Memory Mapped I/O

Certain memory addresses correspond to registers in I/O devices and not normal memory.

**Control Register:** Indicates if it is okay to read/write data register

**Data Register:** Contains I/O data

<table>
<thead>
<tr>
<th>Register</th>
<th>Location</th>
<th>Contains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receiver Control</td>
<td>0xffffffff</td>
<td>Lowest two bits: Interrupt Enable Bit, Ready Bit</td>
</tr>
<tr>
<td>Receiver Data</td>
<td>0xffffffff4</td>
<td>Received data stored at lowest byte</td>
</tr>
<tr>
<td>Transmitter Control</td>
<td>0xffffffff8</td>
<td>Lowest two bits: Interrupt Enable Bit, Ready Bit</td>
</tr>
<tr>
<td>Transmitter Data</td>
<td>0xffffffffc</td>
<td>Transmitted data stored at lowest byte</td>
</tr>
</tbody>
</table>

Write MIPS code to read a byte from the receiver and immediately send it to the transmitter.

```mips
lui $t0 0xffff
receive_wait:  #poll on ready of receiver
    lw $t1 0($t0)
    andi $t1 $t1 1
    beq $t1 $zero receive_wait
    lb $t2 4($t0)  #load data
transmit_wait:  #poll on ready of transmitter
    lw $t1 8($t0)
    andi $t1 $t1 1
    beq $t1 $zero transmit_wait
    sb $t2 12($t0)  #write to transmitter
```

Polling and Interrupts

<table>
<thead>
<tr>
<th>Operation</th>
<th>Definition</th>
<th>Pro/Con</th>
<th>Good For</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Polling</strong></td>
<td>Pretty much the above; forces hardware to wait on ready bit (alternatively, if timing of device is known – the ready bit can be polled at the frequency of the device). It basically means manually checking the ready bit</td>
<td>PRO: -easy to write -poll handler does not have excessively high overhead -deterministic -doesn't require additional hardware CON: -unfeasable on hardware with fast transfer rates that is actually rarely ready (e.g. Ethernet card receiver)</td>
<td>-Slow devices: Mouse, Keyboard</td>
</tr>
<tr>
<td><strong>Interrupts</strong></td>
<td>Hardware fires an “exception” when it becomes ready. CPU changes $PC to execute code</td>
<td>PRO: -Necessary for fast devices that are rarely ready.</td>
<td>Fast devices: Hard drives Network cards</td>
</tr>
</tbody>
</table>
Interrupt Enable: Indicates whether or not to cause an interrupt when the ready bit is set

**Network Overhead vs. Bandwidth**

Assume we have two networks A and B.

- Network A has a 200us overhead and a peak bandwidth of 10MB/s
- Network B has a 500us overhead and a peak bandwidth of 100MB/s

(MB = MebiByte)

How long would it take to send 1000 Bytes over each network?
Which network is better for sending large amounts of data?

Network A: 0.3 ms
Network B: ~0.5 ms

However, with larger amounts of data, network B’s faster bandwidth will dominate the latency effect.

Suppose:
- L1 access is 2 cycles.
- L2 access is 4 cycles.
- Suppose a 95% hit rate in L1 for data, and a global 99% hit rate in L2 for data.
- Main memory is 100 cycles.
- Page table lookup is 150 cycles
- L1 and L2 both are physically tagged.
- A 99% TLB hit rate. TLB lookup can be parallelized with memory access, so a TLB hit is effectively free.
- Ignore effects of having page table entries in data cache.

What is the AMAT for data?

Translation: 0.01 * 150 = 1.5 cycles on average
Data: 2 + 0.05 * 4 + 0.01 * 100 = 3.2
Total: 4.7

What if we disabled virtual memory?
Data: 2 + 0.05 * 4 + 0.01 * 100 = 3.2

Now let's suppose one time in five million we have a page fault and it takes 10 million cycles to service. What happens to AMAT?
Goes up about two cycles to 6.7