Implementation and Utilization of Instruction Level Parallelism in Modern CPU Architectures

Justin Reed

Advisor:
David Wonnacott

Background: How can we increase the speed of the CPU? Fundamental ideas behind hardware parallelism.

Background Control Unit

Already familiar with the basics of pipelining? Skip to the next section.

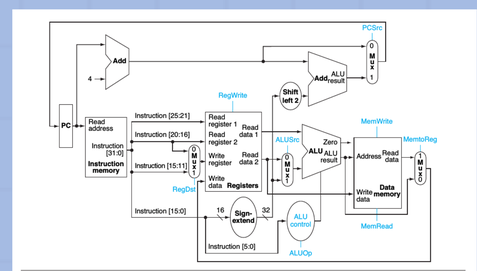

FIGURE 4.15 **The datapath of Figure 4.11 with all necessary multiplexors and all control lines identified.** The control lines are shown in color. The ALU control block has also been added. The PC does not require a write control, since it is written once at the end of every clock cycle; the branch control logic determines whether it is written with the incremented PC or the branch target address.

Fig 1. Decrease clock cycle length using multicycling [PH17a]

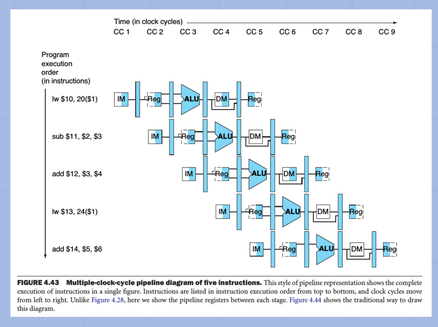

FIGURE 4.43 **Multiple-clock-cycle pipeline diagram of five instructions.** This style of pipeline representation shows the complete execution of instructions in a single figure. Instructions are listed in instruction execution order from top to bottom, and clock cycles move from left to right. Unlike Figure 4.28, here we show the pipeline registers between each stage. Figure 4.44 shows the traditional way to draw this diagram.

Fig 2. Maximize potential of multicycled CPU with pipelining [PH17a]

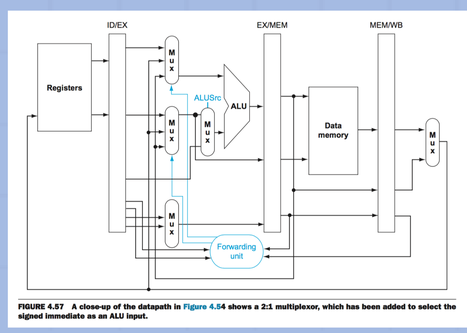

FIGURE 4.57 **A close-up of the datapath in Figure 4.54 shows a 2:1 multiplexor, which has been added to select the signed immediate as an ALU input.**

Fig 3. Technical implementation of basic pipelined CPU [PH17a]

Summary: CPU pipelining and its implications

Though the multi-cycle instruction is slightly slower than its single-cycle equivalent, it has the key advantage of making a pipelined architecture possible. Such an architecture begins executing new instructions before the previous ones have completed, taking advantage of the fact that each sub-instruction only needs a portion of the chip to execute. Multiple instructions are therefore able to execute simultaneously, improving throughput by a factor close to the number of sub-instructions per multi-cycle instruction.

Simultaneous Multithreading: Making use of instruction and thread level parallelism at the same time.

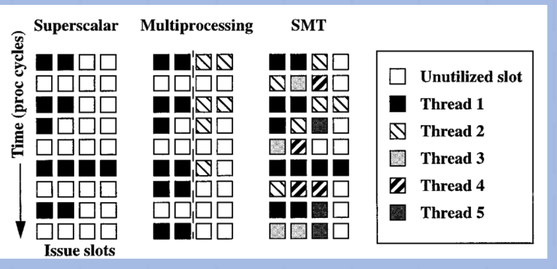

Fig 4. How TLP and ILP are viewed as equivalent by the SMT core [LEL+ 97]

Summary: why SMT matters

The advantage of a SMT CPU is greatest for single-thread applications, as these programs are implicitly converted to multithreaded applications at execution time, allowing for thread-level parallelism where there previously wasn't any. But even for multithreaded applications, SMT CPU's maintained a clear advantage. Since SMT can consider ILP and TLP interchangeably, the SMT CPU achieves high performance for both single and multi-threaded applications. On average, the performance increase from the SMT architecture is 52-percent greater. This performance increase is so decisive that nearly all modern general-purpose processors now make use of SMT.

Recent bottlenecks: Reducing memory latency

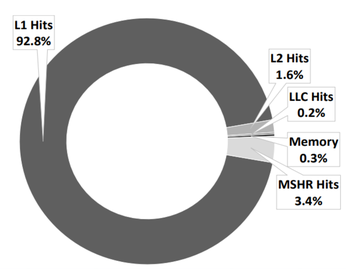

Fig 8. Frequency of cache-misses by memory type [SBGS22]

Utilize maximal ILP with out-of-order and in-order execution

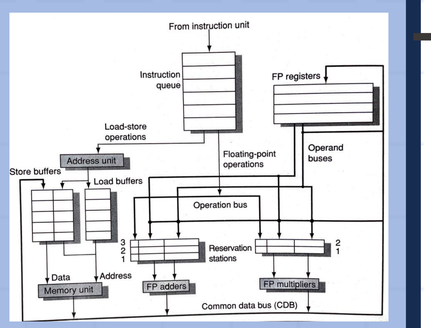

Fig 5. An out-of-order execution processor diagram [PH17b]

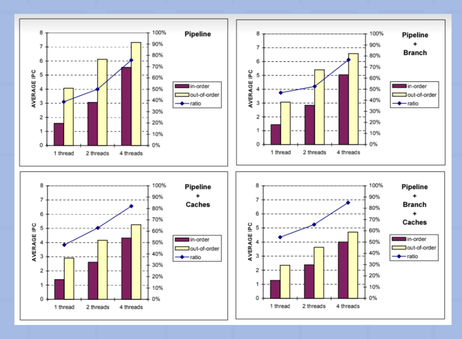

Fig 6. OOO loses its edge in the context of SMT [HS99]

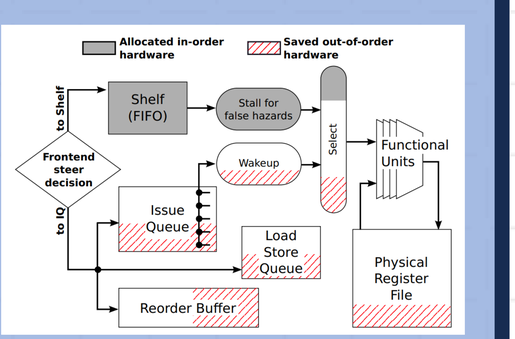

Fig 7. Solution: use a hybrid architecture [SW16]

Sources and future work

While the results of [HS99] indicate an important result regarding in-order execution, the experiment was done on a 1999 processor. One possible area of future work would be to replicate this paper on a modern processor.

[SW16] Faissal M. Sleiman and Thomas F. Wenisch, "Efficiently Scaling Out-of-Order Cores for Simultaneous Multithreading," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ISCA '16 (Seoul, Republic of Korea: IEEE Press, 2016), 431–43, https://doi.org/10.1109/ISCA.2016.45.

[LEL+ 97] Jack L. Lo et al., "Converting Thread-Level Parallelism to Instruction-Level Parallelism via Simultaneous Multithreading," *ACM Trans. Comput. Syst.* 15, no. 3 (August 1997): 322–54, https://doi.org/10.1145/263326.263382.

[PH17a] Patterson, David A., Author, John L Hennessy, and Perry Alexander. *Computer Organization and Design: The Hardware/Software Interface*. [Amsterdam ; Boston: Elsevier/Morgan Kaufmann, ©, 2017]

[PH17b] J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, ISSN (Elsevier Science, 2017), https://books.google.com/books?id=cM8mDwAAQBAJ.

[HS99] S. Hily and A. Seznec, "Out-of-Order Execution May Not Be Cost-Effective on Processors Featuring Simultaneous Multithreading," in *Proceedings Fifth International Symposium on High-Performance Computer Architecture*, 1999, 64–67, https://doi.org/10.1109/HPCA.1999.744321.

[SBGS22] Sudhanshu Shukla et al., "Register File Prefetching," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22 (New York, NY, USA: Association for Computing Machinery, 2022), 410–23, https://doi.org/10.1145/3470496.3527398.