An error could be triggered when the addition computation of Regs(Ra) and offset produces the value which is out of bounds of the memory index. In this case, the computed address index must hold the value within 0-65535 range.	An exception error will be thrown (range checked failed).
78:33 medium: overflow check might fail, in call inlined at machine.adb:128 (e.g. when A = 0 and Regs = (others => -2147483648))	The proof is failing on procedure DoStr. An error could be triggered when the addition computation of Regs(Ra) and offset produces an integer overflow.	An unexpected value will be return.
81:11 warning: unused assignment, in call inlined at machine.adb:128	The proof is failing on procedure DoStr. The warning emerges since the value of Ret is assigned but it does not get used.	It does not give any effect.
89:11 warning: unused assignment, in call inlined at machine.adb:131	The proof is failing on procedure DoMov. The warning emerges since the value of Ret is assigned but it does not get used.	It does not give any effect.

One of the subtle semantic bugs is the lowercase misspelling of IncPc (instead of IncPC)