LANGUAGE PROCESSORS

UNIT 11: FINAL CODE GENERATION

uc3m

David Griol Barres
dgriol@inf.uc3m.es
Computer Science Department
Carlos III University of Madrid
Leganés (Spain)

OUTLINE

- Introduction
- Issues
- Input to the Code Generator
- Target Programs/Target Machine
- Instruction Cost
- Instruction Selection
- Basic Blocks
  - Basic Blocks and Flow Graphs
  - Transformations on Basic Blocks
- Register Allocation

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

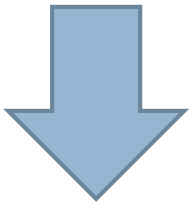
Introduction

- Is the last phase of our compiler model.
- It takes as input an intermediate representation of the source program (optimized or not) and produces as output an equivalent target program.
- The requirements are:
  - the output must be correct,
  - and of high quality.
- Optimal code generation is mathematically undecidable.
- Heuristic techniques are used for the generation of good quality code.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Issues

- Input to the Code Generator.
- Target Programs/Target Machine.
- Memory Management.
- Instruction Selection.
- Register Allocation.



MACHINE-SPECIFIC TASK

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Input to the Code Generator

- Intermediate representation of the source program.
- Symbol Table:
  - The symbol table is used to determine the run-time addresses of the data objects denoted by the names in the intermediate representation.
- The intermediate representation can be of many forms:
  - postfix notation
  - 3-address representation
  - syntax trees, etc.
- **Assumption:** the input is free of errors.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Target Programs/Target Machine

- A code generator must produce correct code.
- Absolute machine language, relocatable machine language or assembly.
- Familiarity with the target machine and its instruction set is very important for designing a good code generator.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

The Target Machine

- A register machine
- Two-address instruction: op source destination
- Address modes:

MODE	FORM	ADDRESS	ADDED COST
Absolute	M	M	1
Register	R	R	0
Indexed	c(R)	c + contents(R)	1
Indirect register	*R	contents(R)	0
Indirect indexed	*c(R)	contents(c+contents(R))	1

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Instruction Cost

- We consider the cost of an instruction to be one plus the costs associated with the source and destination address modes
- Examples:
  - MOV R0,R1 cost=1
  - MOV R2,M cost=2

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Instruction Cost

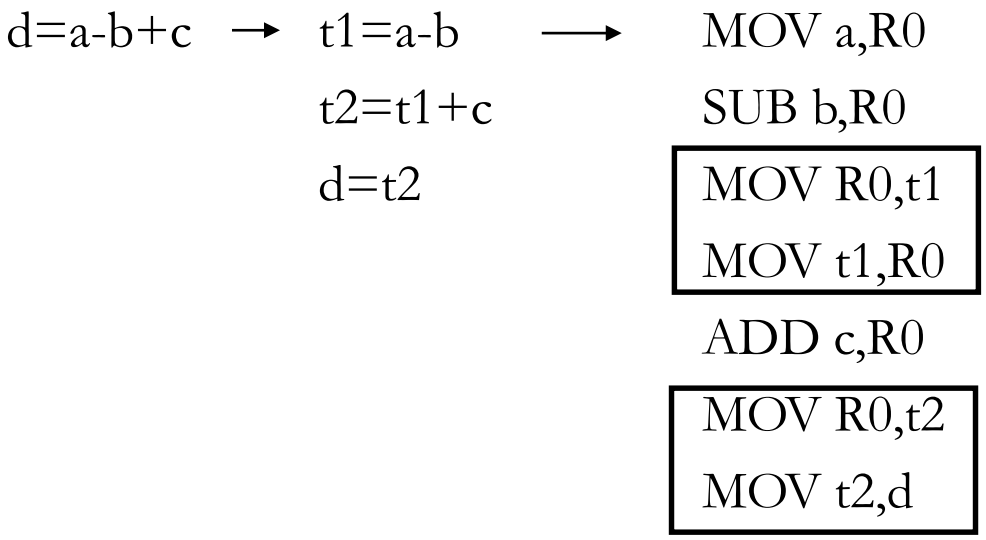
- Intermediate code: temp2 = temp1 + 2

```
MOV temp1, temp2
ADD #2, temp2
(cost=6)

ADD R0, *R1
MOV *R1, *R2
(cost=2)
```


David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Instruction Selection



- A target machine with a rich instruction set may provide several ways of implementing a given operation.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

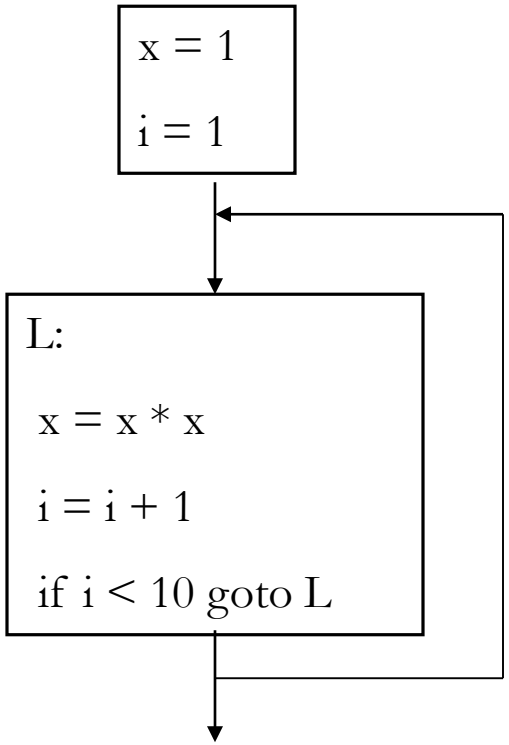
Basic Blocks

- A sequence of consecutive statements in which flow of control enters at the beginning and leaves at the end without halt or the possibility of branching excepts at the end

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Basic Blocks and Flow Graphs

- Flow Graphs represent the flow of control
- The nodes of a Flow Graph are Basic blocks
- Partition into Basic Blocks:
  1. The first statement is a leader
  2. A statement that is the target of a goto is a leader
  3. A statement that follows a goto is a leader
  4. A Basic block contains a leader and all the statements up to but not including the next leader or the end of the program



David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Transformations on Basic Blocks

- Given:
  1. L:
  2. t = 2 * x
  3. w = t + x
  4. If w > 0 goto L'
- (3) cannot be executed unless (2) has been executed before, (3) could be changed to w = 3 * x

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Register Allocation

- Instructions involving register operands are usually shorter and faster than those involving operands in memory.

- Should L be a register or a memory address?
- A simple strategy
  - Computed results can be left in registers as long as possible.
  - Everything must be stored before the end of a basic block.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es