Luke Jaffe
ID#: 001182531
Microprocessor-Based Design
Due 02/23/16

# Homework 2

## Problem 2.1: Compilation Chain

a. What is the difference between a compiler and an assembler?

A compiler transforms a program written in one language into another language. Often, the source language is a high-level language, and the target language a low-level language. While a compiler must produce a functionally equivalent mapping, there is no specification as to exactly how this must be done. An assembler translates assembly language into machine code. Since assembly has a one-to-one mapping with machine code, this is a trivial task.

b. What is the difference between a cross-compiler and a compiler?

A cross-compiler is a compiler which can convert a program into executable code for a computer other than that on which it is run.

## Problem 2.2: Memory

a. The performance of processors is increasing faster than the performance of memory. Name and briefly outline one approach to reduce the performance gap between processor and memory.

Cache memory can be utilized to mitigate the memory wall. Cache memory is smaller, faster memory which stores copies of the data from frequently used main memory locations.

b. In which aspect could the performance enhancement suggested in (a) be detrimental to the predictability of an embedded system?

Using cache memory causes program behavior to become more state dependent. This depends on how earlier execution left the state of the cache. Further, execution time is less predictable: memory access times can vary by 100X.

c. What is the difference between SRAM and DRAM in terms of complexity, cost and performance?

SRAM is more complex (4-6 transistors per bit) and more expensive, but also more performant, and doesn't need refresh. DRAM is less complex (1 transistor, 1 capacitor per bit) and less expensive, but is slower and requires refresh.

d. We discussed in class about different types of ROM. For your next embedded project, which type of ROM would you chose? Please explain your selection.

Flash memory seems to be the most effective ROM for current technology. In particular, NAND flash has low cost per bit and high storage capacity. This makes it well suited for embedded applications (e.g. for storing firmware).

**Problem 2.3: ISA**

a. What is an ISA? What does it describe?

An ISA (instruction set architecture) is an API which can be implemented for different microarchitectures to program the machine. It specifies native data types, instructions, registers, addressing modes, memory architecture, interrupt and exception handling, and external I/O.

b. Highlight some advantages of an ISA when evolving along processor generations.

Since an ISA is abstracted away from any given microarchitecture, a processor can evolve many generations while still using the same ISA.

c. Name an example of a widely used ISA not mentioned in class.

The SPARC ISA is a RISC ISA developed by Sun Microsystems, and introduced in 1987. The ISA is open and royalty free, and is still being implemented in new microarchitectures today.

**Problem 2.4: Zynq Cortex A9**

a. Please categorize the Zynq Cortex A9 processor in terms of RISC/CISC, Harvard/van Neumann, and General Purpose/DSP. For each classification, explain why the Zynq Cortex A9 belongs to each of these categories.

The Cortex A9 implements ARMv7-A, which is a RISC ISA. While the Zedboard SoC has 220 DSP slices, the Cortex A9 is a general purpose processor. The processor has combined main memory (Von Neumann), but separate instruction and data caches (Harvard).

**Problem 2.5: Interrupts**

a. In lab 1, we have used a loop in order to read push buttons via polling and to turn on and off LEDs. Please name 3 disadvantages of this approach and explain their effect.

1. Waste CPU: While a processor iterates through the loop, it is wasting cycles which code be used on other code. When using interrupts instead, the processor will be able to execute useful code most of the time, and only handle interrupts when they are generated.
2. Waste power: The core executing the loop would be 100% consumed for the duration of the program, which would be a tremendous waste of power as compared to using interrupts, which require comparatively little CPU time.
3. Inefficient: In the looping method, the code must check the data register at each iteration (or simply set the led values in the data register to the switch values). Either way, this is a lot of additional code being executed since the loop is iterating as fast as possible. In the interrupt method, this code is only executed when an interrupt is triggered.

b. What is the difference between a Vector Interrupt Controller and a Programmable Interrupt Controller?

A Vector Interrupt Controller directly provides the start address of the user ISR (whereas a

Programmable Interrupt Controller does not).

c. Which type of interrupt controller does the Zynq Cortex A9 have? In how many levels are the interrupt handling decisions taken?

The Zynq Cortex A9 uses a Programmable Interrupt Controller (non-vectored). Interrupt handling decisions are taken on 3 levels for dedicated interrupts:
1. Low-level interrupt handler
2. System interrupt handler
3. Execute user interrupt handler

d. Name and briefly explain 4 exceptions of the Zynq Cortex A9.

1. Undefined Instruction: caused by bad instruction (potentially corrupt memory).
2. Data Abort: occurs during a memory access violation.
3. FIQ: triggered by high-priority interrupt.
4. IRQ: triggered by normal priority interrupt.

**Problem 2.6: Interrupts**

a. Interrupts can be used in the Zynq Cortex A9 to inform the application of changes on a PsGPIO input. Describe the sequence of events from when the GPIO input value changes on the input pin to the interrupt handler clearing the interrupt at the PsGPIO.

1. Upon input meeting interrupt sensitivity, IRQ generated
2. Shared Peripheral Interrupt 20 forwarded to GIC
3. Passes input mask on GIC
4. IRQ ID #52 generated
5. Mapped to IRQ on core
6. Passing I mask in PSR
7. Processor switches to IRQ mode
8. Call 0x18 vector -> FreeRTOS_ApplicationIRQHandler
9. Vector table -> gpio_isr
10. Interrupt handler clears interrupts at PsGPIO

b. What would happen if the user does not clear the interrupt?

If a user does not clear the interrupt, the interrupt handler function will be called again.