EECS 370 - Fall20 Final Exam

Question 1: Short Answer T/F

Specify whether statements a-j are True or False. (On the exam a random set of 10 questions was chosen)

	True/False
a. There are 32-bit two's complement integers that cannot be represented in a 32-bit IEEE-754 single precision number.	
b. Deeper pipelines will generally have a shorter clock period.	
c. It is possible for a write-back and a write-through cache to perform the same number of writes.	
d. For a given cache size and block size, increasing associativity will reduce cache misses for any program.	
e. Resolving data hazards with detect and stall and control hazards with speculate and squash will produce the least number of errors	
f. Direction prediction for jalr is worse than beq.	*Outdated*
g. Fetch stage can read an instruction from memory when there is a lw/sw in the MEM stage.	
h. Physical memory and disk is typically large enough to hold all of a process's virtual memory.	
i. DRAM is a non-volatile memory.	
j. You have scrolled through the whole exam to understand point distribution and allocate your time as such.	

k. If a program accesses the same cache block repeatedly, it is has higher temporal locality	
l. Multi-level page tables will always take up less space than single-level page tables.	
m. Cache is about the same size as main memory.	
n. Dirty bits are needed for write-through caches but not for write-back caches.	
o. Assuming we have an allocate-on-write store policy, an infinitely large cache would have only one miss per cache block.	
p. The number of block offset bits is equal to log_2(size of the cache).	
q. For a given cache configuration, a write-through cache may write fewer bytes to memory than a write-back cache running the same program.	
r. Virtual memory offers the illusion of infinite memory space.	
s. A disadvantage of having a deeper pipeline is that there will be more stalls from data and control hazards.	
t. For a given cache size and block size, increasing associativity will reduce tag size.	

Question 2: Points _/8

Assume the following LC2K program is executed on the 5-stage pipeline from lecture until it halts.

1		lw	0	1	neg1
2		lw	0	2	pos1
3		lw	0	3	five
4		lw	0	4	count
5	loop	add	4	2	4
6		add	3	1	3
7		sw	0	4	count
8		beq	0	3	end
9		beq	0	0	loop
10	end	halt			
11	neg1	.fill			-1
12	pos1	.fill			1
13	five	.fill			5
14	count	.fill			0

A. Select all registers that cause a data hazard in a detect and stall pipeline.
    a. reg0
    b. reg1
    c. reg2
    d. reg3
    e. reg4
    f. reg5
    g. reg6

B. Using detect-and-stall to resolve data hazards, what is the number of stalls that occur due to data hazards?

C. Using detect-and-forward to resolve data hazards, what is the number of stalls that occur due to data hazards?

D. Using speculate-and-squash, along with predicting branches always-not-taken to resolve control hazards, what is the number of stalls that occur due to branch mispredictions?

Question 3: Pipeline Design (11 Points)

Consider the LC2K pipeline with the following change:

- Execution (EX) stage split into two (EX1, EX2).
- EX1 calculates the branch destination and performs the equality check of *beq*.
- EX2 utilizes the ALU for the *add/nor/lw/sw* instructions. EX2 also updates the PC for branches, if necessary.
- Assume all values are forwarded to EX1.

Control hazards resolved using **speculate and squash**, where branches are predicted to be **not-taken**.

Data hazards resolved using **detect and forward**.

Consider the following LC2K program:

```
1.    lw    0    1    0
2.    lw    0    2    1
3.    beq   2    2    1
4.    add   1    1    1
5.    add   1    2    1
6.    add   2    2    3
7.    Halt
```

You are a hacker that has somehow managed to take the following snapshot of the pipeline registers while executing the above program **before cycle C**. Your goal is to determine a secret number that is stored in **virtual memory address 0**.

IF/ID	
instr:	X X X X
PC+1:	X

ID/EX1	
instr:	X X X X
PC+1:	X
regA val:	
regB val:	
offset:	

EX1/EX2	
Instr: (A)	_ X X X
PC Plus 1:	X
regA val:	X
regB val:	X
branchTarget:	X

EX2/MEM	
Instr: (B)	_ _ _ _
AluResult:	36
regB val: (C)	_

MEM/WB	
Instr: (D)	add _ _ _
writeData:	60

WB/END	
instr:	noop X X X
writeData:	X

List all the data hazards. Specify each data hazard in the following format:

line #X, line #Y, number-of-cycles-stalled

where X, Y are lines in the LC2K program that have a read after write dependency causing the data hazard.

____

List all the control hazards. Specify each control hazard in the following format:

line #X, number-of-cycles-stalled

where X is a line in the LC2K program that causes the control hazard.

____

How many instructions does the LC2K program execute? (enter only the number here)

____

Fill in the missing blanks in the highlighted boxes from the pipeline snapshot above.
EX1/EX2 - (A): _________________

EX2/MEM - (B): _________________

EX2/MEM - (C): _________________

MEM/WB - (D): _________________

If we assume that the first instruction of the program is in the fetch stage during Cycle 0, before which cycle (C) was the pipeline snapshot taken? (show your work)

____

What is the secret number at memory location 0? (enter only the number here)

____

Question 4: Pipeline Performance (6 points)
Consider a normal 5-stage LC2K pipeline as discussed in class with the following features:

- Using **detect-and-forward** to handle data hazards.
- Using **speculate-and-squash** to handle control hazards and always predict "Not Taken".
- Branches are resolved in the MEM stage.
- Data memory access (and the critical timing path in the MEM stage) is 10 ns, while the critical path in every other stage is 6 ns

Assume a benchmark with the following characteristics will be run on this pipeline:

- add/nor: 50%
- beq: 15%
- lw: 30%
- sw: 5%
- 40% of all branches are Taken
- 20% of lw instructions are immediately followed by a dependent instruction.
- 10% of lw instructions (disjoint from the previous percentage) are followed by a dependent instruction, but have a single non-dependent instruction between the lw and the dependent instruction.

What is the CPI of this pipeline when running this benchmark?

Now, the MEM stage is split into two stages. It reduces the cycle time by splitting the data memory access latency equally between the MEM stages. Branches are resolved in the first MEM stage.

What is the new CPI?

Assuming 10 billion instructions execute, what is the total execution time for both the original (subquestion 1 above) and modified (subquestion 2 above) pipeline? *Show your work for credit.*

Question 5: Reverse Engineering the Cache (14 points)
Assume the following:

- 8-bit byte-addressable ISA
- Cache size: <= 32B (not including overhead).
- Associativity: Fully-associative cache
- Cache block size and number of sets are both powers of two.
- Cache is initially empty.

Given the following cache outcomes (hit/miss), determine the cache block size and number of cache blocks by answering the following questions.

Access #	**Address**	**Tag**	**Cache Hit/ Miss**
1	0x80		M
2	0xB4		M
3	**0x81**		**H**
4	0xAF		M
5	**0xAC**		**H**
6	0xB3		M
7	0x81		M
8	**0x87**		**H**
9	0x75		M
10	0x79		M

5.1)
Cache **block size** is greater than or equal to (>=) **N** Bytes. Determine maximum value for N and enter **only a number for N** here:

____

Your choice of (>=) cache block size above is known because of which access numbers from the table (choose exactly two)?

____

Cache **block size** is less than (<) **N** Bytes. Determine minimum value for N and enter **only a number for N** here:

____

Your choice of (<) cache block size above is known because of which access numbers from the table (choose exactly two)?

____

Therefore, cache block size is **N** Bytes (enter **only a number for N** here):

____

5.2)
Number of cache blocks is greater than or equal to (>=) **N # of blocks** (enter **only a number for N** here):

____

Your choice of (>=) number of cache blocks above is known because of which access numbers from the table? Choose exactly two accesses with the same tag:

____

Number of cache blocks is less than (<) **N # of blocks** (enter **only a number for N** here):

____

Your choice of (<) number of cache blocks above is known because of which access numbers from the table. Choose exactly two accesses with the same tag:

____

Therefore, number of cache blocks is **N # of blocks** (enter **only a number for N** here)?

____

Question 6: 3 C's (2 Points)
Consider a scenario where we increase the cache block size in a set-associative cache. Cache size and associativity is kept unchanged. Choose the answer for each of the following 4 sub-questions.
6.1) Number of sets:
- ❏ increases
- ❏ decreases
- ❏ remains the same

6.2) Number of cache blocks:
- ❏ increases
- ❏ decreases
- ❏ remains the same

6.3) Compulsory misses:
- ❏ increases
- ❏ decreases
- ❏ remains the same

6.4) Exploits spatial locality:
- ❏ better
- ❏ worse
- ❏ about the same

Question 7: 3 C's Part II (6 Points)
Consider a byte-addressable architecture with the following cache:

Cache size:	64 bytes
Cache block size:	16 bytes
Associativity:	2-way set-associative

Assume that the cache is initially empty, and uses LRU replacement policy. Lower way is chosen when the set is empty.

Answer the following questions for the list of memory references shown in this table. (Empty columns are just workspace).

Reference	Tag	Set Index	Hit/Miss	Type (if miss)	Tag of evicted block (if applicable)
0x0DD5					
0x1F07					
0x02AE					
0x0509					
0x2EDF					
0x1F0F					
0x1F01					
0x48D7					
0x0DDA					
0x0380					

7.1) List all the memory references that incur **conflict misses** in the cache. State each memory reference in the following format:

ROW, address (0xZZZZ), tag (0xZZZ), set-index (0xZ), tag-of-replaced-block-if-any (0xZZZ)

Each letter "Z" above stands for one hexadecimal digit. (e.g.,: ROW-X, 0x1234, 0x120, 0x1, 0x000)

____

7.2) List all the memory references that incur **capacity misses** in the cache. State each memory reference in the following format:

ROW, address (0xZZZZ), tag (0xZZZ), set-index (0xZ), tag-of-replaced-block-if-any (0xZZZ)

Each letter "Z" above stands for one hexadecimal digit. (e.g.,: ROW-X, 0x1234, 0x120, 0x1, 0x000)

____

7.3) At the end of simulating the above address sequence, specify the final state of the 2-way set-associative cache using the tag of the cache blocks stored in the cache.

Way 0, Set 0 Tag: _________________

Way 0, Set 1 Tag: _________________

Way 1, Set 0 Tag _________________

Way 1, Set 1 Tag: _________________

Question 8: Hierarchical Page Tables (5 Points)
Assume 48-bit byte-addressable ISA system that supports virtual memory with the following specifications:

● 3-level page table	
● Page size:	4 KB
● Physical memory size:	16 GB
● Size of each 3rd level page table:	8 pages
● Size of each 2nd level page table:	1 page
● Page table entry size:	4 bytes

Determine the following:
Page Offset Size # bits (enter only the number here):

Size of physical page number (PPN) # bits (enter only the number here):

Size of 3rd level page table index # bits (enter only the number here):

Size of each 3rd level page table, 2^**N** Bytes (enter only the exponent number **N** here):

Size of 2nd level page table index # bits (enter only the number here):

Size of each 2nd level page table 2^**N** Bytes (enter only the exponent number **N** here):

____

Size of 1st level page table index # bits (enter only the number here):

____

Size of 1st level page table 2^**N** Bytes (enter only the exponent number **N** here):

____

In the worst case, how much memory would this 3-level page table occupy?
State your answer in the form 2^1 + 2^2 + … + 2^n bytes.

____

For the same page size, if this system used a single-level page table instead of a three-level page table, how much memory would it occupy in the worst case?
State your answer in the form 2^1 + 2^2 + … + 2^n bytes.

____

Question 9: Hierarchical Page Tables (6.5 Points)
Assume 16-bit ISA that uses a 3-level page table. Each virtual memory address is split into following fields:

1st level index size: 4 bits	2nd level index size: 6 bits	3rd level index size: 4 bits	Physical page offset size: 2 bits

Consider the following virtual addresses accessed in the order given below. Determine the 1st, 2nd and 3rd level indices for each access.

Virtual Address	1st level index	2nd level index	3rd level index
0xA7E4			
0xB09B			
0xA78F			
0xC78F			
0xB098			

How many first-level page tables will have been allocated after these 5 memory accesses? (enter only the number here)

____

How many second-level page tables will have been allocated after these 5 memory accesses? (enter only the number here)

____

How many third-level page tables will have been allocated after these 5 memory accesses? (enter only the number here)

____

Question 10: Virtual Memory (11 Points)
Assume 16-bit ISA and the following system configuration.

Cache:
Physically addressed
2-way set associative

Block size:	4 bytes
Cache size:	16 bytes

Latency for each memory component:

Disk:	1000 ns
Physical memory:	50 ns
TLB:	2 ns
Cache:	1 ns

Memory system:
Physical memory size: 4KB
Single level page table.
The TLB has 2 entries and is fully associative

Assume the following:

- The first 8 pages (PPN: 0 - 7) in the physical memory are empty and free to use. And the rest are reserved for the OS. The page table is pinned in the physical memory. It is not cached, except in TLB.
- Pages that are not in memory are on disk
- All updates are in parallel. Upon retrieval from cache, main memory or disk, the data is sent immediately to the CPU, while other updates occur in parallel

A process accesses the following virtual addresses in order. Latency for each memory access is given.

Based on the access latencies, determine the outcomes of cache and TLB access, and whether it is a page fault or not.

Access Number	Virtual Address	Latency (ns)	TLB(H/M/NA)	Cache (H/M/NA)	Page Fault (Y/N)
1	0x126C	1052			
2	0x122F	1052			
3	0x122B	53			
4	0x126D	3			
5	0x35AC	1052			

6	0x122A	53			
7	0x125B	103			

Based on the above accesses and their latencies, determine the page size by answering the following questions.

Page size is greater than or equal to (>=) 2^**N** Bytes. Determine the maximum value for N, and enter *only the exponent number* **N** here:

____

Your choice of Page size (>=) is known because of which access numbers from the table (choose exactly two):

____

Page size is less than or equal to (<=) 2^**N** Bytes. Determine the minimum value for N, and enter *only the exponent number* **N** here:

____

Your choice of Page size (<=) is known because of which access numbers from the table (choose exactly two):

____

Therefore, page size is 2^**N** Bytes (enter *only the exponent number* **N** here):

____

Question 11: Stack-Based ISA (12 Points)
Consider a new Stack-ISA with the following instructions.

Its semantics are defined by the following two ISA-visible components:
- A stack memory
- A special register called COMPARE. It can store a value of 1 or 0.

Assembly Instruction	**Execution semantics**

`pushi immediate`	`stack.push(immediate)`
	
	`(Labels resolve to their addresses)`

`push addr`	`stack.push(mem[addr])`

`less`	`first = stack.pop();   second = stack.pop()`
	`if (first < second):`
	`    COMPARE = 1`
	`else:`
	`    COMPARE = 0`

`greater`	`first = stack.pop();   second = stack.pop()`
	`if (first > second):`
	`    COMPARE = 1`
	`else:`
	`    COMPARE = 0`

`bcmp`	`destination = stack.pop()`
	`if (COMPARE == 1):`
	`    COMPARE = 0`
	`    PC = destination`

11.1) Use the instructions in Stack-ISA to implement "Unconditional branch to address 20"

`Branch to PC = 20`	`____________`
	
	`pushi 1`
	
	`pushi 0`
	`____________`
	
	`____________`

11.2) Use the instructions in Stack-ISA to implement "conditional branch to 20 if mem[5] is equal to mem[8]".
(Hint: you should use part a in your implementation)

```
if (mem[5] == mem[8]) {
    Branch to PC = 20
}
```

```
     pushi neq
     ____________

     ____________

     less
     ____________

     ____________

     ____________

     ____________

     greater
     ____________

eq   ____________

     pushi 1

     pushi 0
     ____________

     ____________

neq  noop
```