SNS COLLEGE OF TECHNOLOGY

(Autonomous)

COIMBATORE-35

Code Generation

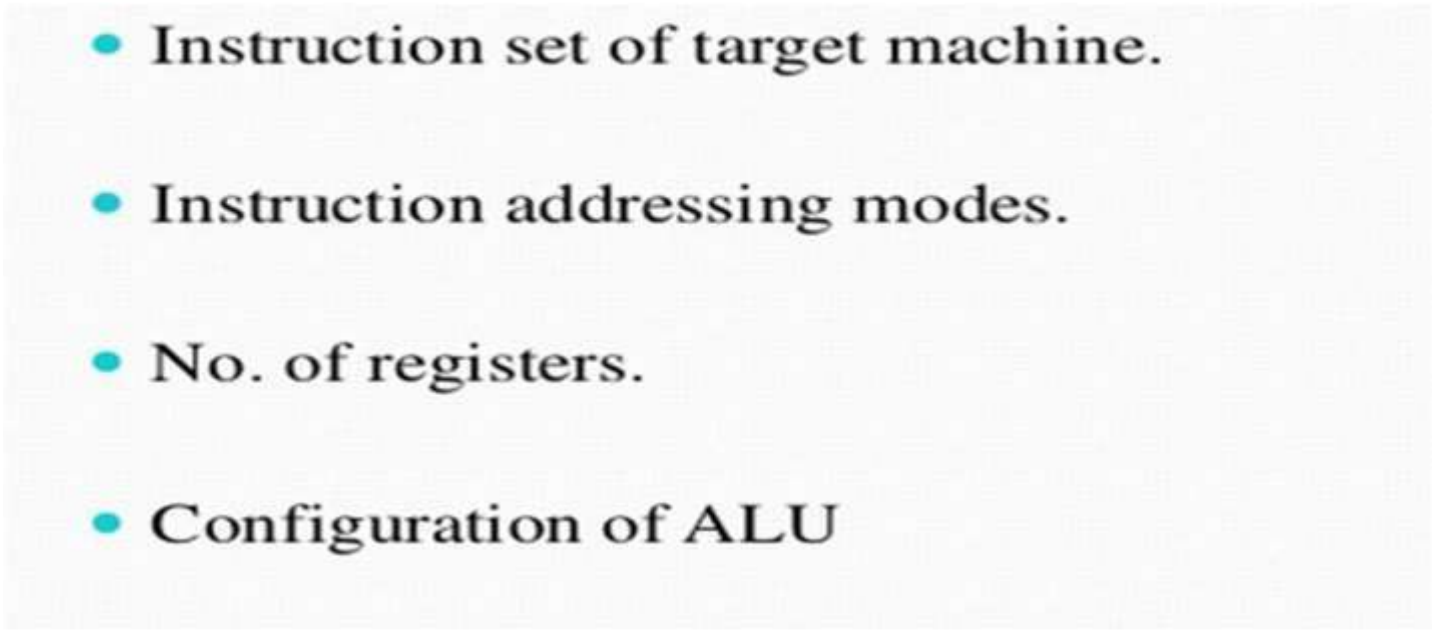
Issues in the Design of Code Generator

Code Generation

- Final phase of Compiler Design
- Optimized intermediate code is provided as input
- It generates target code
- Output code must be correct
- Output code must be high quality
- Code generator should run efficiently

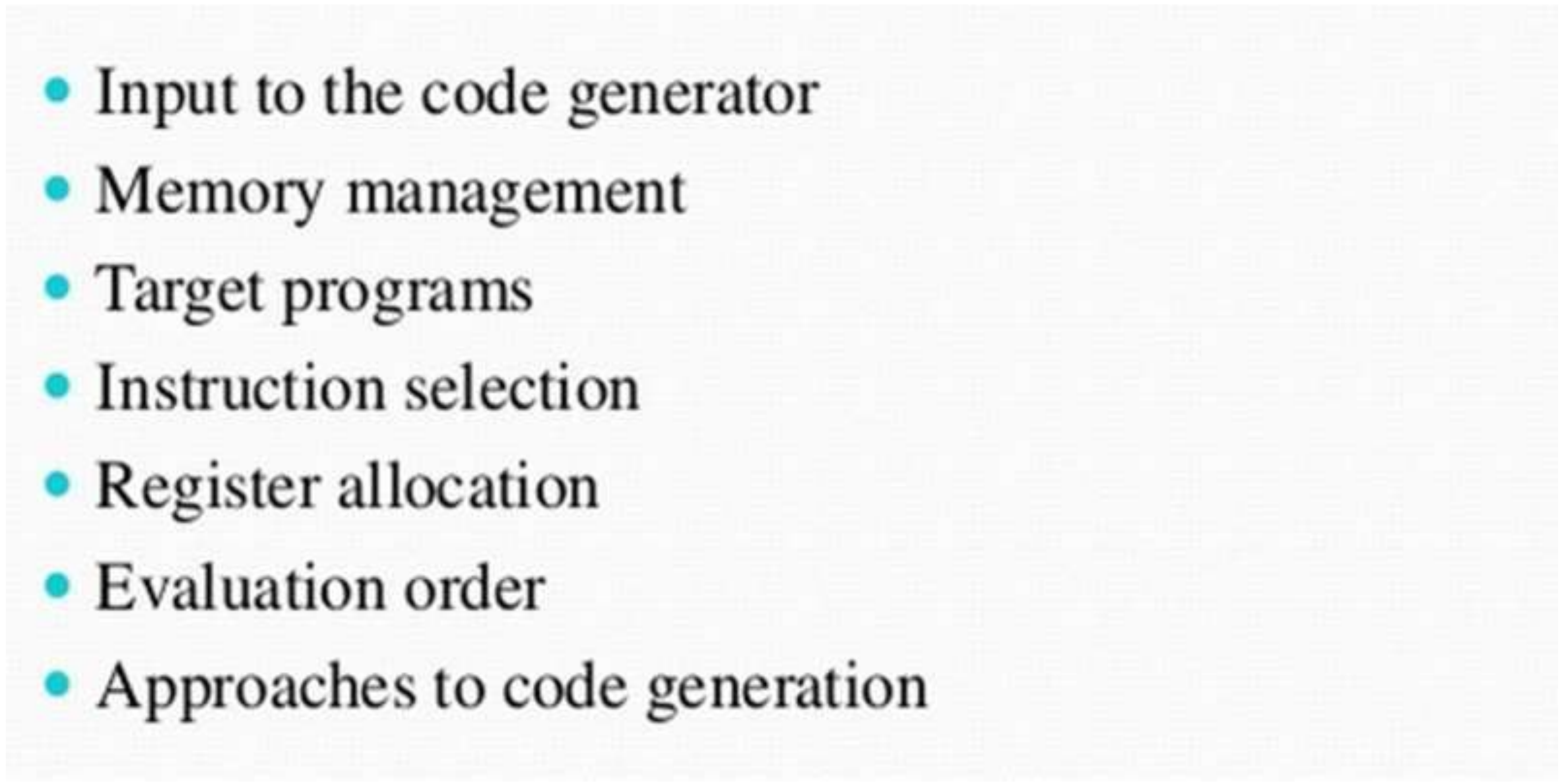
Code Generation

Prerequisites



- Instruction set of target machine.
- Instruction addressing modes.
- No. of registers.
- Configuration of ALU

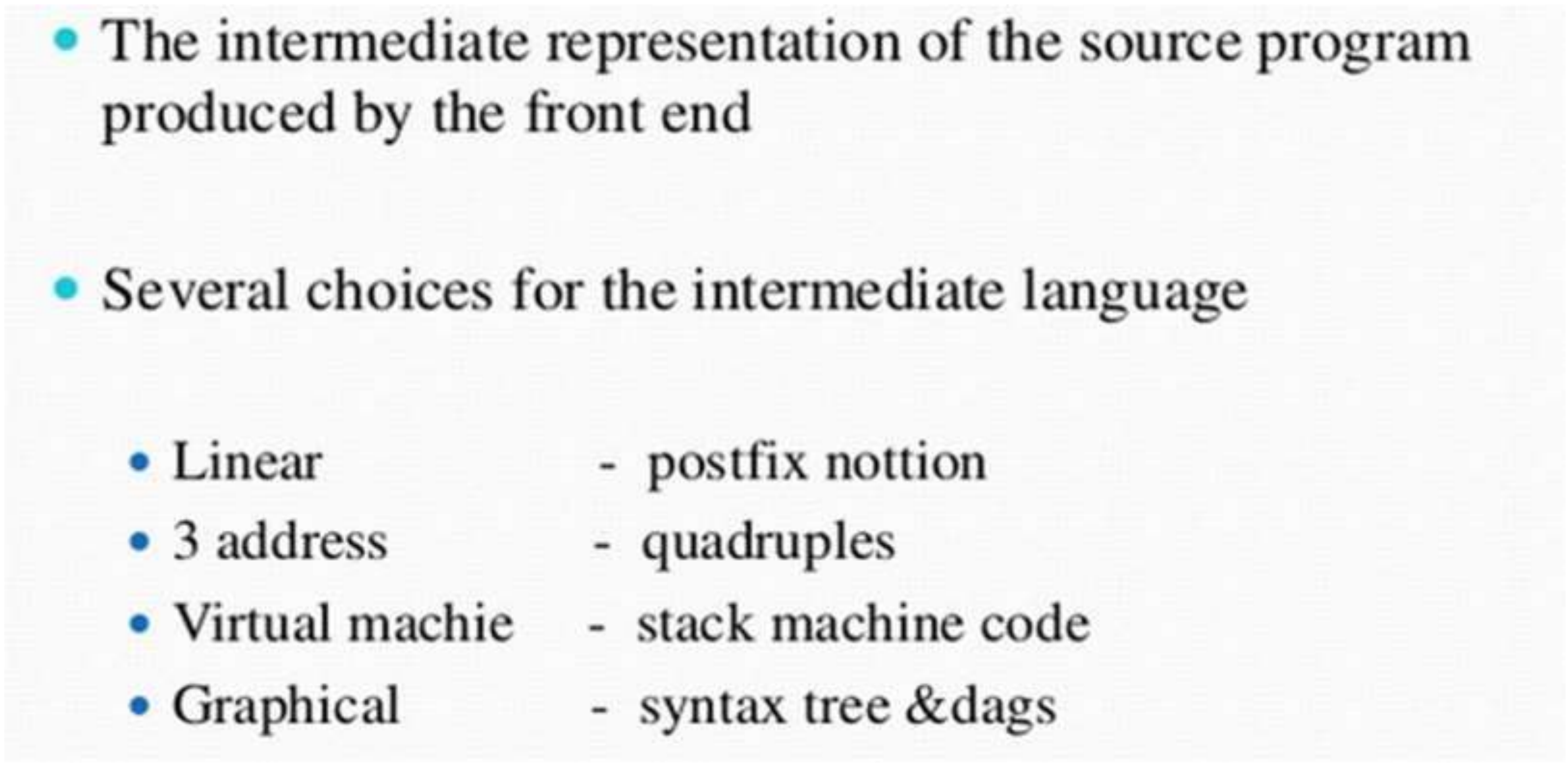
Issues in the Design of Code Generator



- Input to the code generator
- Memory management
- Target programs
- Instruction selection
- Register allocation
- Evaluation order
- Approaches to code generation

Issues in the Design of Code Generator

Input to the code Generator

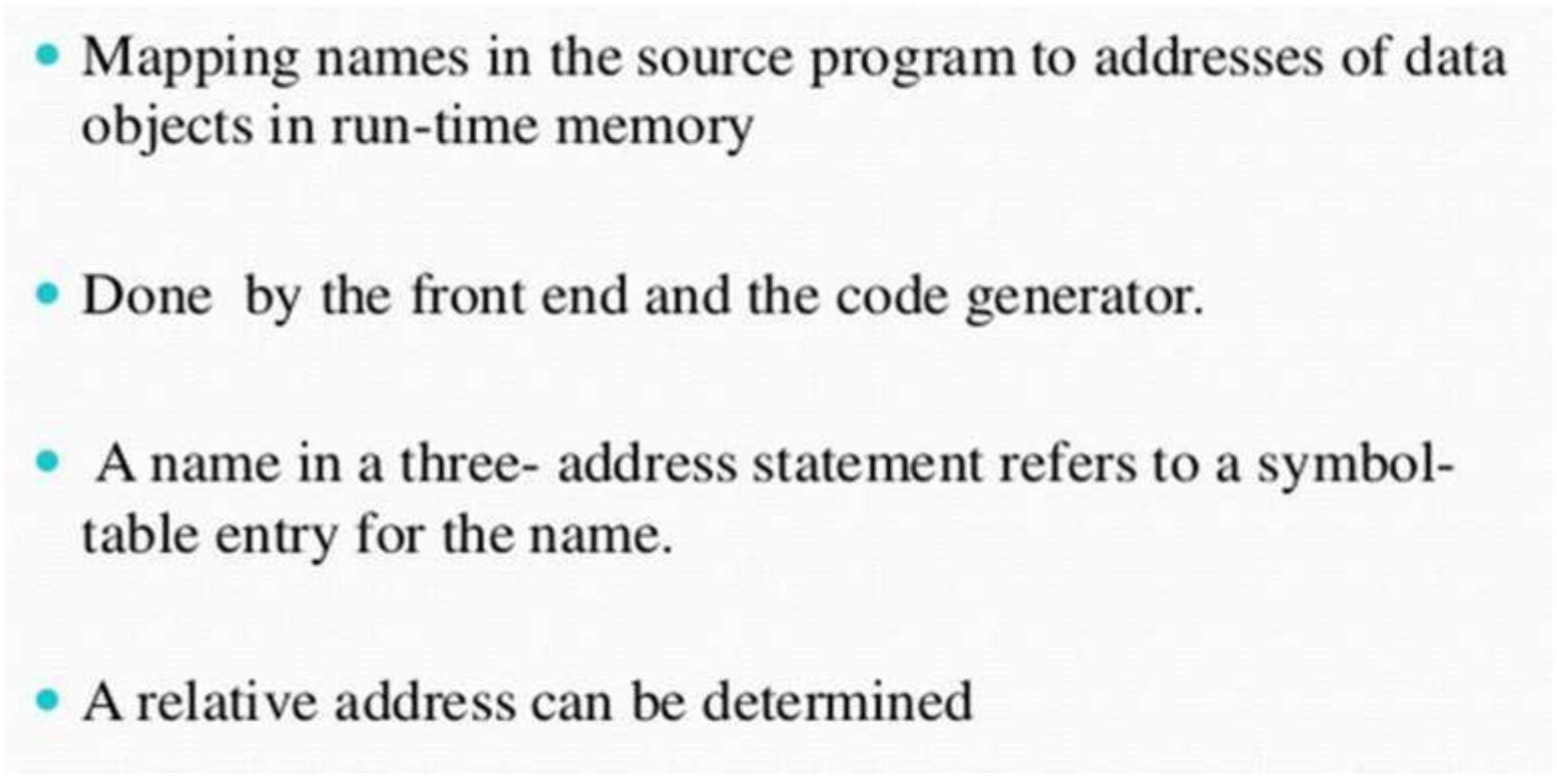


- The intermediate representation of the source program produced by the front end
- Several choices for the intermediate language

• Linear	-	postfix nottion
• 3 address	-	quadruples
• Virtual machie	-	stack machine code
• Graphical	-	syntax tree &dags

Issues in the Design of Code Generator

Memory Management



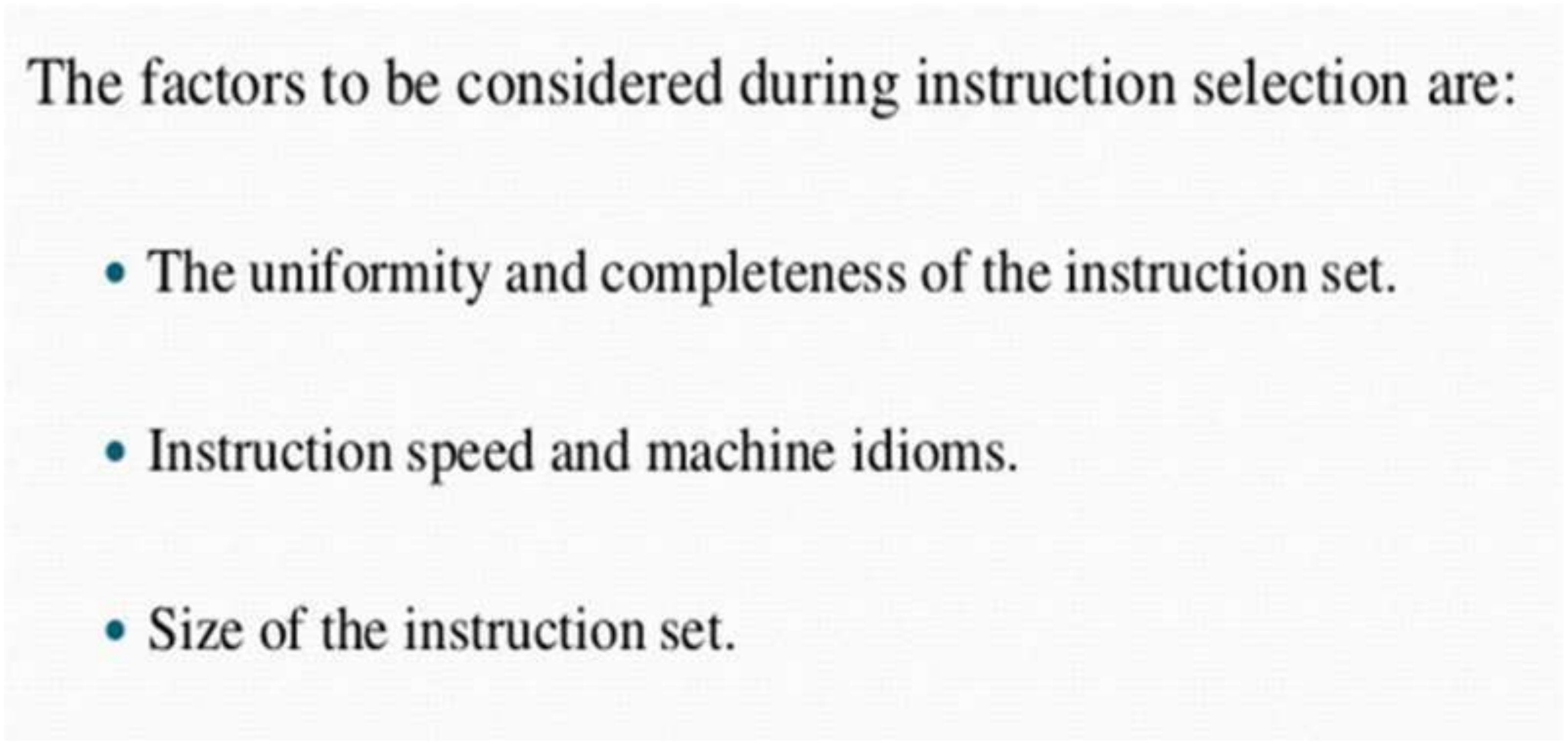
- Mapping names in the source program to addresses of data objects in run-time memory
- Done by the front end and the code generator.
- A name in a three- address statement refers to a symbol-table entry for the name.
- A relative address can be determined

Issues in the Design of Code Generator

- **Target program:**
- • The output of the code generator is the target program. The output may be : a. Absolute machine language
- - It can be placed in a fixed memory location and can be executed immediately. b. Relocatable machine language
- - It allows subprograms to be compiled separately.
- c. Assembly language
- - Code generation is made easier.

Issues in the Design of Code Generator

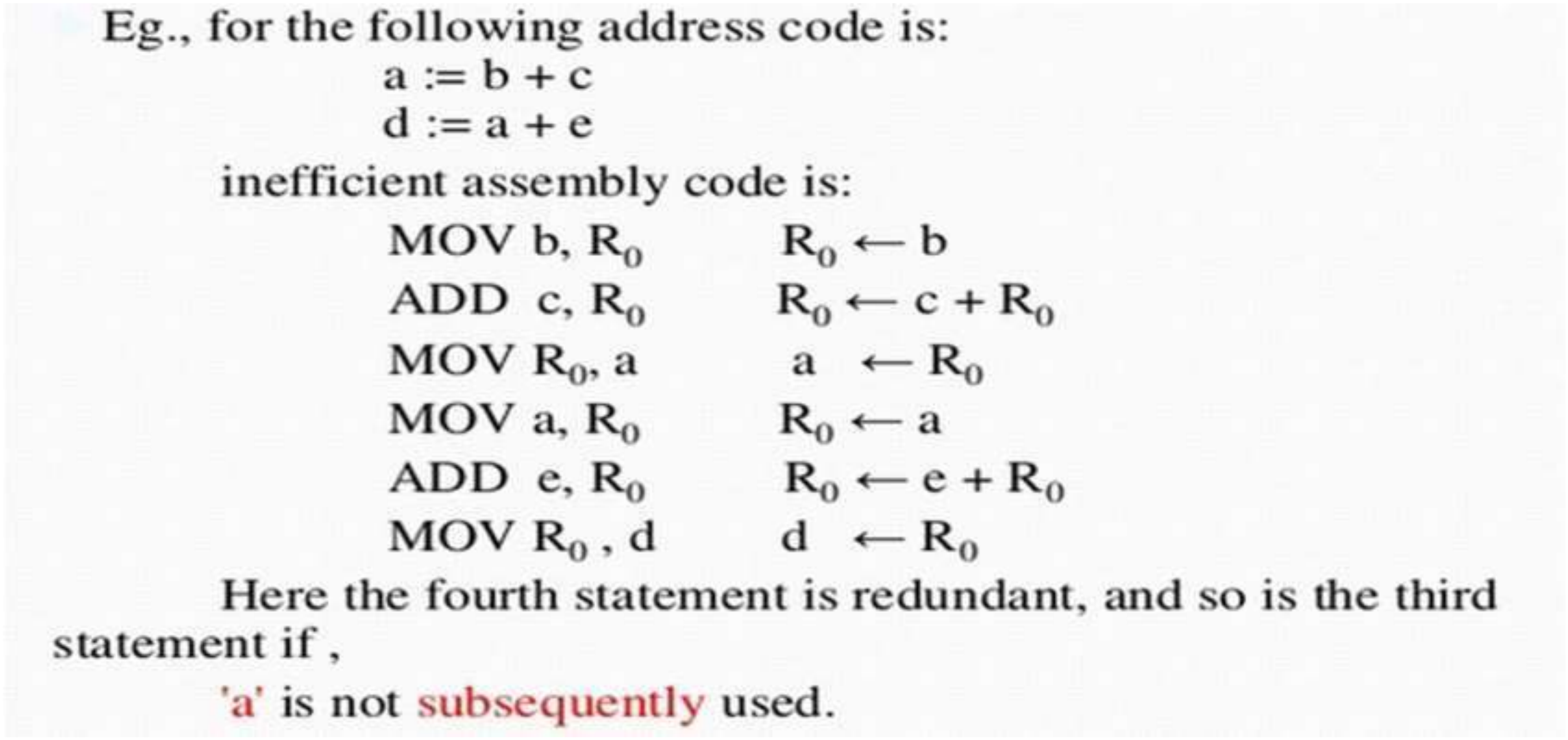
Instruction Selection


The factors to be considered during instruction selection are:

- The uniformity and completeness of the instruction set.
- Instruction speed and machine idioms.
- Size of the instruction set.

Issues in the Design of Code Generator

Instruction Selection



Eg., for the following address code is:

$a := b + c$

$d := a + e$

inefficient assembly code is:

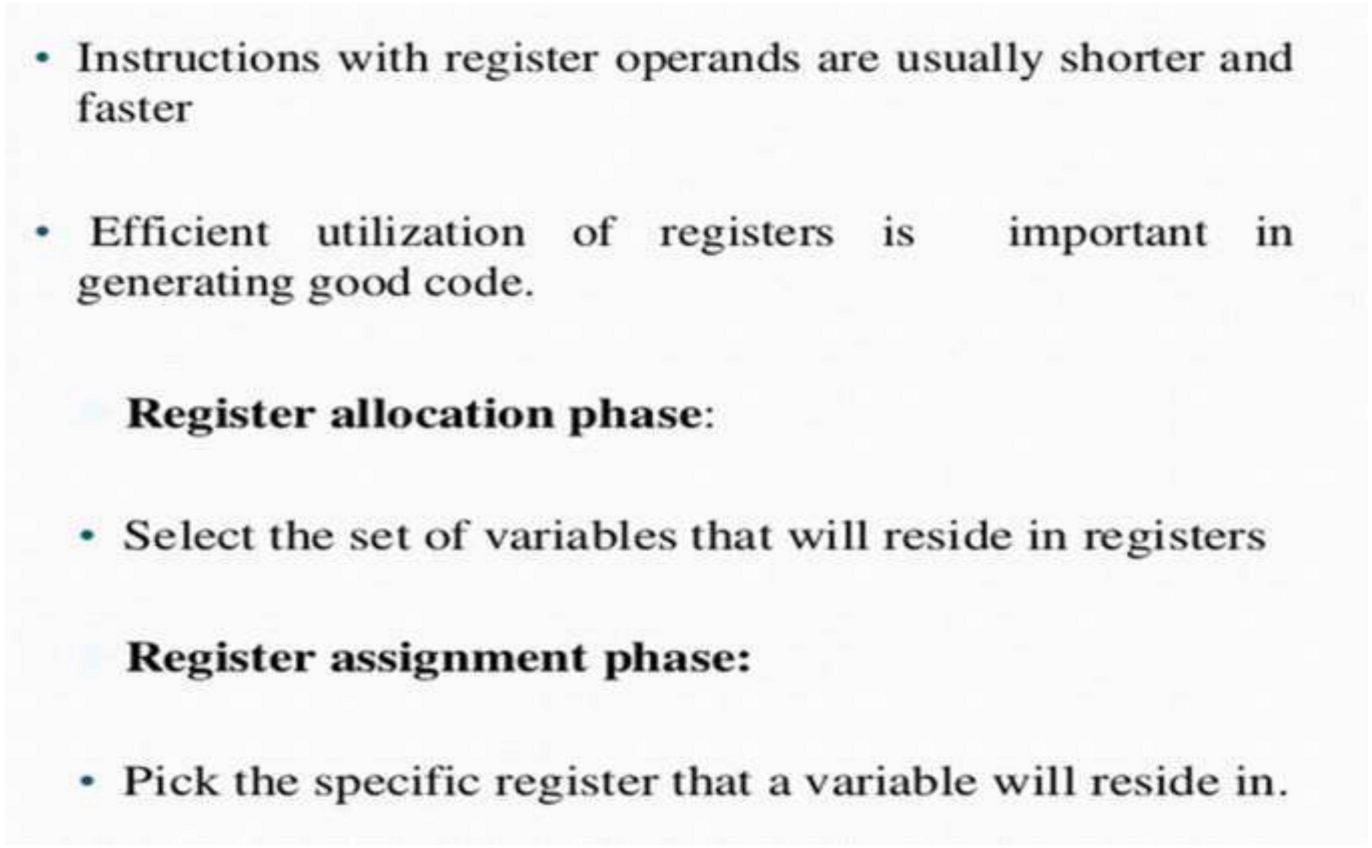
MOV b, R_0	$R_0 \leftarrow b$
ADD c, R_0	$R_0 \leftarrow c + R_0$
MOV R_0, a	$a \leftarrow R_0$
MOV a, R_0	$R_0 \leftarrow a$
ADD e, R_0	$R_0 \leftarrow e + R_0$
MOV R_0 , d	$d \leftarrow R_0$

Here the fourth statement is redundant, and so is the third statement if ,

'a' is not subsequently used.

Issues in the Design of Code Generator

Register Allocation



- Instructions with register operands are usually shorter and faster

- Efficient utilization of registers is important in generating good code.

Register allocation phase:

- Select the set of variables that will reside in registers

Register assignment phase:

- Pick the specific register that a variable will reside in.

Issues in the Design of Code Generator

- **Evaluation order**
- • The order in which the computations are performed can affect the efficiency of the target code.
-
- Some computation orders require fewer registers to hold intermediate results than others.

Summarization