

Unit 17

Improving Performance
Caching and Pipelining

Improving Performance

- We want to improve the performance of our computation
- Question: What are we referring to when we say "performance"?
 - _____
 - _____
 - _____
- We will primarily consider _____ in this discussion

How Do We Measure Speed

- **Fundamental Measurement:** _____
 - Absolute time from _____ to _____
 - To compare two alternative systems (HW + SW) and their performance, start a timer when you begin a task and stop it when the task ends
 - Do this for both systems and compare the resulting times
- We call this the _____ of the system and it works great from the perspective of the _____ task
 - If system A completes the task in 2 seconds and system B requires 3 seconds, then system A is clearly superior
- But when we dig deeper and realize that the single, overall task is likely made of _____ small tasks, we can consider more than just latency

Performance Depends on View Point?!

- What's faster to get from point A to point B?
 - A 747 Jumbo Airliner
 - An F-22 supersonic, fighter jet
- If only _____ to get from point A to point B, then the _____
 - This is known as _____ [units of seconds]
 - Time from the start of an operation until it completes
- If _____ to get from point A to point B, the _____ looks much better
 - This is known as _____ [jobs/second]
- The **overall** execution time (latency) may best be improved by _____ throughput and not the latency of individual tasks

Improving Latency and Throughput

CACHING AND PIPELINING

Hardware Techniques

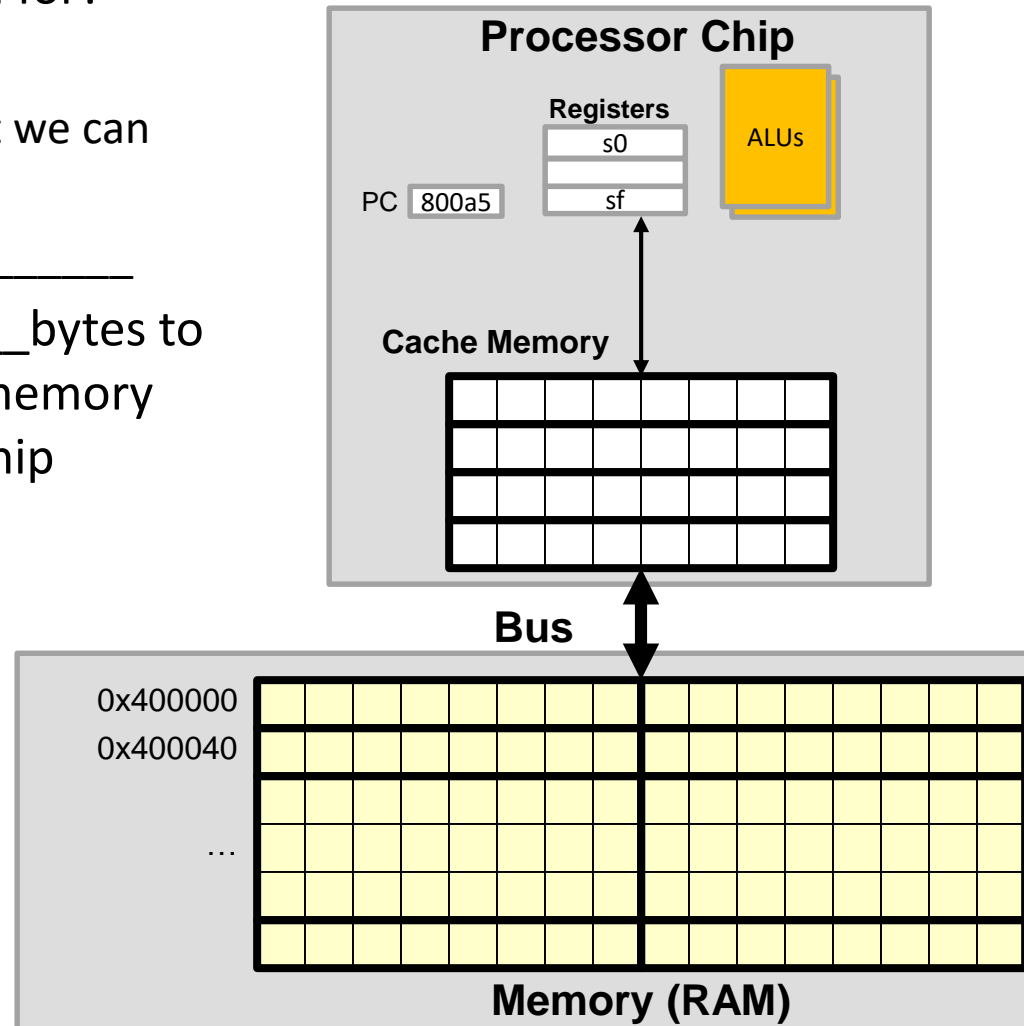
- We can add hardware or reorganize our hardware to improve throughput and latency of individual tasks in an effort to reduce the total latency (time) to finish the overall task
- We will look at two examples:
 - Caching: Improves _____
 - Pipelining: Improves _____

Caching

- **Cache (def.)** – *"to store away in hiding or for future use"*
- Primary idea
 - The _____ you access or use something you expend the _____ amount of time to get it
 - However, store it someplace (i.e. in a cache) you can get it **more _____ the next time** you need it
 - The next time you need something check if it is in the cache first
 - **If it is in the cache, you can get it quickly; else go get it expending the full amount of time** (but then _____ it in the cache)
- Examples:
 - _____
 - _____
 - _____

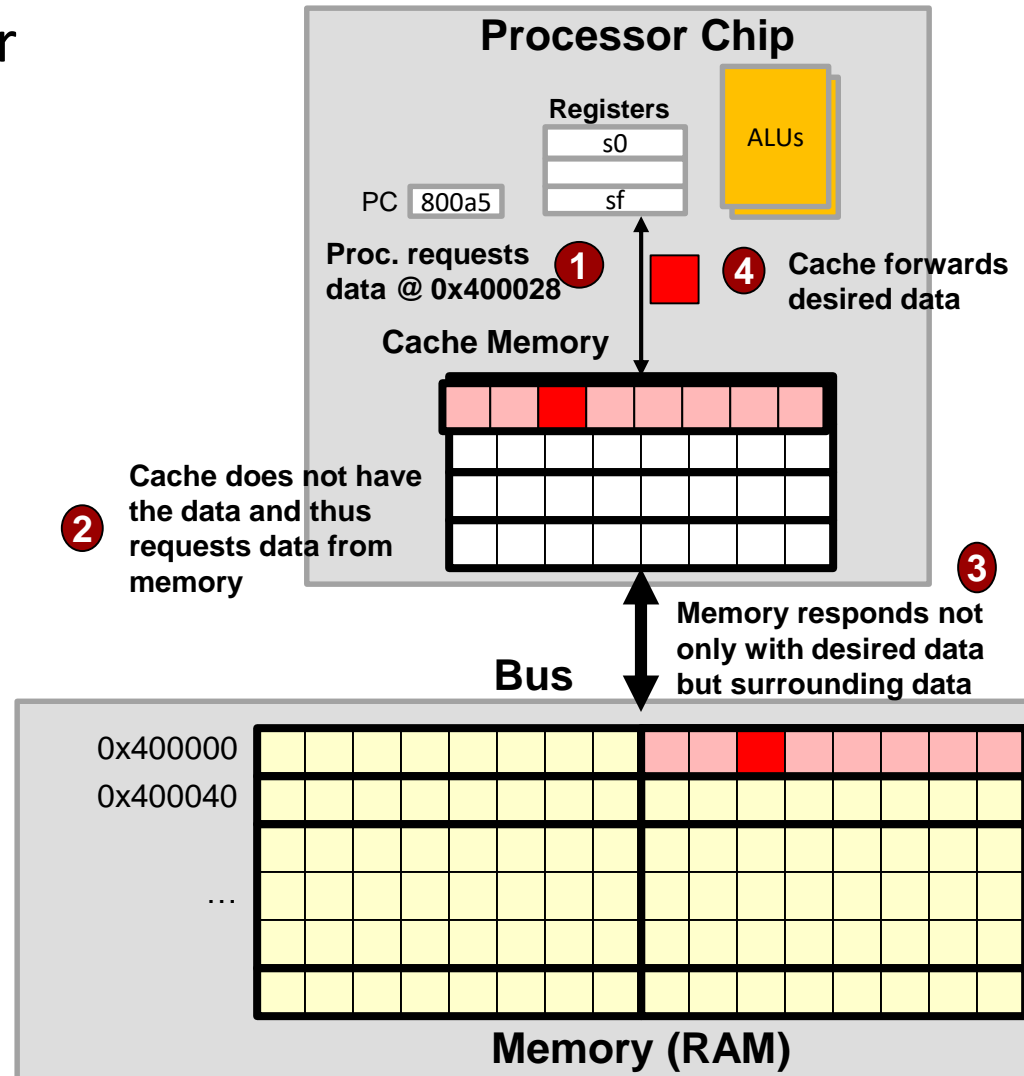
Cache Overview

- Remember what register are used for?
 - Quick access to copies of data
 - Only a _____ (32 or 64) so that we can access really quickly
 - Controlled by the _____
- Cache memory is a **small-ish**, (_____ bytes to a few _____ bytes) "_____ " memory usually built onto the processor chip
- Will hold _____ of the latest data & instructions accessed by the processor
- Managed by the _____
 - _____ to the software



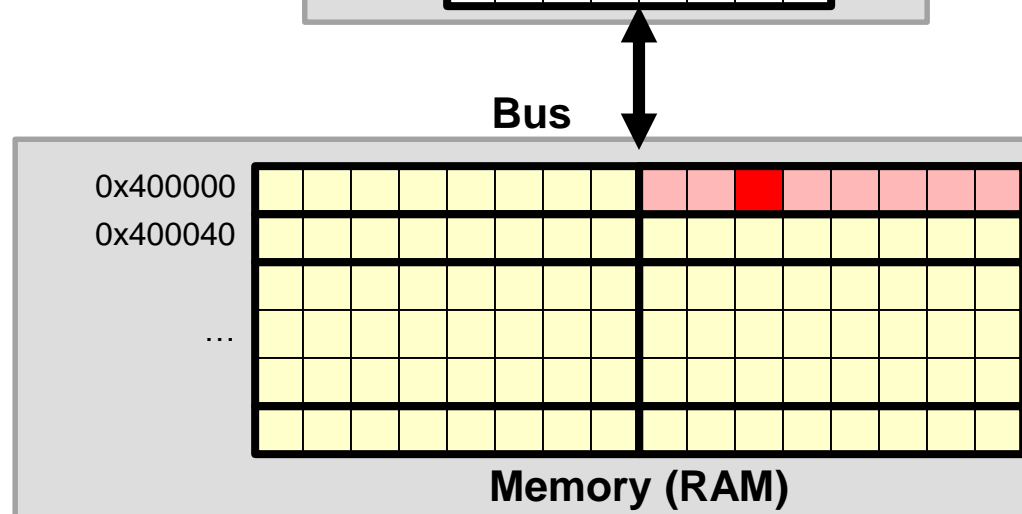
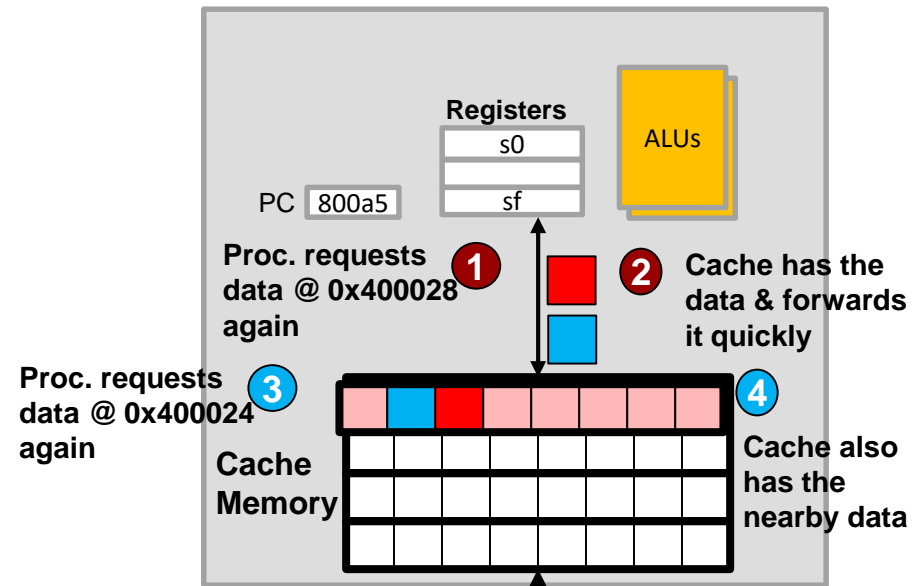
Cache Operation (1)

- When processor wants data or instructions it always _____ in the cache first
- If it is there, _____ access
- If not, get it from _____
- Memory will also supply _____ data since it is likely to be needed soon
 - Why?
 - Things like _____ & _____ (instructions) are commonly accessed sequentially



Cache Operation (2)

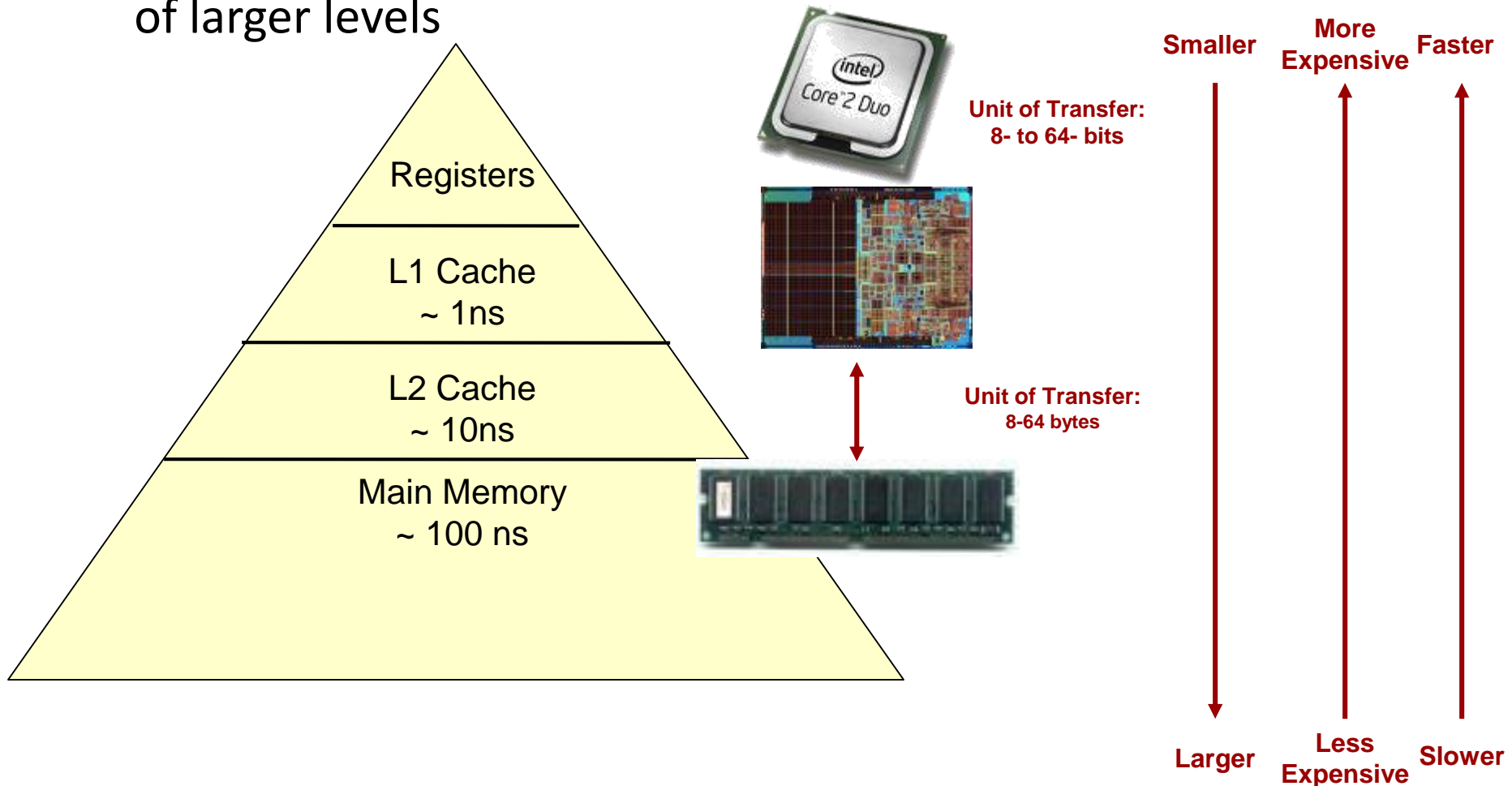
- When processor asks for the data again or for the next data value in the array (or instruction of the code) the cache will likely have it
- Questions?



Main point: Caching reduces the latency of memory accesses which improves overall program performance.

Memory Hierarchy & Caching

- Use several levels of faster and faster memory to hide _____ of larger levels



Pipelining

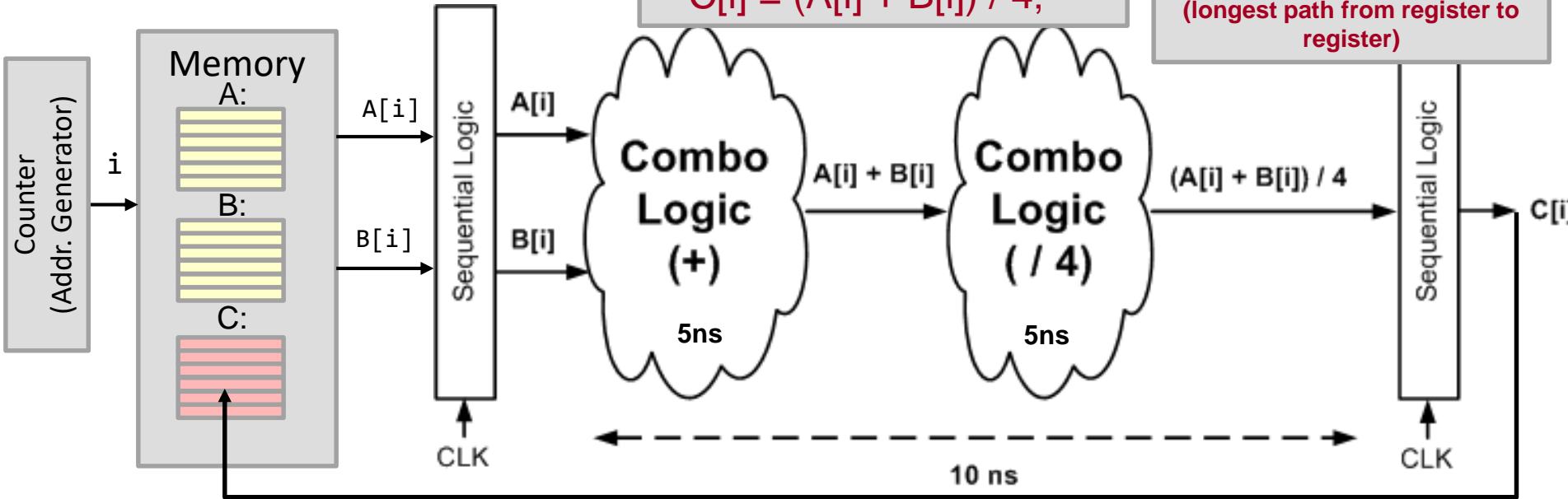
- We'll now look at a hardware technique called **pipelining** to improve _____
- The key idea is to _____ the processing of multiple "items" (either data or instructions)

Example

- Suppose you are asked to build dedicated hardware to perform some operation on all 100 elements of some arrays
- Suppose the operation $(A[i]+B[i])/4$ takes 10 ns to perform
- How long would it take to process the entire arrays: _____ ns
 - Can we improve?

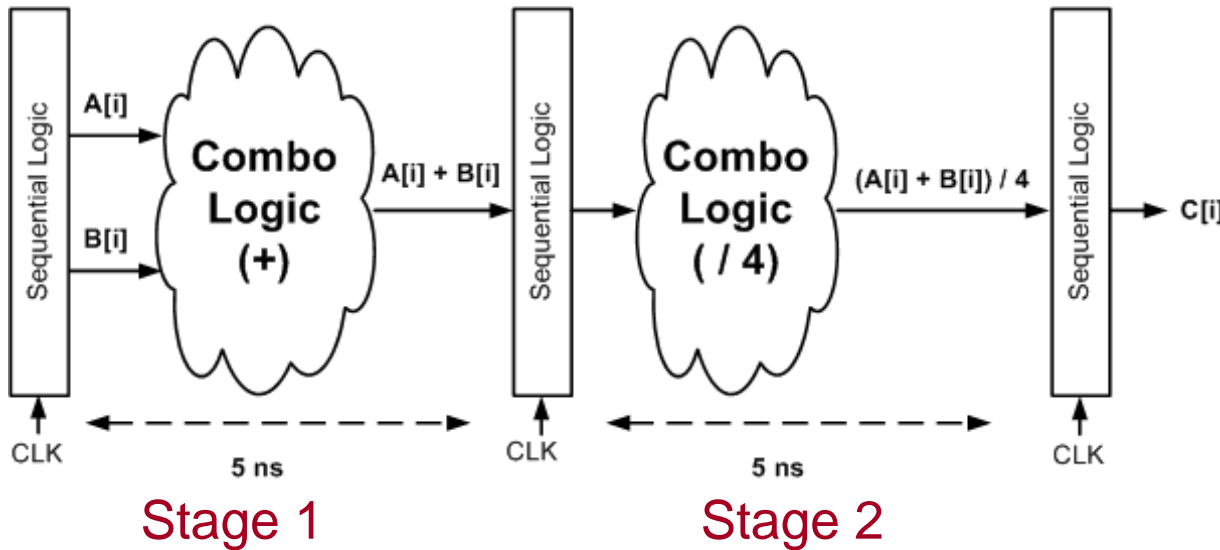
```
for(i=0; i < 100; i++)
    C[i] = (A[i] + B[i]) / 4;
```

Clock Freq. = $1/10\text{ ns} =$ _____ MHz
(longest path from register to register)



Pipelining Example

- Pipelining refers to insertion of registers to split combinational logic into smaller stages that can be overlapped in time (i.e. create an assembly line)



```
for(i=0; i < 100; i++)
    C[i] = (A[i] + B[i]) / 4;
```

Time for 0th elements to complete: _____

Time between each of the remaining 99 element completing: _____

Total: _____

Define:

$$speedup = \frac{Original\ time}{Improved\ Time}$$

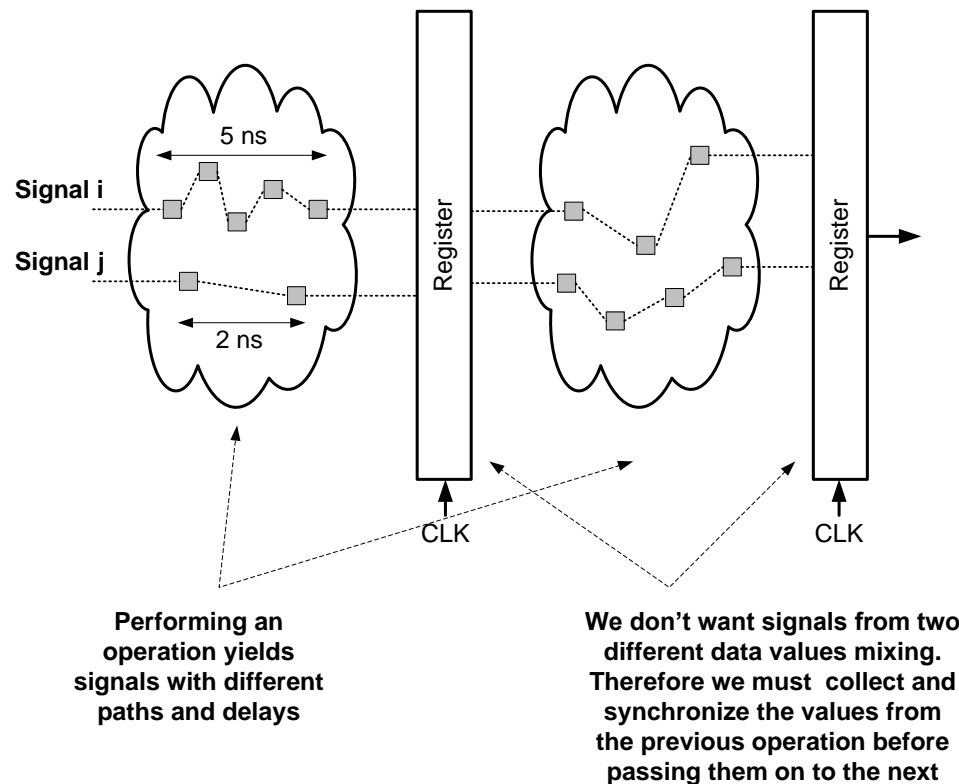
$$speedup = \frac{1000ns}{\quad ns} \approx \quad$$

Clock freq: = _____

	Stage 1	Stage 2
Clock Cycle 0	A[0] + B[0]	
Clock Cycle 1	A[1] + B[1]	(A[0] + B[0]) / 4
Clock Cycle 2	A[2] + B[2]	(A[1] + B[1]) / 4

Need for Registers

- Provides separation between combinational functions
 - Without registers, fast signals could “catch-up” to data values in the next operation stage



Pipelining Example

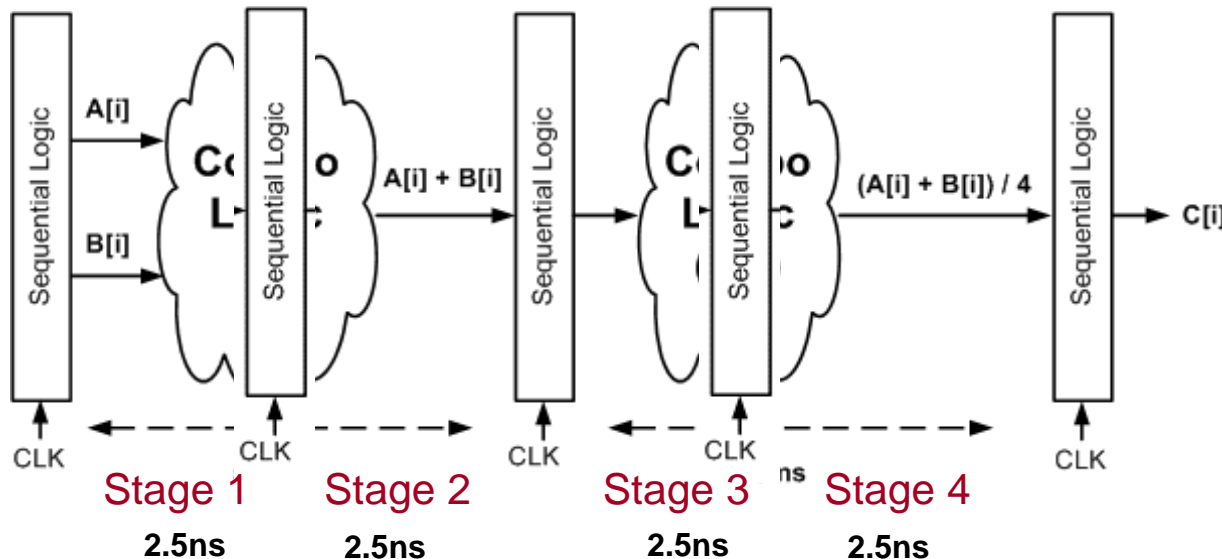
- By adding more pipelined stages we can improve throughput
- Have we affected the latency of processing individual elements? _____
- Questions/Issues?
 - _____ stage delays
 - _____ of registers (Not free to split stages)
 - This limits how much we can split our logic

```
for(i=0; i < 100; i++)
    C[i] = (A[i] + B[i]) / 4;
```

Time for 0th elements to complete: _____

Time between each of the remaining 99 element completing: _____

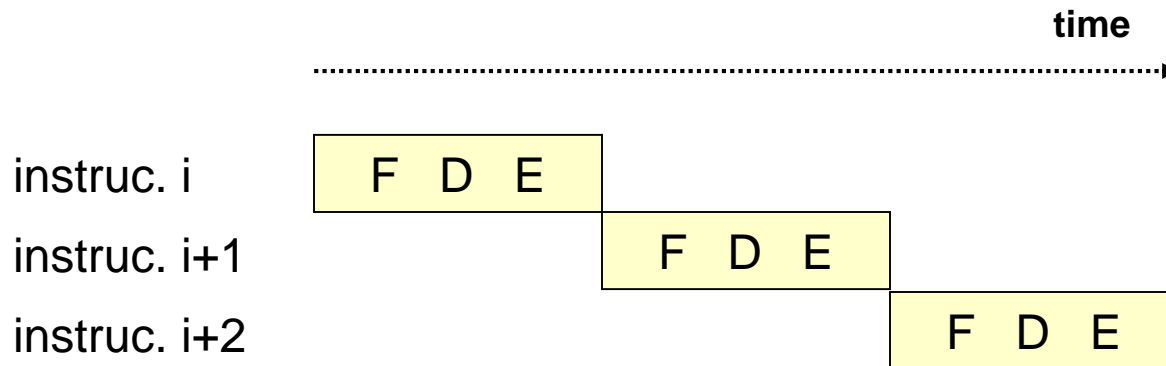
Total: _____

$$speedup = \frac{1000ns}{257.5ns} \approx 4x$$


Non-Pipelined Processors

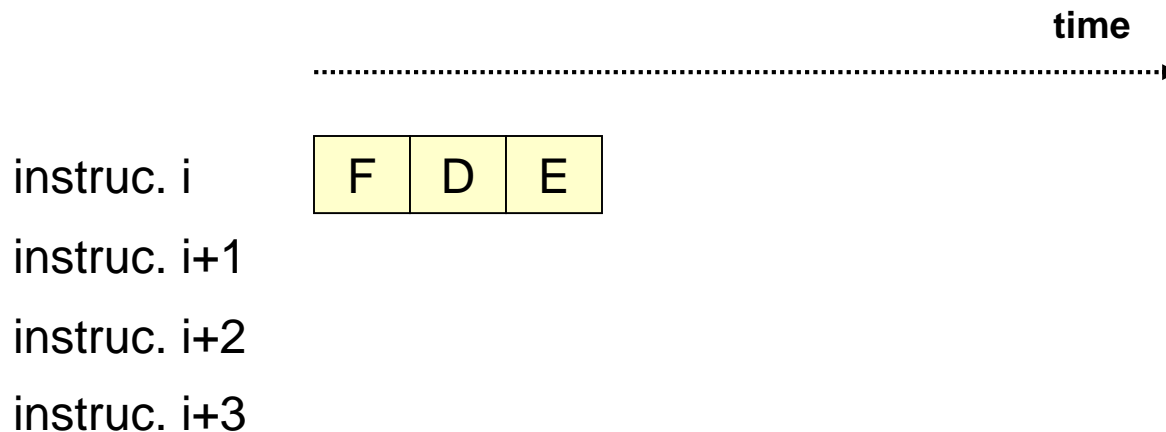
- Currently we know our processors execute software 1 instruction at a time
- 3 steps/stages of work for each instruction are:

- _____
- _____
- _____



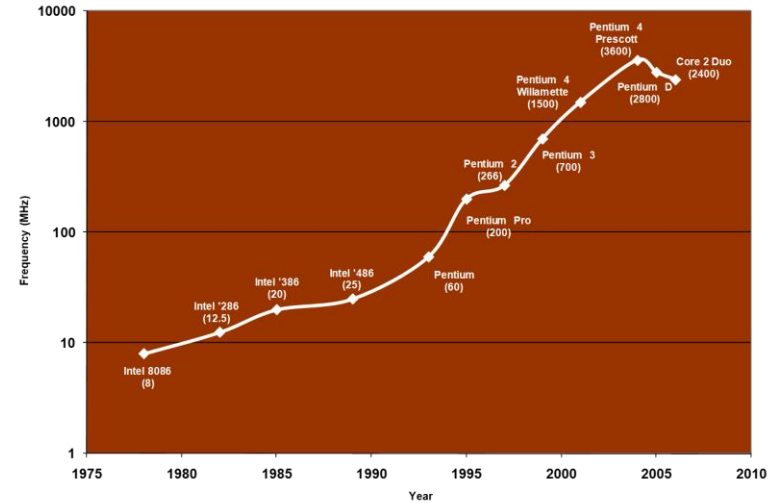
Pipelined Processors

- By breaking our processor hardware for instruction execution into stages we can overlap these stages of work
- Latency for a single instruction is the _____
- Overall throughput, and thus total latency, are greatly improved

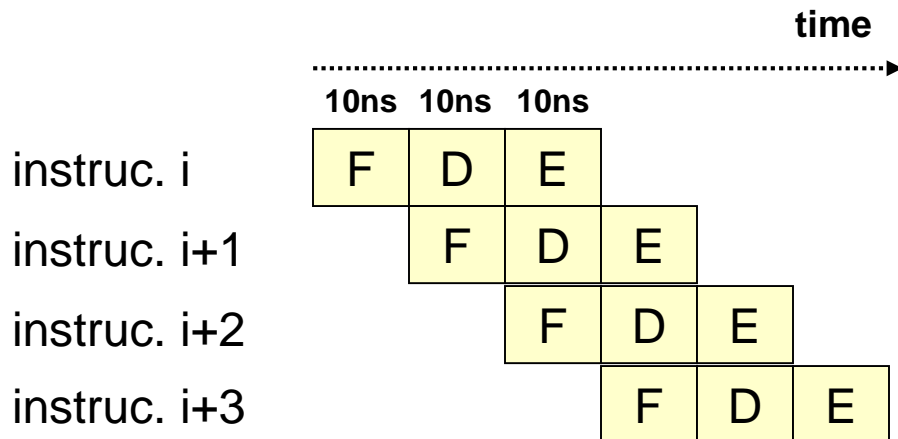


More and More Stages

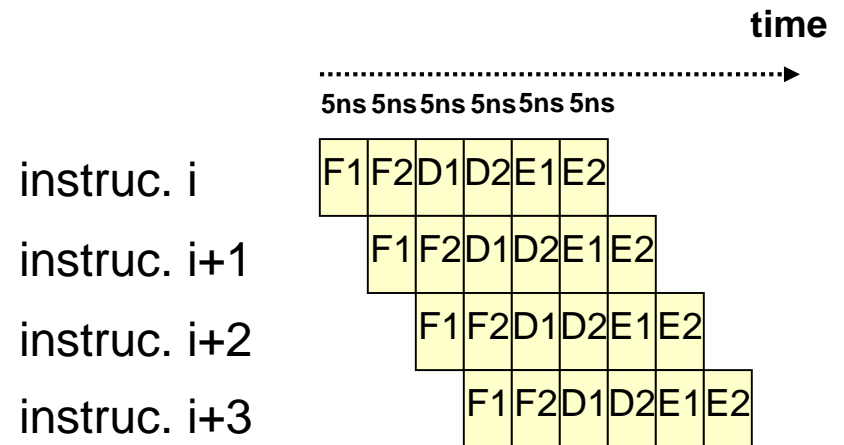
- We can break the basic stages of work into substages to get better performance
- **In doing so our clock period goes _____; frequency goes _____**
- All kinds of interesting issues come up though when we overlap instructions and are discussed in future CENG courses



Clock freq. = $1/10\text{ns} = 100\text{MHz}$



Clock freq. = $1/_ \text{ns} = _ \text{MHz}$



Summary

- By investing extra hardware we can improve the overall latency of computation
- Measures of performance:
 - **Latency** is start to finish time
 - **Throughput** is tasks completed per unit time (measure of parallelism)
- Caching reduces **latency** by holding data we will use in the future in quickly accessible memory
- Pipelining improves **throughput** by overlapping processing of multiple items (i.e. an assembly line)