BCT 2408 – COMPUTER ARCHITECTURE
LAB 2
MURAGURI JOEL KARIUKI **– SCT 212-0042/2020**

Loop Analysis:

```
loop:   LD R1, 0(R2)
        DADDI R1, R1, 1
        SD 0(R2), R1
        DADDI R2, R2, 4
        DSUB R4, R3, R2
        BNEZ R4, loop
```

Initial Setup:
R3 is initialized as R2 + 396.

Number of Loop Iterations:
Since R2 is incremented by 4 each cycle and the difference between R3 and R2 is 396, the loop will execute `396 / 4 = 99` times.

A: Without Forwarding, Branch Decision in ID, Pipeline Flush Applied

Assumptions:

- Pipeline consists of 5 stages: IF, ID, EX, MEM, WB
- No forwarding is available; hence, data hazards introduce delays.
- Branch instructions are evaluated in the ID stage, and any mispredictions flush the pipeline.
- Memory access latency is one cycle.

Data Dependencies and Stalls:

- `LD R1, 0(R2)` → `DADDI R1, R1, 1`: Data dependency on R1; need to wait for WB → **2 stalls**
- `DADDI R1, R1, 1` → `SD 0(R2), R1`: SD also depends on updated R1 → **2 stalls**
- `DADDI R2, R2, 4` → `DSUB R4, R3, R2`: DSUB requires new R2 → **2 stalls**
- `DSUB R4, R3, R2` → `BNEZ R4, loop`: Branch depends on R4 → **2 stalls**

Per Iteration Cycle Breakdown:

- 6 instructions × 5 pipeline stages = **30 cycles**
- 4 data hazards causing 2 cycles of delay each = **8 stalls**
- **2-cycle branch penalty** for misprediction

Total per Iteration: 30 + 8 + 2 = **40 cycles**
Total for 99 Iterations: 99 × 40 = **3960 cycles**

B: With Forwarding, Predict Branch as Not Taken

Assumptions:

- Forwarding is enabled, resolving most data hazards
- Branches are speculated as not taken
- Misjudged branches lead to pipeline flushes
- Branches resolved at the ID stage

Data Dependencies and Stalls:

- `LD` followed by immediate use in `DADDI`: Load-use hazard remains → **1 stall**
- Other instructions are handled through forwarding without delays

Per Iteration Cycle Breakdown:

- Each instruction takes 1 cycle: 6 × 1 = **6 cycles**
- 1 cycle for load-use stall
- 2 cycles for potential branch misprediction

Total per Iteration: 6 + 1 + 2 = **9 cycles**
Total for 99 Iterations: 99 × 9 = **891 cycles**

C: Delayed Branch Implementation with Forwarding

Assumptions:

- Uses delayed branching: the instruction following a branch always executes
- Forwarding resolves data dependencies
- Branch decisions occur during ID

Instruction Scheduling (With Delay Slot Optimization):

```
loop:   LD R1, 0(R2)
        DADDI R1, R1, 1
        SD 0(R2), R1
        DADDI R2, R2, 4
        DSUB R4, R3, R2
        DADDI R5, R5, 0  ; Delay slot filler (NOP or meaningful instruction)
        BNEZ R4, loop
```

Note: The `DADDI R5, R5, 0` serves as a placeholder for the branch delay slot. If a productive instruction can replace it, overall efficiency improves.

Per Iteration Timing:

- No branch penalty, as delay slot is effectively used
- All instructions execute in 1 cycle = **6 cycles**
- 1 stall due to load-use hazard

Total per Iteration: 6 + 1 = **7 cycles**
Total for 99 Iterations: 99 × 7 = **693 cycles**