High-Performance Architecture Lectures

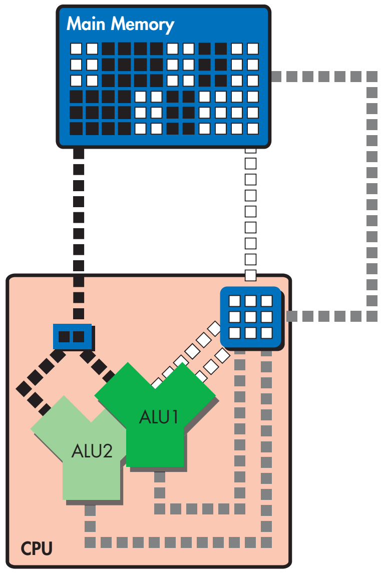
1. Basic Computer Organization – *What is a processor and how it works?*
   – Design of PDcLX-1 processor
2. Program Execution – *How does a Code run on a Processor?*
   – Programming PDcLX-1 processor
3. Pipelined Processor – *Increase Performance of our Processor*
   – How much speed-up with pipelined processor? What it is the cost of it?
4. **Scalar Processor – *Increase Performance of our Processor***
   – **PDcLX-2 and why ISA is important**
5. On the way to Supercomputers – *Caches, Multicore Processor, Networks*
   – Beskow Supercomputer

Superscalar Processor

Stefano Markidis and Erwin Laure
KTH Royal Institute of Technology

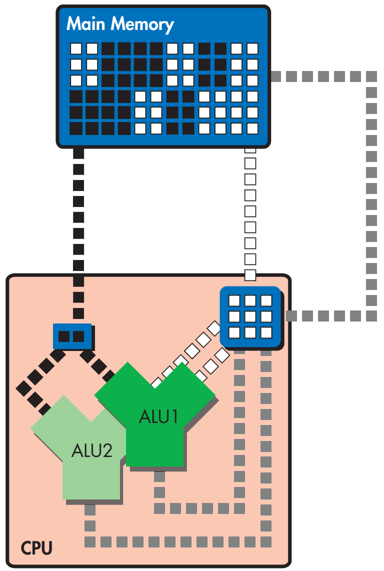
Superscalar Computer

- During the decades following the first integrated circuit, the number of transistors that could be packed onto a single chip increased at a stunning pace.
  - Possible to put **more than one ALU on a chip**
    - have both **ALUs working in parallel** to process code faster
- Since these designs could do **more than one scalar operation at once**, they were called ***superscalar*** **computers**
  - The RS6000 from IBM in 1990 and was the world’s first commercially available superscalar CPU
  - Intel followed in 1993 with the Pentium, which, with its two ALUs, brought the *x*86 into the superscalar era



Our Superscalar Processor PDcLX-2

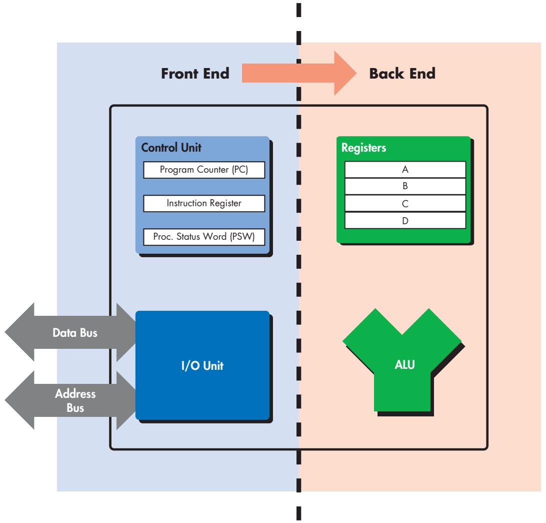
- I introduce a *two-way superscalar* version of the PDcLX-1, called the **PDcLX-2**.
  - It has two ALUs, so it's able to execute **two arithmetic instructions in parallel**
  - These two ALUs share a **single register file**

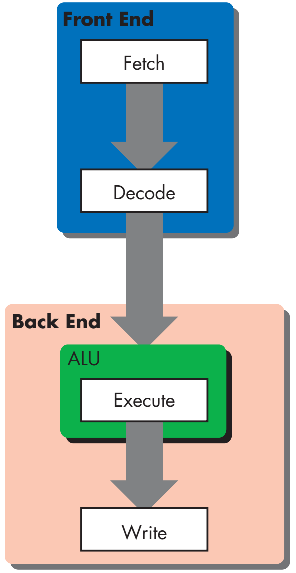


Basic Instruction Flow

One useful division that computer architects often employ when talking about CPUs is that of

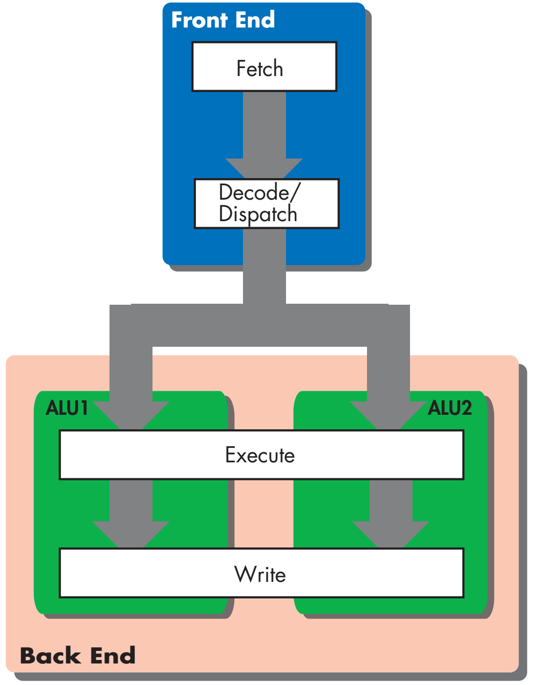
- ***front end*** where fetching and decoding takes place
- ***back end*** where executing and write back take place





Dispatch Circuitry to Determine Parallel Execution

- The superscalar processing adds a bit of complexity design
  - new circuitry to **reorder the linear instruction stream** so instructions can execute in parallel.
    - This circuitry has to ensure **that it's "safe" to dispatch two instructions to the two execution units**.
- Note that we have renamed the second pipeline stage ***decode/ dispatch***.
  - Attached to the latter part of the decode stage is a **dispatch circuitry** to **determine whether or not two instructions can be executed in parallel**
    - If yes, the dispatch unit sends one instruction to the first integer ALU and one to the second integer ALU.
    - If no, the dispatch unit sends them in program order to the first of the two ALUs.



Impact on Programming Model

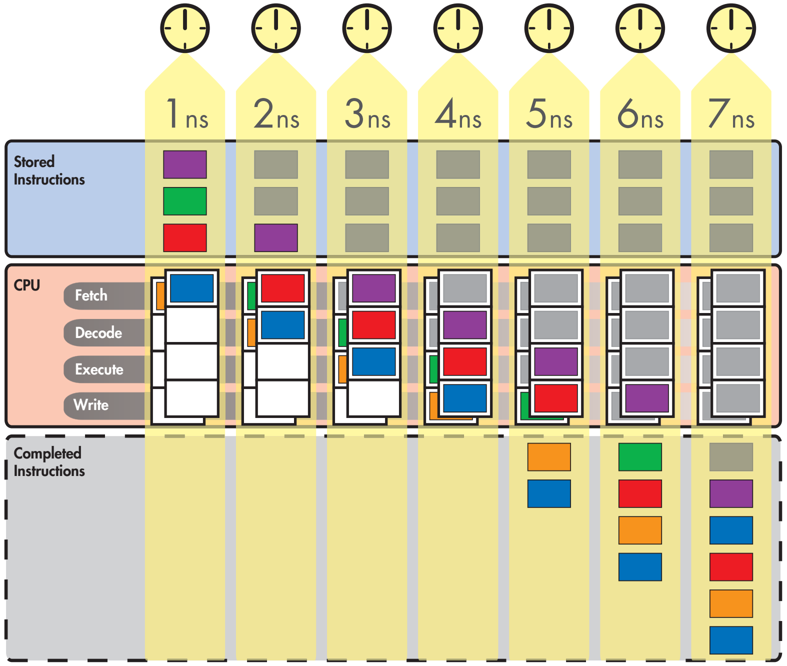
- Even though the superscalar processor has multiple ALUs, the programming model does not change.
  - The programmer still writes to the same interface, even though that interface now represents a fundamentally different type of machine than the processor actually is
  - **The interface represents a sequential execution machine, but the processor is actually a parallel execution machine!**

Branching in Superscalar

- Fetching and decoding two instructions at a time complicates the way the PDcLX-2 deals with branch instructions.
- What if the first instruction in a fetched pair happens to be a branch instruction that has the processor jump directly to another part of memory?
  - The **second instruction in the pair has to be discarded**. This wastes fetch bandwidth and introduces a bubble into the pipeline.

What is the instructions/s performance?

- One instruction per clock was the maximum theoretical instruction throughput for a pipelined processor
- Because a superscalar machine can have multiple instructions n multiple write stages on each clock cycle
  - **the superscalar machine can complete multiple instructions per cycle**
- The more ALU pipelines that a processor has operating in parallel, the more instructions it can add to that box on each cycle.

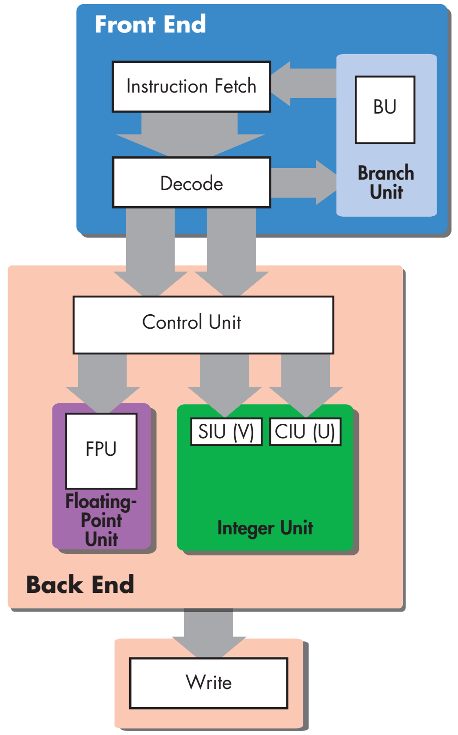


Expanding Superscalar Processing with Execution Units

- Most modern processors do more with superscalar execution than just adding a second ALU.
  - Rather, they distribute the work of handling different types of instructions among **different types of execution units**
    - An ***execution unit*** is a block of circuitry in the processor's back end that **executes a certain category of instruction.**
      - So far the arithmetic logic unit (ALU) was assumed to performs arithmetic and logical operations **on integers.**

Arithmetic Logic Units

- On early microprocessors all **integer arithmetic and logical operations** were handled by the **ALU**
- Initially, **Floating-point operations** were executed by a companion chip, called ***coprocessor*** external to the microprocessor
- **Floating-point capabilities** are now then **integrated** onto the CPU as a separate execution unit



Intel Pentium processor contains two integer ALUs and a floating-point ALU, along with some other units

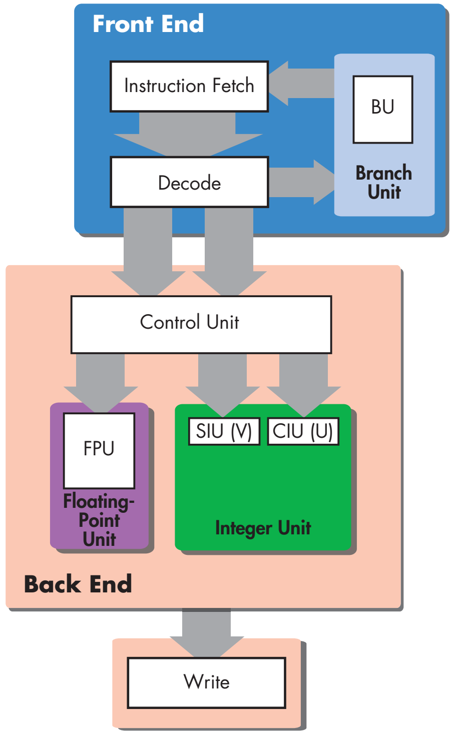
Graphical Processing Units

This Friday, we will look at Graphical Processing Units (GPU) a processor widely in use in supercomputers.

- GPUs concept is similar to the co-processor concepts (or accelerator)

Different Kind of ALUs

- *ALU* is now a general term for any execution unit that performs operations on any type of data.
  - More specific labels will be used to identify the ALUs An ***integer execution unit (IU)*** is an ALU that executes integer arithmetic and logical instructions
  - A ***floating-point execution unit (FPU)*** is an ALU that executes floating-point arithmetic and logical instructions, and so on



Penthium

Memory-Access Units

In almost all of the modern processors, we have two execution units for memory-access instructions

- The ***Load-Store Unit (LSU)*** for the execution of load and store instructions, as well as for *address generation*.
- The ***branch execution unit (BEU)*** is responsible for executing conditional and unconditional branch instructions.
  - The BEU also often has its own *address generation* unit also

Instruction Set Architecture

Our PDcLX-2's *instruction set* still consista of a few instructions for working with different parts of the programming model:

- arithmetic instructions (e.g., *add* and *sub*) for the ALU
- registers
- load and store instructions
- branch instructions for checking the PSW and changing the PC.

We can call this programmer-centric **combination of programming model and instruction set an *instruction set architecture (ISA)*.**

PDcLX-1 and PDcLX-2 have same ISA!

The PDcLX-1 has the same instruction set architecture as the PDcLX-2

- The instruction set and programming model remain unchanged despite the PDcLX-2 ***hardware implementation*** of that ISA is significantly different in that the PDcLX-2 is superscalar.

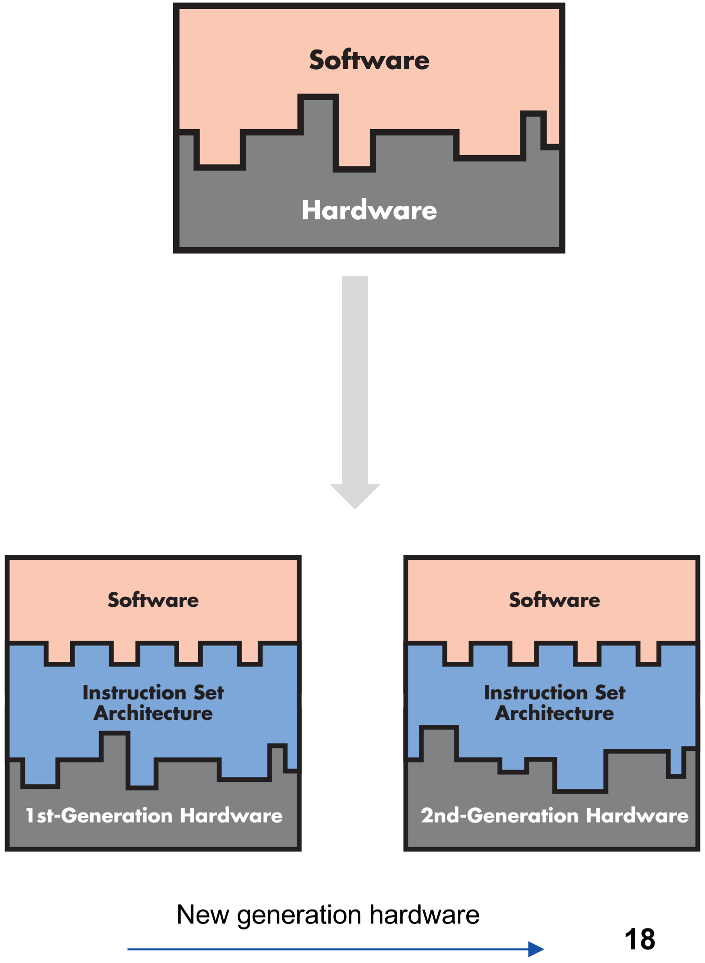
Microarchitecture

A particular processor’s hardware implementation of an ISA is generally referred to as that **processor’s *microarchitecture***

- We call the ISA introduced with the PDcLX-1 the PDcLX ISA.
  - Each successive iteration of our PDcLX line of computers implements the PDcLX ISA using a different microarchitecture
    - The PDcLX -1 has only one ALU, while the PDcLX-2 is a two-way superscalar implementation of the PDcLX-ISA.

Motivation for ISAs

- In the early days of computing, computer makers didn't build a whole line of **software-compatible computer systems**
  - Every time a new machine came out, software needs to be developed
- Once the design and specification of the instruction set was separated from the low-level details of a particular machine's design
  - **programs written for a particular ISA** could run on any machine that **implemented that ISA**



Key Points

- The Superscalar architecture features two or more ALUs sharing the register file increasing the instructions per clock cycle
- Execution units are specialized ALUs: IU, FPU, LSU, BEU
- The usage of same ISA for different microarchitectures guarantees software portability