Exceptions and Conclusions

4-7-2003

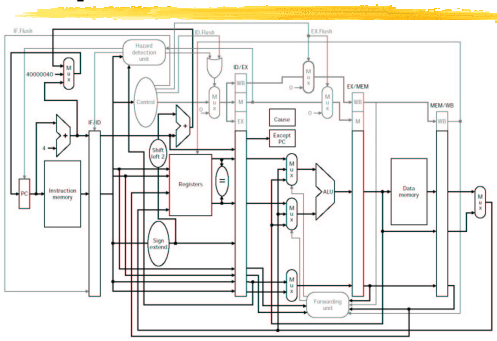
Opening Discussion

- What did we talk about last class?
- Have you seen anything interesting in the news?
- What are the implications of long pipelines on performance? In what ways do they benefit it? In what ways do they hurt it?

Review of Exceptions

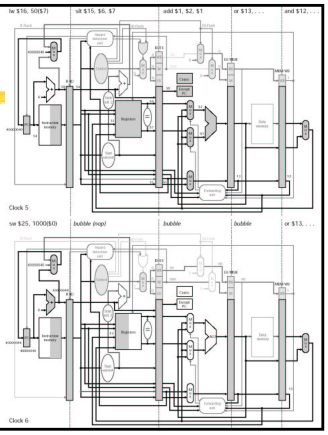
- Recall that the idea of an exception is that something happens in the code that we need to deal with before the code can continue to execute. The main example that we've seen is when there is an overflow in addition or subtraction.

Pipelined Datapath with Exceptions



What Happens?

- This shows us what happens when an exception occurs and we have a full pipeline.



Precise vs. Imprecise Exceptions

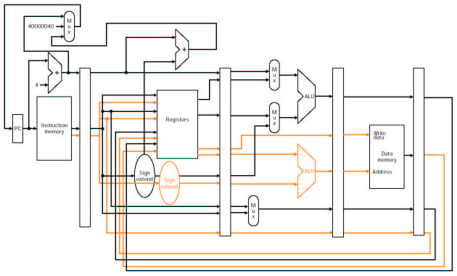
- Some architectures use imprecise exceptions where the EPC might not hold the address of the instruction that caused the exception, just something close to it. This helps with pipelines because the PC is being incremented all the time.
- The MIPS uses precise exceptions so there is no guess work involved.

Superscalar Pipelines

- Superscalar pipelines allow the processor to issue more than one instruction in a given clock cycle. The instructions must fit certain types though. For example, our MIPS architecture might be able to do one branch or ALU as well as one load or store in a single cycle.
- This requires some alteration to some of our basic components.

A Superscalar Pipeline

- Added ALU and a more complex reg. file.



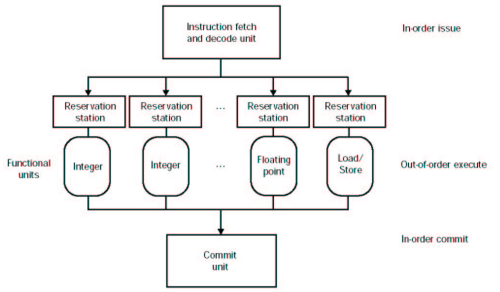
Loop Unrolling

- Loop unrolling is an old optimization technique. Superscalar hardware has made this even more significant if we are willing to use more temporary registers. The reason is that we can be loading data for the next array element while we do the calculations on the first element.
- This can bloat the code because we need different versions of the loop.

Dynamic Pipelining

- This allows other instructions to be executed when one instruction is stalled. Combining this with compilers intelligently ordering instructions can significantly reduce the number of instructions that are wasted or even used non-optimally.
- Speculative execution on branches if required here if we execute things before the branch that really determines if they execute.

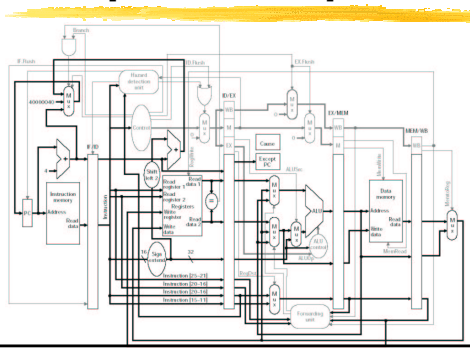
Diagram of Dynamic Pipeline Scheduling



Modern Touches

- Make sure to read sections 6.9-6.11 to get a bit of a feel for what types of additions you have in real modern processors and what is required to build them.

Final Pipelined Datapath



Minute Essay

- The end of the chapter compares VLIW (what the Itanium uses) to superscalar. VLIW forces the compiler to package the instructions together such that the hardware can execute them without hazards. What are the good and bad implications of this relative to the superscalar approach?
- We being chapter 7 next time.