Pipelined RISC Processor
To pipeline MIPS:
- First build MIPS without pipelining with CPI=1
- Next, add pipeline registers to reduce cycle time while maintaining CPI=1

Source: Arvind (MIT Opencourseware licence http://ocw.mit.edu/terms/index.htm#cc)
Clock period can be reduced by dividing the execution of an instruction into multiple cycles

\[ t_C > \max \{ t_{IM}, t_{RF}, t_{ALU}, t_{DM}, t_{RW} \} \approx t_{DM} \text{ probably} \]

However, CPI will increase unless instructions are pipelined

Source: Arvind (MIT Opencourseware licence http://ocw.mit.edu/terms/index.htm#cc)
An Ideal Pipeline

- All objects go through the same stages
  - No sharing of resources between any two stages
  - Propagation delay through all pipeline stages is equal
  - The scheduling of an object entering the pipeline is not affected by the objects in other stages

- These conditions generally hold for industrial assembly lines.
- But can an instruction pipeline satisfy the last condition?

Source: Arvind (MIT OpenCourseware licence http://ocw.mit.edu/terms/index.htm#cc)
How to divide the datapath into stages

› Suppose memory is significantly slower than other stages. In particular, suppose
  
  $t = 10$ units IM
  
  $t = 10$ units DM
  
  $t = 5$ units ALU
  
  $t = 1$ unit RF
  
  $t = 1$ unit RW

› Since the slowest stage determines the clock, it may be possible to combine some stages without any loss of performance

Source: Arvind (MIT Opencourseware licence http://ocw.mit.edu/terms/index.htm#cc)
Alternative Pipelining

\[ t_C > \max \{t_{IM}, t_{RF} + t_{ALU}, t_{DM} + t_{RW}\} = t_{DM} + t_{RW} \]

\[ \Rightarrow \text{increase the critical path by 10%} \]

Write-back stage takes much less time than other stages. Suppose we combined it with the memory phase.

Source: Arvind (MIT Opencourseware licence http://ocw.mit.edu/terms/index.htm#cc)
### Maximum Speedup from Pipelining

#### Assumptions

1. \( t_{IM} = t_{DM} = 10, \)
   \( t_{ALU} = 5, \)
   \( t_{RF} = t_{RW} = 1 \)
   4-stage pipeline
   - Unpipelined: \( t_C = 27 \)
   - Pipelined Speedup: \( t_C = 10, \) \( \text{Speedup} = 2.7 \)

2. \( t_{IM} = t_{DM} = t_{ALU} = t_{RF} = t_{RW} = 5 \)
   4-stage pipeline
   - Unpipelined: \( t_C = 25 \)
   - Pipelined Speedup: \( t_C = 10, \) \( \text{Speedup} = 2.5 \)

3. \( t_{IM} = t_{DM} = t_{ALU} = t_{RF} = t_{RW} = 5 \)
   5-stage pipeline
   - Unpipelined: \( t_C = 25 \)
   - Pipelined Speedup: \( t_C = 5, \) \( \text{Speedup} = 5.0 \)

It is possible to achieve higher speedup with more stages in the pipeline.

---

Source: Arvind (MIT Opencourseware licence http://ocw.mit.edu/terms/index.htm#cc)
5 stage Pipeline

Source: Arvind (MIT Opencourseware licence http://ocw.mit.edu/terms/index.htm#cc)
Design of a simple RISC processor

- First you design datapath and then add control
- All designs (serial, parallel) or otherwise follow a similar methodology

Not covered

- What happens if there are pipeline dependencies e.g. add r1, r2, r3; add r2, r1, r1
- What happens if we branch e.g. add r1, r2, r3; beqz r1 loop
- These are called hazards and can be resolved by stalling (low performance) and to some degree by forwarding (add internal paths to make data available earlier)
- Refer to [1] and [2] for details

Source: Arvind (MIT Opencourseware licence http://ocw.mit.edu/terms/index.htm#cc)