CS147 - Lecture 16

Kaushik Patra
(kaushik.patra@sjsu.edu)

1

- Instruction Level Parallelism (ILP)
- Hardware Threading

Reference Books / Source:

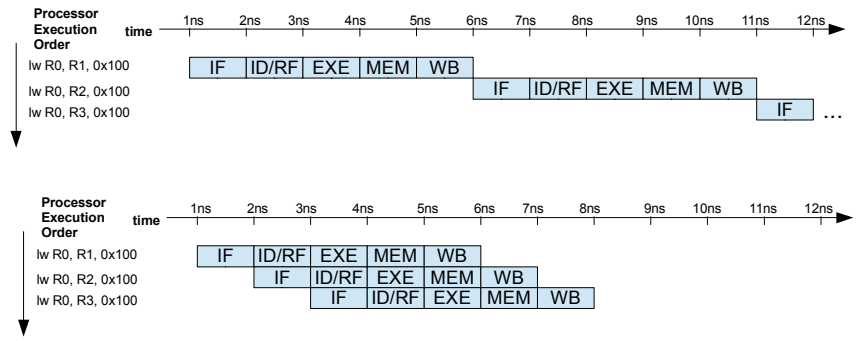
1) *Chapter 16 of 'Computer Organization & Architecture' by Stallings*
2) *http://www.oracle.com/technetwork/systems/opensparc/opensparc-t1-page-1444609.html*

Instruction Level Parallelism (ILP) ...

2

Instruction Level Parallelism (ILP)

- Instruction Level Parallelism (ILP) overlaps different stages of single instruction execution.

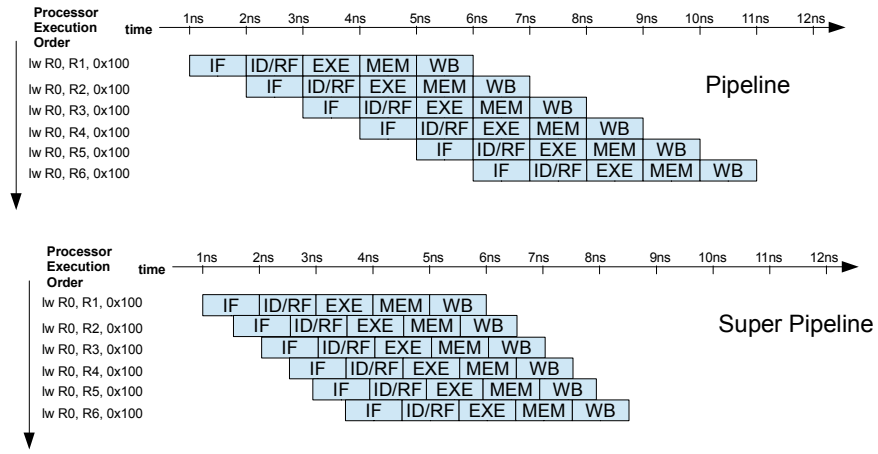


3

- As we have already seen in a pipeline architecture instruction execution are broken down into resource independent stages and stages are overlapped with each other. In essence, even if the target program is executed in serial, each individual instructions are executed in parallel to decrease overall execution time. This type of parallelism is known as Instruction Level Parallelism or ILP.

ILP – Super pipeline

- Super pipeline implements stage overlap at half clock cycle.



4

- The term super pipeline was coined in 1988. In this strategy each stage is done in half clock cycle. It have been observed that many of the stages in the pipeline can be performed with half clock cycle. Hence, we can use 2x faster internal clock to drive pipe line.

- In this super pipeline design an internal clock is used in the design which has double frequency than rest of the system.

Super pipeline performance

- For a balanced n-stage super pipeline structure the maximum speedup obtained is 2n.

$$Speedup_{max} = \frac{E_{non-pipelined}}{E_{pipelined}} = 2n$$

- For a balanced n-stage super pipeline CPI is improved by 2n times – since each stage of super pipeline is completed with double efficiency compare to a normal pipeline.

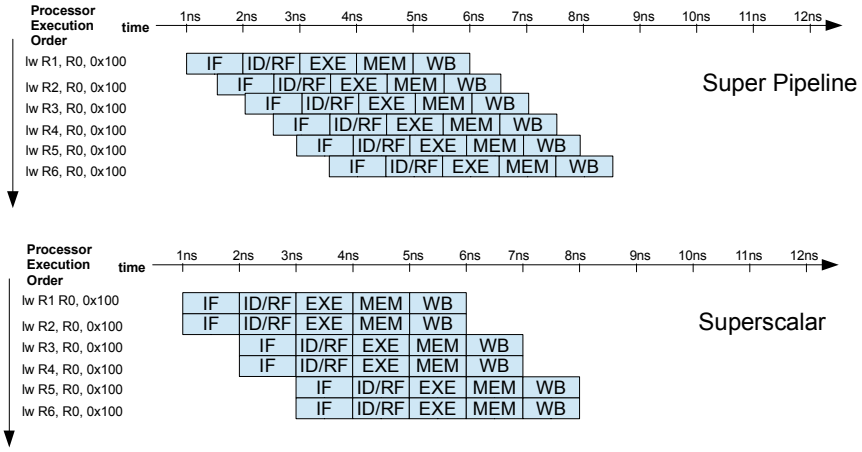
5

- For a n-stage balanced super pipeline, maximum speed up obtained is 2n. For a n-stage general pipeline the maximum speedup is n. Since, for a super pipeline, the stages are completed in half the time that of the normal pipeline, the maximum speedup obtained in 2n.

- Similar to a normal pipeline, we can also claim that the speedup is obtained by increasing the CPI by 2n.

ILP – Superscalar

- Super pipeline implements stage overlap at half clock cycle.

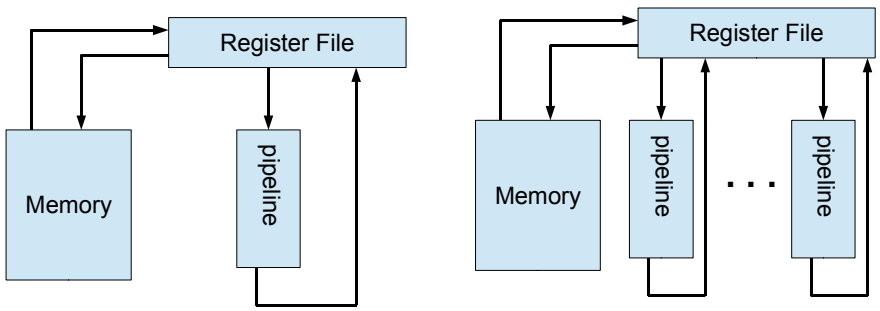


6

- A scalar program is a program where each instruction is completed in sequence. On the other hand, a super scalar architecture performs complete multiple instructions in parallel. Unlike a pipeline or super pipeline, where instructions are completed in serial order, a superscalar architecture can execute and complete independent instructions at the same time parallel to each other in a true sense.

- The number of instructions completed at a point of time depends on how many duplicate execution resources are put together.

Scalar vs. Superscalar



- Superscalar uses multiple pipelines in parallel.

7

- A scalar (with pipeline) architecture contains single pipeline data path taking input from register file and writing back to same register file. For a superscalar architecture, multiple number of such pipelines are attached to the register file. Controller can issue instructions to the corresponding pipelines in parallel and get them done.

Superscalar performance

- For a super scalar structure with m number of balanced n-stage pipeline, the maximum speedup obtained is (m*n).

$$Speedup_{max} = \frac{E_{non-pipelined}}{E_{superscalar}} = m * n$$

- For a super scalar structure with m number of n-stage balanced pipeline CPI is improved by (m*n).
  - If m=2, it matches performance improvement in super pipeline.

8

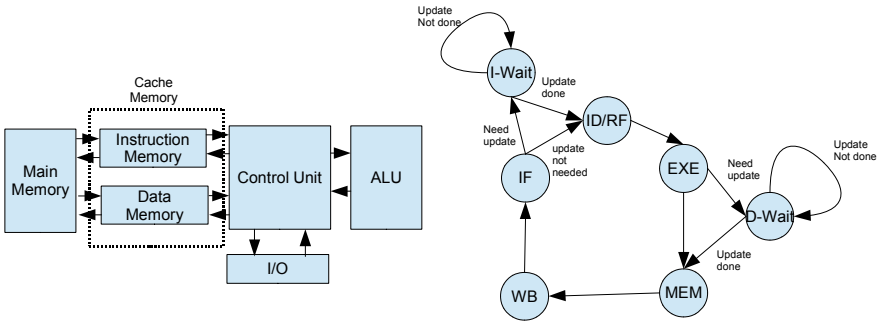
- In comparison to non-pipelined version, a superscalar architecture can give a maximum speedup of (m*n) where m is the number of parallel pipeline in the architecture and n is the number of stages in each of the pipeline. Each n-stage pipeline will give maximum speedup of n and hence m such pipeline will give total (m*n) speedup.

Hardware Threading ...

9

Hardware Threading - Concept

- Fetching instruction or data from cache memory may not always be one cycle operation (later we'll study it in memory hierarchy discussion) – it may take a significant amount of time.
  - It needs update on the Instruction or Data Cache.
  - It will stall / halt the pipe line till the memory access is done.
  - While waiting, other instructions could be started in parallel.

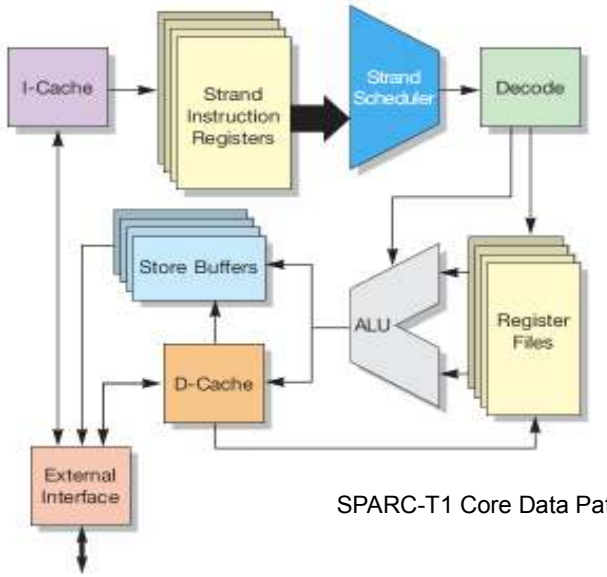


10

- The intermediate memory structures (cache memory), to hold instruction and data memory content in separate, are usually smaller in size compare to main memory. Thus it often happens that complete program of data set can not be stored entirely onto the cache memory. When some part in the cache is not needed or already used and other part needs to be brought into the cache from main memory, system must replace the cache content with new content from the memory. Usually main memory is implemented with slower memory which takes longer time to read and write compare to the cache memory. In additional, this cache content replacement happens in block of data (not just a single data). Therefore, it takes significant amount of time to reload cache memory with relevant information. If such reloading happens (which is more commonly occurred), the entire pipeline is stalled.

- In the state transition diagram of a classic 5-stage pipeline control unit, two new state must be added. One state is to denote that the instruction cache is reloaded and other is to denote that that data cache is reloaded. Once pipeline goes into such state, it is possible to re-use the hardware for a complete independent instruction (may be from a different program or an independent portion of a parallel program).These independent instructions are called a strand in hardware context (a thread in software concept).

Hardware Threading – Data Path



SPARC-T1 Core Data Path

11

- In hardware threading data path, multiple instruction registers and multiple register files and multiple store buffer are maintained for all the strands (typically 32 or 64). The instruction cache (I-cache) is loaded with multiple programs to be executed in parallel on a computer system. Each program is associated with a fixed strand instruction register (SIR). Multiple instructions are fetched at a time and stored into the SIR. If there is a need to reload the I-cache, the pipeline can continue executing instructions from the SIR, serving independent set of programs while the I-cache is being reloaded. The instructions in the SIR are scheduled by a scheduling hardware, strand scheduler. Once the instruction is scheduled, it is decoded and corresponding register file is accessed. Each strand has its own separate register file (therefore this architecture contains multiple register files, one for each strand). The execution is done in a single ALU. The result is stored into data memory (or D-cache) incase of memory operation. However, if the D-cache is busy doing content reloading, the result is into corresponding data storing buffer. Once the D-cache loading is done, controller takes the store buffer content and write it back into the D-cache. The register write back is done to the register file corresponding to the strand under execution.

CS147 - Lecture 16

Kaushik Patra
(kaushik.patra@sjsu.edu)

12

- Instruction Level Parallelism (ILP)
- Hardware Threading

Reference Books / Source:

1) *Chapter 16 of 'Computer Organization & Architecture' by Stallings*
2) *http://www.oracle.com/technetwork/systems/opensparc/opensparc-t1-page-1444609.html*