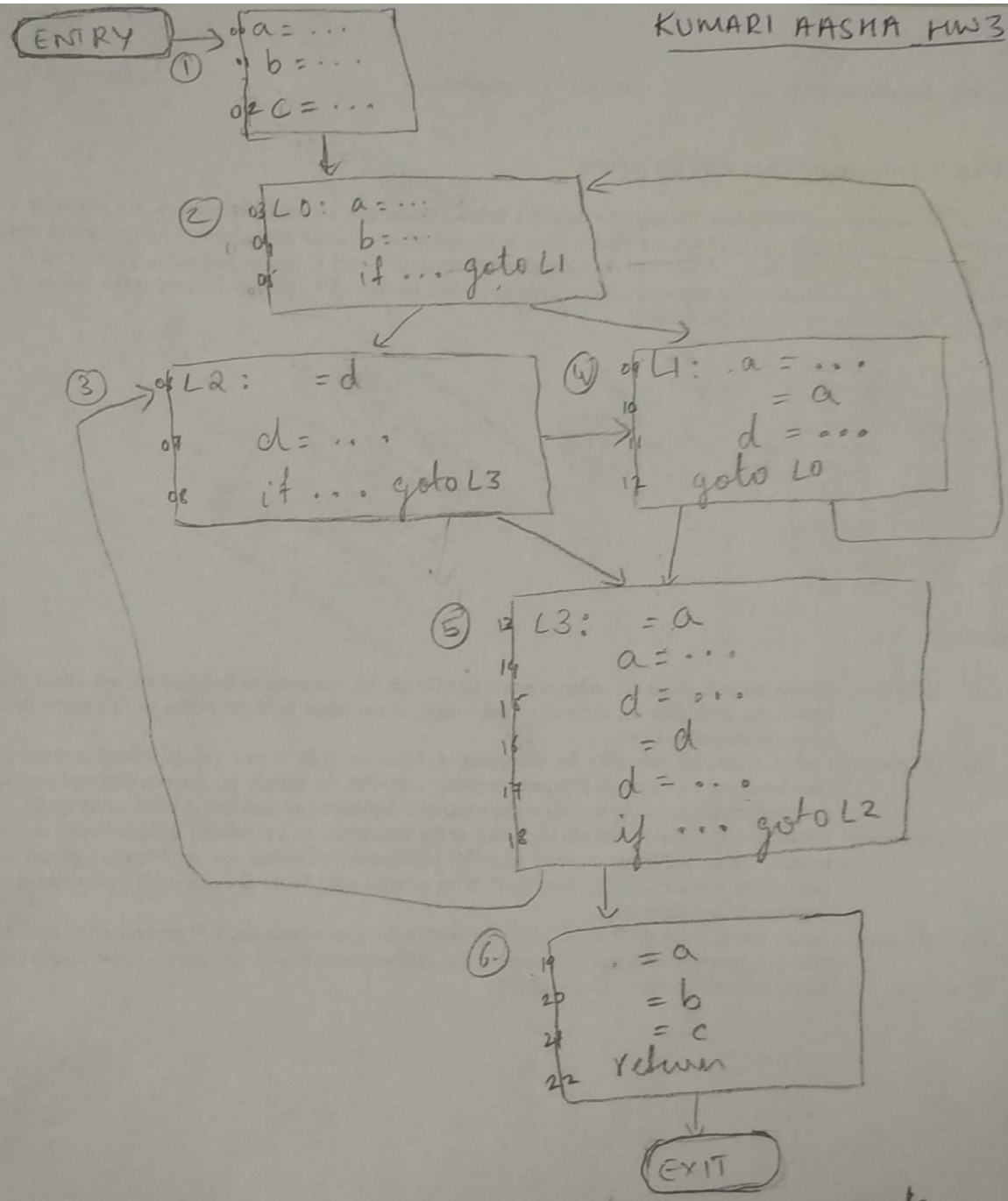
KUMARI AASHA HW3.

Q1 a)

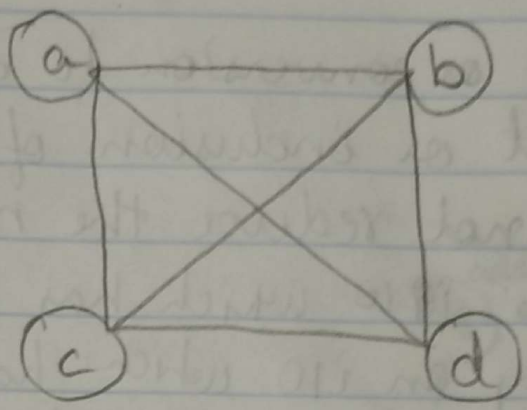


b) & c)

Temp Variable	Live ranges	frequency counts
a	03-08, 09-13, 14-19	7
b	04-20	4
c	02-21	1
d	03-06, 11, 15-18	6

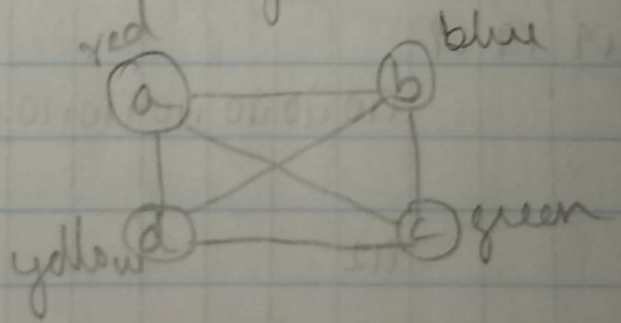
Assuming 3 registers we will store a, d, b as they have more frequent usage

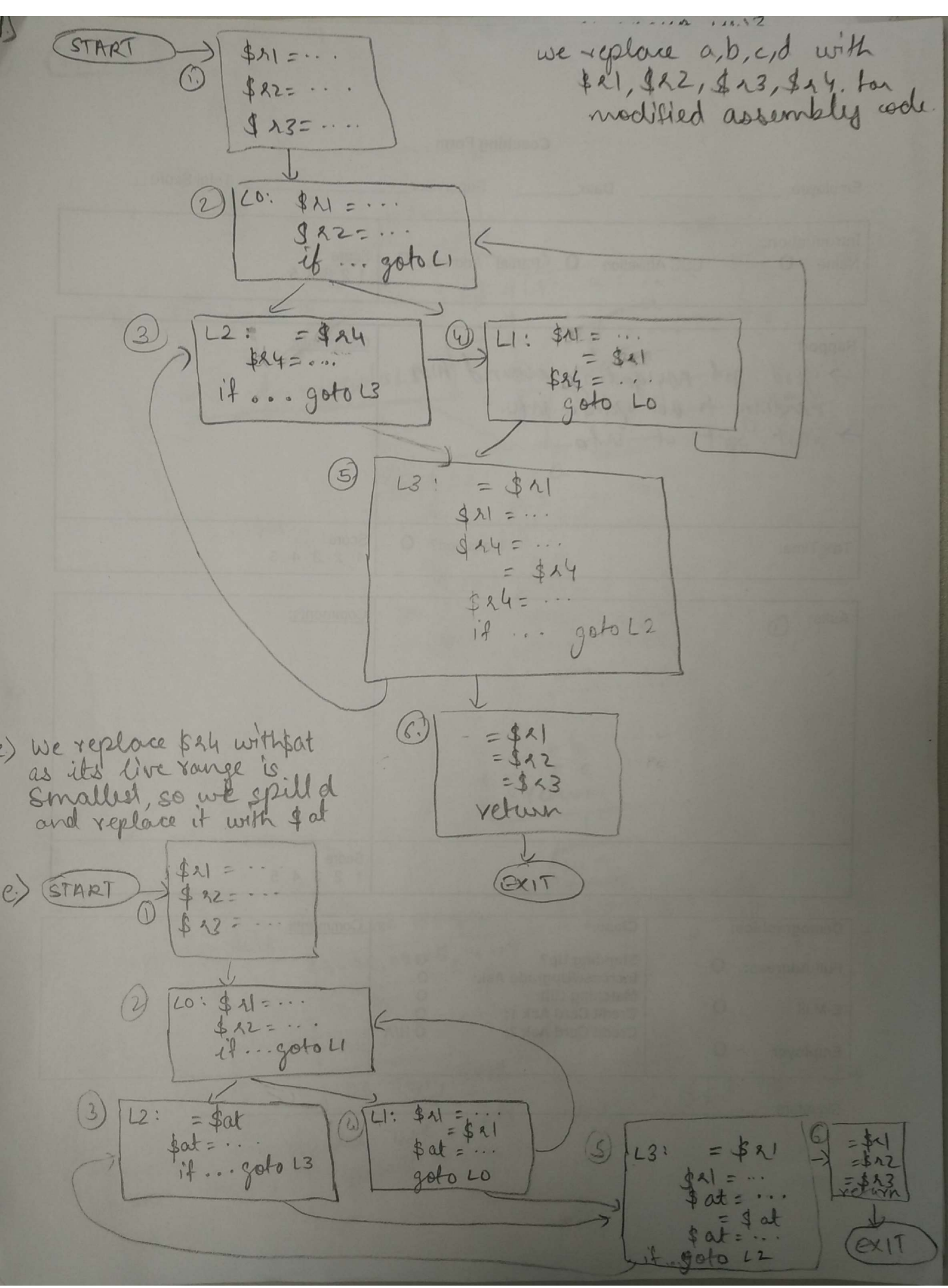
d)



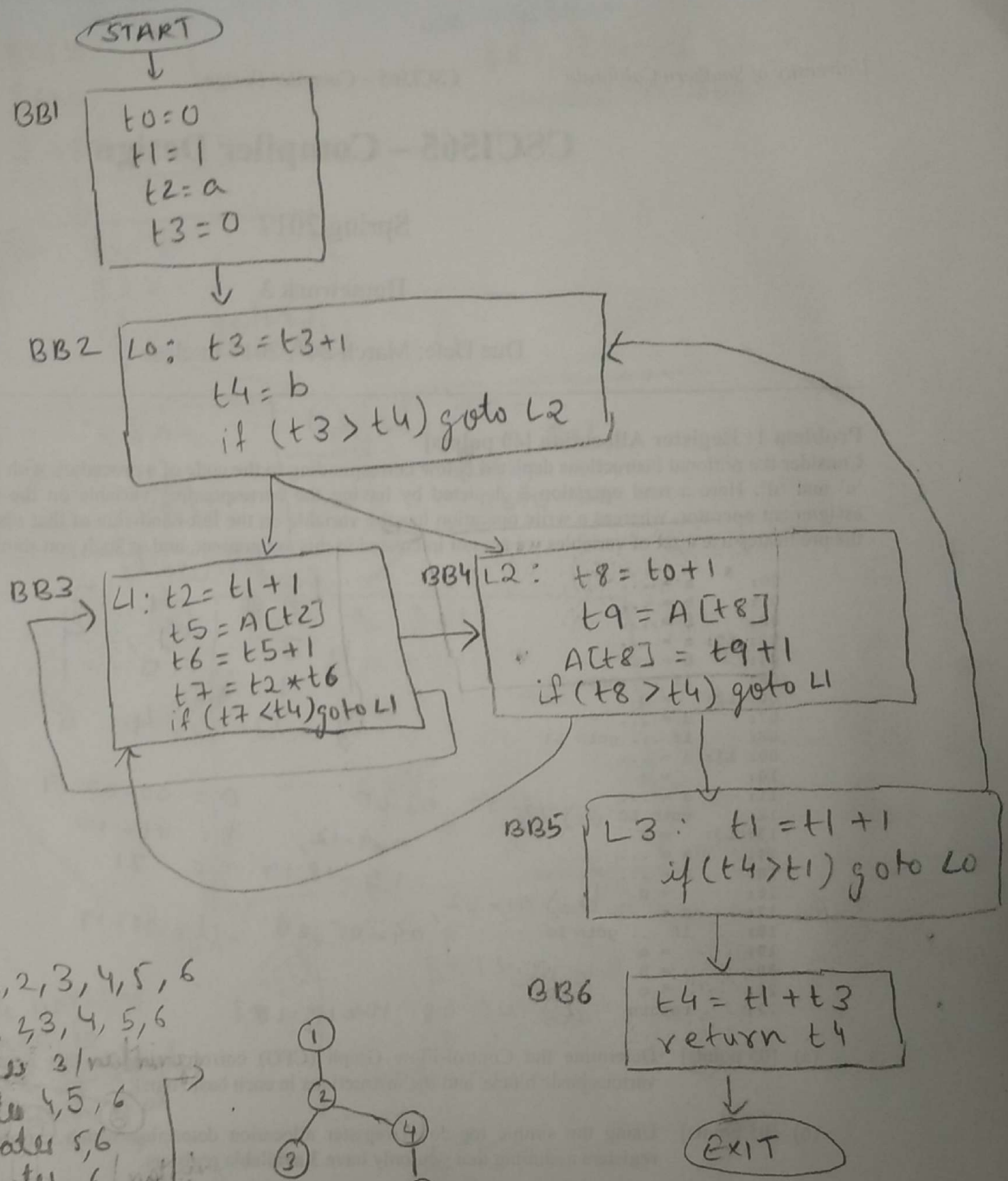
So, if we have 3 registers we need to colour the graph with 3 colours which is not possible as we have clique of size 4

However we can colour the graph using 4 colours ie 4 registers are required





Q2.)
a.)

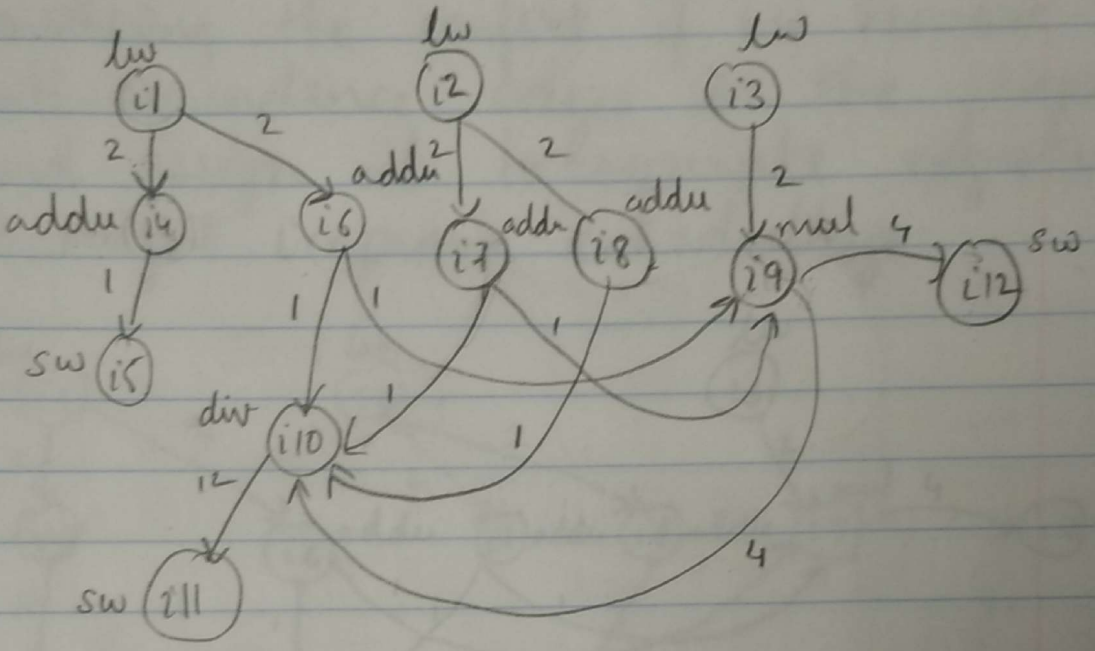


b.) 1 dominates 1, 2, 3, 4, 5, 6
2 dominates 2, 3, 4, 5, 6
3 dominates 3
4 dominates 4, 5, 6
5 dominates 5, 6
6 dominates 6
Dominator tree :

c) Back Edge is from 5 → 2 as 2 dominates 5
Natural loops are → 2 → 4 → 5 → 2 as it includes backedge 5 → 2

DAG :-

Q3.) a.)



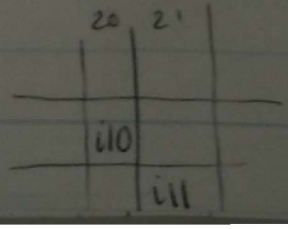
b.)

ALU			i4	i7
ALU				i6
L/S	i1	i2	i3	i5
		i1	i2	i3

b.)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
ALU			i4	i7	i9	i9	i9	i9												
ALU			i6	i8					i10	i10	i10	i10	i10	i10	i10	i10	i10	i10	i10	
L/S	i1	i2	i3	i5					i12											
		i1	i2	i3																

So, considering the DAG above it takes
21 clock cycles



c) Additional ALU or conversion to a pipelined arithmetic unit or inclusion of an additional memory will not reduce the number of clock cycles as i9 i10 which has 12 clock cycles has dependency on i10 which has dependency on statements which indirectly depends on i1, i2, i3, Thus 4 clock cycles for statements that depend on i9 must be there.

So $4 + \underset{(i9)}{4} + \underset{i10}{12} + \underset{i11 \Rightarrow store}{1} = 21$ will always be there.

Trying one more case considering additional ALU:

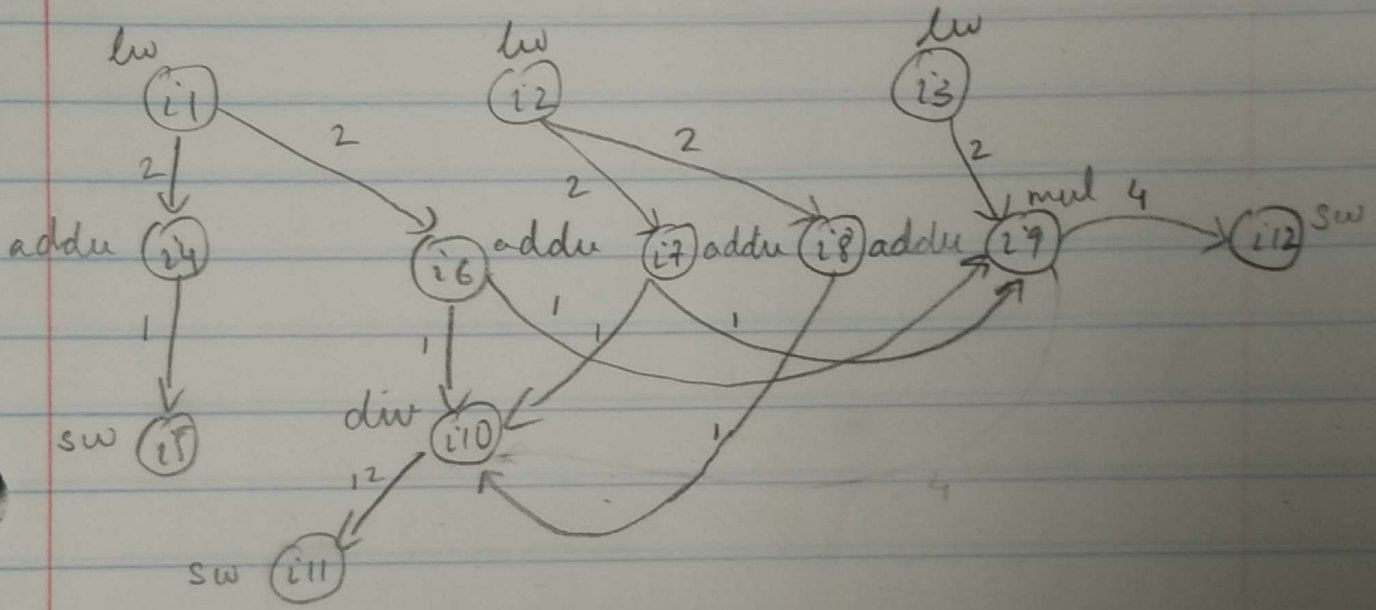
	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
ALU		i4		i9	i9	i9	i9												
ALU		i6	i7					i10	i10	i10	i10	i10	i10	i10	i10	i10	i10	i10	
ALU			i8																
LS	i1	i2	i3					i12											i11
		i1	i2	i3															

Same 21 clock cycles.

Considering pipelined arithmetic units :- (Same 21 cycles)

ALU		i4	i8				i10	i10	i10	i10	i10	i10	i10	i10	i10	i10	i10	i10	
			i7																
ALU		i6		i9	i9	i9	i9												
LS	i1	i2	i3					i12											i11
		i1	i2	i3															

d) Considering the conflict if we remove the anti-dependence edges in the graph and assign it to separate registers it might improve scheduling.



Anti ⇒ read → write So dependency of i9 before i10 is removed as it is anti-dependency. But it also has read after write dependency for $t5 so it cannot be removed. Hence there is no advantage of using additional registers to improve scheduling.