Assignment 2

1 Student Information

Name	: Hady Januar Willi
Student ID	: A0174396R
School Email	: E0210429@u.nus.edu
Email	: hadywilli@outlook.com

2 Initialized Variables Analysis Design

This analysis is a may analysis where the variable listed may or may not be initialized in that block from the previous block. There is no kill in this analysis as all the kill function always results in empty set.

2.1 Entry and Exit

$$IV_{entry}(\ell) = \begin{cases} \varnothing & if \ell = init(S_*) \\ \cup\{IV_{exit}(\ell')|(\ell', \ell) \in flow(S_*)\} & otherwise \end{cases} \tag{1}$$

$$IV_{exit}(\ell) = (IV_{entry}(\ell)) \cup gen_{IV}(B^{\ell}) where B^{\ell} \in blocks(S_*) \tag{2}$$

2.2 Generate

$$gen_{IV} = ([z := a]^{\ell}) = \{z\} \tag{3}$$

$$gen_{IV} = ([skip]^{\ell}) = \varnothing \tag{4}$$

$$gen_{IV} = ([b]^{\ell}) = IV(b) \tag{5}$$

2.3 Kill

$$kill_{IV} = ([z := a]^{\ell}) = \varnothing \tag{6}$$

$$kill_{IV} = ([skip]^{\ell}) = \varnothing \tag{7}$$

$$kill_{IV} = ([b]^{\ell}) = \varnothing \tag{8}$$

2.4 Template

Pattern	Instance
L	$P(Var_*)$
$\sqsubseteq$	$\subseteq$
$\sqcup$	$\cup$
$\bot$	$\varnothing$
l	$\varnothing$
E	$init(S_*)$
F	$flow(S_*)$
$\tilde{F}$	$\{f : L \to L \mid \exists l_g : f(l) = l \cup l_g\}$
f_ℓ	$f_\ell(l) = l \cup gen(B^\ell)\ where\ B^\ell \in blocks(S_*)$

3 Analysis Implementation

The analysis designed above is implemented using LLVM version 7 in a clean Ubuntu 18.04 virtual machine. The full LLVM installation script is included together with the submission package.

The implementation for both task 2 and task 3 uses simple depth first search by recursive function. The notion of node is LLVM basic block and path is LLVM basic block parent - child relation in LLVM flow. For each store instruction in the basic block, the operand is added into the current node set. Graph traversal continues if there is a successor, and if either one of the two conditions below is met:

1. The current node is not traversed yet
2. The initialized variable set size increased for the current node

4 Submission Folder Structure

> zip – Folder containing all files
- > bin – Folder containing compiled assignment binary
- > graphs – Folder containing dot and pdf files generated from LLVM intermediate representation
- > installation – Folder containing LLVM 7 installation script
- > llvms – Folder containing LLVM intermediate representation
- > source – Folder containing assignment source code
- > target – Folder containing target C code

5 Running the Implementation

After extracting the file, the following commands can be run in sequence:

```
./compile
./llvm
./task2
./task3
```

After compilation, to run against any other LLVM intermediate representation file, the following command can be used:

```
./bin/assignment1 other-file.ll
```

To generate the graphs, the following command can be run:

```
./graph
```

If all else fails, the compiled binary is also included in the assignment, and can be used with the following command which hopefully runs with or without LLVM libraries:

```
./bin/assignment1/test.ll
```

6 Notes

1. %1, which is the return variable is always set in all blocks
2. LLVM 7 outputs variables as %N instead of variable name from source code
3. Task 2 command executes against example 1 in assignment handout
4. Task 3 command executes against example 3 in assignment handout