ASSIGNMENT NO: 7.

Title: Instruction Scheduling Algorithm.

Problem Statement:
Implementation of Instruction Scheduling Algorithm.

Learning Objectives:
1] To understand various instruction scheduling algorithms.
2] To implement the above algorithm.

Learning Outcome:
1) Students will be able to implement instruction scheduling algorithm.
2) Students will implement instruction scheduling Algorithm.

H/w & s/w Requirements:
- Eclipse IDE.
- JDK 1.8.
- Dell Desktop System.

Theory:
Instruction Scheduling is a composite optimization used to improve instruction level parallelism which improves performance on machines with instruction pipelines.

It tries to do the following without changing the meani of code
1) Avoid pipline stalls by rearranging the order of instructions.

2) Avoid illegal semantically ambiguous grammar.

Data Hazards:-

There are 3 types of dependencies data hazards:-

1) Read after write - Instruction reads the value that is written by another instruction.

2) Write after read - Instruction reads a location i.e. later overwritten by instruction 2.

3) Write after write - Both instruction write at the same location.

Algorithm:-

- The simplest algorithm to find topological sort is known as list scheduling.

- It repeatedly selects the source of the dependency graph, appends it to the current instruction, schedule & remove it from the graph.

- This may cause other vertices to be sources which will then also be sources and will be considered for scheduling.

- The algorithm terminates if the graph is empty.

- To arrive at a good schedule, stells should be

prevented, this is determined by choice of next instruction.

List Scheduling :

Basic Idea :

Make a ordered list of processes by assigning them same priorites and then repeatedly execute the full steps, until a valid schedule is obtained.

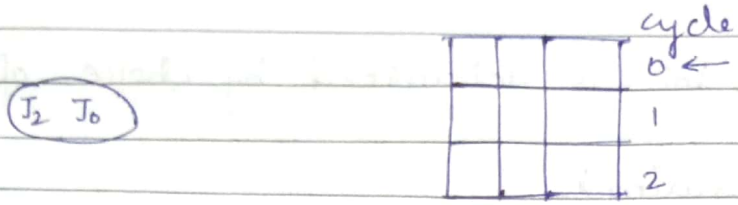
1) Select from the list, the process with the highest priority for scheduling.
2) Select a resource to accomodate the process.
3) If no resource can be found, we select the next process in the list.

- The priorities are determined statistically. The first step is to choose the process with highest priority and second step is to select the best possible resource.

Maintain the list of instruction that are ready to execute.

- data dependant constraints should be preserved.
- machine resources are available.

Moving cycle by cyle through the scheduling template: choose instruction from the list for the next cycle.

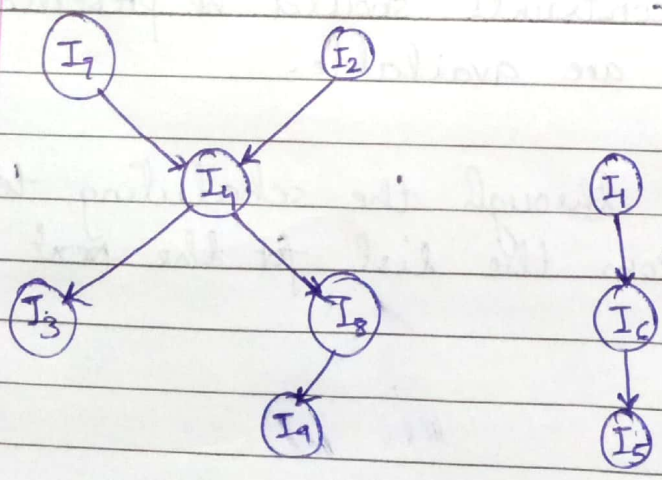


Algorithm:

```
cycle :- 0
ready list = root node in DPr;
{while( !ready_list && an issue slot is available)
   for op = (all nodes in ready list in ↓ priority)
     {
       add op to schedule at time cycle
       if (op has an outgoing antiedge)
            add all targets or op that are ready.
                 to the readylist)
     }
}

cycle = cycle + 1;
end
```

Input: DPG



Node	priority
0	5
2	3
1	2
4	5
6	5
3	2
8	1
5	2
9	4

output Scheduled

Ready list at Clock 1 : 0, 2, 1
$I_1 | I_2$

Ready list at Clock 2 : 0, 4, 6
$I_0 | I_4$

Ready list at Clock 3 : 0, 3, 8, 7
$I_8 | I_3$

Ready list at Clock 4 : 5, 6, 9
$I_5 | I_9$.

Final Scheduled Code Cycle:

		Cycle.
I_1	I_2	0
I_0	I_4	1
I_8	I_3	2
I_5	I_4	3
I_6		4 .

Conclusion:

Hence, we implemented list scheduling algorithm for instruction scheduling for a given data precedence graph, number of functional units & priority of nodes