Chapter 2

1. How many modes does the ARM7TDMI processor have? How many states does it have? How many modes does the Cortex-M4 have?

ARM7TDMI: 7 modes, 2 states
Cortex-M4: 2 modes, Handler mode and Thread mode

2. What do you think would happen if the instruction SMULTT (an instruction that runs fine on a Cortex-M4) were issued to an ARM7TDMI? Which mode do you think it would be in after this instruction entered the execute stage of its pipeline?

It wouldn’t recognize the instruction and would go into Undefined mode.

3. What is the standard use of register r14? Register r13? Register r15?

r14: Link Register
r13: Stack Pointer
r15: Program Counter

4. On an ARM7TDMI, in any given mode, how many registers does a programmer see at one time?

17 for User mode, 18 for remaining modes

5. Which bits of the ARM7TDMI status registers contain the flags? Which register on the Cortex-M4 holds the status flags?

ARM7TDMI: Bits [31:28]
Cortex-M4: APSR bits [31:28]
6. If an ARM7TDMI processor encounters an undefined instruction, from what address will it begin fetching instructions after it changes to Undefined mode? What about a reset?

On the ARM7TDMI: Undefined: 0x04 and Reset: 0x0

7. What is the purpose of FIQ mode?

Handling of highest priority interrupt(s)

8. Which mode on an ARM7TDMI can assist in supporting operating systems, especially for supporting virtual memory systems?

Abort mode

9. How do you enable interrupts on the ARM7TDMI?

Clear bits 6 and 7 of the CPSR

10. How many stages does the ARM7TDMI pipeline have? Name them.

3: Fetch, Decode, and Execute

11. Suppose that the program counter, register r15, contained the hex value 0x8000. From what address would an ARM7TDMI fetch the next instruction (assume you are in ARM state)?

This depends on the state of the machine. It would be 0x8004 if the machine is in ARM state.

Note that it could be 0x8002 if the machine is in Thumb state.

12. What is the function of the Saved Program Status Register?

To save the state of the machine in the CPSR when an exception is taken so it can be restored upon return of handling the exception.
13. On an ARM7TDMI, is it permitted to put the instruction

\begin{verbatim}
  SUB   r0, r2, r3
\end{verbatim}

at address 0x4? How about at address 0x0? Can you put that same bit pattern at address 0x4 in a system using a Cortex-M4?

On an ARM7TDMI, technically, yes and yes. The instruction would be executed. You could put the bit pattern for the SUB instruction at address 0x4 in a Cortex-M4 system, but remember that the bit pattern is seen as an address, so the machine will attempt to fetch an instruction from that address, and mostly likely die a horrible death.

14. Describe the exception vector table for any other microprocessor. How does it differ from the ARM7TDMI processor? How does it differ from the Cortex-M4?

Motorola 68000’s vectors correspond to different exceptions like ARM’s, but they contain addresses instead of instructions. It is actually more like the Cortex-M4’s exception vectors.

15. Give an example of an instruction that would typically be placed at address 0x0 on an ARM7TDMI. What value is typically placed at address 0x0 on a Cortex-M4?

\begin{verbatim}
  LDR   pc, [pc, #offset]
\end{verbatim}

where “#offset” is an offset to the Reset handler.

For the Cortex-M4, the address of the top of the stack is placed at address 0x0.

16. Explain the current program state of an ARM7TDMI if the CPSR had the value 0xF00000D3.

N, Z, C, V flags are set; IRQs and FIQs are enabled; ARM state; Supervisor mode
FIGURE B.1  Creating a new project.
FIGURE B.2 Naming the project.
FIGURE B.3  Device database dialog box.
FIGURE B.4  Sample code.
FIGURE B.5  Adding a source file to the project.
FIGURE B.6  Running code in the debugger.
FIGURE B.7  Creating a new project.
FIGURE B.8 Naming the project.
FIGURE B.9  Device database dialog box.
FIGURE B.10  Sample code.
FIGURE B.11  Running code in the debugger.