1. Read textbook section 5.4. (Also, reread sections 5.5 and 5.6, especially 5.5.7.)
2. Work text problem 5-5.
3. Work text problem 5-23.
4. Print out and read the notes on PDP-11 addressing on the course webpage under the subdirectory Notes. Then work the following problems:
   a. Suppose a DEC PDP-11 instruction word fetched and placed in the IR contains the bit pattern:
      \[0001100100010001\]
      Just before this instruction is executed, relevant registers and memory locations contain the following:
      \[
      \begin{array}{ll}
      \text{[R1]} & 002000_s \\
      \text{[2000_s]} & 002002_a \\
      \text{[R2]} & 002002_s \\
      \text{[2002_s]} & 002004_a \\
      \text{[R3]} & 002004_s \\
      \text{[2004_s]} & 002006_a \\
      \text{[R4]} & 002006_s \\
      \text{[2006_s]} & 002010_a \\
      \text{[R5]} & 002010_s \\
      \text{[2010_s]} & 002012_a \\
      \text{[2012_s]} & 002000_a \\
      \end{array}
      \]
      Give the (octal) contents, immediately after execution of the instruction in the IR, of registers and any memory locations whose contents could have changed by execution of this instruction.
   b. Suppose instead that the PDP-11 instruction fetched into the IR is given by the bit pattern:
      \[0110100100010001\]
      and suppose the contents of registers and memory locations are the same as is given in part a. Now what are the octal contents of registers and memory locations changed by execution of this instruction?
5. Another PDP-11 problem: Suppose the IR contains the following bit pattern:
   \[0001001001001010\]
   and suppose that registers and memory locations contain the following:
   \[
   \begin{array}{ll}
   \text{[R0]} & 001000_s \\
   \text{[1000_s]} & 001000_a \\
   \text{[R1]} & 002000_s \\
   \text{[2000_s]} & 004000_a \\
   \text{[R2]} & 003000_s \\
   \text{[3000_s]} & 005000_a \\
   \text{[R3]} & 004000_s \\
   \text{[4000_s]} & 002000_a \\
   \text{[R4]} & 005000_s \\
   \text{[5000_s]} & 000000_a \\
   \end{array}
   \]
   Give the (octal) contents, immediately after execution of the instruction in the IR, of registers and any memory locations whose contents could have changed by execution of this instruction.
6. The DEC PDP-11 instruction word in memory location \(i\) is fetched and placed in the IR for decoding and execution; this word contains the bit pattern (shown in binary); also shown is the word that follows it in memory:
   \[
   \begin{array}{ll}
   \text{1110100001110011} & \text{[i]} = m \text{[word}\ i\ \text{]} \\
   \text{0000000000000110} & \text{[i+2]} = m \text{[word}\ i+2\ \text{]}
   \end{array}
   \]
   Just before this instruction is executed, relevant registers and memory locations contain the following (in octal shorthand):
   \[
   \begin{array}{ll}
   \text{[R0]} & 002014_s \\
   \text{[2000_s]} & 002002_a \\
   \text{[R1]} & 002002_s \\
   \text{[2002_s]} & 002004_a \\
   \text{[R2]} & 002004_s \\
   \text{[2004_s]} & 002006_a \\
   \text{[R3]} & 002006_s \\
   \text{[2006_s]} & 002010_a \\
   \text{[R4]} & 002010_s \\
   \text{[2010_s]} & 002012_a \\
   \text{[R5]} & 002012_s \\
   \text{[2012_s]} & 002000_a \\
   \text{[R6]} & 002000_s \\
   \text{[2014_s]} & 164163_a \\
   \text{[R7]} & 002016_s \\
   \text{[2016_s]} & 000000_a \\
   \end{array}
   \]
   Give the (octal) contents, immediately after execution of the above instruction and immediately before start of the fetch cycle for the next instruction, of only those registers and memory locations whose contents have been changed during execution of this instruction.
   Continued on page 2.
8. Programming Assignment 1: Consider the following main program and subroutine for the MAC-1 machine:

<table>
<thead>
<tr>
<th>/Main</th>
<th>/Subroutine sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTRN sum</td>
<td>ENTRY sum</td>
</tr>
<tr>
<td>ans1 RES 2</td>
<td>i EQU 1</td>
</tr>
<tr>
<td>ans2 RES 1</td>
<td>j EQU 2</td>
</tr>
<tr>
<td>start loco 4020</td>
<td>k EQU 3</td>
</tr>
<tr>
<td>swap sum ldl j</td>
<td></td>
</tr>
<tr>
<td>stod ans1</td>
<td>addl k</td>
</tr>
<tr>
<td>loco x1</td>
<td>push</td>
</tr>
<tr>
<td>push ldl j</td>
<td>popi</td>
</tr>
<tr>
<td>lodd x2</td>
<td>push</td>
</tr>
<tr>
<td>push ldl k</td>
<td>pop</td>
</tr>
<tr>
<td>loco ans1</td>
<td></td>
</tr>
<tr>
<td>push</td>
<td>psh</td>
</tr>
<tr>
<td>call sum</td>
<td></td>
</tr>
<tr>
<td>stod ans2</td>
<td></td>
</tr>
<tr>
<td>halt</td>
<td>retn</td>
</tr>
<tr>
<td>x1 -40</td>
<td>END</td>
</tr>
<tr>
<td>x2 19</td>
<td></td>
</tr>
</tbody>
</table>

END start

Add comments to each line of code in the main program and subroutine explaining what is taking place (and preferably why); don’t simply restate the obvious. Adjust the comments after you analyze the programs (using the simulator) so that they meaningfully describe the code.

Assemble, link, and simulate execution of the above main program and subroutine. Do two different simulations. Turn in printouts of the separate assemblies, linkages, and the snapshots of memory before and after execution in each case. Write up a summary of the differences in the addresses used in the two different absolute programs, and place comments on the symbolic source statements to describe what the program statements are accomplishing. Explain the final contents of locations ans1, ans1+1, and ans2.

Assuming that the main program is in file “main” and that the subroutine is in the file “sum”, be sure to do the following:

asem main
qpr -q “your favorite printer” main.list
asem sum
qpr -q “your favorite printer” sum.list
load main sum
qpr -q “your favorite printer” main.abs
sim main.abs $EE350/halt $EE350/halt.pascal
produce printout of before and after memory contents and highlight and explain the contents of ans1, ans1+1, and ans2.

Also, do the following: load sum main
qpr -q “your favorite printer” sum.abs
sim sum.abs $EE350/halt $EE350/halt.pascal
produce printout of before and after memory contents and highlight and explain the contents of ans1, ans1+1, and ans2.

Note: Be sure to edit the memory dumps to get rid of excess words with all zeros (however, be sure to keep both the code and stack sections of memory from both sets of dumps). Both sets of dumps should fit on one page with judicious editing and use of enscript.