11-1 from our book
11-2 from our book

The rest is from Computer Organizations book.

6.2 [10] §6.1> A computer architect needs to design the pipeline of a new microprocessor. She has an example workload program core with $10^6$ instructions. Each instruction takes 100 ps to finish.

a. How long does it take to execute this program core on a nonpipelined processor?

b. The current state-of-the-art microprocessor has about 20 pipeline stages. Assume it is perfectly pipelined. How much speedup will it achieve compared to the nonpipelined processor?

c. Real pipelining isn’t perfect, since implementing pipelining introduces some overhead per pipeline stage. Will this overhead affect instruction latency, instruction throughput, or both?

6.3 [5] §6.1> Using a drawing similar to Figure 6.5 on page 377, show the forwarding paths needed to execute the following four instructions:

```
add $3, $4, $6
sub $5, $3, $2
lw $7, 100($5)
add $8, $7, $2
```

![Program execution order diagram](image)

**FIGURE 6.5** Graphical representation of forwarding. The connection shows the forwarding path from the output of the EX stage of add to the input of the EX stage for sub, replacing the value from register $s0$ read in the second stage of sub.
6.4 [10] §6.1> Identify all of the data dependences in the following code. Which dependences are data hazards that will be resolved via forwarding? Which dependences are data hazards that will cause a stall?

\[
\begin{align*}
ad & d \; \$3, \; \$4, \; \$2 \\
\text{sub} & \; \$5, \; \$3, \; \$1 \\
lw & \; \$6, \; 200(\$3) \\
ad & d \; \$7, \; \$3, \; \$6
\end{align*}
\]

6.22 [5] §§6.4, 6.5> Consider executing the following code on the pipelined datapath:

\[
\begin{align*}
lw & \; \$4, \; 100(\$2) \\
\text{sub} & \; \$6, \; \$4, \; \$3 \\
ad & d \; \$2, \; \$3, \; \$5
\end{align*}
\]

How many cycles will it take to execute this code? Draw a diagram like that of Figure 6.34 on page 414 that illustrates the dependences that need to be resolved, and provide another diagram like that of Figure 6.35 on page 415 that illustrates how the code will actually be executed (incorporating any stalls or forwarding) so as to resolve the identified problems.
Program execution order (in instructions)

- lw $2, 20($1)
- and becomes nop
- add $4, $2, $5
- or $8, $2, $6
- add $9, $4, $2

Fig. 6.36