Mid-Terms answers

Results:

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVERAGE</td>
<td>71.1875</td>
</tr>
<tr>
<td>MAX</td>
<td>92</td>
</tr>
<tr>
<td>MIN</td>
<td>36</td>
</tr>
<tr>
<td>STDEV</td>
<td>16.17907599339334</td>
</tr>
</tbody>
</table>

Answers:

1. 
   (a) Option 1: 118 Cycles  
       Option 2: 121 Cycles  

   (b) Option 1: 118/17 = 6.94… Cycles  
       Option 2: 121/20 = 6.05… Cycles  

   (c) Better option is 1 by a ratio of 121/118 = 1.0254… (2~3%)  
       Average CPI is not important, only the overall cycle time

2. 
   (a) Arithmetic left shift is in effect identical to logical left shift. The reason is thanks to two’s complement, where the sign bit is replicated in the upper bits, logical left shifts will maintain the correct sign bit.  
   (b) Loading requires signed/unsigned variants much like shifts require arithmetic and logical variants. When loading a value smaller than 4 bytes into a 4-byte register, the program must tell the hardware whether to interpret this value as a signed or an unsigned value. Specifically, whether to
detect and extend the sign bits as per two’s complement. Writes do not need a signed/unsigned variant because two’s complement means simple truncation maintains the sign bit, and thus the correct value.

(c) Due to the fixed amount of space (4 bytes) used to encode an instruction, the size of an immediate value that can be encoded for addi is limited to 12 bytes. As a result, loading larger values requires a combination of instructions including LUI.

3.
(a) No. The execute stage is too large, resulting in an unbalanced pipeline with a slow clock. While the combinational circuit of the execute stage is still busy with propagation, fetch and writeback stages will be done and idling.
(b) No. Because memory access needs to happen between fetch and execute, as well as between execute and writeback. As a result, at least these three stages are required. While technically any of these stages can be merged into a single large combinational circuit, this circuit will need to have a long enough combinational path (and as a result, a slow enough clock) to accommodate the memory access latency, which would be an extremely ineffective architecture.
(c) The new pipeline will be roughly 20% more performant, as the clock cycle for the new processor will be limited by the 5 ns decode stage, whereas the old processor was limited by the 6 ns decode+execute stage. New throughput: $1/5 = 0.2$, old throughput: $1/6 = 0.166…$. Improvement: 1.2.

4.
(a) Decode and Execute stages will need to be changed. Decode stage will need to be changed to recognize the new instruction encoding, and Execute to actually perform the multiplication.
(b) Major changes will be in fetch and execute stages. Fetch stage will obviously have to change to use the prediction to fetch, and the decode stage to verify the prediction made by the branch predictor is correct. If the prediction is discovered to be incorrect in the execute stage, the instructions currently in the pipeline due to the incorrect prediction will need to be ignored. One way to do this is by using the epoch timestamp. The fetch stage tags each instruction with the current epoch, and the execute stage increments it in the case of an incorrect prediction.
(c) t3 will hold 1, instead of the expected value of 3. This is because by the time addi t3, t0, t2 is in the decode stage, only the very first instruction, addi t0, zero, 1 has been retired, updating the register file. Assuming all registers are initialized to zero, t3 will hold $0 + 1$, which is 1.
(d) Forwarding takes the results from the execute stage and makes them available to the decode stage immediately, so that the processor will not have to stall.
However, forwarding just from execute to decode results in removing the dependency between two back-to-back instructions, such when an instruction in the decode stage depends on the results of an instruction in the execute stage. For example, if an instruction in the decode stage depends on an instruction in the memory stage, or writeback stages, forwarding from execute to decode does not help performance, and will still have to stall. In order to remove all such dependencies, forwarding paths from memory and writeback stages can also be added. However, this also adds on-chip resource utilization and increases the critical path.

5.
(a) If the two stages can be overlapped, there will be minimal to no change in cycle time as they take separate paths in the circuit.
(b) It will likely slow the program execution down, by executing more instructions. If the program as a series of instruction that reads from fixed offsets from a memory location (reading from the stack, for example), they now require more instructions to actually calculate the memory address before executing the memory request instructions.
6.
(a) RISC has a much simpler ISA with a smaller number of instructions, and depends on the
programmer or compiler to implement complex functionalities from the simpler instructions. CISC
such as x86 has a much larger number of more complex instructions. Programs on a CISC paradigm
typically require a fewer number of instructions to execute the same functionality, but because a CISC
processor design is typically much more complex compared to RISC ones, each RISC instruction can
be made to execute faster (typically).
The complex organization of CISC processors make it difficult to incorporate from modern
microarchitectural concepts such as out-of-order or superscalar. As a result, modern CISC processors
typically have an internal RISC-paradigm core, which a hardware decoder translates the CISC ISA into
dynamically.

(b) Programs that have many hazards, or short-distance dependencies.

(c) We don’t know how these instructions are interleaved. If instructions can be organized by the
compiler such that they can be pipelined efficiently, then CPI can be overlapped. As a result,
instructions with high CPI won’t effect program performance. On the other hand, if they are not well
overlapped and data hazards cause stalling, they can cause performance degradation. However, it is
impossible to tell from the given information.