**Shishir Ghorashainee**

CSC 362 Homework Assignment #7

Due Date: Monday, December 2

Word Processor all answers, show work for partial credit.

1. Architectures are considering one of two changes to their processor. The first is to increase the clock speed from 3GHz to 4.5GHz. The estimate is that the 4.5GHz processor will stall 25% of the time. The second is to add a branch prediction buffer (cache) which, when accurate, removes branch penalties. A branch without penalties is 1 cycle whereas with a penalty is 3 cycles. The prediction buffer is thought to be 60% accurate. What are the speedups of both enhancements?

**Speedup from the increase in clock Speed enhancement:**

* **f = 75% = 0.75**

**k = 1 + 0.5 = 1.5**

**Now,**

**S = 1 / (1 – 0.75 + 0.75/1.5) = 1.33**

Again,

**Speedups from Prediction Buffer enhancement:**

* **f = 60% = 0.60**

**k = 3-1 = 2**

**Now,**

**S = 1 / (1 – 0.6 + 0.6 / 2) = 1.43**

1. Answer the following questions about the 4 forms of I/O.
   1. What hardware system was added to our computers so that we no longer needed programmed I/O? **Interrupt Systems**
   2. Of the 4 forms of I/O, which one(s) require that the CPU move the data to and from memory?

= **Programmed I/O , Interrupt-Driven I/O**

* 1. What hardware capability was added to I/O modules to move from interrupt-driven to DMA I/O? **DMA controller**
  2. An I/O operation requires 1024 bytes to be transferred from memory to I/O device. In programmed I/O, how many interrupts arise? = **None** In interrupt-driven I/O? **= 1024** In DMA I/O? **= 1** For each, assume no error arises during the entire transfer.

1. Answer the following questions regarding extra I/O hardware. Each should be answered in 1-2 sentences.
   1. Address decoder

= **to determine if a communication from CPU is intended for it and if so, which specific device / register / storage location.**

* 1. What is the interrupt controller used for?

= **It receives interrupt signals from devices, prioritizes them, interrupts the CPU and communicates the device number to the CPU.**

* 1. What is the bus arbiter used for?

= **It receives I/O bus requests from I/O devices, prioritizes them and selects the device that can next use the bus.**

* 1. What is the I/O processor used for?

= **It is the next level of I/O model that acts as its own mini processor withs own ALU and built-in control programs. It takes care of common errors like disk I/O and rarely interrupting the CPU, and only doing so when the entire I/O process is completed.**

1. Assume we have eight drives in a RAID configuration whereby a parity byte is stored using seven data bytes. The parity byte is then stored on the eighth drive. We have the following bytes in the same location across the disks where the byte on disk 6 is inaccessible. What is this missing byte? (restore it by using the other seven bytes, assume even parity is used)

Disk 1: 01101110

Disk 2: 10101100

Disk 3: 00101110

Disk 4: 01111000

Disk 5: 01101100

Disk 6: ????????? Disk 7: 00111100

Parity: 01100010

= The missing byte on Disk 6 is **10100110**

1. Answer the following regarding RAID levels.
   1. We can simulate RAID in software on a single drive. Would be there be any value in doing so for RAID 0? Explain.

= **Simulating RAID 0 on a single drive would have no real value, as RAID 0 is designed to improve performance through striping across multiple drives. A single drive cannot provide the parallel data access benefits of RAID 0, so there would be no performance improvement.**

* 1. RAID 4 is not used. Why not?

= **RAID 4 is not used because storing all parity data on a single drive creates a bottleneck, as both data and parity must be accessed simultaneously. This defeats the purpose of stripes and limits parallel access. Although RAIDs 5 and 6 solve this by distributing parity across all disks, allowing for potential parallel access**

* 1. Rank the RAID levels that are used by expense from cheapest to most expensive.

1. **Raid 0 (Cheapest)**
2. **Raid 2**
3. **Raid 3**
4. **Raid 4 (Not used)**
5. **Raid 5**
6. **Raid 6**
7. **Raid 1 (Most Expensive)**
   1. Which of RAID 3 or RAID 5 would you select for a server? Why?

**= I’d select RAID 5 for a server because it offers better performance and fault tolerance by distributing parity across all disks. While RAID 3 might work for smaller systems that require some redundancy, it has a performance bottleneck due to the dedicated parity disk, making it less suitable for the high needs of a server.**

1. Answer the following questions regarding using a binary semaphore to implement synchronization.
   1. Consider the following alternate implementation for the Wait and Release functions. Would these work in place of the implementations covered in class? Explain.

S is initialized to 0

Wait: While(S!=0); Release: S--;

S++;

= **No, the alternate implementation won't work correctly because initializing S to 0 means the Wait function will immediately check the condition S! = 0, which evaluates as false and skips the loop. This leads to the Wait function incrementing S without waiting, allowing processes to access the shared resource simultaneously.**

* 1. In what way is the binary semaphore unfair? That is, how could it lead to a process never gaining access to the shared datum?

= **The binary semaphore is unfair because it can cause starvation, where one process keeps accessing the shared resource while the other remains stuck in the wait loop. This happens if the running process keeps releasing the semaphore, preventing the waiting process from accessing the resource.**

* 1. To implement Wait and Release, we use atomic instructions. What are atomic instructions and why do we need them?

= **Atomic instructions are instructions that cannot be interrupted once started and are executed as a single atomic step. We need them to implement functions like wait, and prevent race conditions and data corruption, ensuring that critical operations, such as modifying a semaphore, are executed fully without being interrupted by other processes.**

1. Given the C program below, provide the run-time stack as it would appear at point P1. For each variable/parameter on the stack, list it as well as its current value. You will have to work through the logic of the code to determine the order that functions are called.

void main() {

int x = 1, y = 5, z;

if(x < y) foo(x, y);

else bar(x, y);

}

void foo(int a, int b) { printf(“%d”, f1(a, b));

}

void bar(int a, int b) { printf(“%d”, f2(a, b));

}

int f1(int c, int d) {

if(c<d)

return f1(c+1, d-1)+1; else return f2(c, d);

}

int f2(int e, int f) {

return e+f;  P1

}



**f2:**

**e = 3**

**f = 3**

**return to f1(3, 3)**

**f1:**

**c = 3**

**d = 3**

**return to f1(2, 4)**

**f1:**

**c = 2**

**d = 4**

**return to f1(1, 5)**

**f1:**

**c = 1**

**d = 5**

**return to foo(a, b)**

**foo:**

**a = 1**

**b = 5**

**return to main**

**main:**

**x = 1**

**y = 5**

**z**

**no return value**

**return to OS**