Compiler Design

Assignment-1

SUBMITTED TO-

Dr. Aruna Mailk
CSE Department

SUBMITTED BY-

Name – Ankit Goyal
Roll no - 17103011
Group - G-1
Branch - CSE

Q1. What is Code optimisation? Explain machine dependent and independent code optimisation.

Ans :

The code optimization in the synthesis phase is a program transformation technique, which tries to improve the intermediate code by making it consume fewer resources (i.e. CPU, Memory) so that faster-running machine code will result. Compiler optimizing process should meet the following objectives :

- The optimization must be correct, it must not, in any way, change the meaning of the program.
- Optimization should increase the speed and performance of the program.
- The compilation time must be kept reasonable.
- The optimization process should not delay the overall compiling process.

Types of Code Optimization –The optimization process can be broadly classified into two types :

- **Machine Independent Optimization –** This code optimization phase attempts to improve the **intermediate code** to get a better target code as the output. The part of the intermediate code which is transformed here does not involve any CPU registers or absolute memory locations.
- **Machine Dependent Optimization –** Machine-dependent optimization is done after the **target code** has been generated and when the code is transformed according to the target machine architecture. It involves CPU registers and may have absolute memory references rather than relative references. Machine-dependent optimizers put efforts to take maximum **advantage** of the memory hierarchy.

Q2. Write a short note with example to optimise the code

a. Dead Code elimination

Code that is unreachable or that does not affect the program (e.g. dead stores) can be eliminated.

```
int global;
void f ()
{
  int i;
  i = 1;          /* dead store *
  global = 1;     /* dead store */
  global = 2;
  return;
  global = 3;     /* unreachable */
}
```

This is the code after Dead code elimination

```
int global;
void f ()
```

```
{
  global = 2;
  return;
}
```

b. Variable elimination

Removing extra variables, which are not used in the code.

```
//Before Optimization
c = a * b
x = a
till
d = x * b + 4

//After Optimization
c = a * b
x = a
till
d = a * b + 4
```

c. Code motion

This reduces evaluation frequency of expression.

```
a = 200;
 while(a>0)
 {
     b = x + y;
     if (a % b == 0}
     printf(“%d”, a);
   }

//This code can be further optimized as
a = 200;
b = x + y;
while(a>0)
 {
     if (a % b == 0}
     printf(“%d”, a);
   }
```

d. Reduction in strength

Strength reduction means replacing the high strength operator by the low strength.

```
i=1;
while (i<10)
{
    y = i * 4;
}
```

```
//After Reduction
i = 1
t = 4
{
   while( t<40)
```

```
    y = t;
    t = t + 4;
}
```

Q3. Explain how code motion and frequency reduction used for loop optimisation ?

Ans:

Frequency reduction is a type in loop optimization process which is machine independent. In frequency reduction code inside a loop is optimized to improve the running time of program. Frequency reduction is used to decrease the amount of code in a loop. A statement or expression, which can be moved outside the loop body without affecting the semantics of the program, is moved outside the loop. Frequency Reduction is also called Code Motion.

Objective of Frequency Reduction:

- To reduce the evaluation frequency of expression.
- To bring loop invariant statements out of the loop.

Q4. How to recognise various tokens of high level language program? Write the regular expressions and transitions diagram for each.

Ans:

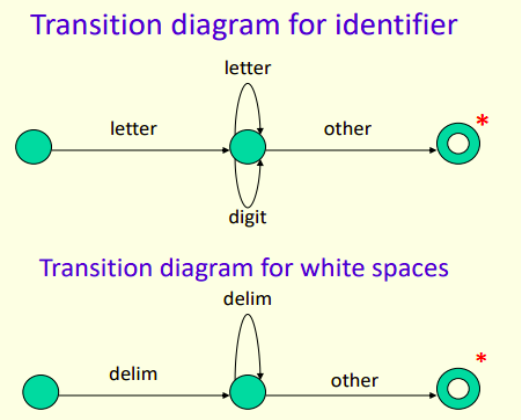
The lexical analyzer needs to scan and identify only a finite set of valid string/token/lexeme that belong to the language in hand. It searches for the pattern defined by the language rules.

Identifier :

Letter → a|b|c|…|z|A|B|…|Z
Digit → 0|1|…|9
identifier → letter(letter|digit)*

Unsigned Number in C

digit → 0|1|…|9
digits → digit^{+}
fraction → '.'digits|€
exponent → (E('+'|'-'|€)digits)| €
number → digits fraction exponent



Q5. Generate 3 address code for given pseudo code
while(i<=100){
A = A/B*20;
++i;
print(A value)
}

Ans:

```
        i = 0

L:      t1  = A/B
        t2 = t1*20
        A = t2
        i = i+1
        print(A)
        if i<=100 goto L
```