Computer Architecture
Pipelines

Cain Susko

Queen's University
School of Computing

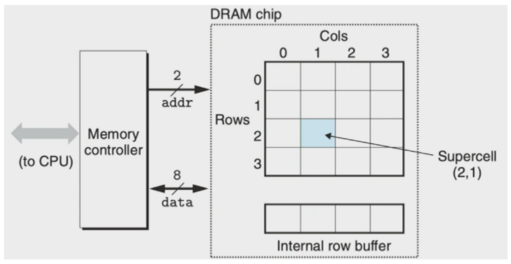
March 15, 2022

Many Bits

Last class, we went over how to implement sequential circuits with a single bit, now we will be exploring how to implement circuits with many bits.

Memory Arrays

Typically, memory is organized as a 2D array of memory cells. Consider the example of a 16x8 memory chip:

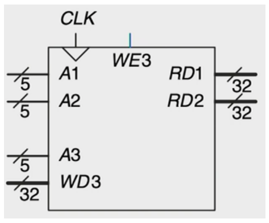


when we read a row, we copy it to the buffer and we address the 8 bits, read them, and send them back to the CPU.

Register File

Processors use many register to store temporary variables. Such a set of registers is calles a register file, and is built as a small, fast memory array. For example, a 32-register, 32-bit, register file with:

- 2 read ports
- 1 write port
- a write-enable control



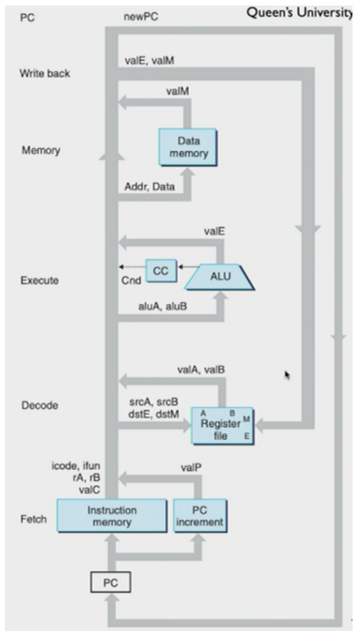
Pipelining

The idea of pipelines is that we will organize our instructions and their execution and instead of having a single instuction being executed we can have many executed at the same time. this is how modern processors work.

Processing an Instruction Processing an instruction requires numerous operations. These operations can be organized in *phases*:

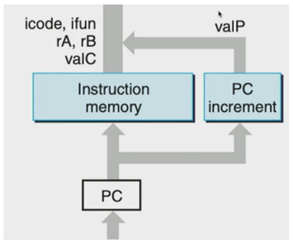
1. fetch
2. decode
3. execute
4. memory
5. write back
6. PC update

This can be represented visually as:



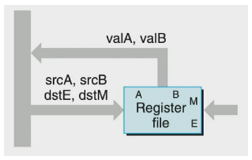
Fetch

goes to memory, gets data, prepares components, and sends it on to be decoded. The fetch operation uses the PC as the address. It then reads bytes of instruction from memory and prepares a new PC, $valP$



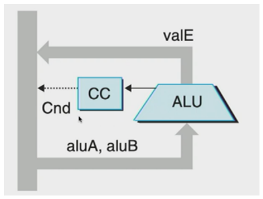
Decode

The decode then uses instructions and function codes to control downstream harware. It Addresses into the register file to read or write values.



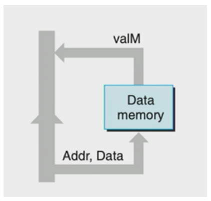
Execute

This part does the arithmatic logic or computation from the decode step. it also sets the codes Cnd which contains the effective address, the stack pointer, and the conditional moves and jumps.



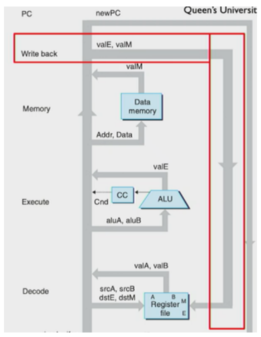
Memory

the memory is contained at $Addr$. It reads from memory, returning $ValM$ and writes $Data$ to memory.



Write Back

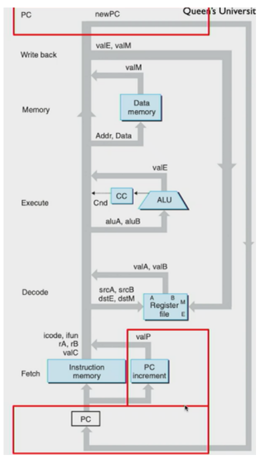
Take the computed value $valE$ and / or value loaded from memeory $ValM$ and write into the register file. This operation is addressed by $dstE$ and $dstM$



this is to take a computed value and save it back to the register.

PC Update

This is to handle the different branches (if there are any) within the program.



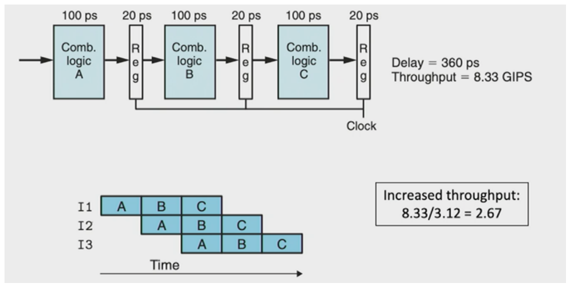
where PC update takes the input:

- the sequential $valP$
- the absolute adress read from memory
- the relative or effective address
- branching condition

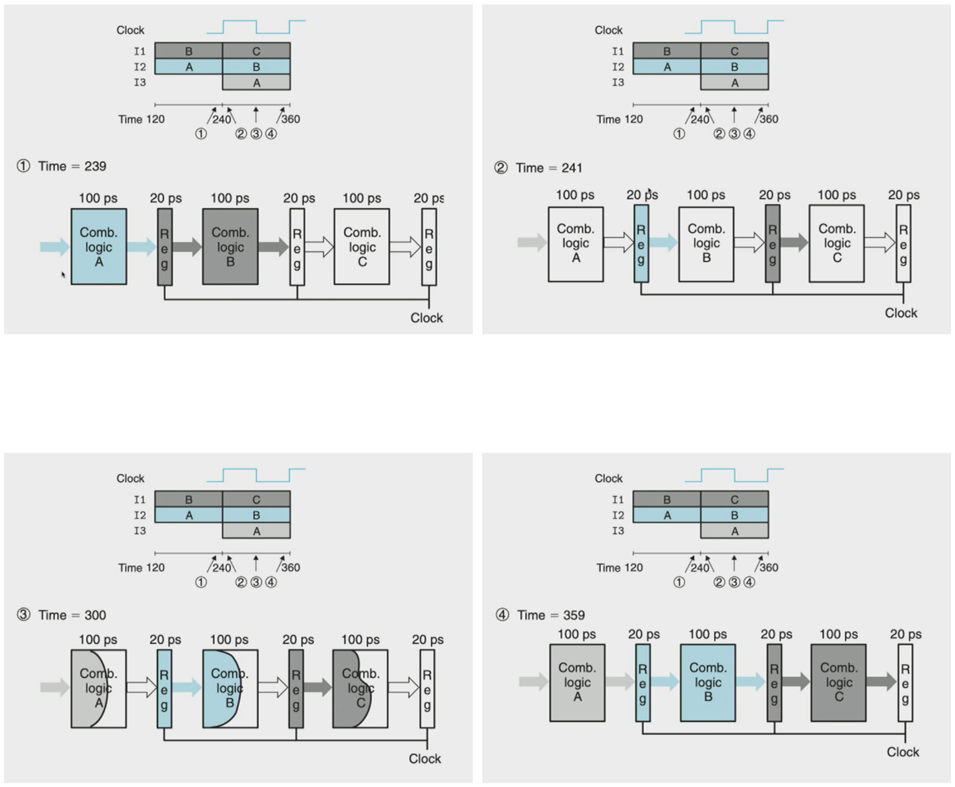
Computation Pipelines

The whole process described above is what is known as Computation Pipeline. An example of a pipeline in the physical world is a car production line.

A 3-stage pipeline can be visually represented as:

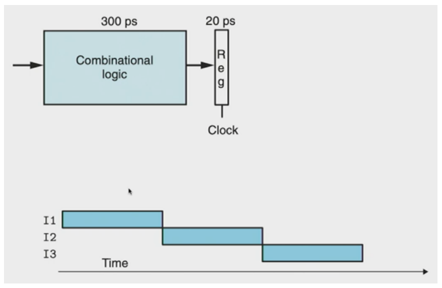


Note: the data is stored between stages in registers, which are all controlled by the same clock. so Pipelining is more accurate but uses more memory.
A closer look at this pipelined circuit we can Observe:



Un-Pipelined Architecture

An example of a architecture that does not use pipelining is the following:

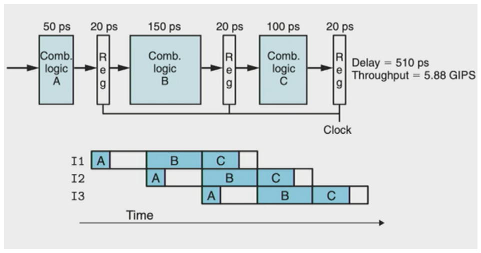


Limitations of Pipelining

There are many structural hazards that that one can encounter when implementing pipelined computing: like when 2 instructions need the same harware resources.
Data hazards arise when one instruction depends on a value produced by a preceding instructions still in the pipeline.
Control hazards include if an instruction should be executed needs a control decision by a preceeding instruction.
Some solutions include:

- sheduling: programmer explicilty avoids
- stall: suspending subsequent instruction
- duplicate: more hardware
- forward: allow data to bypass certain hardware units for early availablity
- speculate: guess the problem and kill speculative results if incorrect

Uninform Partitioning if the logic circuits are not evenly distrubuted over time (non-uniform partitioning) , then the time of setion within the circuit must be allocated as the largest time within the for a section within the circuit. This can be seen more clearly in the following diagram:



Note: we calculate the throughput by the equation:

$$TP = \frac{S}{d}$$

where S is the number of steps or circuits and d is the delay (ammount of time to execute) for the circuit.

Deep Pipelining When one implements a deep (more than 10 sections) pipelining structure. Because pipelining requires a register between each section, the returns from pipelining is diminished.