Luke Jaffe
Microprocessor-Based Design
Due: 03/16/16

Pre-Lab 4 Assignment Part B

Pre 4.9) Read the description about dynamic memory allocation in embedded systems at: http://www.embedded.com/design/programming-languages-andtools/4416457/EMB-tm-6-15-13-Dynamic-memory-and-heap-contiguity Describe in your own words what are the challenges with dynamic memory allocation through malloc in embedded systems.

Whereas static memory allocation is deterministic, dynamic memory allocation is non-deterministic. One issue with using malloc (dynamic allocation) is that heap space may run out, in which case malloc will return a NULL pointer. This case may be challenging to handle. Another issue is that malloc can cause external fragmentation, in which there is enough total memory available for a given request, but not enough contiguous memory available in any one free section.

Pre 4.10) What is a “block memory allocation” (also called memory pool, buffer pool) with fixed block sizes? How does it help to avoid the problems with dynamic memory allocation?

In “block memory allocation”, one or more pools are defined which contain a number of fixed size memory blocks. Using multiple pools allows choice of block size. Using this approach eliminates the problem of fragmentation. While it is possible to run out of blocks when allocating memory, this issue becomes predictable and can therefore be mitigated.

Pre 4.11) When using fixed sized blocks of memory (through buffer pools), still communication occurs through operating system queues. What should be passed in the queue, the data of a memory block or a pointer to the memory? Discuss advantages and disadvantages.

If only the pointer is passed to the queue, then the memory block cannot be reused until the data is read out on the other end. This would require some kind of global flag or another queue to send a message back indicating that the data has been read and block can be used again. If the data is passed by copy, then the block can be reused immediately; however, this would mean transmitting a lot of data through the queue, which could be avoided by just passing a pointer. For the sake of implementation convenience and safety in not overwriting the memory block, it could be simpler just to pass the data by copy.

Pre 4.12) Assume a generic producer / consumer application that uses a fixed block size memory pool, a queue between producer and consumer which only carries the pointer to the buffer. What does the consumer need to do after having processed a data block to avoid a memory leak?

The consumer needs to tell the producer that the block can be used again, after it processes the block. Otherwise, the block will be assumed to be in use, and will not be available for the producer to use, meaning a memory leak has occurred.

Pre 4.13) In this lab, we will be using again the AXI Stream FIFO to interface with the I2S module. The aim is to fill the FIFO with a whole block of samples, and then get an interrupt when the samples have been transferred out of the TX FIFO. Which interrupt should be used (refer to reading reference D)? Name the interrupt and reason about your selection. Write a code snippet to read the according interrupt status register and identify the interrupt type.

The TC or “Transmit Complete” interrupt should be used for this purpose.

```
#define BASE_ADDR               (?)
#define INT_REG_OFFSET          (0x0)
#define INT_REG_ADDR            (BASE_ADDR + INT_REG_OFFSET)
#define TC_BIT                  (0x01<<27)

void isr()
{
        if ((*INT_REG_ADDR) & TC_BIT)
                // transmit is complete
        else
                // transmit is not complete
}
```

Pre 4.14) Assume TX FIFO contains 10 samples in stereo mode and has been configured for a 48kHz. Compute how long does it take to completely drain the FIFO? Show your calculation.

Assuming 10 samples in stereo mode means 5 left and 5 right, then there are 10 packets (not say 20). We assume a transmission rate of 48kHz means that 48k packets are sent per second.
Then it takes 10/48,000 = 208 microseconds to transmit the data.

Pre 4.15) Assuming the core runs at 667MHz, how many core cycles pass during the TX FIFO draining time for 10 cycles? Show your calculation. Do you estimate that this time would be enough for raising and processing an interrupt?

Core runs at 667MHz = $6.67*10^{8}$ Hz
208 microseconds = $2.08*10^{-4}$ s
cycles = $2.08*10^{-4}$ s * $6.67*10^{8}$ Hz = 138,736 cycles

This is plenty of time for raising and processing an interrupt.

Pre 4.16) Assuming audio chunks with 250 samples (125 left, 125 right). That each chunk will be copied into the TX FIFO in an interrupt if the FIFO is almost empty. How many interrupts per second will occur with a sampling rate of 48kHz? Show your calculations.

250 samples / 48kHz = $5.2 * 10^{-3}$ s = interrupt duration
1 s / $5.2 * 10^{-3}$ s = 192 interrupts per second

Pre 4.17) Our audio player will copy an audio chunk into the TX FIFO in an interrupt if the FIFO is almost empty (see earlier question). This works for a steady state. What needs to happen to start the system? Note that in the beginning the FIFO is empty and thus will not give an “almost empty interrupt”. Briefly outline what actions have to be taken to start playing the audio.

One approach is to put the first chunk into the TX FIFO manually, to start the process. When this first chunk has been mostly transmitted, the TX FIFO “almost ready” interrupt will trigger, and the system will transmit the rest of the chunks automatically. The audio player must adjust its current chunk pointer to the second chunk, so that the first chunk is not repeated.

Maybe you could also just put one empty (zero) sample in the TX FIFO, and this would have the same effect. A zero sample shouldn't produce any unexpected sound if it is at the beginning.