# Yb2078 CSAFinals - Final Exam

Computing Systems Architecture (New York University)

Computer Systems Architecture Finals

Student Name: Yudhajit Bhattacharyya

Email: yb2078@nyu.edu

N-number: N13109142

Solution 1

| Instr. | Clk. Cycle | Frequency |
|---|---|---|
| ALU ops. | 1 | 47.5 |
| Jumps | 2.4 | 5 |
| Branches | 4 | 15 |
| Loads | 3.5 | 22 |
| Stores | 2.8 | 10.5 |

$\text{Freq}_{\text{ALU ops}} = \left(\frac{39+56}{2}\right) = 47.5$ ; $\text{Freq}_{\text{Branch}} = \left(30/2\right) = 15$

$\text{Freq}_{\text{jumps}} = \left((7+3)/2\right) = 5$ ; $\text{Freq}_{\text{Loads}} = \left(44/2\right) = 22$

$\text{Freq}_{\text{Stores}} = \left(21/2\right) = 10.5$

∴ Avg. CPI:

~~[illegible]~~

$0.475 + (0.05 \times 2.4) + (0.15 \times 3) + (0.22 \times 3.5) + (0.15 \times \frac{4}{0.6})$

$\Rightarrow 0.475 + 0.12 + [illegible] + 0.77 + 0.294$

Ans = 2.139

Soln 2(i)

the execution time has to be performed in two halves, namely, the fast phase (50%) & the non-fast phase (50%)

∴ ~~Fast~~ As is given in the problem

$\text{Fast}_{\text{non-enhanced}} = (10 \times \text{Non-Fast}_{\text{non-enhanced}})$

i.e.

$= 50\% \times 10 = 500\%$

Relative Exec. Time $_{non\text{-}enhanced}$ = 500% + 50%

= 550%

$$\therefore Speedup_{overall} = \frac{Execution\ Time_{non\text{-}fast}}{Execution\ Time_{fast}}$$

$$\Rightarrow \frac{550\%}{100\%} = 5.5$$

Soln 2.ii.

Using Amdahl's Law:

$$\frac{Exec\ Time_{fast}}{Exec\ Time_{non}} = Speedup_{overall} \times \left(Speedup_{fast} - Speedup_{non\text{-}fast}\right)$$

$$Speedup_{overall} \times Speedup_{fast}$$

Using Amdahl's Law:

% converted to fast mode:—

$$\frac{\left(\left(Speedup_{overall} \times Speedup_{fast}\right) - Speedup_{non\text{-}fast}\right)}{\left(\left(Speedup_{overall} \times Speedup_{fast}\right) - Speedup_{overall}\right)}$$

$$\Rightarrow \frac{(5.5 \times 10) - 10}{(5.5 \times 10) - 5.5} = 90.91\ \%$$

Solution 3.i:

Assume that the number of processors = N

∴ invoking Amdahl's Law:

$$\text{Speedup}_{\text{overall}} = \frac{1}{\left\{\left(1 - \text{Fraction}_{\text{parallelizable}}\right) + \left(\frac{\text{Fraction}_{\text{parallelizable}}}{N}\right)\right.}$$

It's stated that the Intel i5 is a dual core processor.

SO, N = 2 is our case.

x = 40% or 0.4 (parallelized)

$$\therefore \text{Speedup} = \frac{1}{(1 - 0.4) + \left(\frac{0.4}{2}\right)} =$$

$$= \frac{1}{0.6 + 0.2} = 1.25$$

Solu. 3 ii.

Now $x = 0.99$ (factor by which it's parallelized)

$$\therefore \text{Speedup} = \frac{1}{(1-0.99) + \left(\frac{0.99}{2}\right)} = \frac{1}{0.01 + 0.495}$$

$$= 1.98$$

Solution 3. iii.

$$\text{Speedup}_{(1^{st}\ app)} = \frac{1}{(1-0.4) + \left(\frac{0.4}{2}\right)} = 1.25$$

↑ The first app. is running using 80% of the resources.

$$\text{Speedup}_{(2^{nd}\ App)} = \frac{1}{(1-0) + (0/2)} = 1$$

↑ The second app uses only 20% of the resources. Also, since it is running in serial, there is no parallelization.

Again, we shall invoke Amdahl's Law :—

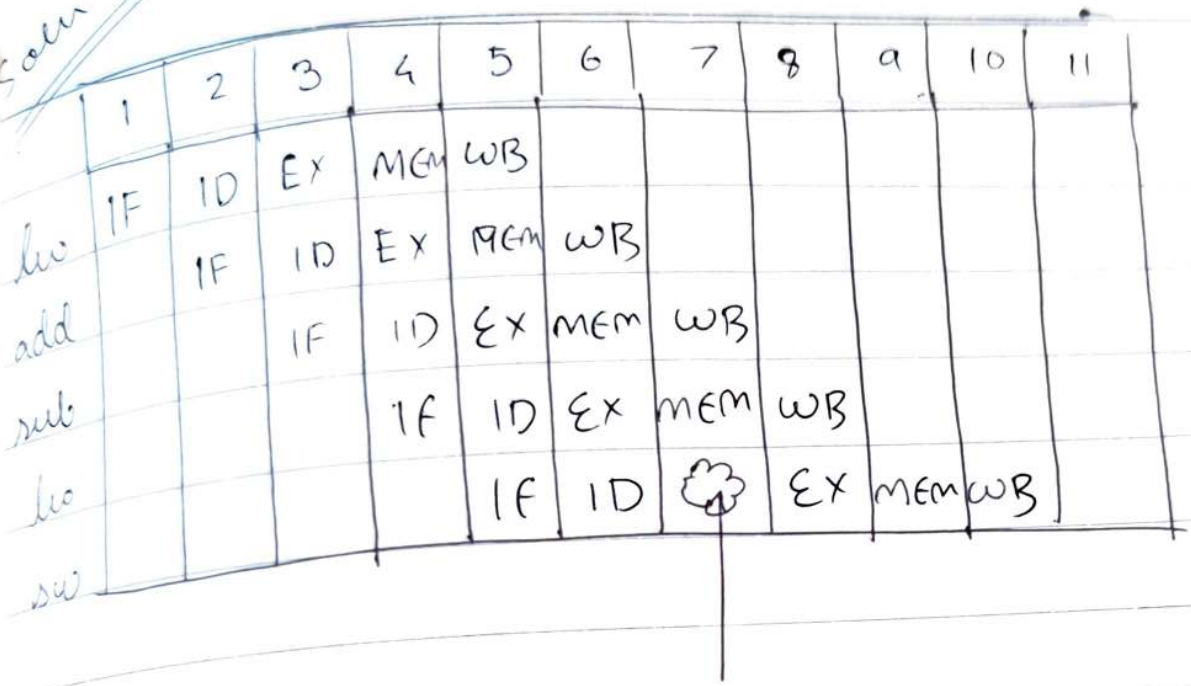
$$\frac{1}{\left(0.2 \times \text{Speedup}_{(2^{nd}\ App)}\right) + \left(0.8 \times \text{Speedup}_{(1^{st}\ App)}\right)}$$

$$= \frac{1}{(0.2) + (0.64)} = 1.19$$

Solution 4 A)

Pipelining increases instruction throughput, thus improving the performance. By having the capacity to execute multiple instructions parallelly, it ensures that the clock period depends only on the latency of the slowest stage. If it were for a non-pipelined case, the dependency would be on all the stages, which would cause both more

Solution 4A.ii



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| lw | IF | ID | EX | MEM | WB | | | | | | |
| add | | IF | ID | EX | MEM | WB | | | | | |
| sub | | | IF | ID | EX | MEM | WB | | | | |
| lw | | | | IF | ID | EX | MEM | WB | | | |
| sw | | | | | IF | ID | EX | MEM | WB | | |

The previous instruction:
[lw x7, 0(x8)]
& the current instruction
[sw x7, 4(x8)]
shall trigger a load use data hazard (The value not being computed in needed time for sw, and there is no avoiding it using forwarding)

4A iii

use

The cause for the load-use hazard triggered is that the x7 register is loaded in the instruction prior to the stall. This is also a uni-cycle stall, as the instruction has to wait until the next cycle, during which the x7 register shall be available in the MEM / WB pipeline, hence forwarded to the "sw".

4B i)

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| beq | IF | ID | EX | MEM | WB | | | | | | |
| lw | | IF | ID | EX | MEM | WB | | | | | |
| sub | | | IF | ID | EX | EX | MEM | WB | | | |
| add | | | | IF | EX | ID | EX | MEM | WB | | |
| lw | | | | | EX | IF | ID | EX | MEM | WB | |
| sub | | | | | | IF | ID | EX | EX | MEM | WB |
| add | | | | | | | IF | ID | ID | EX | MEM |

Load use data hazards triggered

Solution 5 i)

We are aware that in case of direct mapping, each memory block is mapped to a single block in cache.

Let's address the case of more data elements per block & fewer blocks first

Pros

→ Fewer blocks indicate less chance of addresses colliding to the same block. when addressed from the main memory
→ Reduces miss rate due to spatial locality.

Con

Larger blocks indicate fewer no. of total blocks. So, the miss rate is higher due to inter-block access. Also takes a longer duration to fill a block or a miss

Example

Rise is use is spatial locality.

Eg. continuous accss

addrss sequnce is 1, 2, 3, 4, 5, 6 ...

if block size = 4 ; miss rate = $\frac{1}{4}$

due to every 4th accss bring a miss

Although is case of lower block size; miss rate is higher.

In the second case:

Pro: Exploits temporal locality.

Due to multi-variable accss; and more blocks, all ~~cache~~ can be stored in the cache. from conflict misses due to unique mapping

Cons: Can't exploit spatial locality.
In continuous array access miss rate is higher make compulsory miss rate due to ~~higher~~ smaller block size.

E. 1, 2, 3, 4, 5, and repeat

Ans 1

| Instr. | Clock cycles | Frequency (%) |
|---|---|---|
| Alu ops | 1 | 47.5 |
| Loads | 3.5 | 22 % |
| Stores | 2.8 | 10.5 |
| Branches | 4.0 | 15% |
| Jumps | 2.4 | 5% |

i assumed the clock cycles only for the branches taken.

ii Avg. CPI.

$0.475 + (0.22 \times 3.5) + (2.8 \times 0.105) + (4.0 \times 0.15) + (2.4 \times 0.05)$

$= 2.259$