CPSC 313

06W Term 2

Problem Set #7 - Solution

1. (a) $\frac{1}{185ps} = 5.41 \times 10^9$ cycles per second or 5.41 GHz
   (b) 5.41 Gops (instructions per second)
   (c) 5 * 185 = 925 ps per instruction
   (d) i. $\frac{1}{600ps} = 1.66 \times 10^9$ cycles per second or 1.66 GHz
   ii. 1.66 Gops (instructions per second)
   iii. 600ps per instruction
   (e) I would split the largest stage (175 ps) which would allow the clock period to decrease from 185ps to 160ps.
   (f) I would split the largest 2 stages (175 ps and 150ps) which would allow the clock period to decrease from 185ps to 110ps.
   (g) i. $\frac{1}{110ps} = 9.09 \times 10^9$ cycles per second or 9.09 GHz
   ii. 9.09 Gops (instructions per second)
   iii. 7 * 110ps = 770ps per instruction

2. (a) Causal `%ebx`
   (b) Causal `%ebx`
   (c) Causal `%ebx`
   (d) Anti `%ecx`
   (e) Output `(%eax)`
   (f) Control
   (g) Control
   (h) Control

3. While solving this question, it is imperative to understand that causal dependecies and control dependencies cannot be resolved by rewriting the code. The only two dependencies that can be resolved are the anti and output.

   - The anti dependency in part 2d can be resolved by rewriting the code as follows:

     ```
     rmmovl %ecx, 10(%eax)
     irmovl $10, %exx      # exx stands for any register
                           # different from ecx
     ```

     Further, all occurences of ecx in the code following the irmovl instruction will have to be replaced by exx.
   - The output dependency in part 2e can be resolved by removing the first instruction.

4. (a) This is a hazard in PIPE- because the value of ebx will be updated only in the write-back stage, whereas the value has to be read in the decode stage of the next instruction.
   (b) This is a hazard in PIPE-. Same reason as in (a).
   (c) This is also a hazard in PIPE-. Same reason as in (a).

(d) This is not a hazard in PIPE- because ecx is read in the decode stage and is written only in the write-back stage.

(e) Not a hazard, since each memory access occurs at only one stage of the pipeline.

(f) This is not a hazard and the architecture will correctly use valC to fetch the next instruction.

(g) This is a hazard, because the branch target will not be determined until the Execute stage, but a new instruction needs to be fetched in the Fetch stage.

(h) This is a hazard because the next instruction to be fetched can only be determined at the Memory stage.