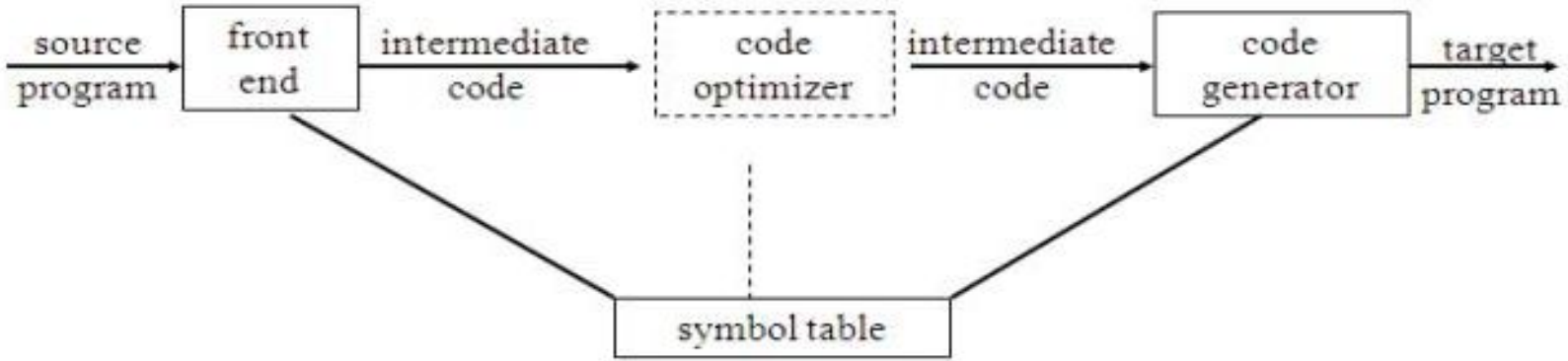
Code Generation

Code Generator

- It takes as input Intermediate Representation (IR), symbol table information and produces as output a semantically equivalent target program
- Target program should preserve the semantic meaning of high quality
- Code generator has three primary tasks
  - Instruction Selection
  - Register Allocation and Assignment
  - Instruction Ordering



Issues in the Design of a Code Generator

- Input to the Code generator
  - Code generator can proceed on the assumption - Input is free of all syntactic and static semantic errors
- The Target Program
  - RISC, CISC and stack-based target machine architectures
  - Target machine language program can be absolute or relocatable
- Instruction Selection
  - Uniformity, Completeness and Instruction speed
  - X = y + z
    - LD R0, y //R0 = y (Load y into register R0)
    - ADD R0, R0, z // R0=R0 + z (add z to R0)
    - ST x, R0 // x = R0 (Store R0 into x)

Issues in the Design of a Code Generator… Contd

- Instruction Selection
  - a= b+ c
  - d= a + e
    - LD R0, b
    - ADD R0, R0, c
    - ST a, R0
    - LD R0, a
    - ADD R0, R0, e
    - ST d, R0

a = a+1

LD R0, a
ADD R0, R0 +1
ST a, R0

INC a

Issues in the Design of a Code Generator… Contd

- Register Allocation
  - **Register allocation** - selects the set of variables that will reside in registers at each point in the program
  - **Register assignment** - picking the specific register that a variable will reside in
  - t=a+b
  - t= t*c
  - t=t/d
    - L R1,a
    - A R1,b
    - M R0,c
    - D R0, d
    - ST R1,t

t=a+b
t=t+c
t=t/d
L R0, a
A R0,b
A R0, c
SRDA R0,32
D R0,d
- ST R1, t

- Evaluation Order

Target Language

- A Simple Target Machine Model
  - Byte-addressable machine with n general-purpose registers R0, R1, …, Rn-1
  - Instruction Types
    - Load operations: LD dst, addr (dst=addr)
    - Store operations : ST x, r (x =r)
    - Computation operations: OP dst, src1, src2
    - Unconditional jumps: BR L
    - Conditional jumps: Bcond r, L
  - Addressing Modes:
    - L-value of x
    - Indexed LD R1, a(R2) (R1= Contents(a+contents(R2))
    - LD R1, 100(R2) (R1= contents(100+contents(R2)
    - LD R1, *100(R2) (R1 = contents (contents(100+contents(R2))
    - Immediate LD R1, #100 (Loads the integer 100 into register R1)

Examples

```
x= y-z      LD R1,y
            LD R2, z
            SUB R1, R1, R2
            ST x, R1
b=a[i]      LD R1, i
            MUL R1, R1, 8
            LD R2, a(R1)      //R2= contents (a+contents (R1))
            ST b, R2
```

Examples

a[j]=c

```
LD R1, c
LD R2, j
MUL R2, R2, 8
ST a(R2), R1
```

x= *p

```
LD R1,p
LD R2, 0(R1)   // R2- contents(0+contents(R1))
ST x, R2
```

Examples

If x < y goto L

```
LD R1, x
LD R2, y
SUB R1, R1, R2
BLTZ R1, L
```

Target Language

- Program and Instruction Costs
  - Interested in optimizing length of compilation time and size, running time and power consumption
  - LD R0, R1 ===> one
  - LD R0, M ===> two
  - LD R1, *100(R2) ===> two

Reference

- Aho A.V., Lam M.S., Sethi R., and Ullman J.D. Compilers: Principles, Techniques, and Tools (ALSU). Pearson Education, 2007.