Lecture 26-27: October 20, 2021

Computer Architecture and Organization-II

Biplab K Sikdar

ILP -III

Eager execution is a form of speculative execution (Figure 24).

Both sides of conditional branch are executed.

However, results are committed depending on fate of branch condition.

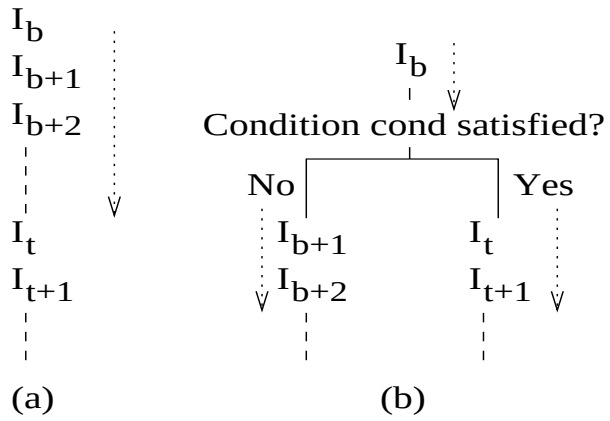


Figure 24: Eager execution

Data value speculation adopts the following prediction schemes.

1. Last value predictor (LVP)

Consists of a single prediction table indexed by hashed PC (Figure 25).

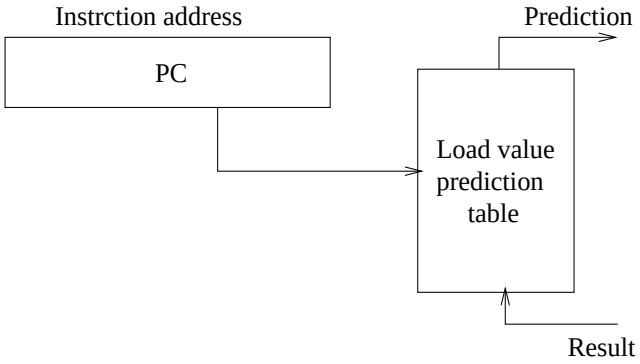


Figure 25: Last value predictor

It predicts: result to be same as what was computed during previous execution.

2. Stride value predictor (SVP)

Stride $\Rightarrow$ difference (d) between two most recent values produced by instruction.

If val_1 and val_2 are two recent values for variable/data x and d = val_2 - val_1,

Then predicted value for x is val_2 + d.

Example 0.4 *Let sequence of values loaded to x in different time steps is*

2, 3, 4, 1, 2, 3, 1, 2, 3, 4.

Number of mis-speculations in stride value prediction for x is .....

Consider speculation starts after second time step.

3. Context-based value predictor (CVP)

Context $\Rightarrow$ sequence of n most recent values produced by instruction for x.

It is classified according to the order.

An n^{th} order model takes last n values and uses this for prediction.

The 0^{th} order predictor is, therefore, purely frequency based.

Example 0.5 *Let sequence of values for a variable x is a a b b c a a a b b c a a a*

Next predicted value for x, in 0^{th}/1^{st}/2^{nd}/3^{rd} order predictor is shown in Figure 26.

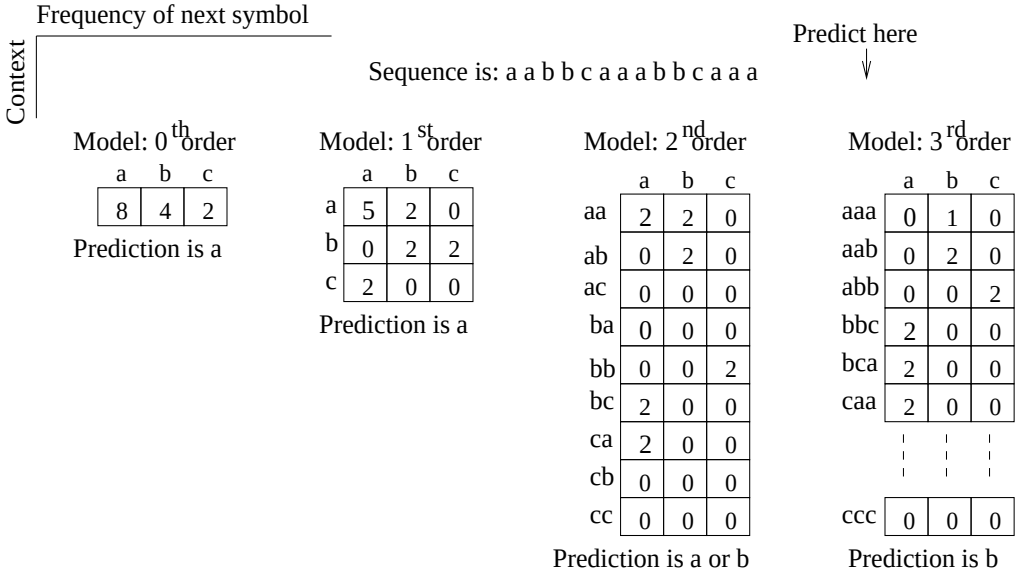


Figure 26: Context based predictor

4. Return value predictor (RVP):

SKIP

0.6.3 Trace Scheduling

A *trace* is a sequence of instructions which may include branches.

This does not include loop branches.

It consists of one or more basic blocks - can have more than one entry/exit point.

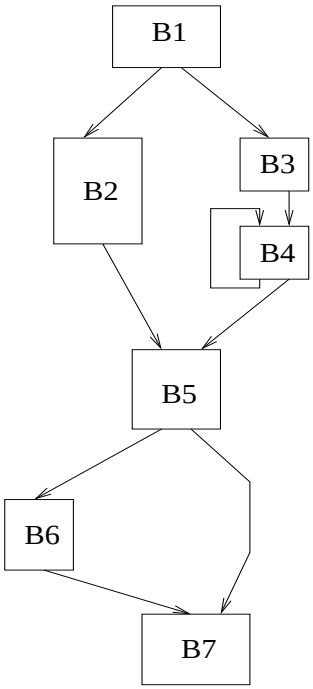


Figure 27: Traces

Traces in the control flow graph of Figure 27 are

B1, B3;

B1,B2;

B4;

B5, B7

B5, B6, B7 and

B1, B2, B5, B6, B7.

Trace scheduling

Compiler can find parallelism across conditional branches through trace scheduling.

Trace scheduling is effective for a superscalar processor with very high issue rate.

Basic idea is to increase ILP along the important execution path.

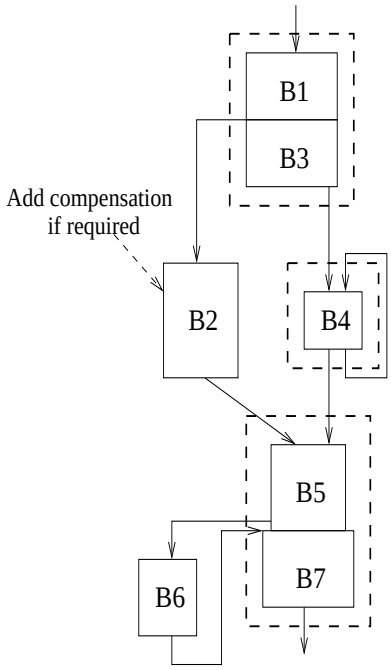


Figure 28: Traces selected

Let B1, B3, B4, B5, B7 is most frequently executed path in Figure 28.

Therefore, the traces are:

B1, B3

B4

B5, B7.

Trace B1, B3 has 1 entry and 2 exits.

Trace B5, B7 has 3 entries and 2 exits.

Follows simple code motion - change order of instruction - preserves dependencies.

The *compensation codes* are needed for side entry points and side exit points.

Compensation codes are difficult to generate specially for entry points.

Example of compensation code is shown in Figure 29.

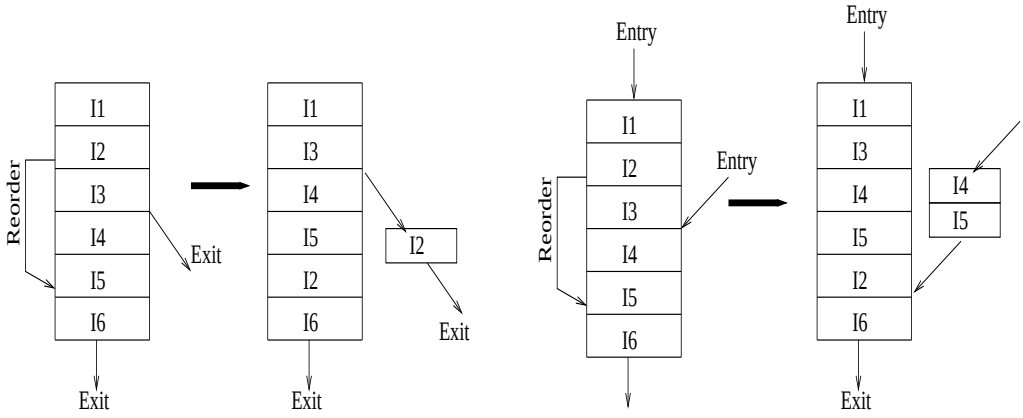


Figure 29: Trace compensation

Let B1 is I1, I2, I3 and B3 is I4, I5, and I6 of Figure 28.

That is, Figure 29(a) is the trace B1, B3. Side exit shows exit after B1.

Case 1*:* Moving an instruction (I2) below a side exit (after I5) -

Shown in Figure 29(a) with no violation of dependencies.

But outcome at side exit path should be I1, I2, I3 as per original program.

After code motion it is I1, I3. Therefore, compensation is required.

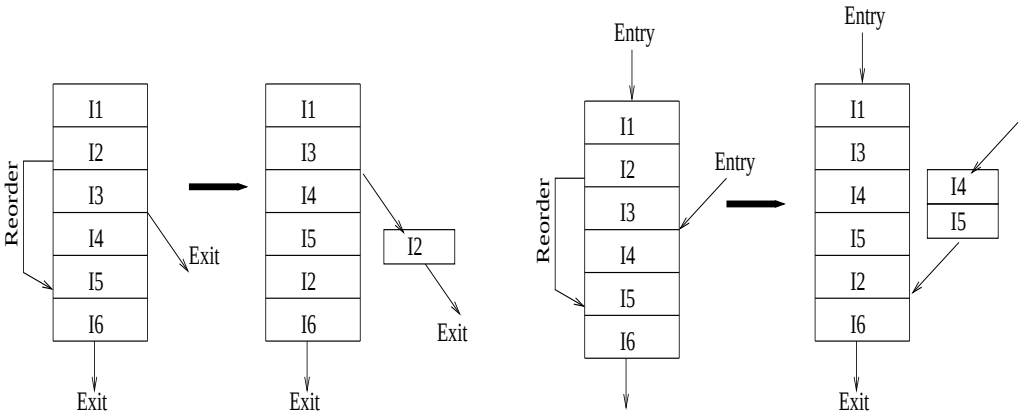
After compensation it is I1, I3, I2.

In original program it is I1, I2, I3 and after code motion it is I1, I3, I2

(no issue, as any order of execution of I2/I3 in this case is acceptable).

Case 2*:* Moving an instruction above a side exit (speculative execution)

- nothing to do at the side exit CHECK.



a) Compensation: moving an instruction below the side exit

b) Compensation: moving an instruction below the side entry

Case 3, 4*:* Are related to side entry - shown in Figure 29(b).

Let original instruction order is I1, I2, I3, I4, I5, I6.

After code motion, below side entry, order is I1, I3, I4, I5, I2, I6.

As per original order, side entry should be after I1, I2, I3

(this is maintained, side entry is shifted below I2).

Further, as per original order, after side entry I4, I5, I6 should be executed.

This requires compensation I4, I5 at side entry.

CHECK compensation for code motion above side entry.

Exercise

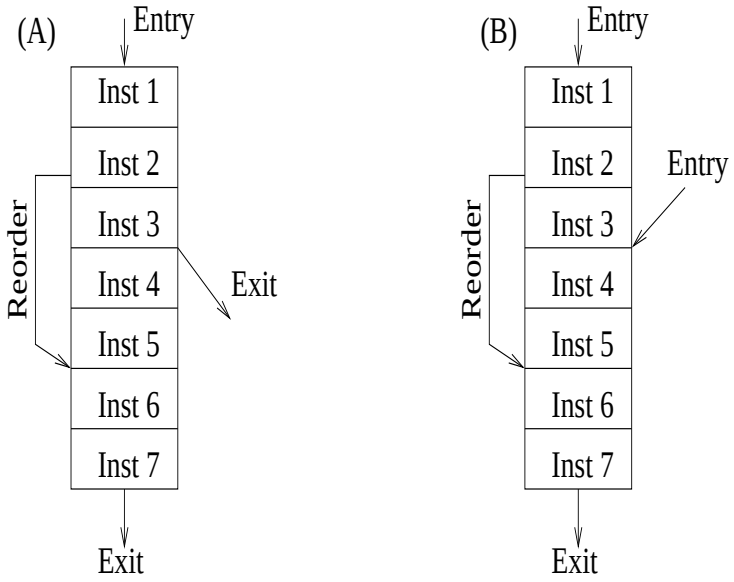


Figure 30: Code motion

Figure 30(A)

The original order is I1, I2, I3, I4, I5, I6, I7.

Order after code motion I1, I3, I4, I5, I2, I6, I7.

Compensation at side exit is .....

Figure 30(B)

The original order is I1, I2, I3, I4, I5, I6, I7.

Order after code motion I1, I3, I4, I5, I2, I6, I7.

Compensation at side entry is .....

QUIZ