B.Karthika
20241A05I7

Assignment - 5

1. Write the differences between DAG and syntax tree.

A:

Aspect	Syntax tree	DAG
Representation	Hierarchial tree structure based on syntax rules	Directed acyclic graph representing expressions.
Program structure	Reflects syntax and nesting of language constructs.	Represents optimized and shared subexpressions.
Cyclicity	Tree structure, no cycles.	Directed graph, acyclic, may have shared nodes.
Purpose	Parsing, understanding syntax, semantic analysis.	Optimizing expressions, reducing redundancy.
Node content	Language constructs, type info, child pointers	Subexpressions, operation, operand references.
Usage in compilation phases.	Parsing, semantic analysis intermediate reps.	Optimization phases like common subexpression elimination, code generation.
Handling Redundancy	No sharing of identical subexpressions	Shares common subexpression to reduce redundancy
Optimization Focus	Structural representation	Efficiency and optimization of expressions.

2. Explain code generation algorithm with example.

A: • Register Descriptor : To perform register allocation we maintain register descriptor keeps track of what is currently in each register and we consult this register descriptor whenever we need a new register. We assume that initially the register description shows that all registers are empty.

• Address Descriptors : For each name in the block we maintain an address descriptor that keeps the locations where the current value of the name can be found at run-time. The location may be a register, a stack location, a memory address or some set of these.

• We need a location to perform the computation specified by each of the three address statements. For this purpose we use the function getreg(). getreg() returns a location for the computation performed by a three address statement.

• Eg: If ~~x = yopz~~ is to perform then getreg() returns a location 'L'. where the computation yopz should be performed and if possible it returns a register.

Algorithm for codegeneration :

• For every 3 address statement of the form $x = y \, op \, z$ in the basic block do.

① call getreg() to obtain the location L in which the computation yop z should be performed.

(ii) Obtain the current location of the operand y by consulting its address descriptor and if the value of y is currently both in the memory location as well as in the register than prefer the register. If the value of y is currently not available in L then generate instruction MOV y,L (where y has assumed to represent the current location of y).

(iii) Generate the instruction op z, L and update the address descriptor of 'x' to indicate that x is now available in L, And if 'L' is in a register than update its register descriptor to indicate that it will contain the run-time value of 'x'.

(iv) ~~If the current values of y and z are in the register & have no further uses for them and they are not live at the end of the block then alter the register descriptor to indicate that after execution of statement x=yop z those registers will no longer contain y & z.~~

(iv) If the current values of y and z are in the register & have no further use cases for them and they are not live at the end of the block then after the register descriptor to indicate that after execution of statement x=yopz those registers will no longer contain y & z.

(v) Store all the results.

3. Explain Global - Data flow Analysis.

A: Global - Data flow Analysis describes that the,

Eg: There's a variable 'A' has value 3 everytime control reaches a certain point 'p', then we can substitute 3 for each use of A at P. Knowing that the value of A is 3 3 at P may require examination of the entire program. This information is gathered by 'global dataflow analysis'.

- For this purpose it uses the following reaching definitions:

To determine the definitions that can reach a given point in a program we assign a distinct number to each definition. After this we compute the following 2 functions for each and every block B(basic block).

These are:

(i) GEN (B) : is a set of generate definitions. these define within Block B, that reach the end of block.

(ii) KILL (B): set of definitions, outside of B that define identifiers, that also have definitions within block B.

4. What are the different types of object forms? Explain the following.

(i) Register Descriptor

(ii) Address Descriptor.

A: Different types of object forms:

- Absolute machine language program: The object code produced is absolute machine language program then it can be placed in a fixed location and immediately executed. The execution of this form of object code is very fast compared to other forms.

- Relocatable machine language program: If the object code produced is relocatable machine language program then it will allow subprograms to be compiled seperately.

- Assembly language program: It is the easiest process to generate the code in Assembly language to generate code in assembly language it uses symbolic instructions and macrofacilities of Assembler.

- High level language program: producing a high level language program as output will be easier for generator.

(i) Register descriptor:

To perform register allocation we maintain register descriptor that keeps track of what is currently in each register and we consult this register desciptor whenever we need a new register we assume that initially the register description shows that all registers are empty.

(ii) <u>Address Descriptor:</u>

For each name in the block we maintain an descriptor that keeps the locations where the current value name can be found at run-time. The location may be a register, a stack locations a memory address or some set of these.

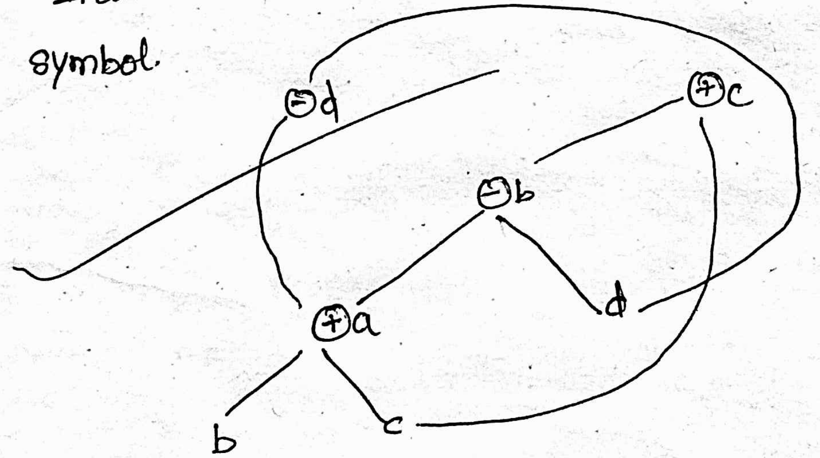
5. Construct Dag for the following block of statements:

$a = b+c$
$b = a-d$
$e = b+c$
$d = a-d$.

A. 3 rules for constructing DAG for 3 address code statements.

(i) Leaves are labeled by unique identifiers, either variable names, or constants.

(ii) Interior nodes are labeled by an operator symbol.



21/11/23

(iii) There is anode in the DAG, for each of the initial values of variables appearing in the basic block. and there is a node n associated with each statement [s] with the block.