UNIVERSITY OF CALIFORNIA, DAVIS
Department of Electrical and Computer Engineering

EEC 170 Introduction to Computer Architecture Winter 2021

Quiz 1 Solutions

1. You are deciding which cell phone to buy. Since the only app you ever run is TikTok, and every phone you're considering runs on an ARM processor and downloads the identical TikTok app for ARM, you are focusing only on maximizing performance on the core TikTok benchmark. Most phones have published useful runtime statistics on their TikTok benchmark performance except one.

   (a) (2 points) In class we discussed that MIPS (millions of instructions per second) is not an ideal figure of merit for performance (instead runtime is the most valid measure of performance). Given the constraints above, why would also it be acceptable to use MIPS to compare different phones in this scenario?

   Solution: Each phone in this problem has the same instruction set and runs the same program, so runs the exact same instructions. Thus any comparison of MIPS yields the same result as any comparison of runtime.

   (b) (10 points) This phone provides the following statistics for the TikTok benchmark:

Instruction class	Fraction of all instructions	MIPS
ALU	0.5	100
Memory	0.3	25
Branch	0.2	50

   Here, the first line means that ALU instructions are 0.5 (50%) of all instructions in the benchmark and, when running *only* ALU instructions, this phone achieves 100 MIPS. Be careful: the second column (fraction of all instructions) is *not* the fraction of time achieving the MIPS metric in the third column.

   You also know that the TikTok benchmark has exactly 100M instructions.

   What is the runtime for the TikTok benchmark on this phone?

   Solution: First, because the TikTok benchmark has 100M instructions and we know the fraction of each class of instructions, we can determine how many instructions we run of each class (instructions per class = total instructions · fraction of total instructions).

   - ALU instructions are 0.5 of 100M instructions = 50M instructions.
   - Memory instructions are 0.3 of 100M instructions = 30M instructions.

- Branch instructions are 0.2 of 100M instructions = 20M instructions.

Now, how long does each class take? Here,

$$\text{runtime in seconds} = \text{instructions}/(\text{instructions}/\text{second})$$

.

- ALU instructions take 50M instructions / 100M IPS = 0.5 s.
- Memory instructions take 30M instructions / 25M IPS = 1.2 s.
- Branch instructions take 20M instructions / 50M IPS = 0.4 s. instructions.

Together this sums to 2.1 seconds.

2. You lead a design team of three junior engineers working on the next generation of your company's "Ash" processor. Each engineer proposes a potential change for the next processor. You bring some statistics from Ash to a design meeting:

Instruction class	Fraction of all instructions	CPI
ALU	0.5	2
Memory	0.3	5
Branch	0.2	3

(a) (4 points) Your first engineer, Bulbasaur, proposes to *reduce the CPI of all instructions by 1.* If you implement Bulbasaur's proposal, **what is the speedup of the new processor over the old one?**

Solution: First, compute Ash's overall CPI.

$$\text{CPI} = \sum \text{CPI}_{\text{class}} \cdot f_{\text{class}} = 0.5 \cdot 2 + 0.3 \cdot 5 + 0.2 \cdot 3 = 3.1$$

If every instruction decreases CPI by one, then we decrease the overall CPI by one also. Bulbasaur's proposal makes the CPI 2.1.

The clock speed and number of instructions don't change, so the speedup is the ratio between the two CPIs, 3.1/2.1 = 1.48 (48% faster).

Post-exam, students indicated that they interpreted "reduce the CPI of all instructions by 1" as "reducing the CPI for every instruction (CPI for ALU goes from 2 to 1, CPI for Memory from 5 to 4, and CPI for Branch from 3 to 2)". This was my intent in writing the question; above I just solved it the fast way (knowing that if we reduce the CPI by one for each instruction type, we just decrease the overall CPI by one). If we do it the long way, we get the same answer:

$$\mathrm{CPI}_{\mathrm{Bulbasaur}} = \sum \mathrm{CPI}_{\mathrm{class}} \cdot f_{\mathrm{class}} = 0.5 \cdot 1 + 0.3 \cdot 4 + 0.2 \cdot 2 = 2.1$$

(b) (8 points) Your second engineer, Charmander, proposes to *improve the compiler so that half of all ALU instructions are eliminated.* If you implement Charmander's proposal, **what is the speedup of the new processor over the old one?**

Solution: Here, two components of the performance equation change, number of instructions and the CPI of the instructions. We consider them each separately.

The number of instructions will decrease. Half of all instructions are ALU instructions and half of those are eliminated, so 25% of all instructions are eliminated. The ratio between old and new instructions is 1.33.

The CPI will also change, since it has fewer ALU instructions. We have to renormalize the fraction-of-all-instruction count since we eliminated half the ALU instructions. The new mix, after eliminating half the ALU instructions, is:

Instr. class	Fraction of all instructions (does not sum to one)	Fraction of all instructions (renormalized, sums to one)	CPI
ALU	0.25	$0.25 \cdot (1/0.75) = 0.33$	2
Memory	0.3	$0.3 \cdot (1/0.75) = 0.4$	5
Branch	0.2	$0.2 \cdot (1/0.75) = 0.267$	3

So the new CPI is $0.33 \cdot 2 + 0.4 \cdot 5 + 0.267 \cdot 3 = 3.461$.

The speedup, then, is old runtime divided by new runtime. The clock speed doesn't change, so this is $(\mathrm{CPI}_{\mathrm{old}}/\mathrm{CPI}_{\mathrm{new}}) \cdot (\mathrm{instructions}_{\mathrm{old}})/\mathrm{instructions}_{\mathrm{new}}) = (3.1/3.461) \cdot 1.33 = 1.19$. The new processor is 19% faster.

Let's look at this a different way. Consider that our program has 20 instructions (because it makes the math work easily). Of those instructions, 10 will be ALU instructions, 6 memory, 4 branch. These instructions will take $10 \cdot 2 + 6 \cdot 5 + 4 \cdot 3 = 62$ cycles to complete. We then eliminate 5 of the ALU instructions. Now our program runs in $5 \cdot 2 + 6 \cdot 5 + 4 \cdot 3 = 52$ cycles. The speedup is $62/52 = 1.19$.

(c) (4 points) Your third engineer, Squirtle, proposes to *increase the clock speed by 50% (1.5× faster), but the overall CPI will go up by a factor of 1.2 (1.2× larger).* If you implement Squirtle's proposal, **what is the speedup of the new processor over the old one?**

Solution: Here, two components of the performance equation change, clock speed and CPI. The clock speed increase yields a speedup overall; the CPI increase yields a slowdown. We know the speedups for each, so just multiply them together: $1.5 \cdot (1/1.2) = 1.25$. So, a $1.25\times$ speedup (25% faster) overall.

(d) (2 points) **If you can only choose one design to implement, whose will it be?** You get 2 points if you get this question correct, 0 if you leave it blank, and -2 is you get it wrong.

Solution: Bulbasaur's proposal yields the largest speedup.