COMP3211 03S2 Lecture 24

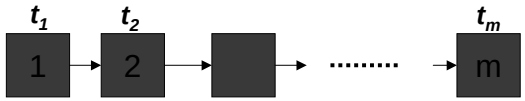
Introduction to Pipelining
(supplementary slides)

Adapted from

CS152: Computer Architecture and Engineering
Dave Patterson (www.cs.berkeley.edu/~pattrsn)

What is "Pipelining"?

- An implementation technique in which multiple instructions (tasks) are overlapped in execution.
- A pipeline consists of several sections, called stages



- Multiple tasks operating simultaneously using different resources (stages)
- Tasks flow from each stage to the next at the same pace. The rate of progress is limited by the slowest pipeline stage – stage time, t_M

Features of Pipelining

- Pipelining doesn't help latency of single task, it helps throughput of entire workload
  - Latency $t_r = m \times t_M$
  - Throughput $\propto 1/(t_r + (n-1) \cdot t_M)$
- Potential speedup = Number pipe stages (m)
- Unbalanced lengths of pipe stages reduces speedup
- Time to "fill" pipeline and time to "drain" it reduces speedup
- Time spent stalled waiting for dependences also reduces speedup

Exercises

Q1. [P&H 6.11] Consider executing the following code on the pipelined datapath of Figure 6.46 on page 492 (P&H's textbook):

```
add  $1,  $2,  $3
add  $4,  $5,  $6
add  $7,  $8,  $9
add  $10, $11, $12
add  $13, $14, $15
```

At the end of the fifth cycle of execution, which registers are being read and which register will be written?

Exercise

Q2. [P&H 6.4] Identify all of the data dependencies in the following code. Which dependencies are data hazards that will be resolved via forwarding?

```
add   $2,   $5,   $4
add   $4,   $2,   $5
sw    $5,   100($2)
add   $3,   $2,   $4
```


Exercise

Q3. [P&H 6.3] How could we modify the following code to make use of a delayed branch slot?

```
LOOP:   lw    $2,   100($3)
        addi  $3,   $3,   4
        beq   $3,   $4,   Loop
```