Spring 2022 Exam 3

You have 70 minutes. The exam includes three bonus points, so the total points tally up to 100. Please be concise: avoid verbose answers. **Good luck!**

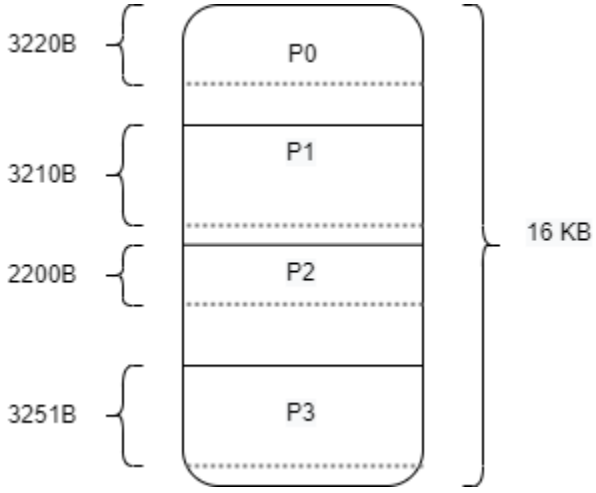
The Code educates all members of the Georgia Tech Community about the Institute's expectations and Students' rights and creates a standard by which Students are expected to conduct themselves for the purpose of establishing an environment conducive to academic excellence. Georgia Tech Students, Registered Student Organizations, and Groups are responsible for their own behavior, and the Institute has the authority to establish an internal structure for the enforcement of its policies and procedures, the terms of which students have agreed to accept by their enrollment.

Question 1: Memory Fragmentation (12 points) (8 minutes)

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:

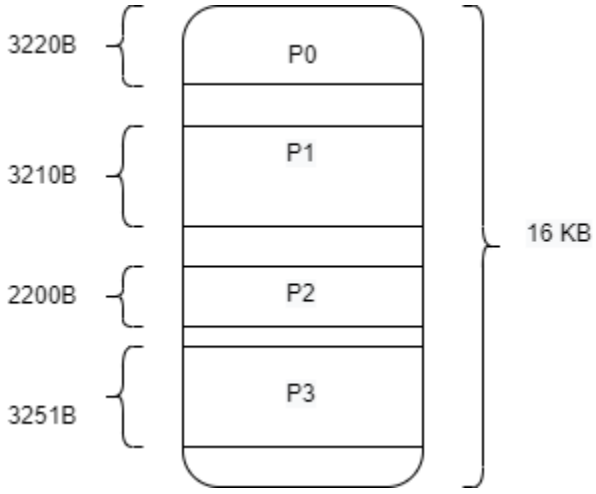
P0: 3220 bytes
P1: 3210 bytes
P2: 2200 bytes
P3: 3251 bytes

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



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

b) 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.

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

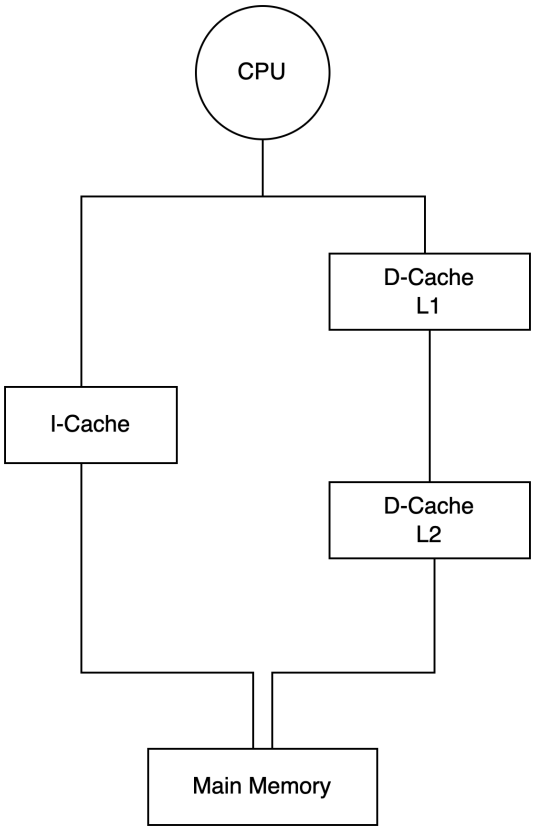
d) This question is independent from all previous assumptions introduced in questions a to c. Charles proposes hardware support for the LC-2200 processor to aid memory management that uses a pair of “bounds” registers to define the memory occupied by a process. The registers are used by the memory manager 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 a correct system behavior? Explain your answer.

Question 2: EMAT and CPI (13 points) (10 minutes)

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.
- Assume **write-through** policy is being used.

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

Question 3: Set-Associative Cache (9 points) (9 minutes)

The CS 2200 TA team is tasked with designing a 16KB cache with 64 bytes block.

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

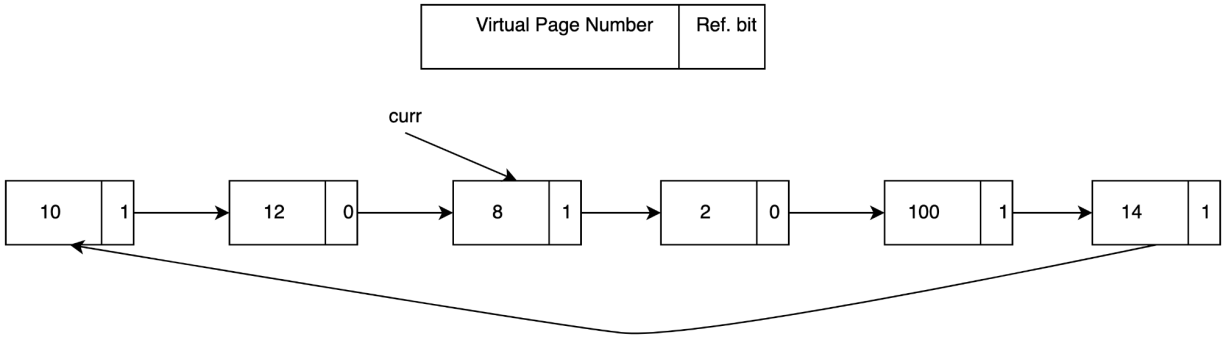
b. Borun sees that modern processors typically feature lower associativity than Malav's design and opts for a 4-way set-associative design. Why might this be an improvement on Malav's system?

c. 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?

Question 4: Page Replacement (11 points) (6 minutes)

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.

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



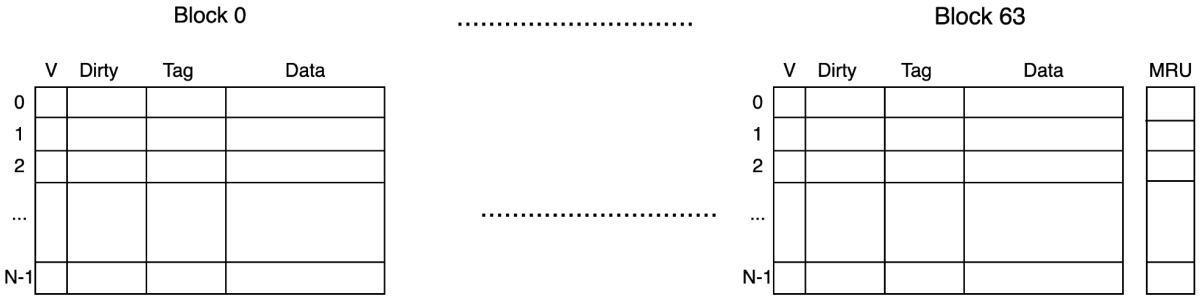
a) At time t = 1, a process accesses virtual page 4. Show the state of the queue after this reference.
b) At time t = 2, a process accesses virtual page 13. Show the state of the queue after this reference.
c) At time t=3, a process accesses virtual page 8. Show the state of the queue after this reference.
d) At time t = 4, virtual page number 25 is brought into the physical memory. Show the state of the queue after this reference.
e) 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.

Question 5: (13 points) (10 minutes)

Bo is designing a **64-way** set-associative cache with the following characteristics.

- Total data size of cache = **1024 KBytes** (1KBytes = 1024 bytes).
- CPU uses **32-bit byte-addressable** memory addresses.
- Each memory word consists of **4 bytes**.
- The cache block size is **256 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.

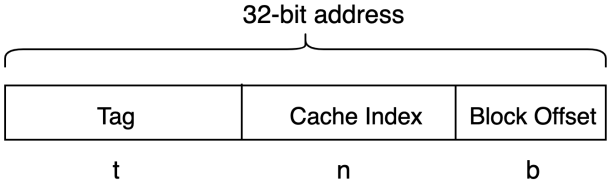
(*NOTE:* The specific numbers above may be different in the actual exam.)



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

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

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



i) What is the value of b?
ii) What is the value of n?
iii) What is the value of t?

c) How many MRU bits are needed per cache set?
d) How many dirty bits are needed per cache block?
e) What's the total size of the cache (in bytes)?
f) The cache requires **x y-bit** tag comparators to operate. What are the values of x and y?

Question 6: (8 points) (8 minutes)

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:

a. What would happen when P1 accesses VPN 22? Explain your answer.
b. What would happen when P1 accesses VPN 40? Explain your answer.
c. What would happen when P1 accesses VPN 17? Explain your answer.
d. What would happen when P1 accesses VPN 1? Explain your answer.

Question 7: Demand Paging (14 points) (8 minutes)

a. Consider a memory system with 64-bit virtual addresses, 32-bit physical memory addresses and 8KB pagesize. Calculate the following.
(*NOTE:* The specific numbers above may be different in the actual exam.)

   i. Size of VPN in bits.
   ii. Size of PFN in bits.
   iii. Number of Page Table entries. (Answer should be in the format of 2^n)
   iv. Number of Frame Table entries. (Answer should be in the format of 2^n)

b. 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).

c. Demand Paging
   a. Is the page table per-process or system-wide?
   b. Is the frame table per-process/system-wide?
   c. What is the purpose of the diskmap?
   d. Is the diskmap per-process or system-wide?
   e. What is the purpose of the free-list?
   f. Is the free-list per-process or system-wide?

Question 8: Hidden! (10 points) (5 minutes)

Question 9: Hidden! (10 points) (6 minutes)

Appendix A: Acronyms

TLB - Translation Look-aside Buffer
PTE - Page Table Entry
PFN - Physical Frame Number
VPN - Virtual Page Number
RLT - Reverse Lookup Table
PCB - Process Control Block

Appendix B: Useful Formulas

Name	Notation	Units	Comment
Memory footprint	-	Bytes	Total space occupied by the program in memory
Execution time	$(\sum CPI_j)$ * clock cycle time, where $1 \le j \le n$	Seconds	Running time of the program that executes *n* instructions
Arithmetic mean	$(E_1+E_2+\ldots+E_p)/p$	Seconds	Average of execution times of constituent *p* benchmark programs
Weighted Arithmetic mean	$(f_1*E_1+f_2*E_2+\ldots+f_p*E_p)$	Seconds	Weighted average of execution times of constituent *p* benchmark programs
Geometric mean	p^{th} root $(E_1*E_2*\ldots*E_p)$	Seconds	p^{th} root of the product of execution times of *p* programs that constitute the benchmark
Harmonic mean	$1/(\ ((1/E_1)+(1/E_2)+\ldots+(1/E_p))/p\ )$	Seconds	Arithmetic mean of the reciprocals of the execution times of the constituent *p* benchmark programs
Static instruction frequency		%	Occurrence of instruction *i* in compiled code
Dynamic instruction frequency		%	Occurrence of instruction *i* in executed code
Speedup (M_A over M_B)	E_B/E_A	Number	Speedup of Machine A over B
Speedup (improvement)	E_{Before}/E_{After}	Number	Speedup After improvement
Improvement in Exec time	$(E_{old}-E_{new})/E_{old}$	Number	New Vs. old
Amdahl's law	$Time_{after} = Time_{unaffected} + Time_{affected}/x$	Seconds	*x* is amount of improvement

Execution time = N * CPI_{Avg} * cycle time

Execution time = N * CPI_{eff} * cycle time

$CPI_{eff} = CPI_{Avg} + Memory\text{-}stalls_{Avg}$

Execution time = $N * (CPI_{Avg} + M\text{-}stalls_{Avg})$ * cycle time

$Memory\text{-}stalls_{Avg}$ = $misses\ per\ instruction_{Avg} * miss\text{-}penalty_{Avg}$

Total memory stalls = N * $Memory\text{-}stalls_{Avg}$