



Parallel Hardware?

Lecture Objectives:

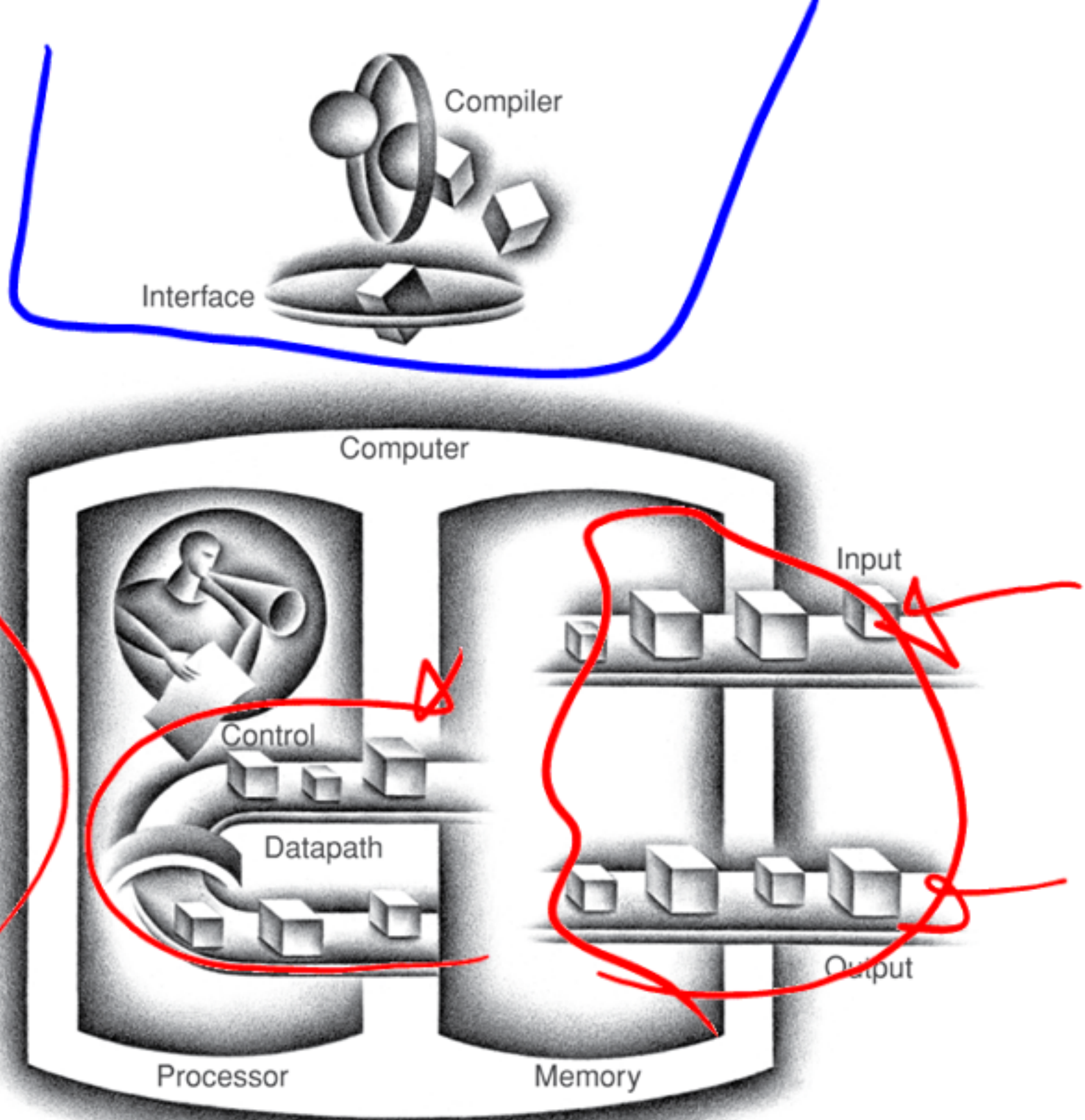
- 1) Draw a picture of the von Neumann computer architecture
- 2) Explain the concept of caching
- 3) Define spatial locality and temporal locality
- 4) List and define the categories of processing under Flynn's taxonomy

Computer Organization in

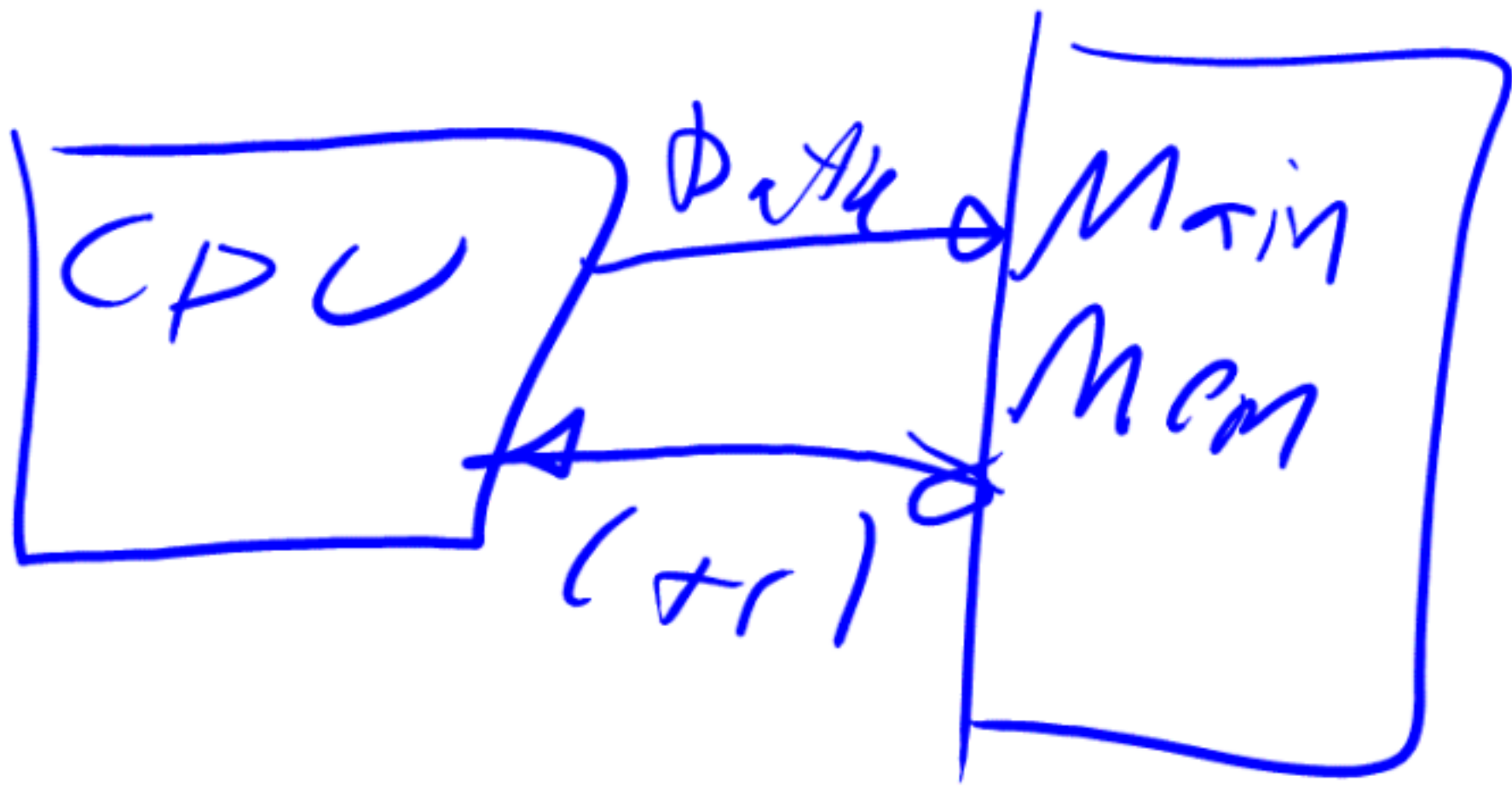
one slide



Evaluating performance

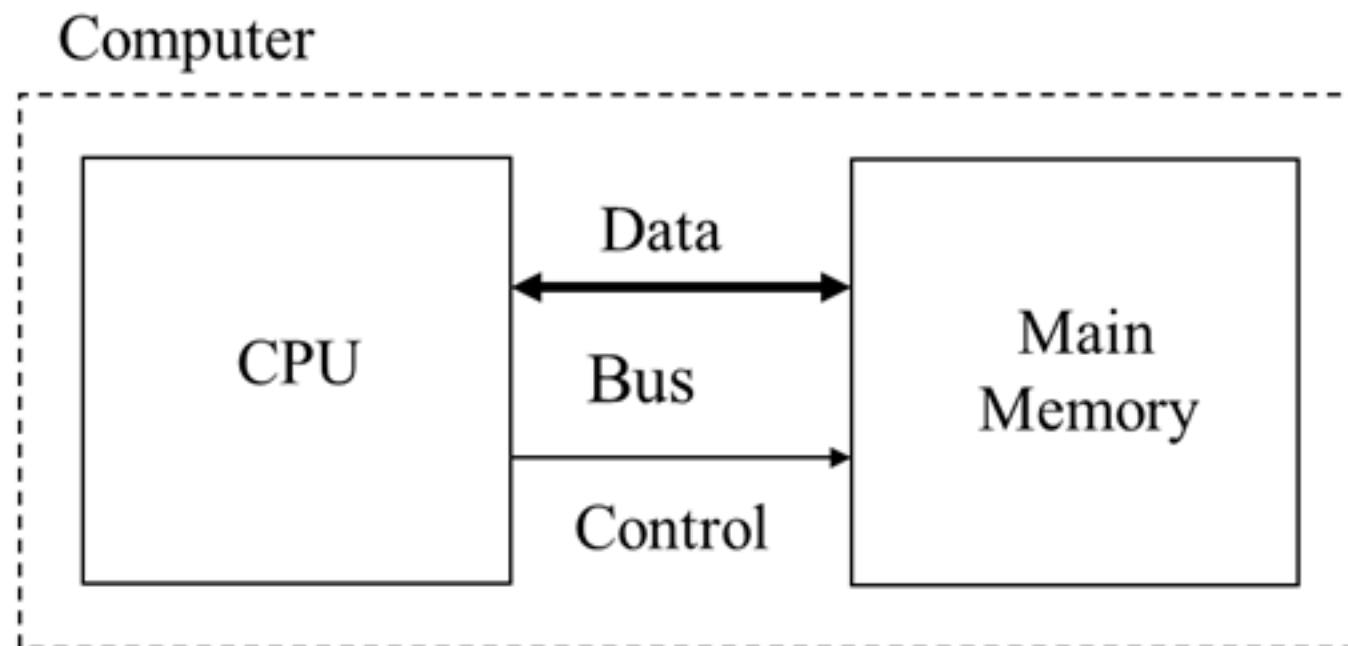


Von Neumann Architecture



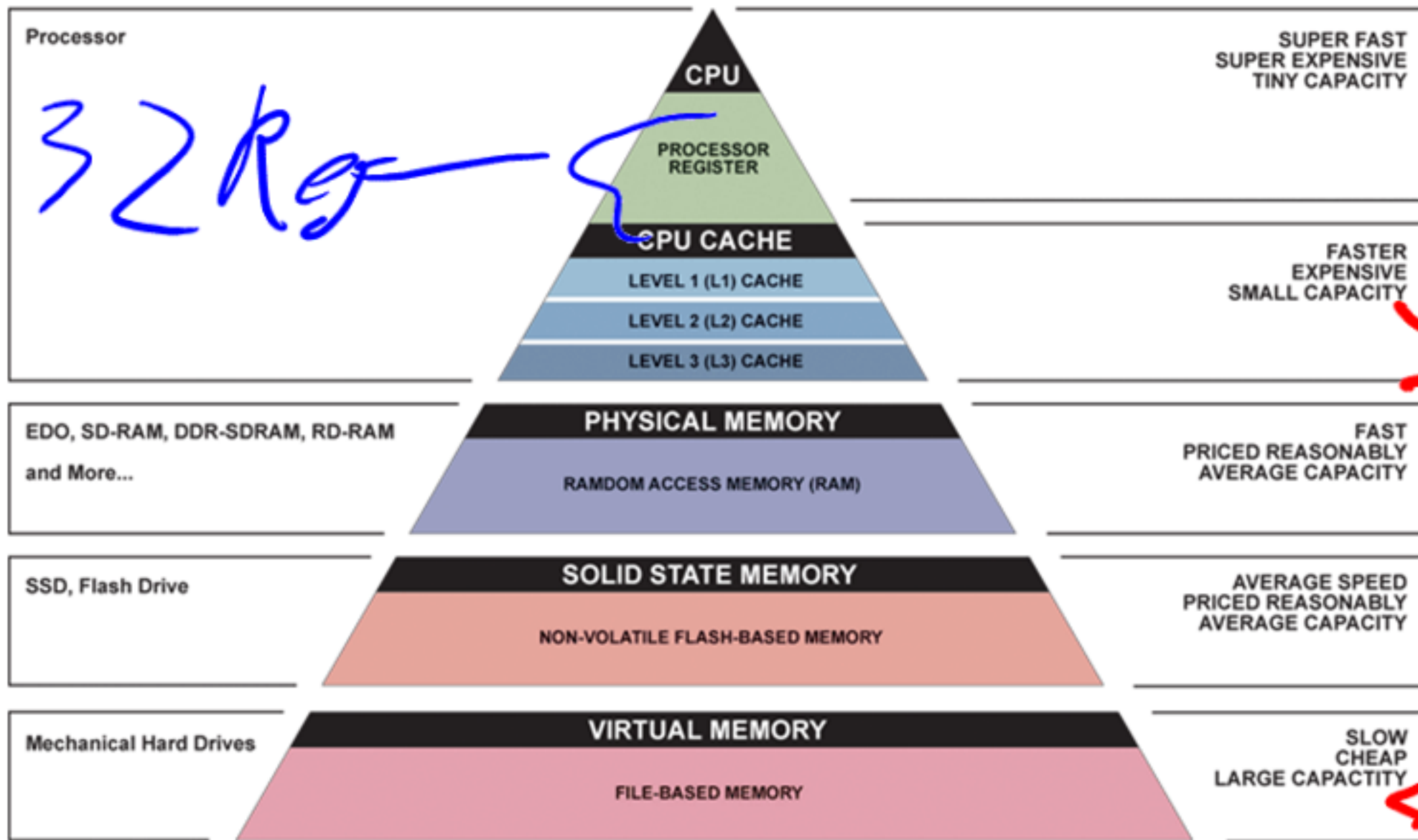
Von Neumann Architecture

- Principles
 - Data and instructions are both stored in the main memory(stored program concept)
 - The content of the memory is addressable by location (without regard to what is stored in that location)
 - Instructions are executed sequentially unless the order is explicitly modified
 - The basic architecture of the computer consists of:



How do we improve this
problem?

The Memory Hierarchy



▲ Simplified Computer Memory Hierarchy
Illustration: Ryan J. Leng



In the grocery store,
which list will be
faster to purchase?

- | | | |
|--------------------------|-------------------------------------|--------------------------|
| 1. Tomatoes | Handwritten red bracket labeled 'S' | 1. Cherry Jelly |
| 2. Corn | | 2. Corn |
| 3. Beans | | 3. Ice Cream |
| 4. Carrots | | 4. Rib Eye Steaks |
| 5. Bread | | 5. Beans |
| 6. Peanut Butter | Handwritten red bracket labeled 'S' | 6. Mozzarella Cheese |
| 7. Cherry Jelly | | 7. Carrots |
| 8. Fresh Chicken Breasts | | 8. Bread |
| 9. Rib Eye steaks | Handwritten red bracket labeled 'M' | 9. Fresh Chicken Breasts |
| 10. Spaghetti Sauce | | 10. Spaghetti Sauce |
| 11. Pasta Noodles | Handwritten red bracket labeled 'P' | 11. Milk |
| 12. Milk | | 12. Tomatoes |
| 13. Yogurt | | 13. Peanut Butter |
| 14. Mozzarella Cheese | | 14. Yogurt |
| 15. Ice Cream | | 15. Pasta Noodles |
| 16. Cherry Lemon Sherbet | | 16. Cherry Lemon Sherbet |



Lets assume you could
obtain anything from the
same aisle in 0 time, but to
change aisles took 10 units
of time.

1. Tomatoes
2. Corn
3. Beans
4. Carrots
5. Bread
6. Peanut Butter
7. Cherry Jelly
8. Fresh Chicken Breasts
9. Rib Eye steaks
10. Spaghetti Sauce
11. Pasta Noodles
12. Milk
13. Yogurt
14. Mozzarella Cheese
15. Ice Cream
16. Cherry Lemon Sherbet

10

10

10

10

10



Lets assume you could
obtain anything from the
same aisle in 0 time, but to
change aisles took 10 units
of time.

1. Cherry Jelly
2. Corn
3. Ice Cream
4. Rib Eye Steaks
5. Beans
6. Mozzarella Cheese
7. Carrots
8. Bread
9. Fresh Chicken Breasts
10. Spaghetti Sauce
11. Milk
12. Tomatoes
13. Peanut Butter
14. Yogurt
15. Pasta Noodles
16. Cherry Lemon Sherbet

22160

- Temporal Locality
 - The principle stating that if a data location is referenced then it will again be referenced soon.
- Spatial Locality
 - The locality principle stating that if a data location is referenced, data locations with nearby addresses will tend to be referenced soon.



Lets assume now that you
are shopping with your
sister. Would the shopping
go faster?

1. Tomatoes
2. Corn
3. Beans
4. Carrots
5. Bread
6. Peanut Butter
7. Cherry Jelly
8. Fresh Chicken Breasts
9. Rib Eye steaks
10. Spaghetti Sauce
11. Pasta Noodles
12. Milk
13. Yogurt
14. Mozzarella Cheese
15. Ice Cream
16. Cherry Lemon Sherbet



Instruction Level Parallelism

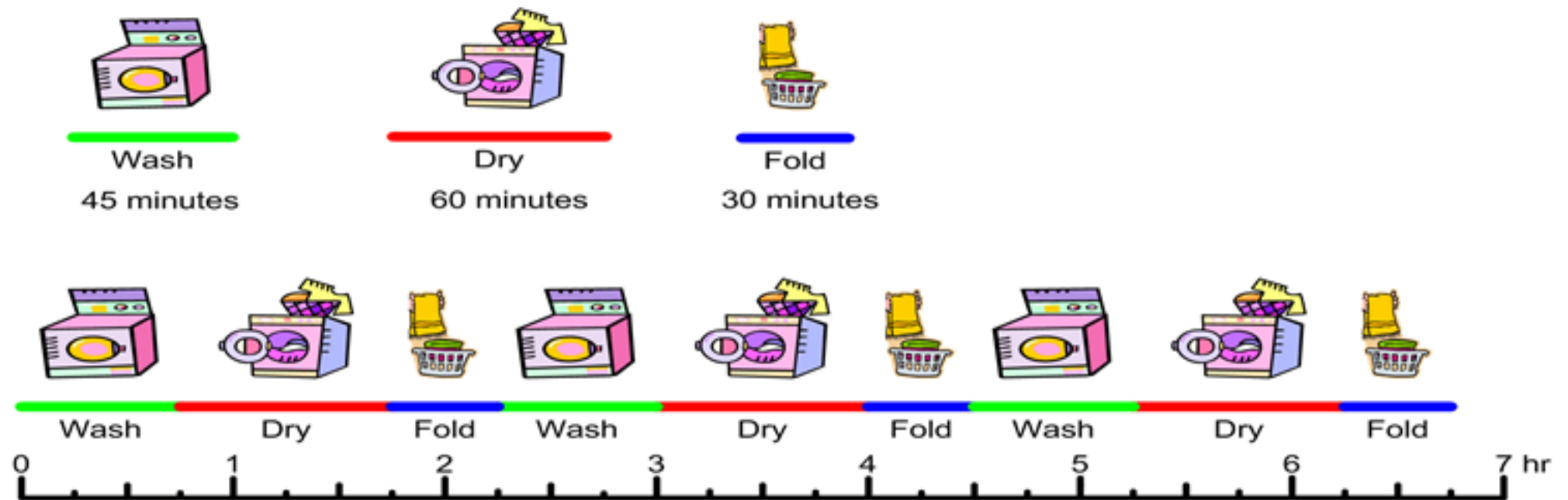
- Attempts to improve processor performance by having multiple processor components (or functional units) simultaneously executing instructions
 - Pipelining
 - Multiple issue

Pipelining Example

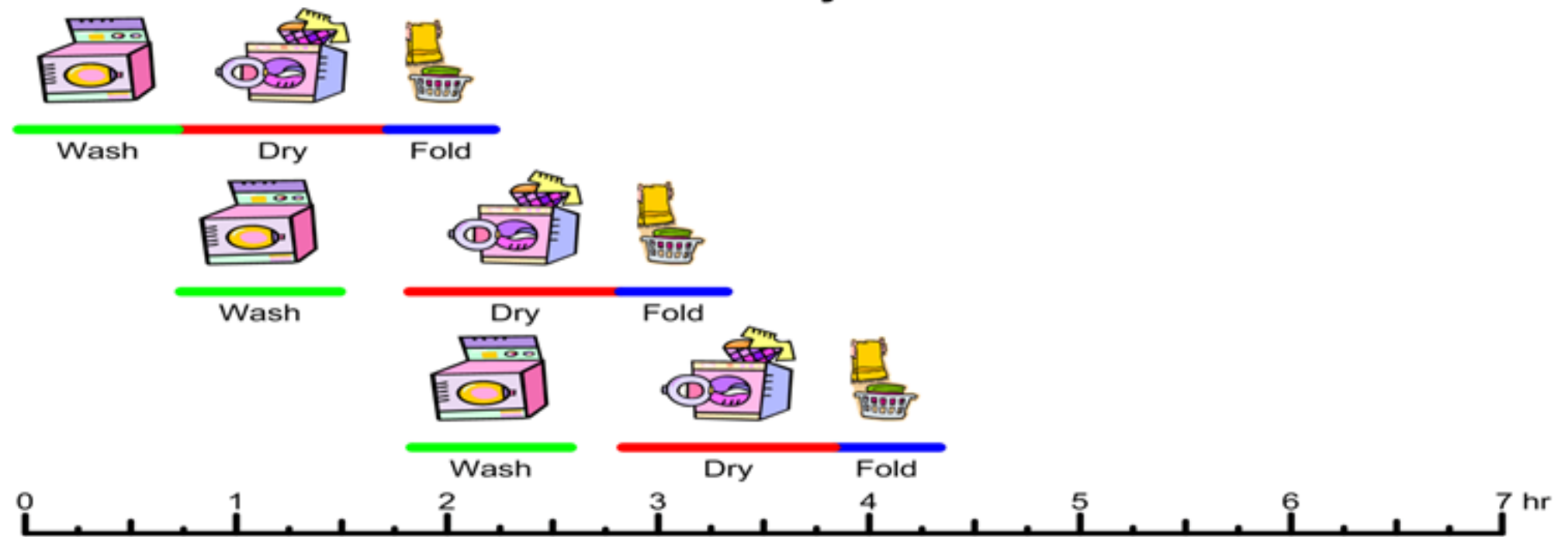
- Lets assume we are sending out a mass mailing by US Postal service and we have multiple people helping. How could we organize the process?
 - Steps



Laundry Pipelining



**One-Task-at-a-time:
3 loads of laundry = 6 ¾ hrs**

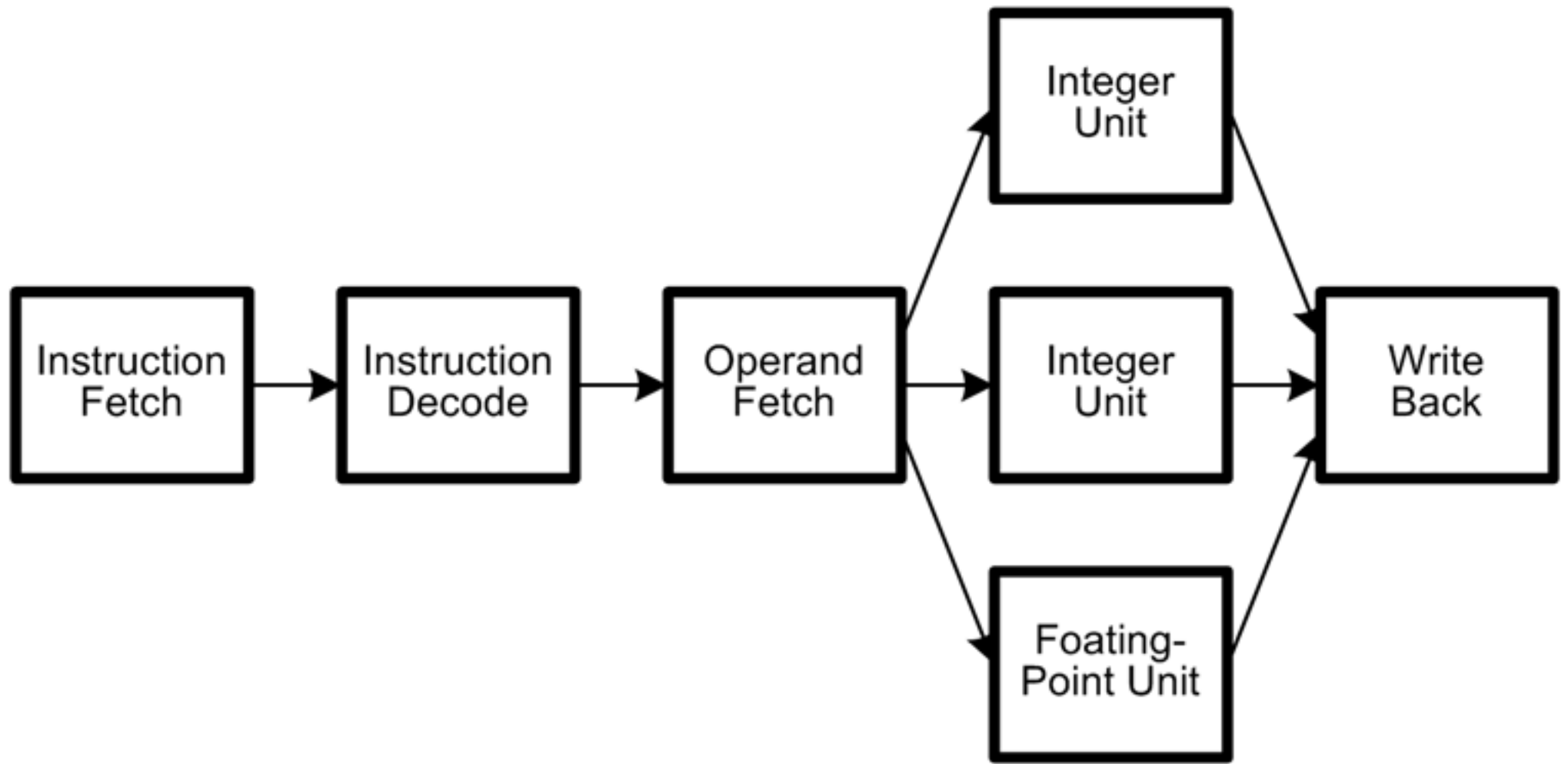


**Overlap Tasks:
3 loads of laundry = 4 ¼ hrs**

*<http://mail.humber.ca/~paul.michaud/Pipeline.htm>



Pipelining in the real world



*<http://mail.humber.ca/~paul.michaud/Pipeline.htm>

Multiple Issue

- Replicate functional units and attempt to execute different instructions in a program in tandem

- Example

```
for (int index = 0; index < 1000; index++)
```

```
{
```

```
    sum[index] = x[index]+y[index];
```

```
}
```

idx++ = 2
sum[index+1] = x[index+1] + y[index+1],

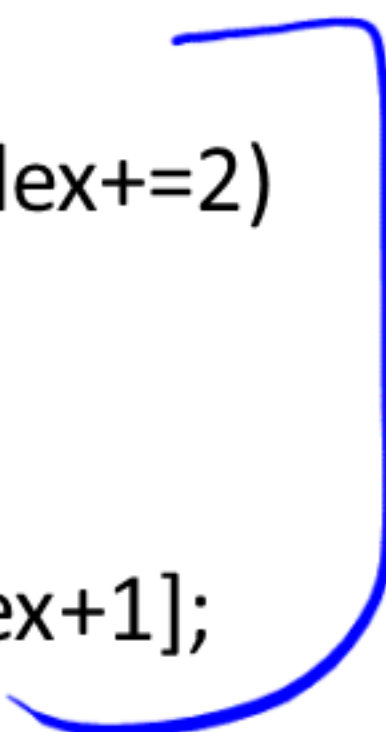
A

B

Multiple Issue

- Replicate functional units and attempt to execute different instructions in a program in tandem
- Example

```
for (int index = 0; index < 1000; index+=2)
{
    sum[index] = x[index]+y[index];
    sum[index+1] = x[index+1]+y[index+1];
}
```



Multiple Issue (2)

- **static** multiple issue - functional units are scheduled at compile time.
- **dynamic** multiple issue – functional units are scheduled at run-time.

↑
superscalar

Multiple issue

- Static
 - Multiple issue functional units are scheduled at compile time by the compiler
- Dynamic
 - Multiple issue functional units are scheduled at run-time

Flynn's Taxonomy

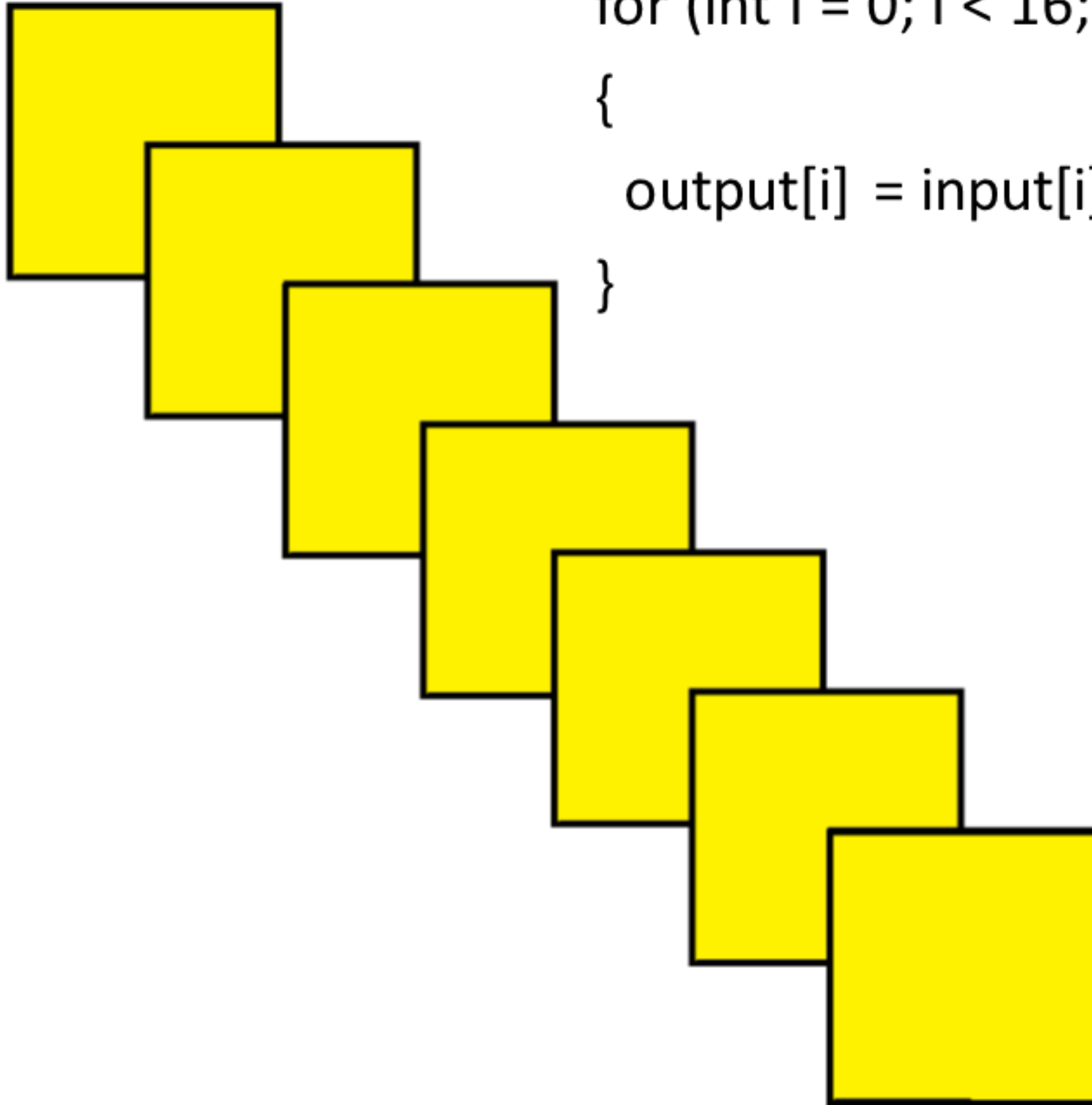
- **SISD**
 - Single Instruction, Single Data. This is the traditional single processor computer.
- **SIMD**
 - Single Instruction, Multiple Data. This scheme, where a single instruction stream is executed by multiple ALUs
- **MISD**
 - Multiple Instruction, Single Data.
- **MIMD**
 - Multiple Instruction, Multiple Data. This is currently The most popular hardware design for parallel processors.

Flynn's Taxonomy

- **SISD**
 - Single Instruction, Single Data. This is the traditional single processor computer.
- **SIMD**
 - Single Instruction, Multiple Data. This scheme, where a single instruction stream is executed by multiple ALUs
- **MISD**
 - Multiple Instruction, Single Data.
- **MIMD**
 - Multiple Instruction, Multiple Data. This is currently The most popular hardware design for parallel processors.



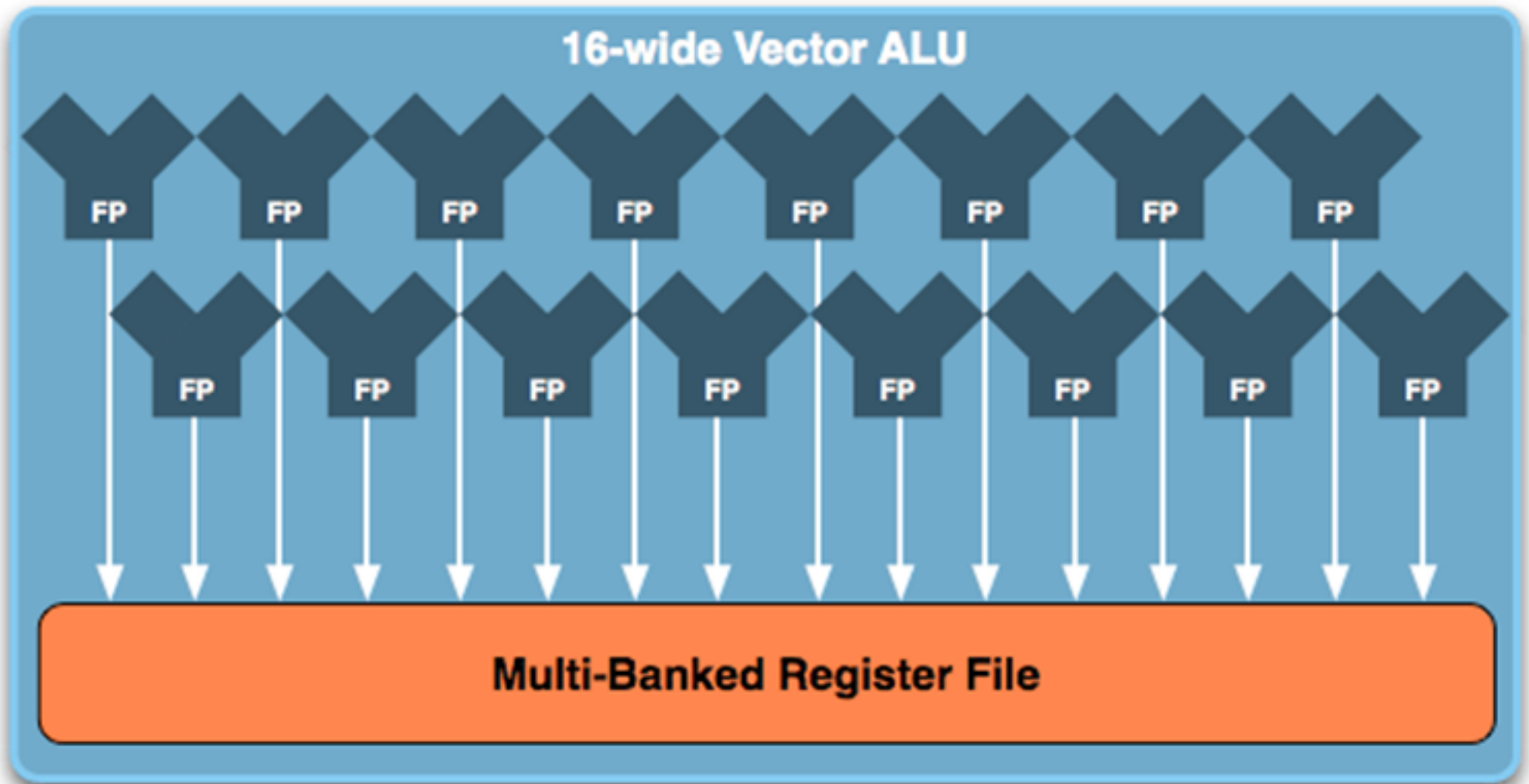
Vector Processor



```
for (int i = 0; i < 16; i++)  
{  
    output[i] = input[i]*2;  
}
```

Vector Processor ALU

structure



[*http://www.anandtech.com/show/6017/intel-announces-xeon-phi-family-of-coprocessors-mic-goes-retail](http://www.anandtech.com/show/6017/intel-announces-xeon-phi-family-of-coprocessors-mic-goes-retail)

SE498 Parallel Computing

