Lecture 7:
Overview of Advanced Processor
Architectures

Processors
- Pipelined
- Advanced Pipelining
- Superscalars
- Very Long Instruction Word (VLIW)

Pipelined Processors
In-order, overlapped execution of instructions.
Eg. 5-stage pipeline
- instruction fetch,
- decode and register operand fetch,
- execute, memory operand fetch, and
- write-back results.

MIPS R4000 has an 8 stage pipeline.
Causes of Pipeline Delays

Data dependences - RAW hazards
  • register bypass and code reordering by the compiler.

Register hazards
  • WAW hazards - instructions may reach the WB stage out-of-order.
  • No WAR hazards.

Branch delays
  • Compiler fills branch delay slots vs hardware performs branch prediction.

Structural hazards due to nonpipelined units.

Register writes when multiple instructions reach WB stage at the same time (issue vs retire rate).

Advanced Pipelining

In-order issue but Out-of-order execution

DIVD FO, F2, F4
ADDD F10, FO, F8
SUBD F8, F8, F14

Execute SUBD before ADDD

Dynamic scheduling - Scoreboard, Tomasulo's

Superscalar Processors

• Multiple instructions can be issued in each cycle.
  • Speculative Execution is incorporated (commit or discard results).

AMD-K7 is a 9-issue superscalar.
PowerPC is a 4-issue superscalar.
VLIW

- Each long instruction contains multiple operations that are executed in parallel.
- Compiler performs speculation and recovery.

Multiflow 500 can issue up to 28 operations in each instruction (instructions can be up to 1024-bits).

Itanium – 128 bit instruction, 3 operations (40-bit), template (8-bits)

Control Dependences - Instruction Window

Superscalar

Hardware branch prediction guides fetching of instructions to fill up the processor’s instruction window.
Instructions are issued from the window as they become ready, that is, out-of-order execution is possible.

VLIW

Programs are first profiled.
The compiler uses the profiles to trace out likely paths. A trace is a software instruction window.
Instruction reordering is performed by the compiler within the trace.

Data Dependences - Exploiting ILP

Superscalar

Memory dependences:
HW load-store disambiguation techniques used for enabling out-of-order execution.
False register dependences: Avoided using register renaming.
True data dependences: Must be honored. Value prediction for out-of-order execution of dependent instructions.

VLIW

Memory dependences:
Detected by the compiler using dependency analysis or using address dependency analysis.
False register dependences:
Avoided by the compiler through renaming (memory) and register allocation.
True data dependences:
Are strictly followed. Reordering is possible with HW support.