CS60003: High Performance Computer Architecture

Mid-Semester Examination (Spring 2023-24)

Sample Solution (Selected Questions only)

1)a) 1st Complication: Arbitrating contention between hit & misses (or misses & misses) needs to be done. This happens because in a non-blocking cache, hit can collide with misses returning from the next level of the memory hierarchy, or misses can similarly collide. The situation is usually handled by giving priority to hit over misses, & by ordering colliding misses.

2nd Complication: to track multiple returning misses, so the result can be transferred to the correct cache block, & the correct pending load/store be notified.

b) In a Write Back cache, when a dirty cache block is replaced during a Read Miss, the dirty block does not necessarily immediately move to main memory - they may wait for a long time in the Write Buffer. Hence, later again if this cache block needs to be brought back to the cache, it is not clear whether the last updates have synced to memory or not — possibly the dirty (last updated) cache block is still waiting in the Write Buffer.

c) Amdahl's law assumes that the problem size remains the same, even with the availability of multiple processor cores. Hence, the execution time for the non-parallelizable portion of the code remains unchanged, in spite of the presence of additional hardware resources. This is not the case for Gustafson's Model, which assumes larger amount of HW resources ⇒ larger problem sizes, which makes the contribution of the non-parallelizable portion of the code relatively insignificant, overall leading to more optimistic estimates of speedup.

d) Advantage: if the way predictor is correct, the cache access latency is small since the hit time is small (the MUX select signal is already set to the value predicted by the predictor).

Disadvantage: There is (ideally) an 1 clock cycle penalty if the predictor is incorrect. However, in very fast processors, since the clock cycle time is extremely small, it is challenging to restrict the misprediction penalty to 1 cycle.

e) Advantage: If the branch condition is evaluated in ID (e.g. in MIPS), then an ALU operation to a register followed by a conditional branch based on this register value is a data hazard, e.g. consider the following pseudocode:

```
    sub  R1, R2, R3
    beq  R1, zero, Label
    :
Label:
```

The evaluation of the branch condition in ID is not possible, since in the same cycle, the "sub" instruction is evaluating the R1 register in its EX stage! Hence, there must be an 1 cycle stall! (assuming EX → ID forwarding present)

2) Let P_1 contain I instructions

a) The run-time of P_1 on M_1:

$$T_1 = \frac{(0.4*2 + 0.5*1 + 0.1*3)I}{f_1} s = \frac{(0.8 + 0.5 + 0.3)I}{2*10^9} s = \frac{1.6I}{2*10^9} s, \text{ since } f_1 = 2\,GHz.$$

Let the clock frequency of M_2 be f_2. Then, the time required to execute P_1 on M_2:

$$T_2 = \frac{(0.4*2 + 0.5*2 + 0.1*2)I}{f_2} s = \frac{2.0I}{f_2} s$$

Given: $T_1 = T_2 \Rightarrow \frac{1.6I}{2*10^9} = \frac{2.0I}{f_2} \Rightarrow f_2 = \frac{4.0}{1.6} GHz = 2.5\,GHz.$

b) Suppose P_1, P_2 each have "I" instructions

M_1, M_2 each run at a frequency "f"

For M_1:

Execution time $= [(0.4*2) + (0.5*1) + (0.1*3)] +$

$$T_1 = \frac{[(0.4*2 + 0.5*1 + 0.1*3) + (0.5*2 + 0.2*1 + 0.3*3)]I}{f} = \frac{3.7I}{f}$$

For M_2, similarly,

$$T_2 = \frac{[(0.4*2 + 0.5*2 + 0.1*2) + (0.5*2 + 0.2*2 + 0.3*2)]I}{f} = \frac{4.0I}{f}$$

$\therefore$ M_1 is faster overall for this workload.

c) Suppose, P_1 runs x times & P_2 runs y times

Under same assumptions at part-(b):

For M_1: $T_1 = \frac{(1.6x + 2.1y)I}{f}$ ——— (1)

For M_2: $T_2 = \frac{(2.0x + 2.0y)I}{f}$ ——— (2).

Given: $T_1 = T_2 \Rightarrow 1.6x + 2.1y = 2.0x + 2.0y$

$\Rightarrow 0.1y = 0.4x$

$\Rightarrow x : y = 1 : 4$

$\therefore$ P_1 should run 1 time, P_2 " " 4 times } Example workload.

3) a) No. of cache blocks = $\frac{\text{Cache size}}{\text{Block size}} = \frac{4KB}{16 bytes} = \frac{2^{12} bytes}{2^4 bytes} = 2^8 = 256$

b) Size of array A = Size of array B = (1024*4) bytes = 256 cache blocks
(Since each integer = 4 bytes)

Each cache block can hold: $\frac{16 bytes}{4 bytes}$ = 4 integer (array elements)

Starting address of A[0]: 0x00004000 = 16384, maps to cache block: (16384/16)[mod 256] = 0

" " of A[1020]: 16384 + 1020*4 = 20464, maps to cache block: (20464/16)[mod 256] = 255

Starting address of B[0]: 0x00010800 = 67584, maps to cache block: (67584/16)[mod 256] = 128

" " of B[1020]: 67584 + 1020*4 = 71664, maps to cache block: (71664/16)[mod 256] = 127

Index(i)	A access pattern	A goes to (Block #)	B access pattern	B goes to (Block #)	Comment
1023:1020	A[1023:1020]	255	B[1023:1020]	127	2 write Misses
1019:1016	A[1019:1016]	254	B[1019:1016]	126	do
					do
515:512	A[515:512]	128	B[515:512]	0	2 write misses, B starts overwriting A blocks, A starts overwriting B blocks
511:508	A[511:508]	127	B[511:508]	255	
127:124	A[127:124]	31	B[127:124]	159	do
				128	do
3:0	A[3:0]	0	B[3:0]		

∴ Total: (2 write Misses) for every 4 array elements

⇒ $\left(\frac{1024}{4} * 2\right) = 512$ write misses, zero read misses.

(c) # of bytes written back to memory: (write back happens when blocks start getting replaced) = $\frac{1}{2}$ of A-array + $\frac{1}{2}$ of B-array

= (1024*4) bytes = 4kB

4) 64-bit V.A.

Page size = 2 kB => Page/frame offset = 11 bits
↳ = 2^{11} bytes

# of P.T. levels = 3.

PTE size: 8 bytes (for every level)

Amount of physical memory: 32 GB = 2^{35} bytes.

Task Virtual Memory Requirement: 16 TB

1st level: 4 pages
2nd level: 8 pages
P.T. occupied

(a) # of bits used for page offset: 11 bits.

(b) # virtual pages = $2^{(64-18)} = 2^{53}$

physical frames = $\frac{\text{Physical mem. size}}{\text{Frame size}} = \frac{2^{35}\text{ bytes}}{2^{11}\text{ bytes}} = 2^{24}$

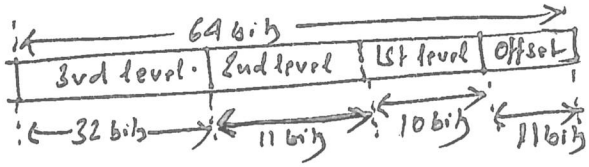
(c) 1st level P.T. size = 4 pages × (2 kB/page) = 8 kB = 2^{13} bytes

∴ # of entries in 1st level P.T. = $\frac{8\text{ kB}}{8\text{ B}} = 2^{10}$

2nd level P.T. size = 8 pages × (2 kB/page) = 16 kB

∴ # of entries in 2nd level P.T. = $\frac{16\text{ kB}}{8\text{ B}} = 2^{11}$

∴ V.A. is organized as:

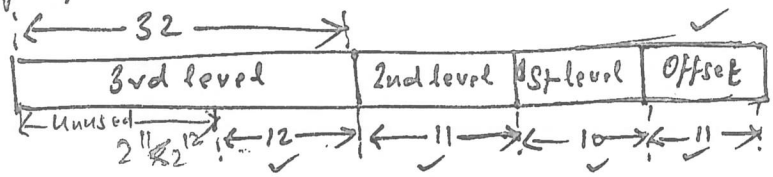


∴ 1st-level # of index bits: 10
2nd- " " " " : 11
3rd- " " " " : 32

(d) Task Virtual Memory requirement: 16 TB = $2^4 \times 2^{40}$ bytes = 2^{44} bytes

∴ Only 44 − (11+10+11) = 12 bits of the 3rd level actually are used.

∴ Usage of virtual address space:



3rd level P.T. requires 2^{12} entries

∴ Size of 3rd level P.T. = $2^{12} \times$ (PTE size) = $2^{12} \times 2^3$ bytes = 2^{15} bytes

∴ # of pages required for 3rd level P.T. = $\frac{2^{15}\text{ bytes}}{\text{Page size}} = \frac{2^{15}\text{ bytes}}{2^{11}\text{ bytes}} = 16$

∴ total # of pages required (for 3-level P.T. scheme) = $(2^4 + 2^{15} + 2^{25})$ pages = 33,587,216 (pages)

= $16 + (2^{12} \times 8) + (2^{12} \times 2^{11} \times 4)$

- 16: 3rd level P.T. Size (in pages)
- 2^{12}: # of 2nd level P.Ts.
- 8 pages: P.T. page
- 4 pages/P.T.: # of 1st level P.T.
- $2^{12} \times 2^{11}$: No. of 2nd level P.T.

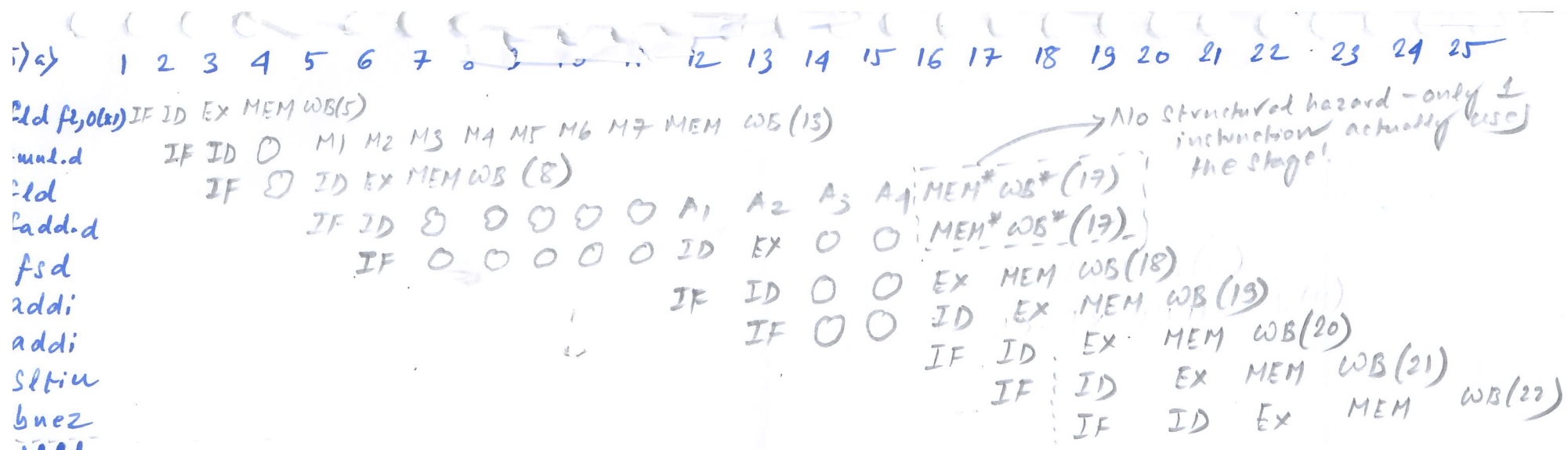
(Actual)/(Flat) ≈ 0.0039 = 0.39%

(e) Flat page table scheme would require: $2^{(44-11)} = 2^{33}$ entries = 2^{33} pages (for task mentioned)

(for task mentioned) [not general case]

which occupies $\frac{2^{33} \times 8\text{ B}}{2^{11}\text{ B}} = 2^{25}$ frames

5) a)



Notes: 1) fmul.d has latency = 6 ∴ cycles ⇒ takes 7 cycles to complete. EX
2) fadd.d has latency = 3 ∴ cycles ⇒ " 4 " " " EX

(b) # of cycles to complete a single iteration: 17 (∵ from clock cycle #18, overlap of next iteration starts)

(c) Reordered Code [with Register Renaming]

```
fld f2, 0(x1)      # no stall
fld f8, 0(x2)      # no stall, f6 renamed to f8
fmul.d f4, f2, f0  # no stall
addi x1, x1, 8     # no stall
addi x2, x2, 8     # no stall
fadd.d f6, f4, f8  # 4 cycle stall, finishes on cycle #17
sltiu x3, x1, done # no stall, 4 cycle stall, finishes on cycle #15
fsd -8(x2), f6,    # offset modified to -8, 1 cycle stall, finishes on cycle #17
bnez x3, foo       #  finishes on cycle #18 after starting from cycle #13
fld f2, 0(x1)      #  finishes on cycle #19 after starting from cycle #14
```

Next iteration

(d.) # of cycles/iteration of re-ordered code: 13 cycles/iteration.

The above situation cannot arise in RISC-V, if EX → EX forwarding is present.
Disadvantage: there is a 2 cycle penalty if there is a branch mispredication, in comparison to an 1-cycle penalty in MIPS.

6. Steps:
1) Issue
2) Read Operands
3) Execute
4) Write Results

} See Appendix-C (C.7) of your textbook for details of these steps. You are expected to describe each of these terms in brief.

RAW Hazards: These are avoided by making the dependent instruction wait in its functional unit, till the operand value is ready to be read from the register file (the Scoreboard controller gives permission to the functional unit as & when it can read its value from the register file.

WAW Hazards: Before issue, the destination register for an instruction is examined. If it is the same as the destination register of a currently active instruction in any of the functional units, instruction issue stalls, & remains stalled until the active unit has completed its WB (Write Results) step.

WAR Hazards: After an instruction has completed it is checked whether any preceding instruction is yet to complete its "Read Operands" phase, with one of its operands same as the destination register of the instruction under question. If this is the case, the "Write Results" step of the instruction that has completed execution is not allowed to perform its "Write Results" — it waits till the other instruction on which it has WAR has Completed its "Read Operands" stage.