COL216 Minor 2

Rajat Jaiswal

TOTAL POINTS

12 / 15

QUESTION 1

1 Question 1 4 / 4

✓ **+ 4 pts Correct**

+ 2 pts for correct design choice

+ 0.5 pts for hazard illustration(structural hazard)

+ 0.5 pts for hazard handling(mentioning about stalls)

+ 1 pts for suggesting the replacement instruction

- 4 pts Click here to replace this description.

+ 1 pts Design not properly explained.

QUESTION 2

2 Question 2 2 / 3

- 0.5 pts Incorrect definition of Synchronous exception

- 0.5 pts Incorrect definition of Asynchronous exception

✓ **- 0.5 pts Incorrect definition of imprecise exception**

✓ **- 0.5 pts Incorrect definition of precise exception**

- 1 pts Incorrect example

- 0.5 pts The example is given without clearly defining the name of exceptions, or miss explaining why the exception is occurring

- 3 pts Incorrect Answer. See: https://moodle.iitd.ac.in/pluginfile.php/148072/mod_resource/content/1/Minor2_Solution.pdf

- 0 pts Correct Answer.

QUESTION 3

3 Question 3 3.5 / 4

+ 4 pts Correct

✓ **+ 3 pts Correct answer with minor mistakes**

+ 2 pts Click here to replace this description.

+ 1 pts Click here to replace this description.

+ 0.5 pts New Execution time is correct

+ 0.5 pts New CPI is correct

+ 0 pts InCorrect

+ 0.5 Point adjustment

10% reduction in C class makes C class to have 27% instructions.

QUESTION 4

4 Question 4 2.5 / 4

✓ **+ 0.5 pts Evaluation of Performance with one level of Cache : 1 KB Cache**

✓ **+ 0.5 pts Evaluation of Performance with one level of Cache : 16 KB Cache**

✓ **+ 0.5 pts Evaluation of Performance with one level of Cache : 256 KB Cache**

✓ **+ 0.5 pts Identification of Best candidate for Single-Level Cache by analysing the above three evaluations**

✓ **+ 0.5 pts Two level Cache Hierarchy : Computation of correct Miss Penalties for both the Levels L1 and L2**

+ 0.5 pts Two level Cache Hierarchy : Identification of correct Miss Rates(from the Data given in the Q4) for both the Levels L1 and L2

+ 1 pts Final Calculation of AMAT in 2-Level Cache Hierarchy

+ 0 pts Wrong

AMAT = L1 Hit Time + (1-h1)*(L2 Hit Time) + (1-h1)*(1-h2)*(MM Access Time) . So, AMAT = 0.6 + (10/100)*(3.5) + (10/100)*(2.4/100)*(70) ns = (0.6 + 0.35 + 0.168) ns = 1.118 ns.

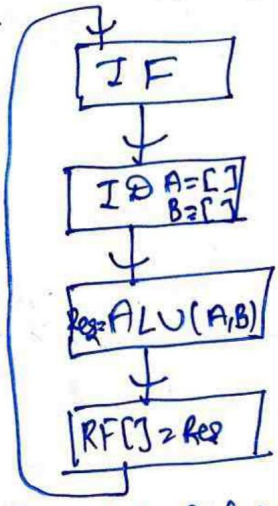
COL216 Computer Architecture Minor Test 2

Date: 25.03.2019 | Time: 09:30 – 10:30 | Max Marks: 15

NAME: RAJAT JAISWAL	ENTRY No.: 2017CS50415

1. Consider a 5-stage pipelined design to execute ARM DP, DT and branch instructions with limited variants (no shift/rotate, no auto-increment/decrement, no register offset for DT instructions, no byte/half-word transfers). The stages are (a) fetch, (b) decode/RF read, (c) ALU/ address calculation, (d) memory access and (e) RF write. It is proposed to add a provision in DP instructions to have operand-2 come from memory. For example, ADD instruction could also have the form ADD Rd, Rn, [Rm], where register Rm gives the address of operand-2. What design changes will be required for this (assume that no additional ALU/adder is to be introduced)? If this change introduces any hazards, indicate how will those be handled. What instruction(s) in a program could be replaced with such instructions? Illustrate your answers. [4]

Traditional DP :-



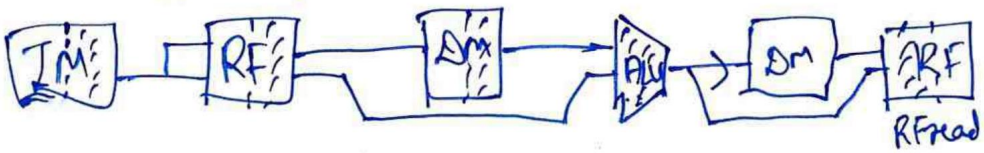
What Rn. We want operand 2 to come from memory so for that first address will have to be calculated in ALU, then that address will be sent to DM for fetching operand & then that operand & Rn will be sent to ALU for addition as in normal DP instruction. Two extra clock cycle will be required for that. These cycles will be introduced in between RF stage & ALU stage & they will be ALU & DM.

It will become 6-cycled process

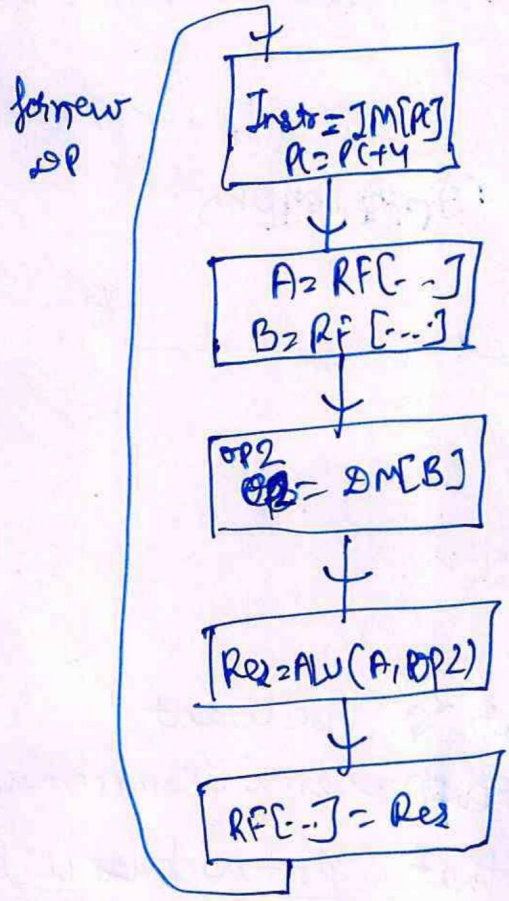
This is all in case when there can be a constant/ offset in address.

But since the question says that address will be completely specified by Rm only & no offset, so no need of ALU in the improved design & hence one less clock cycle will be taken (than previous discussed design)

So overall we need to introduce only a DM stage in between RF read & ALU stage



So DP will become 5 Cycled Process.



This design is not going to produce any structural hazards because DM & IM are seperate.
It is going to produce as usual data hazards, which can be controlled by
(i) stalling / bubbling.
(ii) Data forwarding

instructions such as

ldr R2, [R1]
add R3, R4, R2

can be replaced with this new instruction
as ADD R3, R4, [R1]

Earlier design would have taken 5 cycles for ldr & 4 for add but this design will overall take only 5 cycle.

It requires first loading Operand from memory into a register ~~but now this~~ and then sending that register for calculation. But now, this feature is incorporated in OP itself. So we can combine these two instructions

COL216 Computer Architecture Minor Test 2

Date: 25.03.2019 | Time: 09:30 – 10:30 | Max Marks: 15

NAME: RAJAT JAISWAL	ENTRY No.: 2017CS50415

2. What is the difference between (a) synchronous and asynchronous exceptions and (b) precise and imprecise exceptions. Give an instance when an exception caused by $i+1^{th}$ instruction may be detected before an exception caused by i^{th} instruction in a pipelined design. [3]

instance:
= If we allow inline instructions to be executed in control hazards. Now there is an ith instruction which is branch instruction which raises an exception in its 3rd cycle i.e.. ALU stage. & the next instruction is an undefined instruction which will raised an exception in its first cycle because the instruction address might be out of range for Instruction memory or the instruction is undefined. then in that case exception in (i+1)th instruction is detected before ith instruction.

Since here, 3rd stage of ith instruction = 2nd stage of (i+1)th
& 2nd stage of ith = 1st stage of (i+1)th

(b)
Precise exceptions are those where we exactly know about what kind of exception is raised, giving details about the exception. Whereas imprecise exceptions are those when we don't have much idea about the exception & the error message is very generic. For eg. if in a program division by zero is occurring then exception stating "Division By Zero Not allowed" is a precise exception whereas exception stating "Runtime error" just, is an imprecise exception.

(a) Exception such as invoke OS exception are Synchronous exception & exception such as I/O interrupts are asynchronous exceptions. Mostly Software raised exceptions are synchronous whereas,

mostly asynchronous Exceptions are h/w raised.

COL216 Computer Architecture Minor Test 2

Date: 25.03.2019 | Time: 09:30 – 10:30 | Max Marks: 15

NAME: RAJAT JAISWAL	ENTRY No.: 2017CS50415

3. A processor is designed to work with 2 MHz clock. It has 3 classes of instructions with their CPI and instruction count (in percentage) in a benchmark as given in the table shown.

Instruction class	CPI	% in benchmark
A	3	20
B	4	50
C	5	30

Find the speed-up as a combined result of the following changes.
(a) Clock frequency is increased by 10%.
(b) CPI of instructions of class A is reduced by 1.
(c) Compiler generates 10% fewer instructions of class C for the given benchmark.

[4]

2MHz clock one clock cycle $= \frac{1}{2\times10^6} = 500\,ns = 0.5\,\mu s$

Avg CPI $= 3\times0.2 + 4\times0.5 + 5\times0.3 = 0.6 + 2 + 1.5$

$= 4.1$ CPI

Old_Execution_time/inst $=$ CPI$_{avg}$ * clockcyle $= 500ns \times 4.1$

$= 2050\,ns$

$= 2.05\,\mu s$

(a) new clock freq $= 2.2$ MHz

new one clock cycle time time period $= \frac{1000}{2.2}$ ns

new_execution_time/inst $= 4.1 \times \frac{1000}{2.2} = 1863.6\,ns = 1.8636\,\mu s$

$\frac{\text{old execution time}}{\text{new execution time}}$ = Speed-up$_2$ $= \frac{500\times4.1}{4.1\times\frac{1000}{2.2}} = \frac{500\times2.2}{1000} = 1.1$

It is 1.1 times faster.

(b) new Avg CPI $= 2\times0.2 + 4\times0.5 + 5\times0.3$

$= 0.4 + 2 + 1.5 = 3.9$ CPI

new_execution time $= 3.9\times500ns = 1950ns = 1.95\,\mu s$

$\frac{\text{old time}}{\text{new time}}$ = Speed-up $= \frac{500\times4.1}{500\times3.9} = \frac{41}{39} = 1.051$

It is 1.051 times faster.

(c) Compiler generate 10% fewer C4 instr as it's

new Avg CPI = $3 \times 0.2 + 4 \times 0.5 + 5 \times 0.2$

$= 0.6 + 2 + 1 = 3.6$ CPI

new execution time = $3.6 \times 500\text{ns} = 1800\text{ns} = 1.8\mu s$

Speed-up = $\frac{\text{old execution time}}{\text{new execution time}} = \frac{2.05}{1.8} = \frac{41}{36} = 1.138$

∴ It is 1.138 times faster.

As a combined result of these three changes.

new Avg_CPI = $2 \times 0.2 + 4 \times 0.5 + 5 \times 0.2$

$= 0.4 + 2 + 1 = 3.4$ CPI

new one clock cycle period = $\frac{1}{2.2}\mu s = \frac{1000}{2.2}$ ns.

new execution time = $\frac{1000}{2.2} \times 3.4$ ns.

Speed-up = $\frac{\text{old execution time}}{\text{new execution time}}$

$= \frac{500 \times 4.1}{\frac{1000}{2.2} \times 3.4} = \frac{4.1 \times 1.1}{3.4}$

$= 1.326$

So, Overall combined it is 1.326 times faster.

COL216 Computer Architecture Minor Test 2

Date: 25.03.2019 | Time: 09:30 – 10:30 | Max Marks: 15

NAME: RAJAT JAISWAL	ENTRY No.: 2017CS50415

4. A system has a main memory with access time 70 ns. Hit time and miss rate for three different cache sizes are shown in the table shown. Assume that the processor cycle time is determined by the L1 hit time.

Cache size	Miss rate	Hit time
1 KB	10%	0.6 ns
16 KB	3.4%	1.0 ns
256 KB	2.4%	3.5 ns

If only one level cache is to be used, which cache size will give the best performance? How will the combination of 1 KB L1 cache and 256 KB L2 cache be in comparison with the best 1 level cache?

[4]

for only 1 level cache miss penalty = 70ns.

assuming one memory access/instr.

Performance :

Avg memory access time = Hit time + miss rate * miss penalty
mem_accs/instr

for only one level cache:

(a) 1KB

avg mem access time = $0.6 + 0.1 \times 70 \times 1 = 7.6$ ns

(b) 16KB

avg mem access time = $1 + \frac{3.4}{100} \times 70 \times 1 = 3.38$ ns.

(c) 256KB

avg mem access time = $3.5 + \frac{2.4}{100} \times 70 \times 1 = 1.68 + 3.5$
$= 5.18$ ns

So performance wise if one only one-level cache is used then 16 KB cache gives the best performance.

Reference: 16KB > 256KB > 1KB.

Now combining 1KB as L1 cache and 256KB as L2 cache:

memory stalls/instr = (miss rate$_1$ * miss penalty$_1$ + miss rate$_2$ * miss penalty$_2$) * accs/instr

$= (0.1 * 3.5 + \frac{2.4}{100} * 70) * 1$

2.4
X7
23.8
16.8

$= (0.35 + 1.68)$ ns

$= 2.03$ ns

So avg mem access time in combined cache =

$0.6 + 2.03 = 2.63$ ns

Which is better than using just 16KB cache as one-level cache.

So, L1 (1KB cache) & L2 (256KB cache) is better than just 16KB cache as one-level cache.

It is $\frac{3.38}{2.63}$ times better.

$\frac{3.38}{2.63} = 1.24$

It is 1.24 times better.

$$
\begin{array}{r}
1.24 \\
263\overline{)338} \\
263 \\
\hline
750 \\
526 \\
\hline
1240
\end{array}
$$