CPSC 457

Spring 2018

Assignment 1

Siddharth Kataria

(30000880)

A1)

(a) If the stages are not parallelized:

The minimum time to execute 1 full instruction = sum of time taken by each unit

Thus, time per instruction = 8 nsec + 2nsec + 1nsec

= 11 nsec

1 second = 1,000,000,000 nsec

Thus, in 1 second the CPU cycle, on average, execute (1 sec)/(11 nsec)

= 1000000000 / 11

= 90909090.9091

= 90,909,091 (approx.) instructions per second if the units are not operating in parallel.

(b) If the stages are parallelized:

The minimum time to execute 1 full instruction = slowest unit on the pipeline cycle

Thus, time per instruction = 8 nsec (Time taken by the fetch unit)

1 sec = 1,000,000,000 nsec

Thus, in 1 second in the CPU cycle, on average, execute (1 sec)/(8 nsec)

= 1000000000 / 8

= 125,000,000 instructions per second if the units are pipelined.

A2)

A benefit of using Virtual Machines for -

(a) a company:

It is cheaper to buy one big server instead of many smaller ones. As virtual machines have a potential saving a company a lot of money by system consolidation and hosting multiple servers running on the same server computer concurrently.

(b) a programmer:

Since it is safer to run unsafe programs in VMs, a programmer gets the opportunity to use host systems which are protected from the VMs to run such unsafe programs without any damage to the host system.

(c) a regular user:

A Virtual Machine uses a host system to create the illusion that each user's machine has its own processors and own hardware. Buying multiple machines is expensive, thus VM's give a regular user a fully functional machine with high end specs even though the machine is simply a virtual machine, isolated and safe from other VMs.

(d) a system administrator:

A Virtual machine is the perfect system for operating systems research and development as normal system operations seldom needs to be disrupted from system development while saving a lot of money.

A3)

(a) Interrupts:

External events delivered to the CPU due to I/O, time or user input for informing the CPU about upcoming data movement which is on the bus. The CPU suspends its own work to handle such interrupts by executing an interrupt handler which puts the CPU in kernel mode before switching back to its original operations.

(b) Traps:

Internal events such as system calls and/or error conditions (e.g. division by zero) which occur because of execution of a machine instruction. This instruction switches the system from user mode to kernel mode and invokes a predefined function. When the kernel routine is done, user mode restores and application continues.

(c) Difference between interrupts and traps:

While traps are invoked by system calls which are internal events, occurring because of execution of a machine instruction, an interrupt is an external event usually by I/O or by user input. Interrupts are asynchronous with the current activity of the CPU, thus the time of the event is not known and it's not predictable, whereas, traps are synchronous with the CPU's activity.

(d) Why interrupts and traps are handled in kernel mode instead of user mode:

Kernel mode is a basic unit of the operating system running at all times on the computer. Also known as the unrestricted mode in the CPU, in the kernel mode all instructions and I/O operations are allowed and all memory can be accessed. As user mode has no access to hardware and a limited instruction set, kernel mode has full access to all hardware with a full instruction set. Thus, in user mode, the applications cannot communicate with the hardware directly. The CPU thus switches itself in kernel mode by invoking a system call, aka a trap. When transferring data from drives, an interrupt controller informs the CPU and puts the data on the bus, the CPU switching itself to kernel mode to handle the interrupts by executing the appropriate interrupt handler which is available in the kernel mode instruction set.

A4)

(a) Outputs of time commands:

(i) time ./countLines romeo-and-juliet.txt :

```
Siddharth@Pot8Os /cygdrive/c/Users/Siddharth/Desktop/CPSC 457/Assignment 1
$ time ./countLines romeo-and-juliet.txt
4853 romeo-and-juliet.txt

real    0m0.349s
user    0m0.000s
sys     0m0.000s
```

(ii) time wc -l romeo-and-juliet.txt :

```
Siddharth@Pot8Os /cygdrive/c/Users/Siddharth/Desktop/CPSC 457/Assignment 1
$ time wc -l romeo-and-juliet.txt
4853 romeo-and-juliet.txt

real    0m0.041s
user    0m0.000s
sys     0m0.000s
```