ECEn 528

Study Guide ­ ILP Limits

● Read Section 3.10 of H&P

 Things to focus on

■ Note in the middle of page 215 how the results in Figure 3.26 are explained using characteristics of the benchmarks. This is good practice

■ Section 3.10 is very important to understand

■ Figure 3.27 makes the situation look fairly bleak for ILP; it's actually worse because cache misses and non­unit latencies haven't been taken into account

 Clarifications

■ In Figure 3.26, what is being measured is the instructions per cycle (IPC) for the “perfect” processor.

■ Remember that when H&P say “issue”, we say “dispatch”

 Answer the following questions:

1. Why is the IPC for a perfect processor so low in Figure 3.26? What is the limiting factor?

True data dependences

1. Why does restricting the window size have such a large effect on programs such as tomcatv?

FP benchmarks are limited more by window size than by BP, register renaming, inherent parallelism, etc.

1. Why does imperfect branch prediction limit available ILP?

Imperfect BP requires instructions to stall or get squashed

1. How could compiler improvements relax some of the limits found in this limit study?

Automatic parallelization could reduce unnecessary dependences

● Read “Loose Loops Sink Chips,”  Things to focus on

■ You need read only sections 1 ­ 3

■ This idea of loops is a very powerful way about thinking about design decisions when trying to determine the number of stages

 Clarifications

■ You should refer to Figure 2 for the pipeline diagram while reading section 1, but Figure 3 while reading sections 2 and 3.

■ The dispatch step is called DEC­IQ and the issue step is called IQ­EX in this paper.

 Answer the following questions:

1. Why are loose loops a problem?

The pipeline must either be stalled or predict a value

1. Register renaming can be thought of as a scheme which changes a frequent loose loop to an infrequent one. In what way is this so?

Register renaming decreases the amount of WAW/WAR hazards

1. Why is pipelining dispatch less of an problem than pipelining issue?

Issue has to wait until all the operands have been written to the register file?

Pipelining is much harder for the issue

1. How does forwarding tighten a loose loop?

Forwarding enables instructions to get operands from the ALUs without waiting for them to be written to the register file; it makes the result computed in a previous cycle available in the current cycle