Name: ______________________________ Entry number: ______________________

Instructions

- Verify that your exam has pages 1 to 12. Please fill the name and entry number on *every page* front and back. Please do this first before starting the exam.
- During the exam, you should not have any electronic equipment including laptop, mobile phone, calculator, tablets, etc.
- The exam is open notes, open book, open slides. You can keep the printouts with you during the exam. The printouts should be bundled together. Please write your name and entry number on your bundle.
- Please answer each question in the provided space. Rough space is available on Pages 3 and 12.

Please sign below:

I will not give or receive aid in the examination. I accept that any act of mine that can be considered to be an *IITD Honour Code* violation will invite disciplinary action.

Signature: ______________________

Q1 [4 marks] True/False

A. [0.5 marks] x86 uses big-endian byte order

____

B. [0.5 marks] Linker takes a .s assembly program (ASCII text) and produces a .o binary file

____

C. [0.5 marks] As per gcc calling convention, %ebx contains return value of a function

____

D. [0.5 marks] On x86, LAPIC is a port IO device

____

E. [0.5 marks] LIDT instruction sets the interrupt descriptor table register (IDTR)

____

F. [0.5 marks] IRET instruction and RET instruction work in a completely identical manner

____

G. [0.5 marks] x86 hardware automatically saves the %eax register during an interrupt

____

H. [0.5 marks] The xv6 OS uses multi-segment model since it provides best protection

____

Name: ______________________________ Entry number: ______________________

Q2 [6 marks] Secure Boot

Say you are designing a security-sensitive device, like an ATM machine. Even though you have physically protected the device, you want to further protect against the situation where someone flashes their own bootloader/OS on the machine's hard disk. If we allow booting such a modified OS, it might compromise the integrity of the machine.

To protect against a modified bootloader, you modify the device's BIOS. Recall that BIOS reads the first sector of the disk, copies it to 0x7c00, and jumps %eip to it. In your modified BIOS, you first compute SHA256 of the first disk sector and match it against a known hash value. If the bootloader was modified, the hash will not match. In this case, the BIOS will not give control to the bootloader.

A. *[2 marks] Inspired from the above scheme, describe how your modified bootloader can securely boot a known OS.*

B. *[4 marks] Say with your proposed changes, the bootloader no longer fits in one disk sector (512 bytes). How will you deal with this situation? For this problem, assume that you cannot further change the BIOS behavior, i.e., BIOS will only load the first disk sector and jump to it after matching the hash. Do maintain a secure boot with your new changes.*

Name: ______________________________ Entry number: ______________________

Intentionally left blank for rough work

Name: ______________________________ Entry number: ______________________

Q3 [13 marks] File system for ML training

Say you are a systems engineer in a machine learning lab. Your team trains ML models on datasets containing millions of training samples stored on a hard disk drive. The current approach stores each training sample as a separate file in a dataset directory:

```
/dataset/
      Sample_000001.bin
      Sample_000312.bin
      …
      Sample_983909.bin
```

Your team has realized that disk IO has bottlenecked ML training. They have asked you to design a more efficient file system that can store the dataset.

You have made following observations:

1. Each sample is named `Sample_abcdef.bin` where `abcdef` is a number.
2. Each sample is of known fixed size, consuming *exactly* 512x512 bytes.
3. The training loop looks up files randomly (e.g., load `Sample_382917.bin`).
4. After lookup, the sample file is always read sequentially.
5. The training loop never seeks and modifies sample files.

You may assume that this disk is dedicated to just storing the datasets, i.e., nothing else (bootloader, OS code, model weights, python interpreter, libraries *etc.*) will be stored on it. Therefore, you may assume that your file system will only expose a root folder "/" with `Sample_*.bin` files[1]. Further, the disk can be assumed to have sectors of 512 bytes.

A. *[4 marks] Suggest and draw a better data layout than the indexed file system in xv6 that we studied in the class. Argue why your disk layout might provide better performance for the given ML training workload.*

[1] Your file system will get mounted on the `/datasets` path in the main tree using the `mount` command. You can ignore mounting *etc.* for the problem and just design your file system.

B. *[2 marks] Assume that you have a 1GB disk. At maximum, how many sample files can your file system store?*

C. *[3 marks] Now say we are also interested in writing new sample files and deleting existing sample files. Continue to assume that the file sizes are known a priori (512x512 bytes) and we never seek/append/trim existing files. We either write files end-to-end or just delete them. Please update your file system with appropriate allocation data structures. Discuss fragmentation issues.*

D. *[4 marks] Now on your file system, demonstrate the series of actions required for creating a new file and then writing to it. Demonstrate a file system inconsistency issue that can surface due to crashes. Discuss an ordering approach to perform crash consistent file creation + file write.*

Q4 [15 marks] Ordering vs write-ahead logging

Assume that you have a disk with just a single track (i.e., the arm does not need to do seeks). This track contains a total of 14 sectors, numbered 0 to 13. Further assume that the next disk sector appears under the arm in 1 time unit and that it takes 0 time units to read/write from/to the sector.

As shown below, assume that the disk head starts at sector number 0 for the following problems. From here, to write to sector 12, it will take 12 time units. Then to write to sector 0, it will take another 2 time units (disk sectors wrap around on the track).

0	1	2	3	4	5	6	7	8	9	10	11	12	13

^^

Disk head

Further, assume that we have an xv6-style indexed file system with the following disk layout:

0: Super block
1-5: Log
6: Inode block
7: Data bitmap
8: Inode bitmap
9-13: Data blocks

A. *[3 marks] Say our file system has the following contents:*

```
/
└── exam
    ├── minor.pdf
    └── major.pdf
```

The root folder has a folder `exam` with two files: `minor.pdf` and `major.pdf`. Draw the contents of relevant blocks (sectors 6-13). Assume that both `minor.pdf` and `major.pdf` require one data block each. Assume that the inode block on sector 6 can fit 8 inodes.

B. *[3 marks] Say a file system transaction changes the permissions of file* `/exam/minor.pdf` *to read-only and appends a data block to the file* `/exam/major.pdf`*. Redraw the sectors that will be written by the transaction with their new contents. For an ordering-based crash consistency design, draw ordering arrows between the written sectors.*

C. *[3 marks] Calculate the time it takes to make this transaction durable in ordering-based crash consistency.*

D. *[3 marks] Calculate the time it takes to make this transaction durable in xv6 style journaling-based crash consistency i.e., there is one log-header block that describes contents of a single transaction.*

E. *[3 marks] Say the system crashed after the transaction was made durable in xv6' journaling-based design. Calculate the time it takes for the recovery procedure to apply the above transaction and clear the transaction from the log header.*

Intentionally left blank for rough work

Name: ______________________________ Entry number: ______________________

Q5 [12 marks] Memory APIs and bugs

A. *[3 marks] A developer wrote the following C function:*

```
#define N 3
char* repeatN(char c) {
  char A[N+1];
  for(int i = 0 ; i < N; i ++) {
    A[i] = c;
  }
  A[N] = 0;
  printf("%s\n", A);
  return A;
}
```

The developer tested this function by calling `repeatN('A')` and found that it correctly prints `AAA`. Describe why this function can lead to undefined behavior and how to fix the function.

B. *[4 marks] A developer ran the following C program:*

```
int main() {
  int* x = (int*) malloc(sizeof(int));
  free(x+8);
}
```

The program crashed with the error- `free(): invalid pointer`. How did the memory allocator identify that the pointer is invalid?

Name: ______________________________ Entry number: ______________________

C. *[5 marks] A developer ran the following C program:*

```
int main() {
   int* x = (int*) malloc(sizeof(int));
   int* y = (int*) malloc(sizeof(int));
    free(x + 8);
   char* z = (char*) malloc(sizeof(char));

  *z = 'Z';
  *y = 65;
   printf("z: %c\n", *z);
}
```

The developer expected the program to print `z: Z`, but the program printed `z: A`. Carefully describe why the developer saw this outcome.

Intentionally left blank for rough work