1. Examples of resources that the OS must manage include CPU, memory, and disk. True.
2. The abstraction that the OS provides for the CPU is a virtual address space. False. The CPU abstraction is a process.
3. A process is defined as an execution stream (or thread of control) in the context of a process state. True.
4. The address space of a process is part of its process state. True.
5. When a user-level process wishes to call a function inside the kernel, it directly jumps to the desired function. False; must use system call; after generating a trap which is handled by the OS with a trap handler, the desired system call is invoked through the system call table.
6. An example of a mechanism inside the OS is the process dispatcher. True.
7. Cooperative multi-tasking requires hardware support for a timer interrupt. False; cooperative assumes process will voluntarily relinquish CPU or enter OS.
8. On a uniprocessor system, there may only be one ready process at any point in time. False; just one RUNNING process, but many could be ready and want to be scheduled.
9. A FIFO scheduler schedules ready processes according to their arrival time. True.
10. The convoy effect occurs when high priority jobs must wait for lower priority jobs. True/False both accepted; convoy effect usually implies short jobs must wait for long jobs, but one could consider a short job to be high priority and a long job to be low priority.
11. A SJF scheduler uses the past run time (i.e., cpu burst) of a job to predict future run time (i.e., cpu burst). False; SJF assumes an oracle with perfect knowledge of future behavior.
12. A STCF scheduler guarantees that it will schedule the ready job with the smallest **remaining** cpu burst. True.
13. If all jobs have identical run lengths, a RR scheduler provides better average turnaround time than FIFO. False; if all jobs are identical, RR is horrible for turnaround time because all jobs will complete at nearly the same time.
14. With a MLFQ scheduler, compute-bound jobs are given higher priority. False; jobs that do a lot of computation (long CPU burst) are given low priority.
15. With a MLFQ scheduler, high priority jobs have longer time-slices than low priority jobs. False; high priorities have short time-slices so that interactive jobs can run promptly, but for just a short time.
16. With a MLFQ scheduler, jobs run to completion as long as there is not a higher priority job. False; at the lowest priority level, MLFQ will still RR across jobs of equal priority.
17. The OS provides the illusion to each process that it has its own address space. True
18. The static portion of an address space cannot contain any data. False; read-only data could be in a static portion (along with code).
19. Pointers should not reference the heap. False; it can be bad practice to return pointers to data allocated on the stack.
20. An instruction pointer register is identical to a program counter. True.
21. A modern OS virtualizes memory with time-sharing. False, use space-sharing.
22. With dynamic relocation, hardware dynamically translates an address on every memory access. True.
23. A linear page table efficiently maps physical page numbers to virtual page numbers. False; efficiently maps VPN to PPN (would be expensive to search other way through linear structure).
24. A disadvantage of segmentation is that different portions of an address space cannot grow independently. False; segmentation lets each segment grow independently.

43) A disadvantage of segmentation is that segment tables require a significant amount of space in memory. False; segment tables are usually small (just a base and bounds for each segment and not many segments). 44) With pure segmentation (and no other support), fetching and executing an instruction that performs a store from a register to memory will involve exactly one memory reference. False; fetching the instruction requires one memory reference and executing the store requires a second memory reference.

1. Paging approaches suffer from internal fragmentation, which grows as the size of a page grows. True.
2. A physical page is identical to a frame. True.
3. The size of a virtual page is always identical to the size of a physical page. True.

48) The number of virtual pages is always identical to the number of physical pages. False; the size of the virtual address space can be different than the amount of physical memory.

1. A disadvantage of paging is that it is difficult to track free memory. False; pages are all same size so just use bitmap to track state (free vs. allocated) of each page.
2. A disadvantage of paging is that all pages within an address space must be allocated. False; each page table entry must be allocated (with a linear table), but if the page table entries shows the page isn’t valid, then the page doesn’t have to be allocated.
3. Compared to pure segmentation, a linear page table doubles the required number of memory references. True; need to look up VPN->PPN in page table on every memory access
4. Given a 28-bit virtual address and 1KB pages, each linear page table will consume 218 bytes. False. 1KB pages -> 10 bit offsets. 28 – 10 = 18 bits for VPN. Each PTE is 4 bytes (coversheet assumption). 2^18 \* 4 bytes. address space变大才可以
5. A TLB caches translations from full virtual addresses to full physical addresses. False; TLB translates virtual page numbers to physical page numbers (no offset portion of address in TLB).
6. If a workload sequentially accesses 4096 4-byte integers stored on 256 byte pages, the TLB is likely to have a miss rate around 2-8 (ignore any other memory references). False. Access 16KB of data sequentially; with 256 byte pages (and perfect alignment), this data fits on 2^14 / 2^8 = 2^6 pages. Assume first access to each page misses in TLB, while remaining accesses to that page hit in TLB. Miss rate = # misses / # accesses = # pages / # accesses = 2^6 / 2^12(=4096) = 2^(-6).
7. On a context switch, the TLB must be flushed to ensure that one process cannot access the memory of another process. False; can avoid flushing TLB if have ASIDs.
8. A longer scheduling time slice is likely to decrease the overall TLB miss rate in the system. True; if a process is scheduled for a longer period of time, it will amortize the cost of the cold start misses to load up TLB with needed translations over a longer period of time
9. On a TLB miss, the desired page must be fetched from disk. False; TLB miss means the page tables must be accessed; page fault will need to access disk.
10. With a TLB, only the outermost page table of each process needs to be accessed. False; if a TLB miss, still need to walk entire page table.
11. If the valid bit is clear (equals 0) in a PTE needed for a memory access, the running process is likely to be killed by the OS. True; if process tries to access page not valid in its address space, it is a segmentation fault
12. An inverted page table is efficiently implemented in hardware. False; inverted page tables are implemented in software.
13. One advantage of adding segmentation to paging is that it potentially reduces the size of the page table. True; only need page table entries for valid pages in each separate segment.
14. One advantage of adding paging to segmentation is that it reduces the amount of internal fragmentation. False; Paging has internal fragmentation.
15. A page directory is identical to the outermost level of the page table. True.
16. Given a 2-level page table (and no TLB), exactly 2 memory accesses are needed to fetch an instruction. False; 2 accesses for 2 levels of address translation + 1 for fetch = 3 accesses
17. With a multi-level page table, hardware must understand the format of PTEs. False; not necessary to have hardware support for multi-level page tables
18. If the present bit is clear in a needed PTE, then the running process is likely to be killed by the OS. False; if present bit is clear, page must be brought in from disk
19. A page fault is identical to a page miss. True.
20. When the dirty bit is set in a PTE, the contents of the TLB entry do not match the contents in the page table. False; dirty bit means contents of page in memory do not match contents on disk.
21. When a page fault occurs, it is less expensive to replace a clean page than a dirty page. True; clean page can be simply discarded since it matches what is on disk; dirty page must be written to disk to update that (only) copy.
22. When a page fault occurs, the present bit of the victim page (i.e., the page chosen for replacement) will be cleared by the OS. True; the victim page will no longer be present in main memory
23. A TLB miss is usually faster to handle than a page miss. True; TLB miss just requires accessing main memory; page miss requires accessing much slower disk.
24. Prefetching of pages helps sequential workloads to avoid page misses. True; prefetching works well with sequential accesses since can predict next page to be accessed.
25. LRU is an example of a mechanism for determining which page should be replaced from memory. False; LRU is a policy
26. The OPT replacement policy replaces the page that is used the least often in the future. False; OPT replaces the page that will be used the furthest away in the future (not least often)
27. LRU always performs as well or better than FIFO. False; not necessarily. 96) OPT always performs as well or better than FIFO. True
28. FIFO with N+1 pages of memory always performs as well or better than FIFO with N pages of memory. False, see Belady’s anomaly.

99) LRU-K and 2Q use both how recently and how frequently a page has been accessed to determine which page should be replaced. True.

1. The clock policy replaces the least-recently-used page belonging to any process in the system. False; clock approximates LRU, but it doesn’t necessarily replace the single least-recently-used page

求response还是turnaround time要看清，response time=0说明是刚arrive就开始

要注意page是0x开头还是decimal 如果是decimal，去掉1之后要转化位decimal

找physical address = 倒数第二次读取value去掉1然后组合最后一次从上面移下来5位

1. When executing the return-from-trap instruction, hardware restores the user process’s registers from the kernel stack, changes to user mode, and jumps to a new code location.
2. The fork() system call clones the calling process and overlays a new file image on the child process. False, fork() just clones the calling process; the process needs to call exec() to overlay a new file image
3. When an I/O operation completes, the previously blocked process moves into the RUNNING state. False, the process could move to either READY or RUNNING.
4. An STCF scheduler cannot cause jobs to starve. False, long jobs CAN starve if shorter jobs keep arriving.
5. If all jobs arrive at the same point in time, an SJF and an STCF scheduler will behave the same. True, they only behave differently if a job arrives while some jobs are already running (in which case, STCF may preempt).
6. With an MLFQ scheduler, while a job is waiting for I/O to complete, the job remains in the READY state without changing priority. False; jobs move to the BLOCKED state when waiting for an event to complete that doesn’t need the CPU.
7. Two different address spaces (that do not share any pages) must contain different valid vpn’s from one another. False; two different address spaces can have the same valid virtual pages as one another; a vpn is interpreted in the context of the current address space.
8. With segmentation, the address space of each process must be allocated contiguously in physical memory. False, just each segment must be contiguous.
9. With pure segmentation (and no other support), fetching and executing an instruction that performs a store from a register to memory will involve exactly four memory references. False, with segmentation no extra memory references are need for address translations since the base and bounds of each active segment can be kept in a register. So, 1 ref to fetch the instruction and 1 ref to do the store.
10. Compared to pure segmentation, a linear page table doubles the number of memory references (assuming no TLB). True, a linear page table is kept in main memory; for each memory reference in the application, now need to look up virtual address to physical address translation (need one vpn -> ppn mapping).
11. On a TLB miss, a new page in physical memory is allocated by the OS. False, on a TLB miss, the page translation must be found in the page tables. 34) A page fault occurs if the valid bit is clear (equals 0) in a PTE needed for a memory access. False, on a page fault the present big is clear indicating that the page is valid but swapped out to the backing store
12. A multi-level page table typically reduces the amount of memory needed to store page tables, compared to a linear page table. True; with multiple levels, if there are regions of a sparse address space that are not currently valid (e.g., the space between the heap and the stack), then the page tables for those regions do not have to be allocated.
13. When the dirty bit is set in a PTE, the contents of the TLB entry do not match the contents in the page table. False, the dirty bit means that the page in physical memory does not match the version on the backing store
14. If a clean page and a dirty page have both been accessed recently, the page replacement algorithm should replace the clean page over the dirty page. True, it is less expensive to replace the clean page since its contents can just be thrown away (whereas the changes to the dirty page must first be flushed out to disk before that page can be reused)
15. A TLB miss is usually slower to handle than a page miss. False, TLB miss just requires accessing RAM; page miss requires accessing next level of backing store.
16. A different user-level process should be scheduled when a page miss is being handled. True, since this process is blocked waiting for page miss (which will take awhile), another READY process should be run.
17. Increasing the time-slice of an RR scheduler makes the scheduler behave more like FCFS. True; once the time-slice is as long as the job length, it is equivalent to FCFS. 18) Increasing the time-slice of an RR scheduler makes the scheduler behave more like STCF. False, see above. 19) An MLFQ scheduler assumes the presence of an oracle that knows the length of a job’s CPU burst in advance. False; MLFQ changes the priority of a process after it has begun to run as a way of dynamically adapting to the the length of the CPU bursts that it observes 20) An MLFQ scheduler can preempt running j
18. With dynamic relocation, the OS determines where the address space of a process is allocated in virtual memory. False; the OS allocates the address space in physical memory; the compiler/linker chose the actual virtual addresses to lay out code and data.
19. With dynamic relocation, the OS can move an address space after it has been placed in physical memory. True; this is the idea of dynamic relocation (e.g., base+bounds)
20. 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. False; threads share exact same address space so share both base and bounds.
21. Given a constant number of bits in a virtual address, the size of a linear page table increases with more (virtual) pages. True; a PTE is needed for each virtual page, so the size of the table increases with more pages.
22. Given a fixed number of (virtual) pages, the size of a linear page table decreases with a smaller address space. False; fixed number of virtual pages à same number of entries à size for page table
23. A linear page table must be stored contiguously in physical memory. True; that is the problem with linear page tables
24. For faster translations, the OS looks up page mappings from VPN to PPN in the TLB. False; hardware does this…
25. A workload that sequentially accesses each of the elements of a large array once will often require the same VPN to PPN translation. True; the successive elements of the array will reside on the same page which need the same VPN to PPN translation (leading to good TLB hit rates).
26. With a linear page table, all virtual pages within an address space must be allocated. False; while the PTEs have to be allocated, the actual pages do not; they can be invalid (not mapped).
27. A multi-level page table typically reduces the amount of memory needed to store page tables, compared to a linear page table. True; portions of the address space that are not allocated do not need corresponding (inner) page tables.
28. Increasing the number of jobs simultaneously submitted to a system may decrease the throughput of that system. True; if the jobs use more physical memory than available, system will thrash and throughput decreases.