Instruction Scheduling

Hardware implementation of register renaming and out-of-order execution using Tomasulo's Algorithm.

Improving IPC

ILP should be at least 4 instructions per cycle
Control Dependencies are mitigated with Branch Prediction
WAR and WAW data dependencies are removed with register renaming
RAW Data dependencies can be improved through out of order execution
Structural Dependencies can be improved with wider issue processors.

Tomasulo's Algorithm

Tomasulo's algorithm determines which instructions have inputs ready for execution, it uses a form of register renaming, and it is very similar to the method used today.
The differences between what is used today and Tomasulo's algorithm are:

1. All instructions use the algorithm, not just floating point ones.
2. Hundreds of instructions are considering when performing out-of-order execution and register renaming, not just a few.
3. There is now support for exception handling.

Instruction to Broadcast Path

Data Manipulation Path:
Instruction Queue → Reservation stations (values come from the registers)→ Execution Units (Adders and Multiplier) → Broadcast on the Bus

Load/Store Path:
Instruction Queue → Adder (for PC) → Load or Store Buffer → Memory → Broadcast on the bus

Issue = instructions exit the queue and go to either the RS or the PC-Adder.
Dispatch = Instruction exits RS and goes to either the Adder or the Multiplier for execution.
Write Result or Broadcast = the instruction exits the Adder or the Multiplier and is put on the bus.

Issue

1. Next instruction from the IQ - instructions are issued in program order
2. Determine origin of inputs
3. Get free RS of the correct kind - if there are no free RS, the instruction needs to wait for an open RS.
4. Put instruction in the appropriate RS
5. Tag the destination register - this register will hold the result of the instruction and all other instructions will be able to access this result if necessary.

Issue Example

Steps for Issue

1. Take instruction from IQ
2. Look in the RAT
   if RAT has a value for the register - use the register pointed to
   if RAT does not have a value - look in the register file

3. Reservations Stations - some are adders and some are for multipliers
   if an RS is open - use it
   if an RS is not available - wait for one to be open

4. Store in the RAT the register that holds the result of the instruction.

Dispatch

Dispatch needs to latch the results and determine which instructions are ready to execute.

1. Free the RS
2. Match the broadcast tag with the operations in the RS
3. Insert the value in the appropriate RS
4. Once an RS has all of its required inputs - it can execute in the Adder or the Multiplier.
5. When the result is ready from the Adder or the Multiplier, broadcast it on the bus.

If more than 1 instruction is ready to Broadcast

Which instruction should be dispatched first.

- Can choose the oldest instruction first
- The instruction with the most dependencies goes first (how many instructions are waiting for this result)- this is difficult to implement in hardware
- Random selection

Broadcast

1. Put the tag and the result on the bus.
2. Write the result to the register file.
3. Update the RAT. An empty RAT entry means the value is ready in the register file
4. Free the reservation station (change the valid bit)

What if there is More the 1 Broadcast Ready?

How to decide which result is broadcast first?

1. A separate bus for each arithmetic unit - this increases the hardware needs
2. Give priority to the slower unit - the slower unit instructions will most likely have more dependencies. The multiplication/divide unit takes more cycles to complete - so it is usually given priority.

Broadcasting a Stale Result

A result is stale if it no longer has an entry in the RAT.
A stale result is broadcast and placed in the appropriate RS entry. The RAT is not altered because any future instructions will not use this value.
Remember this on Midterms and Finals

Review

For each instruction:

- Issue
- Dispatch
- Write Result

In each cycle (at the same time)

- Issue
- Dispatch - one for each arithmetic unit
- Write results for different instructions

Can an issue and dispatch be done in the same cycle? It depends on the processor.
Issue time is shorter than the cycle time, so there can be time to do the dispatch. But the cycle should be as short as possible, so it should not be feasible.

Capturing operands and dispatching should not be done in the same cycle.

Broadcasting and issuing to the same RS can be done in the same cycle but it requires additional logic to work.

Load and Store Instructions

There can be data dependencies through memory.

- RAW occurs when a store word to a memory location occurs after a load word of that same memory location.
- WAR occurs when a load word to a memory location occurs after a store word.
- WAW occurs when there are two store words to the same memory location.

These need to obeyed or eliminated by;

- -Doing loads and stores in-order
- -Identify dependencies and reorder - this is done in modern processors, but not in Tomasulo's algorithm.