1. Consider the following Y86 code fragment:

\begin{verbatim}
 irmovq  $3, %rax
 irmovq  $4, %rbx
 addq    %rax, %rbx
\end{verbatim}

(a) Explain carefully the data dependency problem for the naive pipelined implementation (without data forwarding) of the Y86 we’ve been discussing. Be sure to say at which stages of the naive pipeline, the values of %rax and %rbx are needed by the addq statement and at which they become available.

(b) Insert the minimum number of nop’s into the code to resolve the problem.

(c) Explain the introduction of stalls and bubbles into the pipeline to resolve the problem without explicit nop’s.

(d) Explain how adding data forwarding to the pipeline solves the problem. Be explicit about what values are forwarded and from what stages of the pipeline, and to which stage.
