Lund
University

Computer Architecture, EITF20

Exercise 1

November 27, 2018

Problem 1

a) Pair a concept from column I in the table below to one in column II
(Each answer should be a number-letter pair)

1. WAR	A. Output dependence
2. Little-endian	B. Anti-dependence
3. ROB	C. LSB at the lowest memory address
4. WAW	D. Result bus providing bypassing
5. Exception	E. Unscheduled event
	F. Multiple instruction dispatch
	G. A buffer for in-order instruction commit
	H. Static optimization for memory system
	I. Improve performance of conditionals
	J. Register renaming

Briefly (1-3 sentences) describe each of the following items concerning computer architecture (where relevant also give an example).

b) Tomasulo's algorithm
c) Precise exceptions
d) Speculative execution

Problem 2

a) Why are the exceptions imprecise in the Tomasulao's algorithm? Is the same statement true for hardware-based speculation with in-order commit?
b) What is the key difference between Scoreboarding and Tomasulo in handling data hazards? Explain.
c) Are there events other than branches that can change the normal flow of instruction execution? Explain.

Problem 3

Three enhancements with the following speed-ups are proposed for a new architecture:

- Enhancement A: Speed-up = 30;
- Enhancement B: Speed-up = 20;
- Enhancement C: Speed-up = 15.

Only one enhancement is usable at a time.

1. How can Amdahl's Law be formulated to handle multiple enhancements?
2. If enhancements A and B are usable for 25% of the time, what fraction of the time must enhancement C be used to achieve an overall speed-up of 10?

Assume the enhancements can be used 25%, 35% and 10% of the time for enhancements A, B, and C respectively.

3. For what fraction of the reduced execution time is no enhancement in use?

Assume, for some benchmark, the possible fraction of use is 15% for each of the enhancements A and B and 70% for enhancement C. We want to maximize performance.

4. If only one enhancement can be implemented, which should it be?
5. If only two enhancements can be implemented, which should be chosen?

Problem 4

Make the following calculations on the raw data in order to explore how different measures color the conclusions one can make. (Doing these exercises will be much easier using a spreadsheet.)

a) Create a table similar to that shown in the figure below, except express the results as normalized to the Pentium D for each benchmark.
b) Calculate the arithmetic mean of the performance of each processor. Use both the original performance and your normalized performance calculated in part (a).
c) Calculate the geometric mean of the performance of each processor. Use both the original performance and your normalized performance calculated in part (a).
d) Given your answer from part (b), can you draw any conflicting conclusions about the relative performance of the different processors? Repeat for part (c).

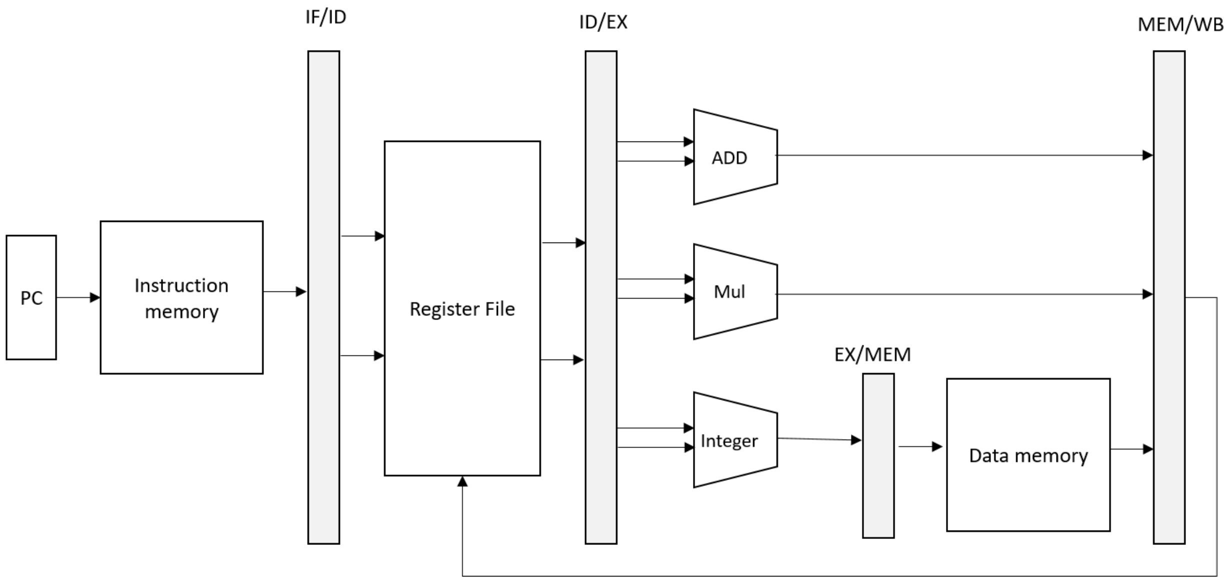
Chip	# of cores	Clock frequency (MHz)	Memory performance	Dhrystone performance
Athlon 64 X2 4800+	2	2,400	3,423	20,718
Pentium EE 840	2	2,200	3,228	18,893
Pentium D 820	2	3,000	3,000	15,220
Athlon 64 X2 3800+	2	3,200	2,941	17,129
Pentium 4	1	2,800	2,731	7,621
Athlon 64 3000+	1	1,800	2,953	7,628
Pentium 4 570	1	2,800	3,501	11,210
Processor X	1	3,000	7,000	5,000

Problem 5

A computer architect has made the following changes to the standard 5-stage pipeline (figure below). But due to not paying attention during his studies (especially his computer architecture course), he has made some critical mistakes in his design!

The pipeline works exactly as before, except that, to speed things up, there are now 3 functional units available, namely, an Add unit (2 cycles), a Multiplication unit (4 cycles) and an Integer unit (1 cycle) for memory address calculation. These Add and Mul units are pipelined, and can have several instructions of the same type in flight.

Additionally, the arithmetic instructions skip the EX/MEM stage of the pipeline (“why not?”, thought the architect) and go directly to the MEM/WB. The forwarding and stalling infrastructure have also been removed entirely to make the clock faster.



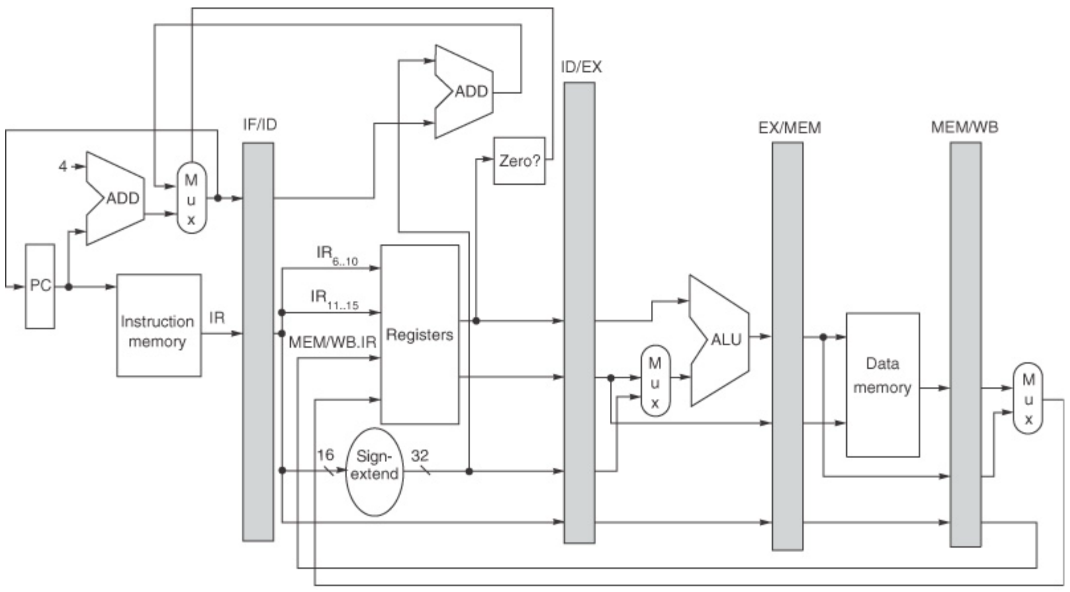
Fill out the following table for the code sequence below. Articulate your reasons as to why the design has problems.

	1	2	3	4	5	6	7	8	9	10	11	12	13
ADDI R0, R0, 4													
LD R1, 0(R0)													
MUL R2, R1, R1													
ADD R2, R1, R2													
ADD R2, R2, R3													
SD R2, 0(R0)													

Problem 6

Consider the following MIPS assembly code sequence running on the standard 5-stage pipeline (figure below):

```
1. LD    R1, 0(R3)        ; R1 = Mem[R3]
2. LD    R2, 8(R3)        ; R2 = Mem[R3+8]
3. ADD   R3, R1, R2       ; R3 = R1 + R2
4. SUB   R4, R1, R2       ; R4 = R1 – R2
5. MUL   R4, R3, R4       ; R4 = R3 * R4
6. SD    R4, 0(R3)        : Mem[R3] = R4
```



Answer the following questions assuming the pipeline stalls when needed, but is lacking any forwarding capability at the moment (except at the WB stage of course).

a) Identify and state all dependences in the code fragment above (including the ones through memory, if any). Which of these dependences leads to hazards?
b) How many clock cycles does it take to execute and finish all the instructions in the code? Draw an instruction execution timing table (clock number vs instruction number with entries showing the pipeline stages).
c) Propose your own suggestions to put the required forwarding / stall mechanism in place, and then redo the timing table based on the improvements. How many clock cycles does it take now to finish all instructions?

Problem 7

Consider the loop below running on a **two-issue** processor.

Loop:	LD	R2, 0(R1)
	ADDI	R2, R2, #1
	SD	R2, 0(R1)
	ADDI	R1, R1, #8
	BNE	R2, R3, Loop

a) Fill out the tables for the first three iterations of the loop, once without speculation and once with it. Assume there are separate units for address calculation, ALU operations, and branch condition evaluation. Assume that up to 2 instructions of any type can be committed per clock.

b) Is the speculative execution faster for two iterations? How do you think it is affected by running more iterations?

Instruction	Issues at cycle	Starts execution at cycle	Mem access at cycle	Writes CDB at cycle
LD R2, 0(R1)				
ADDI R2, R2, #1				
SD R2, 0(R1)				
ADDI R1, R1, #8				
BNE R2, R3, Loop				
LD R2, 0(R1)				
ADDI R2, R2, #1				
SD R2, 0(R1)				
ADDI R1, R1, #8				
BNE R2, R3, Loop				
LD R2, 0(R1)				
ADDI R2, R2, #1				
SD R2, 0(R1)				
ADDI R1, R1, #8				
BNE R2, R3, Loop				

Instruction	Issues at cycle	Starts execution at cycle	Mem read access at cycle	Writes CDB at cycle	Commit/Mem write at cycle
LD R2, 0(R1)					
ADDI R2, R2, #1					
SD R2, 0(R1)					
ADDI R1, R1, #8					
BNE R2, R3, Loop					
LD R2, 0(R1)					
ADDI R2, R2, #1					
SD R2, 0(R1)					
ADDI R1, R1, #8					
BNE R2, R3, Loop					
LD R2, 0(R1)					
ADDI R2, R2, #1					
SD R2, 0(R1)					
ADDI R1, R1, #8					
BNE R2, R3, Loop					