Consider the following loop and its equivalent assembly code:

```c
int total = 0;
for (int i = 0; i < n; i = i+1) {
    total = total + a[i];
}
```

```assembly
// x10:total, x11:a[i], x12:a[n]
loop:    lw x13, 0(x11)
        addi x11, x11, 4
        add x10, x10, x13
        bne x11, x12, loop
        sub x20, x21, x22
        slli x23, x24, 1
```

The loop above has been executing for a while on our standard 5-stage pipelined RISC-V processor with branch annulment and full bypassing.

**Fill in the pipeline diagram** below for cycles 500-508 assuming that at cycle 500 the lw instruction is fetched. Also, assume that the bne branch is taken. How many cycles does each iteration of the loop take?

<table>
<thead>
<tr>
<th>Cycle #</th>
<th>500</th>
<th>501</th>
<th>502</th>
<th>503</th>
<th>504</th>
<th>505</th>
<th>506</th>
<th>507</th>
<th>508</th>
</tr>
</thead>
<tbody>
<tr>
<td>IF</td>
<td>lw</td>
<td>addi</td>
<td>add</td>
<td>bne</td>
<td>bne</td>
<td>sub</td>
<td>slli</td>
<td>lw</td>
<td>addi</td>
</tr>
<tr>
<td>DEC</td>
<td>NOP</td>
<td>lw</td>
<td>addi</td>
<td>add</td>
<td>add</td>
<td>bne</td>
<td>sub</td>
<td>NOP</td>
<td>lw</td>
</tr>
<tr>
<td>EXE</td>
<td>NOP</td>
<td>NOP</td>
<td>lw</td>
<td>addi</td>
<td>NOP</td>
<td>add</td>
<td>bne</td>
<td>NOP</td>
<td>NOP</td>
</tr>
<tr>
<td>MEM</td>
<td>bne</td>
<td>NOP</td>
<td>NOP</td>
<td>lw</td>
<td>addi</td>
<td>NOP</td>
<td>add</td>
<td>bne</td>
<td>NOP</td>
</tr>
<tr>
<td>WB</td>
<td>add</td>
<td>bne</td>
<td>NOP</td>
<td>NOP</td>
<td>lw</td>
<td>addi</td>
<td>NOP</td>
<td>add</td>
<td>bne</td>
</tr>
</tbody>
</table>

**Cycles per loop iteration (in steady state): ____7____ cycles**