B.S. Abdur Rahman Crescent Institute of Science & Technology
Deemed to be University u/s 3 of the UGC Act, 1956
GST Road, Vandalur, Chennai 600 048

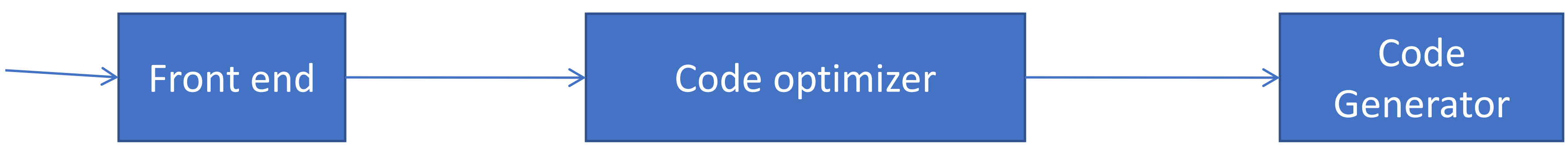
CSD 3202-COMPILER DESIGN

MODULE IV

BASIC BLOCKS

CODE GENERATOR

- The final phase of a compiler is code generator
- It receives an intermediate representation (IR) with supplementary information in symbol table
- Produces a semantically equivalent target program
- Code generator main tasks:
  - Instruction selection
  - Register allocation and assignment
  - Instruction ordering



Basic blocks and flow graphs

- Partition the intermediate code into basic blocks
  - The flow of control can only enter the basic block through the first instruction in the block. That is, there are no jumps into the middle of the block.
  - Control will leave the block without halting or branching, except possibly at the last instruction in the block.
- The basic blocks become the nodes of a flow graph

Basic blocks

- A *basic block* is a sequence of consecutive statements in which flow of control enters at the beginning and leaves at the end without any halt or possibility of branching except at the end.
- The following sequence of three-address statements forms a basic block:

$t_1 := a * a$

$t_2 := a * b$

$t_3 := 2 * t_2$

$t_4 := t_1 + t_3$

$t_5 := b * b$

$t_6 := t_4 + t_5$

Basic block Construction

Algorithm: Partition into basic blocks

- **Input:** A sequence of three-address statements
- **Output:** A list of basic blocks with each three-address statement in exactly one block
- **Method:**
- We first determine the set of *leaders*, the first statements of basic blocks. The rules we use are of the following:
  - The first statement is a leader.
  - Any statement that is the target of a conditional or unconditional goto is a leader.
  - Any statement that immediately follows a goto or conditional goto statement is a leader.
- For each leader, its basic block consists of the leader and all statements up to but not including the next leader or the end of the program.

Example-Basic block Construction

- Consider the following source code for dot product of two vectors a and b of length 20.

```
begin
prod :=0; i:=1;
do begin
prod :=prod+ a[i] * b[i]; i :=i+1;
end
while i <= 20
end
```

- The three-address code for the above source program is given as :

(1)	**prod := 0**	
(2)	i := 1	
(3)	t_1 := 4* i	
(4)	t_2 := a[t_1]	/*compute a[i] */
(5)	t_3 := 4* i	
(6)	t_4 := b[t_3]	/*compute b[i] */
(7)	t_5 := t_2*t_4	
(8)	t_6 := prod+t_5	
(9)	prod := t_6	
(10)	t_7 := i+1	
(11)	i := t_7	
(12)	**if i<=20 goto (3)**	

Basic block 1: Statement (1) to (2)

Basic block 2: Statement (3) to (12)

Optimization of Basic Blocks:

- Optimization process can be applied on a basic block. While optimization, we don't need to change the set of expressions computed by the block.
- There are two type of basic block optimization. These are as follows:
  - Structure-preserving transformations
  - Algebraic transformations

Structure preserving transformations:

a). Common subexpression elimination:

a : = b + c	a : = b + c
b : = a – d	b : = a - d
c : = b + c	c : = b + c
d : = a – d	d : = b

- Since the second and fourth expressions compute the same expression, the basic block can be transformed as above.

Structure preserving transformations:

b) Dead-code elimination:

- Suppose *x* is dead, that is, never subsequently used, at the point where the statement x : = y + z appears in a basic block.
- Then this statement may be safely removed without changing the value of the basic block.

Structure preserving transformations:

c) Renaming temporary variables:

- A statement **t : = b + c** (t is a temporary) can be changed to **u : = b + c** (u is a new temporary) and all uses of this instance of **t** can be changed to **u** without changing the value of the basic block.
- Such a block is called a *normal-form block*.

Structure preserving transformations:

d) Interchange of statements:

- Suppose a block has the following two adjacent statements:

  t1 : = b + c

  t2 : = x + y

- We can interchange the two statements without affecting the value of the block if and only if neither **x** nor **y** is $\mathbf{t_1}$ and neither **b** nor **c** is $\mathbf{t_2}$.

Algebraic transformations:

- Algebraic transformations can be used to change the set of expressions computed by a basic block into an algebraically equivalent set.

Examples:

- x : = x + 0 or x : = x * 1 can be eliminated from a basic block without changing the set of expressions it computes.
- The exponential statement x : = y * * 2 can be replaced by x : = y * y.