Lecture 4: Multiple issue

Hossam A. H. Fahmy

Cairo University, Faculty of Engineering

Overview

Out of the bottleneck

Let us look at the example of the fire and the well again.

- One person is at the well. There are two lines of people from the well location to the fire. The person at the well fills a bucket and hands it to the next person in one of the lines.
- Two persons are at the well each filling a bucket and then providing it to the lines.

Which one will put the fire down faster? Why?

The "issue rate" of CPI = IPC = 1 is called Flynn bottleneck.

ILP

Instruction Level Parallelsim (ILP) is a property of the software not the hardware. The hardware supports ILP by

- pipelining,
- superscalar in order execution such as in Sun UltraSparc, or
- superscalar out of order execution such as in Intel Pentium4.

The reordering (scheduling) may be dynamic at run time by the hardware or static at compile time by the software.

Going to multiple issue

We will look today at in order execution to solve its problems and detect any dependences. How can we issue two, four, or in general n instructions per cycle?

- Fetch n instructions per cycle,
- decode n instructions per cycle,
- execute n instructions per cycle,
- may access n locations in memory per cycle, and
- may write into n locations in the registers.

Wide fetch

We are not getting the instructions from the real memory but from an instruction cache.

- Instructions are sequential
  - Do they fall on the same 'line' in cache? Similar to the issue of aligned and non-aligned accesses to half-words in the memory.
- Instructions are *not* sequential
  - Two serial accesses? No! You will not know the target address and complete the second fetch within one clock cycle.

Solution to wide fetch

Problem: On a taken branch all the fetch slots after the branch are thrown away. ⇒ a low utilization of the fetch unit and eventually a low IPC.

Solution: *Trace cache*

- In addition to the regular cache, store the dynamic instruction sequence.
- Fetch from the trace cache but make sure that the branch directions are correct.
- If you miss get the correct instructions from the regular cache or even from the memory.

A trace cache is used in Pentium4.

Wide decode

Decode: The decoding of a number of instructions

- is easy if they are of fixed length and fixed formats
- but is harder (although possible) for variable length.

Read operands: We should check the dependencies and read the operands.

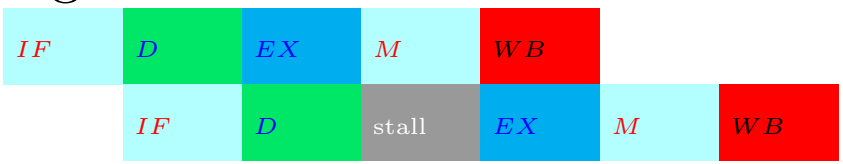
- With n instructions, we have *at most* $2n$ operands to read in one cycle. $\Rightarrow 2n$ read ports and the register file becomes proportionally slower.

Dependences for n instructions

Remember that we have to stall sometimes even with a complete bypassing network.

Ld R5, 0(R3)

Sub R6, R5, R1



We check

$$(s_1{}_{\text{Dec}} = D_{\text{Ex}})\&(op_{\text{Ex}} = \textbf{Ld}) \quad | \quad (s_2{}_{\text{Dec}} = D_{\text{Ex}})\&(op_{\text{Ex}} = \textbf{Ld})$$

With two instructions going in the decode, the number of checks quadruples and not just doubles! *n^2 growth in circuits for stall and bypass.*

$$\begin{aligned}
&(s_1{}_{\text{Dec1}} = D_{\text{Ex1}})\&(op_{\text{Ex1}} = \textbf{Ld}) \quad | \quad (s_2{}_{\text{Dec1}} = D_{\text{Ex1}})\&(op_{\text{Ex1}} = \textbf{Ld}) \\
&|(s_1{}_{\text{Dec1}} = D_{\text{Ex2}})\&(op_{\text{Ex2}} = \textbf{Ld}) \quad | \quad (s_2{}_{\text{Dec1}} = D_{\text{Ex2}})\&(op_{\text{Ex2}} = \textbf{Ld}) \\
&|(s_1{}_{\text{Dec2}} = D_{\text{Ex1}})\&(op_{\text{Ex1}} = \textbf{Ld}) \quad | \quad (s_2{}_{\text{Dec2}} = D_{\text{Ex1}})\&(op_{\text{Ex1}} = \textbf{Ld}) \\
&|(s_1{}_{\text{Dec2}} = D_{\text{Ex2}})\&(op_{\text{Ex2}} = \textbf{Ld}) \quad | \quad (s_2{}_{\text{Dec2}} = D_{\text{Ex2}})\&(op_{\text{Ex2}} = \textbf{Ld})
\end{aligned}$$

Wide execute

Shall we put n execution units?

- Yes for ALU.
- No for floating point division since it is big and used infrequently.

$\Rightarrow$ based on the instruction statistics, provide a mix of units.

RS/6000: 1 ALU/memory/branch + 1 FP

Pentium II: 1 ALU/FP + 1 ALU + 1 load + 1 store + 1 branch

Alpha 21164: 1 ALU/FP/branch + 2 ALU + 1 load/store

n^2 bypass

The bypass detection logic grows as n^2. This is acceptable since the sources and destinations are small fields (5 bits for 32 registers).
However, the bypass buses also grow as n^2. This is *not* acceptable. The busses are 32 or 64 bits wide each.

- It is difficult to layout and route all of these wires.
- Wide multi-input multiplexers are slow.

$\Rightarrow$ Group functional units into *clusters* and issue the dependent instructions to the same cluster.

Wide memory and write back

There is nothing too special about these two stages for wide issue. Their complexity just grows and they may become slower.

- Additional ports.
- Conflict detection logic for simultaneous multiple reads and writes to the same bank.

Summary

The are some problem spots for in order superscalar processors.

Fetch and branch prediction: may use trace cache.

Decode: the dependence checks grow as n^2.

Execution: Clustering may solve the n^2 bypass buses problem.