SchoolofComputing

CIA-III Exam–JUNE2022

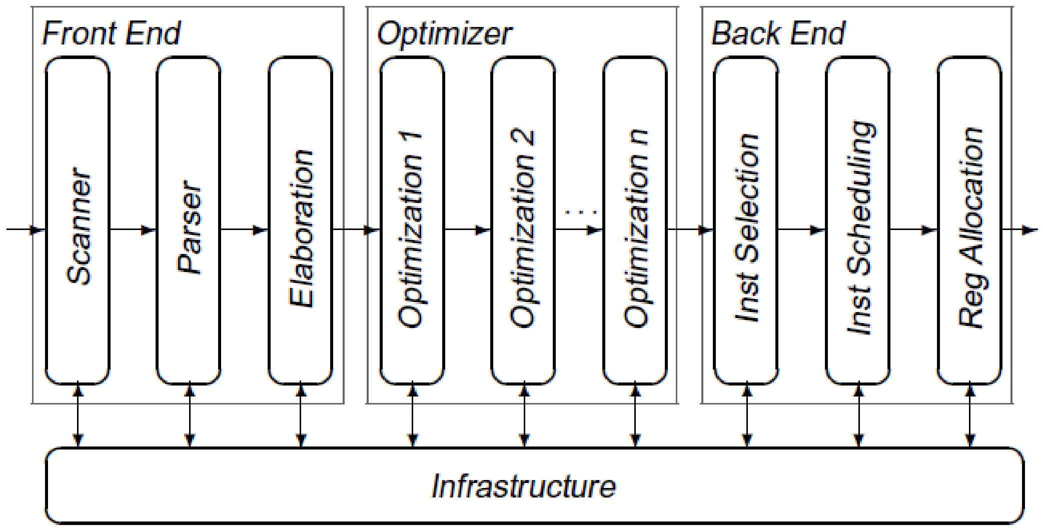
CourseCode:**CSE402**

CourseName:**Compiler Engineering**

Duration:90minutes MaxMarks:50

PART A: Answer all the questions 10 x 2 = 20 Marks

1. Structure of a typical compiler.



2. Table driven scanner use a general code or an algorithm to handle different DFAs and Res. Direct code scanners implement a specific code for each state in a DFA and the code in these DFAs differ from an RE to another.

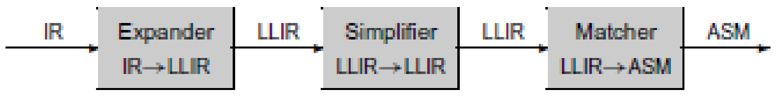
3. Top-down local allocation keeps heavily used virtual registers in physical registers. It dedicates a physical register to one virtual register for the entire basic block. Thus, a value that sees heavy use in the first half of the block and no use in the second half of the block effectively wastes that register through the second half of the block.

4. Bottom-up local allocator algorithm sequence this given string as store x2, load x3 and load x2.

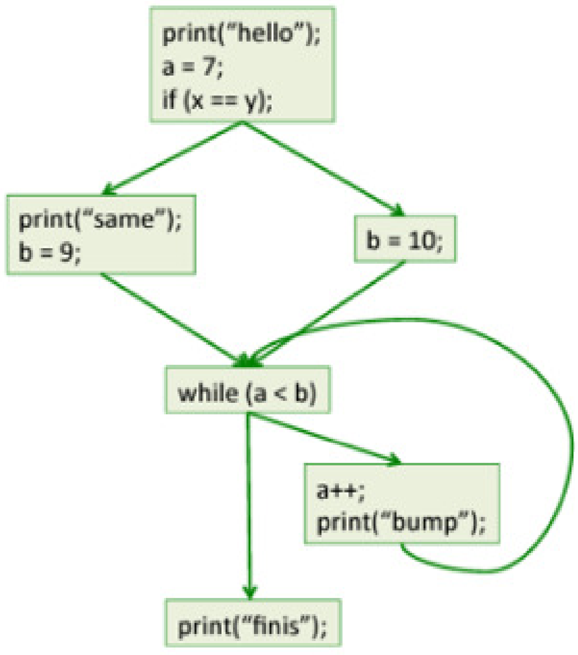
5. Compensation code: Code inserted into a block Bi to counteract the effects of cross-block code motion along a path that does not include Bi.

6. Cloning for context converts the graph into single EBB. Thus, it schedules the hot paths first, and scheduled path is used as prefix for critical paths.

7. Peephole optimization:



8. CFG:



9. The compiler inserts ϕ -functions at points where different control-flow paths merge and it then renames variables to make the single-assignment property hold.

Eg: loop: x_1<- ϕ(x_0, x_2)

10. a. Unrolling only inner loop. b. Unrolling outer loop and fuse inner loops.

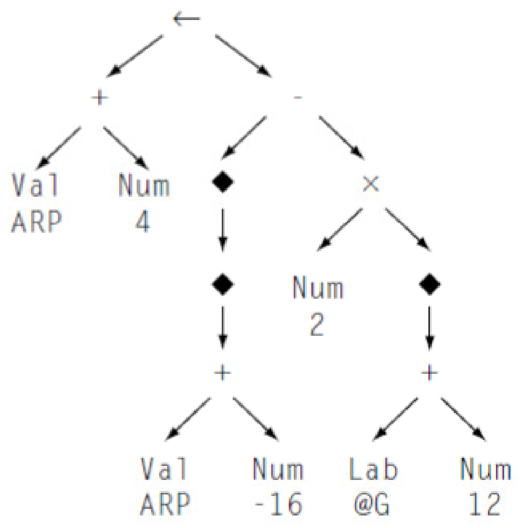
PART B: Answer any two of the following questions 2 x 10 = 20 Marks

11. Algorithm:

```
Cycle ← 1
Ready ← leaves of $\mathcal{D}$
Active ← ∅
while (Ready ∪ Active ≠ ∅)
    for each op ∈ Active
        if S(op) + delay(op) < Cycle then
            remove op from Active
            for each successor s of op in $\mathcal{D}$
                if s is ready
                    then add s to Ready
    if Ready ≠ ∅ then
        remove an op from Ready
        S(op) ← Cycle
        add op to Active
    Cycle ← Cycle + 1
```

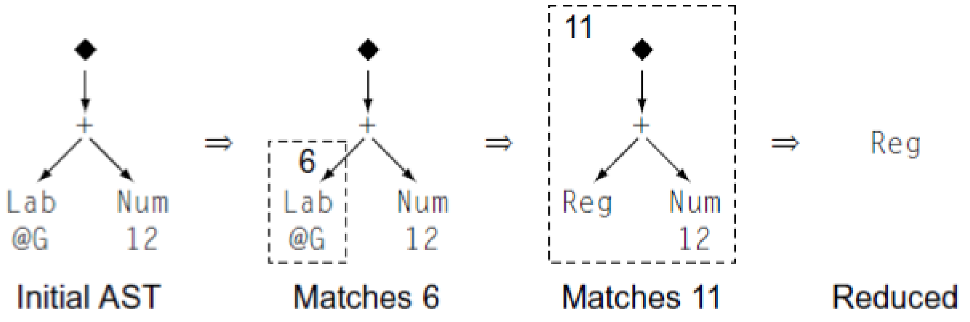
Tie-breaking priority:
1. Number of immediate successors it has in D
2. Total number of descendants it has in D.
3. Rank equal to its delay
4. Rank equal to the number of operands for which this operation is the last use

12. AST:

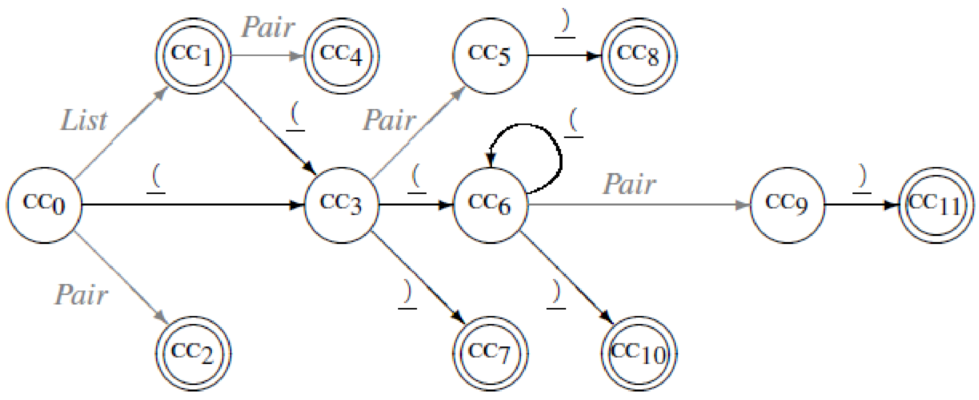


Low-Level AST for a ←b - 2 × c.

Reducing the subtree that references the variable ‘c’:



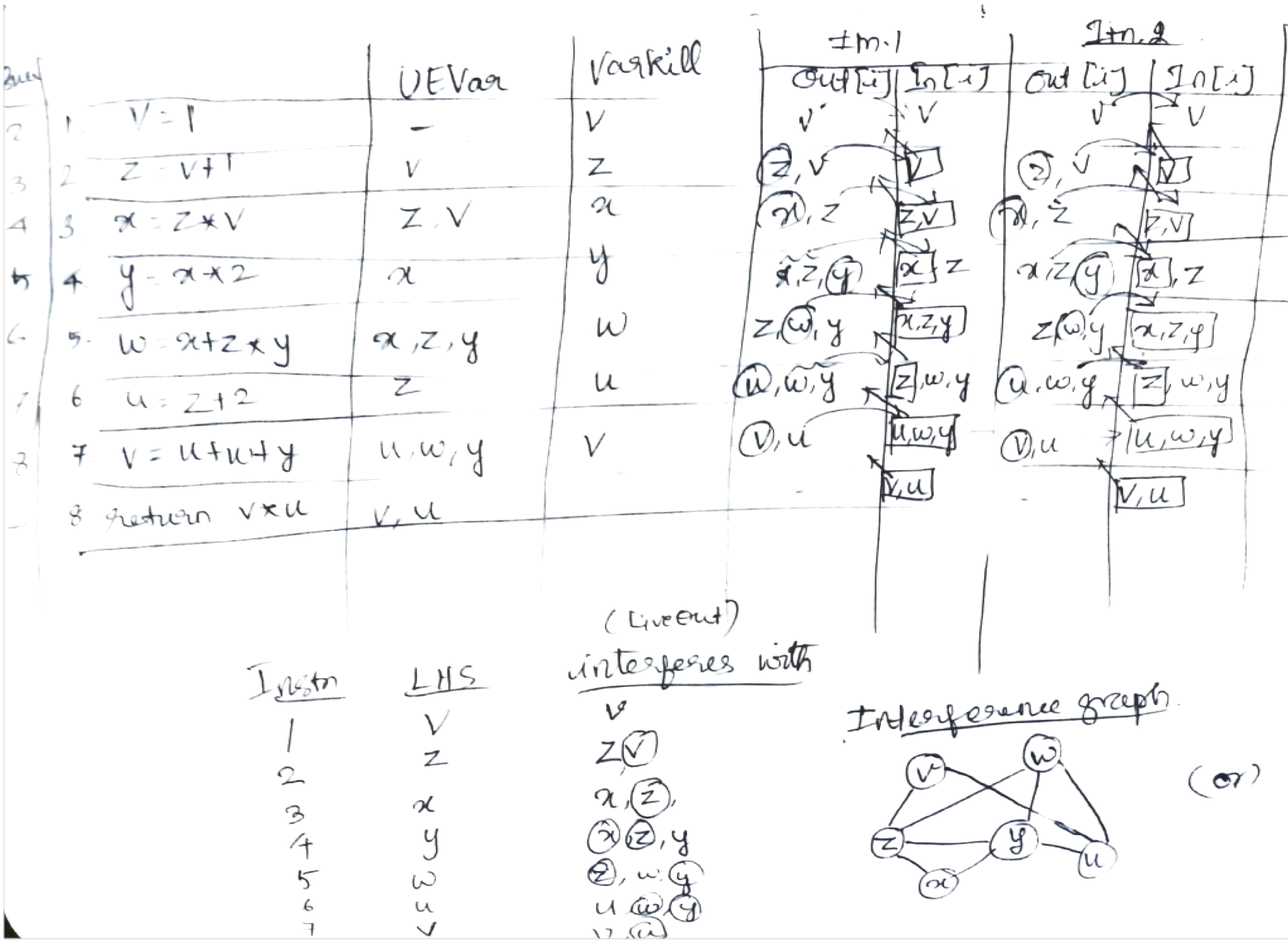
13. Canonical construction algorithm for parenthesis grammar.



PART C: Answer the following question 1 x 10 = 10 Marks

14. Bottom-up graph coloring register allocation:

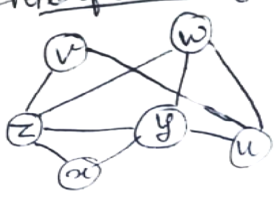
Procedure: Discover live ranges, build an interference graph, attempt to color it, and generate spill code when needed. Push nodes into stack by checking the degree of the nodes until there exists a node in Graph.



			UEVar	VarKill	Itn.1 Out[i]	Itn.1 In[i]	Itn.2 Out[i]	Itn.2 In[i]
2	1.	v = 1	–	v	v	v	v	v
3	2	z = v+1	v	z	z, v	$\boxed{v}$	z, v	$\boxed{v}$
4	3	x = z*v	z, v	x	x, z	$\boxed{z, v}$	x, z	$\boxed{z, v}$
5	4.	y = x*2	x	y	x, z, y	$\boxed{x}$, z	x, z, y	$\boxed{x}$, z
6	5.	w = x+z*y	x, z, y	w	z, w, y	$\boxed{x, z, y}$	z, w, y	$\boxed{x, z, y}$
7	6	u = z+2	z	u	u, w, y	$\boxed{z}$, w, y	u, w, y	$\boxed{z}$, w, y
8	7.	v = u+u+y	u, w, y	v	v, u	$\boxed{u, w, y}$	v, u	$\boxed{u, w, y}$
	8	return v*u	v, u		-	$\boxed{v, u}$	-	$\boxed{v, u}$

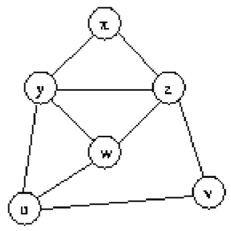
Instn	LHS	(LiveOut) interferes with
1	v	v
	z	z, v
2		
3	x	x, z,
4	y	x, z, y
5	w	z, w, y
6	u	u, w, y
7	v	v, u

Interference graph.

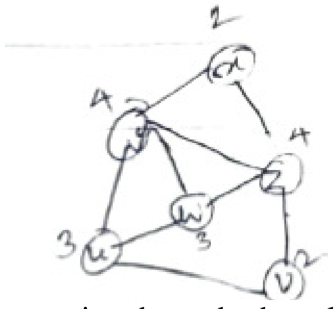


(or)

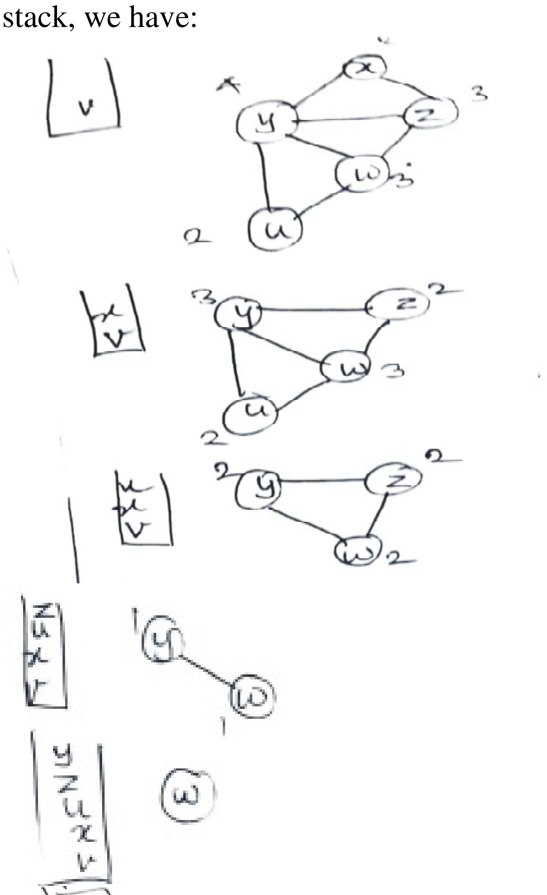
Interference graph:

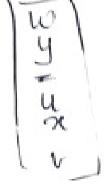


Annotating the Interference Graph with the degree of the nodes,



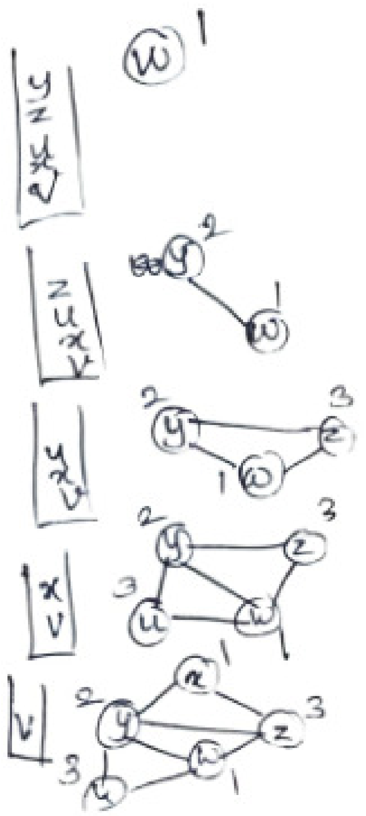
removing the nodes by selecting the node with least degree and pushing the node into the stack, we have:

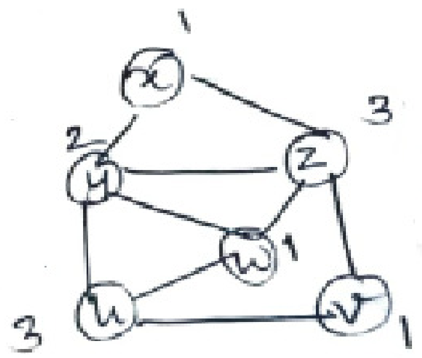




G_I is empty

Pop nodes from stack until empty and color for the nodes.





The given problem is 3-color graph. Code can be allocated in a three register machine. No need for spill code.

****