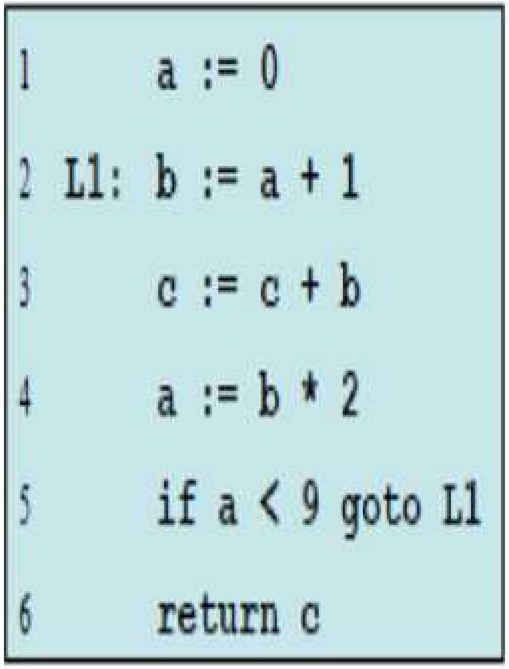
Live Variable Analysis

M.B.Chandak

hodcs@rknec.edu

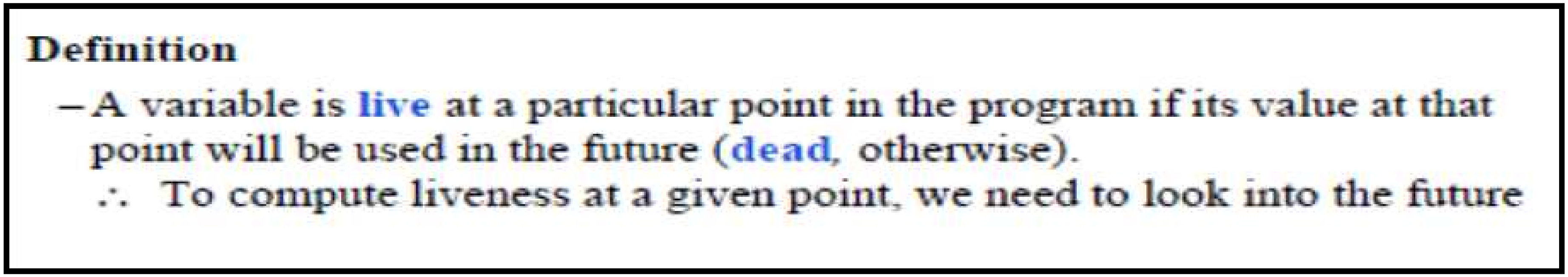
Live Variable

- Live variable analysis is one approach to perform Data Flow Analysis.
- Data Flow Analysis determines dynamic behavior of program by examining the static code.
- Example: of dynamic behavior: how many registers required for executing the program.
- For example in the code, since there are three variables, primarily it may require three registers for executions.
- Question: Can we reduce the number of register.
- Answer: Perform analysis on dynamic behavior of program
- Implementation: Find Live variables



```
    a := 0
L1: b := a + 1
    c := c + b
    a := b * 2
    if a < 9 goto L1
    return c
```

Live Variable

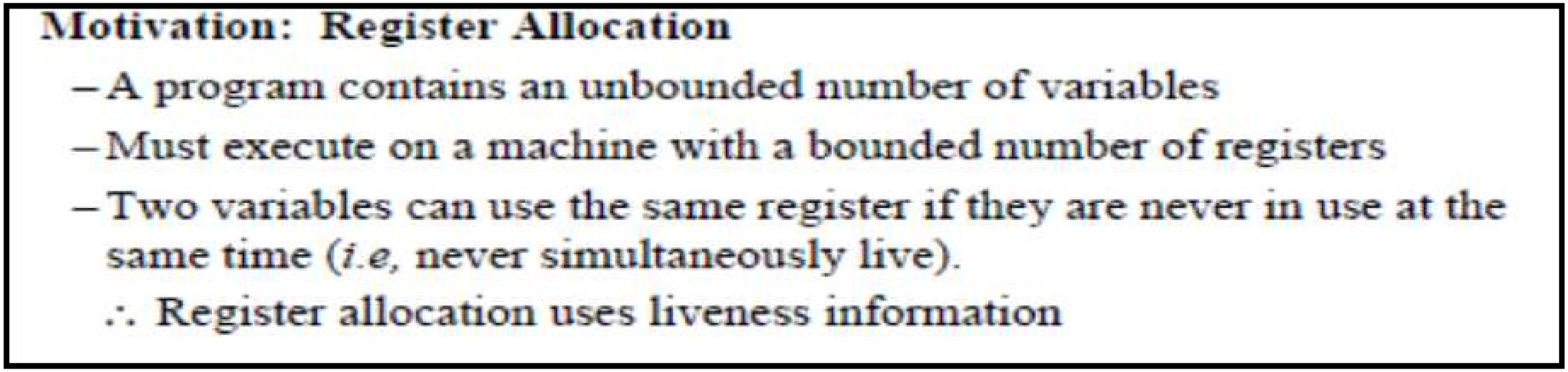


Definition

- A variable is live at a particular point in the program if its value at that point will be used in the future (dead, otherwise).

  ∴ To compute liveness at a given point, we need to look into the future

The variable "X" at point "P" is considered as Live if the value of "X at point P" could be used along some path in the flow graph, starting at "P".

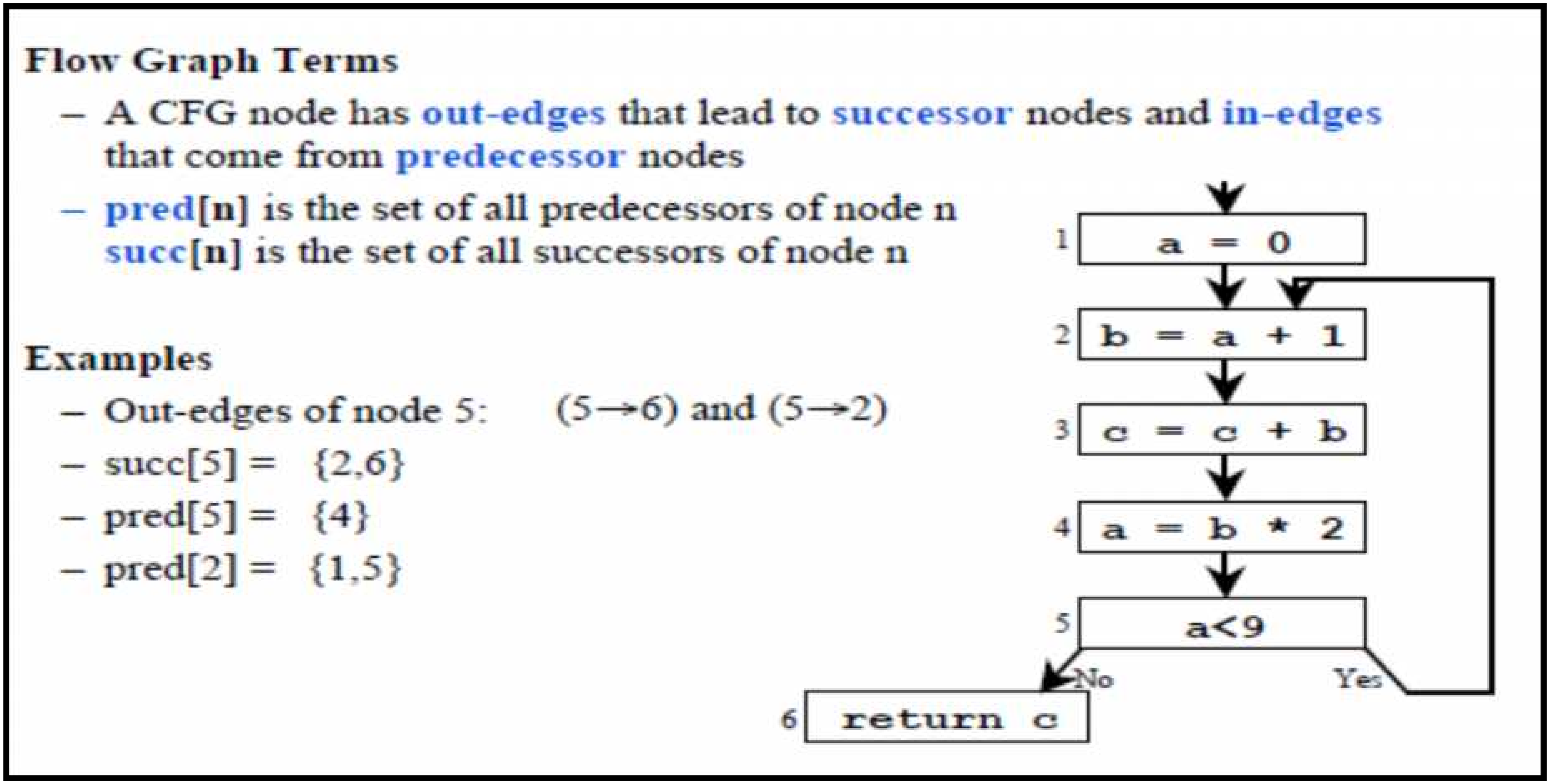


Motivation: Register Allocation

- A program contains an unbounded number of variables
- Must execute on a machine with a bounded number of registers
- Two variables can use the same register if they are never in use at the same time (*i.e*, never simultaneously live).

  ∴ Register allocation uses liveness information

Example-1: Live variable computation



Flow Graph Terms

- A CFG node has out-edges that lead to successor nodes and in-edges that come from predecessor nodes
- pred[n] is the set of all predecessors of node n
  succ[n] is the set of all successors of node n

Examples

- Out-edges of node 5: (5→6) and (5→2)
- succ[5] = {2,6}
- pred[5] = {4}
- pred[2] = {1,5}

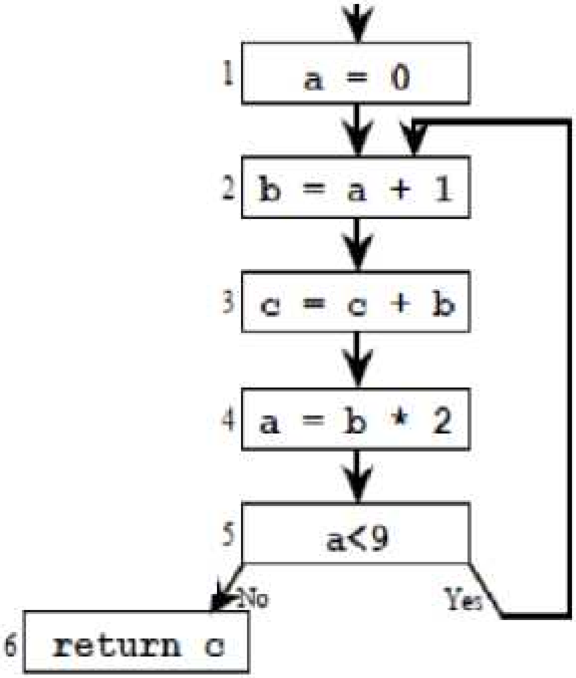
Live Variable:

- To compute Live Variable following sets are designed:
- 1. IN[B] = Set of variables live at beginning of block B
- 2. OUT[B] = Set of variables live just after block B
- 3. DEF[B] = Set of variables definitely assigned values in block “B” prior to any use of the variable in same block “B”**
- 4. USE[B] = Set of variables whose value may be used in “B” prior to any definition of the variable.*
- Live variable computation is BACWARD FLOW problem. The behavior of future nodes determines the live-ness at given node.

Cont.. Format def=use [EXAMPLE – 1]

- ***Formulation: In conditions and return statement variables are used.***
- OUT[B] = U IN[S] ➔ "S" = Successor
- IN[B] = USE[B] U [OUT{B} – DEF{B}]

BLOCK	SUCCESSOR	DEF	USE
B1	B2	a	-
B2	B3	b	a
B3	B4	c	bc
B4	B5	a	b
B5	B2, B6	-	a
B6	-	-	c



BLOCK	IN[B]	OUT[B]		IN[B]	OUT[B]		IN[B]	OUT[B]
B1	-	a		0Ua-a=0	ac		ac-a=a	ac
B2	a	bc		aUbc-b=ac	bc		aUbc-b=ac	bc
B3	bc	b		bcUb-c=bc	b		bc	bc
B4	b	a		bUa-a=b	ac		bc	ac
B5	a	ac		aUac-0=ac	ac		ac	ac
B6	c	-		cU0=c	0		c	0

BLOCK	SUCCESSOR	DEF	USE
B1	B2	a	-
B2	B3	b	a
B3	B4	c	bc
B4	B5	a	b
B5	B2, B6	-	a
B6	-	-	c

OUT[B] = U IN[S] ➔ "S" = Successor
IN[B] = USE[B] U [OUT{B} – DEF{B}]

U = Union operator

BLOCK		IN[B]	OUT[B]	IN[B]	OUT[B]
B1		ac-a=a	ac	a	ac
B2		aUbc-b=ac	bc	ac	bc
B3		bc	bc	bc	bc
B4		bc	ac	bc	ac
B5		ac	ac	ac	ac
B6		c	0	c	0

BLOCK	SUCCESSOR	DEF	USE
B1	B2	a	-
B2	B3	b	a
B3	B4	c	bc
B4	B5	a	b
B5	B2, B6	-	a
B6	-	-	c

OUT[B] = U IN[S] ➔ "S" = Successor
IN[B] = USE[B] U [OUT{B} – DEF{B}]

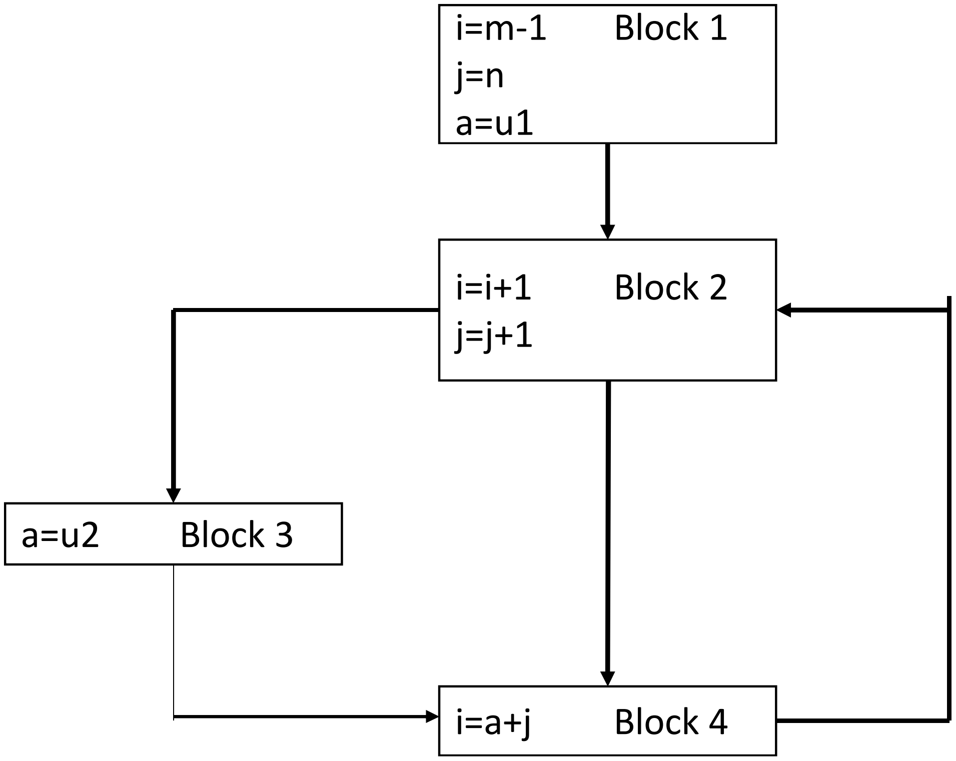
Live variable at start of Block B1 = a
Live variable at end of Block B1 = ac
Live variable at start of Block B2 = ac
Live variable at end of Block B2 = bc
Live variable at start of Block B3 = bc
Live variable at end of Block B3 = bc

Live variable at start of Block B4 = bc
Live variable at end of Block B4 = ac
Live variable at start of Block B5 = ac
Live variable at end of Block B5 = ac
Live variable at start of Block B6 = c
Live variable at end of Block B6 = 0

Variable "C" is used in all blocks, when variable "A" is used, variable "B" is not used, & vice-versa. Registers required = 2

Example-2

- Perform Live Variable analysis for following PFG



Block	Succ	USE	DEF	IN[B]	OUT[B]
B1	B2	m,n,u1	i,j,a	m,n,u1	i,j
B2	B3, B4	i,j	---**	i,j	a,j,u2
B3	B4	u2	a	u2	a,j
B4	B2	a,j	i	a,j	i,j

OUT[B] = U IN[S] ➔ "S" = Successor
IN[B] = USE[B] U [OUT{B} – DEF{B}]

Example-2[cont..]

Block	Successor	USE	DEF	IN[B]	OUT[B]	IN[B]	OUT[B]	IN[B]	OUT[B]
B1	B2	m,n,u1	i,j,a	m,n,u1	i,j	m,n,u1	i,j,a,u2	m,n,u1,u2	i,j,a,u2
B2	B3, B4	i,j	----	i,j	a,j,u2	i,j,a,u2	a,j,u2	i,j,a,u2	a,j,u2
B3	B4	u2	a	u2	a,j	j, u2	a,j	j,u2	a,j,u2
B4	B2	a,j	i	a,j	i,j	a,j	a,j,u2	a,j,u2	i,j,a,u2

OUT[B] = U IN[S] ➔ "S" = Successor
IN[B] = USE[B] U [OUT{B} – DEF{B}]

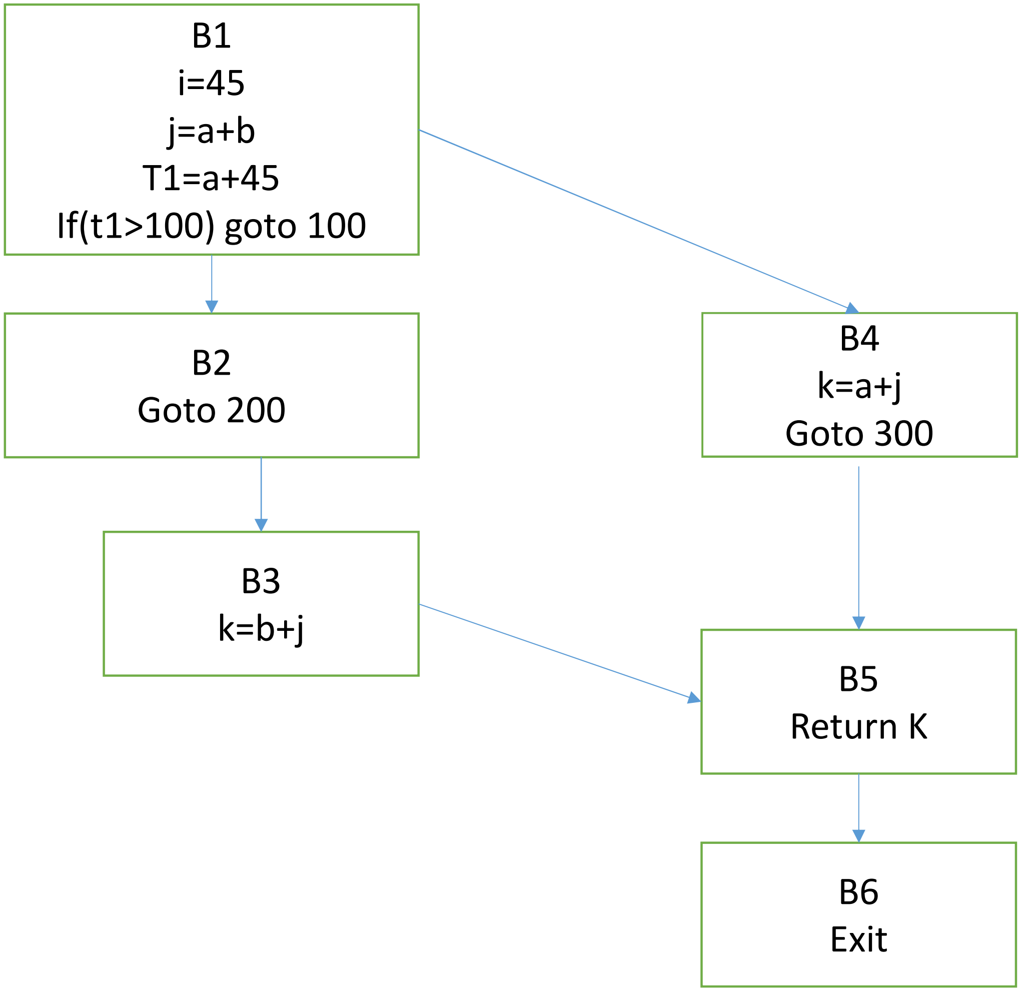
Example-2[cont..]

Block	Successor	USE	DEF	IN[B]	OUT[B]	IN[B]	OUT[B]
B1	B2	m,n,u1	i,j,a	m,n,u1,u2	i,j,a,u2	m,n,u1,u2	i,j,a,u2
B2	B3, B4	i,j	----	i,j,a,u2	a,j,u2	i,j,a,u2	a,j,u2
B3	B4	u2	a	j,u2	a,j,u2	j,u2	a,j,u2
B4	B2	a,j	i	a,j,u2	i,j,a,u2	a,j,u2	i,j,a,u2

OUT[B] = U IN[S] ➔ “S” = Successor
IN[B] = USE[B] U [OUT{B} – DEF{B}]

In the above table live variable computations are shown.
Variables live at start of block B1 are: m,n,u1,u2
Variables live at end of block B1 are: i,j,a,u2
Similar statements for remaining blocks

Assignment Question: Example 3



B1
i=45
j=a+b
T1=a+45
If(t1>100) goto 100

B2
Goto 200

B3
k=b+j

B4
k=a+j
Goto 300

B5
Return K

B6
Exit

Hint: In block B1
DEF={i,j,t1}
USE={a,b}
If t1 is not assigned value in
Previous instruction to IF then
USE will be {a,b,t1}