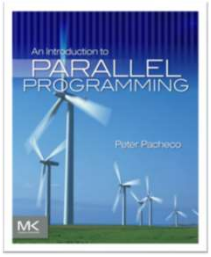
An Introduction to Parallel Programming

Peter Pacheco



Chapter 2

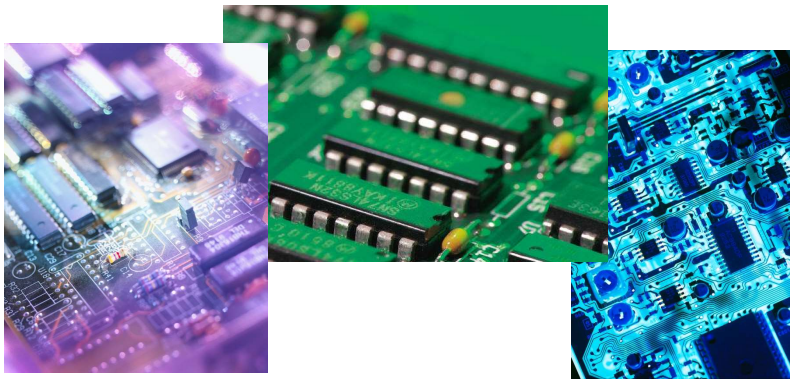
Parallel Hardware and Parallel Software

1

Roadmap

- Some background
- Modifications to the von Neumann model
- Parallel hardware
- Parallel software
- Input and output
- Performance
- Parallel program design
- Writing and running parallel programs
- Assumptions

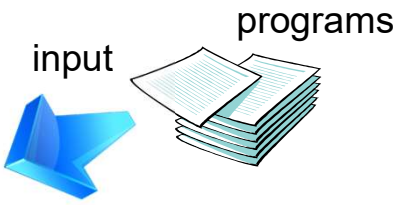
2



SOME BACKGROUND

3

Serial hardware and software



output

Computer runs one program at a time.

4

The von Neumann Architecture

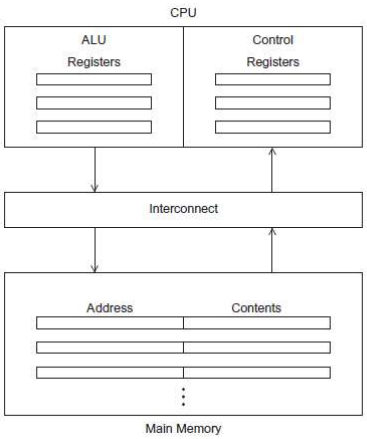
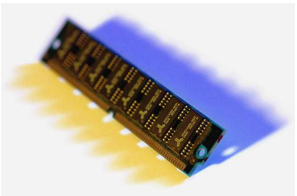


Figure 2.1

5

Main memory

- This is a collection of locations, each of which is capable of storing both instructions and data.
- Every location consists of an address, which is used to access the location, and the contents of the location.



6

Central processing unit (CPU)

- Divided into two parts.
- **Control unit** - responsible for deciding which instruction in a program should be executed. (*the boss*)

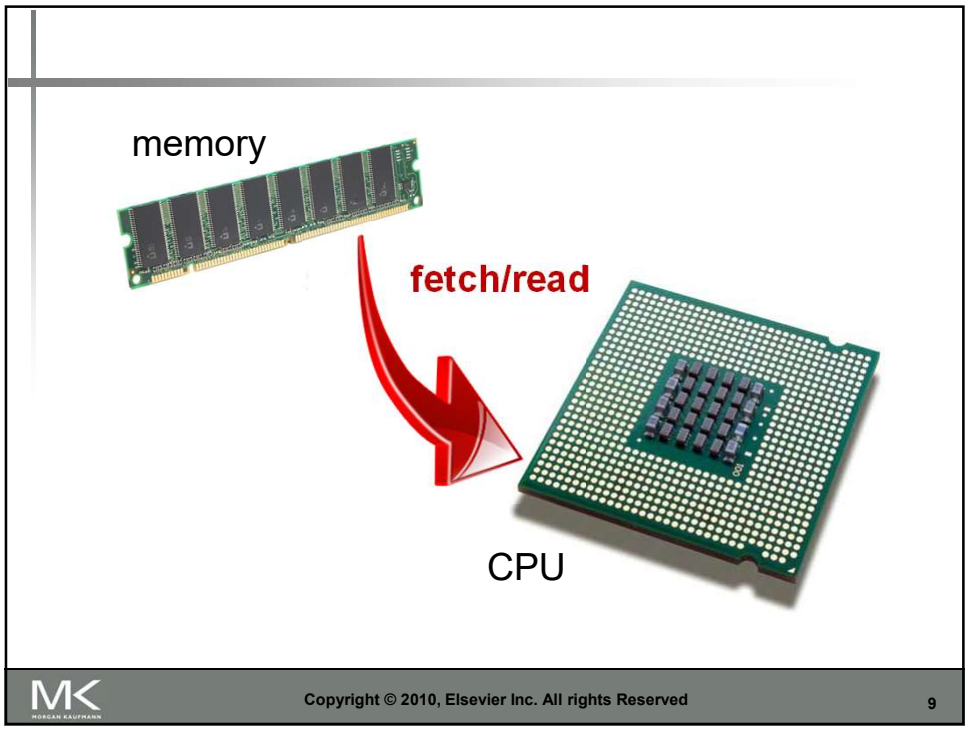
- **Arithmetic and logic unit** (ALU) - responsible for executing the actual instructions. (*the worker*)

7

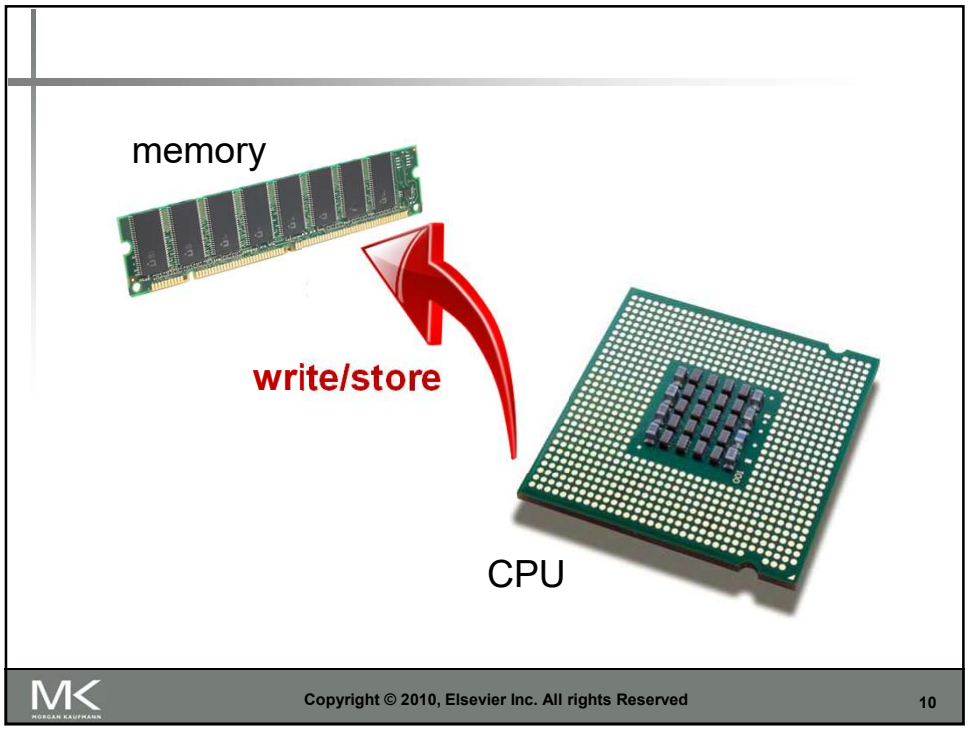
Key terms

- **Register** – very fast storage, part of the CPU.
- **Program counter** – stores address of the next instruction to be executed.
- **Bus** – wires and hardware that connects the CPU and memory.

8

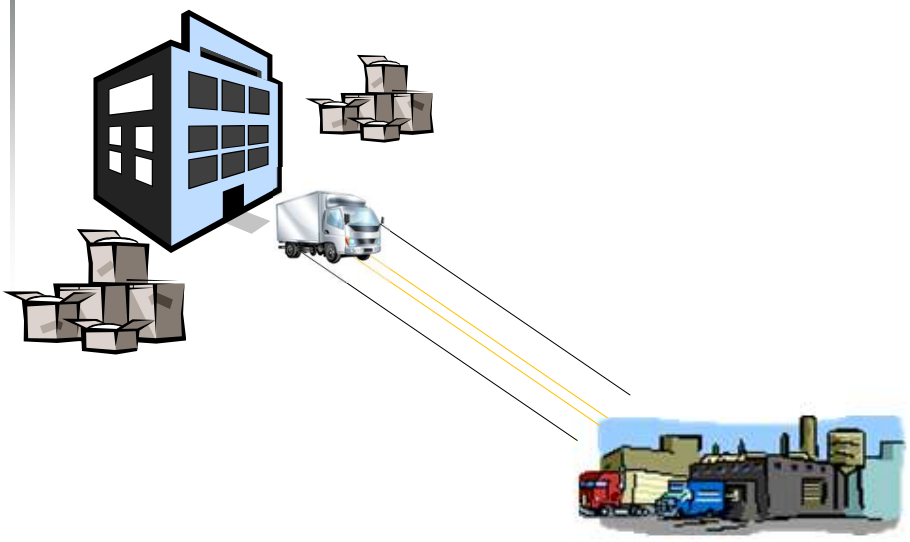

memory
fetch/read
CPU
MK
MORGAN KAUFMANN
Copyright © 2010, Elsevier Inc. All rights Reserved
9

9


memory
write/store
CPU
MK
MORGAN KAUFMANN
Copyright © 2010, Elsevier Inc. All rights Reserved
10

10

von Neumann bottleneck



11

An operating system "process"

- An instance of a computer program that is being executed.
- Components of a process:
  - The executable machine language program.
  - A block of memory.
  - Descriptors of resources the OS has allocated to the process.
  - Security information.
  - Information about the state of the process.

12

Multitasking

- Gives the illusion that a single processor system is running multiple programs simultaneously.
- Each process takes turns running. (**time slice**)
- After its time is up, it waits until it has a turn again. (**blocks**)

Threading

- Threads are contained within processes.
- They allow programmers to divide their programs into (more or less) independent tasks.
- The hope is that when one thread blocks because it is waiting on a resource, another will have work to do and can run.

A process and two threads

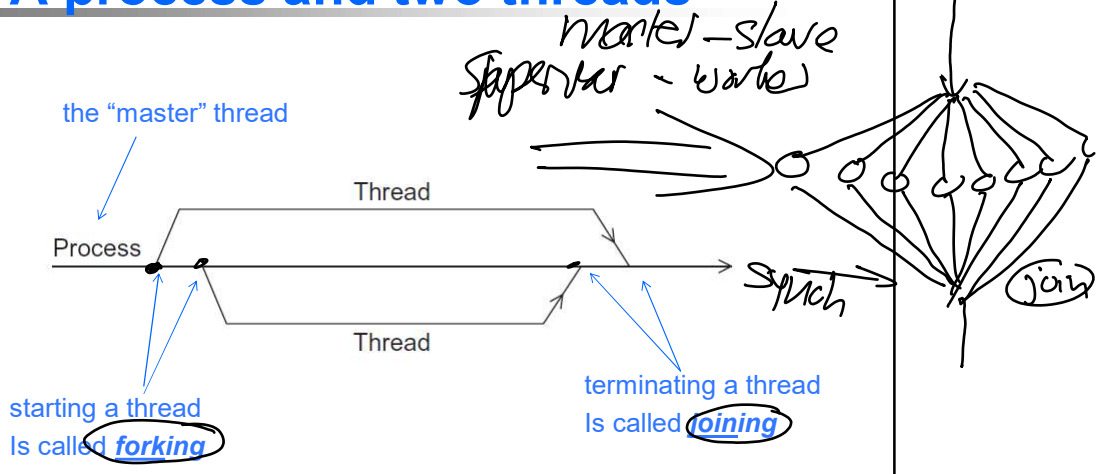


Figure 2.2

15

PARALLEL SOFTWARE

98

The burden is on software

- Hardware and compilers can keep up the pace needed.
- From now on…
  - In shared memory programs:
    - Start a single process and fork threads.
    - Threads carry out tasks.
  - In distributed memory programs:
    - Start multiple processes.
    - Processes carry out tasks.

SPMD – single program multiple data

- A SPMD programs consists of a single executable that can behave as if it were multiple different programs through the use of conditional branches.

```
if (I'm thread process i)
    do this;
else
    do that;
```

Writing Parallel Programs

1. Divide the work among the processes/threads
   (a) so each process/thread gets roughly the same amount of work
   (b) and communication is minimized.
2. Arrange for the processes/threads to synchronize.
3. Arrange for communication among processes/threads.

```
double x[n], y[n];
...
for (i = 0; i < n; i++)
    x[i] += y[i];
```


message-passing

```
char message [ 1 0 0 ] ;
...
my_rank = Get_rank ( ) ;
i f ( my_rank == 1) {
    sprintf ( message , "Greetings from process 1" ) ;
    Send ( message , MSG_CHAR , 100 , 0 ) ;
} e l s e i f ( my_rank == 0) {
    Receive ( message , MSG_CHAR , 100 , 1 ) ;
    printf ( "Process 0 > Received: %s\n" , message ) ;
}
```