**EXERCISES ON INSTRUCTION SCHEDULING**

1. a) Explain how list scheduling can be used to generate a schedule for the following basic block and show the schedule. First, assume a processor that can issue up to three instructions per cycle, where at most one instruction is a `load`, at most one instruction is a `mult` and at most one instruction is an `add` or a `mov`. All instructions have a latency of one cycle.

```
1. load r1, @x
2. load r2, @y
3. load r3, @z
4. mov  r4, 7
5. add  r5, r4, r1
6. mult r7, r1, r4
7. mult r8, r2, r5
8. add  r9, r8, r3
```

Then, show the schedule for another processor that can issue up to two instructions per cycle (any instructions). Compare the two schedules you derived.

(8 marks)

2.

d) Apply list scheduling to generate a schedule for the following basic block on a processor that can issue up to two instructions per cycle. Your answer should show the precedence graph, explain how you set priorities and show the schedule. Assume that all instructions have a latency of 1 cycle except `mult` that has a latency of 2 cycles.

```
1. mult r1, r1, 7
2. mov  r2, 1
3. mult r3, r3, 12
4. add  r4, r1, r2
5. add  r5, r2, 4
6. add  r6, r2, r3
7. add  r7, r5, 6
8. add  r8, r5, 9
```

(5 marks)

e) A critical aspect of list scheduling is the mechanism for setting priorities and breaking ties when several operations with the same priority are ready at the same cycle. Using as an example the schedule generated in d) above, demonstrate how different mechanisms for setting priorities and breaking ties can lead to different schedules.

(3 marks)

3.

b) (i) Describe an efficient list scheduling algorithm for instruction scheduling on a hypothetical system with two functional units, if the cost of executing an instruction (delay latency of the instruction) on the second functional unit is one cycle more than the cost of executing the same instruction on the first functional unit.

(6 marks)

(ii) Apply your algorithm above to the basic block below and show the schedule.

```
1. load r1, @x
2. load r2, [r1+4]
3. add r3, r3, 1
4. mult r6, r6, 17
5. store r6
6. div r5, r5, 29
7. add r4, r2, r5
8. mult r5, r2, r4
9. store r4
```

You should assume that, on the fastest functional unit, the cost of a load and a store is 3 cycles, the cost of a mult is 2 cycles and the cost of all remaining operations is 1 cycle.

(6 marks)