1) Assume that we have the following values in the given memory locations:

Location	Value
1000	1300
1100	1200
1200	800
1300	1200

Also assume that the base register R1 stores 200 and is always implicitly used for the indexed addressing mode. What datum is loaded into the accumulator if the instruction is LOAD 1000 for each of the following addressing modes?

a. Immediate
   1000

b. Direct
   1300

c. Indirect
   1200

d. Base (Indexed)
   800

2) A computer has a fixed length 32 bit instruction size. The computer has 87 instructions. All instructions are of the format: op code, mode, operand1, operand2, operand3 or opcode, mode, operand1, operand2 or opcode, mode, operand. There are 6 addressing modes available. The 3 operand mode only allows registers. The two operand mode permits a register and an address or a register and an immediate datum (in 2's complement). The one operand mode permits only an address. Answer the following questions.

   a. If the machine has 64 registers, what is the largest immediate datum that can be specified?

      7 bits for the opcode, 3 bits for the addressing mode, leaves 22 bits for the 2 operands (1 register, 1 immediate datum). With 64 registers, we need 6 bits for the register. This leaves 16 bits for the immediate datum. Since the immediate datum is in two's complement, the largest will be 2^{15}-1 or 32767.

   b. If the machine instead had 256 registers, would we still be able to use the 3 register mode? Why or why not?

      7 bits for the opcode, 3 bits for the addressing mode, 8 bits for each of the 3 registers gives us 7 + 3 + 24 = 34 bits, no we do not have enough space for the 3 register mode.

   c. What is the biggest memory address than can be specified?

      The 1 operand mode with just the address would be the mode that allows us to provide the largest address, so we will use it. We have 7 bits for the opcode, 3 bits for the addressing mode leaving 22 bits for an address. So the biggest address would be 2^{22} -1 = 4M – 1 (or just 4M).

   d. If we allowed for a memory address of 1M, how many registers could we address using the 2 operand mode (1 register, 1 address)?

      We use 7 bits for the opcode and 3 bits for the addressing mode leaving 22 bits for the register and memory address. An address of up to 1M will use 20 bits leaving just 2 bits for registers, so we could only address up to 4 registers.

3) Assume our computer has a 32 bit bus and 32 bit Instruction Register to store an instruction. What are the difficulties in using the PDP-11 instruction set if we want to pipeline our computer? What are the difficulties in using the Intel Pentium instruction set?

   The PDP-11 instructions range from 16 to 48 bits. If we fetch exactly 32 bits at a time, we may fetch 1, 2 or 2/3s of an instruction. We can fit the entire op code into the 32 bit

IR to decode it, but we may need to perform an additional fetch of 16 bits to get more operand information if the instruction uses format 13. Therefore, we may need to stall the pipeline occasionally.

For the Intel Pentium, instructions can range from 1 byte to 17. If the prefix uses 4 bytes, then this will fill the entire IR and we would have to decode that before we could even fetch the real op code. So decoding could very well take more than 1 cycle to complete, impacting our pipeline. Fetching can take from 1 cycle (1-4 bytes) up to 5 cycles (17 bytes) definitely impacting our pipeline's performance. So, both the IR may be too small and the fetching may be impacted.

4) Two computers have 5-stage fetch-execute cycles where branches are determined in stage 4. One computer is not pipelined, and the other is pipelined. Assuming that $t_p = 1$, answer the following questions when running a program with 50,000 instructions where 1,000 of the instructions are branches and each branch skips over 10 instructions.
   a. How much faster is the pipelined machine over the non-pipelined machine assuming that no branches are taken.

      If no branches are taken, then the pipelined machine takes n + k – 1 cycles or 50,000 + 5 – 1 = 50,004 cycles. The non-pipelined machine takes n * k cycles or 50,000 * 5 = 250,000 cycles. The pipelined machine is 250,000 / 50,004 = 4.9996 times faster (almost 5 times faster).

   b. How much faster is the pipelined machine over the non-pipelined machine assuming that all branches are taken.

      If all branches are taken, then n drops from 50,000 to 50,000 – 1,000 * 10 = 40,000. So our non-pipelined machine's performance becomes n * k = 40,000 * 5 = 200,000 cycles. The pipelined machine has fewer instructions, but each branch accrues a branch penalty. Since branches are computed in stage 4, the branch penalty is 3 cycles. Our new formula is n + k – 1 + b * 3 where b is the number of branches. So we have 40,000 + 5 – 1 + 1,000 * 3 = 43,004. Our pipelined machine is still much faster by a factor of 200,000 / 43,004 = 4.6507 but the speeds are getting (slightly) closer.