*On my honor, I/we have not given, nor received,*
*nor witnessed any unauthorized assistance on this work.*

Name/Signature: _____

Name/Signature: _____

Name/Signature: _____

Name/Signature: _____

Name/Signature: _____

| Question: | 1 | 2 | 3 | 4 | 5 | 6 | Total |
|---|---|---|---|---|---|---|---|
| Points: | 20 | 16 | 16 | 20 | 12 | 16 | 100 |
| Score: | | | | | | | |

1. **Topic: Paging Mechanics**
   Consider a system with a 64 byte virtual address space, 16 byte pages, and 128 bytes of physical memory. *Hint: this problem is walking your through the logical process of address translation. Do the parts in order! I recommend you keep your answers for parts of virtual and physical addresses (parts c-f) in binary representation.*

   (a) (2 points) How many bits are required to encode each of the 64 *virtual* addresses? How about the 128 *physical* addresses?

   (b) (2 points) How many pages are in the virtual address space? How many bits do we need to represent this number of pages?

   (c) (4 points) Show how to divide each of the following virtual addresses into a virtual page number (VPN) and offset:

| Virtual Addr. | VPN | Offset |
|---|---|---|
| 0x07 | | |
| 0x0C | | |
| 0x15 | | |
| 0x32 | | |

   (d) (2 points) Each page table entry has the following format:

```
| 1 valid bit | 3 PFN bits |
```

   Why are there 3 PFN bits?

(e) (4 points) The page table has four entries. Indicate if an entry is valid or not and give the associated PFN for the virtual page.

| PTE | Valid? | PFN |
|---|---|---|
| 0xA | | |
| 0xF | | |
| 0x0 | | |
| 0xC | | |

(f) (6 points) Translate the following virtual addresses their corresponding physical addresses:

i. 0x7 _____

ii. 0xC _____

iii. 0x32 _____

iv. 0x24 _____

2. **Topic: More Paging** We have a system which has an 8K virtual address space, a page size of 1K, and 64K of physical memory.

   Suppose we want to execute the following instruction, which is stored at *virtual* address 4096 (4K).

   ```
   load 2048, r1
   ```

   (a) (2 points) List the virtual addresses generated when executing this instruction.

   (b) (2 points) Show how to divide the address into its virtual page number (VPN) and offset.

   (c) (2 points) How many bits are required to encode a physical address? _____

   (d) (2 points) Suppose the system is equipped with a **hardware managed** linear page table. The page table base register (PTBR) has the value 32,768 (32K, 0x8000) in it. Each page table entry is a single byte with with following format:

   ```
   | 1 valid bit | 6 PFN bits | 1 write protected bit|
   ```

   Why are there 6 PFN bits in each page table entry?

(e) (8 points) The contents of the page table are:

| |
|---|
| 0x00 |
| 0x00 |
| 0xA8 |
| 0x00 |
| 0x8C |
| 0x00 |
| 0x00 |
| 0x00 |

List ALL the **physical memory** addresses that are accessed when the instruction is executed. *Hint: remember that page table lookups also access physical memory!*

3. (16 points) **Topic: Page Table Structure**
A system has a simple, linear page table containing page table entries (PTEs). Specifically, this system has:

   - virtual address space size is 32KB
   - page size of 4KB
   - physical memory size is 64KB

   Here is a trace of the virtual addresses and the physical addresses they translate to (or don't translate to if there's an error).

| **Virtual** | **Physical** |
|---|---|
| 0x1063 | 0x2063 |
| 0x67b4 | 0x67b4 |
| 0x584a | 0xe84a |
| 0x4dfe | invalid |
| 0x388a | invalid |
| 0x1c6b | 0x2c6b |
| 0x50a9 | 0xe0a9 |
| 0x0bc6 | invalid |
| 0x2a9f | 0x9a9f |
| 0x742b | invalid |
| 0x4b5e | invalid |
| 0x5597 | 0xe597 |

   Your job is to reconstruct the page table entries (as much as possible) from this trace. If it is not possible to reconstruct the PTE, you can mark it as "unknown." The format of each PTE is a valid bit(0 or 1) followed by the PFN:

   ```
   | Valid bit | PFN |
   ```

| **Page Table Entry** | **Valid Bit** | **PFN** |
|---|---|---|
| 0 | | |
| 1 | | |
| 2 | | |
| 3 | | |
| 4 | | |
| 5 | | |
| 6 | | |
| 7 | | |

4. **Topic: TLB Mechanics**
Assume we are using a system with a 32-bit virtual address space and the system uses paging. The virtual address is a 20-bit virtual page number (VPN) and a 12-bit offset.

The TLB has the following contents in each entry: a 20-bit VPN, a 20-bit PFN, and an 8-bit PID field. This TLB only has four entries, and they look like this:

| VPN | PFN | PID |
|---|---|---|
| 0x00000 | 0x00FFF | 0x00 |
| 0x00000 | 0x00AAB | 0x01 |
| 0x00010 | 0xF000A | 0x00 |
| 0x010FF | 0x00ABC | 0x01 |

For each of the following virtual addresses, say whether we have a TLB hit or a TLB miss. If it is a hit, provide the resulting physical address (in hex).

(a) (2 points) PID 00 generates the virtual address: 0x00000000 _____

(b) (2 points) PID 01 generates the virtual address: 0x00000000 _____

(c) (2 points) PID 00 generates the virtual address: 0xFF00FFAA _____

(d) (2 points) PID 00 generates the virtual address: 0x0010FFAA _____

(e) (2 points) PID 01 generates the virtual address: 0x0010FFAA _____

(f) (2 points) PID 00 generates the virtual address: 0x000000FF _____

(g) (2 points) PID 01 generates the virtual address: 0x00000FAB _____

(h) (2 points) PID 00 generates the virtual address: 0x010FFFFF _____

(i) (2 points) PID 02 generates the virtual address: 0x00000000 _____

(j) (2 points) PID 00 generates the binary virtual address

00000001000011111111010100001111 _____

5. **Topic: Tradeoffs in page sizing**
   Label each of the following statements as either True or False AND explain your reasoning.

   (a) (4 points) A smaller page size leads to smaller page tables.

   (b) (4 points) A smaller page size leads to more TLB misses.

   (c) (4 points) A smaller page size leads to fewer page faults.

6. **Topic: Swapping and Caching basics**
   Here is a timeline of 10 virtual memory page references:

   ```
   0   1   4   0   1   3   0   1   4   1
   ```

   For each scenario below, determine whether each virtual page access in the sequence will lead to a hit (the page is found in the memory of the system) or miss (not found and must be retrieved from the disk swap space) given the cache size and replacement policy given.

   All the pages begin on disk and no pages are in memory at the start of the trace above (and thus must be brought into memory).

   (a) (4 points) Policy: FIFO, Cache size 3

   ```
   0   1   4   0   1   3   0   1   4   1
   ```

   (b) (4 points) Policy FIFO, Cache Size 5

   ```
   0   1   4   0   1   3   0   1   4   1
   ```

   (c) (4 points) Policy LRU, Cache Size 3

   ```
   0   1   4   0   1   3   0   1   4   1
   ```

   (d) (4 points) Policy: LRU, Cache Size 1000

   ```
   0   1   4   0   1   3   0   1   4   1
   ```