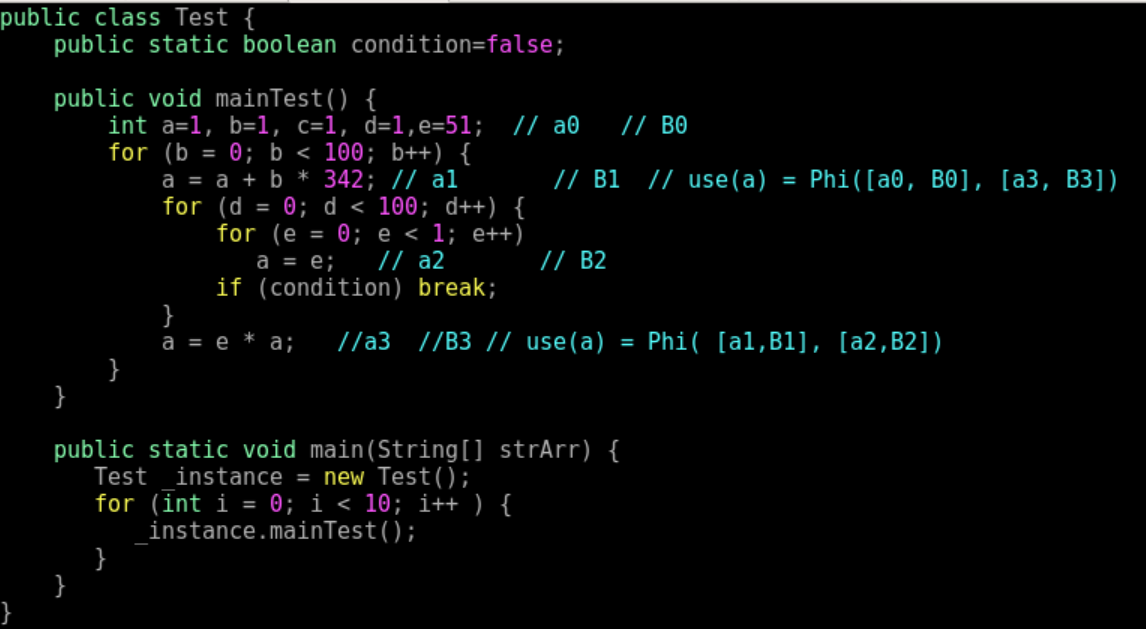
[JDK-8230185](https://bugs.openjdk.java.net/browse/JDK-8230185)

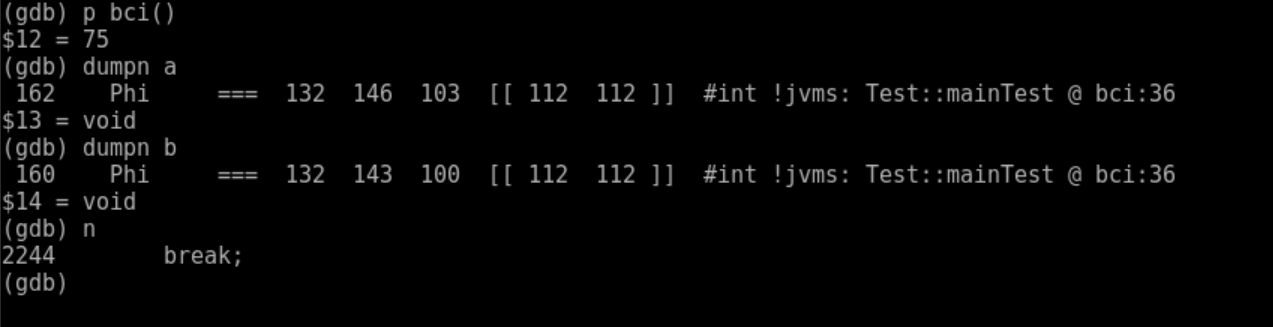
Following is a reduced version of test case attached with the bug-report.

Compilation options: -Xcomp -XX:+TieredCompilation (default) , first level compilation by C1 at level 2 and next inner loop OSR compilation by C2.

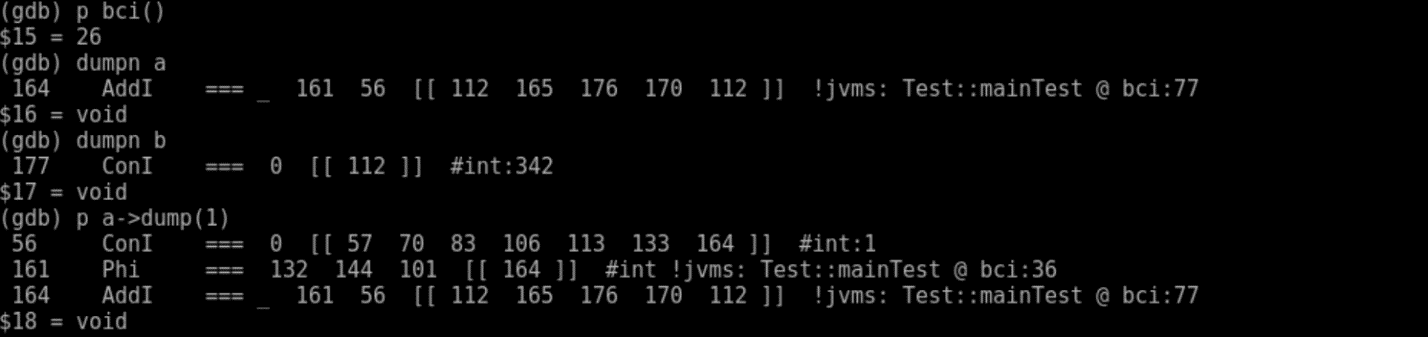


Ideal Creation (OSR compilation for hot-loop involving e induction variable.

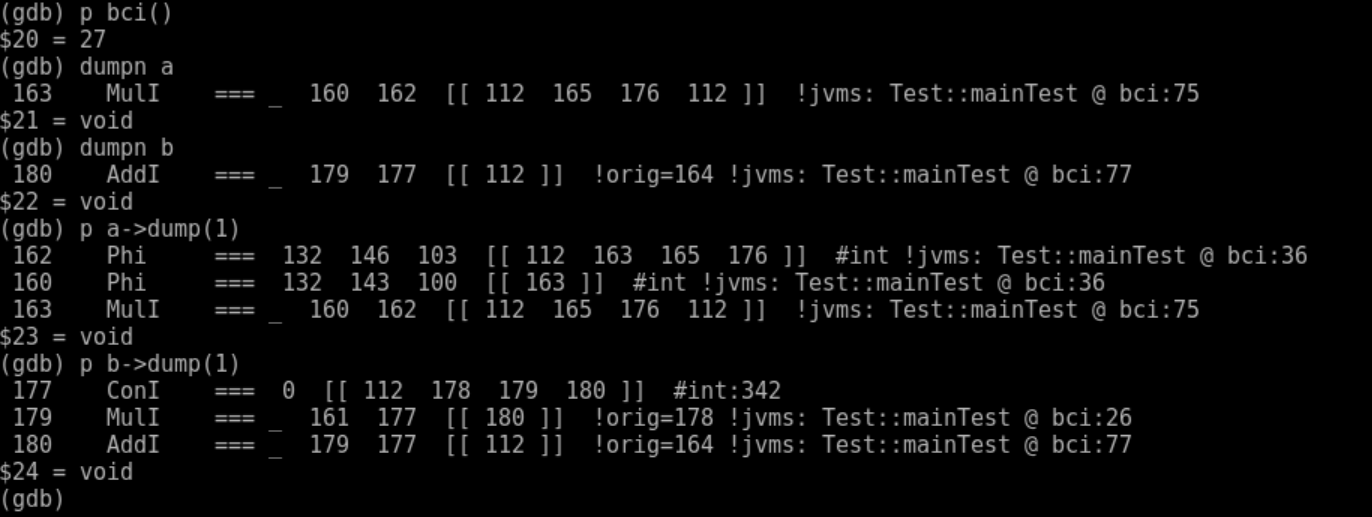
1. Multiply (e\*a in B3) : Both the inputs are Phi since both uses ‘e’ and ‘a’ have access to values coming from different control flows.



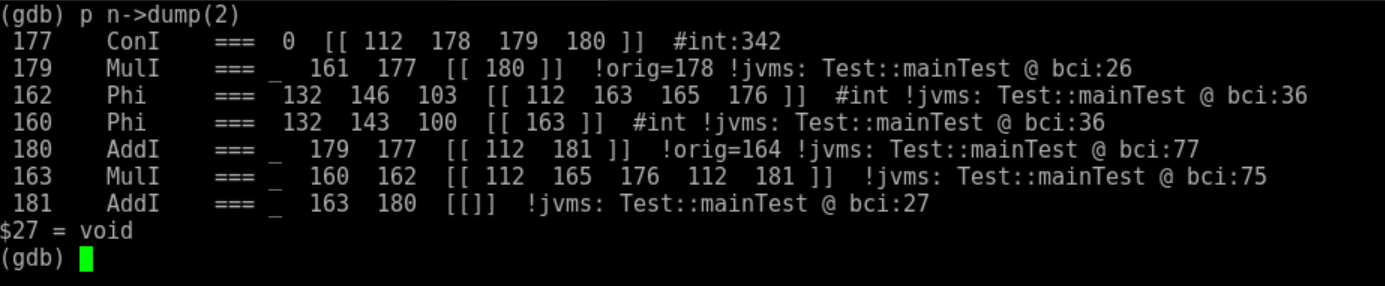
1. Multiply (b\*342 in B3) : One input is constant and other should have been Phi node since use of ‘b’ can get values ‘0’ from loop header and incremented values from loop body (b++).



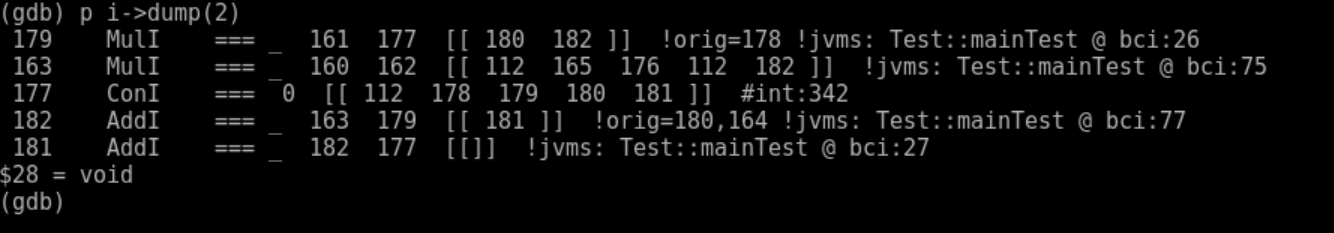
1. Add operation ( a **+** b \* 342) : Both its inputs should be Phi-values but its not the case.



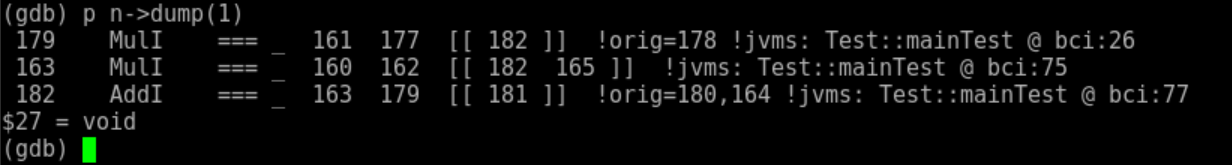
1. Newly created AddINode:



1. Newly created AddINode after AddINode::Ideal transformation:



1. Graph node seen by PhaseIdealLoop::create\_MulAdd() : Expected pattern was created and matched which is incorrect.



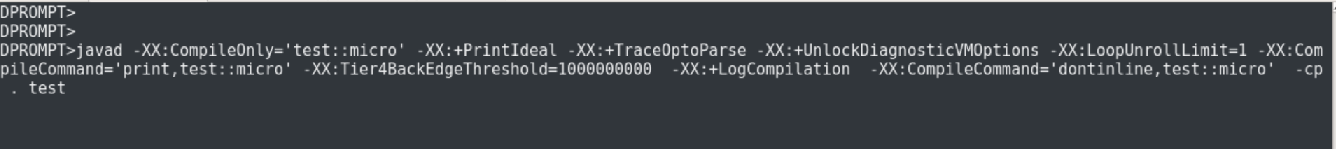
Problem occurs because basic blocks were processed in a non-RPO order. Due to OSR compilation which was kicked for inner loop, ideal graph construction begins from inner loop header.

Traversal continued by following the successor blocks (DFS). In this case we have an outer loop having a use for ‘a’ in block B1 which sees the only one definition of ‘a’ coming from block B3 which was processed before outer loop body. This leads to incorrect graph shape which does not comply with the program semantics.

Correct graph construction takes place if entire method ‘mainTest’ is compiled since BasicBlocks are then processed in a RPO order, this is verified using -Xcomp -XX:-TieredCompilation which will cause method compilation by C2.

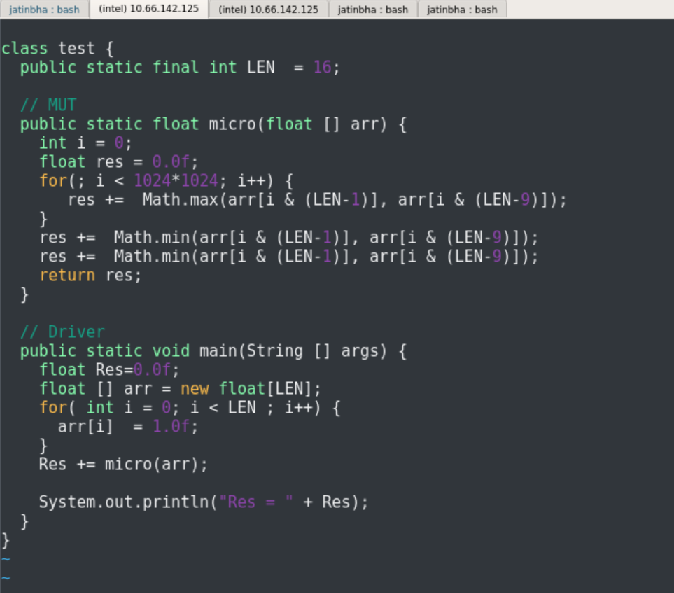
On Stack Replacement Compilation

CMD:



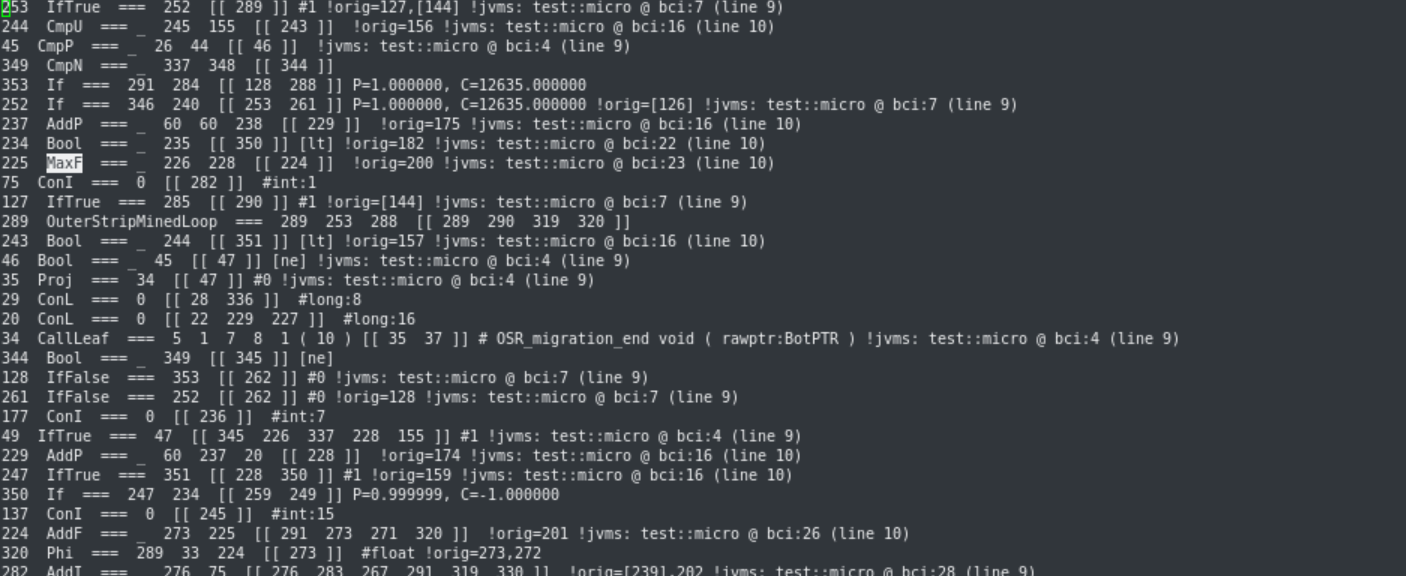
TraceOptoParser walks over bytecode for OSR start block and block involved in loop.

TEST:



Only loop blocks gets compiled,

Ideal graph does not create MinF node



Ends into deoptimization since post loop bytecode not compiled.

Compilation can be controlled using Tier3BackedgeThreshold and Tier4BackedgeThreshold for tiered compilation scenario.

Argument Packing from interpreter: OSR\_Migration\_begin (Runtime)

Argument Unpacking in Compiled code + (OSR\_Migration End) free argument buffer.