Homework 8 ● GRADED

Q1 Translation Lookaside Buffer (TLB)

18 Points

The table below represents the current entries in the translation lookaside buffer (TLB).

No.	User/Kernel	VPN	PFN	Valid/Invalid
1	Kernel	5	21	Valid
2	User	105	32	Valid
3	User	120	10	Invalid

Q1.1 Hit / Miss 1

4 Points

While executing the current user process, an attempt to access the page at VPN $= 105$ is made. Is this TLB lookup a hit or miss?

- (●) Hit
- ( ) Miss

1.1 Hit / Miss 1 **4** / 4 pts
- ✔ − 0 pts Correct
- − 4 pts Incorrect

Q1.2 Hit / Miss 2

4 Points

While executing the current user process, an attempt to access the page at VPN $= 120$ is made. Is this TLB lookup a hit or miss?

- ( ) Hit
- (●) Miss

1.2 Hit / Miss 2 **4** / 4 pts
- ✔ − 0 pts Correct
- − 4 pts Incorrect

Q1.3 Hit / Miss 3

4 Points

While executing the current user process, an attempt to access the page at VPN $= 103$ is made. Is this TLB lookup a hit or miss?

- ( ) Hit
- (●) Miss

1.3 Hit / Miss 3 **4** / 4 pts
- ✔ − 0 pts Correct
- − 4 pts Incorrect

Q1.4 Context Switch

6 Points

The processor executes a context switch. Which entries in the TLB are purged?

- [ ] Entry 1
- [x] Entry 2
- [x] Entry 3

1.4 Context Switch **6** / 6 pts
- ✔ − 0 pts Correct
- − 6 pts Incorrect
- − 2 pts Did not mark Entry 2
- − 2 pts Did not mark Entry 3
- − 2 pts Marked Entry 1
- − 6 pts Blank/no answer

Q2

20 Points

In this class, one of the first caching schemes we discuss is direct mapped cache. With that in mind, we will explore how direct mapped cache utilizes locality.

Q2.1 Direct Mapped Locality

5 Points

Which type of locality does direct mapped caching take advantage of?

Temporal

Q2.2 Direct Mapped Locality Explanation

5 Points

Explain how direct mapped cache utilizes one type of locality to speed up memory accesses?

uses temporal localicty to bring recent memory accesses (likely used a lot) into our calculation rather than going to memory. Bring more from memory into cache at a time

Direct Mapped Locality Explanation **5** / 5 pts

✔ + 5 pts	Correct
+ 3 pts	Discusses Spatial locality by assuming the improvement where multiple bytes can be loaded into one cache location.
+ 0 pts	Incorrect (Fails to discuss how direct-mapped cache provides quicker access times if the cpu tries to access a byte that is already stored in the cache)

Q2.3 Direct Mapped Improvement

10 Points

In class, we discuss improvements to direct mapped caching. Discuss what this improvement is and how it utilizes the other form of locality.

It uses spatial locality to bring more words into the program

2.3 Direct Mapped Improvement **10** / 10 pts

✔ + 10 pts	Correct: Discusses spatial locality by loading multiple addresses into the cache at given access.
+ 10 pts	Alternative answer: Discusses Set or Fully associated caching in the context of spatial locality
+ 5 pts	Discusses Set or Fully associated caching in the context of temporal locality
+ 0 pts	Incorrect/Blank/No answer

Q3 Test-and-Set

14 Points

In order to support synchronization between multiple processes, why do we need a test-and-set (T&S) instruction instead of just using existing load, store, and branch if zero instructions?

To support synchronization, we need mutexes. The process for correctly utilizing a mutex is
1.) read a memory location
2.) test if the value read is 0
3a.) if the value is 0 then the mutex is free and we need to set the value to 1
3b.) if the value is 1 then the mutex is locked and we cannot enter a critical section

While we could normally execute these steps using existing load, store, and "branch if zero" instructions, the above steps need to happen atomically (they all need to happen sequentially--no interrupts can split them up). If we used the aforementioned steps and another instruction due to a thread or interrupt splits apart from our process, then we might erroneously read the mutex. The test-and-set instruction is a single instruction so all steps in this process happen atomically meaning nothing intervenes with the execution of the test-and-set instruction.

QUESTION 3

Test-and-Set **14** / 14 pts

✔ − 0 pts	Correct
− 8 pts	Did not identify test-and-set as being used for mutual exclusion/locks/etc. (a synchronization primitive)
− 6 pts	Did not explain why test-and-set is needed for a lock implementation (atomic test-and-set to ensure no data race/consistency/etc.)
− 14 pts	Blank/no answer

Q4 Cache Addressing

24 Points

Listed below are the cache parameters that describe the cache layout for a byte-addressable memory system:

- 4 way set associative
- 32 bit address
- block size of 64 words
- word size of 4 bytes

- total cache size of 128K words (512 KB)

Use these parameters to evaluate how this cache will interpret an address and the sizes of each part of the address. (Also, use 1 KB = 1024 B = 2^10 B)

4.1 — Offset size — **2** / 8 pts

	− 0 pts	Correct
	− 8 pts	Incorrect
✔	− 6 pts	Incorrect offset bit size (8)
	− 2 pts	Incorrect cache address offset (11100100; accept follow through from bit size)
	− 8 pts	Blank/no answer/incorrect

Q4.1 Offset size

8 Points

What is the size (in bits) of the offset?

2

Given the following cache address, what is the offset (in binary)? Assume the cache address follows the format *tag* | *index* | *offset*.

Cache address = 01100100110100101001001111100100

00

Q4.2 Index

8 Points

What is the size (in bits) of the index?

16

Given the following cache address, what is the index (in binary)? Assume the cache address follows the format *tag* | *index* | *offset*.

4.2 — Index — **2** / 8 pts

	− 0 pts	Correct
	− 8 pts	Incorrect
	− 2 pts	Did not use set associativity while collecting bit size (did not divide by 4)
	− 2 pts	Used words/block instead of bytes/block when calculating bit size (divided by 64 instead of 64 * 4 = 256)
✔	− 6 pts	Incorrect index bit size for other/unknown reason (8)
	− 2 pts	Incorrect cache address index (00100111; accept follow through from bit size and previous parts)
	− 8 pts	Blank/no answer/incorrect

Cache address = 01100100110100101001001111100100

0110010011010010

Q4.3 Tag

8 Points

What is the size (in bits) of the tag?

14

Given the following cache address, what is the tag (in binary)? Assume the cache address follows the format *tag* | *index* | *offset*.

4.3 — Tag — **8** / 8 pts

	− 0 pts	Correct
	− 8 pts	Incorrect
✔	− 0 pts	Incorrect bit size due to follow-through from a previous part
	− 6 pts	Incorrect tag bit size for other/unknown reason (15)
	− 2 pts	Incorrect cache address tag (011001001101001; accept follow through from bit size and previous parts)
	− 8 pts	Blank/no answer/incorrect

Cache address = 01100100110100101001001111100100

10010011111001

Q4.4 Work (Optional)

0 Points

For partial credit, show your work in the field below or attach it as a file.

No files uploaded

Q5 Effective Memory Access Time

24 Points

Say we have two setups for hierarchical memory with the following miss rates and hit times:

Memory 1

Hardware	Miss Rate	Hit Time
L1 Cache	0.24	5 ns
L2 Cache	0.11	17 ns
Main Memory	0	55 ns

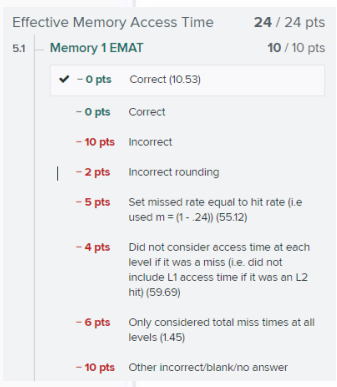
Memory 2

Hardware	Miss Rate	Hit Time
L1 Cache	0.50	5 ns
L2 Cache	0.31	7 ns
Main Memory	0	54 ns

Q5.1 Memory 1 EMAT
10 Points

What is the EMAT (effective memory access time) for Memory 1 in ns? Round your answer to 2 decimal places.

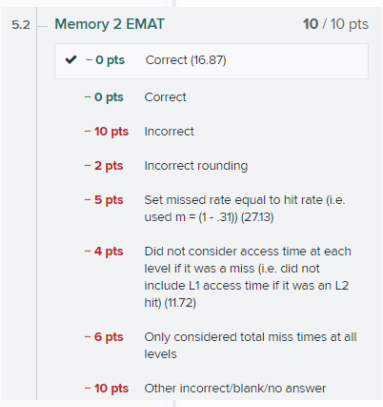
10.53



Q5.2 Memory 2 EMAT
10 Points

What is the EMAT (effective memory access time) for Memory 2 in ns? Round your answer to 2 decimal places.

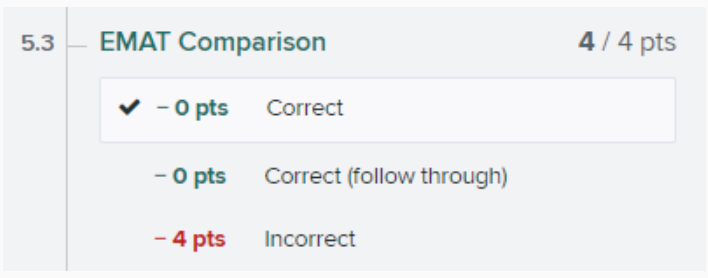
16.87



Q5.3 EMAT Comparison
4 Points

Which setup has a better EMAT?

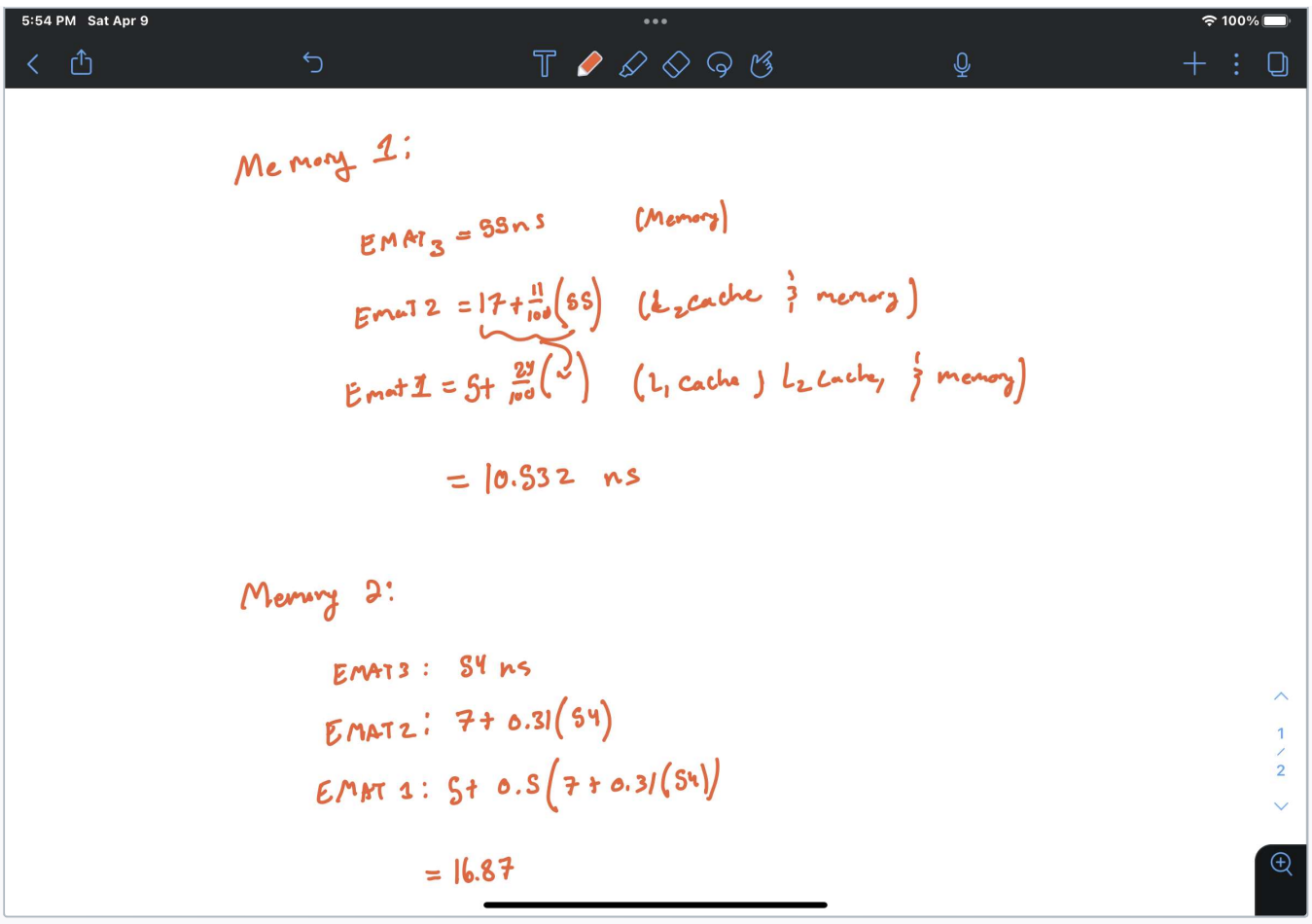
- (●) Memory 1
- ( ) Memory 2



Q5.4 Work (Optional)
0 Points

If you would like partial credit in case of an incorrect answer on the previous parts, show your work in the field below or attach it as a file:

36DD1A40-9C86-43C5-91F5-C24F04229A29.png Download



Homework 8

GRADED

3 DAYS, 1 MINUTE LATE

STUDENT
Eric Anders Gustafson

TOTAL POINTS
88 / 100 pts

QUESTION 1

Translation Lookaside Buffer (TLB) — **18** / 18 pts

1.1 Hit / Miss 1 — **4** / 4 pts

1.2 Hit / Miss 2 — **4** / 4 pts

1.3 Hit / Miss 3 — **4** / 4 pts

1.4 Context Switch — **6** / 6 pts

QUESTION 2

(no title) — **20** / 20 pts

2.1 Direct Mapped Locality — **5** / 5 pts

2.2 **Direct Mapped Locality Explanation** — **5** / 5 pts

✔ **+ 5 pts** Correct

+ 3 pts Discusses Spatial locality by assuming the improvement where multiple bytes can be loaded into one cache location.

+ 0 pts Incorrect (Fails to discuss how direct-mapped cache provides quicker access times if the cpu tries to access a byte that is already stored in the cache)

2.3 Direct Mapped Improvement — **10** / 10 pts

QUESTION 3

Test-and-Set — **14** / 14 pts

QUESTION 4

Cache Addressing — **12** / 24 pts

4.1	Offset size	**2** / 8 pts
4.2	Index	**2** / 8 pts
4.3	Tag	**8** / 8 pts
4.4	Work (Optional)	**0** / 0 pts

QUESTION 5

Effective Memory Access Time — **24** / 24 pts

5.1	Memory 1 EMAT	**10** / 10 pts
5.2	Memory 2 EMAT	**10** / 10 pts
5.3	EMAT Comparison	**4** / 4 pts
5.4	Work (Optional)	**0** / 0 pts