Pipeline Processors
Pipeline Processors

- A common feature of modern processors
- Works like a series production line
- An operation is divided into ‘k’ decoupled (independent) elementary sub-operations
- A ‘k’ stage pipeline is formed
- The pipeline can handle ‘k’ sets of data simultaneously
Pipeline Types

Instruction Pipeline

- Different stages of instruction fetch and execution are handled by the pipeline
- Very common in current processors

Arithmetic Pipeline

- Different stages of an arithmetic operation are handled along the segments of a pipeline
- Highly specialised digital design - uncommon
Pipelining – Laundry Example
(from Dan Conners)

Ann, Brian, Cathy, Dave each have one bag of clothes to wash, dry, fold, put away

- Washer takes 30 minutes
- Dryer takes 30 minutes
- “Folder” takes 30 minutes
- “Put-away-er” takes 30 minutes to put clothes into drawers
Sequential Laundry

- Sequential laundry takes 8 hours for 4 loads
- If they used a pipeline, how long would laundry take?
Pipelined Laundry

- Pipelined laundry takes 3.5 hours for 4 loads!
Pipelining Observations

- Pipelining doesn’t change duration of a single task, but increases throughput
- Multiple tasks execute simultaneously using different (decoupled) resources
- Potential speedup = Number of pipeline stages
- Pipeline rate limited by slowest pipeline stage
- Unbalanced lengths of pipe stages reduces speedup
- Time to “fill” pipeline and time to “flush” it reduces speedup
### Instruction Pipeline

- **Simple example of a (4-stage) instruction pipeline** (this is not a PIC18...)

<table>
<thead>
<tr>
<th>Cycle</th>
<th>F</th>
<th>D</th>
<th>E</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i+1</td>
<td>F</td>
<td>D</td>
<td>E</td>
<td>W</td>
</tr>
<tr>
<td>i+2</td>
<td>F</td>
<td>D</td>
<td>E</td>
<td>W</td>
</tr>
<tr>
<td>i+3</td>
<td>F</td>
<td>D</td>
<td>E</td>
<td>W</td>
</tr>
</tbody>
</table>

- **Ensure each segment completes in one CPU cycle**
  - Usually possible only for RISC

![Clock ticks diagram](image-url)
PIC18Fxxx Instruction Pipeline

- Microchip PIC1 8Fxxx μCs have a 2-stage pipeline

- First stage is *Fetch* – 4 Q cycles, uses instruction bus:
  - Q1: increment PC; Q2: write instruction address; Q3: fetch opcode; Q4: latch instruction

- Second stage is *Execute* – 4 Q cycles, uses data bus:
  Cycle varies, but is typically
  - Q1: decode instruction; Q2: fetch operand; Q3: execute; Q4: write result

- Pentium III has 10-stage pipeline, P4 has 20 stages…
Pipeline Hazards

A pipeline hazard is anything that disrupts orderly flow of data through the pipeline

- Structural Hazards
- Data Hazards
- Control Hazards
Structural Hazards

- Occur when more than one segment of the pipeline needs access to the same hardware resource
  - This is a failure of hardware design modularity
  - e.g. pipeline segments fetching opcode and operand simultaneously both need to use the memory access register

- May be alleviated by duplication of resources
  ⇒ Harvard architecture
Data Hazards

- When the structure of the program causes a pipeline segment to access data before it has been updated by a prior segment
- Generally, where an instruction depends on the result of a prior instruction that is still in the pipeline

- Can be eliminated by
  - Inserting some (wasted) cycles of stall to allow the data to be refreshed
  - Adding specialised hardware for passing a result directly back to the ALU input without storage in the register file
Control Hazards

- When a jump or branch instruction is encountered, subsequent (fetched) instructions must be flushed from the pipeline since they don’t need to execute.

- Can be minimised by:
  - Detecting the branch early in the pipeline
  - Getting the target address into the PC as early as possible
  - Attempting to predict the branch target
  - Using a branch target cache containing first instructions of both possible branches
Superscalar Design

- Number of instructions issued simultaneously for execution – e.g. many modern processors are two-issue superscalar (dual pipelines)
- Instruction cache loads the prefetch unit.
- Prefetch unit issues ‘i’ instructions at a time and forwards them to decoder
- Dispatch unit generates activation commands for a number of pipelined segments (multiple execution units)
- Often used to avoid pipeline hazards by
  - Branch prediction
  - Speculative execution (executing both branches)
For Interest

- 25 Microchips that Shook the World

References