JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY
Setting Trends in Higher Education, Research and Innovation

JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTING

BCT 2408 COMPUTER ARCHITECTURE LAB 1

NAME: KELVIN SIKAMOI

REG no: SCT212-0677/2021

DATE: 13/3/2025

Computer Architecture Worksheet

E1: Performance Comparison.

Problem Statement:
Procedure calls introduce performance overhead due to excessive load and store operations. A new optimization reduces the number of loads and stores. Given:
- The unoptimized version has a 5% higher clock rate.
- 30% of its instructions are loads and stores.
- The optimized version reduces loads and stores to 2/3 of their original count.
- Each instruction executes in a single clock cycle.

Solution:

1. Establishing the Performance Metrics:
   Execution Time = (Instruction Count × CPI) / Clock Rate
   Since CPI remains 1 for all instructions, execution time is inversely proportional to clock rate and instruction count.

2. Instruction Count Reduction:
   - Unoptimized: 100 instructions
   - Loads/stores: 30% = 30 instructions
   - Optimized version executes 2/3 of these: 20 loads/stores.
   - Total optimized instruction count = 90.

3. Execution Time Calculation:
   - Unoptimized Execution Time: 100 / CR
   - Optimized Execution Time: 90 / (0.95CR)
   - Speedup: 1.17

Result:

The optimized version is 17% faster due to a lower instruction count, even with a slightly reduced clock rate.

E2: Effect of Register-Memory Addressing Mode on Performance

Problem Statement:
A new register-memory addressing mode eliminates separate load instructions:
LOAD Rx, 0(Rb)
ADD Ry, Ry, Rx

becomes:
ADD Ry, 0(Rb)
This modification increases the CPU clock period by 5%.

Solution:

1. Performance Consideration:
   Execution Time = (Instruction Count × CPI) / Clock Rate
   The CPI remains the same, but the clock rate decreases by 5%, making it 0.95CR.

2. Required Load Reduction Calculation:
   - To maintain performance: (1 - L) × 0.95 = 1
   - L = 5.26%

3. When Replacement Fails:
   - If a loaded register is modified before being used elsewhere, replacement is not possible.

Results:

To maintain performance, at least 5.26% of loads must be eliminated. However, dependencies may prevent full replacement in some cases.

D1: Are Modern RISC Processors Still RISC?

Modern RISC processors retain key RISC principles but have evolved:
- Load/Store Architecture: Most instructions still operate on registers rather than memory.
- Fixed-Length Encoding: Common in RISC-V and ARM architectures for efficient decoding.
- Pipelining Optimization: Instructions execute in minimal cycles, optimizing speed.

However, modern RISC incorporates:
- Additional Instruction Sets: Vector processing, floating-point operations, and multimedia extensions.
- Microcoding: Some RISC processors use microcode for complex tasks, blurring distinctions with CISC.
- Compressed Instructions: Some architectures adopt variable-length instruction sets (e.g., ARM Thumb).

Final Conclusion:

RISC processors have evolved but maintain core principles. They are RISC in essence but adapted for modern performance demands.

D2: Is Intel's x86 Now a RISC or Still a CISC?

Modern x86 architectures integrate RISC principles internally but remain CISC externally:

1. RISC-like Microarchitecture:
- Internally, x86 processors use micro-operations that resemble RISC instructions.
- They employ deep pipelines and out-of-order execution for speed.
- Register-based execution is prioritized over direct memory operations.

2. CISC Legacy:
- Externally, x86 still uses variable-length instructions with complex decoding.
- Legacy compatibility forces it to retain older complex instructions.
- Instruction decoders translate CISC instructions into RISC-like micro-operations.

Final Conclusion:

At the hardware level, x86 behaves like a RISC processor. However, at the software interface, it remains CISC, ensuring compatibility with decades of existing software.