The following assembly language (MIPS) function `toupper` converts any lowercase characters (with ASCII codes between 97 and 122) in the null-terminated argument string to uppercase. (Assume that a0 initially stores the starting address of the string)

```
toupper:
    lb $t2, 0($a0)           # Stop at end of string
    beq $t2, $0, exit       # Not lowercase
    blt $t2, 97, next       # Not lowercase
    bgt $t2, 122, next      # Not lowercase
    sub $t2, $t2, 32        # Convert to uppercase
    sb $t2, 0($a0)          # Store back in string

next:
    addi $a0, $a0, 1
    j toupper

exit:
    jr $ra
```

ASCII: American Standard Code for Information Interchange, used by computer to represent characters.

<table>
<thead>
<tr>
<th>Assembly Language Instructions</th>
<th>Example</th>
<th>Meaning</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data transfer</td>
<td>lb $r1, 100($r2)</td>
<td>$r1=Memory[$r2+100]</td>
<td>Load byte from memory</td>
</tr>
<tr>
<td></td>
<td>sb $r1, 100($r2)</td>
<td>Memory[$r2+100]=$r1</td>
<td>Store byte to memory</td>
</tr>
<tr>
<td>Arithmetic</td>
<td>addi $r1, $r2, 100</td>
<td>$r1=$r2+100</td>
<td>Add immediate</td>
</tr>
<tr>
<td></td>
<td>sub $r1, $r2, 100</td>
<td>$r1=$r2-100</td>
<td>Subtract</td>
</tr>
<tr>
<td>Conditional Branch</td>
<td>beq $r1, $r2, label</td>
<td>If($r1===$r2) PC=label</td>
<td>Branch on equal</td>
</tr>
<tr>
<td></td>
<td>blt $r1, 100, label</td>
<td>If($r1&lt;100) PC=label</td>
<td>Branch less than</td>
</tr>
<tr>
<td></td>
<td>bgt $r1,100, label</td>
<td>If ($r1&gt;100) PC=label</td>
<td>Branch greater than</td>
</tr>
<tr>
<td>Unconditional Jump</td>
<td>j label</td>
<td>PC=label</td>
<td>Jump to target address</td>
</tr>
<tr>
<td></td>
<td>jr $ra</td>
<td>PC=$ra</td>
<td>Jump register</td>
</tr>
</tbody>
</table>

Assume that this function is called with a string that contains exactly 100 lowercase letters, followed by the null terminator.

(1) How many instructions would be executed for this function call? (10%)

(2) Assume that we implement a single-cycle processor, with a cycle time of 8ns. How much time would be needed to execute the function call? What is the CPI (cycle per instruction)? (25%)

(3) Now assume that our processor uses a 5-stage (e.g. fetch, decode, execution, memory access, write back) pipeline, with the following characteristics:
• Each stage takes one clock cycle.
• The register file can be read and written in the same cycle.
• Assume forwarding is done whenever possible, and stalls are inserted otherwise.
• Branches are resolved in the ID stage and are predicted correctly 90% of the time (assume that all mispredictions have the same performance penalty).
• Jump instructions are fully pipelined, so no stalls or flushes are needed.

How many total cycles are needed for the call to `toupper` with these assumptions? (50%)

(4) If the cycle time of the pipelined machine is 2ns, how would its performance compare to that of the single-cycle processor from (2)? (15%)
A MIPS processor has a 16KB, 4-way set-associative (i.e., each set consists of 4 blocks) data cache with 32-byte blocks. Here, a “KB” is $2^{10}$ bytes.

(1) How many total blocks are in the Level 1 cache? How many sets are there? (20%)

(2) Assuming that memory is byte addressable and addresses are 35-bits long, give the number of bits required for each of the following fields: tag, set index, block offset. (15%)

(3) What is the total size of the cache, including the valid, tag and data fields? Give an exact answer, in either bits or bytes. (15%)

Assume that the cache communicates with main memory via a 64-bit bus that can perform one transfer every 10 cycles. Main memory itself is 64-bits wide and has a 10-cycle access time. Memory accesses and bus transfers may be overlapped.

(4) What is the miss penalty for the cache? In other words, how long does it take to send a request to main memory and to receive an entire cache block? (20%)

(5) If the cache has a 95% hit rate and a one-cycle hit time, what is the average memory access time? (15%)

(6) If we run a program which consists of 30% memory access instructions (e.g. loads or stores), what is the average number of memory stall cycles per instruction? (Assume that cache miss rate is 10% and miss penalty is 100 cycles) (15%)