

1. Short answer questions

- (a) (3 points) What are three specific benefits of using virtual memory?
1. Adding disks to hierarchy.
 2. Simplifying memory for apps.
 3. Protection between processes.
- (b) (2 points) Do you think the virtual and physical page number must be the same size? what about the page size?
- The virtual and physical page number don't have to be the same size.
 - The page size must be same.
- (c) (2 points) Do different processes have (typically) access to each other's memory (Yes/ No). What is the name of the technique that is responsible for this?
- No.
 - Virtual memory, address translation and protection.
- (d) (3 points) What does TLB stand for and what does it do?
- Translation lookaside buffers.
 - Since translation is expensive, the TLB caches some translations.
- (e) (3 points) What should happen to the TLB when a new value is loaded into the page table address register?(only need to consider the valid bits of TLB)
- The valid bit of all entries in TLB should be set to 0, since a new page table is used.

2. Memory Access

Consider a 32-bit physical memory space and a 32 KiB 4-way associative cache with LRU replacement. You are told that the hit rate of loop two is 9/16.

```
1  int ARRAY_SIZE = 32 * 1024;
2  int arr[ARRAY_SIZE]; // *arr is aligned to a cache block
3
4  /* loop one */
5  for (int i = 0; i < ARRAY_SIZE; i += 4) arr[i] = i;
6  /* loop two */
7  for (int i = ARRAY_SIZE - 4; i >= 0; i -= 4) arr[i+1] = arr[i]-1;
```

- (a) (4 points) Fill the number of bits in the tag, index and offset fields in the figure below.

Tag	Index	Offset
19	9	4
19	8	5

- (b) It's Not My Fault

Consider the following OpenMP snippet:

```
1  int values[size];
2  #pragma omp parallel
3  {
4      int i = omp_get_thread_num();
5      int n = omp_get_num_threads();
6      for(int j = i * (size / n); j < (i + 1) * (size / n); j++){
7          values[j] = j;
8      }
9  }
```

All cores share the same physical memory and we are running 2 threads. This is the sole process running. Each page is 1 KiB, and you have 2 pages of physical memory. The code snippet above starts at virtual address 0x400, and the values array starts at 0x800. The size, n, i, and j variables are all stored in registers. The functions `omp_get_thread_num` and `omp_get_num_threads` are stored in virtual addresses 0x440 to 0x480. The replacement policy for the page table is Least Recently Used. At the start of the `pragma omp parallel` call the page table looks as follows:

Virtual Page Number	Valid	Dirty	Physical Page Number
0	0	0	0
1	1	0	0
2	1	1	1
3	0	0	1

- (c) (3 points) How many page faults will occur if `size=0x080`?

- 0

(d) (5 points) What is the minimum number of page faults that will occur if size=0x200? And what is the maximum?

- The minimum number is 1.
- The maximum number is 512.

(e) (3 points) How could you reduce the maximum page faults for part (b) (updated to (d))? Choose the valid options amongst the following (- 1 pt for every incorrect answer):

1. increase virtual address space
2. decrease number of threads
3. increase number of threads
4. use SIMD instructions
5. change page table replacement policy to Random
6. increase physical address space
7. add a 4-entry TLB

- 2 4 6

3. Meltdown

- (a) (4 points) Shortly explain (on a high level - we are mainly looking for the correct keywords) how meltdown works.
- The out of order execution in pipelining allows some instructions to be executed in advance. Attacker can construct a probe array and use Flush+Reload to see which cache line of the probe arrays hits, then the attacker can figure out the value in memory, even if the attacker has no privilege in accessing that memory.

DMA, Dependency and RAID

- (a) (2 points) Please briefly explain what DMA is in 2 sentences.
- DMA stands for direct memory access and is a mechanism that allows I/O devices to directly read/write main memory.
- (b) (2 points) During the DMA procedure, which of them may raise an interrupt when handling the incoming data? circle them out.
- A. CPU
 - B. ☒ I/O Device
 - C. ☒ DMA engine
 - D. ALU
- (c) (2 points) Which of the following will decrease the availability? Circle them.
- A. ☒ decrease MTTF
 - B. increase MTTF
 - C. decrease MTBF
 - D. increase MTBF
 - E. decrease MTTR
 - F. ☒ increase MTTR
- (d) (4 points) True or False Questions, please circle the correct answer.
- T / ☒ F : RAID 1 will result in slower read but it has very high availability
- ☒ T / F : RAID 1 is the most expensive architecture among RAID 1,3,4,5
- T / ☒ F : RAID 3 uses hamming ECC to check and correct the data
- ☒ T / F : RAID 4 is still slow on small writes