Global data flow analysis

For a global optimization a program is represented in the form of program flow graph. The program flow graph is a graphical representation in which each node represents the basic block & edges represented the flow of control from one block to another.

→ The global data flow analysis is a process of ~~or collect~~ gathering the information about entire program & distributed this information to each block in the flow graph.

→ The gathered information helps to achieve a number of optimization.

→ Global data flow analysis is used to solve a specific problem ~~"Use-definition (ud-) chaining"~~.
" Use - definition (Ud-) chaining".

• Given that the identifier A is used at a point p. at what point could the value of A used at p have been defined.

Reaching Definition:- The Reaching definition implies the determination of definitions that apply at a point p in flow graph.

→ A definition D reaches at point P if there is a path from D to P along which D is not killed.
It follows the given steps

① Assign a distinct number to each definition as $d_1, d_2, d_3, \ldots, d_n$

② for each variable 'i', make a list of all definitions in entire program where it is used.

③ for each basic block B, compute the following

(a) GEN[B] :- The set GEN[B] consists of all the definitions generated in block B.

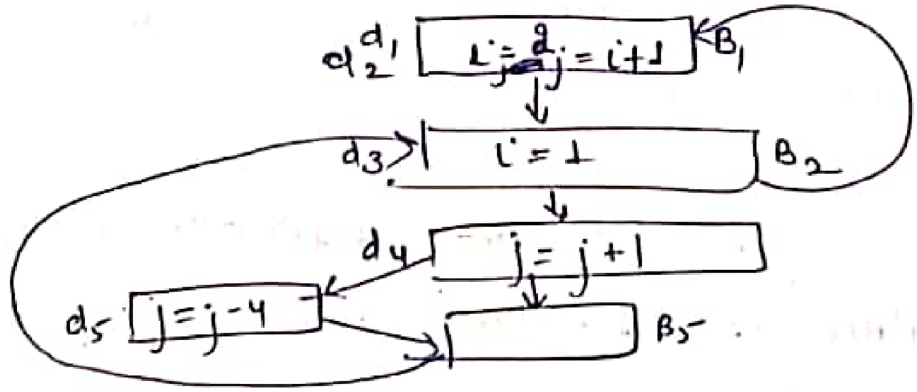
b) KILL[B]:- The set of all the definitions outside Block B that defines the same variables having definitions in Block B also.

i.e A definition D of variable x is killed when there a redefinition of x.

④ for all the basic block B, compute the following.

(a) IN[B] :- The set of all the definition reaching the point just before the statement of block B.

(b) OUT[B]: The set of all the definitions reaching the point just after the last statement of basic block B.



flow equations :- There are two set of equations called data-flow equations.

(1) $OUT[B] = IN[B] - KILL[B] \cup GEN[B]$

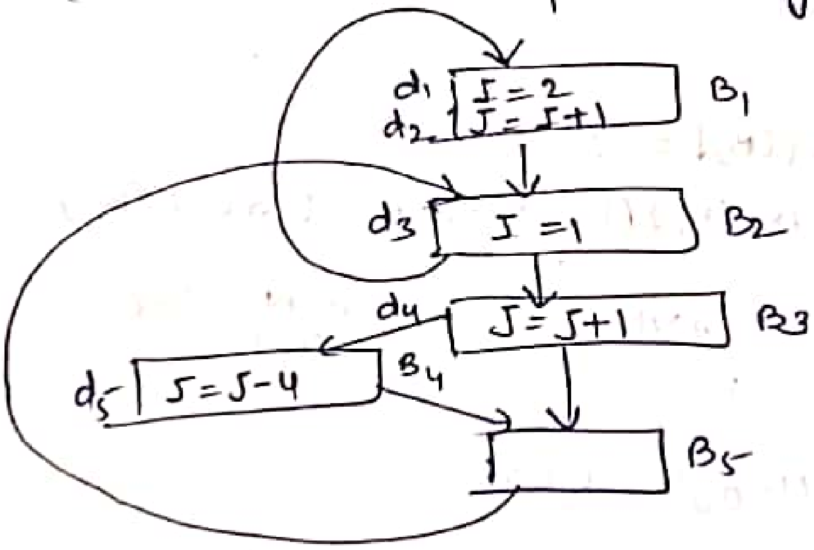
$= (IN[B] \text{ AND } [\neg KILL[B]]) \cup GEN[B]$

(2) $IN[B] = \bigcup_{P \text{ is predecessor of } B} OUT[B]$

Q Consider the flow graph.

~~I = 2~~
~~J = I+1~~

Find
(1) GEN and KILL for each block.
(2) IN and OUT for reaching definitions



generation is the pro...

Block B	Gen[B]	Bit vector d_1 d_2 d_3 d_4 d_5	KILL[B]	Bit vector d_1 d_2 d_3 ...
B_1	[d_1, d_2]	1 1 0 0 0	[d_3, d_4, d_5]	0 0 1 1
B_2	[d_3]	0 0 1 0 0	[d_1]	1 0 0 0
B_3	[d_4]	0 0 0 1 0	[d_2, d_5]	0 1 0 0 1
B_4	[d_5]	0 0 0 0 1	[d_2, d_4]	0 1 0 1 0
B_5	ϕ	0 0 0 0 0	ϕ	0 0 0 0 0

(2) Initially IN[B] = ϕ, OUT[B] = GEN[B]

Block	IN[B]	OUT[B]
B_1	0 0000	1 1000
B_2	0 0000	00100
B_3	0 0000	00010
B_4	00000	00001
B_5	00000	0 0000

for pass-1

IN[B_1] = OUT[B_2] = 00100

OUT[B_1] = $\underbrace{\text{IN}[B_1] \cap}_{\text{AND}} \underbrace{(-\text{KILL}[B_1])}_{\text{1's complement KILL}[B_1]} \underbrace{\cup}_{\text{OR}}$ GEN[B_1]

= (00100) ∩ (11000) ∪ 11000

= 00000 ∪ 11000

= 11000

Scanned with CamScanner

$= OUT[B_1] \cup OUT[B_5]$

$= 11000 \cup 00000$

$= 11000$

$OUT[B_2] = IN[B_2] \cap (-KILL(B_2)) \cup GEN[B_2]$

$= 11000 \cap (-10000) \cup 00100$

$= 11000 \cap (01111) \cup 00100$

$= 01100$

$IN[B_3] = OUT[B_2]$

$= 01100$

$OUT[B_3] = IN[B_3] \cap [-KILL[B_3]] \cup GEN[B_3]$

$= 00110$

$IN[B_4] = OUT[B_3]$

$= 00110$

$OUT[B_4] = IN[B_4] \cap (-KILL[B_4]) \cup GEN[B_4]$

$= 00110 \cap (-01010) \cup (00001)$

$= 00110 \cap (10101) \cup (00001)$

$= 00100 \cup 00001$

$= 00101$

$IN[B_5] = OUT[B_4] \cup OUT[B_3]$

$= 00101 \cup 00110$

$= 00111$

$OUT[B_5] = IN[B_5] \cap (-KILL[B_5]) \cup GEN[B_5]$

$= 00111$

Block	Pass-1 IN[B]	Pass-1 OUT[B]	Pass-2 IN[B]	Pass-2 OUT[B]	Pass3 IN[B]	Pass3 OUT[B]	Pass4
B_1	00100	11000	01100	11000	01111	11000	
B_2	11000	01100	11111	01111	11111	01111	Same as Pass3 so we stop
B_3	01100	00110	01111	00110	01111	00110	
B_4	00110	00101	00110	00101	00110	00101	
B_5	00111	00111	00111	00111	00111	00111	

Similarly calculate the Pass-2, Pass3 & Pass4. We receive the same result in Pass3 & Pass4, So we will stop process after Pass 4

Code Generation:- It is the final activity of compiler. Code generation is the process of creating Assembly/ Machine language. There are some properties of Code generation.

1. **Correctness :-** It should produce a correct code & do not alter the purpose of source code.
2. **High Quality :-** It should produce a high quality code.
3. **Efficient use of resource of target machine.** While generating
4. ~~Quick code generation~~ the code it is necessary to know the target machine on which it is going to get generated