1.) In the above table, which processor has the highest performance?

2.) If the processors in the above table each execute a program in 10 seconds, find the number of cycles and the number of instructions executed.

3.) We redesign processor P1 to run the same program in only 7 seconds. However, the CPI increases by 1.2x over the current version. What clock rate does this new processor have to run at to achieve the 7 second goal?

4.) Find a new clock rate for P5 that reduces its execution time to that of P4.

5.) Instead of the change from the previous problem, find a new number of instructions for P5 that reduces its execution time to that of P6.

6.) Given a program with $10^6$ instructions divided into classes as follows: 10% class A, 20% class B, 50% class C, 20% class D. Which processor is faster for this program?
7.) What is the CPI for each processor for the program in the previous problem?

8.) Compute the number of clock cycles required to execute the program on each processor.

9.) Take the single cycle processor developed in class, and add the following instructions. Note that each instruction will require a separate datapath and control logic that extends the one in class – do not do one processor that does all of the following in one machine. Instead, you will have 5 separate processors, which can do ALL the instructions from class plus one of the instructions below. Your machines should be as simple as possible. If you need new hardware units and/or wires you may add them, or change the existing units, but only as needed. Your machine must complete each of these instructions in a single clock cycle.

**Li rt, imm16**: Put the value imm16 into register rt. imm16 is considered an unsigned number (not 2’s complement), and so the top 16 bits of the registers should be set to 0.

**LW_2reg rd, rs, rt**: a load in r-type format, where Reg[$rd] = Mem[Reg[$rs]+Reg[$rt]]

**Swap rs, rt**: takes the values from $rs and $rt and puts them into registers $rt and $rs respectively, thus swapping their contents.

**Jrmi imm16(rs)**: Jump to the instruction at the address specified in the data memory. The immediate16 (from an I-type format) and the register rs specify an address in the same format as a standard lw/sw (base register and offset). The value in data memory at that address is the address of the next instruction to execute (i.e. the data memory holds the new PC value).

**Wai rd**: where am I. Puts the address of the wai instruction into register rd.

  a.) Show the changes to the single-cycle datapath (including instruction fetch) needed to support this instruction, as well as all the ones in class.

  b.) Show the control signal settings needed for this new instruction.

10.) Create a single-cycle CPU that can perform the following two instructions ONLY. Your CPU should be as simple, and as fast, as possible. If anything is not required for this CPU, do not include it. Show both the datapath and the control.

  beq rs, rt, imm16 if (Reg[rs] == Reg[rt]) PC = PC + 4 + SE(Imm16)<<2;

  jr rs PC = Reg[rs];