Operating Systems:
Internals and Design Principles
William Stallings

Chapter 5

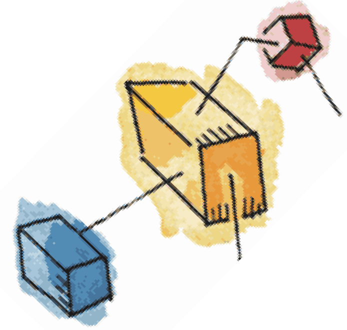
Co tual

d

Synchro

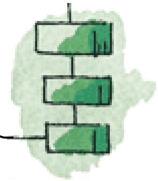
Outline

- Race condition
- Critical section
- Mutual exc
- Hardware
  - Atomic operations
  - Special machine instructions
    - Compare&Swap
    - Exchange



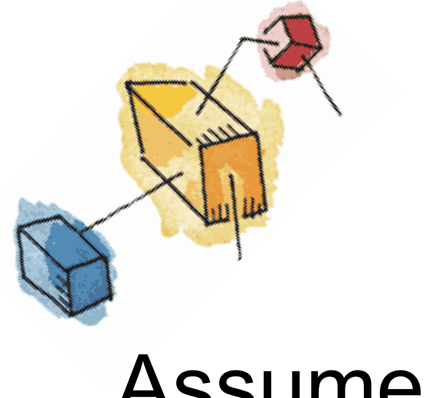
Multiple Processes

- The design of modern Operating Systems is concerned with the management of multiple pr ds
  - Multiprogr
  - Multiprocessing
- Big Issue is Concurrency
  - Managing the interaction of processes



Race Condition

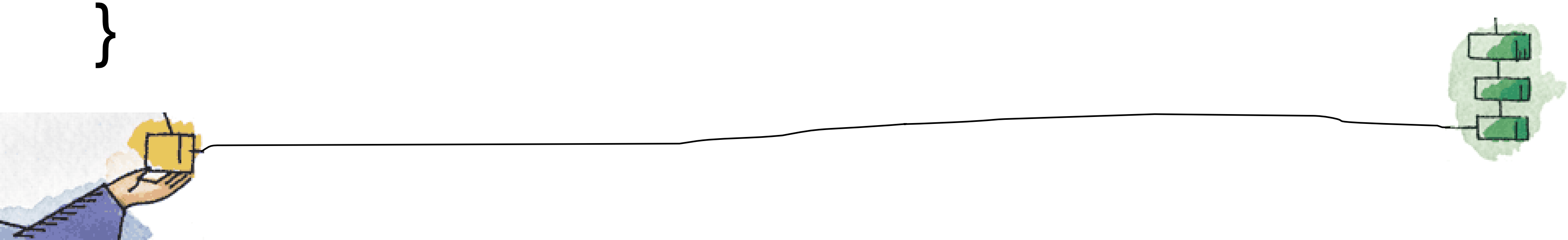
- A race condition occurs when
  - Multiple processes or threads read and write shared data items
  - They do s final result depends on the order on of the processes.
- The output depends on who finishes the race last.

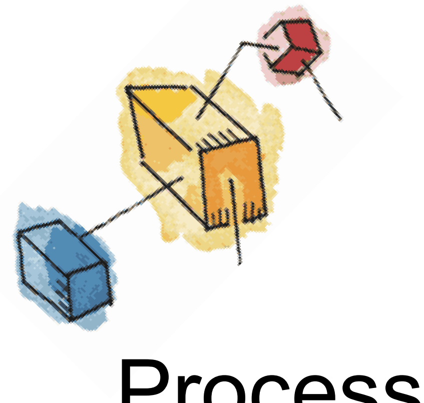


A Simple Example

Assume chin is a shared variable.

```
void echo()
{
   chin = getchar();
   chout = chin;
   putchar(chout);
}
```

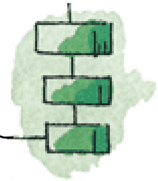




A Simple Example: On a Multiprocessor

```
Process P1                    Process P2

      .                             .
chin = getch                        .
      .                               getchar();
chout = chin;                           chin;
      .                       putchar(chout);
putchar(chout);                     .
      .                             .
```



A Simple Example on a Single Processor System

- count++ could be implemented as

```
register1 = count
register1 = register1 + 1
count = r
```

- count-- could be impleme

```
register2 = count
register2 = register2 - 1
count = register2
```

A Simple Example on a Single Processor System

- Consider:
  - process A increment count and process B decrement count simultaneously
  - the executi   nt = 5” initially:

S0: process A execute register1 =   ter1 = 5}
S1: process A execute register1 =   {register1 = 6}
S2: process B execute register2 = count   {register2 = 5}
S3: process B execute register2 = register2 - 1   {register2 = 4}
S4: process A execute count = register1   {count = 6 }
S5: process B execute count = register2   {count = 4}

Critical Section

- When a process executes code that manipulates shared data (or resource), we say that the process is in its Critical Section.

- Need to desig esses can use to cooperate.

- A general structure:

```
…
entry section
    critical section
exit section
    noncritical section
    …
```

Mutual Exclusion

- Only one process at a time is allowed in the critical section for a resource

- No assumption ve process speeds or number of p

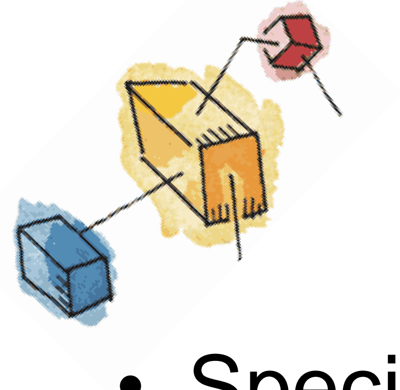
- A process must not be delay c a critical section when there is no othe sing it

- A process that halts in its noncritical section must do so without interfering with other processes

Mutual Exclusion

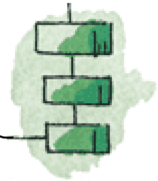
Mutual Exclusion: Hardware Support

- Interrupt Disabling
  - A process runs until it invokes an operating system service or until it is interrupted
  - Disabling in ual exclusion
  - Work in uni
- Disadvantages:
  - the efficiency of execution could be noticeably degraded
  - this approach will not work in a multiprocessor architecture



Mutual Exclusion: Hardware Support

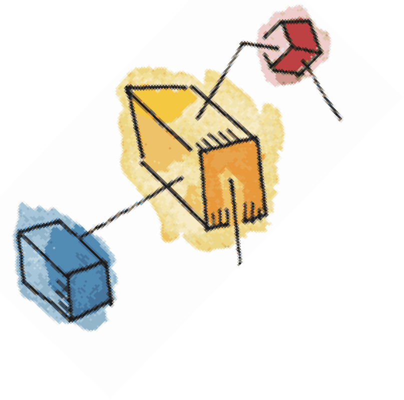
- Special Machine Instructions:
  - Compare&Swap Instruction
    - also called a "compare and exchange instruction"
  - Exchange
- These are atomic instr
  - Operations are indivisible



Compare&Swap Instruction

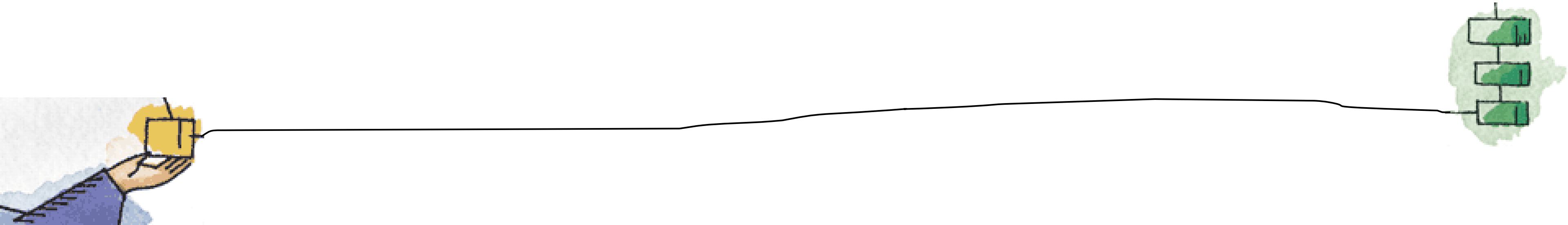
```
int compare_and_swap (int *word,
   int testval, int newval)
{              =0          =1
    int oldval;
    oldval = *wo
wval;if (oldval =
    return oldval;
}
```

- If `word = 1`, unchange, and return `1`
- If `word = 0`, `word = 1`, and return `0`



Compare&Swap Instruction

Busy waiting

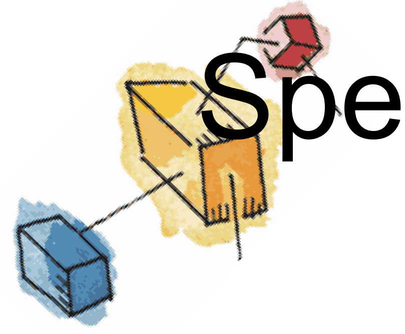


Exchange instruction

```
void exchange (int register, int
  memory)
{
   int temp;
   temp = memory;
   memory = register;
   register = temp;
}
```

Exchange Instruction

Busy waiting



Special Machine Instructions: Advantages

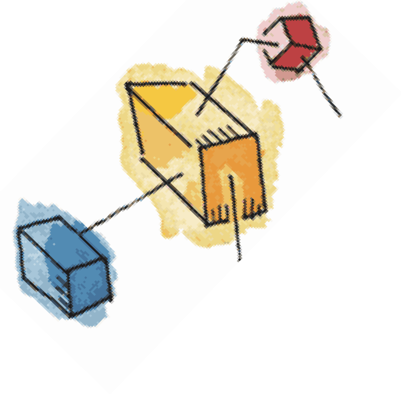
- Applicable to any number of processes on either a single processor or multiple processors sharing main memory
- It is simple a verify
- It can be used to support ritical sections; each critical se be defined by its own variable

Special Machine Instructions: Disadvantages

- Busy-waiting is employed, thus while a process is waiting for access to a critical section it continues to consume processor time

- Starvation is cess leaves a critical sectio e process is waiting.

  - Some process could indefinitely be denied access.
- Deadlock is possible



Key Terms