Exam 3

● GRADED

Q1 Memory Fragmentation

12 Points

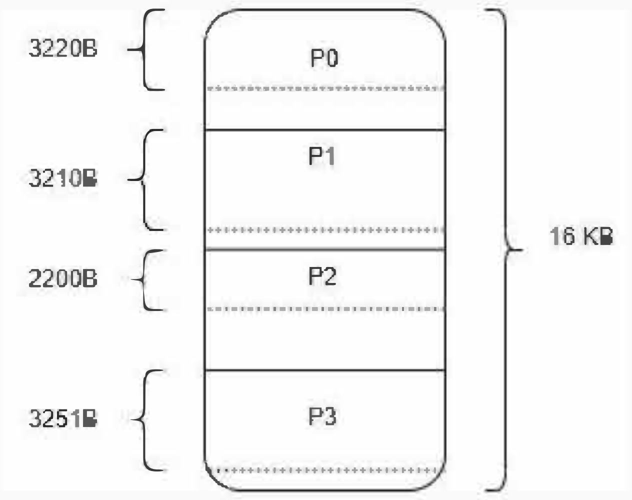
Yesha's memory system has 16KB (1KB = 1024 Bytes) of memory. At a given point in time, there are 4 active processes with the following memory footprints:

P0: 3220 bytes
P1: 3210 bytes
P2: 2200 bytes
P3: 3251 bytes

Q1.1

2 Points

A memory management system that uses memory paging with a page size of 4KB results in the following memory layout:



1.1	(no title)	**0** / 2 pts
	+ 2 pts	(4503 bytes) Correct
✓	**+ 0 pts**	Incorrect

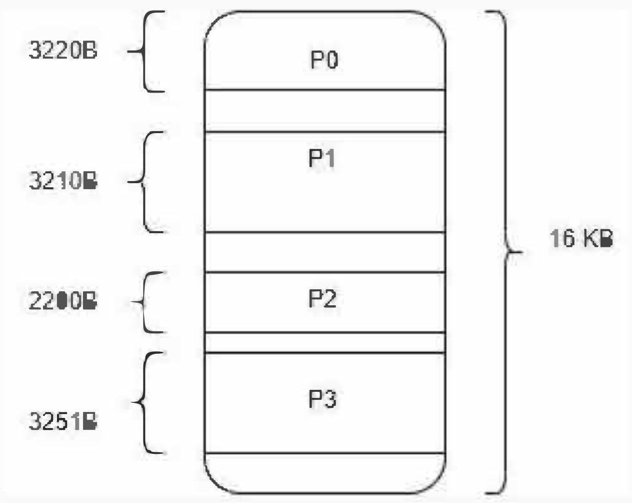
Calculate the total amount of memory lost to internal fragmentation at this time.

16(1024) - (3220+3210+2200+3251) =
4053

Q1.2

3 Points

A different memory management system tries to solve the paged system's internal fragmentation problem by using bounds registers to allocate static memory partitions matching exactly the memory footprint of each process. At a given time, memory looks like this:



We then try to insert a new process, P5, with a memory footprint of 4001 bytes. Is this possible? Explain your reasoning.

No, we would need 4001 contiguous bytes of memory but we do not have that. Once the memory is allocated statically so we won't get 4001 contiguous bytes.

1.2	(no title)	**3** / 3 pts
✓	**+ 1 pt**	Correct answer, but no explanation.
✓	**+ 2 pts**	Mentions about external fragmentation and contiguous memory.
	+ 0 pts	Incorrect

Q1.3

3 Points

Explain an approach that could minimize both internal and external fragmentation in our system.

We could implement virtual memory with paging. This would allow us to eliminate external fragmentation since all pages are contiguous and we can reduce internal fragmentation by reducing the page size.

1.3 (no title) **3** / 3 pts

✓ **+ 1 pt** Correct (base+limit/compaction/segmented virtual memory)

✓ **+ 2 pts** Explains the approach chosen.

+ 0 pts Incorrect

Q1.4
4 Points

This question is independent from all previous assumptions introduced in questions a to c. Charles proposes hardware support for the LC-2200 processor to aid memory management that uses a pair of “bounds” registers to define the memory occupied by a process. The registers are used by the memory manager as follows:

- A process P4 is currently running occupying memory from address 1024 to address 2048 (i.e., the lower bound register would be set to 1024 and the upper bound register would be set to 2048, when P4 is running).
- P4 makes a blocking I/O call and is swapped out.
- Later when the I/O is complete and P4 is ready to run again, the memory manager brings P4 into memory in the address range 4096 to 5120.

Is this the correct system behavior? Explain your answer.

No, we cannot do this. Once the program is allocated statically we cannot then allocate a different section of memory and expect our program to work properly. For example, originally the program will allocate memory for addresses 1024 through 2048 and that's what the program expects, so once the program is swapped back in after the I/O completes then it will still expect to have the program in the original addresses of 1024 through 2048 but it won'tbe there.

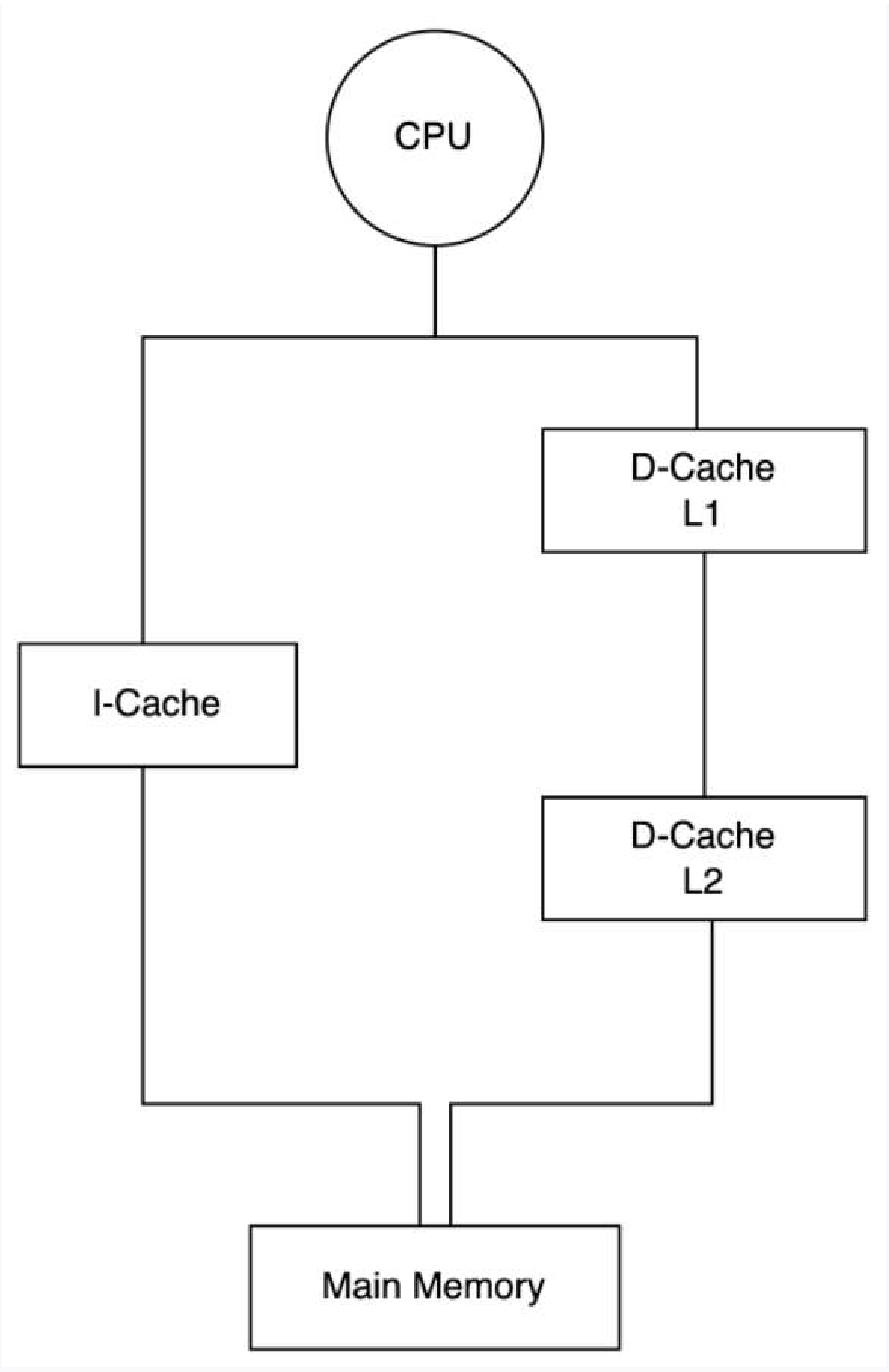
1.4 (no title) **4** / 4 pts

✓ **+ 4 pts** Correct (no the bound register doesn't allow for memory addresses other than 1024 - 2048)

+ 0 pts Incorrect

Q2 Cache Performance
13 Points

Prit’s pipelined processor features separate instruction and data caches. The instruction cache (I-cache) has a single level and the data cache (D-cache) has two levels, as shown in the following figure.



- Assume the processor achieves an **average CPI of 1.2** without accounting for memory stalls.
- I-Cache has a hit ratio of **99%**.
- D-Cache L1 has a hit ratio of **75%**.
- D-Cache L2 has a hit ratio of **85%**.
- Memory reference instructions account for **20%** of all the instructions executed.
- Out of these memory reference instructions **70%** are reads and **30%** are writes.
- Memory access latency for read is **120 cycles**.
- Memory access latency for write is **6 cycles**.
- L1 Cache access time is **2 cycles**.
- L2 Cache access time is **40 cycles**.
- All cache blocks are word-sized.
- Assume **write-allocate (write-through)** policy is being used.

Compute the effective CPI of the processor accounting for the memory stalls. Show your work to receive credit.

Q2.1

2 Points

Calculate the loads and stores as percentages of total instructions.

Loads:

0.14

Stores:

0.06

Show all work here:

reads = .2*.7 = .14
loads = .2*.3 = .06

2.1 (no title) **2** / 2 pts

✓ **+ 2 pts** Correct (Loads: 0.14, Stores: 0.06)

+ 1 pt Only Loads Correct

+ 1 pt Only Stores Correct

+ 0 pts Incorrect Work

+ 0 pts Incorrect

Q2.2
2 Points

Calculate the EMAT for I-Cache read, in terms of CPU cycles.

1.2

Show all work here:

0.01*120

2.2 (no title) **2** / 2 pts

✓ **+ 2 pts** Correct (1.2, I-cache access time = 0)

+ 2 pts Correct (3.2, I-cache access time = 2)

+ 0 pts Incorrect Work

+ 0 pts Incorrect/No response

Q2.3
3 Points

Calculate the EMAT (Effective Memory Access Time) of L2 D-Cache Reads, in terms of CPU cycles. Then, calculate the EMAT of L1 D-Cache Reads, in terms of CPU cycles.

L2 D-Cache Reads:

L1 D-Cache Reads:

16.5

Show all work here:

L1:
2 + .25(40+.15*120) = 16.5

2.3 (no title) **2** / 3 pts

+ 3 pts Correct (L2: 58, L1: 16.5)

+ 1 pt Only L2 Read Correct (58)

✓ **+ 1 pt** Only L1 Read Correct (16.5)

✓ **+ 1 pt** Correct work

+ 0 pts Incorrect work/No work

+ 0 pts Incorrect

Correct Formula, but no show of where the values came from.

Q2.4
3 Points

Calculate the EMAT of L2 D-Cache Writes, in terms of CPU cycles. Then, calculate the EMAT of L1 D-Cache Writes, in terms of CPU cycles. Round to three decimal places.

L2 D-Cache Writes:

L1 D-Cache Writes:

48

Show all work here:

2+1*(40+1*6=48

2.4 (no title) **2** / 3 pts

+ 3 pts Correct (L2: 46 L1: 48)

+ 1 pt Only L2 Read Correct (46)

✓ **+ 1 pt** Only L1 Read Correct (48)

✓ **+ 1 pt** Correct work

+ 0 pts Incorrect Work

+ 0 pts Incorrect

Q2.5
3 Points

Calculate the effective CPI of this system.

Show all work here:

2.5 (no title) **0** / 3 pts

	+ 3 pts	Correct
	+ 1 pt	Correct (7.59, I-cache access time = 0)
	+ 1 pt	Correct (9.59, I-cache access time = 2)
	+ 2 pts	Correct work
✓	**+ 0 pts**	Incorrect Work
	+ 0 pts	Incorrect/No response

Q3 Cache Organization
9 Points

The CS 2200 TA team is tasked with designing a 16KB cache with 64 byte blocks.

Q3.1
3 Points

Malav decides to design the cache as 256-way set-associative. What is the drawback of his decision?

16kB/64bytes = 256, thus this will be a fully associative cache. The drawback here is that we will have to search the entire cache which is very time consuming.

3.1 (no title) **3** / 3 pts

✓	**+ 3 pts**	Correct (Increase in hardware complexity or will lead to an increase to cache access time or increase in search time due to linearly searching through the cache)
	+ 0 pts	Incorrect

Q3.2
3 Points

Borun sees that modern processors typically feature lower associativity than Malav's design and opts for a 4-way set-associative design. Why might this be an improvement on Malav's system?

With a 4-way design the cache is segmented into 4 partitions (0-63, 64-127, 128-191, 192-255) so there are only 4 places we need to check which is a vast improvement over the 256-way set associative (4 places instead of 256).

3.2 (no title) **3** / 3 pts

✓	**+ 3 pts**	Correct (Increasing associativity leads to diminishing returns and reduces hardware complexity)
	+ 0 pts	Incorrect

Q3.3
3 Points

Aidan likes to keep things simple and proposes a cache with 256 sets. What is another name for Aidan's design? Why might it not perform as well as Borun's?

Aidan's design is a 1-way associate set (direct-mapped cache) and each index of the cache has multiple potential mapped locations. This is potentially bad because we may have a lot of conflict misses.

3.3 (no title) **3** / 3 pts

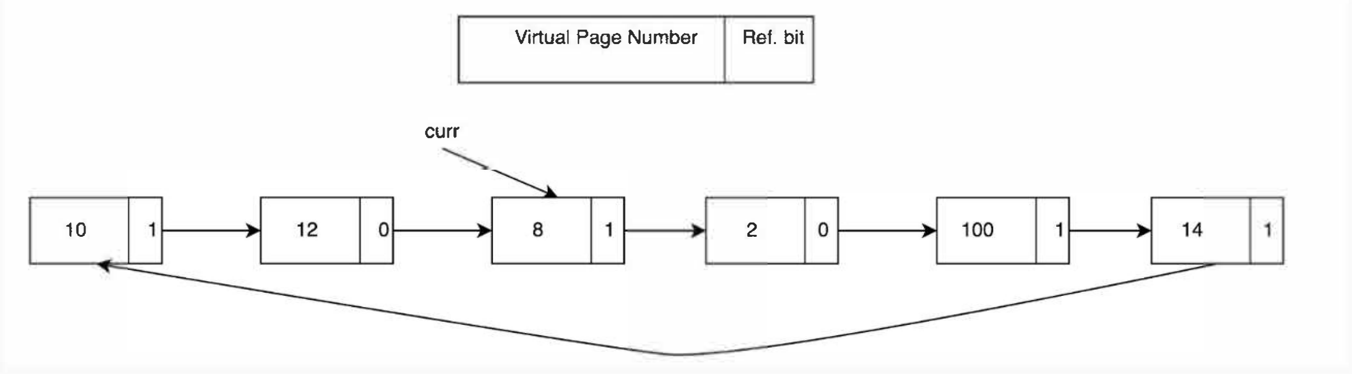
✓	**+ 3 pts**	Correct (Direct-mapped cache suffers from conflict misses.)
	+ 0 pts	Incorrect

Q4 Page Replacement
11 Points

Suppose Ishaan's low-budget memory system consists of 6 physical frames and uses a second chance page replacement algorithm. The following figure shows the current state of the circular queue, showing 6 pages currently occupying the 6 physical frames in the order they were referenced. The 'curr' pointer points to the least recently used page in the queue.

Assume that the implemented policy used to set the reference bit is as follows: when the replacement algorithm identifies the victim page, it sets the corresponding frame's ref bit to 1 and moves the "curr" pointer forward.

Type your queue in this format: (10, 1), (12, 0), (8, 1, curr), (2, 0), (100, 1), (14, 1).



Q4.1
2 Points

At time t = 1, a process accesses virtual page 4. Show the state of the queue after this reference.

(10, 1)→(12, 0)→(8, 0)→(4, 1)→(100, 1, curr)→(14, 1)

4.1 (no title) **2** / 2 pts

✓ **+ 2 pts** Correct (10, 1), (12, 0), (8, 0), (4, 1), (100, 1, curr), (14, 1)

+ 0 pts Incorrect

Q4.2
2 Points

At time t = 2, a process accesses virtual page 13. Show the state of the queue after this reference.

(10, 0)→(13, 1)→(8, 0, curr)→(4, 1)→(100, 0,)→(14, 0)

4.2 (no title) **2** / 2 pts

✓ **+ 2 pts** Correct (10, 0), (13, 1), (8, 0, curr), (4, 1), (100, 0), (14, 0)

+ 0 pts Incorrect

Q4.3
2 Points

At time t=3, a process accesses virtual page 8. Show the state of the queue after this reference.

(10, 0)→(13, 1)→(8, 1, curr)→(4, 1)→(100, 0,)→(14, 0)

we have a hit so no need to do run page eviction algo to find a page to evict so curr stays at VPN 8 and since we accessed it recently the bit is set.

4.3 (no title) **2** / 2 pts

✓ **+ 2 pts** Correct (10, 0), (13, 1), (8, 1, curr), (4, 1), (100, 0), (14, 0)

+ 0 pts Incorrect

Q4.4
2 Points

At time t = 4, virtual page number 25 is brought into the physical memory. Show the state of the queue after this reference.

(10, 0)→(13, 1)→(8, 0)→(4, 0)→(25, 1)→(14, 0, curr)

4.4 (no title) **2** / 2 pts

✓ **+ 2 pts** Correct (10, 0), (13, 1), (8, 0), (4, 0), (25, 1), (14, 0, curr)

+ 0 pts Incorrect

Q4.5
3 Points

At time t = 5, a sequence of accesses occurs to pages 8, 10, 13, 14, 4, 25, in that order. Immediately after, at time t = 6, virtual page 2 is accessed. At this specific point in time (t = 6), would you rather implement FIFO replacement or second chance replacement? Explain your answer.

We should do FIFO since at this point we'll have to go through the entire circularly queue to dequeue an element where with FIFO we can just evict the element where the head of the queue is (O(n) vs O(1)).

4.5 (no title) **3** / 3 pts

✓ **+ 3 pts** Correct (fifo otherwise, you have to loop through the entire frame table)

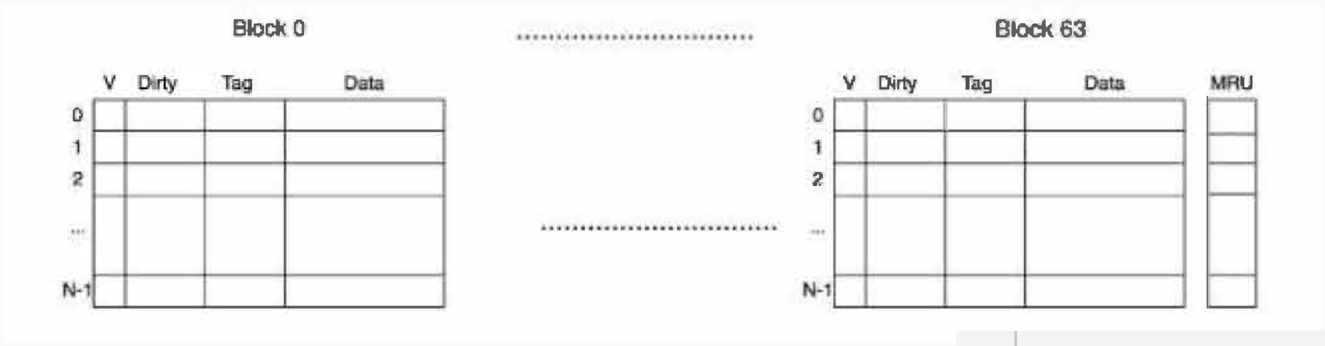
+ 1 pt Incorrect Explanation

+ 0 pts Incorrect

Q5 Cache Calculations
13 Points

Bo is designing a **32-way set-associative cache** with the following characteristics.

- Total data size of cache = **2048 KBytes** (1KBytes = 1024 bytes).
- CPU uses **32-bit byte-addressable** memory addresses.
- Each memory word consists of **4 bytes**.
- The cache block size is **512 bytes**.
- The cache has one valid bit per cache block.
- The cache uses a **write-back** policy with one dirty bit per word.
- The cache's replacement policy only guarantees that the most-recently used cache block will not be evicted on a set conflict. For that purpose, the **MRU field** records only the **most recently used** cache block in that cache set.



5.1	(no title)	**2** / 2 pts
✓	**+ 2 pts**	Correct (128, 32-way)
	+ 2 pts	Correct (64, 64-way)
	+ 0 pts	Incorrect Work
	+ 0 pts	Incorrect

NOTE: Show your work for each of the questions below for credit.

Q5.1
2 Points

What is the value of **N** in the diagram above? (Show your work to receive credit)

128

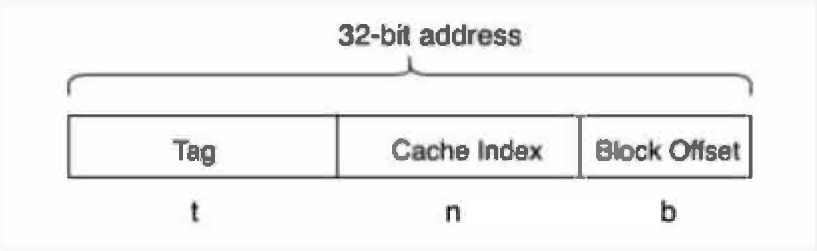
Show all work here:

NOTE: should say block 31 instead of 63 in the image above on the right.

N = S/B = 2(1048576)/(32*512) = 128

Q5.2
3 Points

The 32-bit memory address is split into block offset, tag, and index as shown below:



5.2	(no title)	**1** / 3 pts
	+ 3 pts	Correct (b = 9, n = 7, t = 16 ; 32-way)
	+ 3 pts	Correct (b = 9, n = 6, t = 17 ; 64-way)
✓	**+ 1 pt**	B correct (9)
	+ 1 pt	n correct (7)
	+ 1 pt	t correct (16)
	+ 1 pt	n correct (6)
	+ 1 pt	t correct (17)
	+ 0 pts	Incorrect Work
	+ 0 pts	Incorrect

What is the value of b?

9

What is the value of n?

5

What is the value of t?

18

Show all work here:

b = log2(512) = 9
n = log2(32) = 5
t = 32 - 9 - 5 = 18

Q5.3
2 Points

How many MRU bits are needed per cache set?

5

Show all work here:

log2(32) = 5

5.3 (no title) **2** / 2 pts

✓	**+ 2 pts**	Correct (5) ; 32-way
	+ 2 pts	Correct (6) ; 64-way
	+ 0 pts	Incorrect Work
	+ 0 pts	Incorrect

Q5.4
2 Points

How many dirty bits are needed per cache block?

128

Show all work here:

512/4 = 128

5.4 (no title) **2** / 2 pts

✓	**+ 2 pts**	Correct (128) ; both 64-way and 32-way
	+ 0 pts	Incorrect Work
	+ 0 pts	Incorrect

Q5.5
2 Points

What's the total size of the cache (in KB)? (1 KB = 1024 bytes). Round to two decimal places.

Show all work here:

5.5 (no title) **0** / 2 pts

	+ 2 pts	Correct (2120.58KB / 2171472B / 17371776bits)
	+ 2 pts	Correct (2121.05KB / 2171952B / 17375616 bits)
✓	**+ 0 pts**	Incorrect Work
	+ 0 pts	Incorrect

Q5.6
2 Points

The cache requires **x y-bit** tag comparators to operate. What are the values of x and y?

x:

128

y:

18

Show all work here:

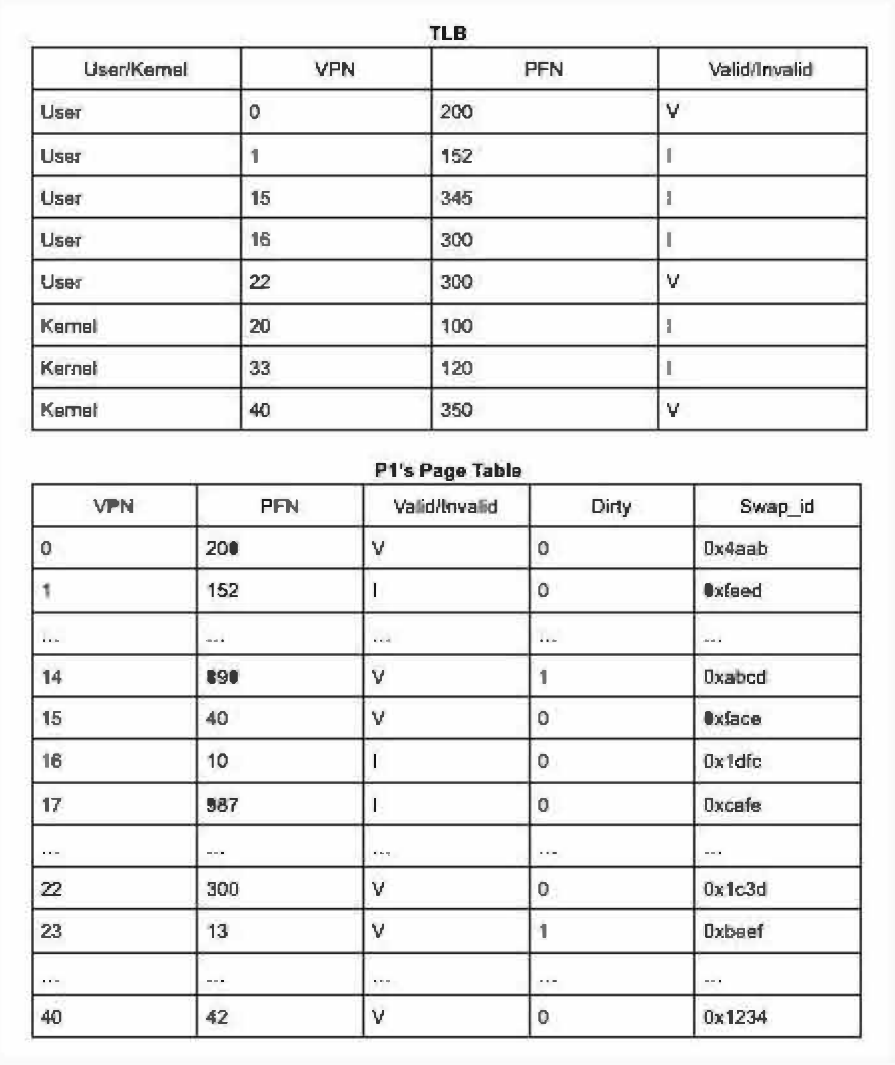
128 tags, t=18

5.6 (no title) **0** / 2 pts

	+ 2 pts	Correct (x = 32, y = 16) ; 32-way
	+ 2 pts	Correct (x = 64, y = 17) ; 64-way
	+ 1 pt	Only x correct (32)
	+ 1 pt	Only y correct (16)
	+ 1 pt	Only x correct (64)
	+ 1 pt	Only y correct (17)
✓	**+ 0 pts**	Incorrect Work
	+ 0 pts	Incorrect

Q6 TLB Hit/Miss Problem
8 Points

Suppose John is running process P1 on his processor. The following table shows the initial contents of the processor's TLB and P1's page table. Assume that the system currently has plenty of free page frames available in physical memory.



TLB

User/Kernel	VPN	PFN	Valid/Invalid
User	0	200	V
User	1	152	I
User	15	345	I
User	16	300	I
User	22	300	V
Kernel	20	100	I
Kernel	33	120	I
Kernel	40	350	V

P1's Page Table

VPN	PFN	Valid/Invalid	Dirty	Swap_id
0	200	V	0	0x4aab
1	152	I	0	0xfeed
...	...	...	...	...
14	890	V	1	0xabcd
15	40	V	0	0xface
16	10	I	0	0x1dfc
17	987	I	0	0xcafe
...	...	...	...	...
22	300	V	0	0x1c3d
23	13	V	1	0xbeef
...	...	...	...	...
40	42	V	0	0x1234

Given the state of the TLB and Page Table given above, answer the following questions:

Q6.1
2 Points

What would happen when P1 accesses VPN 22? Explain your answer.

We will see that it is mapped in the TLB and that it is valid and return the PFN of 300 and continue on

6.1 (no title) **2** / 2 pts

- ✓ **+ 2 pts** Correct (TLB hit and retrieve the memory at PFN 300)
- **+ 1 pt** Only mentions retrieving from PFN 300
- **+ 1 pt** Only mentions TLB hit
- **+ 0 pts** Incorrect

Q6.2
2 Points

What would happen when P1 accesses VPN 40? Explain your answer.

We will see that VPN 40 is mapped to a Kernel process which is not allowed meaning we have a miss and we'll have to check the page table

6.2 (no title) **0** / 2 pts

- **+ 2 pts** Correct (TLB miss at VPN 40, then check the page table for PFN 42, retrieve the data and update the TLB.)
- **+ 1 pt** Doesn't mention updating the TLB after the miss
- ✓ **+ 0 pts** Incorrect

Q6.3
2 Points

What would happen when P1 accesses VPN 17? Explain your answer.

VPN 17 is not in the TLB so we'll have to check the page table and when we check the page table we see that the invalid bit is set for VPN 17 meaning we have a page fault and need to get the data from disk. After the being brought back into the page, the valid bit will be reset and the TLB will be updated.

6.3 (no title) **2** / 2 pts

- ✓ **+ 2 pts** Correct (TLB miss and incur a page fault, and update the TLB with this new value.)
- **+ 1 pt** Doesn't mention updating the TLB after the miss
- **+ 0 pts** Incorrect

Q6.4
2 Points

What would happen when P1 accesses VPN 1? Explain your answer.

It is similar to q6.3. Although VPN 1 is mapped in the TLB its valid bit is not set so it is invalid meaning we check the page table to find the same result (invalid bit) so we have to go to the disk. So in total we need 2 trips to memory because we'll have to take a free frame and then set VPN 1's PFN to that frame. We'll also update the TLB accordingly and reset the valid bit just like 16.3

6.4 (no title) **2** / 2 pts

✓	+ 2 pts	Correct (TLB miss and incur a page fault, and update the TLB with this new value.)
	+ 1 pt	Doesn't mention updating the TLB after the miss
	+ 0 pts	Incorrect

Q7 Demand Paging

14 Points

Consider the following paging questions:

Q7.1

3 Points

Consider a memory system with **32-bit virtual addresses, 20-bit physical memory addresses** and **4096-byte pagesize**. Calculate the following.

7.1 (no title) **3** / 3 pts

✓	+ 3 pts	Correct (20, 8, 2^20, 2^8)
	+ 1 pt	VPN bits correct (20)
	+ 1 pt	PFN bits correct (8)
	+ 0.5 pts	Page table entries correct (2^20)
	+ 0.5 pts	Frame table entries correct (2^8)
	+ 0 pts	Incorrect Work
	+ 0 pts	Incorrect

Size of the **VPN** in bits. (1 point)

20

Size of the **PFN** in bits. (1 point)

8

Number of **page table** entries. Answer your question as a power of two (that is, 2^n). (0.5 points)

2^20

Number of **frame table** entries. Answer your question as a power of two (that is, 2^n). (0.5 points)

2^8

Show all your work here:

VPN = virtual addy size - offset size
offset size = log2(page size) = log2(4kb) = 12
VPN = 32-12=20 bits

PFN = phys addy size - offset size
offset size = log2(page size) = log2(4kb) = 12
VPN=20-12=8 bits

7.2 (no title) **3** / 3 pts

✓	+ 3 pts	Correct. Includes a couple of the following Pros: Spatial Locality Less memory overhead Less TLB missed Cons: Internal Fragmentation Slow context switching
	+ 2 pts	Minor Errors
	+ 1 pt	Major Errors
	+ 0 pts	Incorrect

Q7.2

3 Points

A standard page size for virtual memory is **4 kilobytes**, but this is an arbitrary choice. Some systems support page sizes of 1 megabyte or larger.

Explain the pros and cons of a larger page (e.g, 1Mb) compared to a smaller page (e.g., 4KB).

Pros: There is less overhead associated with a larger page since there are fewer pages. Less overhead improves system performance

Cons: With a larger page there is greater internal fragmentation because way more memory may be allocated than needed thus leading to allocated but unutilized memory.

Q7.3
8 Points

Answer the following demand-paging questions:

Is the page table per-process or system-wide? (1 point)

per-process

Is the frame table per-process/system-wide? (1 point)

system-wide

What is the purpose of the diskmap? (2 points)

Provides a mapping from VPN to disk space and improves system performance as we can do O(1) lookup

Is the diskmap per-process or system-wide? (1 point)

per-process

What is the purpose of the free-list? (1 point)

It is a list of all of the available memory locations. We can check the size of the free list to see if there is an available page

Is the free-list per-process or system-wide? (2 points)

system-wide

7.3 (no title) **8** / 8 pts

- ✓ **+ 1 pt** Correct page table (per-process)
- ✓ **+ 1 pt** Correct frame table (system-wide)
- ✓ **+ 2 pts** Correct diskmap (maps process's virtual pages to location on the disk)
- ✓ **+ 1 pt** Correct diskmap (per-process)
- ✓ **+ 2 pts** Correct free-list (keeps track of unoccupied frames)
- ✓ **+ 1 pt** Correct free-list (system-wide)
- **+ 0 pts** Incorrect

Q8 Hidden Question
10 Points

Answer the following:

Q8.1
4 Points

At a given point in time, Tristan notices the following virtual page accesses are recorded for four processes P1, P2, P3, and P4 respectively:

P1: 0, 0, 1, 2, 1, 2, 1, 1, 0, 3, 1
P2: 0, 100, 105, 103, 2, 103, 105
P3: 0, 1, 5, 6, 0, 0, 6, 7, 5
P4: 5, 5, 4, 3, 21, 26

What is the cumulative memory pressure on the system during this interval?

Show all work here:

8.1 (no title) **0** / 4 pts

- **+ 1 pt** Correct answer (19)
- **+ 0.75 pts** Working set for P1 correct (4)
- **+ 0.75 pts** Working set for P2 correct (5)
- **+ 0.75 pts** Working set for P3 correct (5)
- **+ 0.75 pts** Working set for P4 correct (5)
- **+ 0 pts** Incorrect work (should show the working set of all the processes and result in 19 as final answer)
- ✓ **+ 0 pts** Incorrect

Q8.2

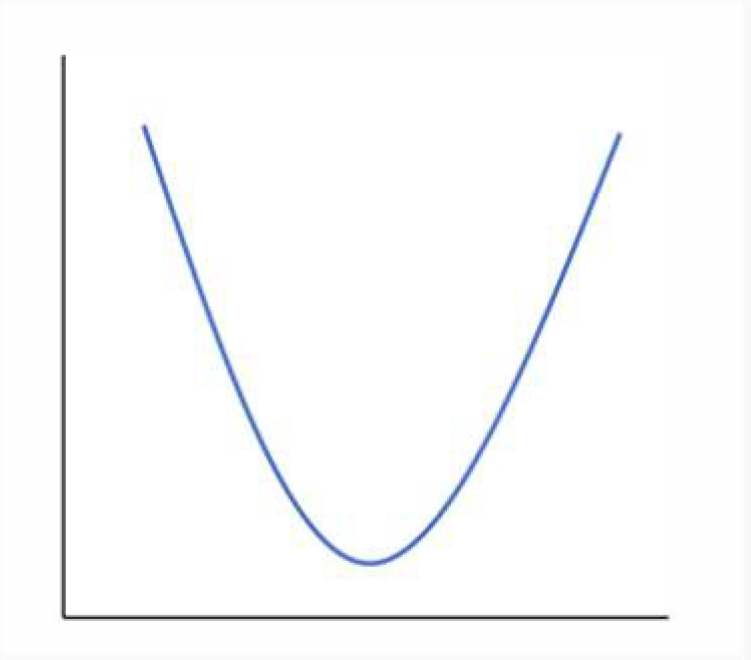
6 Points

Determine what each of the following three graphs shows, by labeling each graph's x and y axis. You are given the following axis label pair options to choose from.

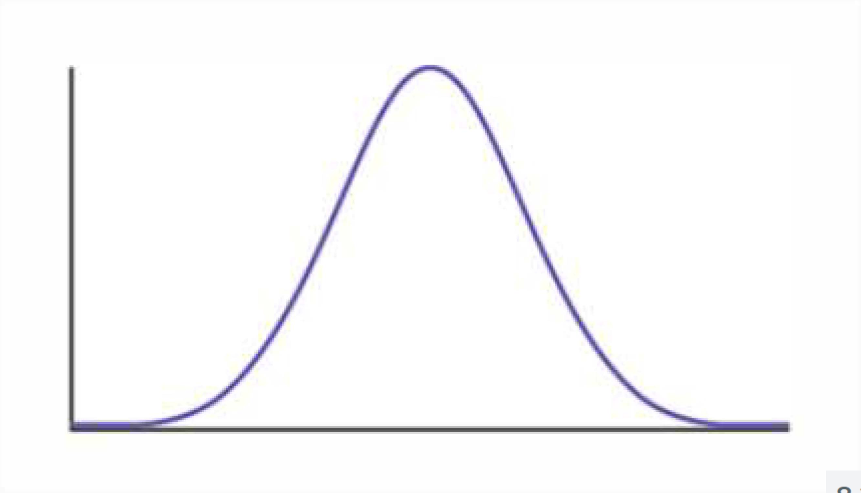
Options (notation: y-axis vs. x-axis):

1. CPU Utilization vs. Multiprogramming degree
2. Cache Hit ratio vs. Cache size
3. CPI vs. Working Set size
4. Page faults vs. Block Size
5. Cache Miss Ratio vs. Cache Associativity
6. Cache Miss Ratio vs. Cache Block Size

Select the best option for each graph. There are more options than correct answers. Write the corresponding number for the selected axis label pair under each graph.

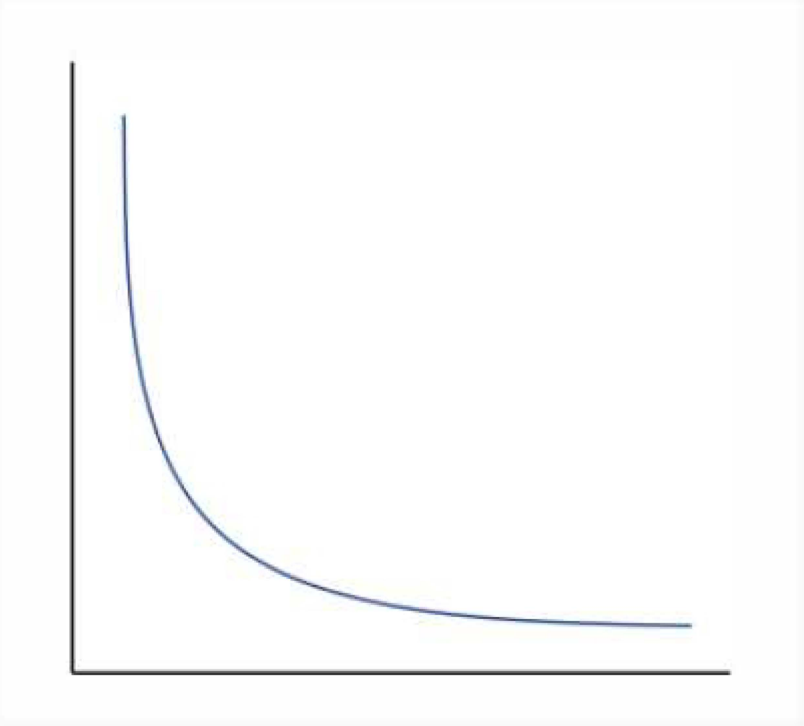


6



1

8.2	(no title)	**4** / 6 pts
	+ 6 pts	Correct
✓	+ 2 pts	Image 1 correct (6)
✓	+ 2 pts	Image 2 correct (1)
	+ 2 pts	Image 3 correct (5)
	+ 0 pts	Incorrect



3

Q9 Hidden Question

10 Points

Match the following terms with their definitions. Some definitions may be used more than once, or not at all.

Write the corresponding definition number under each term.

Definition options:
1. Holds mapping of a process' virtual page numbers to their corresponding physical frame
2. Holds metadata about every physical page in the system, including the process that owns the page and whether that page is protected
3. Metadata that marks whether it is safe to evict the page
4. Unusable memory as a result of allocating more memory for a process than it needs
5. Unusable memory as a result of being unable to partition some areas of memory for a new process
6. Hardware unit that contains recent mappings from VPNs to PFNs, and can be accessed faster than main memory
7. Hardware unit that contains frequently/recently used process data, and can be accessed faster than main memory
8. Metadata that marks whether the page is still held by the process
9. A program is more likely to access data if it has accessed that data recently
10. Occurs when several used memory addresses map to the same cache location
11. Paging algorithm commonly used in theory but not yet possible to implement.
12. Replacement algorithm that selects a page based on no additional parameters.
13. Occurs when the program has just begun execution
14. A program is more likely to access data if it has accessed adjacent data
15. Occurs when a cache runs out of space

External Fragmentation

5

Belady's Min

12

Conflict Miss

10

Page Table

1

Valid Bit

13

Random Replacement

12

Frame Table

2

Translational Lookaside Buffer

QUESTION 9

Hidden Question **8** / 10 pts

	+ 10 pts	Correct
✓	+ 1 pt	Correct External Fragmentation (5)
	+ 1 pt	Correct Belady's Min (11)
✓	+ 1 pt	Correct Conflict Miss (10)
✓	+ 1 pt	Correct Page Table (1)
	+ 1 pt	Correct Valid Bit (8)
✓	+ 1 pt	Correct Random Replacement (12)
✓	+ 1 pt	Correct Frame Table (2)
✓	+ 1 pt	Correct TLB (6)
✓	+ 1 pt	Correct Temporal Locality (9)
✓	+ 1 pt	Correct Protected Bit (3)
✓	+ 0 pts	Incorrect

6

Temporal Locality

9

Protected bit

3

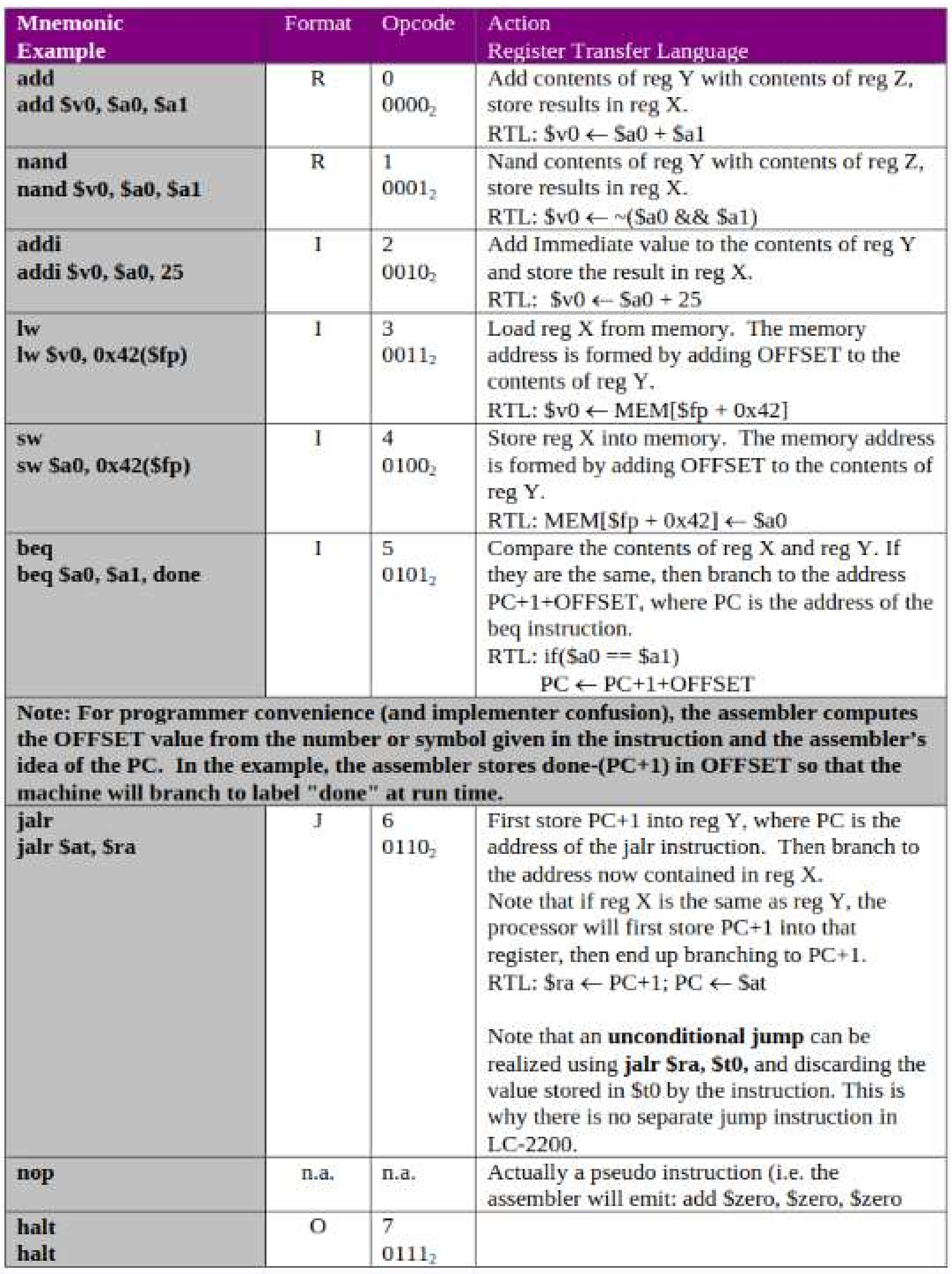
Q10 Appendix A: Useful Formulas

0 Points

Name	Notation	Units	Description
CPU Utilization	-	%	Percentage of time the CPU is busy
Throughput	n/T	Jobs/sec	System-centric metric quantifying the number of jobs *n* executed in time interval *T*
Avg. Turnaround time (t_{avg})	$(t_1+t_2+\ldots+t_n)/n$	Seconds	System-centric metric quantifying the average time it takes for a job to complete
Avg. Waiting time (w_{avg})	$((t_1-e_1) + (t_2-e_2)+ \ldots + (t_n-e_n))/n$ or $(w_1+w_2+ \ldots +w_n)/n$	Seconds	System-centric metric quantifying the average waiting time that a job experiences
Response time/ turnaround time	t_i	Seconds	User-centric metric quantifying the turnaround time for a specific job *i*
Variance in Response time	$E[(t_i – e_i)^2]$	Seconds^2	User-centric metric that quantifies the statistical variance of the actual response time (t_i) experienced by a process (P_i) from the expected value (t_{avg})

Q11 Appendix B: LC-2200 ISA

0 Points



Mnemonic Example	Format	Opcode	Action Register Transfer Language
add **add $v0, $a0, $a1**	R	0 0000_2	Add contents of reg Y with contents of reg Z, store results in reg X. RTL: $v0 ← $a0 + $a1
nand **nand $v0, $a0, $a1**	R	1 0001_2	Nand contents of reg Y with contents of reg Z, store results in reg X. RTL: $v0 ← ~($a0 && $a1)
addi **addi $v0, $a0, 25**	I	2 0010_2	Add Immediate value to the contents of reg Y and store the result in reg X. RTL: $v0 ← $a0 + 25
lw **lw $v0, 0x42($fp)**	I	3 0011_2	Load reg X from memory. The memory address is formed by adding OFFSET to the contents of reg Y. RTL: $v0 ← MEM[$fp + 0x42]
sw **sw $a0, 0x42($fp)**	I	4 0100_2	Store reg X into memory. The memory address is formed by adding OFFSET to the contents of reg Y. RTL: MEM[$fp + 0x42] ← $a0
beq **beq $a0, $a1, done**	I	5 0101_2	Compare the contents of reg X and reg Y. If they are the same, then branch to the address PC+1+OFFSET, where PC is the address of the beq instruction. RTL: if($a0 == $a1) PC ← PC+1+OFFSET
Note: For programmer convenience (and implementer confusion), the assembler computes the OFFSET value from the number or symbol given in the instruction and the assembler's idea of the PC. In the example, the assembler stores done-(PC+1) in OFFSET so that the machine will branch to label "done" at run time.			
jalr **jalr $at, $ra**	J	6 0110_2	First store PC+1 into reg Y, where PC is the address of the jalr instruction. Then branch to the address now contained in reg X. Note that if reg X is the same as reg Y, the processor will first store PC+1 into that register, then end up branching to PC+1. RTL: $ra ← PC+1; PC ← $at Note that an **unconditional jump** can be realized using **jalr $ra, $t0,** and discarding the value stored in $t0 by the instruction. This is why there is no separate jump instruction in LC-2200.
nop	n.a.	n.a.	Actually a pseudo instruction (i.e. the assembler will emit: add $zero, $zero, $zero
halt **halt**	O	7 0111_2	

Exam 3

● GRADED

STUDENT
Eric Anders Gustafson

TOTAL POINTS
77 / 100 pts

QUESTION 1
Memory Fragmentation **10** / 12 pts

1.1	(no title)	**0** / 2 pts
1.2	(no title)	**3** / 3 pts
1.3	(no title)	**3** / 3 pts
1.4	(no title)	**4** / 4 pts

QUESTION 2
Cache Performance **8** / 13 pts

2.1	(no title)	**2** / 2 pts
2.2	(no title)	**2** / 2 pts
2.3	(no title)	**2** / 3 pts
2.4	(no title)	**2** / 3 pts
2.5	(no title)	**0** / 3 pts

QUESTION 3
Cache Organization **9** / 9 pts

3.1	(no title)	**3** / 3 pts
3.2	(no title)	**3** / 3 pts
3.3	(no title)	**3** / 3 pts

QUESTION 4

Page Replacement **11** / 11 pts

4.1	(no title)	**2** / 2 pts
4.2	(no title)	**2** / 2 pts
4.3	(no title)	**2** / 2 pts
4.4	(no title)	**2** / 2 pts
4.5	(no title)	**3** / 3 pts

QUESTION 5

Cache Calculations **7** / 13 pts

5.1	(no title)	**2** / 2 pts
5.2	(no title)	**1** / 3 pts
5.3	(no title)	**2** / 2 pts
5.4	(no title)	**2** / 2 pts
5.5	(no title)	**0** / 2 pts
5.6	(no title)	**0** / 2 pts

QUESTION 6

TLB Hit/Miss Problem **6** / 8 pts

6.1	(no title)	**2** / 2 pts
6.2	(no title)	**0** / 2 pts
6.3	(no title)	**2** / 2 pts
6.4	(no title)	**2** / 2 pts

QUESTION 7

Demand Paging **14** / 14 pts

7.1	(no title)	**3** / 3 pts
7.2	(no title)	**3** / 3 pts
7.3	(no title)	**8** / 8 pts

QUESTION 8

Hidden Question **4** / 10 pts

8.1	(no title)	**0** / 4 pts
8.2	(no title)	**4** / 6 pts

QUESTION 9

Hidden Question **8** / 10 pts

	+ 10 pts	Correct
✓	**+ 1 pt**	Correct External Fragmentation (5)
	+ 1 pt	Correct Belady's Min (11)
✓	**+ 1 pt**	Correct Conflict Miss (10)
✓	**+ 1 pt**	Correct Page Table (1)
	+ 1 pt	Correct Valid Bit (8)
✓	**+ 1 pt**	Correct Random Replacement (12)
✓	**+ 1 pt**	Correct Frame Table (2)
✓	**+ 1 pt**	Correct TLB (6)
✓	**+ 1 pt**	Correct Temporal Locality (9)
✓	**+ 1 pt**	Correct Protected Bit (3)
✓	**+ 0 pts**	Incorrect

QUESTION 10

Appendix A: Useful Formulas **0** / 0 pts

QUESTION 11

Appendix B: LC-2200 ISA **0** / 0 pts