RISC and Performance

I. RISC = Reduced Instruction Set Computers
   A. Principles
      1. Streamline hardware, especially complex functions that slow clocks or complicate pipelining.
         a) Optimize frequently used commands.
         b) Reduce in RAM accesses by increasing register usage.
      2. Use extensive simulations of RISC architectures and HLL programs. Retain those features that demonstrate
         quantitatively a net performance gain.
         a) Roughly half of all hll instructions are assignment statements.
         b) The vast majority of arithmetic expressions involve a single operation.
         c) Conditional jumps form the next most common group of instructions.
         d) Calls to procedures are also quite common. Most calls involve relatively few parameters. Deep nesting of
            procedures with little intermediate work is rare. The majority of variable references within called procedures
            are to local variables.
         e) Most instructions employ the simpler addressing modes.
      3. Depend on compiler to convert hll to routines that use simple instructions.
   B. Common Properties:
      1. Most instructions execute in a single clock cycle.
      2. Simple load/store architecture. Only LOAD and STORE instructions may access memory. No use of memory in
         other type of instructions, such as ADD in Intel.
      3. Orthogonal register set. Any register may be used for any operation.
      4. Every instruction has the same length.
      5. Only those instructions which are truly justified in terms of performance/space/pipeline tradeoffs.
         a) SPARC has only 32 instructions with no multiply.
      8. Hardwired implementations, not microcoded.

II. Beneficial Effects for compiler writers.
   A. Hard to match complex instructions to their source code, e.g. VAX poly instruction.
   B. Limiting roles of registers in CISC makes it harder.
   C. However, RISC does have delay slots that must be filled.

III. Performance
   A. Time per Task = T * E * I where T = Time per Cycle (clock speed); E = Execution rate of Instructions; I =
      Instructions per Task.
      1. RISC tries to minimize the first two factors (E & T) to save time.
      2. CISC tried to minimize I to save space. Since space is no longer at a premium this factor is irrelevant for most
         applications.
   B. Execution Rate of Instructions = Instructions completed per cycle.
      1. In CISC, this varies based on the instruction-simple instruction take fewer cycles than complex instructions. This
         makes for very efficient use of time in non-pipelined CPU.
      2. In RISC, the goal is to have an execution rate of one or better by using instruction pipelines.
   C. Instruction Pipelines overlaps the execution of multiple instructions by dividing the execution of each instruction
      into discrete portions.
      1. A simple four-stage pipeline could break the instruction cycle into four parts: 1) (F) fetch instruction, 2) (A) ALU
         operation, 3) (M) Memory access, and 4) (W) Write results.
      2. To determine the time per cycle, you survey the duration of all stages for all instructions and use the duration of
         the longest stage found. To work efficiently the time required to execute each instruction sub-part should be
         approximately equal.
      3. RISC techniques for efficient pipelines:
         a) Load/Store architecture. Since M stage for main memory access takes longer than other instructions, RISC
            CPUs have many registers to minimize the number or reloads of values from memory.
         b) Main memory is only accessed through special load/store instructions. Delayed Load Instructions do not have
            their operand available soon enough in the M stage for the A stage of the next instruction in one cycle. So
            next instruction must not depend on that operand, else use a NOP.
c) Delayed Branch Instructions handle problem of the A stage result of a branch or jump instruction (the address of the instruction to jump to) is unknown when the next instruction is in its F stage. Delayed branch instructions provide this to the second instruction after the branch, so the slot immediately following the branch must be filled with an instruction that will be done regardless of the branch, else use a NOP.

4. Time Per Cycle,
   a) In unpipelined there is no need to set to any faster than quickest instruction.
   b) In pipelined, it is set to the duration of the longest stage.
      (1) Fetching is helped with caches. Address caches facilitate reading ahead. Uniform instruction size of RISC means uniform (and hence efficient) fetch time.
      (2) ALU stage can take a long time in CISC instructions because of the many addressing modes and the time to decode and execute some complex operations. RISC has fewer addressing modes and no complex instructions so A stage duration is more uniform.
      (3) M stage in CISC may have to accommodate to main memory accesses. Delayed Load takes care of the class of instructions that would cause a longer M stage.

IV. Additional MIPS stuff:
   A. Pseudoinstructions converted by assembler
   B. Distance field in jump is \( \frac{1}{4} \) the jump because addresses are ALWAYS four apart.