USE YOUR IMAGINATION™

Blue Book

EXAMINATION BOOK

BOX NO. _____

NAME RAMAN KUMAR JHA

SUBJECT CSA

CLASS _____

SECTION A

INSTRUCTOR AZEEZ

DATE 12/18/2025

11" x 8.5" 8 LEAVES 16 PAGES

ROARING SPRING® PAPER PRODUCTS
ROARING SPRING, PA 16673

① This problem uses Amdahl's law, which calculates the overall system speedup based on the improvement of a specific component.

The formula –

$$\text{Speedup} = \frac{1}{(1-f) + \frac{f}{S}}$$

where,

f = The fraction of execution time affected by the improvement.

S = The speedup factor of the specific component.

Given data:

⇒ Scenario 1 (FPSR Hardware):

→ Fraction affected (f_{FPSR}) = 20% or 0.20

→ Speedup (S_{FPSR}) = 10

⇒ Scenario 2 (All FP instructions):

→ Fraction affected (f_{FP}) = 50% or 0.5 (since fp is half the time)

→ Speedup (S_{FP}) = X

Goal: The performance of Scenario 2 must match the scenario 1.

Step 1:

The Speedup achieved by Scenario 1,

$$\text{Speedup}_1 = \frac{1}{(1-0.2) + \frac{0.2}{10}}$$

$$= \frac{1}{0.8 + 0.02}$$

$$= \frac{1}{0.82} \approx 1.2195$$

Step 2:

The Speedup of scenario 2 must match the speedup of Scenario 1.

$$1.2195 = \frac{1}{(1-0.5) + \left(\frac{0.5}{X}\right)}$$

$$0.82 = (1-0.5) + \frac{0.5}{X}$$

$$0.82 = 0.5 + \frac{0.5}{X}$$

$$0.32 = \frac{0.5}{X}$$

$$X = \frac{0.5}{0.32}$$

$$X = 1.5625$$

Answer —

The floating-point (FP) instructions must be accelerated by a factor of 1.5625 to achieve the same performance as the specialized hardware.

(2) Given data —

→ Memory computation weight : 40% (0.4)

→ Processor computation weight : 60% (0.6)

→ The table provides performance metrics (higher is better). To get execution time (lower is better), we use the inverse of performance.

$$\text{Time} \propto \frac{1}{\text{Performance}}$$

(1)

$$\text{Time}_{\text{weighted}} = 0.4 \times \frac{1}{\text{Memory Performance}} + 0.6 \times \frac{1}{\text{Dhrystone Performance}}$$

chip	weighted execution time.
Athlon 64 X2 4800+	0.000146
Pentium EE 840	0.000156
Pentium D 820	0.000173
Athlon 64 X2 3800+	0.000171
Pentium 4	0.000225
Athlon 64 3000+	0.000214
Pentium 4 570	0.000168
Processor X	0.000177

Analysis:
The Athlon 64 X2 4800+ has the lowest weighted execution time; indicating it's the fastest overall for the specific workload mix. The Pentium 4 has the highest time, making it the slowest.

(2) (2) Speedup from Pentium 4570 to Athlon 64 X2 4800+ on a CPU intensive hardware-

Since the application is CPU- intensive, we look specifically for Dhrystone performance column which measures processor integer performance.
old performce (Pentium) : 11210
New performance (Athlon) : 20718

$$\text{Speedup} = \frac{\text{Performance}_{new}}{\text{Performance}_{old}}$$

$$= \frac{20718}{11210}$$

$$\approx 1.848$$

So the anticipated speedup is 1.848

(2) (3) We compare the pentium 4 570 vs the Pentium D 820, We need to find the workload mix where their execution time are equal.

Let x be the fraction of memory computation, the $(1-x)$ is the fraction of processor computation.

Equation —

$$x\left(\frac{1}{Mem_{570}}\right) + (1-x)\left(\frac{1}{Proc_{570}}\right) = x\left(\frac{1}{Mem_{820}}\right) + (1-x)\left(\frac{1}{Proc_{820}}\right)$$

$$\frac{x}{3501} + \frac{1-x}{11210} = \frac{x}{3000} + \frac{1-x}{15220}$$

$$\frac{1-x}{11210} - \frac{1-x}{15220} = \frac{x}{3000} - \frac{x}{3501}$$

$$(1-x)\left(\frac{1}{11210} - \frac{1}{15220}\right) = x\left(\frac{1}{3000} - \frac{1}{3501}\right)$$

$$(1-x)\ (0.0000235) = x\,(0.0000477)$$

$$235(1-x) = 477x$$

$$235 - 235x = 477x$$

$$235 = 712x$$

$$x = \frac{235}{712}$$

$$x \approx 33\%.$$

Answer —

The ratio is approximately 33% memory computation and 67% processor computation.

(3) System constraints —

→ Registers: 32, requires $\log_2(32) = 5$ bits
→ Immediates: 16 bits
→ Instructions: 142, requires $\log_2(142) = 8$ bits
→ All instruction must be a multiple of 8 bits

(3) (V) Bits required for each instruction types.

(a) Type 1 (20%): 1 input Reg, 1 output reg.
Raw: opcode (8) + In(5) + out(5) = 18 bits
aligned: Round 18 upto nearest multiple of 8 = 24

(b) Type 2: (30%): 2 input Reg, 1 output reg.
Raw: opcode(8) + In(5) + In(5) + out(5) = 23 bits
aligned: Round 23 upto nearest multiple of 8 = 24

(c) Type 3: (25%): 1 input reg, 1 output reg, 1 immediate.
Raw: opcode (8) + In(5) + out(5) + Imm(16)
= 34 bits
aligned: Round 34 upto nearest multiple of 8 = 40

(d) Type 4: (25%): 1 Immdiate input, 1 output reg.

Raw: opcode (8) + Im (16) + out (5)
= 29 bits.
aligned: Round 29 upto nearest multiple of 8 = 32

(3)(2) Fixed length encoding :-
To support all instructions, the fixed length must be equal to the largest instruction size found in Part 1.
Fixed size : 40 bits (dictated by type 3)

Variable length encoding -
We calculate the weighted average instructions length based on frequency.

$$\text{Avg} = (0.20 \times 24) + (0.30 \times 24) + (0.25 \times 40) + (0.25 \times 32)$$
$$= 4.8 + 7.2 + 10 + 8$$
$$= 30 \text{ bits}$$

Calculate saving -

$$\text{Savings} = \frac{\text{Fixed} - \text{Variable}}{\text{Fixed}}$$
$$= \frac{40 - 30}{40} = \frac{10}{40}$$
$$= 0.25$$

Variable length encoding takes 25% less memory than fixed length encoding.

④ Given data:

→ Base CPI : 1.5
→ Memory latency : 40 cycles
→ Transfer rate : 4 bytes / cycle
→ Block size : 32 bytes
→ Dirty blocks : 50% (0.5)
→ Instruction mix : 1 (instruction fetch)
+ 0.2 (Data transfer)
= 1.2 memory access per instruction.

Step 1 : Average miss penalty

Since there is no write buffer and the cache uses a "write-back" policy.

⇒ Time to read a block :
latency + (Block size / Transfer rate)
= 40 + (32/4)
= 40 + 8
= 48

Time to write a dirty block : same as read time = 48 cycles.

⇒ Miss penalty calculation —
→ If the replaced block is clean (50% chance) : we only read the new block = 48 cycles.

→ If the replaced block is dirty (50% chance): we write the old block first, then read the new one.
Penalty = 48 + 48 = 96 cycles.

Average miss penalty = (0.5 × 48) + (0.5 × 96)
= 72 cycles.

Step2: Calculate effective CPI for each cycle system.
Effective CPI = Base CPI + (Accesses / Inst × Miss penalty)

⇒ System 1 (16 KB direct Mapped)
→ Cycle factor: 1.0 (No cycle time change)
→ Miss rate: 2.9% (0.029)

⇒ System 2 (16 KB) 2-way associative)
→ Cycle factor: 1.2
→ miss rate: 2.2% (0.022)

⇒ System3 (32 KB Direct mapped)
→ cycle factor: 1.25
→ Miss rate: 2.0% (0.020)

Answers

(a) Effective CPI

→ System 1 : 4.01

→ System 2 : 3.40

→ System 3 : 3.23

(4)(4)(2) We compare the execution time

→ System 1 : 4.006 (Fastest)

→ System 3 : 4.035

→ System 2 : 4.081

System 1 is best as it yields the lowest overall execution time per instruction despite having the highest miss rate, because it does not incur a clock cycle penalty.

(5)(a) Write through Cache —

A write through cache is a policy where information is written to both the cache block and the corresponding block in the lower-level memory (main memory) simultaneously.

Faster / Slower —— for writing

It is generally slower than a write-back cache. Write-back cache only write to main memory when a block is replaced (evicted), allowing multiple writes to the same block to occur quickly within the cache. Write through the incur the main memory latency on every write operation. (unless a write buffer is effectively used to mask this)

(5) (b) Pipeline Speedup —

⇒ Unpipelined processor ——

Average CPI = (%.ALU X4) + (%.Branch X5) + (%.Mem X4)

= (0.5X4) + (0.35X5) + (0.15X4)

= 2.0

Time per instruction = 4.35 cycles X 1 ns/cycles

= 4.35 ns

⇒ Pipelined processor:
ideally the CPI is 1.
→ The new clock time is determined by the stage latency plus overhead.
Assuming that the original stages were roughly balanced to 1ns clock.

New clock cycle = 1 ns + 0.15ns
= 1.15ns

Time per instruction = 1CPI x 1.15ns
= 1.15ns

Speedup calculation:

$$\text{Speedup} = \frac{\text{Unpipelined time}}{\text{Pipelined time}}$$

$$= \frac{4.35\ ns}{1.15\ ns}$$

$$= 3.78$$

Answer —
The pipeline provides a speedup of 3.78 x.

(6) (a) Identify data dependencies and Forwarding code —

1. ADD R2, R5, R4 (Writes R2)
2. ADD R4, R2, R5 (Reads R2, Writes R4)
3. SW R5, 100(R2) (Reads R2)
4. ADD R3, R2, R4 (Reads R2, R4)

Dependencies —

1. R2 between Inst 1 and Inst 2: Read-After-Write (RAW). Resolved via forwarding (EX stage of Inst 1 to EX stage of Inst 2)

2. R2 between Inst 1 and Inst 3: RAW, Resolved via forwarding (MEM stage of Inst 1 to EX stage of Inst 3).

3. R2 between Inst 1 and Inst 4: RAW, Resolved via register files (write in first half of the cycle, Read in second half) or forwarding.

4. R4 between Inst 2 and Inst 4: RAW. Resolved via forwarding (MEM stage of Inst 2 to EX stage of Inst 4)

(6)(b) Pipeline status at Cycle 5:

Code sequence —

1. ADD R1, R2, R3
2. ADD R4, R5, R6
3. ADD R7, R8, R9
4. ADD R10, R11, R12
5. ADD R13, R14, R15

Pipeline stages in Cycle 5:

→ Inst 1 (WB) : Write back stage
→ Inst 2 (MEM) : Memory access stage
→ Inst 3 (EX) : Execute stage
→ Inst 4 (ID) : Instruction Decode/ Register read stage
→ Inst 5 (IF) : Instruction Fetch stage

Registers Read/ written:

→ Read: Occurs in the ID stage. Instruction 4 is in ID. It reads R11 and R12.

→ Written: Occurs in the WB stage. Instruction 1 is in WB. It writes R1.