Performance

Eric McCreath

Australian National University

Increasing Word Size

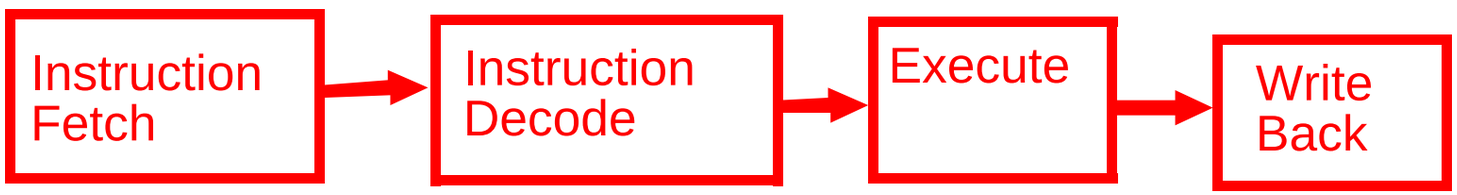
- A simple way of improving performance is to increase the data word size. This means that each instruction operates on a larger amount of data.
- This will involve more gates within the CPU. Also it means some overhead when you wish to operate on data which is smaller than the word size.

On Chip Caches

- CPUs have moved caches onto the CPU die which enables the CPU to be physically closer to the cache. This reduces latency.

Pipelining

- The execution of 1 instruction normally involves a number of stages. These stages are generally independent of each other and work on different parts of the CPU. e.g. while the CPU is executing one instruction it can be fetching the next. This is a little like a factory assembly line.
- So although it may take a number of clock cycles to execute one instruction one instruction can be started on every clock cycle.



Pipelining

http://en.wikipedia.org/wiki/Instruction_pipeline

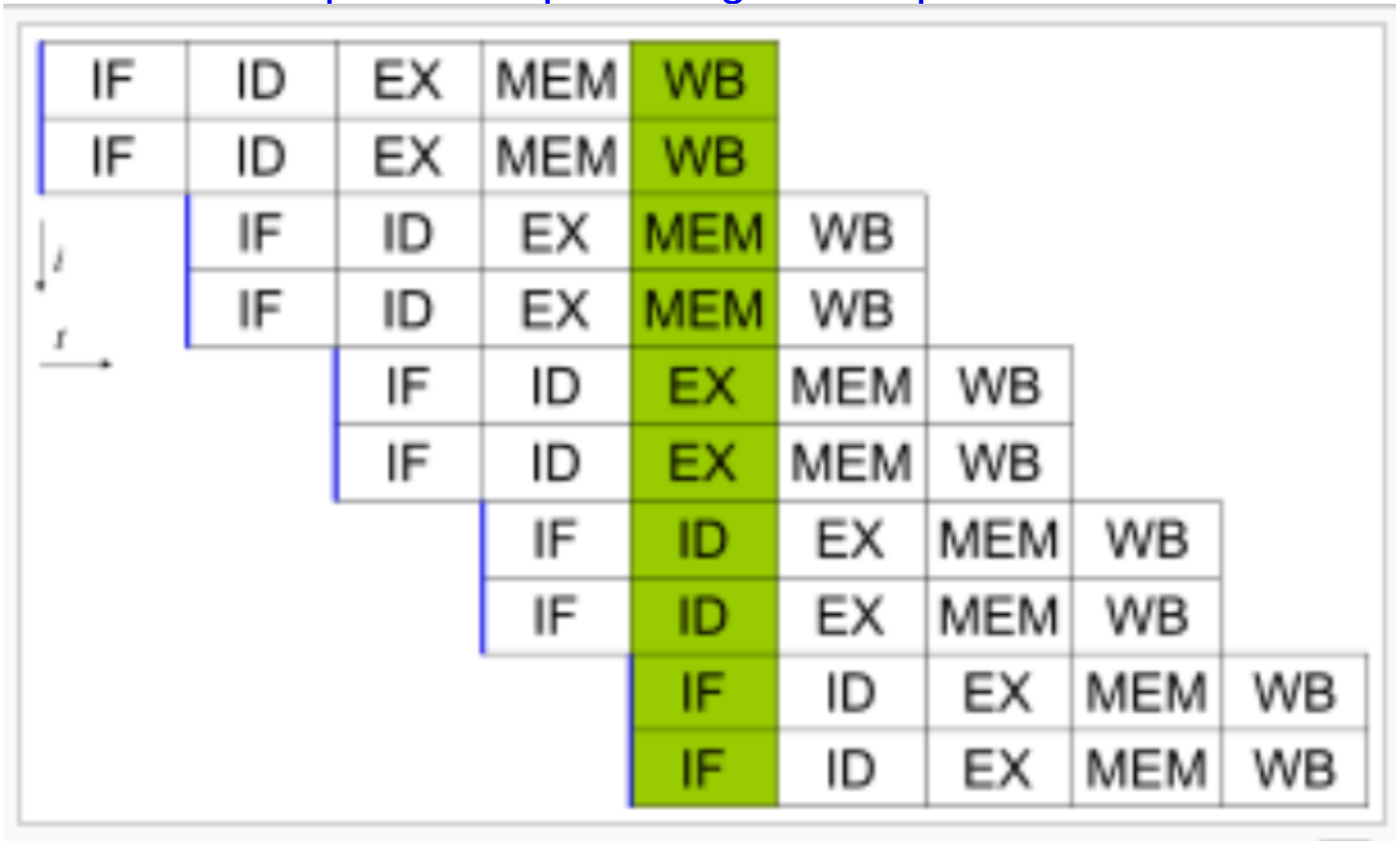
Instr. No.	Pipeline Stage						
1	IF	ID	EX	MEM	WB		
2		IF	ID	EX	MEM	WB	
3			IF	ID	EX	MEM	WB
4				IF	ID	EX	MEM
5					IF	ID	EX
Clock Cycle	1	2	3	4	5	6	7

Basic five-stage pipeline in a RISC machine (IF = Instruction Fetch, ID = Instruction Decode, EX = Execute, MEM = Memory access, WB = Register write back). In the fourth clock cycle (the green column), the earliest instruction is in MEM stage, and the latest instruction has not yet entered the pipeline.

Superscale

- Superscale architectures involve duplicating functional units within the cpu and then starting more than one instruction on the same clock cycle in the pipeline. This enables a larger throughput of instructions.

http://en.wikipedia.org/wiki/Superscalar



Outer of order execution

- Sometimes instructions will require data from memory before they can execute, this will stall the pipeline. This can slow the CPU down greatly.
- The "Outer of order execution" approach loads the next few instructions and starts executing the instruction that has the required data, this means instructions may be executed "out of order".
- Often there is dependencies between instructions, the CPU must be mindful of these.

Multi-Threading

- CPUs can maintain the programming context of multiple threads (so duplication of state and register information), without the duplication of processing units, caches, TLBs, etc, this enables multiple threads to be executed within the one core. This can hide latency. So while one thread is waiting on a result another can be executing. Switching between threads is very cheap as it is all done in hardware.
- From the programers perspective it just looks like you have a SMP (Symmetric Multiprocessing) system.

Multi-core

- The CPU can be duplicated, so effectively you have a number of CPUs which share the same memory.
- Often they will also share an L2 Cache.