PES UNIVERSITY

CELEBRATING 50 YEARS

Compiler Design

Preet Kanwal

Department of Computer Science & Engineering

Teaching Assistant : Sree Pranavi G

Compiler Design

Unit 3: Control Flow Graph Generation

Preet Kanwal

Department of Computer Science & Engineering

Lecture Overview

In this lecture, you will learn about -

- CFG Generation
- Converting program to SSA
- Basic blocks
- Generate TAC

CFG Representation of Intermediate Code

- A flow graph is graphical representation that exhibits flow of control information.

- Helps in performing machine independent optimization.

- Benefits of code generation:
  - – Better register allocation.
  - – Better instruction selection.
  - – Helps reduce program cost.

CFG Representation of Intermediate Code

Rules for constructing flow graph:

- The nodes of the flow graph are the basic blocks.
- Number the nodes(For example:B1,B2).
- The first basic block (i.e B1)is called initial block.
- Draw a directed edge from the initial block i.e B1 to the next block following B1 i.e B2 if B2 immediately follows B1.

CFG Representation of Intermediate Code

- Partition intermediate code into basic blocks.

- Nodes of FG : Basic blocks

- Add nodes :
  - Entry node (edge to first block)
  - Exit node (edge from last block)

- Edges of FG : indicate which blocks can follow other blocks. (determine predecessor and successor of a Block).

Basic Blocks

- A basic block consists of set of statements which are executed sequentially without branching.
- Maximal sequence of consecutive instructions such that,
  - Flow of control can only enter the basic block from the first instruction
  - Control leaves the block only at the last instruction
- Each instruction is assigned to exactly one basic block.

Basic Blocks

Rules for determining basic blocks

- Identify leaders
  - The first three address instruction in the intermediate code is a leader.
  - Any instruction that is the target of a conditional or unconditional jump is a leader.
  - Any instruction that immediately follows a conditional or unconditional jump is a leader.

- The first instruction in the basic block is a leader and the basic block ends just before another leader instruction.

Basic Blocks

Example

```
1.  i = 1
2.  j = 1
3.  t1 = 10 * i
4.  t2 = t1 + j
5.  t3 = 8 * t2
6.  t4 = t3 - 88
7.  a[t4] = 0.0
8.  j = j + 1
9.  if j <= 10 goto (3)
10. i = i + 1
11. if i <= 10 goto (2)
12. i = 1
13. t5 = i - 1
14. t6 = 88 * t5
15. a[t6] = 1.0
16. i = i +1
17. if i <= 10 goto (13)
```

Example (contd.)

```
1.   i = 1
2.   j = 1
3.   t1 = 10 * i
4.   t2 = t1 + j
5.   t3 = 8 * t2
6.   t4 = t3 - 88
7.   a[t4] = 0.0
8.   j = j + 1
9.   if j <= 10 goto (3)
10.  i = i + 1
11.  if i <= 10 goto (2)
12.  i = 1
13.  t5 = i - 1
14.  t6 = 88 * t5
15.  a[t6] = 1.0
16.  i = i +1
17.  if i <= 10 goto (13)
```

First instruction in IC is a leader

Any Instruction that is the Target of a conditional or unconditional jump is a Leader

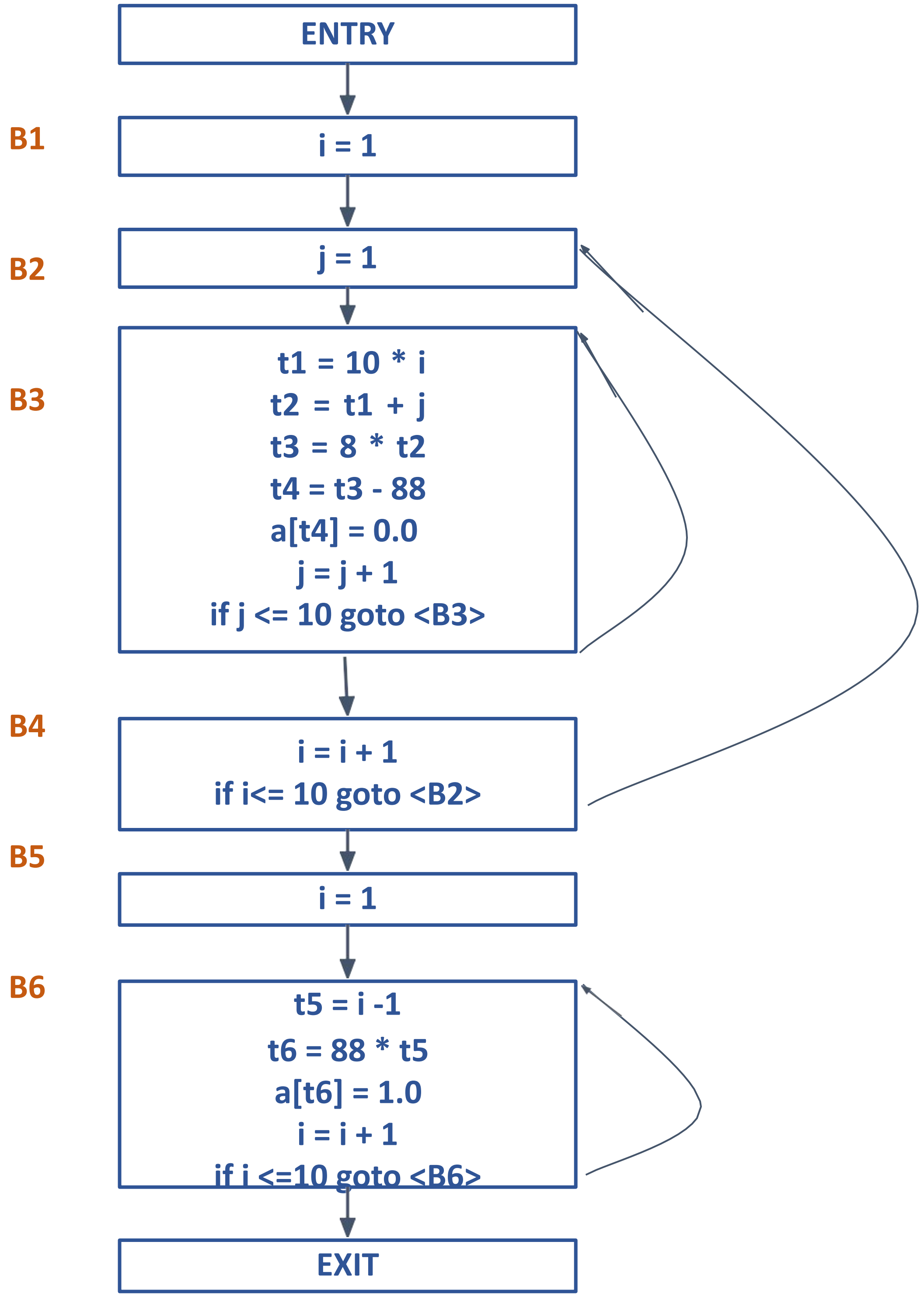
Any Instruction that follows a conditional or unconditional jump is a Leader

Example (contd.)

```
1.   i = 1
2.   j = 1
3.   t1 = 10 * i
4.   t2 = t1 + j
5.   t3 = 8 * t2
6.   t4 = t3 - 88
7.   a[t4] = 0.0
8.   j = j + 1
9.   if j <= 10 goto (3)
10.  i = i + 1
11.  if i <= 10 goto (2)
12.  i = 1
13.  t5 = i - 1
14.  t6 = 88 * t5
15.  a[t6] = 1.0
16.  i = i +1
17.  if i <= 10 goto (13)
```

For each leader, its basic block consists of itself and all instructions up to but not including the next leader

Example (contd.)



CFG

Example

```
int add(n, k){
s = 0;
a = 4;
i = 0;
if(k == 0)
    b = 1;
else
    b = 2;
while(i < n) {
    s = s + a * b;
    i = i + 1;
}
return s;
}
```

CFG

Example (contd.)

ICG

```
s = 0
a = 4
i = 0
ifFalse k == 0 goto L1
b = 1
goto L2
L1 : b = 2;
```

```
L2 : ifFalse i < n goto L3
t1 = a * b
t2 = s + t1
s = t2
t3 = i + 1
i = t3
goto L2
L3 : return s;
```

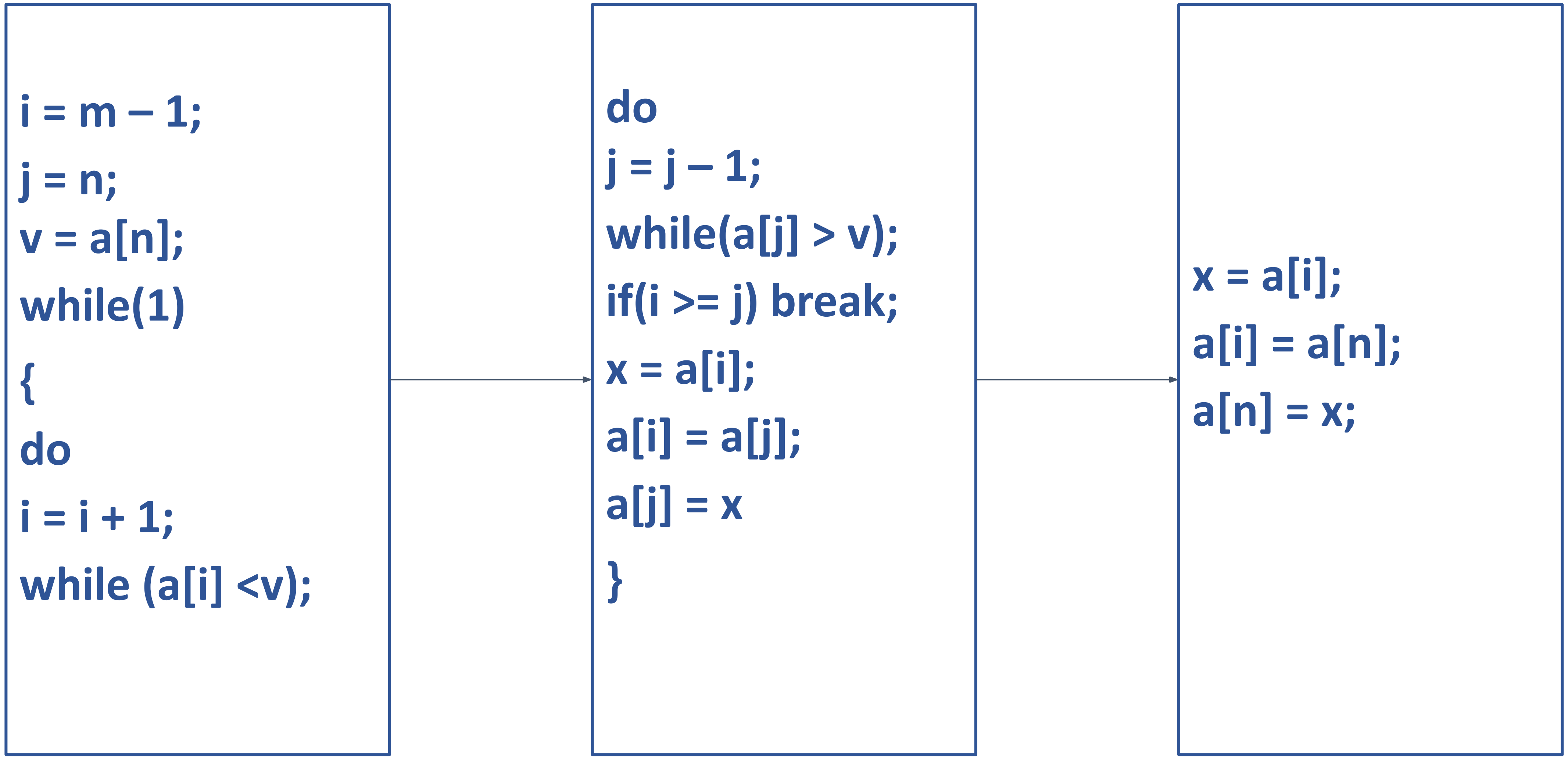
Example (contd.)

CFG

```
s = 0
a = 4
i = 0
ifFalse k == 0 goto L1
b = 1
goto L2
L1 : b = 2;
```

```
L2 : ifFalse i < n goto L3
t1 = a * b
t2 = s + t1
s = t2
t3 = i + 1
i = t3
goto L2
L3 : return s;
```

Example

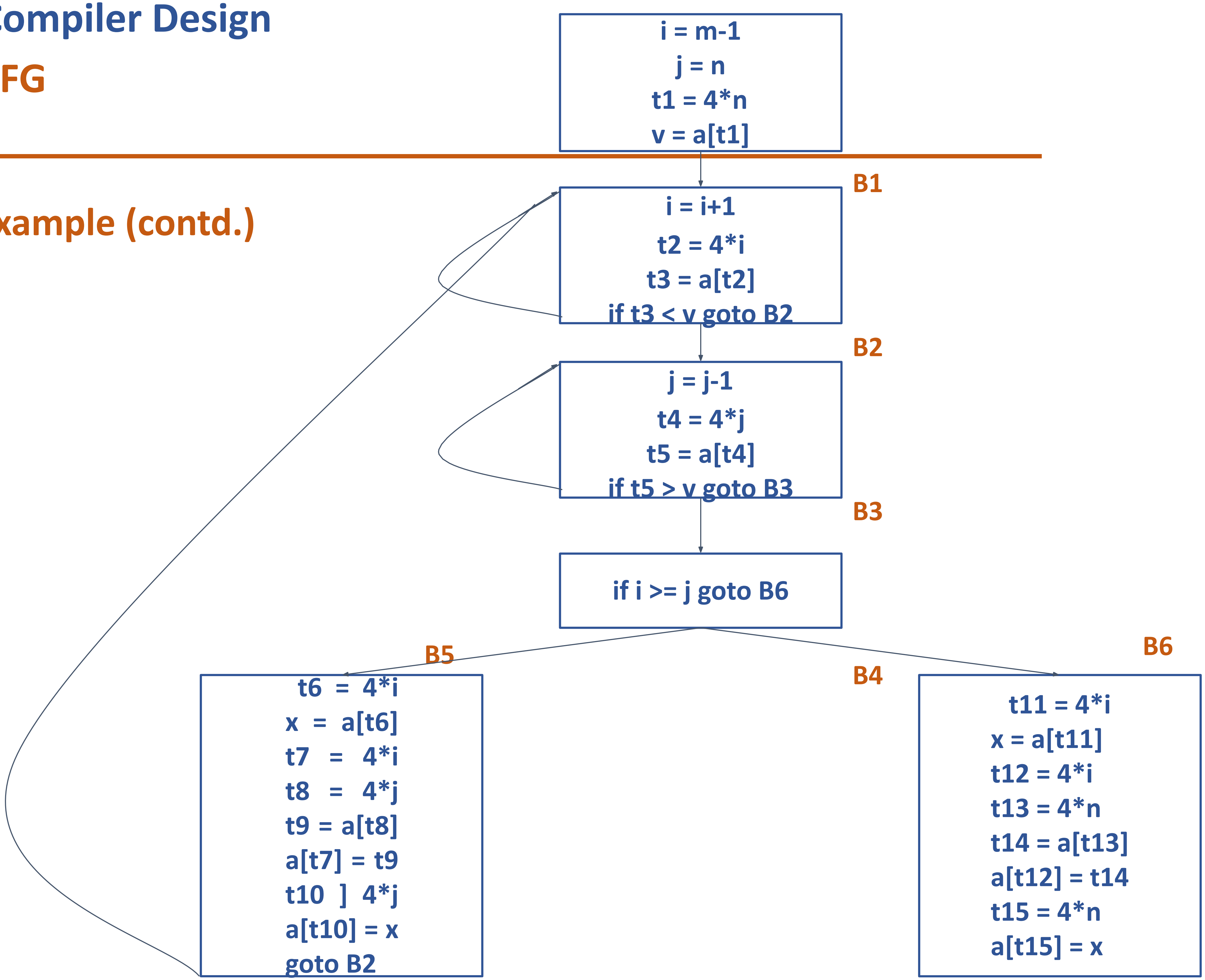


```
i = m – 1;
j = n;
v = a[n];
while(1)
{
do
i = i + 1;
while (a[i] <v);
```

```
do
j = j – 1;
while(a[j] > v);
if(i >= j) break;
x = a[i];
a[i] = a[j];
a[j] = x
}
```

```
x = a[i];
a[i] = a[n];
a[n] = x;
```

Example (contd.)



i = m-1
j = n
t1 = 4*n
v = a[t1]

B1

i = i+1
t2 = 4*i
t3 = a[t2]
if t3 < v goto B2

B2

j = j-1
t4 = 4*j
t5 = a[t4]
if t5 > v goto B3

B3

if i >= j goto B6

B4

B5

t6 = 4*i
x = a[t6]
t7 = 4*i
t8 = 4*j
t9 = a[t8]
a[t7] = t9
t10] 4*j
a[t10] = x
goto B2

B6

t11 = 4*i
x = a[t11]
t12 = 4*i
t13 = 4*n
t14 = a[t13]
a[t12] = t14
t15 = 4*n
a[t15] = x

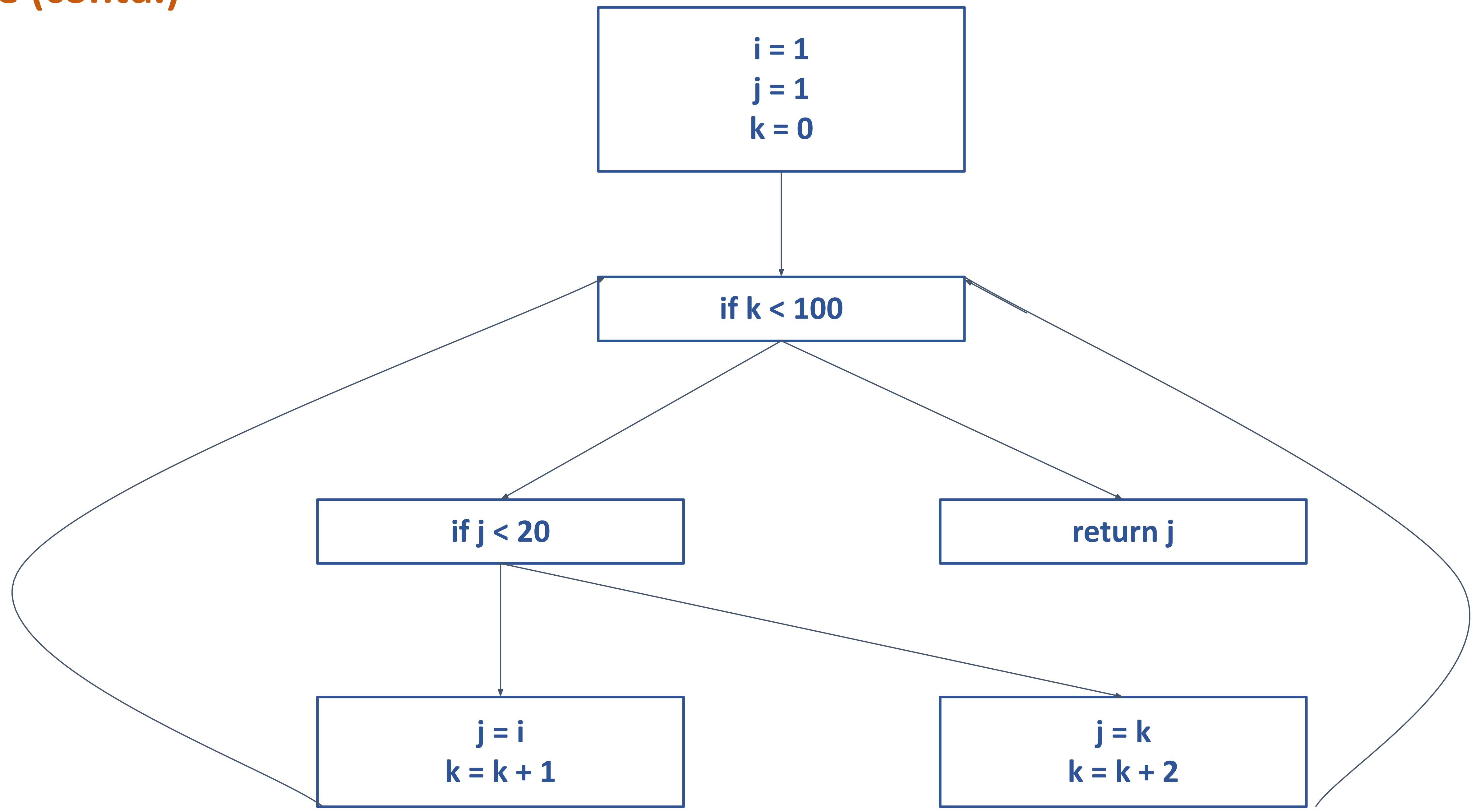
CFG

Example

```
    i = 1
    j = 1
    k = 0
    while k < 100
        if j < 20
            j = i
            k = k + 1
        else
            j = k
            k = k + 1
        end
    end
return j
```

CFG

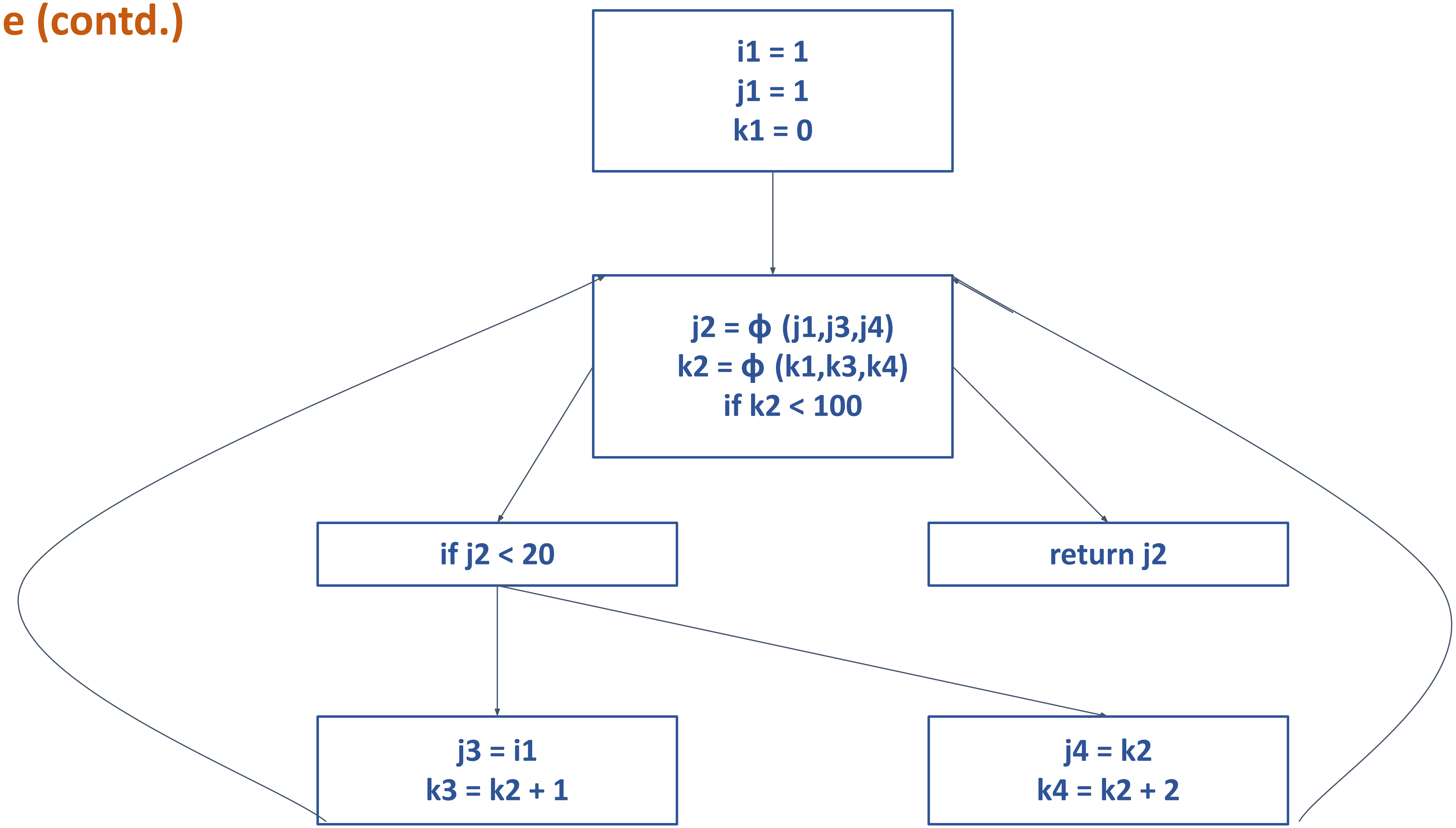
Example (contd.)



CFG

Example (contd.)

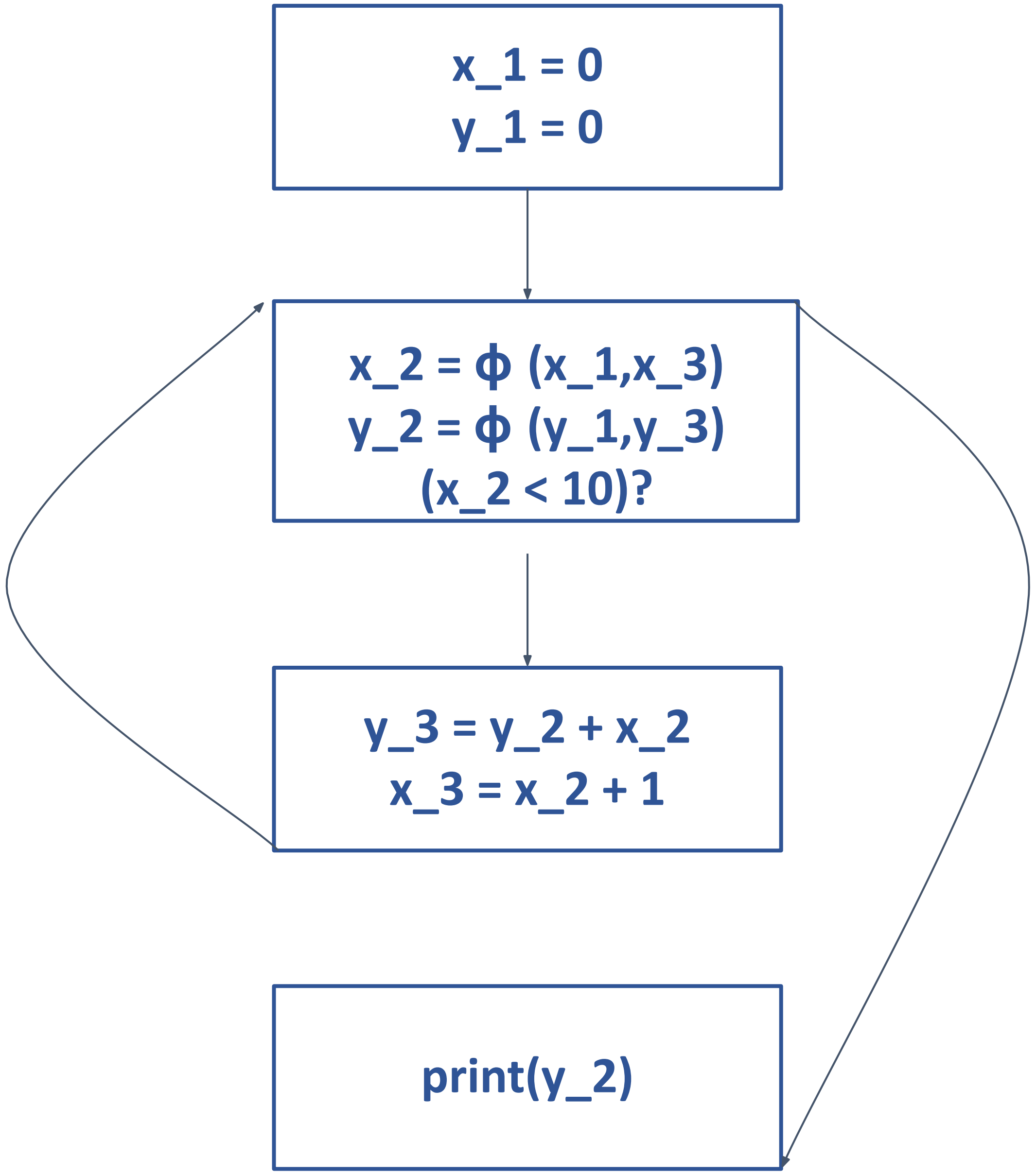
SSA



CFG

Example

```
x = 0;
y = 0;
while (x<10){
y = y+x;
x = x+1;
}
print(y);
```



CFG

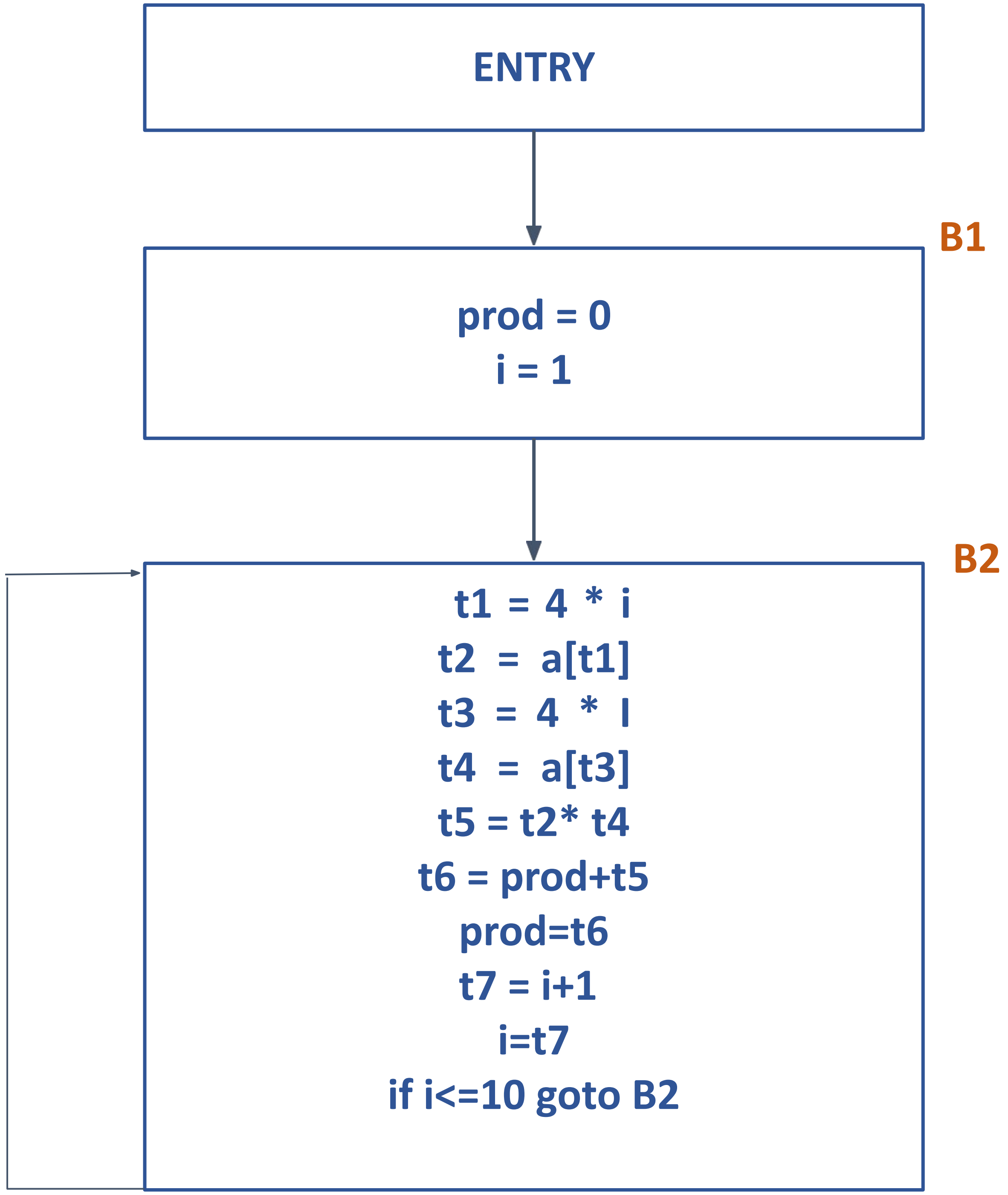
Example

```
prod = 0
i =1
do
{

prod = prod + a[i] * b[i];
i = i +1

}
while(i <=10)
```

CFG

Example (contd.)

```
prod = 0
i = 1
L: t1 = 4 * i
t2 = a[t1]
t3 = 4 * i
t4 = a[t3]
t5 = t2* t4
t6 = prod+t5
prod=t6
t7 = i+1
i=t7
if i<=10 goto L
```



ENTRY

B1

```
prod = 0
i = 1
```

B2

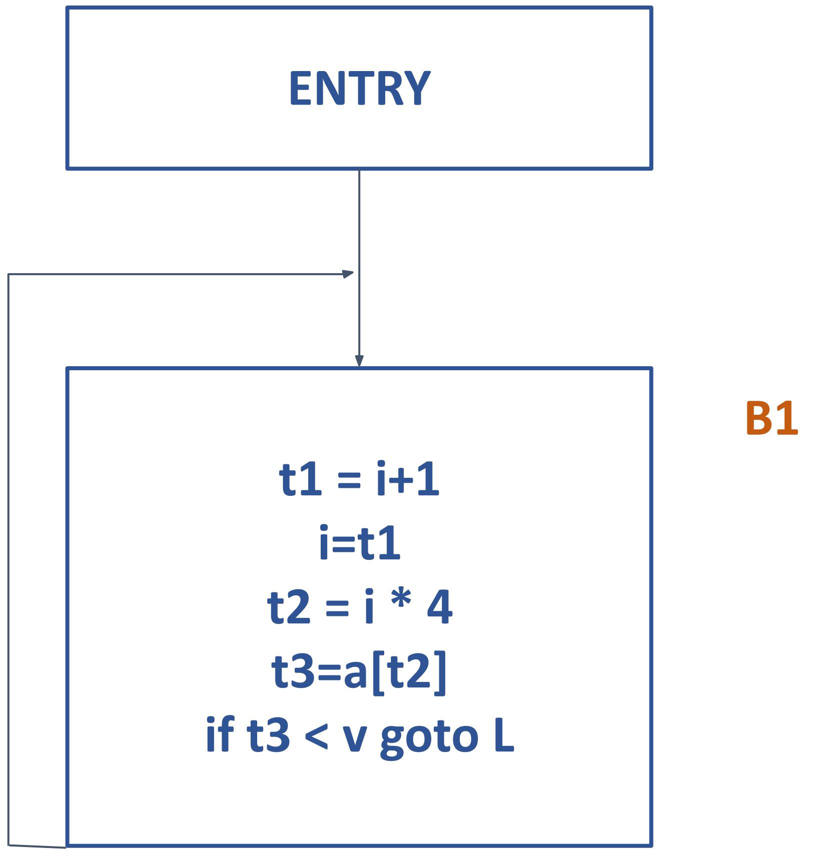
```
t1 = 4 * i
t2 = a[t1]
t3 = 4 * I
t4 = a[t3]
t5 = t2* t4
t6 = prod+t5
prod=t6
t7 = i+1
i=t7
if i<=10 goto B2
```

CFG

Example

```
do
{
      i = i +1;
}
while (a[i] <v)
```

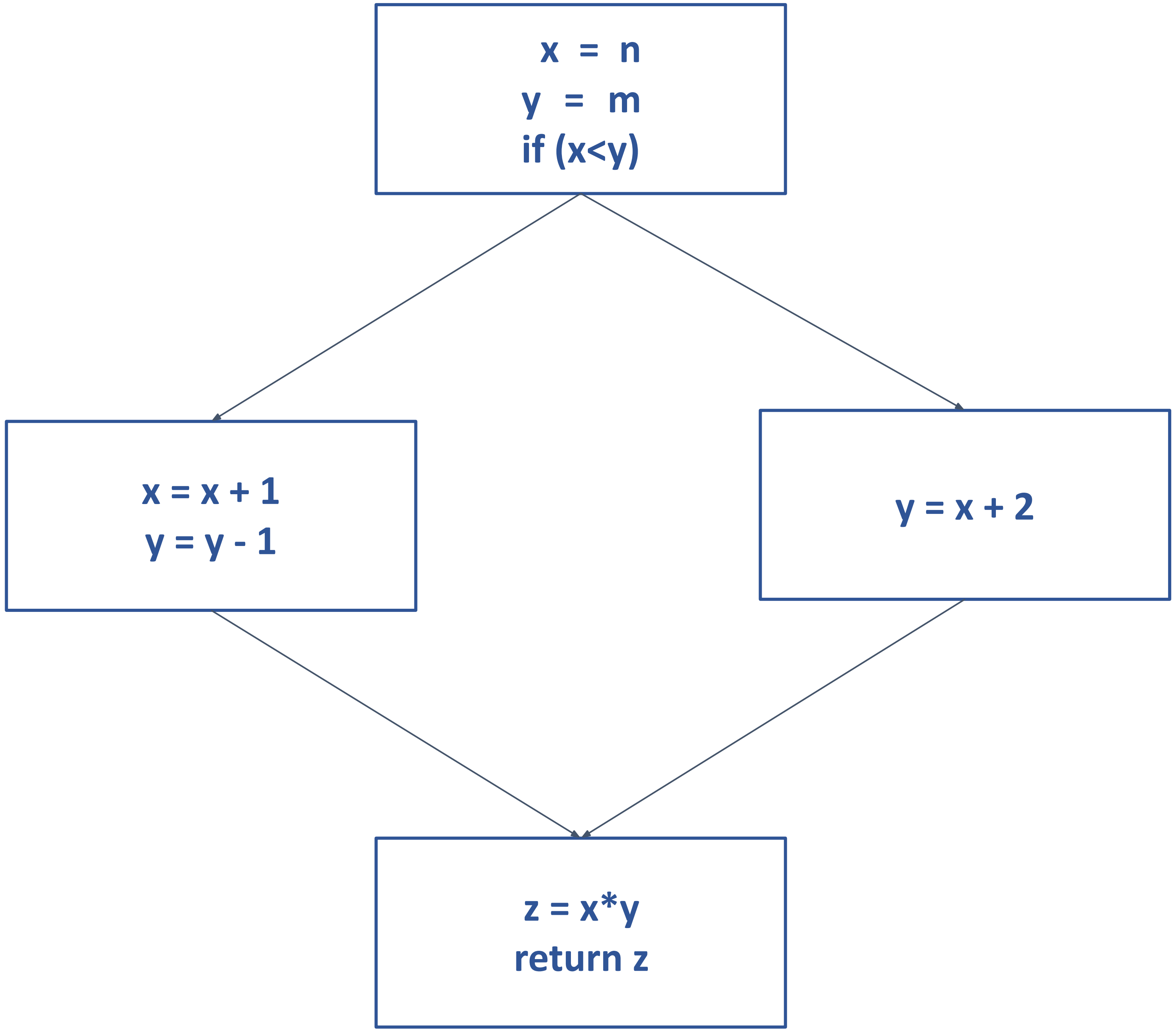
```
L : t1 = i+1
i=t1
t2 = i * 4
t3=a[t2]
if t3 < v goto L
```



CFG to SSA

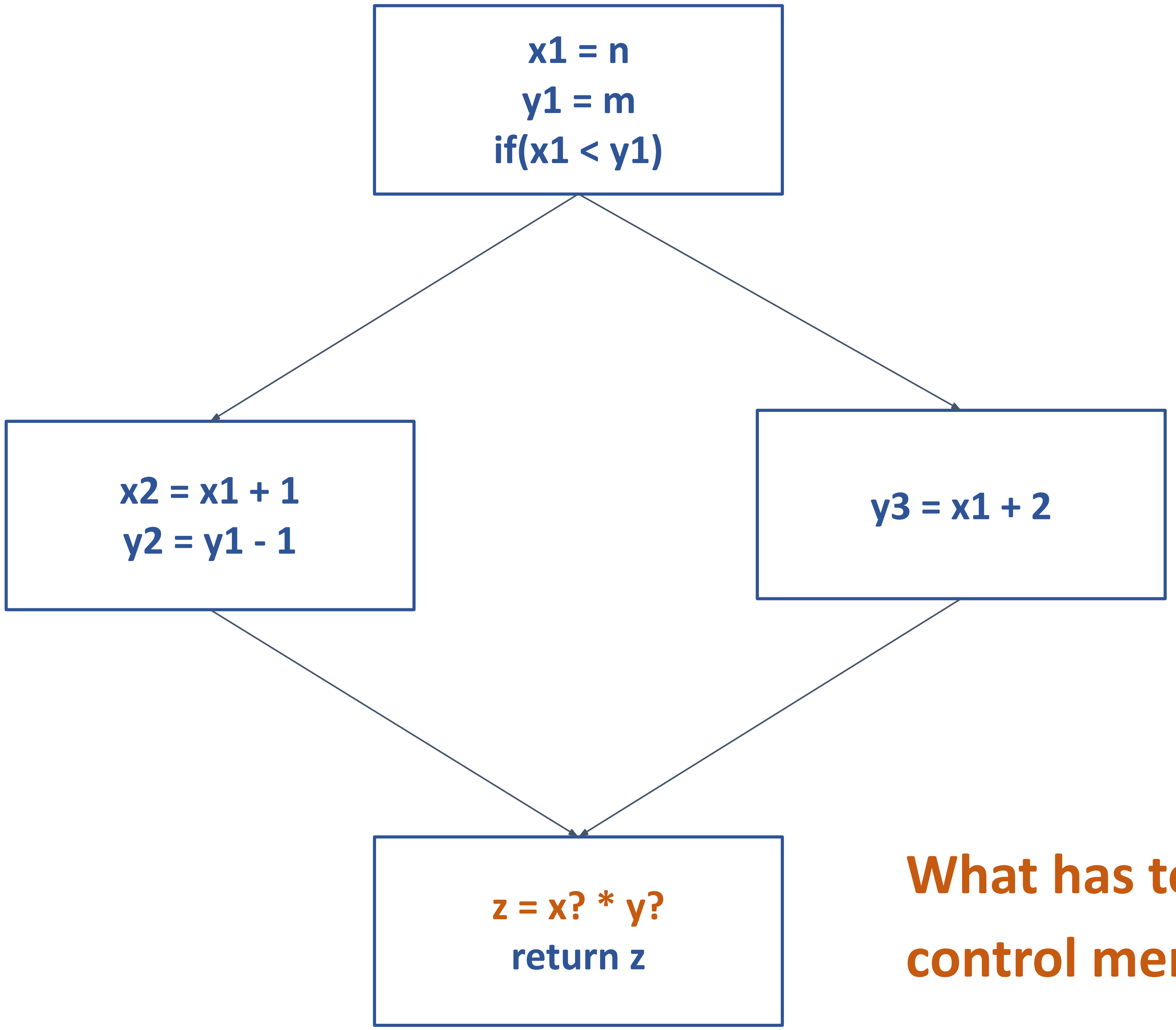
Example - *Program to SSA*

```
x = n
y = m
if(x < y)
{
     x = x + 1;
     y = y – 1;
}
else
{
     y = x + 2;
}
z = x * y;
return z;
```



CFG to SSA

Example(contd.)



x1 = n
y1 = m
if(x1 < y1)

x2 = x1 + 1
y2 = y1 - 1

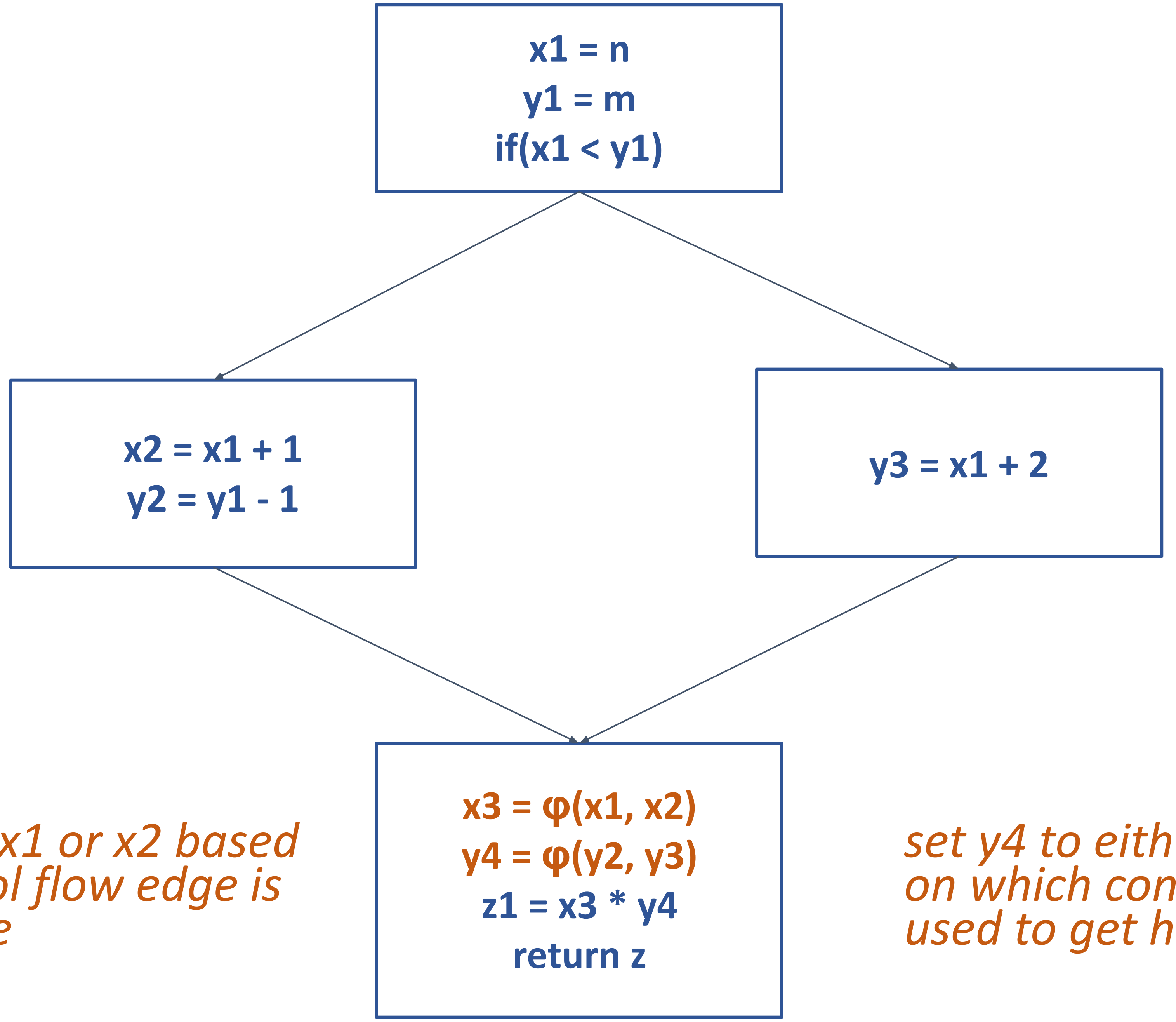
y3 = x1 + 2

z = x? * y?
return z

What has to be done when control merges?

CFG to SSA

Example(contd.)

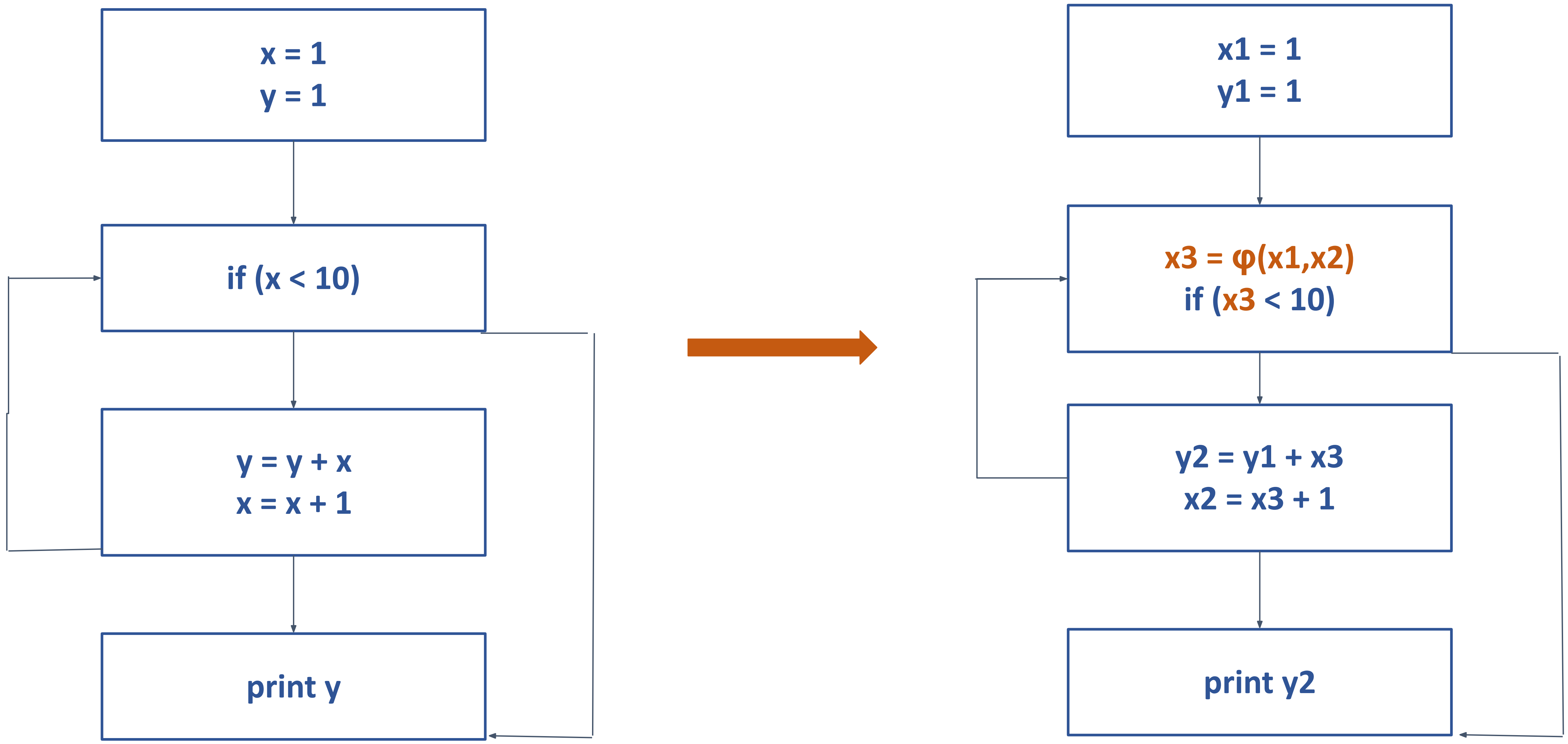


set x3 to either x1 or x2 based on which control flow edge is used to get here

set y4 to either y2 or y3 based on which control flow edge is used to get here

CFG

Example - *CFG to SSA*



Two Types of Questions

1) Convert a given program to Three address code and then construct the CFG.

2) Convert a given program to SSA
    - The question could specify the student to convert the program to CFG (change the working of every loop - while or for in terms of if loop) and then SSAify it.

THANK YOU

Preet Kanwal

Department of Computer Science & Engineering

preetkanwal@pes.edu