# CS & IT ENGINEERING

## Compiler Design

Intermediate code and code optimization

Lecture No. 2

By- DEVA Sir


- CFG (Control Flow Graph)
- 3AC

10 correction:

$$\left.\begin{aligned} c &= a+b \\ d &= a+b \\ e &= c*a \\ f &= d*b \\ g &= e+f \end{aligned}\right\} \Rightarrow g = \left[\underbrace{\cancel{(a+b)}}_{t_1} * a\right] + \left[\underbrace{\cancel{(a+b)}}_{t_1} * b\right]$$

Take (a+b) common

$$= (a+b) * [a+b]$$

$$\boxed{\begin{aligned} g &= a+b \\ g &= a*g \end{aligned}}$$

2 variable

```
for (i=0; i<n; i++)
{
    x=x+i;
}
```

$\Longrightarrow$

1. $i=0$
2. if($i \geq n$) goto 5
3. $x=x+i$
4. $i=i+1$; goto 2
5. exit

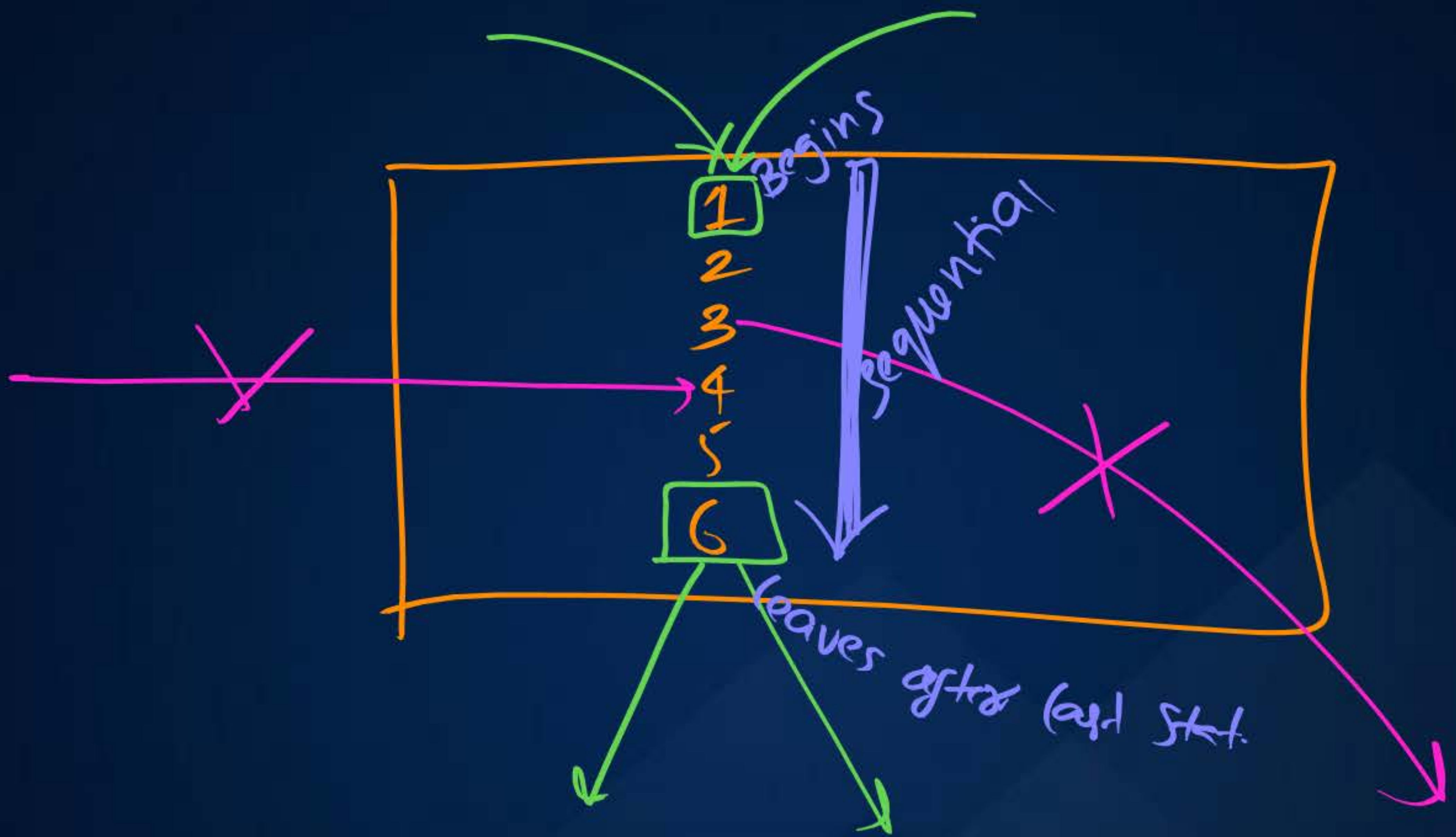
3AC

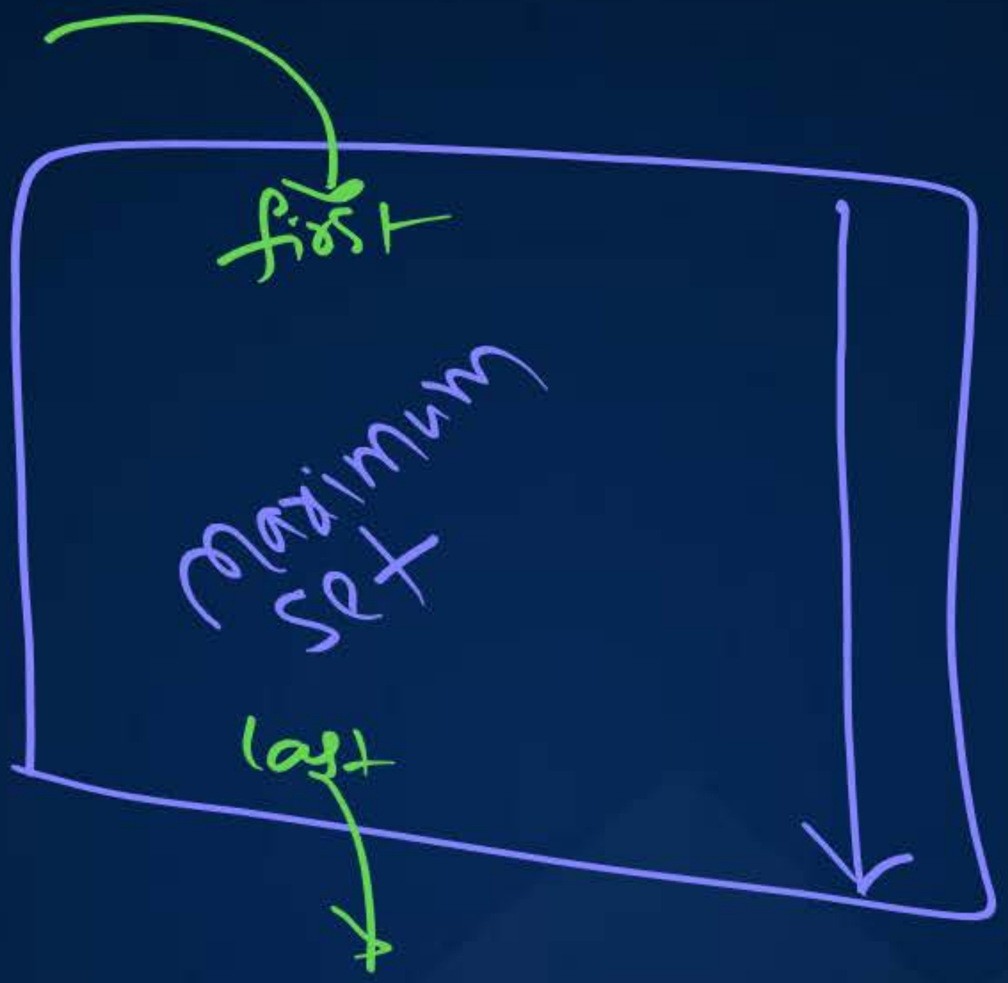
# Control Flow Graph :

- → It is Flow graph to represents "controls"
- → It is collection of ***Basic Blocks & controls
- → It is collection of nodes & edges

## Basic Block :

- → Set of statements
- → Maximum set of statements executed Sequentially
- → Control only enters at 1$^{st}$ statement (leader) and leaves after last statement in set.


Begins
1
2
3
4
5
6
Sequential
leaves after last stmt.



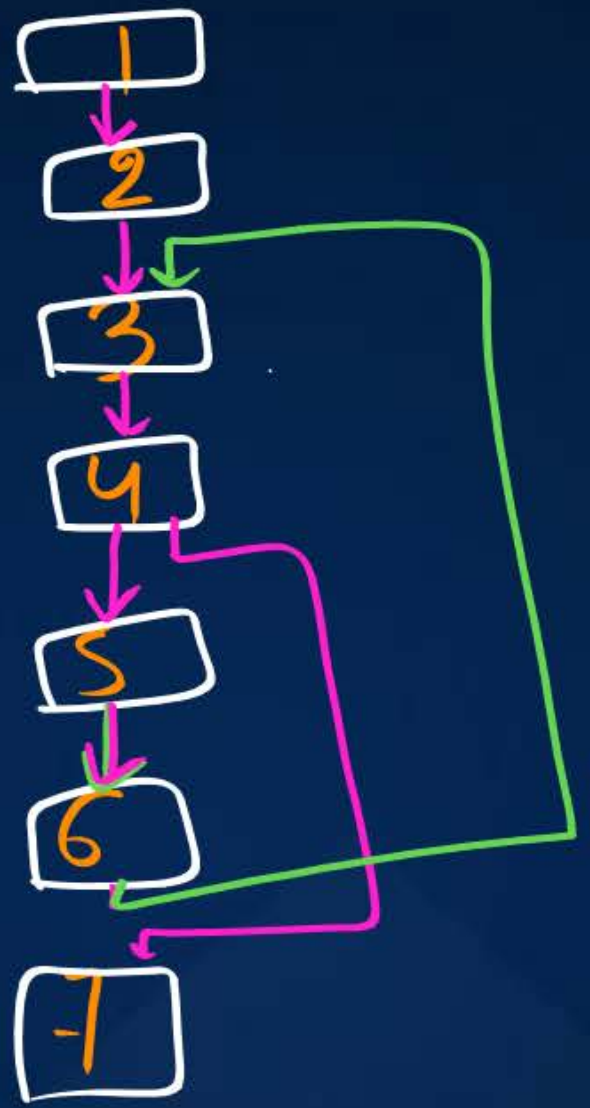

I) Maximum statements

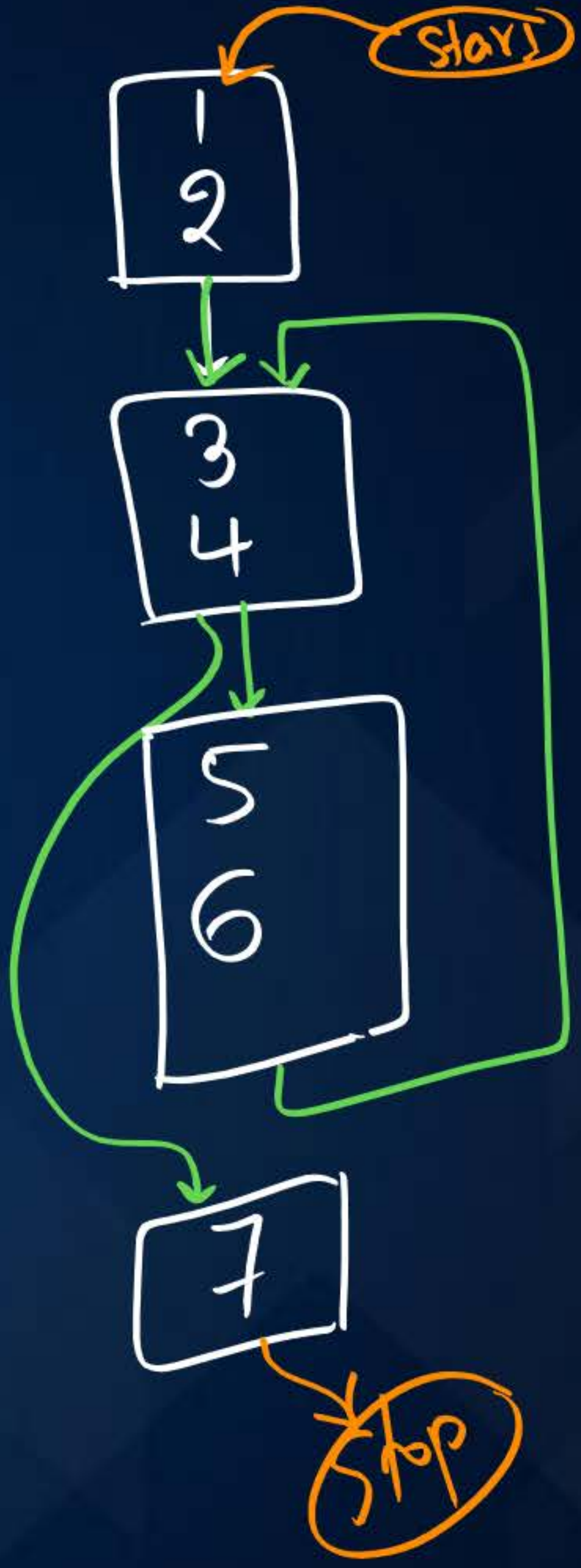
II) Sequential execution

III) Control only enters at $1^{st}$

IV) Control leaves after last statement

1. $x = a+1$
2. $y = x*2$
3. $z = b-y$
4. if $(y>z)$ goto 7
5. $x = x-1$
6. $y = y-1$, goto 3
7. $a = a+y$




4 Basic Blocks
4 controls

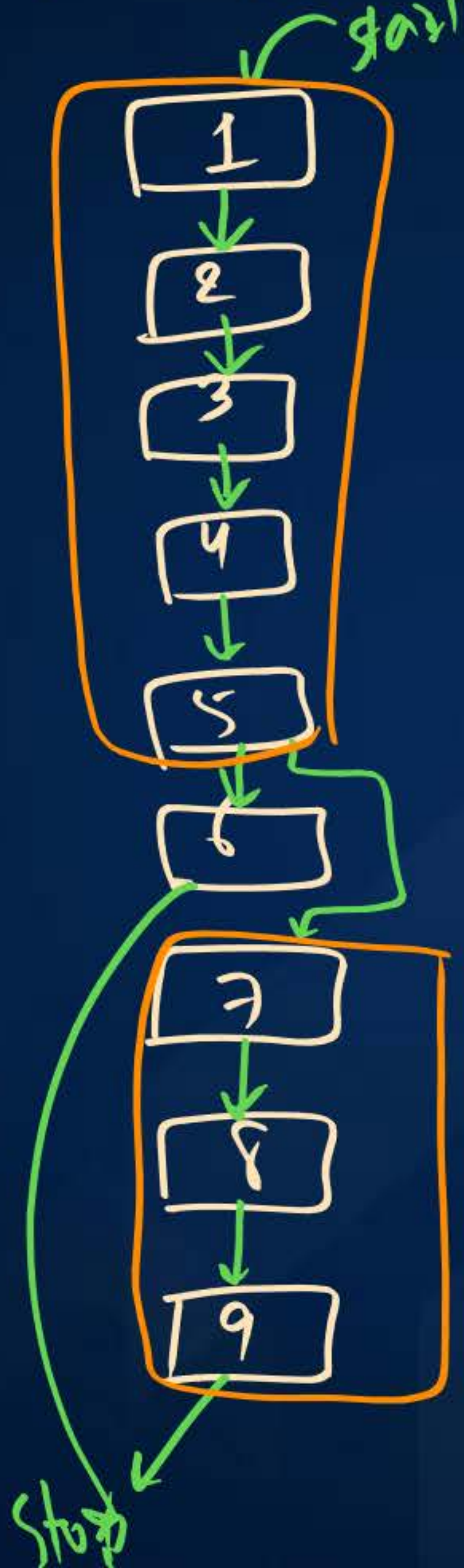
6 nodes
6 edges

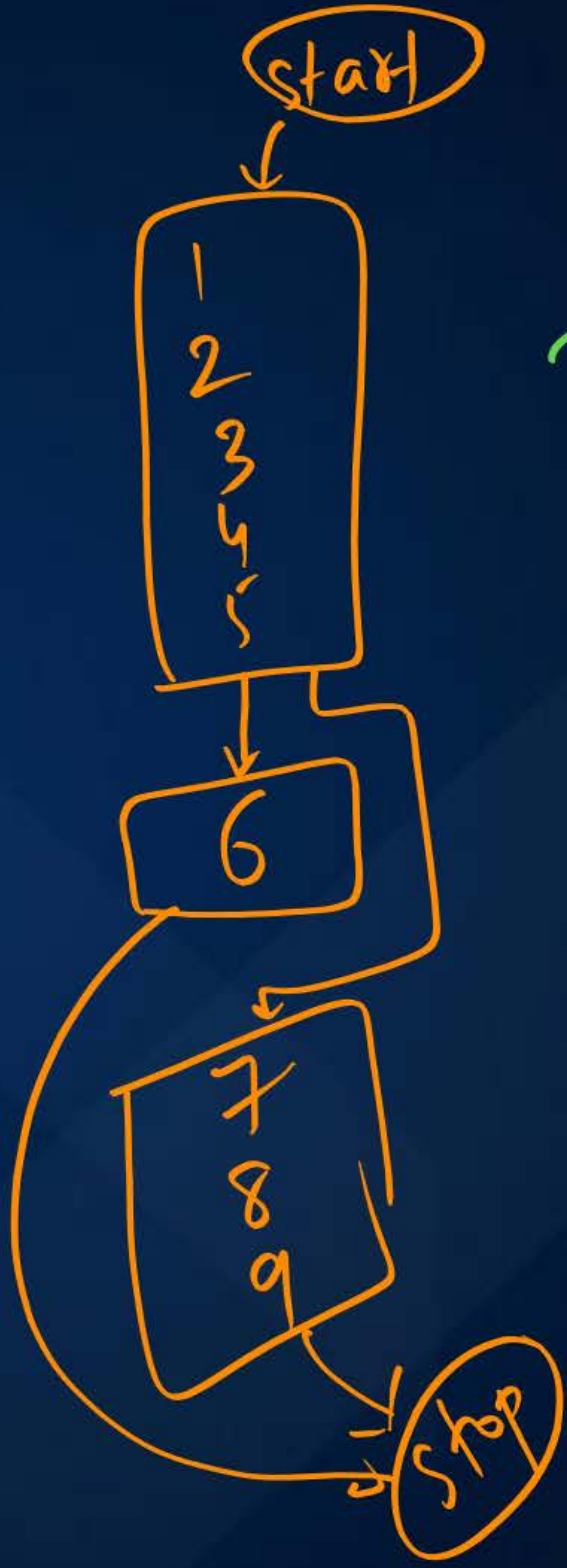
Total nodes = 6

Total nodes exclude Start & Stop = 4

② Find no. of nodes & edges

1. c=a+b
2. d=c*a
3. e=c+a
4. x=c*c
5. if (x>a)
6. { y=a*a }
7. else
8. { d=d*d;
9. e=e*e;
}

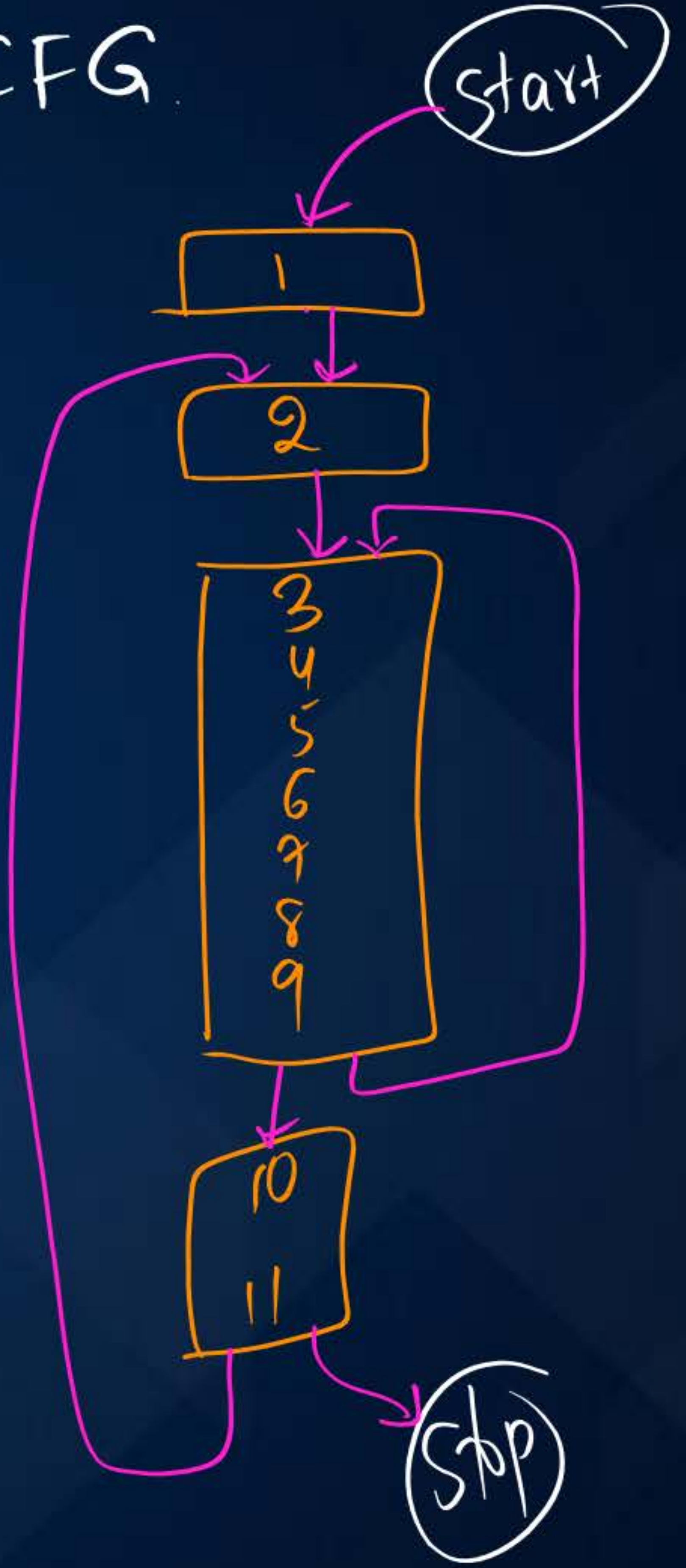



3 basic blocks
5 nodes
5 edges

③ Find no. of nodes & edges in CFG.

1. $i = 1$
2. $j = 1$
3. $t_1 = 5 * i$
4. $t_2 = t_1 + j$
5. $t_3 = 4 * t_2$
6. $t_4 = t_3$
7. $a[t_4] = 1$
8. $j = j + 1$
9. if ($j <= 5$) goto 3
10. $i = i + 1$
11. if ($i < 5$) goto 2


4 BBs
6 nodes
7 edges (controls)

4 Find no. of nodes & edges in CFG.

Start

1. $i = 1$
2. $j = 1$
3. $t_1 = 5 * i$
4. $t_2 = t_1 + j$
5. $t_3 = 4 * t_2$
6. if $(t_4 == t_3)$ goto 4
7. $a[t_4] = 1$
8. $j = j + 1$
9. if $(j <= 5)$ goto 3
10. $i = i + 1$
11. if $(i < 5)$ goto 2

Stop

Home work

# Code Generator :

Intermediate code (M/C Independent) $\longrightarrow$ $\boxed{\text{Code Generator}}$ $\longrightarrow$ Assembly code (M/C Dependent)

M/C Dependent phase

$x = x + y$

x and y are variables

ADD $R_1$, [1000]

x and y are registers/memory addresses

ADD $R_1$, 15

Assembly Instruction

- Type of Instruction
- Type of operation
- Type of addressing mode
- Data
- Memory Address / Register

Operands

Depends on m/c Architecture

Register
↳ speed

Memory
↳ Not speed compared to Register

ADD $R_1, R_2$

$x = x + y$

$\boxed{\text{Code Generator}}$

↳ Register Allocation
↳ Graph coloring
↳ To allocate min no. of registers

We have only 1 register

How many min memory spills required ?

$a = a + \boxed{b}$

$b = a * c$

$R_1 \leftarrow a$

$Mem_1 \leftarrow b$

$R_1 \leftarrow R_1 + mem_1$

$Mem_1 \leftarrow c$ (Reuse $mem_1$ to avoid more spills)

$R_1 \leftarrow R_1 * mem_1$

$\Rightarrow 1$ //

$$a = a + \boxed{b}$$
$$b = a * c$$

we have only 2 register

How many min memory (Register) spills required ?

$$R_1 \leftarrow a$$
$$R_2 \leftarrow b$$
$$R_1 \leftarrow R_1 + R_2$$
$$R_2 \leftarrow c$$
$$R_1 \leftarrow R_1 * R_2$$

$$= 0$$

## ICG

Analyzed C code → $\boxed{\text{ICG}}$ → Intermediate code

## CG

Intermediate code → $\boxed{\text{CG}}$ → Assembly code

3AC
or
CFG

# Three Address Code :

→ To store 3AC, we have 3 data structures.

## ① Triple Notation

$x = y * z$
$y = x + a$
$z = (-y)$

| Address | Operator | Left operand$_1$ | Right operand$_2$ |
|---|---|---|---|
| 1000 | * | y | z |
| 1010 | + | [1000] | a |
| 1015 | − | | [1010] |

## ② Quadruple Notation

| | operator | Left operand | Right operand | Result |
|---|---|---|---|---|
| 5000 | * | y | z | x |
| 5012 | + | x | a | y |
| 5020 | − | | y | z |

## ③ Indirect Triple

| | |
|---|---|
| 6000 | [1000] |
| 6005 | [1010] |
| 6010 | [1015] |

Triple Notation

| | | | |
|---|---|---|---|
| 1000 | * | y | z |
| 1010 | + | [6000] | a |
| 1015 | − | | [6005] |

## Triple Notation

$x = y * z$
$y = x + a$
$z = (-y)$

| Address | Operator | Left operand$_1$ | Right operand$_2$ |
|---|---|---|---|
| 1000 | * | y | z |
| 1010 | + | [1000] | a |
| 1015 | − | | [1010] |

1000 : (*, y, z)
1010 : (+, [1000], a)
1015 : (−, , [1010])

(1)

$$x = a+b$$
$$y = a+b$$
$$z = x-y$$
$$p = z+c$$
$$q = p+d$$

$\vdots$

$$\Rightarrow q = [(x-y)+c]+d$$
$$= [\cancel{(a+b)} - \cancel{(a+b)}] + c + d$$
$$= \boxed{q = c+d}$$

$\underbrace{q = c+d}$ 3 variably in SSA

$\underbrace{c = c+d}$ 2 variably 3AC

2

$$x = \underbrace{y+z}_{\text{Commutative}} - y + a$$

$$= z + \underbrace{\cancel{y} - \cancel{y}}_{\text{Cancellation}} + a$$

$$= z + a$$

$$\Rightarrow \underbrace{x = z + a}$$

Algebraic laws

Reordering

Common sub expressions

Find no. of variable in SSA

exp given $\Rightarrow$ Best 3AC $\Rightarrow$ Best SSA

3AC given $\Rightarrow$ exp / Best 3AC $\Rightarrow$ Best SSA code

DAG given $\Rightarrow$ Best 3AC $\longrightarrow$ Best SSA code

Syntax Tree $\Rightarrow$ exp $\Rightarrow$ Best 3AC $\Rightarrow$ Best SSA code

# Summary

CFG ✓

TAC ✓


THANK YOU
GW
SOLDIERS !