

To illustrate the principle of pipelining, we will take a look at the dreaded chore: laundry.

1. If they do laundry sequentially (one person in the laundry room at a time),



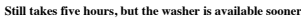
2. If the start their clothes washing as soon as the washer becomes available



3. If the heating element is broken on the dryer and it takes twice as long to dry clothes



4. If the next person does the unthinkable and takes laundry out of the washer and PUTS IT ON TOP OF THE DRYER



**latency** is the time it takes to accomplish one task. In the case of our laundry room, it is the time it takes to wash, dry, fold and stash laundry.

**throughput** is the number of tasks completed per unit of time.

The latency + throughput in our examples above:

1. latency: 1.5 hours average throughput:  $(4/6) = .67$  loads/hour
2. latency: 1.5 hours average throughput:  $(4/3) = 1.33$  loads/hour
3. latency: 2 hours average throughput:  $(4/5) = .8$  loads/hour
4. latency: 2 hours average throughput:  $(4/5) = .8$  loads/hour

**Example:** Consider an unpipelined system that takes 320ps ( $10^{-12}$ s) to complete an instruction. Compute the throughput:

In this examples, the *clock* would cycle every 320ps.

### Key observations

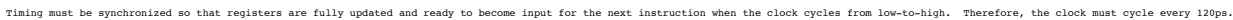
- Pipelining does not change latency, but can improve throughput
- The potential for speedup is limited by the number of unique tasks. We call each unique task a *pipeline stage*
- Overall improvement is limited by the slowest pipeline stage
- Unbalanced stage lengths, reduces speeded
- When there is resource competition, we must stall in order to maintain the pipeline

Lets say that our throughput example from above comes from a system that has three unique computational tasks that each take 100ps to complete and then takes 20ps to update the register file. It might look something like this:

By introducing *pipeline registers* for maintaining state between each unique computational task, we can build a pipelined system with stages A, B and C. We call this a *3-stage pipeline*:

Question:How often does the clock cycle? What is the latency and throughput for this system?

The clock cycles in a wave pattern from high to low:



Latency is the time for one instruction:

$$(100)+(20)+(100)+(20)+(100)+(20) = 360\text{ps}$$

Throughput:

$$\frac{1 \text{ op}}{\text{-----}} * \frac{1000\text{ns}}{\text{-----}} = 8.33 \text{ GOPS}$$

1ps

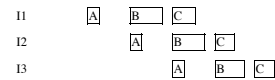
**Example** We have a three stage pipeline. Stage A takes 50ps, Stage B 150ps and Stage C 100ps. Register loading takes 20ps. What is the clock cycle, latency and throughput of such a system?

$$\text{Clock} = (150\text{ps}) + (20\text{ps}) = 170\text{ps}$$

Latency = nClock where n is the number of stages:  
 $(3)(170\text{ps}) = 510\text{ps}$

$$\text{Throughput} = \frac{1 \text{ operation}}{170\text{ps}} * \frac{1000\text{ps}}{1\text{ns}} = 5.88 \text{ GOPS}$$

Instruction	CC 1	CC 2	CC 3	CC 4	CC 5
-------------	------	------	------	------	------



1. Where should we insert a register to create a two-stage pipeline? What would be the throughput and the latency?

if we place the register between C and D, the delays are:

```
stage 1: A+B+C+r = 80+30+60+20 = 190ps
stage 2: D+E+F+r = 50+70+10+20 = 150ps
```

```
latency = n*Clock
    our clock will have to cycle at 190ps, so latency = 2*(190) = 380ps
```

$$\text{Throughput} = \frac{1 \text{ op}}{190\text{ps}} * \frac{1000\text{ps}}{1\text{ns}} = 5.26 \text{ GOPS}$$

2. Where should we insert two registers to create a three-stage pipeline? What would be the throughput and the latency?

if we place our pipeline registers between B&C and between D&E, the delays are:

```
stage 1: A+B+r = 80+30+20 = 130ps
stage 2: C+D+r = 60+50+20 = 130ps
stage 3: E+F = 70+10+20 = 100ps
```

```
latency = n*Clock
    clock will cycle at 130ps, so latency = 3*130 = 390ps
```

$$\text{Throughput} = \frac{1 \text{ op}}{130\text{ps}} * \frac{1000\text{ps}}{1\text{ns}} = 7.69 \text{ GOPS}$$

3. What is the throughput and latency of a 5-stage pipeline implementation?

```
stage 1: A+r = 100ps
stage 2: B+r = 50ps
stage 3: C+r = 80ps
stage 4: D+r = 70ps
stage 5: E+F+r =100ps
```

```
latency = n*Clock
    clock will cycle at 100ps, so latency = 5*100ps = 500ps
```

$$\text{Throughput} = \frac{1 \text{ op}}{100\text{ps}} * \frac{1000\text{ps}}{\text{ns}} = 10.00 \text{ GOPS}$$