1. Considere el siguiente programa de lenguaje de ensamblador Easy I y responda a las preguntas que siguen utilizando la tabla proporcionada:

| #Suma de numeros pares
| #Utilizando recursos demas
| andi 0
| addi 10
| storei 500  #int length = 10
| andi 0
| storei 502  #int num = 0
| andi 0
| storei 504  #int result = 0
| andi 0
| storei 506  #int i=0
| andi 0
| storei 508  #int j=0

loop: loadi 506
addi 1
storei 508  # j=i+1
loadi 508
comp
addi 1
add 500
brni end_loop  # !(j>length)
loadi 502  #result = num + result
add 504
storei 504
loadi 502  #num = num + 2
addi 2
storei 502
loadi 506  # i = i + 1
addi 1
storei 506
jumpi loop

end_loop:
end
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Cycles per Instruction</th>
<th>Times executed</th>
<th>Contribution to Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTALS</td>
<td>CPI =</td>
<td>Exec Time =</td>
<td></td>
</tr>
</tbody>
</table>

a. Calculate the number of cycles that each instruction takes to execute including the fetch and fetchop cycles  
b. Calculate the number of times that each instruction is executed  
c. Calculate the contribution of each instruction to the program execution time assuming a clock rate of 2GHz.  
d. Calculate the average CPI achieved by the program  
e. Calculate the total runtime of the program  
f. Provide a new equivalent version of the program that achieves lowest possible CPI by making

2. For each instruction in the following table show the changes that must be made to the Easy I implementation (datapaths, control unit flowcharts, control unit state transition diagram) discussed in class in order to incrementally support each of the following instructions.

<table>
<thead>
<tr>
<th>Symbolic Name</th>
<th>Assembler Example</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>BrNr – Branch on negative relative to PC</td>
<td>BrNr X</td>
<td>PC ← PC + 2 + X</td>
</tr>
<tr>
<td>XOR – Exclusive OR</td>
<td>XOR X</td>
<td>AC ← AC xor X</td>
</tr>
<tr>
<td>NEG – Negate</td>
<td>NEG</td>
<td>AC ← - AC</td>
</tr>
<tr>
<td>RET - Return</td>
<td>Ret X</td>
<td>PC ← MEM[X-2]</td>
</tr>
</tbody>
</table>
| CALL – Call function | Call X | MEM[X-2] ← PC+2  
PC ← X |

3. Complete the necessary changes to the Easy I implementation in order to make the cycle time independent of the memory cycle time. TO accomplish this you need to add a loop to every cycle accessing memory. The control unit remains in the memory access cycle until the memory ready signal is received from the memory module.  
4. Complete the necessary changes to the Easy I Control Unit state transition table to incorporate the indirect addressing mode (FetchOp cycles).  

5. All the problems on Chapters 3 and 4 of Patterson and Hennessy Computer Organization and Design.