

---

# Multiprocessor Exercises part2

1401215-3 - Computer Architecture

Umm Al-Qura University

Computer Science Department

Fall 2012

## True/False

---

1. Like SMPs, message-passing computers rely on locks for memory access synchronization.
2. Unlike SMPs, message –passing computers need multiple copies of the parallel processing program and the operating system.
3. Both multithreading and multicore rely on parallelism to get more efficiency from a chip.
4. Multithreading uses multiple threads to improve resource utilization.

## Example 1

On a CC-NUMA system, the cost of accessing non-local memory can limit our ability to utilize multiprocessing effectively. The following table shows the costs associated with access data in local memory versus non-local memory and the locality of our application expresses as the proportion of accesses that are local.

Local load/store (cycles)	Non-local load/store (cycles)	%local accesses
25	200	20

Assume that memory accesses are evenly distributed through the application. Also, assume that only a single memory operation can be active during any cycle. State all assumptions about the ordering of local versus non-local memory operations

## Example 1 – part a

- a. If on average we need to access memory once every 75 cycles, what is the impact on our application?

### Answer

$$\text{percentage of load / store} = \frac{1}{75}$$

$$\text{percentage of local access} = 0.2$$

$$\Rightarrow \text{percentage of non local access} = 0.8$$

$$\Rightarrow \text{extra stall cycles per cycle} = \frac{1}{75} (0.2 * 25 + 0.8 * 200)$$

$$= \frac{165}{75} \text{ stall cycles per cycle}$$

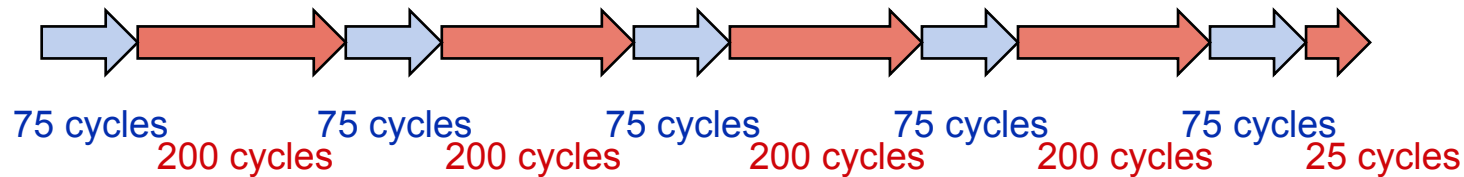
$\Rightarrow$  On average, for every 75 cycles there will be additional 165 stall cycles.

## Example 1 – part a- cont.

To make the answer more visible:

Local accesses are 20% = 0.2 = 1/5

=> for every 5 memory accesses one is local and 4 are non-local



=> *stall cycles percentage* =

$$\frac{200 * 4 + 25}{75 * 5 + (200 * 4 + 25)} = \frac{825}{375 + 825} = \frac{825}{1200}$$

=> Out of every 1200 cycles, 825 cycles are stall cycles.

## Example 1 – part b

- b. If on average we need to access memory once every 50 cycles, what is the impact on our application?

### Answer

$$\text{percentage of load / store} = \frac{1}{50}$$

$$\text{percentage of local access} = 0.2$$

$$\Rightarrow \text{percentage of non local access} = 0.8$$

$$\Rightarrow \text{extra stall cycles per cycle} = \frac{1}{50} (0.2 * 25 + 0.8 * 200)$$

$$= \frac{165}{50} \text{ stall cycles per cycle}$$

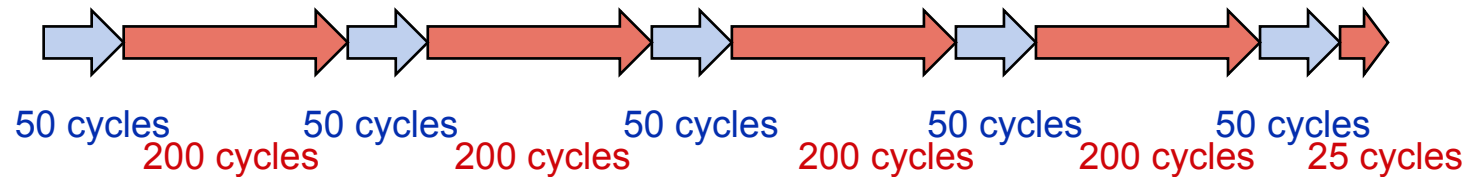
$\Rightarrow$  On average, for every 50 cycles there will be additional 165 stall cycles.

## Example 1 – part b - cont

To make the answer more visible:

Local accesses are 20% = 0.2 = 1/5

=> for every 5 memory accesses one is local and 4 are non local



=> *stall cycles percentage* =

$$\frac{200 * 4 + 25}{50 * 5 + (200 * 4 + 25)} = \frac{825}{250 + 825} = \frac{825}{1075}$$

=> Out of every 1075 cycles, 825 cycles are stall cycles.

## Example 2

Consider the following three CPU organizations:

- (a) CPU SS: 2-core superscalar microprocessor that provides out-of-order issue capabilities on 2 function units (FUs). Only a single thread can run on each core at a time.
- (b) CPU FGMT: A fine-grained multithreaded processor that allows instructions from 2 threads to be run concurrently (i.e. there are two functional units), though only instructions from a single thread can be issued on any cycle.
- (c) CPU SMT: AN SMT processor that allows instructions from 2 threads to be run concurrently (i.e. there are two functional units), and instructions from either or both threads can be issued to run on any cycle.



## Example 2 – cont.

Assume we have two threads X and Y to run on these CPUs that include the following operations:

### Thread X:

A1- takes 3 cycles to execute

A2- no dependencies

A3 – conflicts for a functional unit with A1

A4 – depends on the result of A3

### Thread Y:

B1 – takes 2 cycles to execute

B2 – conflicts for a functional unit with B1

B3 – depends on the result of B2

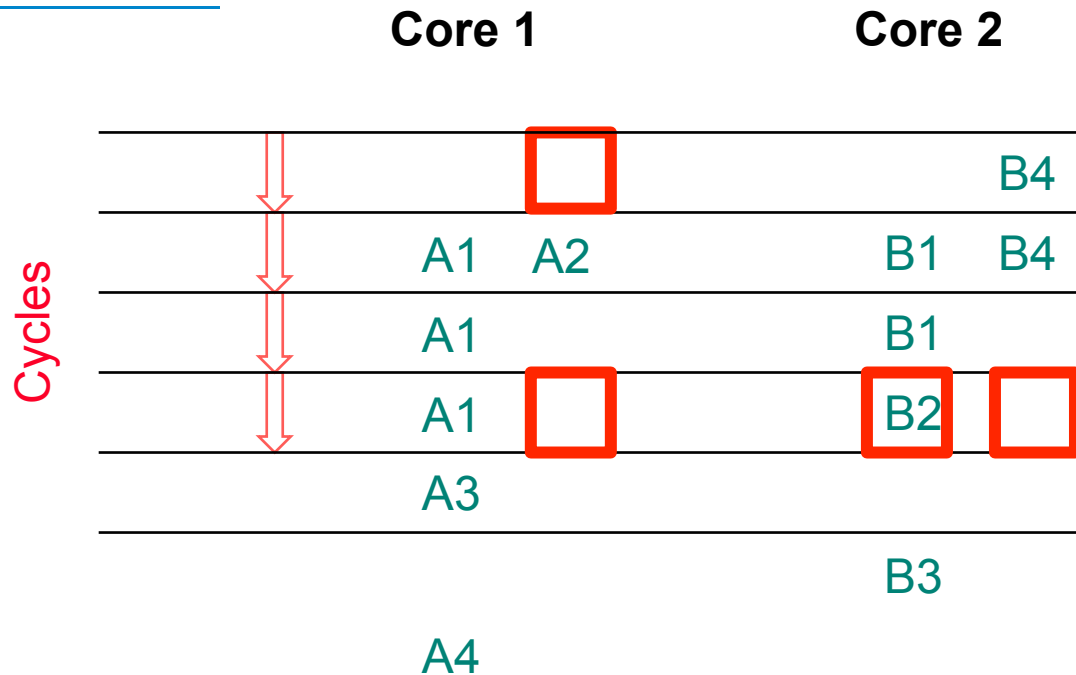
B4 – no dependencies and takes 2 cycles to execute

Assume all instructions take a single cycle to execute unless noted otherwise or they encounter a hazard. Assume also that you have to leave one cycle between two instructions if they depend on each other.

## Example 2 – part I

Assume that you have 1 SS CPU (as described in (a)).  
How many cycles will it take to execute threads X and Y? How many issue slots are wasted due to hazards.

### Answer

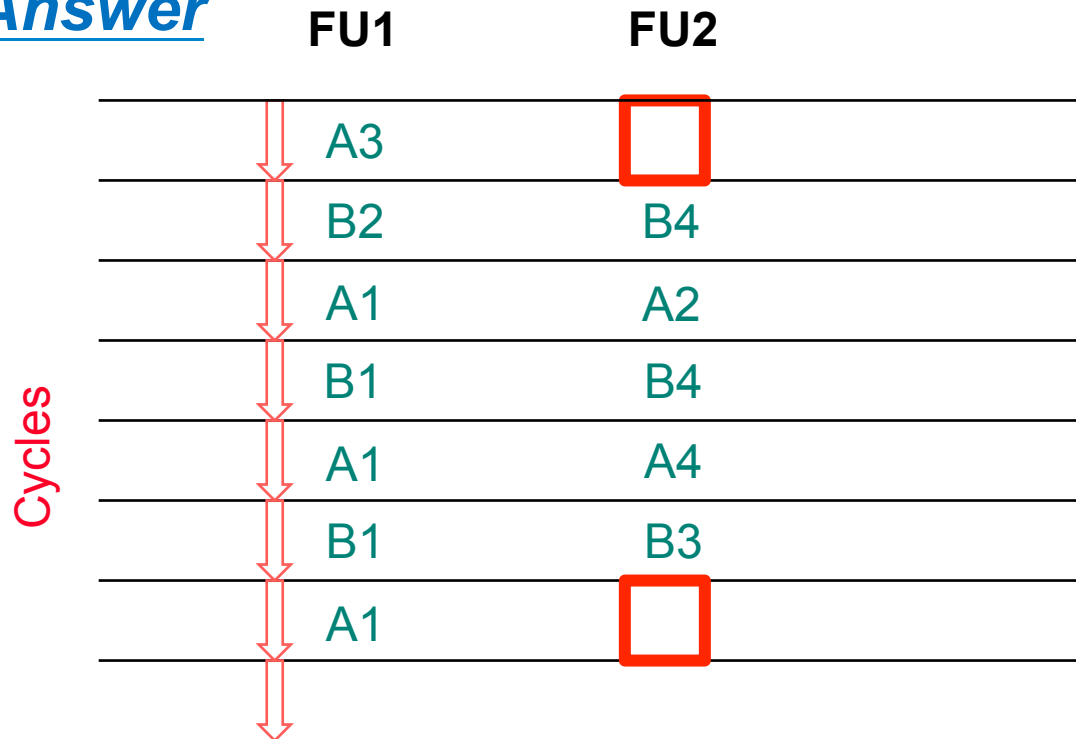


So, in total,  
we will need  
4 cycles to  
execute.  
However, 4  
issue slots are  
wasted

## Example 2 – part II (in case multiple cycle instructions can be switched before completion)

Now assume you have 1 FGMT CPU (as described in (b)).  
How many cycles will it take to execute threads X and Y? How many issue slots are wasted due to hazards?

Answer



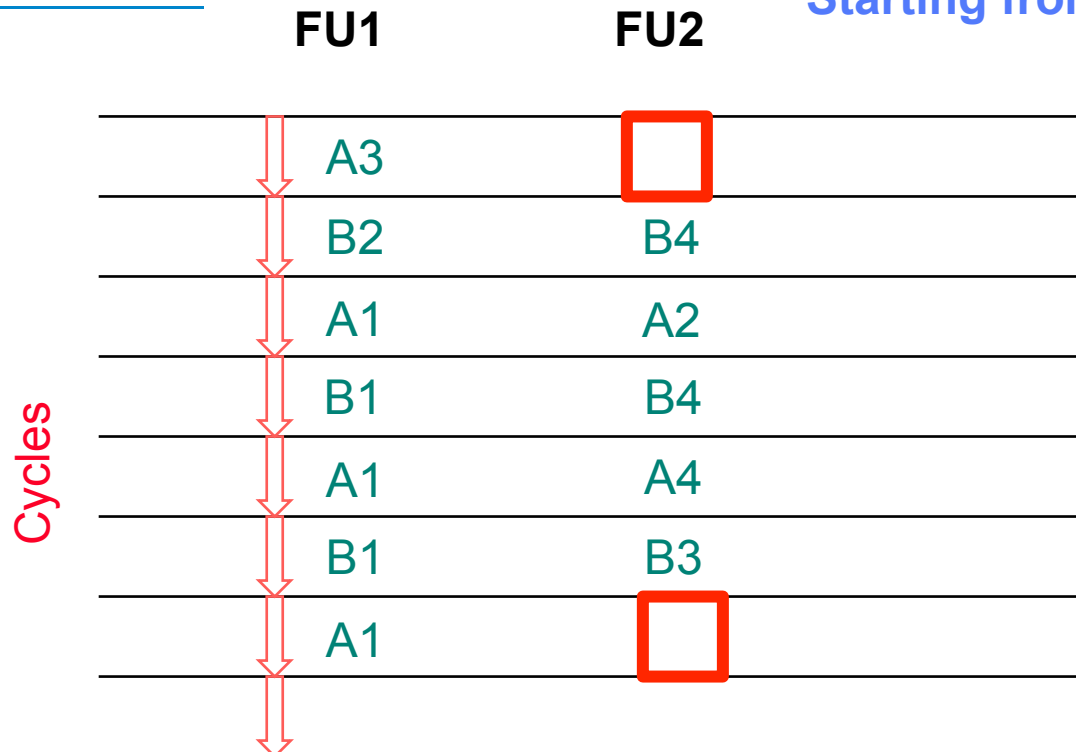
So, in total,  
we will need  
7 cycles to  
execute.  
and 2  
issue slots are  
wasted

## Example 2 – part III - (in case multiple cycle instructions can be switched before completion

Now assume you have 1 SMT processor (as described in (c)). How many cycles will it take to execute threads X and Y? How many issue slots are wasted due to hazards?

### Answer

Starting from the FGMT case



So, in total,  
we will need  
6 cycles to  
execute.  
and no  
issue slots are  
wasted.