MURAGURI JOEL KARIUKI
SCT212-0042/2020

COMPUTER TECHNOLOGY
Lab 1
Computer Architecture

Exercise 1: Evaluating Performance Differences

Given Information:

- The unoptimized system has a clock rate that is 5% faster (meaning its cycle time is shorter).
- 30% of the unoptimized code consists of load/store instructions.
- The optimized system reduces load/store operations to two-thirds of the original amount.
- Every instruction takes exactly one clock cycle.

Step 1: Define the Variables

Let:

- `Iu` be the instruction count in the unoptimized version.
- `Io` be the instruction count in the optimized version.
- `Clku` be the clock cycle duration for the unoptimized system.
- `Clko` be the clock cycle duration for the optimized system.

Since the unoptimized processor runs 5% faster, its cycle time is 95% of the optimized version's:

Clku = 0.95 × Clko

Step 2: Calculating Instruction Count

In the **unoptimized version**:

- 30% of the instructions are load/store operations.
- 70% are other types.

In the **optimized version**:

- Load/store operations are reduced by one-third (to two-thirds of the original amount).
- All other instructions remain unchanged.

So:

$I_o = 0.7 \times I_u + (2/3 \times 0.3 \times I_u)$
$I_o = 0.7I_u + 0.2I_u$
$I_o = 0.9I_u$

The optimized version runs 90% of the total instructions compared to the unoptimized one.

Step 3: Comparing CPU Time

The general CPU time formula:

CPU Time = Instruction Count × CPI × Clock Cycle Time

Given that CPI = 1 (each instruction takes one cycle), this becomes:

CPU Time = Instruction Count × Clock Cycle Time

- **Unoptimized:**

$T_u = I_u \times Clk_u$

- **Optimized:**

$T_o = I_o \times Clk_o = 0.9I_u \times Clk_o$

Substitute $Clk_u = 0.95 \times Clk_o$:

$T_u = I_u \times 0.95 \times Clk_o$

Now comparing both:

$T_o / T_u = (0.9I_u \times Clk_o) / (0.95I_u \times Clk_o) = 0.9 / 0.95 = 0.947$

So:

$T_o = 0.947 \times T_u$

Conclusion:
The optimized version executes faster — approximately **5.3% improvement** in performance.

Exercise 2: Register-Memory Addressing

Part 1: What Proportion of Load Instructions Should Be Removed?

Given:

- The clock cycle time increases by 5% due to the new addressing method.
- Instruction frequency from the reference table:
  - Load: 22.8%
  - Store: 14.3%
  - Add: 14.6%
  - (Others not shown here)

Assumptions:

- Execution time is affected by:

Execution Time $\propto$ Instruction Count × CPI × Clock Cycle Time

- CPI is unchanged.
- Only instruction count and clock time are influenced by the addressing change.

Let x be the portion of load instructions that get removed.

Then:

New Instruction Count = (1 - x) × Load Instructions + Other Instructions
= (1 - x) × 0.228 + (1 - 0.228)

The updated execution time becomes:

Execution Time Ratio = 1.05 × [(1 - x) × 0.228 + 0.772]

We want performance to stay the same, so set this ratio to 1:

1.05 × (1 - 0.228x) = 1
1.05 - 0.2394x = 1

0.2394x = 0.05
x = 0.05 / 0.2394 ≈ 0.2088

Result:
Roughly **20.9% of the load instructions** must be eliminated to maintain performance after adopting the new addressing format.

Part 2: When Elimination Cannot Be Applied

Example Code:

```
LOAD R1, 0(R2)
LOAD R3, 0(R4)
ADD R5, R1, R3
```

Explanation:

- Here, two separate load instructions are used to fetch values into `R1` and `R3`.
- These values are later added and stored in `R5`.

Problem:

- The optimization strategy assumes we can merge a LOAD and an ADD into one register-memory ADD.
- This is not feasible when different values need to be loaded from memory before the ADD can take place.