Assignment 1

C Shruti - EE18BTECH11006

Download all files from

https://github.com/shruti-chepuri/C-DS/tree/main/Assignment-1

1 Problem

(Q 12) Consider the following intermediate program in three address code

```
p = a - b
q = p * c
p = u * v
q = p + q
```

Which one of the following corresponds to a static single assignment from the above code?

2 Solution

Answer:
Static single assignment form is a property of an intermediate representation (IR), which requires that each variable be assigned exactly once, and every variable be defined before it is used. Existing variables in the original IR are split into versions, new variables typically indicated by the original name with a subscript, so that every definition gets its own version.

A compiler can be divided into two parts : the front end (analysis part) and the back end (synthesis part). The intermediate code comes in between these two parts. The front end after preforming lexical, syntax and semantic analysis produces the intermediate code which is consumed or used or fed into the back end for producing the target code or machine code.

The front end is language specific and the back end is machine specific. In other worlds the front end of a C compiler is same in two machines but the back end might be different because of the difference in the architecture and instruction set of the machines.

Static Single Assignment form (SSA form)
According to Static Single Assignment:

- A variable cannot be used more than once in the LHS.
- A variable should be initialized atmost once.

Now keeping in mind the above two rules we have, let us look at the options given:
A)

$p_1 = a - b$
$q_1 = p_1 * c$
p_1 = u*v // p_1 as already been used once in the LHS
$q_1 = p_2 + q_1$ // q_1 has already been assigned once before

This code violates condition 1 as p_1 has been assigned more than once.
B)

$p_3 = a - b$
$q_4 = p_3 * c$
p_4 = u*v // p_3 as already been used once in the LHS so we introduce p_4
$q_5 = p_3 + q_4$ // q_1 has already been used once before hence q_5

Option B satisfies all the rules of SSA.
C)

$p_1 = a - b$
$q_1 = p_2 * c$ // p_2 does not exist
$p_3 = u * v$
$q_2 = p_4 + q_3$ // p_4 and q_3 do not exist

Option C is invalid since p_2, p_4, q_3 are not initialized anywhere.
D)

$p_1 = a - b$
$q_1 = p * c$
$p_2 = u * v$
$q_2 = p + q$

This code is invalid as well since in the original intermediate program, q is sum of the modified variables p and q(they are modified in instructions 1-3), but here q_2 is written as a sum of the original p and q values and not the modified values(i.e. $p_2 + q_1$).
Thus the SSA form is

$p_3 = a - b$
$q_4 = p_3 * c$
p_4 = u*v
$q_5 = p_3 + q_4$

Advantages of using SSA form:
Using the SSA opens a bunch of opportunities for compiler optimisations like dead code elimination, strength reduction, register allocation, etc