# CS 537: Introduction to Operating Systems
# Fall 2019: Midterm Exam #1

This exam is closed book, closed notes.

All cell phones must be turned off and put away.

No calculators may be used.

You have two hours to complete this exam.

Write all of your answers on the accu-scan form with a #2 pencil.

These exam questions must be returned at the end of the exam, but we will not grade anything in this booklet.

**You may separate the pages of this exam if it helps you.**

Unless stated (or implied) otherwise, you should make the following assumptions:

- The OS manages a single uniprocessor (single core)
- All memory is byte addressable
- The terminology lg means $\log_2$
- $2^{10}$ bytes = 1KB
- $2^{20}$ bytes = 1MB
- Page table entries require 4 bytes
- Data is allocated with optimal alignment, starting at the beginning of a page
- Assume leading zeros can be removed from numbers (e.g., 0x06 == 0x6).
- Hex numbers are represented with a preceding "0x"

**This exam has 92 questions. Each question is worth the same number of points.**

**Good luck!**

**Part 1: Virtualizing the CPU**

Designate if the statement is True (a) or False (b).

1) One of the roles of the OS is to directly expose all details of a hardware resource to user-level applications.
2) A program can be composed of multiple processes.
3) Each process has its own virtualized stack pointer.
4) Different processes accessing the same virtual memory address may see different contents.
5) Whenever a process is descheduled by the scheduler, the process is moved to the BLOCKED state.
6) System calls are handled by the OS in kernel mode.
7) A context switch is the same as a time slice.
8) A running job may relinquish the CPU and become BLOCKED when it is waiting for the user to type at the terminal.
9) With cooperative multi-tasking it is possible for a malicious user-level process to hold on to the CPU until the machine is rebooted.
10) On a timer interrupt, the OS is responsible for transitioning from user to kernel mode.
11) Multiple processes may be in the READY state at the same time.
12) A reasonable performance goal for a scheduler is to minimize job throughput (i.e., the number of jobs completed per second).
13) If all jobs have the same length (i.e., the same cpu burst length), an FCFS and a SJF scheduler give the same average response time for the workload.
14) SJF is provably optimal for preventing starvation.
15) A non-preemptive scheduler will only schedule a new job when the RUNNING job exits or becomes BLOCKED.
16) Increasing the time-slice of an RR scheduler is likely to decrease the overhead imposed by scheduling.
17) Increasing the time-slice of an RR scheduler makes the scheduler behave more like FCFS.
18) Increasing the time-slice of an RR scheduler makes the scheduler behave more like STCF.
19) An MLFQ scheduler assumes the presence of an oracle that knows the length of a job's CPU burst in advance.
20) An MLFQ scheduler can preempt running jobs.

**Part 2: Virtualizing Memory**

Designate if the statement is True (a) or False (b).

21) The fork() system call creates a child process whose execution begins at the entry point main().
22) A process can close() the stdout file descriptor.
23) The address space of a process contains all of physical memory.
24) The heap is more likely to suffer from fragmentation than the stack.
25) Local variables within a procedure are allocated on the heap.
26) With dynamic relocation, the OS determines where the address space of a process is allocated in virtual memory.
27) With dynamic relocation, the OS can move an address space after it has been placed in physical memory.
28) A Gantt chart illustrates how virtual pages are mapped to physical pages.
29) An MMU translates logical addresses generated by the OS running in kernel mode to physical addresses.
30) Threads running within the same process are likely to share the same value for the base register, but not the bounds register, in the MMU.
31) With pure segmentation (with no paging or TLBs), each segment of each process must be allocated contiguously in physical memory.
32) With pure segmentation (with no paging or TLBs), fetching and executing an instruction that performs a store from a register to memory will involve exactly two memory references.
33) The number of bits used to specify a virtual page in a virtual address can be different than the number of bits used to specify a physical page in a physical address.
34) If 11 bits are used in a virtual address to designate an offset within a page, each page must be **2 KB**.
35) If a physical address is 16 bits and each page is 1KB, then the top 10 bits exactly designate the physical page number.
36) If a virtual address is 12 bits and each page is 512 bytes, then each address space can contain up to 16 virtual pages.
37) Given a constant number of bits in a virtual address, the size of a linear page table increases with more pages.
38) Given a fixed number of pages, the size of a linear page table decreases with a smaller address space.
39) Given a 22-bit virtual address and 1KB pages, each linear page table will consume 8KB.
40) Compared to pure base+bounds, a linear page table doubles the number of memory references (assuming no TLB).
41) A linear page table must be stored contiguously in physical memory.
42) For faster translations, the OS looks up page mappings from VPN to PPN in the TLB.

43) A workload that sequentially accesses each of the elements of a large array once will often require the same VPN to PPN translation.
44) If the TLB does not support ASIDs, all page table entries (PTEs) are marked invalid on a context switch.
45) With ASIDs, the performance penalty as seen by process A for a context switch from process A, to process B, and back to process A, could decrease when B performs fewer memory operations.
46) A TLB miss must be handled by the OS.
47) With a linear page table, all virtual pages within an address space must be allocated.
48) A multi-level page table typically reduces the amount of memory needed to store page tables, compared to a linear page table.
49) Given a 2-level page table (and no TLB), exactly 3 memory accesses are needed to fetch and execute each instruction.
50) In the memory hierarchy, a backing store is usually larger and faster than the memory layer above that uses that backing store.
51) Given a fixed-size address space, increasing the size of a page generally increases the number of levels that are needed in a multi-level page table.
52) The size of a PTE is equal to the number of bits needed to select an offset within a page.
53) When the dirty bit is set in a PTE, accessing the associated page will cause a segmentation fault.
54) When the present bit is cleared in a PTE, accessing the associated page will cause a segmentation fault.
55) On a TLB miss, the process is moved to the BLOCKED state and another process may be scheduled.
56) OPT always performs better than LRU.
57) LRU with N+1 pages of memory always performs as well or better than LRU with N pages.
58) FIFO with N+1 physical pages will contain (cache) the same virtual pages as FIFO with N pages, plus the contents of one more virtual page.
59) Demand paging loads the address space of a process into physical memory when that process is started.
60) When the hand used by the clock algorithm is moving quickly, the same pages in physical memory are being repeatedly accessed.
61) The clock algorithm must be able to determine if a page has been written to in some particular time interval.
62) The goal of the clock algorithm is to approximate the behavior of a FIFO replacement algorithm.
63) Increasing the number of jobs simultaneously submitted to a system may decrease the throughput of that system.

**Part 3. CPU Job Scheduling**

If needed, assume a time-slice of 1 sec. If needed to break ties, give preference to jobs in the order listed (A, B, C). Assume a workload with the following characteristics:

| Job Name | Arrival Time (seconds) | CPU Burst Time (seconds) |
|---|---|---|
| A | 0 | 4 |
| B | 0 | 2 |
| C | 0 | 3 |

64) Given an RR scheduler, what is the average response time for the 3 jobs?
   a. 1 second
   b. 2 seconds
   c. 3.33 seconds
   d. 10 seconds
   e. None of the above

65) Given an RR scheduler, what is the average wait time for the 3 jobs?
   a. 2 seconds
   b. 3.33 seconds
   c. 4.33 seconds
   d. 6.67 seconds
   e. None of the above

66) Given an RR scheduler, what is the turnaround time for job B?
   a. 1 second
   b. 2 seconds
   c. 4 seconds
   d. 5 seconds
   e. None of the above

67) Given a FIFO scheduler, what is the average turnaround time for the 3 jobs?
   a. 6.33 seconds
   b. 6.67 seconds
   c. 8.67 seconds
   d. 9 seconds
   e. None of the above

68) Given a SJF scheduler, what is the average turnaround time for the 3 jobs?
   a. 3.33 seconds
   b. 5.33 seconds
   c. 6.33 seconds
   d. 6.67 seconds
   e. None of the above

Assume a new workload with the following characteristics:

| Job Name | Arrival Time (seconds) | CPU Burst Time (seconds) |
|---|---|---|
| A | 0 | 7 |
| B | 4 | 4 |
| C | 6 | 2 |

69) Given a SJF scheduler, what is the average turnaround time for the 3 jobs?
   a. 6.33s
   b. 6.67s
   c. 7.33s
   d. 9.67s
   e. None of the above

70) Given a STCF scheduler, what is the average turnaround time for the 3 jobs?
   a. 6.33s
   b. 6.67s
   c. 7.33s
   d. 9.67s
   e. None of the above

**Part 4. Reverse Engineering for Dynamic Relocation**

Assume you have an architecture with a 64KB address spaces and 128KB of physical memory. Assume you are performing dynamic relocation with a base-and-bounds register. You collect a trace of virtual address to physical address translations and see the following results:

```
VA  0: 0x6baa (decimal: 27562) --> VALID: 0x1efbc (decimal: 126908)
VA  1: 0x4248 (decimal: 16968) --> VALID: 0x1c65a (decimal: 116314)
VA  2: 0x82e2 (decimal: 33506) --> SEGMENTATION VIOLATION
VA  3: 0x67a9 (decimal: 26537) --> VALID: 0x1ebbb (decimal: 125883)
VA  4: 0xc8a7 (decimal: 51367) --> SEGMENTATION VIOLATION
VA  5: 0x2568 (decimal: 9576 ) --> VALID: 0x2568 (decimal: 9576 )
```

What can you infer about state of the memory system? Assume only a single user process or the OS is running throughout the trace.

71) The base register contains
   a. 0x2434 (decimal 9268)
   b. 0x18412 (decimal 99346)
   c. 0x25b66 (decimal 154470)
   d. Contents cannot be determined
   e. None of the above

72) The OS was running when which virtual address was traced?
   a. VA 0
   b. VA 1
   c. VA 3
   d. VA 5
   e. None of the above or multiple of the above

73) Assuming the user process is running, virtual address 0x4826 (decimal: 18470) translates to what physical address?
   a. 0x84a6 (decimal: 33958)
   b. 0xcc38 (decimal: 117816)
   c. Answer cannot be determined from information provided
   d. Segmentation Fault
   e. None of the above

74) Assuming the user process is running, virtual address 0x9558 (decimal: 38232) translates to what physical address?
   a. 0x2196a (decimal: 137578)
   b. 0x1f220 (decimal: 127520)
   c. Answer cannot be determined from information provided
   d. Segmentation Fault
   e. None of the above

75) Assuming the user process is running, virtual address 0x739a (decimal: 29594) translates to what physical address?
   a. 0x1f7ac (decimal: 128940)
   b. 0x1fce2 (decimal: 130274)
   c. Answer cannot be determined from information provided
   d. Segmentation Fault
   e. None of the above

**Part 5. Linear Page Tables**

What is the size of a linear page table for one process given the following assumptions? Remember, unless otherwise specified, assume PTEs are 4 bytes each.

76) Bits in virtual address: 32, Page Size: 4KB
   a. 1MB
   b. 4MB
   c. Situation not possible
   d. Not enough information to determine page table size
   e. None of the above

77) Bits in virtual address: 32, Page Size: 4KB, PTE size: 8 bytes
   a. 2MB
   b. 8MB
   c. Situation not possible
   d. Not enough information to determine page table size
   e. None of the above

78) Bits in virtual address: 20, Page Size: 512 bytes
   a. 8KB
   b. 16KB
   c. Situation not possible
   d. Not enough information to determine page table size
   e. None of the above

79) Bits in virtual address: 10, Page Size: 4KB
   a. 4KB
   b. 4MB
   c. Situation not possible
   d. Not enough information to determine page table size
   e. None of the above

80) Bits in virtual address: 12, Number of Virtual Pages: 128, Page Size: 1KB
   a. 64KB
   b. 512KB
   c. Situation not possible
   d. Not enough information to determine page table size
   e. None of the above

81) Number of Virtual Pages: 4096, PTE size: 16 bytes
   a. 64KB
   b. 128KB
   c. Situation not possible
   d. Not enough information to determine page table size
   e. None of the above

82) Bits in virtual address: 24, Number of Physical Pages: 1024
   a. 4KB
   b. 16KB
   c. Situation not possible
   d. Not enough information to determine page table size
   e. None of the above

## Part 6. Multi-level Page Tables

Same setup as homework. Assume dynamic relocation is performed with a **two-level page table** with no TLB. Assume the page size is 32 bytes, the virtual address space for the current process is 1024 pages, or 32 KB, and physical memory consists of 128 pages. A virtual address needs 15 bits (5 for the offset, 10 for the VPN) and a physical address 12 bits (5 offset, 7 for the PFN). The upper five bits of a virtual address index into a page directory; the page directory entry (PDE), if valid, points to a page of the page table. Each page table holds 32 page-table entries (PTEs). Each PTE, if valid, holds the desired translation (physical frame number, or PFN) for the virtual page. The format of an 8-bit PTE is VALID | PFN6 ... PFN0. Contents of memory are as follows:

```
hex offset   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
dec offset   0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

0x 0 (  0): 11 0d 16 1b 07 13 04 0d 0a 19 1c 1a 10 16 14 10 04 08 1d 12 03 0b 04 03 0b 09 00 0f 1c 0e 0c 0b
0x 1 (  1): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x 2 (  2): 1b 0c 1e 19 12 0d 11 1a 0a 04 1b 13 0b 0c 03 11 18 0f 15 16 0a 13 1e 16 05 14 13 18 15 1c 1e 06
0x 3 (  3): 05 03 14 0a 1b 07 1d 09 12 1c 15 1c 15 01 1b 12 09 05 1a 12 1d 17 1d 11 15 12 08 1d 09 18 1d 14
0x 4 (  4): 0e 0b 0b 17 00 08 1d 0e 00 13 14 1c 1b 1d 08 09 15 0a 0b 01 06 04 0b 15 06 08 02 0c 12 12 14 10
0x 5 (  5): 03 0d 13 0d 05 0f 06 0f 16 09 0d 13 10 18 1b 1d 05 0e 03 15 0c 1a 0e 18 00 13 0c 19 03 02 15 0a
0x 6 (  6): 06 06 10 01 02 02 16 1e 17 0f 04 18 04 02 0f 1a 09 1a 0a 1d 16 1a 1e 19 06 0f 10 16 07 11 08 05
0x 7 (  7): 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f d2 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
0x 8 (  8): 7f 7f d8 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
0x 9 (  9): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x a ( 10): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x b ( 11): 0e 0f 15 1d 18 10 15 10 1e 12 12 0c 0c 17 0b 1c 16 1d 15 11 0b 00 14 17 16 17 01 15 08 0c 1a 0c
0x c ( 12): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x d ( 13): 14 06 16 0c 19 11 1b 0d 0c 1c 15 0d 1b 16 13 00 1b 03 00 0a 06 07 0f 0f 15 1a 15 1a 0d 04 16 1e
0x e ( 14): 7f 7f 7f 7f 7f 7f 7f 8d 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f e8 7f d9 7f 7f 7f 7f 7f 7f 7f ca 7f 7f
0x f ( 15): 14 05 1e 0e 07 13 01 14 1a 03 07 08 06 13 19 0e 15 1c 19 0b 1b 05 19 02 09 03 11 03 12 05 14 0f
0x10 ( 16): 14 08 00 0d 19 04 1d 0f 04 03 0b 0e 05 02 0f 07 0d 0d 15 11 07 05 1a 01 03 02 19 10 1c 13 0a 0d
0x11 ( 17): 00 01 06 18 18 18 15 1e 07 15 0d 1d 0d 16 18 1d 17 0a 14 18 1d 07 16 16 04 03 11 12 0a 13 04 11
0x12 ( 18): 17 0b 10 08 01 1e 14 15 13 01 19 1a 07 1a 1d 05 1c 01 17 1e 04 0d 00 10 01 0f 05 05 14 07 04 08
0x13 ( 19): 07 1b 0e 0d 17 0a 07 03 09 07 16 0b 15 08 02 09 0c 1c 06 06 04 07 1d 1c 07 01 0f 12 03 0c 17 00
0x14 ( 20): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x15 ( 21): 19 06 16 0e 0f 02 1b 00 1a 00 0b 0d 12 08 1e 0a 06 16 14 12 1d 1b 09 1b 1b 01 0f 1a 0c 0a 0f 1a
0x16 ( 22): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x17 ( 23): 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f c6 7f 7f 7f 7f 7f 7f bb 7f 7f 7f 7f 7f 7f 7f 7f 7f 86 7f
0x18 ( 24): 7f 7f 7f 7f fc 7f 7f 7f 7f 7f f4 7f 7f 7f 7f 7f 7f fd 7f 7f ba 7f 7f 7f f9 7f 7f 7f 7f 7f 7f 7f
0x19 ( 25): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1a ( 26): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1b ( 27): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1c ( 28): 09 0d 09 0b 1a 00 1c 1c 13 17 16 0d 05 03 12 15 16 1e 09 12 08 1a 02 12 1a 07 1e 0c 1e 09 1c 0b
0x1d ( 29): 0c 01 0a 11 04 1c 1c 0f 1c 15 10 04 1d 1c 0e 0b 02 13 1a 0c 08 13 18 05 08 13 01 19 09 19 0e 05
0x1e ( 30): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x1f ( 31): 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f b1 7f 7f
0x20 ( 32): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x21 ( 33): 18 13 09 14 16 0b 02 0e 06 05 0c 02 1b 08 1b 16 00 1b 19 10 1b 0e 00 04 09 1a 19 03 0a 03 07 0a
0x22 ( 34): 7f 7f b4 7f b7 7f 7f 7f eb 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f d7
0x23 ( 35): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x24 ( 36): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x25 ( 37): 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f cf 7f
0x26 ( 38): 7f 7f 7f 7f b9 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 91 7f 7f 7f 7f 7f 7f
0x27 ( 39): 7f 7f 92 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f ae 7f 7f 7f 7f 7f ef 7f 7f 7f 7f 7f 7f 7f 82 7f
0x28 ( 40): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x29 ( 41): 1c 0b 1b 0f 13 0e 0c 1b 12 05 09 0c 11 0b 08 09 1e 16 1a 1b 04 10 16 0f 10 13 13 05 01 08 01 09
0x2a ( 42): 7f 7f 7f ea 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f ac 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
0x2b ( 43): 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 9c 7f 7f 7f 7f 7f 7f 7f 7f
0x2c ( 44): 0b 05 01 10 18 14 06 15 01 0c 19 1c 0e 1a 0f 10 12 1e 17 0e 16 13 0a 18 19 12 1d 00 0f 13 0a 05
0x2d ( 45): 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f e5 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 8b 7f 7f
0x2e ( 46): 1c 13 10 0d 0e 00 11 1d 03 01 13 19 0d 02 13 14 07 19 10 18 07 0b 14 14 1a 1d 16 00 0e 14 18 09
0x2f ( 47): 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 90 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
0x30 ( 48): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x31 ( 49): 17 02 09 12 1a 1a 0a 08 0b 03 08 17 1e 08 02 16 1d 18 0c 17 17 17 0f 1e 18 07 0d 02 0b 19 0f 10
0x32 ( 50): 11 05 0c 1c 11 19 08 1a 18 06 10 15 11 12 07 11 0a 18 0b 1e 11 07 0c 03 16 01 12 09 18 03 15 16
0x33 ( 51): 1e 13 19 1a 1a 0d 08 0f 1b 1b 0e 1c 17 01 13 18 01 0b 14 0f 10 02 08 1a 0a 0e 12 08 0c 0f 1a 1a
0x34 ( 52): 11 15 01 0c 0c 19 18 0d 0c 18 16 01 19 1a 08 14 0e 08 01 10 16 0c 13 05 19 16 0f 03 00 0c 07 01
0x35 ( 53): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x36 ( 54): 7f 7f 7f 7f 7f 7f d3 7f 7f 7f 7f 7f 7f 7f 7f 7f a9 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f 7f
0x37 ( 55): 01 1d 0a 03 06 07 1a 05 08 03 0e 06 0a 16 1c 19 01 0b 1a 01 06 0f 0b 17 16 0e 07 07 08 02 14 04
0x38 ( 56): 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x39 ( 57): 0a 04 1b 03 02 09 04 07 15 0a 1a 0b 1a 19 08 11 14 05 01 1b 00 05 04 1a 14 0f 0f 1d 06 19 09 08
0x3a ( 58): 05 05 03 0f 1a 1c 17 02 13 07 03 03 17 02 1a 0a 0b 1c 05 0d 1a 13 07 03 19 13 1d 05 08 15 09 1e
0x3b ( 59): 06 0d 1a 01 1d 06 0f 09 01 13 05 0c 19 0d 0a 04 00 19 0f 09 17 10 07 1d 0c 0a 1e 0c 0e 19 14 07
```

| hex offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1a | 1b | 1c | 1d | 1e | 1f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| dec offset | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 0x3c ( 60): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x3d ( 61): | 13 | 0c | 02 | 0d | 00 | 05 | 0e | 0d | 13 | 12 | 17 | 11 | 18 | 01 | 0e | 1d | 17 | 19 | 1e | 04 | 0f | 1b | 09 | 10 | 05 | 02 | 1c | 16 | 18 | 1b | 11 | 1a |
| 0x3e ( 62): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x3f ( 63): | 7f | 7f | 7f | 7f | 7f | 8f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | ee | 7f | 7f | 7f | 7f | e4 | 7f | 7f | 7f | 7f | 7f |
| 0x40 ( 64): | 10 | 13 | 14 | 14 | 1b | 1e | 1e | 03 | 16 | 0c | 08 | 03 | 1c | 1b | 10 | 0f | 06 | 1c | 17 | 05 | 11 | 1e | 08 | 0b | 0b | 0b | 03 | 13 | 07 | 15 | 0e | 1a |
| 0x41 ( 65): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | c4 | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f |
| 0x42 ( 66): | 1a | 14 | 0c | 01 | 0a | 0f | 0b | 0d | 07 | 06 | 1c | 18 | 0b | 19 | 02 | 13 | 08 | 08 | 0f | 11 | 09 | 07 | 0c | 19 | 1c | 1b | 16 | 1d | 01 | 1b | 0b | 01 |
| 0x43 ( 67): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x44 ( 68): | 0f | 0b | 0f | 15 | 0f | 17 | 0b | 13 | 0c | 11 | 17 | 03 | 01 | 06 | 07 | 04 | 11 | 0e | 0e | 17 | 03 | 1c | 18 | 07 | 17 | 1a | 00 | 0a | 16 | 0a | 05 | 08 |
| 0x45 ( 69): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x46 ( 70): | 0a | 07 | 0b | 0c | 08 | 09 | 07 | 09 | 14 | 13 | 13 | 1c | 14 | 04 | 0c | 10 | 17 | 10 | 02 | 0e | 14 | 18 | 0d | 06 | 01 | 0f | 17 | 03 | 10 | 12 | 11 | 07 |
| 0x47 ( 71): | 0d | 1d | 17 | 02 | 02 | 08 | 19 | 18 | 19 | 1a | 16 | 11 | 1a | 19 | 19 | 1b | 10 | 02 | 13 | 15 | 1d | 17 | 1b | 1c | 0a | 1c | 11 | 0d | 0d | 0f | 0c | 16 |
| 0x48 ( 72): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x49 ( 73): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x4a ( 74): | 05 | 10 | 07 | 1b | 00 | 11 | 1b | 13 | 04 | 10 | 19 | 13 | 1c | 19 | 0b | 03 | 15 | 0e | 19 | 0d | 16 | 04 | 17 | 1a | 19 | 18 | 14 | 05 | 0a | 08 | 09 | 12 |
| 0x4b ( 75): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x4c ( 76): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | f1 | 7f | 7f | 7f | 7f | 7f |
| 0x4d ( 77): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x4e ( 78): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 80 | 7f | 7f | 7f | 7f | 7f | 95 | 7f | 7f | 7f | 7f | 7f |
| 0x4f ( 79): | 00 | 0d | 1e | 1e | 15 | 06 | 06 | 04 | 0d | 1d | 06 | 19 | 11 | 08 | 0e | 01 | 13 | 1c | 0a | 1e | 08 | 0a | 15 | 06 | 01 | 0f | 0f | 05 | 05 | 03 | 05 | 04 |
| 0x50 ( 80): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x51 ( 81): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x52 ( 82): | 0f | 12 | 01 | 0f | 00 | 17 | 00 | 13 | 1e | 1a | 07 | 10 | 04 | 0a | 1a | 19 | 00 | 0b | 13 | 05 | 18 | 1b | 11 | 0b | 1a | 0f | 05 | 08 | 04 | 1a | 0f | 15 |
| 0x53 ( 83): | 1e | 16 | 11 | 0b | 14 | 1c | 00 | 0b | 08 | 18 | 14 | 00 | 1b | 0c | 04 | 03 | 02 | 07 | 02 | 15 | 04 | 0d | 14 | 0d | 09 | 0e | 0e | 11 | 12 | 09 | 15 | 1a |
| 0x54 ( 84): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 93 | 7f | 7f | 7f | df | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | b3 | 7f | 7f |
| 0x55 ( 85): | 19 | 01 | 1d | 1b | 0c | 06 | 16 | 11 | 03 | 12 | 0f | 12 | 00 | 10 | 0d | 1a | 16 | 16 | 11 | 07 | 19 | 08 | 1d | 10 | 0c | 0c | 0e | 0f | 04 | 0c | 01 | 11 |
| 0x56 ( 86): | 7f | 7f | 7f | 7f | bd | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f |
| 0x57 ( 87): | 18 | 0f | 0e | 10 | 04 | 0f | 07 | 09 | 01 | 04 | 01 | 10 | 07 | 11 | 0f | 07 | 03 | 12 | 14 | 15 | 0f | 11 | 00 | 09 | 18 | 0b | 03 | 0d | 19 | 03 | 03 | 0f |
| 0x58 ( 88): | 02 | 11 | 14 | 08 | 1e | 03 | 19 | 10 | 12 | 0d | 01 | 0d | 15 | 17 | 19 | 09 | 0f | 03 | 0d | 0d | 18 | 02 | 01 | 13 | 18 | 02 | 15 | 04 | 1a | 13 | 06 | 0b |
| 0x59 ( 89): | 09 | 1a | 0e | 16 | 01 | 18 | 06 | 0d | 0f | 0a | 07 | 0d | 01 | 04 | 04 | 1a | 09 | 1e | 1b | 05 | 1a | 1e | 0d | 0e | 14 | 0f | 08 | 16 | 06 | 0b | 00 | 0a |
| 0x5a ( 90): | 1d | 14 | 08 | 14 | 16 | 04 | 18 | 12 | 12 | 14 | 13 | 04 | 04 | 00 | 09 | 06 | 12 | 03 | 04 | 10 | 18 | 0f | 01 | 1c | 11 | 1c | 17 | 1c | 19 | 17 | 1a | 14 |
| 0x5b ( 91): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x5c ( 92): | 0f | 0a | 1e | 0a | 10 | 1b | 1e | 0d | 0b | 13 | 10 | 10 | 11 | 05 | 05 | 11 | 07 | 0e | 01 | 0b | 05 | 0f | 1c | 05 | 06 | 0a | 08 | 02 | 02 | 15 | 1a | 00 |
| 0x5d ( 93): | 7f | 7f | 7f | f7 | 7f | 7f | 7f | 7f | 7f | 7f | 7f | c2 | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | f8 | 7f | 7f | 7f | 7f | b2 | 7f | 7f | 7f | 7f | 7f | 7f |
| 0x5e ( 94): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x5f ( 95): | 08 | 16 | 0d | 1c | 06 | 07 | 09 | 02 | 03 | 1a | 10 | 18 | 17 | 13 | 0c | 1b | 09 | 1b | 02 | 12 | 1d | 10 | 0b | 0e | 0d | 05 | 0d | 06 | 0f | 10 | 11 | 15 |
| 0x60 ( 96): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | da | 7f | 7f | 7f | 7f | 7f | 7f | a1 | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f |
| 0x61 ( 97): | 12 | 17 | 03 | 06 | 1d | 16 | 0d | 1e | 01 | 12 | 04 | 09 | 06 | 0c | 01 | 05 | 15 | 05 | 0f | 1e | 18 | 03 | 16 | 12 | 10 | 18 | 1d | 0b | 19 | 06 | 11 | 1b |
| 0x62 ( 98): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x63 ( 99): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x64 (100): | 19 | 18 | 15 | 1c | 10 | 1a | 08 | 04 | 07 | 0c | 18 | 17 | 09 | 17 | 1b | 0c | 15 | 09 | 0c | 06 | 09 | 02 | 0c | 18 | 05 | 06 | 1c | 0a | 07 | 05 | 01 | 1b |
| 0x65 (101): | 17 | 17 | 02 | 09 | 1a | 10 | 18 | 07 | 1a | 1b | 05 | 16 | 1e | 01 | 07 | 1c | 15 | 0b | 05 | 1b | 01 | 1e | 01 | 0a | 0f | 12 | 19 | 0c | 1d | 0a | 0b | 1e |
| 0x66 (102): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | fb | 7f | 7f | 7f | 7f | 7f | 9d | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | ed | 7f | 7f | 7f | 7f |
| 0x67 (103): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x68 (104): | 0e | 18 | 0f | 02 | 0a | 0b | 1c | 12 | 0e | 13 | 01 | 15 | 19 | 14 | 0b | 08 | 00 | 02 | 04 | 0e | 18 | 06 | 0e | 13 | 13 | 00 | 1c | 12 | 09 | 07 | 15 | 1c |
| 0x69 (105): | 14 | 05 | 01 | 10 | 01 | 1c | 0f | 10 | 0c | 14 | 1c | 09 | 0b | 12 | 09 | 12 | 10 | 0d | 14 | 10 | 02 | 1e | 0b | 0d | 0c | 1c | 1d | 0f | 0c | 02 | 1b | 05 |
| 0x6a (106): | 16 | 02 | 11 | 03 | 10 | 15 | 15 | 00 | 10 | 05 | 0f | 0f | 0f | 17 | 09 | 0e | 07 | 06 | 13 | 04 | 18 | 0b | 03 | 18 | 00 | 09 | 14 | 00 | 15 | 11 | 1e | 09 |
| 0x6b (107): | 00 | 1c | 0e | 1e | 17 | 0e | 01 | 03 | 18 | 00 | 01 | 15 | 12 | 14 | 02 | 1d | 17 | 00 | 11 | 0a | 07 | 13 | 1e | 00 | 18 | 1c | 02 | 1b | 00 | 09 | 00 | 05 |
| 0x6c (108): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x6d (109): | 04 | 18 | 18 | 09 | 00 | 03 | 17 | 1e | 02 | 07 | 0c | 02 | 13 | 1a | 1a | 05 | 1c | 18 | 04 | 1a | 1c | 1b | 17 | 19 | 10 | 00 | 15 | 0f | 04 | 04 | 1b | 1c |
| 0x6e (110): | 14 | 02 | 09 | 1c | 1e | 1d | 1b | 1c | 1b | 0e | 07 | 10 | 01 | 11 | 08 | 06 | 0d | 14 | 0e | 03 | 1e | 06 | 11 | 19 | 15 | 0d | 1b | 02 | 10 | 03 | 0f | 09 |
| 0x6f (111): | 0d | 0b | 0a | 1c | 15 | 0a | 0c | 0b | 03 | 1d | 1a | 1d | 10 | 01 | 02 | 06 | 0e | 09 | 1d | 1b | 0d | 09 | 0b | 0d | 0d | 03 | 19 | 16 | 1e | 09 | 0d | 18 |
| 0x70 (112): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | dc | 7f | e1 | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | d5 | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | f5 |
| 0x71 (113): | 07 | 11 | 1d | 0b | 16 | 00 | 17 | 1a | 10 | 1b | 12 | 18 | 0f | 0b | 08 | 11 | 17 | 01 | 14 | 01 | 13 | 16 | 1c | 14 | 14 | 15 | 14 | 15 | 15 | 1a | 0b | 1d |
| 0x72 (114): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 84 | 7f | c7 | 7f | 7f | 7f | 7f | 7f | c0 | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f |
| 0x73 (115): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x74 (116): | 11 | 17 | 17 | 1c | 19 | 13 | 16 | 12 | 07 | 00 | 1c | 03 | 0d | 05 | 07 | 05 | 10 | 03 | 1a | 12 | 04 | 06 | 16 | 1d | 0c | 00 | 07 | 11 | 0b | 16 | 01 | 1e |
| 0x75 (117): | 16 | 13 | 1a | 19 | 19 | 17 | 0c | 17 | 0c | 11 | 00 | 04 | 14 | 02 | 03 | 1b | 1b | 1e | 18 | 09 | 12 | 10 | 02 | 04 | 13 | 0d | 11 | 1c | 02 | 04 | 1e | 03 |
| 0x76 (118): | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 0x77 (119): | 10 | 15 | 0d | 09 | 05 | 11 | 15 | 10 | 0e | 12 | 06 | 1c | 1a | 05 | 1a | 15 | 01 | 17 | 19 | 09 | 06 | 0e | 01 | 04 | 15 | 0b | 0c | 1c | 18 | 09 | 01 | 01 |
| 0x78 (120): | 1b | 18 | 14 | 09 | 08 | 07 | 01 | 1c | 09 | 00 | 09 | 03 | 0d | 0a | 05 | 0e | 0c | 0a | 19 | 09 | 10 | 1b | 1d | 0b | 01 | 1d | 0f | 06 | 03 | 07 | 09 | 04 |
| 0x79 (121): | 01 | 0d | 03 | 1d | 1d | 0d | 1a | 07 | 1a | 05 | 09 | 10 | 17 | 0e | 03 | 01 | 04 | 08 | 14 | 05 | 1c | 14 | 09 | 11 | 1a | 07 | 14 | 13 | 19 | 14 | 01 | 1a |
| 0x7a (122): | e0 | 8e | a7 | ab | ff | f2 | bf | af | b6 | f0 | dd | ce | 88 | d4 | 97 | fe | 7f | aa | 7f | 7f | cc | c1 | a2 | e6 | ad | a6 | 98 | 9f | 87 | 7f | a5 | d6 |
| 0x7b (123): | 1b | 13 | 1d | 07 | 16 | 19 | 04 | 1c | 1e | 06 | 05 | 03 | 05 | 0a | 18 | 06 | 07 | 15 | 19 | 00 | 18 | 15 | 09 | 0e | 01 | 0d | 00 | 17 | 10 | 05 | 06 | 12 |
| 0x7c (124): | 0c | 1b | 17 | 0b | 14 | 06 | 05 | 01 | 0b | 07 | 10 | 1e | 04 | 11 | 1a | 12 | 10 | 17 | 00 | 1d | 15 | 19 | 19 | 03 | 02 | 05 | 0f | 1d | 1c | 0c | 13 | 16 |
| 0x7d (125): | 0f | 02 | 0a | 19 | 00 | 13 | 10 | 16 | 0b | 1d | 03 | 0c | 09 | 11 | 06 | 0b | 0d | 1c | 00 | 08 | 04 | 16 | 17 | 00 | 18 | 1b | 0e | 12 | 18 | 1c | 06 | 15 |
| 0x7e (126): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 85 | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f |
| 0x7f (127): | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 83 | 7f | e9 | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f | 7f |
| **hex offset** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **a** | **b** | **c** | **d** | **e** | **f** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **1a** | **1b** | **1c** | **1d** | **1e** | **1f** |
| **dec offset** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** |

**PDBR: 0x7a (122)**

83) When accessing **virtual address 0x5891**, what will be the first page accessed (hexadecimal)?
   a. 0x32
   b. 0x58
   c. 0x63
   d. 0x7a
   e. Error or None of the above

84) What **index** of the page directory will be accessed first (hexadecimal)?
   a. 0x05
   b. 0x16
   c. 0x17
   d. 0x1f
   e. Error or None of the above

85) What will be the **second page** accessed? (hexadecimal)
   a. 0x08
   b. 0x22
   c. 0x28
   d. 0x36
   e. Error or None of the above

86) What are the **contents** of the corresponding PTE that will be read?
   a. 0x7f
   b. 0xb7
   c. 0xe7
   d. 0xf7
   e. Error or None of the above

87) What is **the final physical address** for this virtual address?
   a. 0x091
   b. 0x6f1
   c. 0x991
   d. 0xdf1
   e. Error or None of the above

88) What are the **contents** (i.e., the value) at that final physical address?
   a. 0x14
   b. 0x01
   c. 0x0b
   d. 0x7f
   e. Error or None of the above

89) How many memory accesses were required in this example to obtain the contents at the final physical address?
   a. 1
   b. 2
   c. 3
   d. 4
   e. Error or None of the above

**Part 7. Page Replacement Policies**

Assume you have the following stream of accesses to virtual page numbers for some workload:

```
5, 0, 5, 4, 1, 3, 0, 2, 4, 5, 3, 2, 1
```

90) If physical memory contains 4 pages, how many misses will be incurred with the OPT replacement algorithm?
   a. 7
   b. 8
   c. 9
   d. 10
   e. None of the above

91) If physical memory contains 4 pages, how many misses will be incurred with the LRU replacement algorithm?
   a. 7
   b. 8
   c. 9
   d. 10
   e. None of the above

92) If physical memory contains 4 pages, how many misses will be incurred with the FIFO replacement algorithm?
   a. 7
   b. 8
   c. 9
   d. 10
   e. None of the above

**Congratulations on finishing your first CS 537 Exam!**