LLVM Example: Instruction\_Move

작성일 2019-04-40

작성자 강석원

# Instruction의 위치 이동

Instruction의 실행 순서는 일반적으로 LLVM IR 내에서 Instruction의 위치와 매핑됩니다.[[1]](#footnote-1)

따라서 Instruction의 실행 순서를 바꾸기 위해서는, LLVM IR 내에서 Instruction의 위치를 조정하면 됩니다.

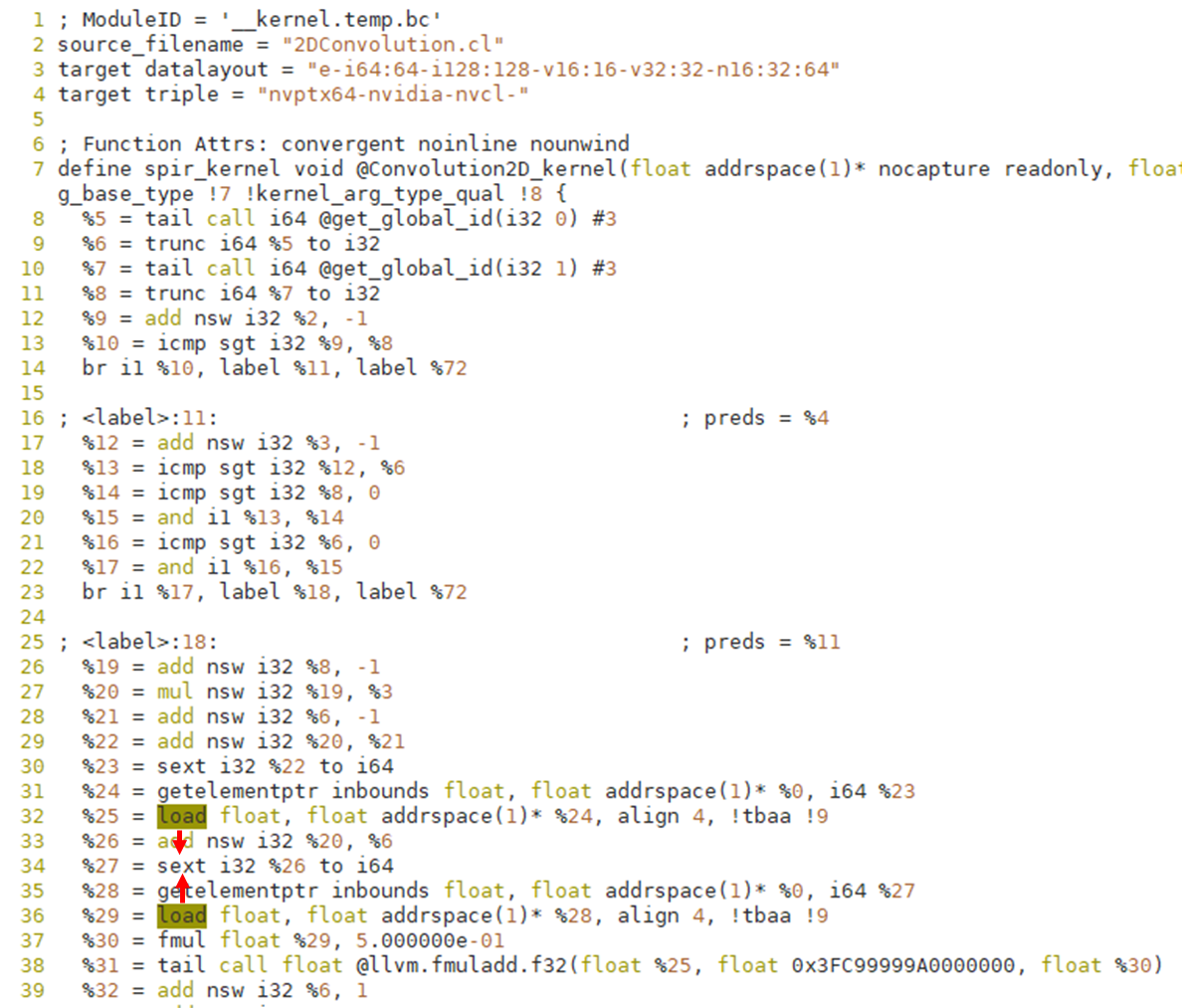
이러한 동작을 도와주는 API는 llvm::Instruction 클래스의 moveBefore(), moveAfter() 가 있습니다.

|  |
| --- |
|  |
| /usr/local/llvm/include/llvm/IR/Instruction.h |

각 API를 이용해 현재 가지고 있는 llvm::Instruction 객체가 대표하는 Instruction의 순서를 특정한 Instruction 전/후로 이동시킬 수 있습니다.

# Instruction\_Move

LLVM 예제 Instruction\_Move는 이러한 일련의 작업을 하는 OpenCL Kernel to NVPTX 컴파일러로써, 0번째 Load Instruction을 moveAfter()를 사용해 바로 앞으로(즉 실행 순서를 늦추는 방향으로), 1번째 Load Instruction을 moveBefore()를 사용해 바로 뒤로(즉 실행 순서를 앞당기는 방향으로) 변경합니다.[[2]](#footnote-2)



# Dependency와 Error

하지만 이렇게 Instruction을 이동할 때에는 Instruction간의 Dependency를 고려해야 합니다.

0번째 Instruction을 앞 쪽으로 옮기는 것은 (원래의 IR에서) %25의 명령어와 %26의 명령어 간의 Dependency가 없어서 잘 처리되었지만, 1번재 Instruction을 뒤쪽으로 옮기는 것은 (원래의 IR에서) %28의 결과 값을 %29에서 사용하기 때문에 오류가 발생합니다.

이 오류는 메모리 상에서 IR 단계의 처리나, IR을 human-readable IR(.ll)로 출력할 때는 발생하지 않지만, 이를 bitcode(.bc)로 컴파일하려고 할 때 발생합니다.

|  |
| --- |
|  |
| 0번째 Instruction이 바로 앞으로 옮긴 결과. |
|  |
| 1번째 Instruction이 바로 뒤로 옮긴 결과. 나중에 수행되는 명령어(%29)의 결과를 그 이전에 수행되는 명령어(%28)이 사용하고 있다. |
|  |
| Instruction간의 Dependency가 무너졌을 때 Instruction does not dominate all users 오류나, Segmentation Fault가 발생한다. |

# User

이러한 Dependency는 llvm::User 클래스와 llvm::Use 클래스로부터 알 수 있습니다.

|  |
| --- |
|  |
| /usr/local/llvm/include/llvm/IR/Instruction.h |

User클래스는 Operand가 있는 Value들이 가지는 Class로, Instruction 클래스의 부모 클래스 입니다.

이러한 User 클래스에는 getNumOperands(), getOperand() 같은 API들이 있어, 이를 사용하여 현재 Value에 사용되는 Operand들을 알 수 있습니다.

만약 어떤 Instruction을 보다 뒤로(실행순서를 이전으로) 옮기고 싶은데, Instruction의 Operand 중 일부가 Instruction인 경우, 이러한 Operand Instruction도 같이 뒤로 옮겨져야 합니다. 동시에, Instruction과 Operand Instruction의 상대적인 순서는 지켜져야 합니다.

또한 최상위 클래스인 llvm::Value에서는 UseList가 존재하여 현재 Value가 사용되는 부분, 즉 Operand의 입장에서 Operand들 사용하는 Instruction들을 알 수 있습니다.

|  |
| --- |
|  |
| /usr/local/llvm/include/llvm/IR/Value.h |

주로 use\_begin(), use\_end()를 사용하여 접근을 주로 사용합니다.

참고자료

[1] “llvm/IR/Instruction.h”, “llvm/IR/User.h”, “llvm/IR/Value.h”  
[2] <https://llvm.org/doxygen/classllvm_1_1Instruction.html>[[3]](#footnote-3)  
[3] <http://llvm.org/doxygen/classllvm_1_1User.html>  
[4] <https://llvm.org/doxygen/classllvm_1_1Value.html>

1. 물론 1) 타겟 하드웨어 Instruction 레벨로 내려가고 최적화 됨에 따라 2) 추가적인 IR-level 최적화에 따라 3) 타겟 하드웨어의 Out-of-Order 실행 유닛에 따라 LLVM IR과 변경될 수 있지만, 1), 3)은 변경할 수 없거나 매우 어려운 부분이고, 2)의 경우, llc를 이용한 llvm static compiler를 사용하면 추가적인 IR-level 최적화 없이 LLVM IR을 타겟 하드웨어 Instruction으로 변경할 수 있습니다. [↑](#footnote-ref-1)
2. 여기서 의미하는 순서는 전체 llvm::Module에서의 순서를 의미함 [↑](#footnote-ref-2)
3. [2], [3], [4] 사용하는 LLVM Version이 다름에 유의 [↑](#footnote-ref-3)