Part 1

The computer.hdl file is written to execute the hack computer. The main parts of the hack computer are;

- Instruction memory (ROM)
- Memory (RAM)
    - Data memory
    - Screen
    - Keyboard
- CPU

ROM32K (address=pc, out=instruction);

This line declares a ROM32K component with two connections: "address" and "out". The "address" connection is associated with the "pc" (program counter), and the "out" connection is associated with the "instruction". This line essentially represents the ROM that stores the program instructions.

CPU (inM=memOut, instruction=instruction, reset=reset, outM=outM, writeM=writeM, addressM=addressM, pc=pc);

This line declares a CPU component with multiple connections. It connects the ROM's output ("instruction") to the CPU, along with other signals such as "reset", "inM", "outM", "writeM", "addressM", and "pc". This line represents the central processing unit of the computer.

Memory (in=outM, load=writeM, address=addressM, out=memOut);

This line declares a Memory component with connections to the CPU. It connects the CPU's output ("outM") to the Memory, along with signals such as "in", "load", and "address". This line represents the memory unit of the computer.

Part 2

Instruction 2: 1110110000010000

This is a C instruction as the opcode is 1. The binary representation of this instruction is 1110 1100 0001 0000. The destination is A register (111 in the destination field). The computation is A + D (110000 in the computation field). The jump condition is JMP (000 in the jump field). The ALU computes the sum of A and D (A + D), and the result is stored in the A-register.

Instruction 3: 0000000000000011

As the opcode (the first bit) is 0, this is an A instruction. The symbolic syntax for A instruction is @value. In this case, the given instruction can be expressed as @3. The A-register is set to 3. This is done by setting the load bit (the control bit) in the A register to 1. The A-register puts 3 onto the address bus. This is used to access RAM[3]. The value of RAM[3] is put onto the data bus and is output as the M register (M input).

Instruction 4: 1110000010010000

This is a C instruction as the opcode is 1. The binary representation of this instruction is 1110 0000 0100 1000. The destination is A register (111 in the destination field). The computation is A - D (000100 in the computation field). The jump condition is JMP (000 in the jump field). The ALU computes the difference of A and D (A - D), and the result is stored in the A-register.

Instruction 5: 0000000000000000

As the opcode (the first bit) is 0, this is an A instruction. The symbolic syntax for A instruction is @value. In this case, the given instruction can be expressed as @0. The A-register is set to 0. This is done by setting the load bit (the control bit) in the A register to 1. The A-register puts 0 onto the address bus. This is used to access RAM[0]. The value of RAM[0] is put onto the data bus and is output as the M register (M input).

Instruction 6: 1110001100001000

This is a C instruction as the opcode is 1. The binary representation of this instruction is 1110 0011 0000 1000. The destination is M register (001 in the destination field). The computation is D (010000 in the computation field). The jump condition is JMP (000 in the jump field). The ALU computes the value in the D register, and the result is stored in the M register, effectively writing the result to RAM[0].

So, the series of instructions loads values into the A-register, performs computations using the ALU, and writes the result to RAM[0]. The specific computations involve adding values from RAM[2] and RAM[3] and then storing the result in RAM[0].