CSE Qualifying Exam, Spring 2025

CSCE 513-Computer Architecture

1. Given the below latencies for the dependant Floating-Point (FP) and Integer operations in a processor:

Instruction Producing Results	**Instruction Using Results**	**Latency in clock cycles**
FP ALU operation	FP ALU operation	4
FP ALU operation	Store double	3
Load double	FP ALU operation	2
Load double	Store/Load double	0
Load integer	Integer ALU operation	1
Integer ALU operation	Integer ALU operation	0

(a) What is the CPI (Clock per Instruction) of a scalar processor when executing the below loop? Find the CPI without any optimization such as scheduling and loop-unrolling.

```
Loop: l.d    $f0,0($s1)      #FP Load
      add.d  $f4,$f0,$f2     #FP Add
      sub.d  $f6,$f4,$f0     #FP sub
      s.d    $f6,0($s1)      #FP Store
      addi   $s1,s1,-8       #integer add
      bne    $s1,$zero,Loop #Branch
```

(b) What is the minimum unrolling factor required to achieve at least a 1.5× speedup when running the loop on a **2-way superscalar processor**? You can use register-renaming, and scheduling to speed up the execution.

2. A processor has a clock rate of **2 GHz** and a **base CPI of 1**. The instruction mix of a benchmark program consists of **20% load and store instructions**. The memory hierarchy of the processor is as follows:

- **L1 cache:** Hit time = **1 cycle**, miss rate = **5%**
- **L2 cache:** Hit time = **5 cycles**, miss rate = **10%**
- **Main memory:** Access time = **100 cycles**

To improve performance, an architect proposes adding an **L3 cache** with a **hit time of 4 cycles** and a **miss rate of 10%**.

(a) **Compute the effective CPI** for the original system (without L3 cache).

(b) **Analyze the impact of adding the L3 cache.** Compute the new CPI and determine the speedup achieved by this modification.

(c) Suppose that **instead of adding the L3 cache**, the base processor frequency is **increased from 2 GHz to 2.5 GHz**, but the base CPI increases to **1.2** due to additional pipeline stages. Would this alternative approach yield better performance than adding the L3 cache? Justify your answer with calculations.

3. You are a research scientist working on optimizing a high-performance computing system designed for real-time space simulations. The system has the following characteristics:

- **Main memory bandwidth:** 8 GB/s
- **Peak computational throughput:** 5 GFlops/s
- **No on-chip cache**

To test its efficiency, you run an adaptive force computation kernel used in a space physics simulation:

```
for (int i = 0; i < n; i++) {
    force[i] = mass[i] * acc[i];
    for (int j = 0; j < n; j++)
        force[i] += G * (mass[j] / (dist[i][j] * dist[i][j]));
}
```

Here, `force`, `mass`, `dist`, and `acc` are **single-precision floating-point (float) arrays**, and G is a gravitational constant.

(a) What is the expected performance of this system when executing the above kernel for **`n=5`**? Determine if the system is **compute-bound** or **memory-bound**, and justify your answer with calculations.

(b) To boost performance, engineers propose the following optimizations:

1. Increase the main memory bandwidth to 32 GB/s
2. Introduce an on-chip cache large enough to store the entire `mass` and `acc` arrays

What is the overall speedup achieved with these enhancements? Discuss whether the optimizations shift the system's bottleneck (compute-bound or memory-bound).

Spring 2025 Qualifying Exam—Algorithms (750)

Question 1 (A Recurrence). Find tight asymptotic bounds on any positive function $T(n)$ satisfying the following recurrence for all sufficiently large n:

$$T(n) = 2T(n/2) + T(\sqrt{n}) + \sqrt{n}\ .$$

You may assume that any implicit floors and ceilings are of no consequence.

Prove your upper bound by the substitution method. (You do not need to prove the matching lower bound, but if your upper bound is not tight, you will not receive credit even for a correct substitution proof.)

Question 2 (Optimal Change-Making). You are the cashier at a bodega, and your cash register has an unlimited supply of coins of various denominations $d_1, \dots, d_k$ (in cents, say, but the currency unit is not important). A customer comes in and pays cash for a purchase, handing you a large bill. After subtracting the price of the purchased item, you need to give exact change back to the customer, say N cents, and you want to do this using as few coins as possible.

For example, let $k = 3$ and $\langle d_1, d_2, d_3 \rangle = \langle 5, 10, 25 \rangle$. If $N = 45$, then three coins suffice ($45 = 25 + 10 + 10$); if $N = 17$, then you cannot make exact change.

Describe an algorithm that takes as input an integer $N \geq 0$ (the amount of change required) and an array $d[1 \dots k]$ of positive integers (the coin denominations; assumed pairwise distinct), and returns the least possible number of coins whose denominations add up to N. If making exact change for N is not possible, your algorithm should return some (any) number greater than N (including ∞).

Describe your algorithm in enough detail so that someone who did well in CSCE 750 can implement it without specific knowledge of the problem.

(Formally, your algorithm returns the least possible value of $\sum_{j=1}^{k} c_j$ subject to the constraints that $c_1, \dots, c_k \geq 0$ are integers and $\sum_{j=1}^{k} c_j d[j] = N$. Here, each c_j is the number of coins of denomination $d[j]$ used for the change. If no such $c_1, \dots, c_k$ exist, then some number greater than N is returned.)

Your algorithm only needs to return the total number of coins used, not the number of each denomination used.

For full credit, your algorithm must run in time $O(kN)$.

Question 3 (Single-Source Shortest Paths). You are given a directed graph G with weight function $w : G.E \to \mathbb{R}$ such that $w(e)$ is an *integer* between 1 and 10 for all $e \in G.E$. You are also given a source vertex $s \in G.V$.

Show how to compute, in *linear time*, shortest distances (d-attributes) and shortest path information (π-attributes) to every vertex from s. Linear time means time $O(|G.V| + |G.E|)$ with the usual adjacency-list representation of G. Your explanation should be clear enough so that someone who did well in CSCE 750 can implement it.