HW1 solutions

1.2 No.

1.4 Ambiguity.

1.15 Advantages of a higher level language: Fewer instructions are required to do the same amount of work. This usually means it takes less time for a programmer to write a program to solve a problem. High level language programs are generally easier to read and therefore know what is going on. Disadvantages of a higher level language: Each instruction has less control over the underlying hardware that actually performs the computation that the program frequently executes less efficiently.

NOTE: this problem is beyond the scope of Chapter 1 or most students.

1.16 Possible operations, data types, addressing modes.

1.17 An ISA describes the interface to the computer from the perspective of the 0s and 1s of the program. For example, it describes the operations, data types, and addressing modes a programmer can use on that particular computer. It doesn’t specify the actual physical implementation. The microarchitecture does that. Using the car analogy, the ISA is what the driver sees, and the microarchitecture is what goes on under the hood.

1.23 ISA’s don’t change much between successive generations, because of the need for backward compatibility. You’d like your new computer to still run all your old software.
3.2

IN = 1

OUT = 0

N Type

P Type

3.4

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A

B

C = 1

N Type

P Type
3.5

\[
\begin{array}{cccc}
A & B & C & \text{OUT} \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
\end{array}
\]

3.6 \quad C = A'; \quad D = B'; \quad Z = (C+D)' = (A'+B')' = A \cdot B

\[
\begin{array}{cccc}
A & B & C & D & Z \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1 \\
\end{array}
\]

3.7 There is short circuit (path from Power to Ground) when either A = 1 and B = 0 or A = 0 and B = 1.

3.9

\[
\begin{array}{cc}
A & B & \text{NOT(NOT(A) OR NOT(B))} \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

AND gate has the same truth table.
3.11 a. Three input And-Gate

Three input OR-Gate
b. (1) A = 1, B = 0, C = 0.

AND Gate

OR Gate
b. (2) A = 0, B = 0, C = 0

AND Gate

OR Gate
b. (3) $A = 1$, $B = 1$, $C = 1$

AND Gate

OR Gate
3.13 A five input decoder will have 32 output lines.

3.14 A 16 input multiplexer will have one output line (of course!). It will have 4 select lines.

3.15

<table>
<thead>
<tr>
<th>C_i</th>
<th>A</th>
<th>B</th>
<th>S</th>
<th>C_out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

A = 7, B = 11, A + B = 18.
In the above calculation, the result (S) is 2!! This is because 18 is too large a number to be represented in 4 bits. Hence there is an overflow - Cout[3] = 1.

3.18. (a)

(b)

(c)

(d) No. The carry is not being generated/propagated.
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>