CSE/EE 5/7385 Microcontroller Architecture and Interfacing

HOMEWORK 4

1. An embedded systems designer is creating an arbitrary waveform generator that will output a periodic waveform based upon a set of data values stored in a lookup table. The designer has created the table that consists of 400 32-bit Q.31 values. The arbitrary waveform is specified to have a frequency of 10 kHz. (NOTE: k is the SI unit of 1000).

a) To meet the 10kHz specification, give the delay value (in units of time) between the output of each value stored in the lookup table.

Waveform Period:

$$\tau = \frac{1}{f} = \frac{1}{10 \times 10^3} = 100\mu s$$

400 different values must be output with equally space delays over the interval of the waveform period. Delay value is:

$$t_{DELAY} = \frac{100\mu s}{400} = 250ns$$

b) Give an appropriate ARM instruction for loading the waveform value from the lookup table into register **r0** and whose base address is stored in **r1**. Furthermore assume the pointer register is **r2**. The pointer value is an integer that cyclically has incremented values running from 0 to 399 (in decimal); that is, **r2** is the offset from the base address of the lookup table that points to the current waveform value.

```
ldr  r0, [r1, r2, LSL #2]
```

c) Assume your answer in part a) above is 500μs (NOTE: this is NOT necessarily the correct answer for part a), it may or may not be). Also assume that the ARM program that outputs the values is of this form:

```
loop: ;load lookup value (1 instruction)
      ;output (str) value to output device (1 instruction)
      ;increment the pointer register by 1 (1 instruction)
      ;if r1 is 400 or greater, set r1 to 0 (3 instructions)
      ;go to loop (1 instruction)
```

If each instruction above requires 1 CPU clock cycle to execute, what is the minimum ARM core clock speed (ie. frequency) required to meet the 10kHz specification for the arbitrary waveform generator?

The code requires 7 ARM core clock cycles and these 7 cycles must not exceed the 500μs value. The minimum core clock frequency requires that the 7 instructions take exactly 500μs to execute, thus the answer is:

$$\tau_{ARM} = \frac{500\mu s}{7} = 71.4\mu s$$

$$f_{ARM} = \frac{1}{\tau_{ARM}} = 14kHz$$

d) How much memory will the lookup table occupy (in units of KB or MB)?

Need 400 ARM words and each word is 4 bytes, so 1600 bytes required.

ANSWER: 1600/1024=1.5625KB

2) How many external interrupt inputs (electrical inputs) are present on an ARM processor core and what are their names?

ANSWER: Two (2) named IRQ and FIQ

3) Give two reasons why the FIQ is faster than the IRQ for processing interrupts.

There are more than two reasons, here are some:

1) FIQ is the last entry in the vector table, therefore, the actual interrupt handler code can be inserted at the FIQ address instead of a jump to another section of memory. This sppeds up processing by removing the delay of executing a jump before the interrupt/exception is processed.

2) FIQ has a larger dedicated register file than IRQ, therefore the interrupt/exception handler does not have save off as many registers and restore as many after processing is complete. Specifically, r8 through r14 do NOT have to be saved for FIQ while r8 through r12 do have to be saved and restored for IRQ.

3) An IRQ exception can be delayed if it receives an FIQ during its processing. An FIQ exception does NOT suffer this potential delay because once an FIQ is received, an further FIQ exceptions are ignore (ie. FIQ interrupts are disabled).

4) Most interrupt/exception table entries are jumps to another area in memory where the actual exception handler code is present. Why are these exception table entries jump instructions?

Because only 4 bytes are available for the first six entries, therefore a jump to another location memory must be accomplished with these 4 available bytes to prevent overwriting the following exception vectors with exception handling code.

5) In addition, to a jump, another type of instruction can be stored in the exception vector table that causes the processor to execute a handler stored elsewhere in the memory. Give an example of this type of instruction and indicate why it might be used instead of a jump.

ANSWER: `ldr pc, =HANDLER_ADDRESS`

This type of instruction loads the absolute address of the handler into the **`pc`** (**`r15`**) instead of providing a jump using a **`ldr`** with a literal pool. Jump instructions provide an offset to added to the **`pc`** (**`r15`**) and the number of bits allowed for the offset limit how far away the handler can be located from the exception vector table. When a designer wishes to load the handler in a section of memory that is far away, this form of the instruction must be used.

6) Why might an embedded system designer choose to use "**`lr`**" rather than "**`r14`**" in their source code?

Both **`lr`** and **`r14`** are the same register. Because **`r14`** can be used as a general purpose register and as a special purpose link register, using **`lr`** makes the code more readable and provides a self-documenting procedure since it allows one to recognize that **`r14`** is being used for its special purpose as a link register rather than as a general purpose register.