Homework 1 Solutions

1) Execution Time:

Time O0: ~8s

Time O3: ~5s

Speedup: ~50% or ~1.5x
(2.5 pts)

Arithmetic mean. The question could be interpreted to be average of a few trials for each binary so the means would already be listed above. Others may interpret it as finding the average between O0 and O3.
(0.5 point)

(4 Points)

2) Parallel Speedup:

Speedup(1->2) O0: (runtime ~4.3s), speedup ~1.9x
Speedup(1->4) O0: (runtime ~3.0s), speedup ~2.7x
Speedup(1->8) O0: (runtime ~2.0s), speedup ~4.0x

Speedup(1->2) O3: (runtime ~2.8s), speedup ~1.8x
Speedup(1->4) O3: (runtime ~2.1s), speedup ~2.4x
Speedup(1->8) O3: (runtime ~1.4s), speedup ~3.6x

Graph of speedup as a function of thread count.

(3 points for comparable speedup values. 1 point for the graph)

(4 Points)

3. Amdahl's Law

$$\text{Overall Speedup} = \frac{\text{Old execution time}}{\text{New execution time}}$$

$$= \frac{1}{\left(1 - \text{Fraction}_{\text{enhanced}}\right) + \left(\frac{\text{Fraction}_{\text{enhanced}}}{\text{Speedup}_{\text{enhanced}}}\right)}$$

Amdahl's Law Formula:
(ok to use different variable names)
Exec Time 1 Thread O3: ~5s
Exec Time 8 Thread O3: ~1.4s
Comparison: A sentence, or perhaps mentioning speedup (~3.6x).
Parallelizable fraction: ~82% (s_x or $\text{speedup}_{\text{enhanced}}$ is 8)

(0.5 pts for correct formula,
1 pt for correct inputs to formula,
0.5 pts for correct answer roughly)

4. Perf Record

Function	% O0	% O3	Exec time O0	Exec time O3	Speedup
x264_8_me_search_ref	8.49%	6.94%	~0.85s	~0.465s	~1.83x
get_ref_avx2	7.74%	5.81%	~0.77s	~0.389s	~1.98x
refine_subpel	5.55%	6.69%	~0.56s	~0.448s	~1.25x
macroblock_size_cabac	4.12%	NA	~0.41s	NA	NA

(Grading can be flexible here since exact percentages/exec time/functions can vary. The % O3 and O0 are from the output of the command. The exec times can be calculated based on total exec time from question 1. Speedup is calculated from the exec times.)
2 points for output (%O3, %O0)
1 point for reasonable exec time based on that output
1 point for correct speedup based on exec times.

(4 Points)

5. Perf Stat

Use perf stat to determine the instruction count and average CPI for the two compiled binaries. Compute the clock cycle time or your system.

Optimization	Exec time	IC	CPI	Cycle Time
O0	~10s	42,324,707,394	~0.49	0.48ns or 4.8e-10s
O3	~7s	26,833,053,947	~0.47	0.56ns or 5.6e-10s

(2 Points)

Note: Some students used either 4 threads or no thread specification (defaults to 8 threads). While this is fine, you would need to use “user time” to get accurate cycle time.

6. Perf Top-down

Use perf stat's top-down methodology to determine the fraction of cycles spent on the frontend, backend, retired and bad speculation [1]. Compute the theoretical optimal average CPI that code can run on the system. Interpret your result (what does the processor need to be able to do to achieve this CPI?)

(4 Points)

Performance counter stats for 'system wide':

		retiring	bad speculation	frontend bound	backend bound
S0-C0	2	60.5%	9.6%	22.2%	7.8%
S0-C1	2	61.1%	9.4%	22.3%	7.2%
S0-C2	2	61.8%	9.3%	22.2%	6.7%
S0-C3	2	61.8%	8.7%	22.2%	7.3%

(1 pt for percentages)

From the definition of each section (frontend, backend, retired and bad speculation), only retired fraction is useful.
Thus, in an optimal case, the percentage would be 100% for retired and 0% for all the remainder.

(2 pt for correct interpretation)

Given the CPI of ~0.5 from the previous question, and the percentage for retired instructions of 60%, measured with perf topdown, in the ideal case, the expected CPI would be of ~0.6x0.5 = 0.3
(1 pt for correct computation of optimal CPI)