


National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Advance Computer Architecture	Course Code:	EE502
	Program:	MS (Computer Science)	Semester:	Spring 2017
	Duration:	180 Minutes	Total Marks:	70
	Paper Date:	10-05-2017	Weight	45
	Section:	ALL	Page(s):	9
	Exam Type:	Final		

Student : Name: _____ Roll No. _____ Section: _____

Instruction/Notes: 1. Attempt all question in the provided space. You can use rough sheets but it should not be attached.
2. If you think some information is missing, write your assumption and solve accordingly.

Question1: _____ Marks 15

Explain the purpose of following techniques. How we use them and what benefit we obtain by employing these techniques.

1. Reorder buffer

If we speculate a branch and are wrong, we need to back up and restart execution to point at which we predicted incorrectly:

- This is exactly the same as precise exceptions as in case of exception we need to roll back those instruction which are after that instruction that produced exception.
- Technique for both precise interrupts/exceptions and speculation: *in-order completion or commit*

Reorder buffer (ROB) helps to achieve this purpose by buffering the results of uncommitted instructions.

- An instruction commits when it completes its execution and all its predecessors have already committed
- Once instruction commits, result is put into register
 - Therefore, easy to undo speculated instructions on mis-predicted branches or exceptions
- It also supplies operands between execution complete & commit

2. Pipelined caches

This concepts is similar to the processor pipelining, a single task is divided into multiple tasks to reduce the clock cycle time. In pipelined cache, the functionality of accessing cache is divided into multiple tasks thus requiring multiple cycles with reduced cycle time to access cache.

- Increases branch mis-prediction penalty
- Makes it easier to increase associativity

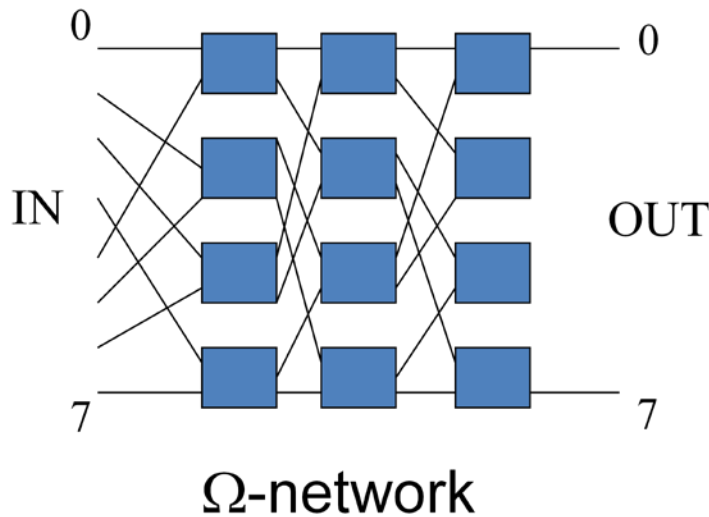
3. Non blocking caches

Normally when a cache miss occur, cache stops and wait till the required resources is loaded from memory however a non-blocking cache does not block on a miss but allow hits before previous misses complete. There are multiple types of strategies available. For example:

- “Hit under miss”
- “Hit under multiple miss”
- In general, processors can hide L1 miss penalty but not L2 miss penalty.
- To implement this strategy, Out of Order processor is required.
- It makes cache control much more complex.

4. Omega Network

Omega network is a technique to connect multiple processors to memory modules. It is less costly than cross bar in terms of connection required and more robust than bus network as it has more than one path shared by multiple processors.



5. VLIW

A VLIW computer is based on an architecture that implements Instruction Level Parallelism (ILP).

In this technique, a Very Long Instruction Word (VLIW) specifies multiple numbers of primitive operations that are grouped together. They are passed to a register file that executes the instruction with the help of functional units provided as part of the hardware.

1. The compiler analyzes dependence of all instructions among sequential code, tries to extract as much parallelism as possible.
2. Based on the analysis, the compiler re-codes the piece of sequential code in VLIW instruction words.
3. Finally, the work left with VLIW hardware is only fetch the VLIWs from cache, decode them, and then dispatch the independent primitive instructions to corresponding function units and execute.

Question2:

Marks 15

Consider the following program that will be executed on a **2-issue Superscalar MIPS** architecture with **Static Branch Prediction (BACKWARD TAKEN)**

Consider the following specification for our pipelined processor:

- Five stages MIPS pipeline with **execution stage consuming 2 clock** cycles whereas all other stages consume 1 cycle.
- **Forwarding is implemented**
- Computation of PC and TARGET ADDRESS for branch & jump instructions anticipated in the **ID stage**

L1: lw R2, 50 (R4)
 addi R2, R2, 4
 lw R3, 150 (R4)
 addi R3, R3, 8
 add R5, R2, R3
 sw R5, 250 (R4)
 addi R4, R4, 4
 bne R4, R7, L1

- a. In-order issue and out-of-order execution and completion rules are applied. For each cycle, show which instructions is at which stage by filling the following table. You don't need to write complete instructions. Just write the instruction numbers from the code above. Remember you cannot re-arrange the instructions for this part of the question.

Cycle No.	IF	ID	EX	MEM	WB
1	1, 2				
2	3, 4	1, 2			
3	5, 6	2, 3	1		
4	7, 8	2, 4	1, 3		
5		2, 4, 5	2, 3	1	
6		5, 6	2, 4	2	1
7		5, 6	2, 4	3	3
8		5, 6	4	4	
9		6, 7	5	4	2
10		6, 8	5, 7		4
11			6, 7	5	
12			6, 8	7	5
13			8	6, 8	7
14				8	6, 8
15					8

- b. Now unroll the loop for level-2 (2 iterations only) and reschedule (rearrange) the instructions to get best execution time with the specifications described above. Also mention the changes required in instructions.

Unroll		Reschedule	
LW	R2, 50(R4) — 2 stall	LW	R2, 50(R4)
add	R2, R2, 4	LW	R6, 54(R4)
LW	R3, 150(R4) — 2 stall	LW	R3, 150(R4)
addi	R3, R3, 8 — 1 stall	addi	R2, R2, 4
add	R5, R2, R3 — 1 stall	LW	R7, 154(R4)
SW	R5, 250(R4)	addi	R6, R6, 4
LW	R6, 54(R4) — 2 stall	addi	R3, R3, 8
addi	R6, R6, 4	addi	R7, R7, 8
LW	R7, 154(R4) — 2 stall	add	R5, R2, R3
addi	R7, R7, 8 — 1 stall	add	R8, R6, R7
add	R8, R6, R7 — 1 stall	add	R8, R6, R7
SW	R8, 254(R4)	SW	R5, 250(R4)
addi	R4, R4, 8 — 1 stall	addi	R4, R4, 8
bne	R4, R7, L1	SW	R8, 246(R4)
		bne	R4, R7, L1

- c. Show the execution of this sequence of instruction for In-order issue and out-of-order execution and completion. Don't write whole instructions instead use the Instruction number assigned in part a.

completion. Don't write whole instructions instead use the instruction number assigned in part a.

Cycle No.	IF	ID	EX	MEM	WB
1	1,2				
2	3,4	1,2			
3	5,6	3,4	1,2		
4	7,8	5,6	3,4		
5	9,10	7,8	5,6	1,2	
6	11,12	9,10	7,8	3,4	1,2
7	13,14	11,12	9,10	5,6	3,4
8		13,14	11,12	7,8	5,6
9			13,14	9,10	7,8
10				11,12	9,10
11				13,14	11,12
12					13,14
13					
14					
15					
16					
17					
18					

Question 3:**Marks 10**

Consider a machine with a byte addressable main memory of 2^{16} bytes and block size of 8 bytes. Assume that a direct mapped cache consisting of 32 lines is used with this machine.

- a. How is a 16-bit memory address divided into tag, index and offset?

Offset = 3 bits

Index = 5 bits

Tag = 8 bits

- b. Into what line would bytes with each of the following addresses be stored?

0001 0001 0001 1011

1101 0000 0001 1101

Both addresses will map to block no. 3.

- c. Suppose the byte with address 0001 1010 0001 1010 is stored in the cache. What are the addresses of the other bytes stored along with it?

As the block size is 16 bytes so all 16 addresses with tag+offset equal to 0001 1010 0001 1 will reside in this block alongwith the given address

- d. If the cache is 4-way set associative then tell the location of values (given in part b) in this cache

There would be 8 sets so the index bits are three. Both addresses will reside in set **no. 3**.

- e. If the cache is fully associative then what would be the size of index bits and tag bits?

There are no index bits and offset bits will remain same 3 bits so the tag bits will be 13.

Question4:**Marks 10**

Consider the following MIPS assembly code to be executed on a pipelined CPU.

```

Loop: beq R6, R7, exit
      lw R2, 0(R6)
      sw R2, 100(R6)
      lw R2, 200(R6)
      addi R2, R2, R6
      sw R2, 300(R6)
      addi R6, R6, 4
      j Loop
exit

```

Your task is to do dynamic scheduling using **Scoreboard**. There are:

- 3 LOAD/STORE Functional Units (LDU1, LDU2, LDU3) with latency of 4 cycles
- 2 ALU/BR/J Function Units (ALU1, ALU2) with latency 2 cycles

Moreover you have to identify:

- Structural hazards in ISSUE phase
- RAW hazards in Read Operand phase
- Check WAR and WAW in Write back phase
- Forwarding is implemented

Assume static branch prediction (backward taken, forward not taken) is used.

Instruction status:

Instruction	Issue	Read Operands	Exec Start	Exec Comp	Write Result
Loop: beq R6, R7, exit	1	2	3	4	5
lw R2, 0(R6)	2	3	4	7	8
sw R2, 100(R6)	3	8	9	12	13
lw R2, 200(R6)	4	5	6	9	10
addi R2, R2, R6	5	10	11	12	13
sw R2, 300(R6)	6	13	14	17	18
addi R6, R6, 4	7	11	12	13	14
j Loop	8	12	16	17	18
exit	9	13	16	17	18

Handwritten annotations: RAW hazards are marked between instructions 2-3, 3-4, 4-5, and 5-6. Structural hazards (S.H.) are marked for instructions 6 and 8.

Register result status:

R0	R1	R2	R3	R4	R5	R6	R7

Question 5:**Marks 10**

Processor X has a clock speed of 1 GHz, and takes 1 cycle for integer operations, 2 cycles for memory operations, and 4 cycles for floating point operations. Empirical data shows that programs run on Processor X typically are composed of 35% floating point operations, 30% memory operations, and 35% integer operations. You are designing Processor Y, an improvement on Processor X which will run the same programs and you have 2 options to improve the performance:

1. Increase the clock speed to 1.2 GHz, but memory operations take 3 cycles
2. Decrease the clock speed to 900 MHz, but floating point operations only take 3 cycles

Compute the speedup for both options and decide the option Processor Y should take.

Speedup

Processor Y

① Clock speed = 1.2 GHz
Mem op take = 3 cycles

Execution time Processor Y (i) =
$$\frac{(0.35)(4) + (0.30)(3) + (0.35)(1)}{1.2 \text{ GHz}}$$

$$= 2.65 / 1.2 \text{ GHz} = 2.20 \text{ ns}$$

Speed up =
$$\frac{\text{exe old}}{\text{exe new}} = \frac{2.35}{2.20} = 1.06^+$$

Processor Y

② Clock speed = 900 MHz
FP take = 3 cycle

Execution time Processor Y (ii) =
$$\frac{(0.35)(3) + (0.30)(2) + (0.35)(1)}{900 \text{ MHz}}$$

$$= \frac{2}{900 \text{ MHz}} = \frac{2 \times 10^3}{900 \text{ GHz}} = 2.22 \text{ ns}$$

Speed up =
$$\frac{\text{exe old}}{\text{exe new}} = \frac{2.35}{2.22} = 1.058$$

Department of Computer Science

Page 8

Option A is better.

Question 6:**Marks 10****Suppose we have following two configurations**

- a. 32-bit operating system, 4-KB pages, 1 GB of RAM
- b. 64-bit operating system, 16-KB pages, 16 GB of RAM

1. How many bits are required for each of the following entries: P stands for Physical and V stands for Virtual

Config.	V. Addr bits	P. Addr bits	V. Page no. bits (index in V. addr)	P. Page no. bits (index in P. addr)	Offset
a	32	30	20	18	12
b	64	34	50	20	14

2. What are some advantages and disadvantages of using a larger page size?

Large page size will reduce the number of entries of page table thus reducing the size required to save page table. It may result in reduced number of page faults if the data accesses are spatially located. However, if the data is random then the page faults may increase. Another disadvantage of using large page size is that the time required to load page will increase thus increasing the latency of page fault.

3. What is TLB cache and what purpose it serves?

Translation look aside buffer are the caches that are used to reduce the time required to translate the virtual address to physical address. A portion of the page table is saved in TLB cache. So a virtual address is checked in this cache first. In case of a hit, translation is instantly available without accessing main memory thus reducing the time required for translation.