1)

Int x (4 bytes)	main's activation record
double y (8 bytes)	
float f (8 bytes)	
saved registers (4*8 bytes)	
return value for foo (8 bytes)	
argument a(y+f) for foo (4 bytes)	
main's return address (0X10001A1B - 8 bytes)	
main's old frame pointer (8 bytes)	foo's activation record
double g (8 bytes)	
saved registers (4*8 bytes)	
return value for bar (8 bytes)	
argument a for bar (4 bytes)	
argument r (fname) for bar (8 bytes)	
foo's return address (0X1002A2B - 8 bytes)	
foo's old frame pointer	bar's activation record
float h (8 bytes)	
double q (8 bytes)	

2)

	Instruction	Line	Registers			Code
			r1	r2	r3	
1	A = B+C	A, B	B	A*		ld B r1 ld C r2 add r1 r2 r2
2	C = A+B	A, B, C	B	A*	C*	add r2 r1 r3
3	T1 = B+C	A, B, C, T1	T1*	A*	C*	add r1 r3 r1
4	T2 = T1+C	A, B, C, T2	T2*	A*	C*	add r1 r3 r1
5	D = T2	A, B, C, D	D*	A*	C*	
6	E = A+B	C, D, E	D*	E*		st r3 C ld B r3 add r2 r3 r2
7	B = E+D	B, C, D	D*	B*		add r2 r1 r2
8	A = C+D	A, B	A*	B*		ld C r3 add r3 r1 r1
9	T3 = A+B	T3	T3*			add r1 r2 r1
10	WRITE(T3)					write r1

3) i)

Reservation Tables

ADD

ALU1	ALU2	LD/ST
	✓	

ALU1	ALU2	LD/ST
✓		

ADD's take either ALU1/ALU2 & occupy that ALU for a single cycle.

MUL

ALU1	ALU2	LD/ST
✓		

MLU's can execute on one of the ALUs only (Eg: ALU1). Then they take up 2 cycles to complete

LD

ALU1	ALU2	LD/ST
✓		
		✓
		✓

ALU1	ALU2	LD/ST
	✓	
		✓
		✓

LDs take up either of the ALUs for one cycle & LD/ST unit for 2 cycles.

ST

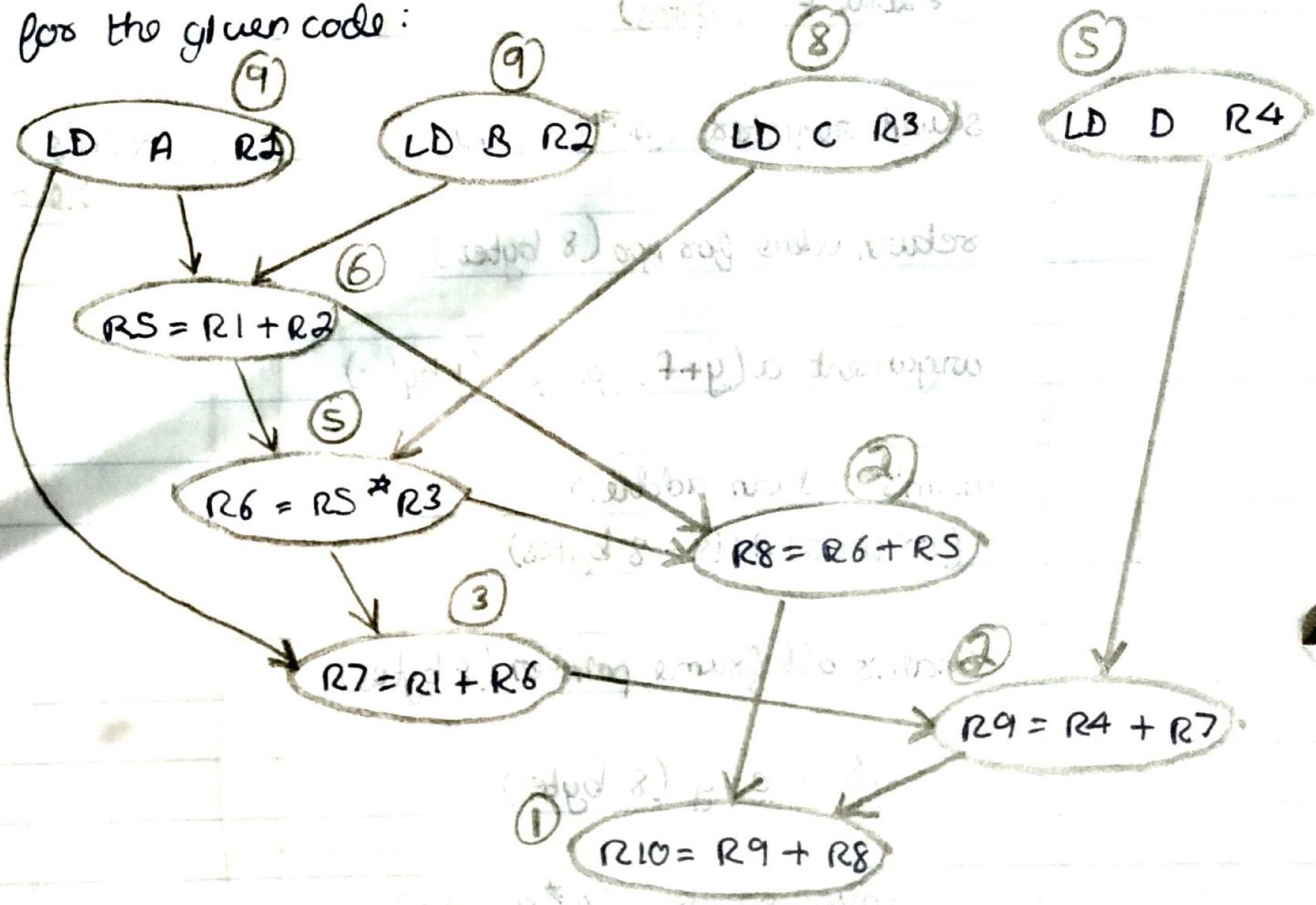
ALU1	ALU2	LD/ST
✓		
		✓

ALU1	ALU2	LD/ST
	✓	
		✓

ST's take up either of the ALU's for 1 cycle & LD/ST unit for 1 cycle.

Scanned by TapScanner

3) ii) DAG for the given code :



Heights assigned to the nodes =>

a) height of leaf node = latency of that Instruction

b) height of an interior node = maximum of heights of all children + latency of that instruction

3) iii)

Cycle No.	Available Instructions	Scheduled Instructions	Completed Instructions	ALU1	ALU2	LD/ST
0	1, 2, 3, 4	1	–	1		
1	2, 3, 4	–	–			1
2	2, 3, 4	2	–	2		1
3	3, 4	–	1			2
4	3, 4	3	–	3		2
5	4, 5	5	2	5		3
6	4	4	5	4		3
7	6	6	3	6		4
8						4
9	7, 8	7, 8	4, 6	7	8	
10	9,	9.	7, 8	9		
11	10	10	9	10		
12			10			