1) If a BEQ instruction would cause the PC to go into instruction memory outside of the program, what should happen? (For instance, if I have a 5 line program with a line "beq, $0, $0, 500", this is clearly going to try to go to a line that is past the last instruction in the instruction memory). Does the program exit with an error immediately? Does it just treat that as a halt instruction?

You are writing a parser, which handles instructions line by line.  At the time it parses the beq instruction, it has no knowledge of what the remaining part of the program looks like.  It cannot therefore exit with an error or treat it as a halt.  It simply causes a branch to the indicated location in Instruction Memory.  While executing, the machine will go to that location; most likely that location will contain an illegal opcode which will trigger a halt.  If it contains an actual instruction, then the machine will execute that instead.

2) Are the cycles done by IF when it fetches the haltSimulation instruction considered 'useful'?

No: haltSimulation is a simulator directive, not a MIPS instruction.

3) Let's say my memory access time (c) is 100 cycles, and my normal operation time (n) is 5 cycles. I now run a program that has a LW followed by an ADD instruction. Clearly the MEM stage is going to be stalling the program. While the EX stage waits for the MEM stage to clear the EX\_MEM latch, can the EX stage do it's 5 useful cycles of work, and then idle while it waits for MEM? Or does the EX stage have to wait for it's output latch (the EX\_MEM latch) to be emptied out before it can start doing its useful cycles?

EX should not wait for its output latch to clear before it can start.  Instead, it should simply hold that instruction until EX\_MEM has been cleared out by MEM after the 100-cycle lw has been done by that stage.  So, if EX starts work on the subsequent instruction 1 cycle after the lw instruction, it will be done before the lw is done at MEM and have to wait there (holding up the next instruction in the ID stage) until the EX\_MEM latch clears.

4) Once the IF stage fetches a haltSimulatoin instruction, when should the simulation stop? Does the halt need to be passed all the way through to the WB stage? Or can the simulation stop earlier if none of the stages have any data in their latches?

haltSimulation will go through the pipeline like any other instruction, shutting off each stage as it passes through it.  So, WB will turn off last of all.

5) In the lab description it says that multiplication time is m cycles. Does that mean that it takes m cycles for multiplication to go through the EX stage or does this involve any other stages?

This is the time it takes for the multiplication to be executed by the EX stage.

6) Do you want us to display the Utilization of IF, ID, EX, MEM, and WB in terms of clock cycles or in percentages of utilized clock cycles divided by Exec Time in clock cycles.

In percentages of utilized clock cycles divided by the total program execution time, in clock cycles.

7) If a BEQ would cause the PC to either be negative or go beyond the limit of the instruction memory array (not just past the end of the valid instruction memory, but outside the instruction memory entirely), would this also result in the fetching of an illegal opcode (which is just treated as a halt)?

This should be treated as illegal and cause the simulator to stop.

8) When each stage is passing the HALT instruction along, is this considered useful work?

No; this is a simulator directive, not a MIPS instruction.  It does not even count in the total cycles count.

9) When the HALT instruction is being passed through the EX stage, does it take only 1 cycle, or does it take n cycles (like ADD, ADDI, SUB, etc.)?

Just one cycle.

10) The total execution cycles should include the cycles needed to propagate the HALT instruction through the pipeline to the WB, right?

No; we only count until the last MIPS instruction has left the pipeline.

11) When the IF stage is fetching the haltSimulation directive, does it still take c cycles to fetch?

No, just one cycle. It is not part of the MIPS simulation; just a simulator directive to tell the simulator when each stage should be turned off.

12) We had started to use multiple C files with a Makefile to try and divide concerns. Will we need to put this in 1 file for submission or is a tar archive ok?

No tar files: just concatenate all the files together into a single text file and submit it.