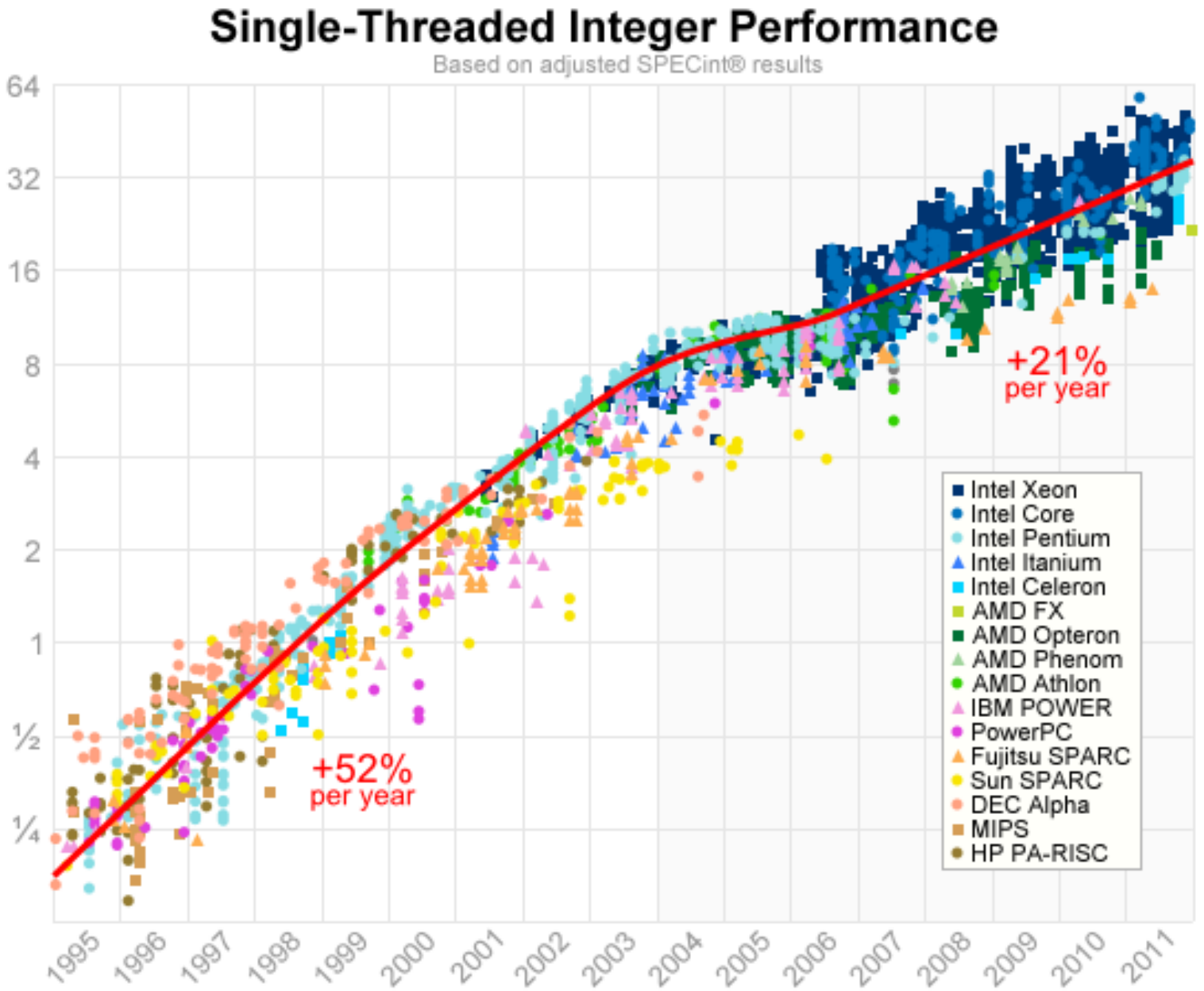
Calcul haute performance pour les systèmes embarqués (HPEC)
Lab 1 - Advanced CPU features

Bertrand LE GAL

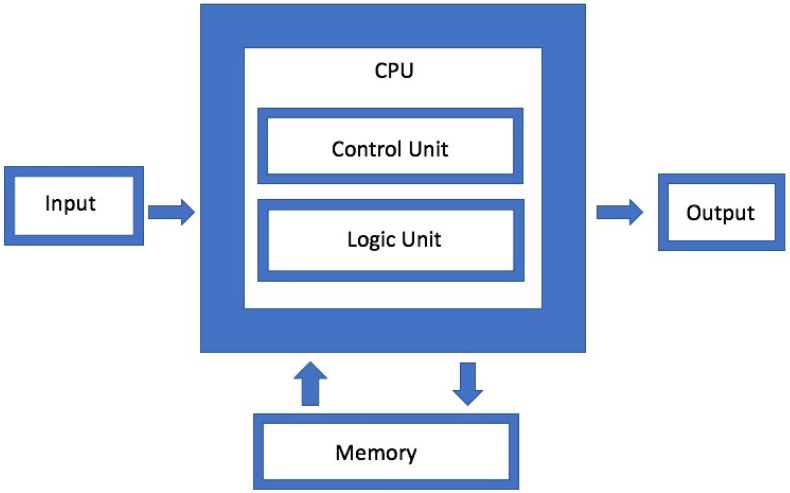
IMS laboratory, CNRS UMR 5218
Digital Circuits and Systems team
Bordeaux-INP, University of Bordeaux
France

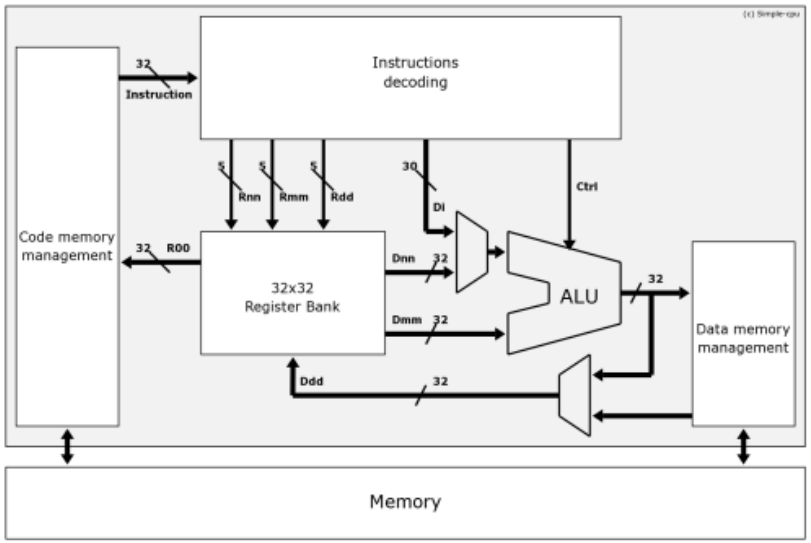
bertrand.legal@enseirb-matmeca.fr

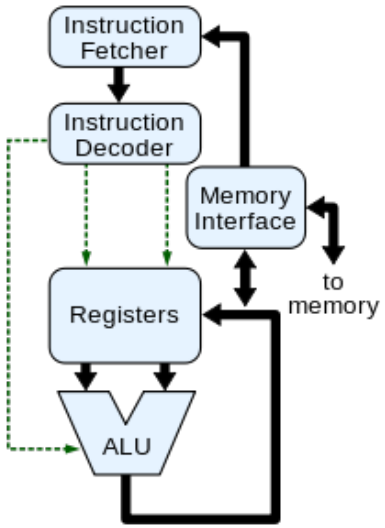
Evolution of the processing performance of CPUs



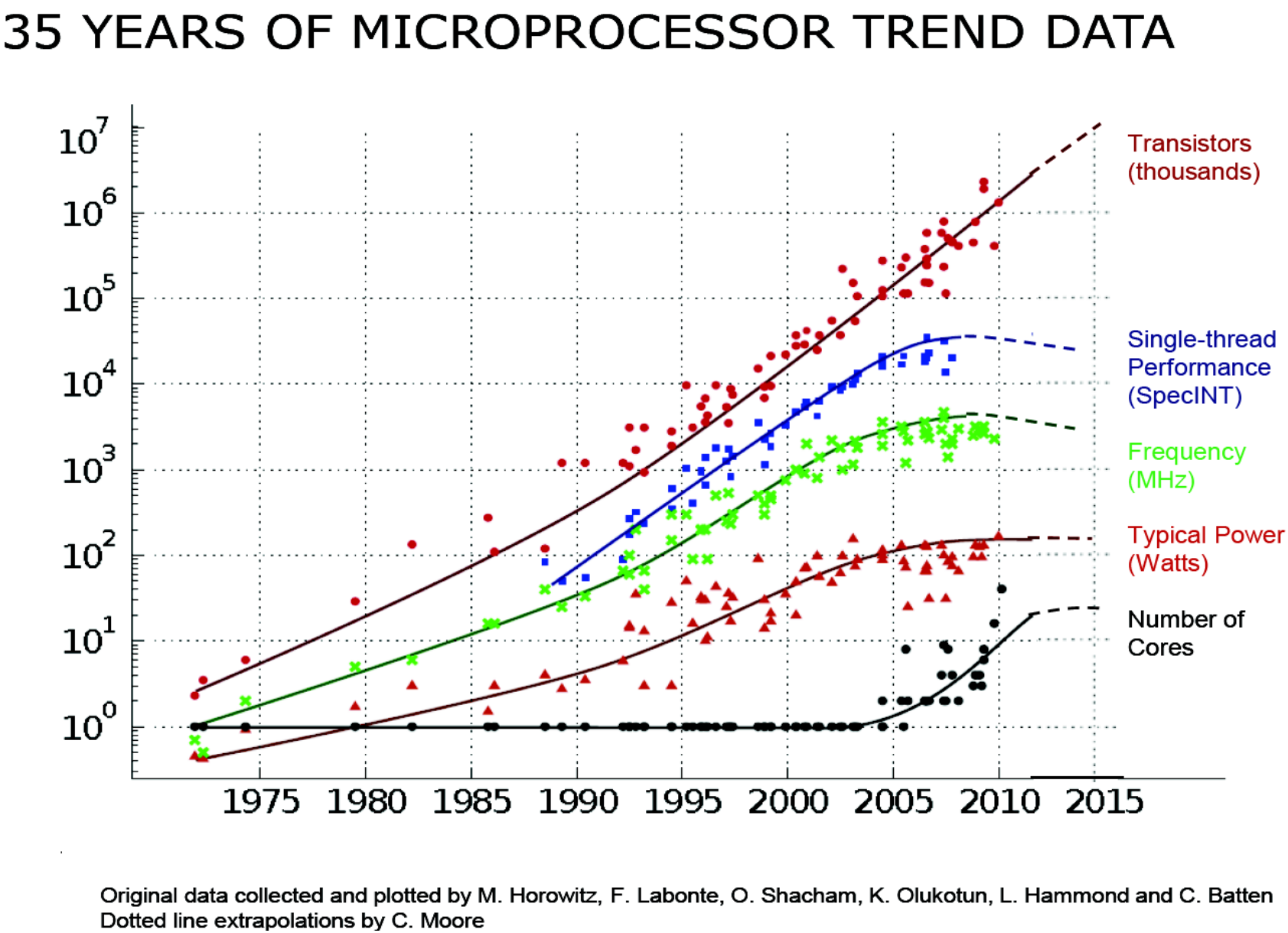
From microcontrollers to microprocessors





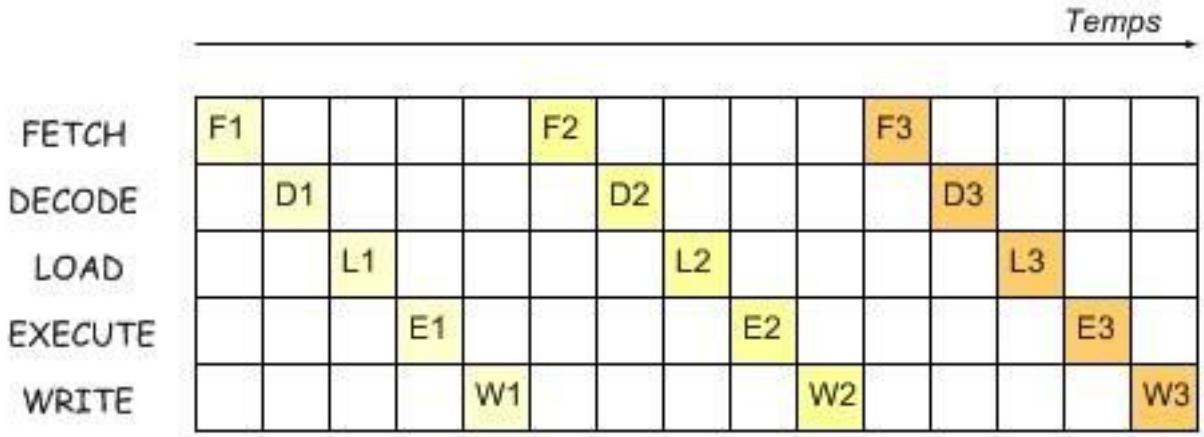


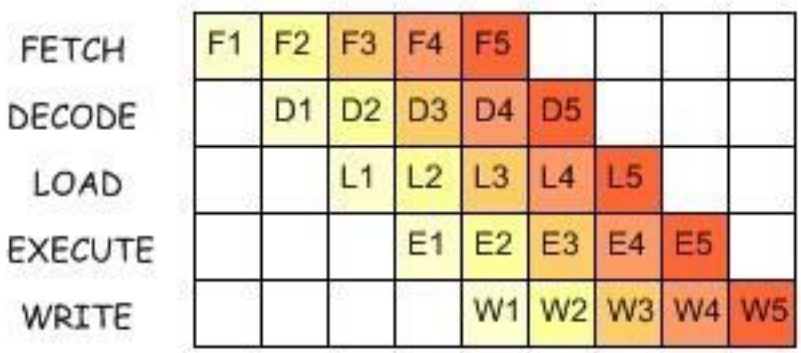
Introduction au pipelined architectures



Instruction pipelining in processors

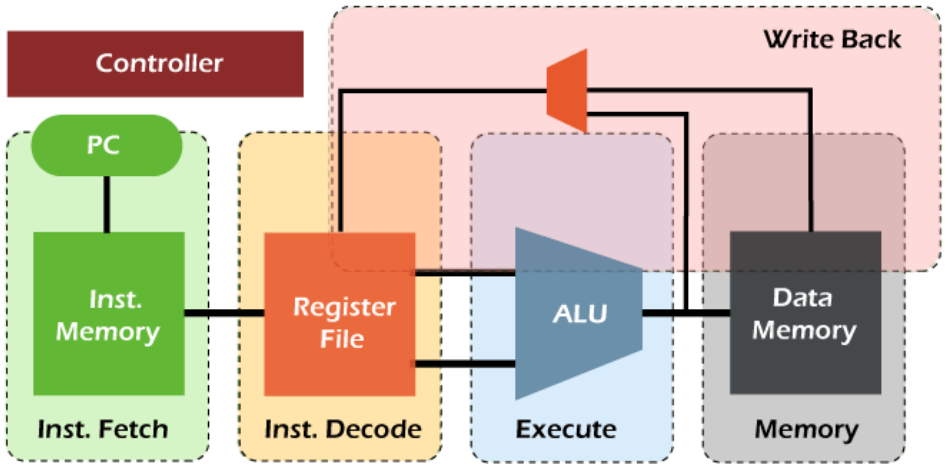
- To be executed a instr. needs to execute: fetch, decode, load, execute and write operations.
  - 1 intr. needs 5 clock cycles
- This sequential task can be split into parallel sub-tasks
  - The same operation are executed over diff. instructions
- Performances
  - Reduces silicon critical paths,
  - Increase execution parallelism,
  - Need hardware to manage pipeline.





Some characteristics of INTEL processors

- Number of pipeline stages
  - 5 stages - (Pentium, 1993)
  - 14 stages - (Pentium III, 1995)
  - 31 stages - (Pentium IV, 2006)
  - 14~19 stages - (Core-i7, 2016)



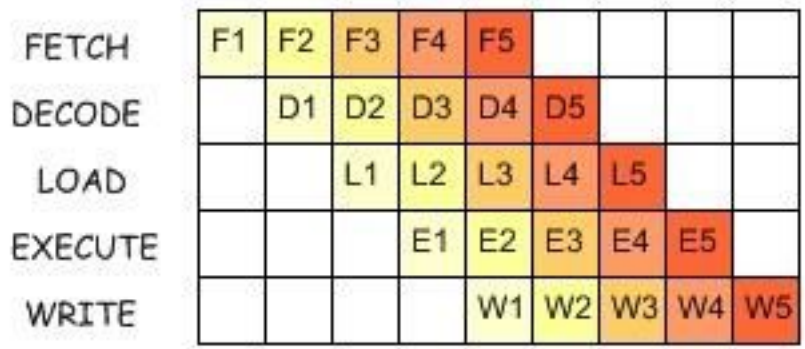
- Execution time of x86 instr. (EX)
  - Integer addition (1 cycle)
  - Float addition (5 cycles)
  - Integer multiplication (3 cycles)
  - Float multiplication (6~7 cycles)
  - Integer division (13~44 cycles)
  - Float division (15 cycles)

https://www.agner.org/optimize/instruction_tables.pdf

https://en.wikipedia.org/wiki/Comparison_of_CPU_microarchitectures

Execution hasards & data dependencies

- Pipeline is (usually) efficient when it is full of (independent) instructions,
  - Not so easy from an algorithmic point of view !



- Pipeline stalls happened frequently:
  - Data dependency
  - Control dependency
  - High latency operation
  - Memory access (see after)

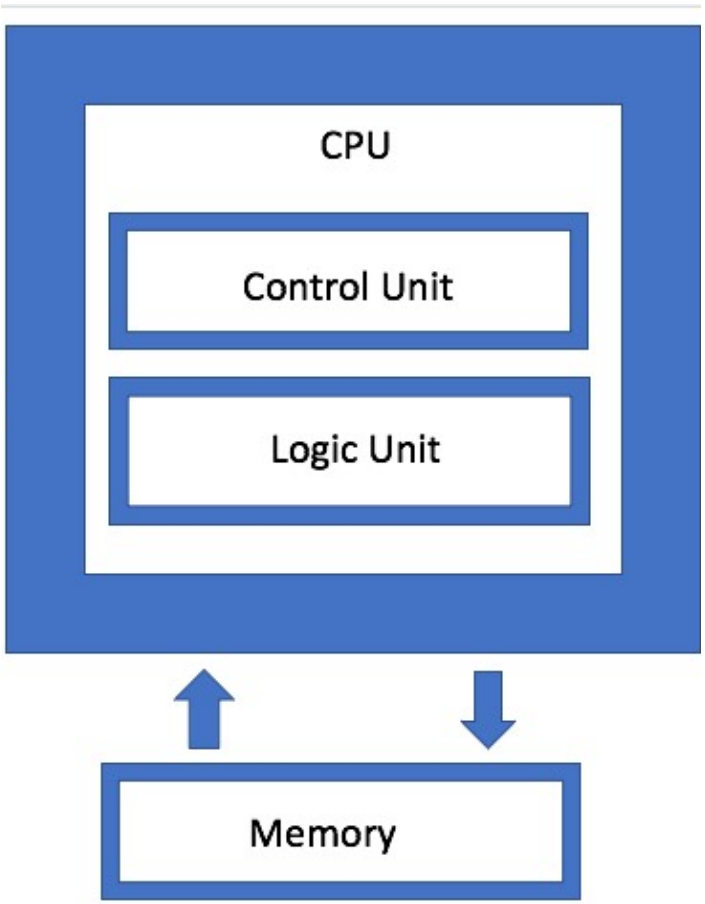
https://fr.wikipedia.org/wiki/Pipeline_(architecture_des_processeurs)

Pipeline execution and control statements

- Another pipeline usage issue comes from control statements
  - Control statement involve compare and jump instructions,
  - Which instr. should be loaded ???
  - NOP vs branch prediction.
- Branch prediction unit
  - Different approaches,
  - Impacts on performance,
  - Impacts on hardware cost.
- It is possible to avoid parts of them at software level.

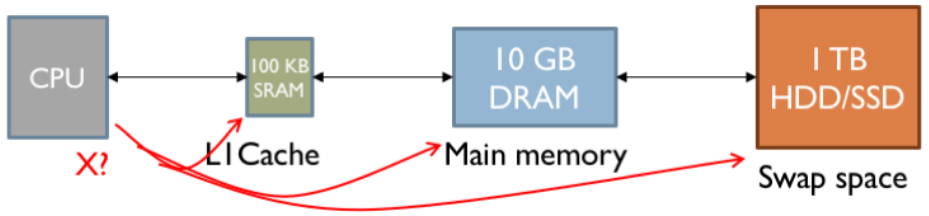
Introduction to memory hierarchy

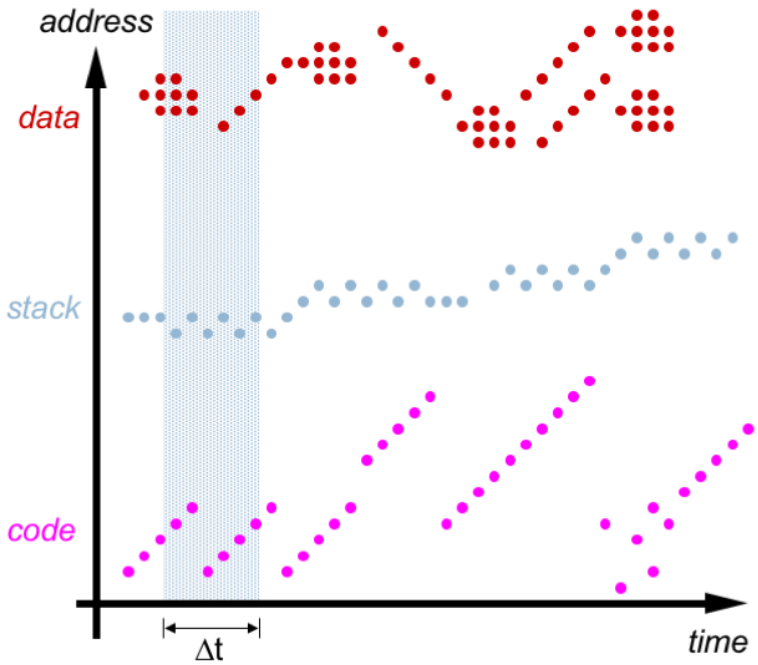
- A software program
  - Execute arithmetic computations
  - Load/store values in memory
- Hardware memory characteristics
  - Some GB on data,
  - Works @GHz frequencies,
  - High energy consumption (refresh),
  - Are very slow compared to CPU.
- Between CPU and DDR memory
  - Smaller & faster memories are inserted,
  - They manage data locality efficiently.



Memory cache behavior

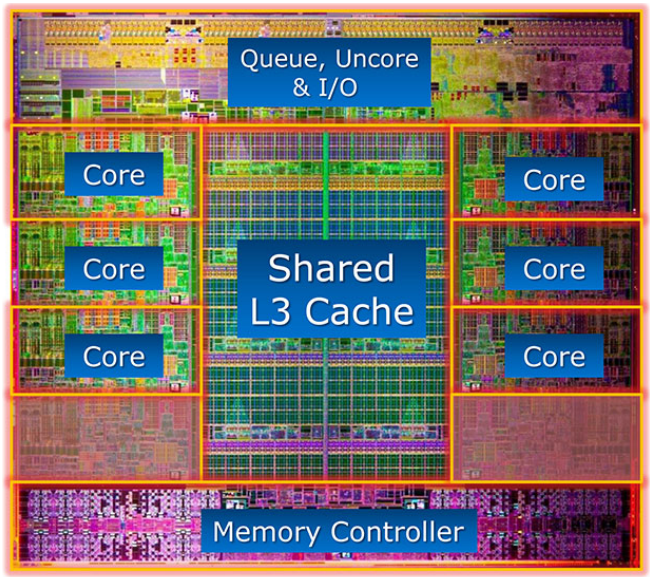
- Data move (automatically) from memory to registers,
  - Dedicated hardware resources make the job for you,
- Date are stored in cache for future usage (reduce time),
  - Interesting when data locality is high,
- Cache have limited sizes
  - Replace values when caches are full,
- Cache ctrl predicts memory access to anticipate moves.

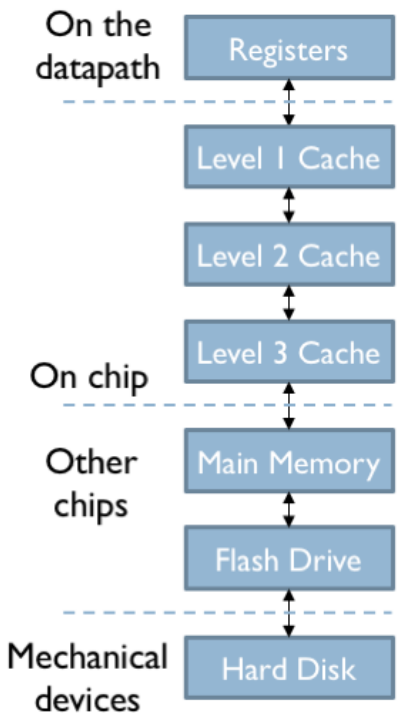




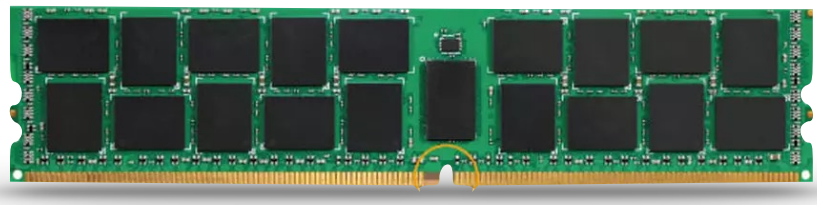
Properties of memory levels

- The memory hierarchy depends on processor targets
  - Large caches increases performances
  - Cache memory are costly,
  - Tradeoff (energy, cost, performance),
- Software designer should take to cache characteristics.





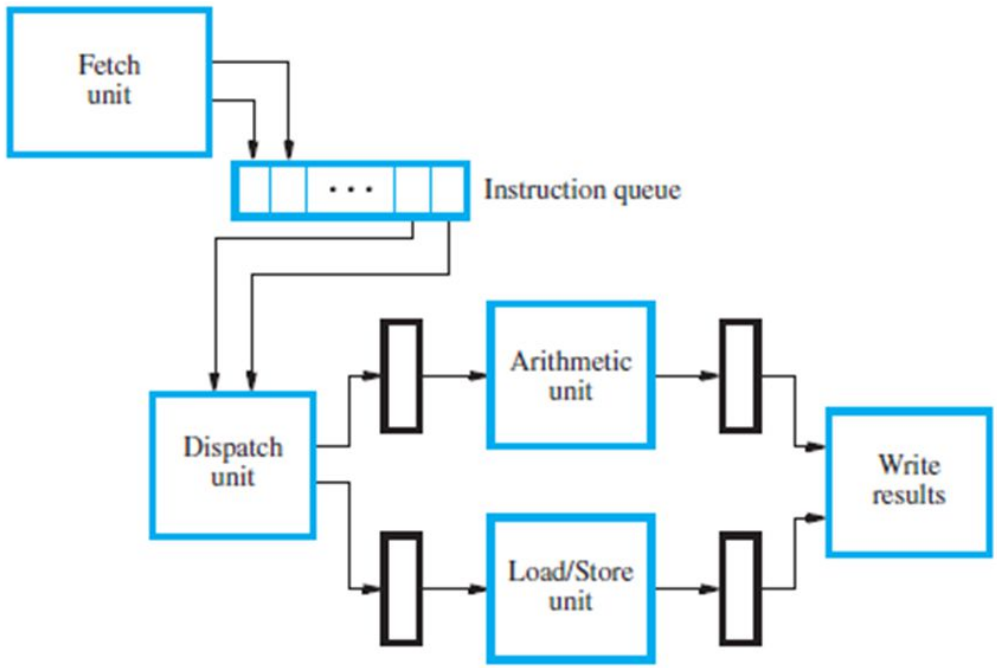
Access time	Capacity
1 cycle	1 KB
2-4 cycles	32 KB
10 cycles	256 KB
40 cycles	10 MB
200 cycles	10 GB
10-100us	100 GB
10ms	1 TB



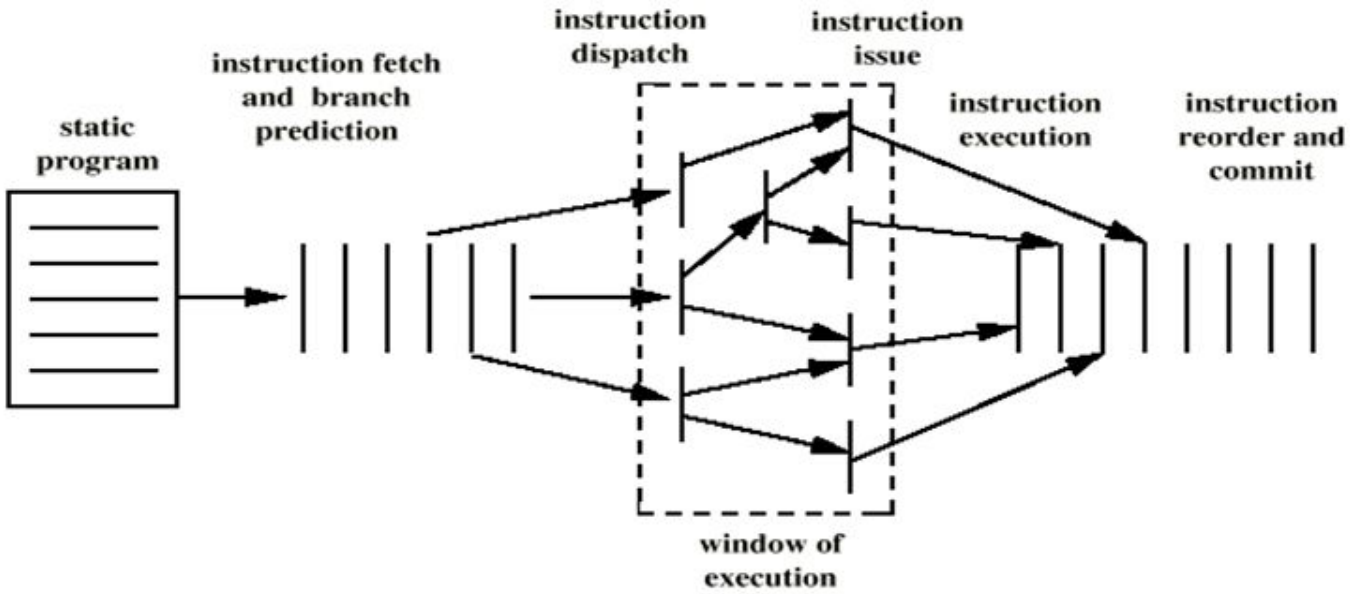
https://computationstructures.org/lectures/caches/caches.html

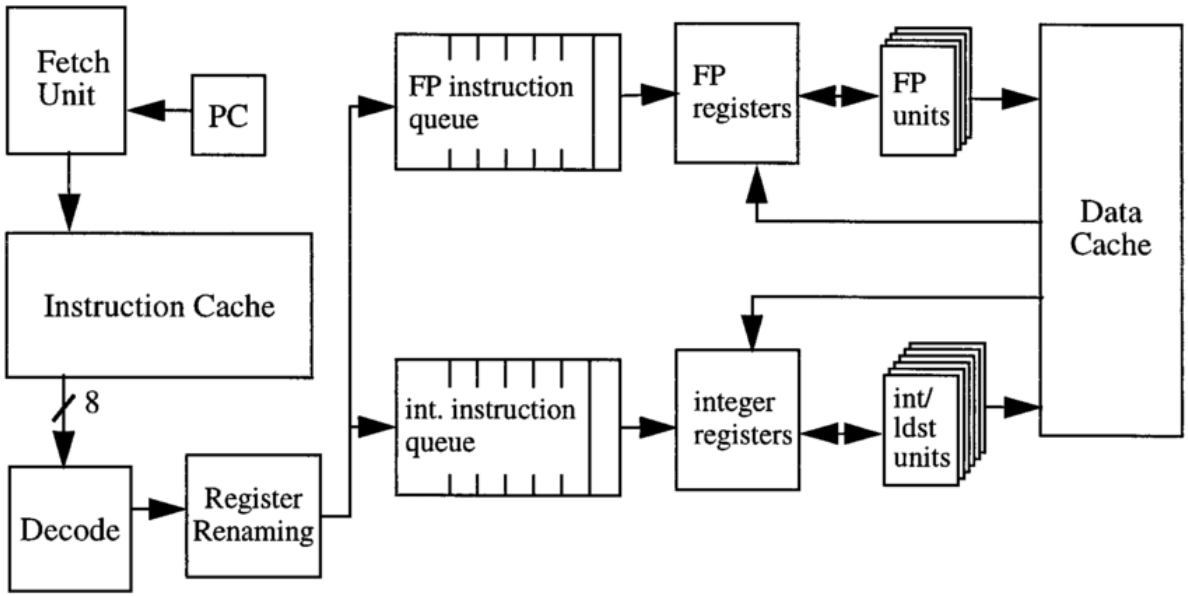
Introduction to superscalar feature

- Executed instruction
  - Targets an hardware resource,
  - Has a variable execution time.
- Architecture offers
  - A large set of hardware resources,
  - ALU & others resources underuse.
- Superscalar architecture
  - Execute more than one instruction at a time depending on resource availability.
- Architectures can support in-order or out-of-order execution.



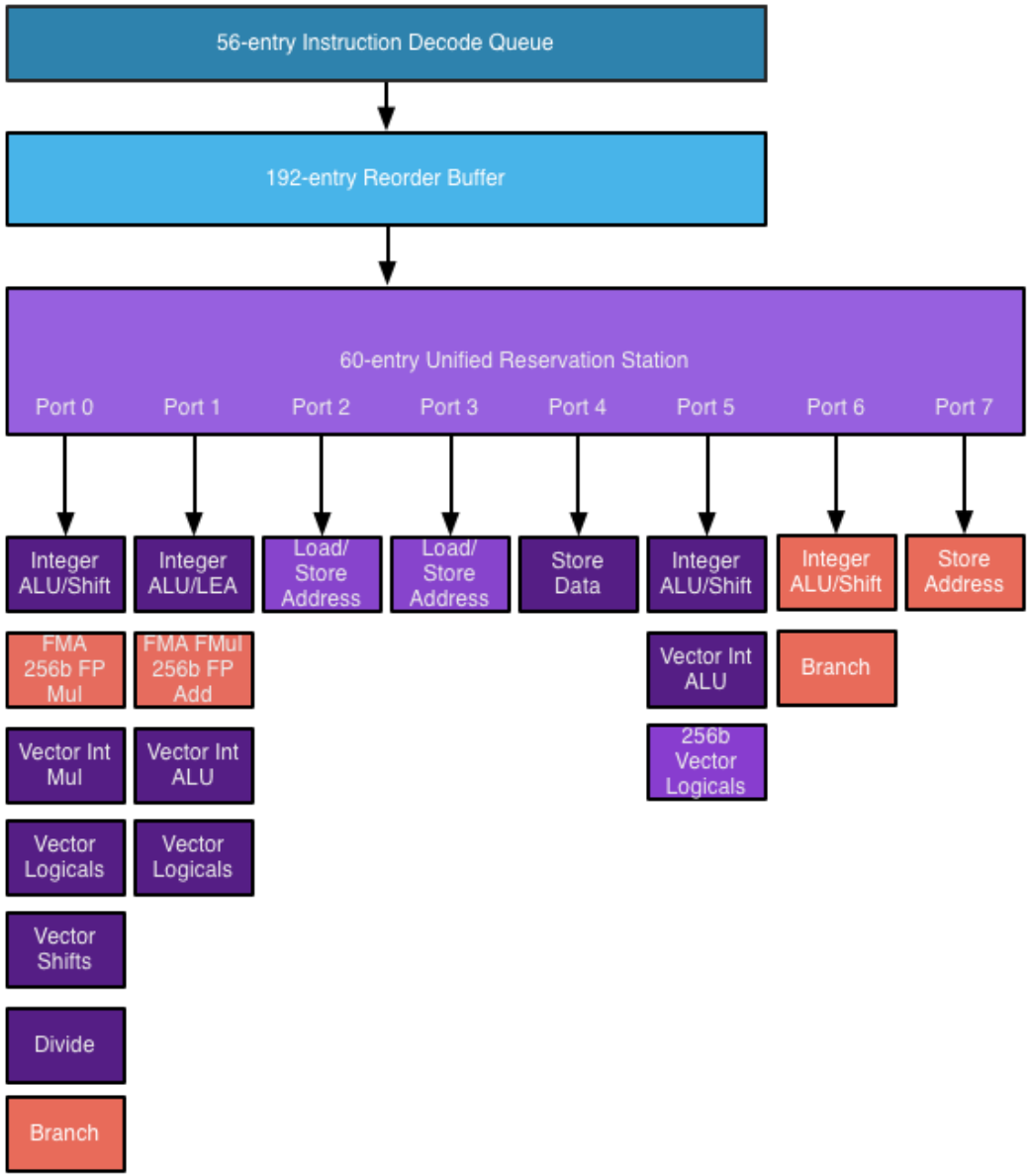
Behavior of superscalar architectures





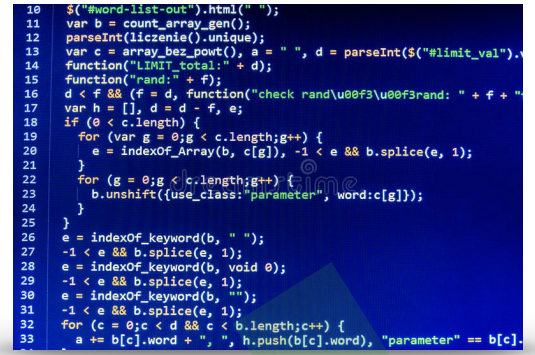
Intel Haswell Execution Engine

- On INTEL architectures
  - Up to 8 instr. can be executed per cc.
  - IPC level grows from 1 to 8 theoretically.
- Units are grouped into ports
  - Asymmetric port capabilities,
  - ex. division and multiplication can't be processed in parallel.
- A processor can execute
  - 8 × clock frequency instructions per second,
  - Not so easy to achieve…
- Software descriptions can help…



Source code and compiler impacts on perf.

- Off-the-shelf processors
  - General Purpose Processor can execute word processing or games,
  - (very) complex architectures to be as efficient as possible,
- Processor cannot do everything
  - Software designer can help (code writing style, algo choice),
  - Software compiler can help too !
- Between two software programs executing the same behavior,
  - Speed up from 2 to 100 !



Conclusion

- CPU architectures are complex
  - Features depend on CPU vendors,
  - Pro/Con trade-offs
- Algorithms impacts on performance level,
  - Source code writing too !
  - Don’t forget the compiler…
- To obtain an efficient application, YOU should take care about:
  - The processor features,
  - Your code writing style,
  - The compiler optimizations.