CS M151B: Homework 6

Jonathan Woong

804205763

Spring 2017

Discussion 1A

Wednesday 17th May, 2017

Problem 1. A disk has an average seek time of 4.5 ms and rotates at 7200 RPM. There are 900 sectors per track. How long (in seconds) does it take to read the data from two consecutive sectors on the same track? Explain.

$$\text{average rotational time} = T_R = \frac{7200\ rev}{min} \times \frac{1\ min}{60\ sec} \times \frac{1}{2} = 0.0042\ sec$$

$$\text{time to read one sector} = T_S = \frac{1\ min}{7200\ track} \times \frac{1\ track}{900\ sector} \times \frac{60\ sec}{1\ min} = 9.2593 \times 10^{-6} sec$$

$$\boxed{\text{total time} = T_R + \text{average seek time} + 2(T_S) = 0.0042 + 0.0045 + 2(9.2593 \times 10^{-6}) = 0.0087\ sec}$$

Problem 2. Consider a system where all I/O is done using programmed I/O. The I/O device, CPU, and memory are connected to the same data bus. The data bus is 56 bits wide. It takes 8 nanoseconds for one transfer across the bus. What is the maximum possible bandwith (in bytes per second) at which data can be trasferred from the I/O device to memory? Explain.

$$\left(56\ bits \times \frac{1\ byte}{8\ bits}\right) \times \left(\frac{1\ transfer}{8\ ns} \times \frac{1 \times 10^9\ ns}{sec}\right) = \boxed{\frac{8.75 \times 10^8\ bytes}{sec}}$$

Problem 3. A program repeatedly performs a three-step process: it reads in a 4 KB block of data from disk, does some processing on that data, and then writes out the result as another 4 KB block elsewhere on the disk. Each block is contiguous and randomly located on a single track on the disk. The disk drive rotates at 7200 RPM, has an average seek time of 5.2 ms, and has a transfer rate of 95 MB/sec. The controller overhead is 0.3 ms. No other program is using the disk or processor, and there is no overlapping of disk operation with processing. The processing step takes 25 million clock cycles, and the clock rate is 2.8 GHz. What is the overall speed of the system in blocks processed per second?

$$T_R = \frac{7200\ rev}{min} \times \frac{1\ min}{60\ sec} \times \frac{1}{2} = 0.0042\ sec$$

$$\text{time to read 4 KB} = T_4 = \frac{4\ KB}{\frac{95\ MB}{sec}} = 4.2 \times 10^{-5} sec$$

$$\text{total read time} = T_R + \text{seek time} + \text{overhead} + T_4 = 0.0042 + 0.0052 + 0.0003 + 0.000042 = 9.7421 \times 10^{-3}$$

$$\text{time to process data} = T_P = \frac{2.5 \times 10^7 cycles}{2.8\ GHz} = 8.928 \times 10^{-5} sec$$

$$\text{time to write to disk} = T_R$$

$$\text{time to process 1 block} = 2(T_R) + T_P = 2(9.7421 \times 10^{-3}) + 8.928 \times 10^{-5} = 1.957 \times 10^{-2} sec \rightarrow \boxed{\frac{510\ blocks}{sec}}$$

Problem 4. An important advantage of interrupts over polling is the ability of the processor to perform other tasks while waiting for communication from an I/O device. Suppose that a 2.4 GHz processor needs to read 2000 bytes of data from a particular I/O device. The I/O device supplies 1 byte of data every 0.03 ms.

The code to process the data and store it in a buffer takes 1100 cycles. Clearly state your assumptions.

A) If the processor detects that a byte of data is ready through polling, and a polling iteration takes 60 cycles, how many cycles does the entire operation take?
Assume that the processor processes data before polling for the next byte.

$$\#\text{ of polling cycles} = 0.03\ ms \times 2.4\ GHz - 1100\ cycles = 70900\ cycles$$

$$\#\text{ of polls} = \frac{70900\ cycles}{\frac{60\ cycles}{poll}} = 1181.67\ polls$$

$$\#\text{ cycles spent polling} = 1182\ polls \times 60\ cycles = 70920\ cycles$$

$$\#\text{ cycles per byte} = 70920\ cycles + 1100\ cycles = 71920\ cycles$$

$$\boxed{\text{total operation time} = 71920\ cycles \times 2000\ bytes = 1.4384 \times 10^8\ cycles}$$

B) If instead, the processor is interrupted when a byte is ready, and the processor spends the time between interrupts on another task, how many cycles of this other task can the processor complete while the I/O communication is taking place? The overhead for handling an interrupt is 210 cycles.

$$\#\text{ cycles to process one byte} = 210 + 1100 = 1310\ cycles$$

$$\#\text{ cycles on other task for each byte} = 0.03\ ms * 2.4\ GHz - 1310\ cycles = 70690\ cycles$$

$$\boxed{\text{total operation time} = 70690\ cycles \times 2000\ bytes = 1.14138 \times 10^8\ cycles}$$

Problem 5. A pipelined MIPS implementation with support for overflow exceptions is described in the book, pp. 327-331. Consider the following claim: The description in the book is incomplete since it fails to modify the operation of the *Forwarding unit* when an exception occurs.

A) Is the claim above corect?
Yes.

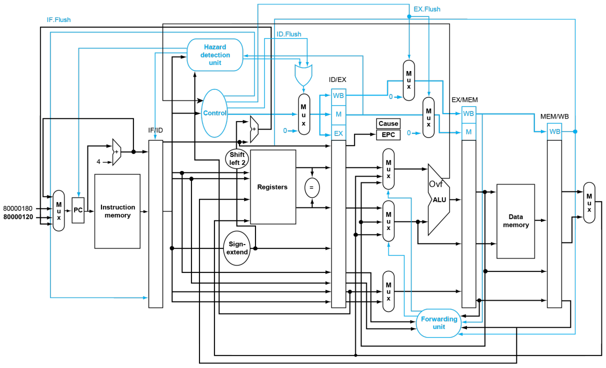
B) Explain your answer to Part A.
The forwarding unit may attempt to forward data to registers after an interrupt has occured. This can be fixed if the forwarding unit is notified of the interrupt.

Problem 6. Consider the pipelined MIPS implementation shown in slides 9.30-9.32 and explained in the book, pp. 327-331. Your task is to enhance this implementation to add support for the illegal opcode exception. As described on page 327, the *Cause* register must be modified based on whether the exception is an illegal opcode (value 10) or an overflow (value 12). Your implementation must work correctly even for the case where one instruction causes an overflow while the very next instruction contains an illegal opcode.

A) Provide a list of your modifications, where each one of the modifications is described in 1-2 clear sentences.

- Add new exception vector address 8000 0120 (value 10).
- Add input 8000 0120 to the MUX that outputs to the PC register. This is used as the initial address to begin fetching instructions in the event of illegal opcode.

B) The existing datapath implementation is shown on slide 9.30. Are any modifications to this figure required? If so, you can show them on a copy of the figure from slide 9.30.



C) Are any new control signals required? If so, list them with an explanation and identify them on the datapath diagram.

No new control signals are required.

D) Changes are required to the main *Control* circuit. Show those changes using a table similar to the one shown on slide 9.32. The contents of the table you provide must reflect correct operation when there are no exceptions, when there is an overflow exception, when there is an illegal opcode exception, and when overflow and illegal exceptions are detected simultaneously.

overflow	opcode	IF.Flush	ID.Flush	EX.Flush	orig signals	EPCWrite	Cause Write	Cause SRC
0	illegal	1	1	1	X	1	1	0

Where an illegal opcode is one that cannot be decoded into an existing instruction.