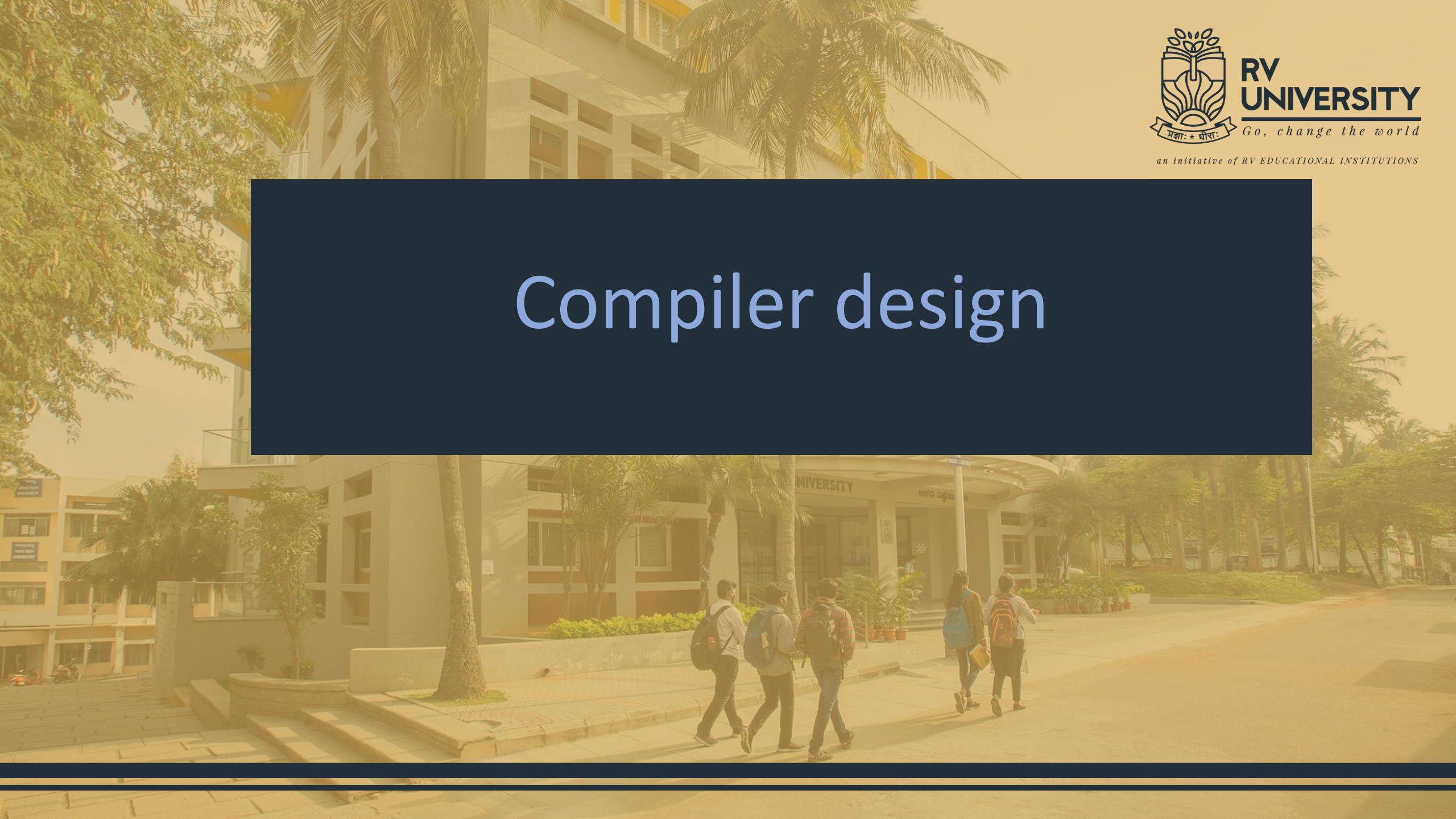
RV UNIVERSITY

Go, change the world

an initiative of RV EDUCATIONAL INSTITUTIONS

NEW-AGE GLOBAL UNIVERSITY FOR LIBERAL EDUCATION



Compiler design

CODE OPTIMIZATION

IT IS AN OPTIONAL PHASE IN THE COMPILATION PROCESS WHICH IMPROVES EXECUTION EFFICIENCY OF TARGET PROGRAM AND THE REDUCTION OF ITS SIZE.

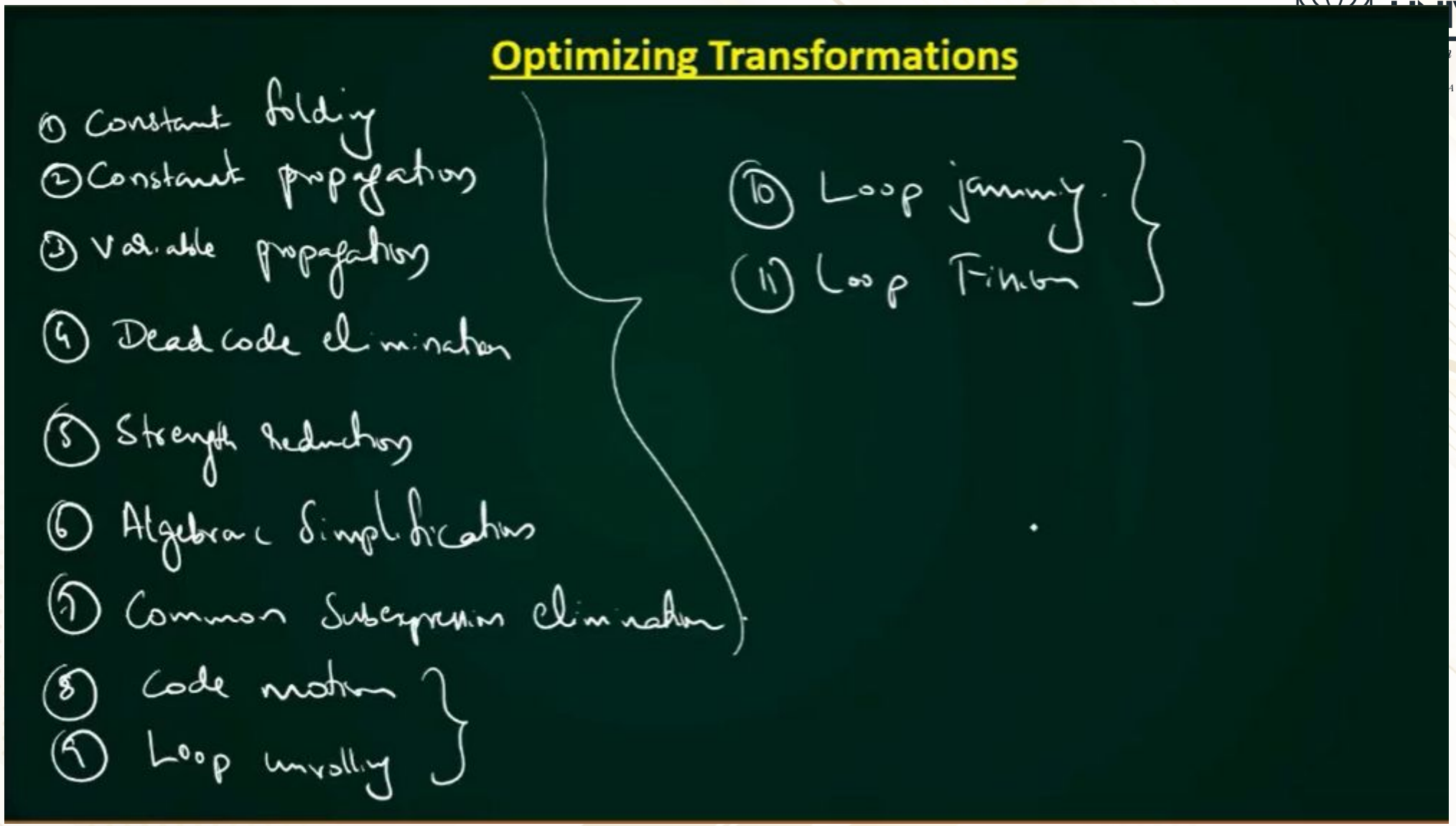
CLASSIFICATION OF OPTIMIZATION

SCOPE OF OPTIMIZATION:

- LOCAL OPTIMIZATIONS
- GLOBAL/INTRAPROCEDURAL OPTIMIZATION
- INTERPROCEDURAL/WHOLEPROGRAM OPTIMIZATION
- LOOP OPTIMIZATIONS
- PEEPHOLE OPTIMIZATIONS

LANGUAGE DEPENDENT *VS.* LANGUAGE INDEPENDENT OPTIMIZATION:

MACHINE DEPENDENT *VS.* MACHINE INDEPENDENT OPTIMIZATION:


Optimizing Transformations
1 Constant folding
2 Constant propagation
3 Variable propagation
4 Dead code elimination
5 Strength Reduction
6 Algebraic Simplifications
7 Common Subexpression elimination
8 Code motion
9 Loop unrolling
10 Loop jamming
11 Loop Fission

Constant Folding

Eg1.- area = (22.0/7.0) * r * 2;

x

area = x * r * 2

Eg2: x = y + 2 + 3;

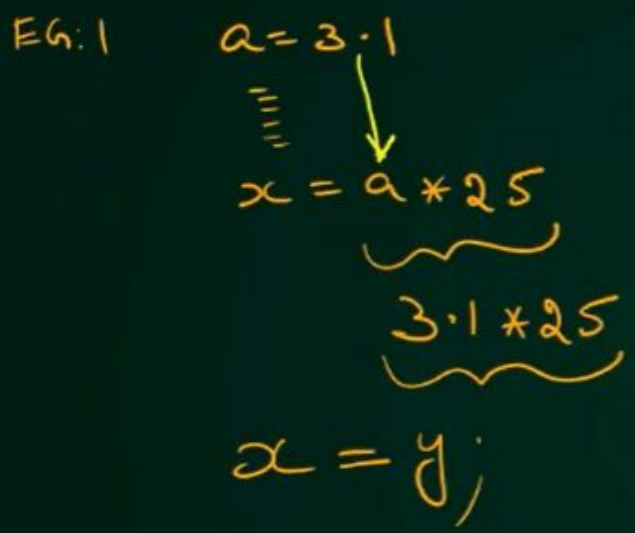
x = y + 5;

REPLACING A RUNTIME COMPUTATION WITH THE COMPILE TIME COMPUTATION IS KNOWN AS CONSTANT FOLDING.

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

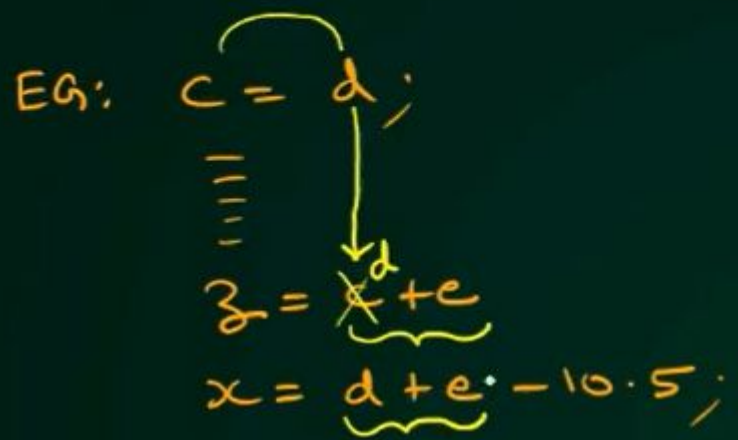
Constant Propagation

EG: 1



REPLACING CONSTANT VARIABLES BY THEIR ASSOCIATED VALUES KNOWN AT COMPILE TIME IS KNOWN AS CONSTANT PROPAGATION.

Variable Propagation



VARIABLE PROPAGATION IS SIMILAR TO CONSTANT PROPAGATION AND IT IS REPLACING A VARIABLE BY ANOTHER VARIABLE HOLDING IDENTICAL VALUE.

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS

Dead Code Elimination

Eg1:- int flag = 0;

flag is not changing

```
if (flag)
{
    block 1
}
else
{
    block 2
}
```

block 1 → DEAD CODE

Eg2:-

```
#define x 0
if (x)
{
    block
}
```

block → DEAD CODE

Eg:3

1. x = x; DEAD CODE
2. x = x+1; DEAD CODE
3. x = x+0; DEAD CODE

Strength Reduction

EG: 1 `x = 2 * y;`

⇓

`x = y + y;`

EG: 2 `x = y * 2^2`

⇓

`x = y << 2`

`x << 3` → `x * 2^3`

`x << 2` → `x * 2^2`

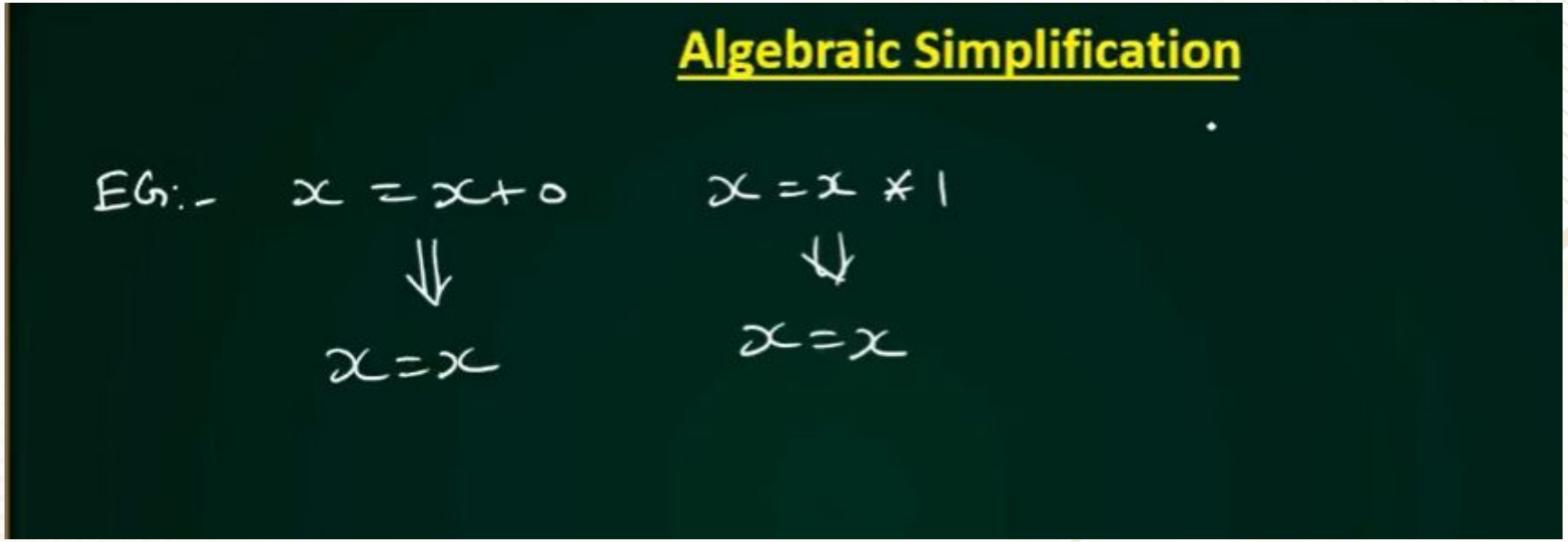
`x << 1` → `x * 2^1`

`x >> 1` → $\frac{x}{2^1}$

`x >> 2` → $\frac{x}{2^2}$

REPLACING A HIGH STRENGTH OPERATOR WITH LOW STRENGTH OPERATOR WITHOUT CHANGING THE MEANING OF A PROGRAM.

RV UNIVERSITY
Go, change the world
an initiative of RV EDUCATIONAL INSTITUTIONS



Common Subexpression Elimination

EG1:- $a = b + c$

$b = a - d$

$c = b + c$

$d = a - d$

$\Rightarrow$

$a = b + c$

$b = a - d$

$c = b + c$

$d = b$

$\Rightarrow$

$a = b + c$

$b = a - d$

$c = a$

$d = b$

EG: 2; $c = -b * a + -b * a$

Three address Code

$t_1 = -b$

$t_2 = t_1 * a$

$t_3 = -b$ X

$t_4 = t_3 * a$ X

$t_5 = t_2 * t_4$

$c = t_5$

$\Rightarrow$

$t_1 = -b$

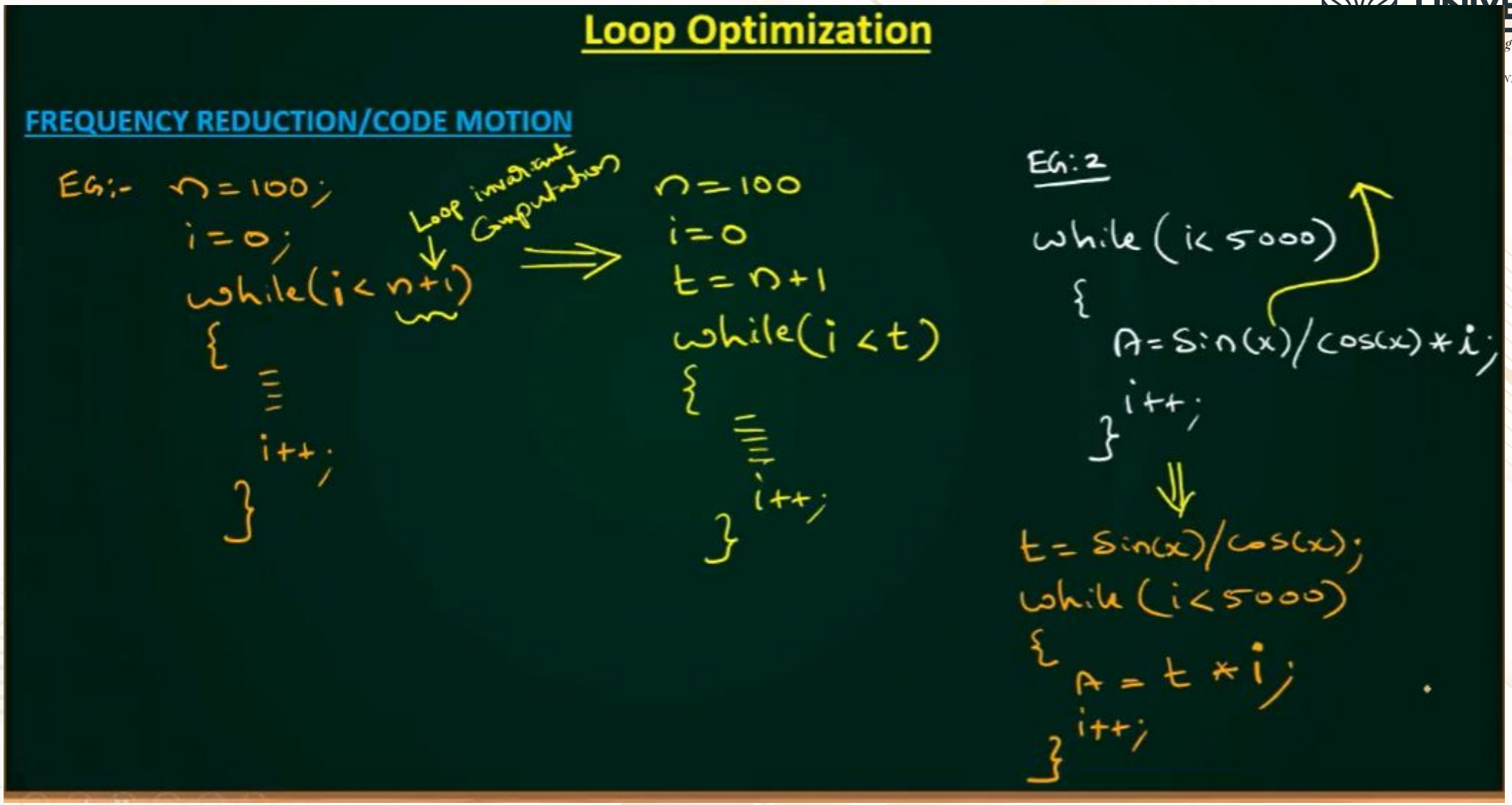
$t_2 = t_1 * a$

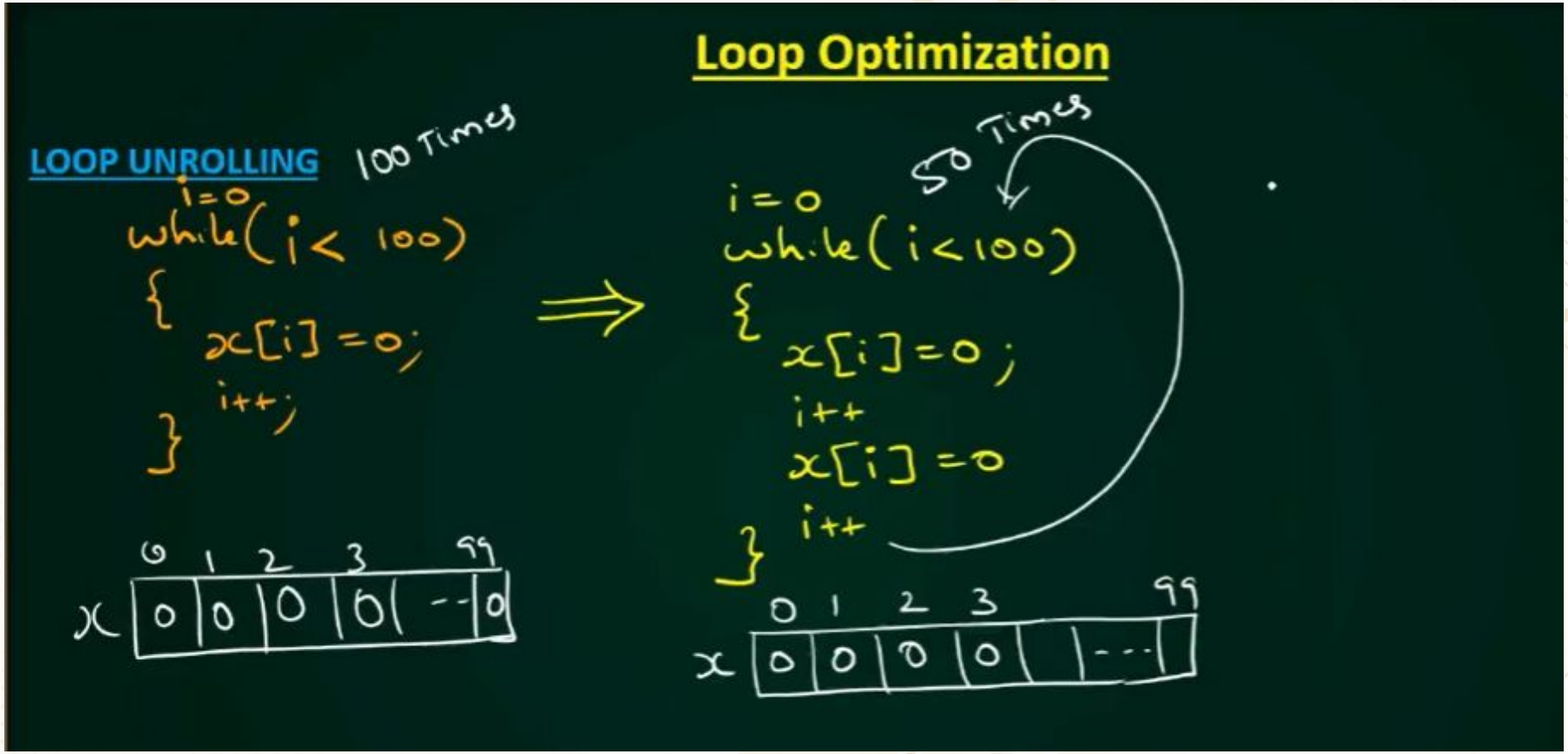
$t_5 = t_2 * t_2$

$c = t_5$

Loop Optimization

FREQUENCY REDUCTION/CODE MOTION




Loop Optimization
LOOP UNROLLING
100 Times
i=0
while(i<100)
{
x[i]=0;
i++;
}
0 1 2 3 99
x 0 0 0 0 -- 0
50 Times
i=0
while(i<100)
{
x[i]=0;
i++
x[i]=0
i++
}
0 1 2 3 99
x 0 0 0 0 ---

Loop Optimization

LOOP FUSION/JAMMING/COMBINING

Eg:-

```
for(i=0; i<10; i++)
    x[i] = 0;
for(j=0; j<10; j++)
    y[i] = 0;
```

⇒

```
for(i=0; i<10; i++)
{ x[i] = 0;
  y[i] = 0;
}
```

Loop Optimization

LOOP FISSION

Eg:-

```
for i = 1 to 100 do
    a[i] = ...
    b[i] = ....
end loop;
```

⟹

```
for i=1 to 100 do
    a[i] = ...
end loop;
for i=1 to 100 do
    b[i] = ...
end loop;
```