Task 1: MEMORY SYSTEM

PRITISH WADHWA

Question 1

Associative Arrays – A Technical Description

General Working

An associative array is a special kind of array which has as its subscript, not integers from 0 or 1 up to some integer 'n,' but instead, any arbitrary thing is the subscript of the array. The arbitrary thing for the subscript can be anything from characters to strings or words. Associative arrays are known with different names across different programming languages. If someone is using Java or JavaScript, then they know it as "Hash Tables," while python users call them "dictionaries." The main advantage of using associative arrays is that they are a lot more natural and provide a great deal of flexibility. They are not only very useful, but they are even simpler to implement.

Building Associative Arrays

When an associative array is created, in the memory, a standard array of linked lists is built. With the help of some hashing function, the required subscripts are converted into integers. These integers are used as the subscripts to store the data in the array created in memory. There might be some cases when the hashing function produces the same output for two different subscripts. This is known as "Hash Collision." When this is the case, the length of the linked list is increased by adding a new node. When any of the linked list's length exceeds by a certain number, the definition of the hashing function is changed to accommodate each subscript individually. Now they are stored in a new array.

Runtime Analysis

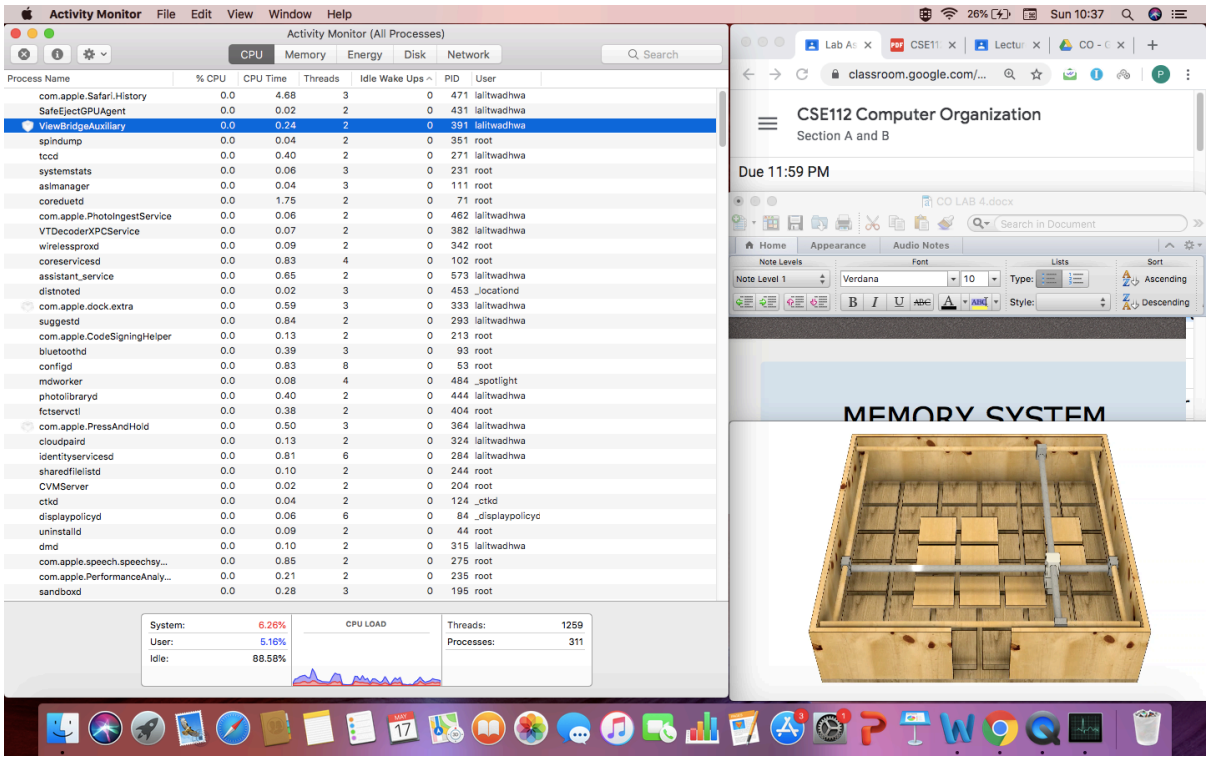
Usually, the maximum length of any linked list in the array does not exceed 2 or 3. Thus the data can be retrieved in constant time.

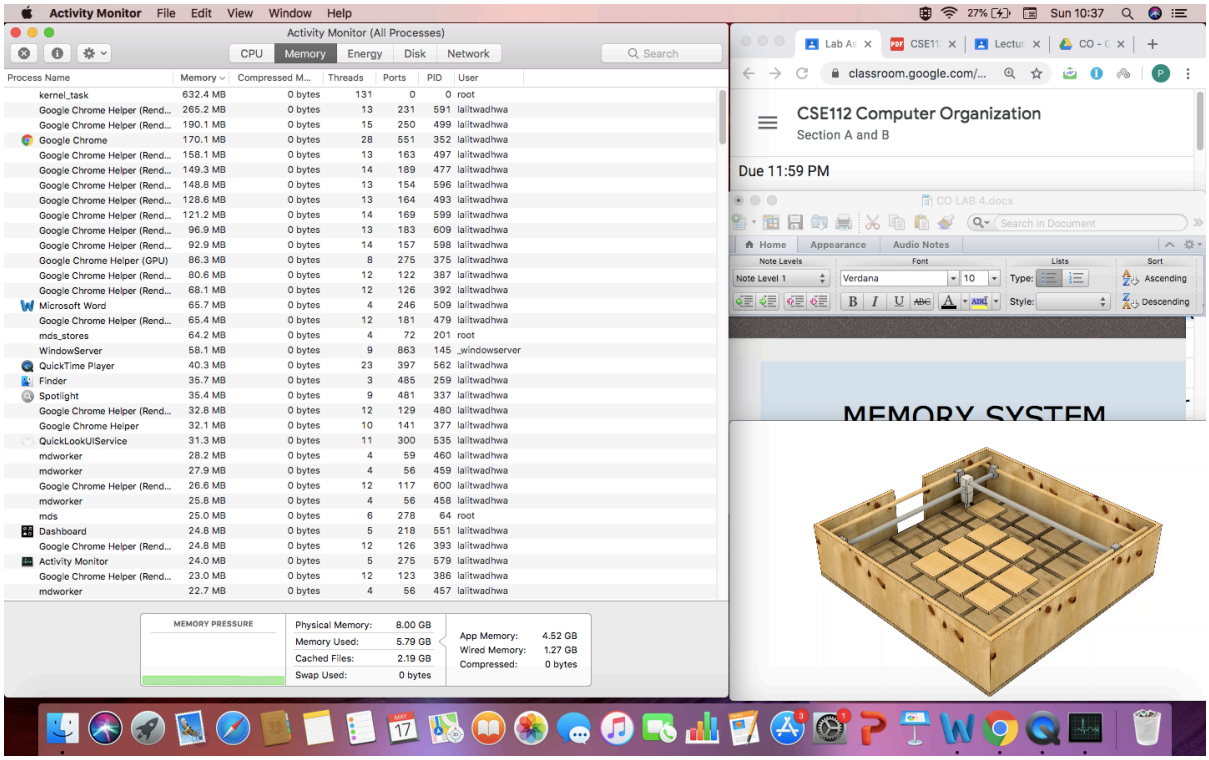
Inbuilt Libraries

Various programming languages offer inbuilt libraries for associative arrays. Some of them are:

1. Java - Hashtable (Class)
2. C++ - Map (STL)
3. Python - Dictionary (Inbuilt Data Structure)
4. Perl - # (Yes! Just the symbol)

Question 2


Activity Monitor File Edit View Window Help
Activity Monitor (All Processes)
CPU Memory Energy Disk Network
Search
Process Name % CPU CPU Time Threads Idle Wake Ups PID User
com.apple.Safari.History 0.0 4.68 3 0 471 lalitwadhwa
SafeEjectGPUAgent 0.0 0.02 2 0 431 lalitwadhwa
ViewBridgeAuxiliary 0.0 0.24 2 0 391 lalitwadhwa
spindump 0.0 0.04 2 0 351 root
tccd 0.0 0.40 2 0 271 lalitwadhwa
systemstats 0.0 0.06 3 0 231 root
aslmanager 0.0 0.04 3 0 111 root
coreduetd 0.0 1.75 2 0 71 root
com.apple.PhotoIngestService 0.0 0.06 2 0 462 lalitwadhwa
VTDecoderXPCService 0.0 0.07 2 0 382 lalitwadhwa
wirelessproxd 0.0 0.09 2 0 342 root
coreservicesd 0.0 0.83 4 0 102 root
assistant_service 0.0 0.65 2 0 573 lalitwadhwa
distnoted 0.0 0.02 3 0 453 _locationd
com.apple.dock.extra 0.0 0.59 3 0 333 lalitwadhwa
suggestd 0.0 0.84 2 0 293 lalitwadhwa
com.apple.CodeSigningHelper 0.0 0.13 2 0 213 root
bluetoothd 0.0 0.39 3 0 93 root
configd 0.0 0.83 8 0 53 root
mdworker 0.0 0.08 4 0 484 _spotlight
photolibraryd 0.0 0.40 2 0 444 lalitwadhwa
fctservctl 0.0 0.38 2 0 404 root
com.apple.PressAndHold 0.0 0.50 3 0 364 lalitwadhwa
cloudpaird 0.0 0.13 2 0 324 lalitwadhwa
identityservicesd 0.0 0.81 6 0 284 lalitwadhwa
sharedfilelistd 0.0 0.10 2 0 244 root
CVMServer 0.0 0.02 2 0 204 root
ctkd 0.0 0.04 2 0 124 _ctkd
displaypolicyd 0.0 0.06 6 0 84 _displaypolicyd
uninstalld 0.0 0.09 2 0 44 root
dmd 0.0 0.10 2 0 315 lalitwadhwa
com.apple.speech.speechsy... 0.0 0.85 2 0 275 root
com.apple.PerformanceAnaly... 0.0 0.21 2 0 235 root
sandboxd 0.0 0.28 3 0 195 root
System: 6.26%
User: 5.16%
Idle: 88.58%
CPU LOAD
Threads: 1259
Processes: 311
Lab As
CSE11
Lectur
CO - C
classroom.google.com/...
CSE112 Computer Organization
Section A and B
Due 11:59 PM
CO LAB 4.docx
Search in Document
Home Appearance Audio Notes
Note Levels Font Lists Sort
Note Level 1 Verdana 10 Type: Ascending Style: Descending
MEMORY SYSTEM


Activity Monitor File Edit View Window Help
Activity Monitor (All Processes)
CPU Memory Energy Disk Network
Search
Process Name Memory Compressed M... Threads Ports PID User
kernel_task 632.4 MB 0 bytes 131 0 0 root
Google Chrome Helper (Rend... 265.2 MB 0 bytes 13 231 591 lalitwadhwa
Google Chrome Helper (Rend... 190.1 MB 0 bytes 15 250 499 lalitwadhwa
Google Chrome 170.1 MB 0 bytes 28 551 352 lalitwadhwa
Google Chrome Helper (Rend... 158.1 MB 0 bytes 13 163 497 lalitwadhwa
Google Chrome Helper (Rend... 149.3 MB 0 bytes 14 189 477 lalitwadhwa
Google Chrome Helper (Rend... 148.8 MB 0 bytes 13 154 596 lalitwadhwa
Google Chrome Helper (Rend... 128.6 MB 0 bytes 13 164 493 lalitwadhwa
Google Chrome Helper (Rend... 121.2 MB 0 bytes 14 169 599 lalitwadhwa
Google Chrome Helper (Rend... 96.9 MB 0 bytes 13 183 609 lalitwadhwa
Google Chrome Helper (Rend... 92.9 MB 0 bytes 14 157 598 lalitwadhwa
Google Chrome Helper (GPU) 86.3 MB 0 bytes 8 275 375 lalitwadhwa
Google Chrome Helper (Rend... 80.6 MB 0 bytes 12 122 387 lalitwadhwa
Google Chrome Helper (Rend... 68.1 MB 0 bytes 12 126 392 lalitwadhwa
Microsoft Word 65.7 MB 0 bytes 4 246 509 lalitwadhwa
Google Chrome Helper (Rend... 65.4 MB 0 bytes 12 181 479 lalitwadhwa
mds_stores 64.2 MB 0 bytes 4 72 201 root
WindowServer 58.1 MB 0 bytes 9 863 145 _windowserver
QuickTime Player 40.3 MB 0 bytes 23 397 562 lalitwadhwa
Finder 35.7 MB 0 bytes 3 485 259 lalitwadhwa
Spotlight 35.4 MB 0 bytes 9 481 337 lalitwadhwa
Google Chrome Helper (Rend... 32.8 MB 0 bytes 12 129 480 lalitwadhwa
Google Chrome Helper 32.1 MB 0 bytes 10 141 377 lalitwadhwa
QuickLookUIService 31.3 MB 0 bytes 11 300 535 lalitwadhwa
mdworker 28.2 MB 0 bytes 4 59 460 lalitwadhwa
mdworker 27.9 MB 0 bytes 4 56 459 lalitwadhwa
Google Chrome Helper (Rend... 26.6 MB 0 bytes 12 117 600 lalitwadhwa
mdworker 25.8 MB 0 bytes 4 56 458 lalitwadhwa
mds 25.0 MB 0 bytes 6 278 64 root
Dashboard 24.8 MB 0 bytes 5 218 551 lalitwadhwa
Google Chrome Helper (Rend... 24.8 MB 0 bytes 12 126 393 lalitwadhwa
Activity Monitor 24.0 MB 0 bytes 5 275 579 lalitwadhwa
Google Chrome Helper (Rend... 23.0 MB 0 bytes 12 123 386 lalitwadhwa
mdworker 22.7 MB 0 bytes 4 56 457 lalitwadhwa
MEMORY PRESSURE
Physical Memory: 8.00 GB
Memory Used: 5.79 GB
Cached Files: 2.19 GB
Swap Used: 0 bytes
App Memory: 4.52 GB
Wired Memory: 1.27 GB
Compressed: 0 bytes
Lab As
CSE11
Lectur
CO - C
classroom.google.com/...
CSE112 Computer Organization
Section A and B
Due 11:59 PM
CO LAB 4.docx
Search in Document
Home Appearance Audio Notes
Note Levels Font Lists Sort
Note Level 1 Verdana 10 Type: Ascending Style: Descending
MEMORY SYSTEM

- To observe the activity of virtual memory, a web browser, a word processor and a video player were opened simultaneously. To monitor their performances, a window of activity monitor is also opened.

    - Physical memory refers to the actual RAM and is directly accessible by the CPU.
    - Virtual memory is the imaginary memory area and is used to enlarge the address space. Virtual memory is converted to physical memory by means of an address translation system done by mapping the actual memory location of a physical memory.
    - When the physical memory can't accommodate all the processes, they are passed on to the virtual memory and then the virtual memory manager loads processes sequentially on physical memory and waits while the process ends and then rewrites the new process and this cycle is executed till all processes in the virtual space get executed.

- The two screenshots attached highlight the CPU usage and the Memory usage respectively.
    - In the first screenshot, one can easily observe the CPU usage, as used by the processor and the user simultaneously. The graph in the given screenshots, make it really easy to observe the activity.
    - In the second screenshot, the memory division is clear.
        - **Memory Pressure**: The Memory Pressure graph helps illustrate the availability of memory resources. The graph moves from right to left.The current state of memory resources is indicated by the color at the right side of the graph:
            - Green: Memory resources are available.
            - Yellow: Memory resources are still available but are being tasked by memory-management processes, such as compression.
            - Red: Memory resources are depleted, and macOS is using your startup drive for memory.
        - **Physical Memory:** The amount of RAM installed.

- **Memory Used:** The total amount of memory currently used by all apps and processes.
    - **App Memory:** The total amount of memory currently used by apps and their processes.
    - **Wired Memory:** Memory that can’t be compressed or paged out to your startup drive, so it must stay in RAM. The wired memory used by a process can’t be borrowed by other processes. The amount of wired memory used by an app is determined by the app's programmer.
    - **Compressed:** The amount of memory in RAM that is compressed to make more RAM memory available to other processes.

- **Cached Files:** Memory that was recently used by apps and is now available for use by other apps. For example, if you've been using Mail and then quit Mail, the RAM that Mail was using becomes part of the memory used by cached files, which then becomes available to other apps. If you open Mail again before its cached-files memory is used (overwritten) by another app, Mail opens more quickly because that memory is quickly converted back to app memory without having to load its contents from your startup drive.

- **Swap Used:** The space used on the startup drive by the memory management. It's normal to see some activity here. As long as memory pressure is not in the red state, macOS has memory resources available.

Question 3 (Exercise 8)

We are given a two-level cache configuration (L1 and L2) where L1 can store 2 words and L2 can store 4 words.

The Write Back Policy.

- In this method, the values in L2 and subsequent caches is updated when the particular address is removed from its on level higher cache. That is, in this example, values in L2 would be updated, when the address in L1 would be removed from L1.
- Final cache contents:
  - Cache L1:

Address	Contents
22	401
23	501

  - Cache L2:

Address	Contents
20	201
21	301
22	400
23	500

Write Through Policy

- In this method values in all levels of caches is updated as soon as the value of any address is changed in L1.
- Final cache contents:
  - Cache L1:

Address	Contents
22	401
23	501

  - CacheL2:

Address	Contents
20	201
21	301
22	401
23	501

- Question 4:
- Exercise 17:
    - Given:
        - 32 bit Machine
        - 4 KB page size
        - Single Level Page Table
    - To Find:
        - Entries in single page level table
        - Size of each entry in bits
    - The 32 bit virtual address will just include- 20 bit page number and 12 bit offset.
    - The number of entries in a page table is equal to the number of combinations of the 20 page number bits.
    - Therefore, the number of entries = $\mathbf{2^{20}}$
    - Size of each entry in the page table = **20 bits**

- Question 18:
    - Given:
        - 32 bit Machine
        - 4 KB page size
        - Two Level Page Table
    - To Find:
        - Entries in secondary page level table
    - The 32 bit virtual address will just include- 20 bit page number and 12 bit offset
    - Since it is a two level page table, the page is divided into two parts - the MSB (the one with the least variance, or the primary table) and the LSB (the one with greater variance, or the secondary table).
    - Given that out of 20 bits, the primary page table is addressed with 12 bits of the page address, the remaining bits should be addressed to the secondary pages (in this case, only one).
    - Therefore, the secondary page table would have the page number of 20-12 = 8 bits.
    - The number of entries in the secondary page table is equal to the number of combinations of the 8 page number bits.
    - Therefore, the number of entries on the secondary page table = $\mathbf{2^8 = 256}$

Task 2: Multiprocessor System

PRITISH WADHWA

Question 1:

- SSE instructions (**S**treaming **S**IMD **E**xtensions) is an extension of the SIMD execution model introduced with the MMX technology.
- SSE instructions are divided into four subgroups:
  - SIMD single-precision floating-point instructions that operate on the XMM registers
  - MXSCR state management instructions
  - 64–bit SIMD integer instructions that operate on the MMX registers
  - Instructions that provide cache control, pre-fetch, and instruction ordering functionality.
- They were designed by Intel and introduced in 1999 in their Pentium III series.
- SSE contains 70 instructions which mostly works on single precision floating point data.
- SSE was originally called Katmai New Instructions (KNI), Katmai being the code name for the first Pentium III core revision.
- SIMD instructions can greatly increase performance when exactly the same operations are to be performed on multiple data objects. Typical applications are digital signal processing and graphics processing.
- ARM does not support SSE instructions, in turn it supports Neon instructions.
- x86-64 supports SSE instructions.

Question 2:

Moore's Law:

- Gordon Moore, the co-founder and chairman of Intel Corporations put forward an empirical observation in 1965.
- He said that "the number of transistors per chip doubles roughly every two years, though the cost of computers is halved"
- This observation has been termed as the Moore's Law.

Amdahl's Law:

- Gene Amdahl, the founder of Amdahl Corporations, put forward a formula which defines the theoretical speedup in latency of the execution of a task at fixed workload that can be expected of a system whose resources are improved.
- He presented the same in 1967.
- He presented his relation as:

$$S_{\text{latency}}(s) = \frac{1}{(1-p) + \frac{p}{s}}$$

  - $\mathbf{S_{latency}}$ is the theoretical speedup of the execution of the whole task
  - **s** is the speedup of the part of the task that benefits from improved system resources
  - **p** is the proportion of execution time that the part benefiting from improved resources originally occupied.
- The above relation is termed as **Amdahl's Law** or **Amdahl's Argument.**

Question 3 (Exercise 12):

Sequentially Consistent System:

- In this type of system, when 2 or more threads are executed, their internal ordering can not be changed.
- In the given question t1 can only take the value of 1.
- One of the possible arrangements for t1 = 1 can be:
  - x = 1; //thread 1 statement 1
  - y = 1; //thread 1 statement 2
  - while(y == 0){} //thread 2 statement 1
  - t1 = x; //thread 2 statement 2

Weakly Consistent System:

- In this type of system, when 2 or more threads are executed, their internal sequence can also be reordered.
- In the given question t1 can take the value of 0 and 1.
- One of the possible arrangements for t1 = 0 can be:
  - y = 1; //thread 1 statement 2
  - while(y == 0){} //thread 2 statement 1
  - t1 = x; //thread 2 statement 2
  - x = 1; //thread 1 statement 1

- One of the possible arrangements for t1 = 1 can be:
  - x = 1; //thread 1 statement 1
  - y = 1; //thread 1 statement 2
  - while(y == 0){} //thread 2 statement 1
  - t1 = x; //thread 2 statement 2

Question 4:

Exercise 17:

We write back data to the main memory upon a M to S transition in the snoopy protocol.

- A multi-reader, single-writer bus, connects all the caches in a snoopy protocol.
- When one of the caches writes data, the data is broadcasted on the multi-reader and as a result, all the caches can see all the messages in the same order.
- The task at hand is to reduce the traffic on the shared bus.
- This is done by the use of the write invalidate protocol through which a given variable is updated in a single cache only, invalidating the rest of the caches.
- As a result, the data does not need to be broadcasted every time it is modified.
- Simply, at one time only one cache containing the data can be in the modified(M) state.
- When a modified data item is read, we perform a M(shared) to S(shared) transition.
- To consistently evict data from the shared(S) state, we need to ensure that the data is updated in the lower levels before sharing it with other caches to maintain coherence.
- Before we share the modified data to other caches, it must be updated in the lower levels to allow access and eviction done on the updated data.
- Hence, we while doing a M to S transition and write back data in main memory.

- Exercise 18:

    - In the snoopy protocol, all the caches are connected to a multi-reader and a single-writer bus.
    - The shared bus broadcasts data, as a result, the caches are able messages in the same order.
    - The shared bus has the property of mutual exclusivity that only allows one node one node to enter the M state at a time.
    - Since the snoopy protocol ensures that only one node enters the M state in order to write the message, the other node will wait for the first node to complete the write operation and then start to perform its write operation.
    - Hence the order of messages would be preserved since all the other caches will see the same order of the messages.

- Exercise 19:

    - It will take a long time to broadcast a message to all the caches, if for instance we have 64 cores with a private cache per core.
    - Broadcasting of data to all the private caches through the shared bus increases traffic on the bus which leads to increased cache access time and the power consumption.
    - One of the ways to circumvent this problem is to Adopt the Write Invalidate Protocol instead of the Write Update Protocol.
    - In Write Invalidate Protocol, it is ensured that the data is updated in just one cache at a time. This reduces broadcasting of data every time we are modifying it.
    - Data blocks are not always present in all caches at a given time.
    - Another solution to solve the scalability issue is to maintain a directory of caches that stores, dynamically updates all the caches containing the concerned data block at a given time.
    - This approach is inspired from Directory Protocol.
    - The update signal is sent by the shared bus only to those caches that are in the directory, instead of all the private caches on the shared bus, when we modify the data block.

Task 3: Devices, Interrupts and Operating Systems

PRITISH WADHWA

Question 1:

- CPUs define different types of instructions based on access levels.
- This is done for security purposes and fault tolerance.
- In any CPU there are the two levels, Privileged/Kernel level and User level.
- This is also known as ring-based security, where ring 0 is kernel level and ring n is user level.
- There are provided two types of instructions:
    - Privileged (I/O instructions, clearing memory, handling interrupts, etc.)
    - User-Level Instructions(Reading system time, etc).
- The program status word(PSW) is a 64 bits long data block, that functions as a status register.
- The 64 bit PSW stores the current status of the system.
- It describes, among other things, comparison flags and CPU modes.
- Many systems provide more than one level of privileges, for more fine-grained control.
    - For example, the Multics OS introduced multiple levels of protection or privilege which was a revolutionary concept at a time.
    - OS/2 used 3 levels of privileges, ring 0 for kernel code and device drivers, ring 1 for privileged code(User level programs with I/O permissions), and ring 2 for unprivileged code.
    - Recent CPUs such as the Intel VT-X and the AMD-V offer another level above ring 0, called ring -1, for controlling the hardware of ring 0. This helps in Virtualization software.
- Another privilege level is IOPL(I/O privilege level).
- It provides access to I/O ports.
- If the Current Privilege Level(CPL) is less than the 2 bits defining IOPL, then the current program doesn't have I/O access.
- Only a program running in level 0 privilege can change the IOPL bits.

Question 2 (Exercise 29):

- Given:
  - Hard disk with following parameters:
    - Seek Time = 50 ms = 0.05 sec
    - Rotational Speed = 600 RPM = 10 RPS
    - Bandwidth = 100 MB/s
- Time taken to read the data = Average time taken to reach the concerned sector + time needed to access data from the required track + seek time
- Time taken to reach the concerned sector will we given by the average rotation time which is :
  - Time_taken_for_1_rotation/2=0.1 x ½=0.05s
  - since 10 rotations in 1sec, therefore 1 rotation in 1/10 sec.
- Time to access data from track :
  - Amount_of_data_to_be_read/bandwidth = 25/100 = 0.25s

Part a:

Time taken to read the 25 MB data = 0.05 + 0.25 + 0.05 = **0.35 sec**

- Now, we have to take into consideration that data can be read in chunks of 5MB only.
- Total Time taken to read the data = Time required for disk rotation + time needed to access data from the required track + seek time
- Since 5 MB data is to be read at once, the time taken to access it from the required track:
  - Amount_of_data_to_be_read/ bandwidth = 5/100 = 0.05 s
- Time taken to read 5(since 5 x 5=25) chunks of 5 MB data:
  - 5 * 0.05 = 0.25 s
- Next, 25 MB data is read in chunks of 5 MB, the platter will rotate 5 times.
- For the first rotation, we need to find the average time and for the next four rotations (which are known) we need to find the max time.
- Average rotation time of 1 rotation:
  - Time_taken_for_1_rotation/2=0.1 x ½=0.05s
  - Since 10 rotations in 1sec, therefore 1 rotation in 1/10 sec.
- Rotation time for 5-1 = 4 rotations = 0.1 x 4 = 0.4s

- Time required for disc rotation:
    - Average Time of 1 rotation + Time for 4 rotations:
        - 0.05 + 0.4 = 0.45 sec

Part b:

Maximum Time to read the entire 25 MB data:

- 0.45+ 0.25 + 0.05 = **0.75 sec**

Question 3 (Exercise 34):

- Data in the following disks:

Disk	D0	D1	D2	D3
Data (Contents)	0xFF00	0x3421	0x32FF	0x98AB

- To find the value of the parity block, we need to perform bitwise XOR of the contents of the Disks D0, D1, D2, D3
- For this we need to convert the data present in these disks in binary format as it is originally represented in hexadecimal format.

Disk	Data (Hexadecimal)	Data (Binary)
D0	0xFF00	$(1111\ 1111\ 0000\ 0000)_2$
D1	0x3421	$(0011\ 0100\ 0010\ 0001)_2$
D2	0x32FF	$(0011\ 0010\ 1111\ 1111)_2$
D3	0x98AB	$(1001\ 1000\ 1010\ 1011)_2$

- Let the value of Parity block be represented by P.
  - Hence, $P = D0 \oplus D1 \oplus D2 \oplus D3$
    - $P = (1111\ 1111\ 0000\ 0000)_2 \oplus (0011\ 0100\ 0010\ 0001)_2 \oplus (0011\ 0010\ 1111\ 1111)_2 \oplus (1001\ 1000\ 1010\ 1011)_2$
    - $P = (0110\ 0001\ 0111\ 0101)_2 = (6175)_{16}$ = **0x6175**

- Now, the contents of D0 is changed to 0xABD1, hence we update our table and represent disk D0 by D0’

Disk	Data (Hexadecimal)	Data (Binary)
D0’	0xABD1	$(1010\ 1011\ 1101\ 0001)_2$
D1	0x3421	$(0011\ 0100\ 0010\ 0001)_2$
D2	0x32FF	$(0011\ 0010\ 1111\ 1111)_2$
D3	0x98AB	$(1001\ 1000\ 1010\ 1011)_2$

- Let P’ be the new value of the parity block
  - $P' = D0' \oplus D1 \oplus D2 \oplus D3$
    - $P' = (1010\ 1011\ 1101\ 0001)_2 \oplus (0011\ 0100\ 0010\ 0001)_2 \oplus (0011\ 0010\ 1111\ 1111)_2 \oplus (1001\ 1000\ 1010\ 1011)_2$
    - $P' = (0011\ 0101\ 1010\ 0100)_2 = (35A4)_{16}$ = **0x35A4**

Bonus Task PRITISH WADHWA

Question 1:

Journal on Interrupts

1 May 2020, Friday 8:00 PM

- OS (Operating system) is the event-driven system software.
- It gets executed only when there is a system call, traps, or interrupts.
- When a user process occurs and an event is completed, the control jumps to an OS.
- The OS is turns executes the necessary process and feeds the control back to user-space.
- OS is event-driven as it can't rust user processes as they might be malicious or contain bugs.
- Also, the user process crash should not affect the OS.
- The OS also needs to give fairness to all the processes and can't let one process hog the CPU time.
- Events can be of two types:
    - Interrupts (software interrupts and hardware interrupts)
    - Exceptions.
- Hardware devices raise hardware interrupts.
- They are asynchronous and may occur at any time.
- Software interrupts, also known as traps, are raised by user programs to invoke OS functionality.
- The processor generates exceptions automatically when it is fed an illegal instruction.
- Exceptions in turn are of two types:
    - Faults (easily recoverable)
    - Aborts (difficult to recover)

2 May 2020, Saturday 7:00 PM

- If we observe the Event View of the CPU, we keep on running while the next instruction is being fetched
- When the instruction is executed, it calls to check if an event is completed or not.
- If the event is completed, the event is executed in the event handler, else the control goes to fetch new instruction.
- When an event is executed, Each exception/Interrupt is given a number that is used to index into an Interrupt descriptor table (IDT).
- IDT provides the entry point into an interrupt/exception handler.
- 256 vectors are available:
    - 0-31 are used internally (like 0 stands for divide error)
    - 32-255 can be defined by the OS.
- We also have xv6 interrupt vectors in which:
    - 0 to 31 are reserved by Intel
    - 32 to 63 are used for hardware interrupts
    - 64 is used for system call interrupt

3 May 2020, Sunday 8:00 PM

- Generally CPUs have 2 interrupt pins:
  - INT: Interrupt
  - NMI: Non-maskable – for very critical signals.
- To support more than 2 interrupts, CPUs have interrupt controllers that take in multiple interrupts and channelize them to the CPU using the INT pin.
- One such controller is 8259 Programmable Interrupt Controller (PIC), which relays up to 8 pins.
- Interrupts are raised by interrupt request (IRQ)
- Interrupts can be of 2 types:
  - Edge Interrupts
    - In Edge-Triggered Interrupt, exactly one interrupt occurs when the IRQ line is asserted.
  - Level Interrupts.
    - Level triggered Interrupts remain active even after interrupt service is complete.

4 May 2020, Monday 6:00 PM

- What happens when there is an Interrupt?

    i. After current instruction completes CPU senses interrupt line and obtains IRQ number from LAPIC.
    ii. Switch to kernel stack (if necessary)
    iii. Basic program state saved
    iv. Jump to interrupt handler
    v. Interrupt handler (top half)
    vi. Return from interrupt
    vii. Interrupt handler (bottom half)

- There are two types of stacks:
    - User-Space Stack
    - Kernel Space Stack
- If necessary we need to switch amongst the above two stacks.
- When an event occurs, the control in the user space privilege changes from low to high.
- The reason we switch the stack is that the OS cannot trust the stack (SS and ESP) for the user process.
- To switch stacks, CPU should know the locations of the new SS and ESP, which in turn, are done by a task segment descriptor.
- Now, we need to save the program state during this process as the current program being executed must be able to resume after interrupt service is completed.
- When a stack switch occurs we need to also save the previous SS (stack segment) and ESP (stack pointer).
- For finding the Interrupt/Exceptions we have IDT: Interrupt descriptor table (also called Interrupt vectors)
- They are stored in the memory and are pointed to by IDTR.
- To get Interrupt Procedure, the interrupt vector invokes the trap gate which in turn, returns a segment selector that goes to the segment descriptor (present in GDT) and a base address is generated.
- This base address is added to the offset (also generated by the trap gate) and is given to the interrupt procedure.
- The CPU does this automatically.

5 May 2020, Tuesday 9:00 PM

- Interrupt Handler:
  - Saves additional CPU context (done by alltraps in xv6)
  - Processes interrupts
  - Invokes kernel scheduler
  - Restores CPU context
  - Returns (written in assembly).
- **Interrupt latency** is the time that elapses from when an interrupt is generated till when the source of the interrupt is serviced.
- OS should 'guarantee' interrupt latency is less than a specified value.
- Maximum Interrupt Latency occurs when the interrupt handler cannot be serviced immediately.
- Due to the Interrupt controller, latency can also be minimized.
- To improve system responsiveness during handler execution, interrupts are enabled within handlers, which cause nested interrupts.
- These make the system more responsive but difficult to develop and validate.
- Therefore interrupt handlers should be small so that nested interrupts are less likely.
- To solve the same problem in LINUX, the top and bottom half techniques are used.
  - Top half:
    - Do minimum work (only the work which is important)
    - Return from the interrupt handler.
  - Bottom half:
    - Deferred Processing
- To take the example of the keyboard interrupt handler:
  - When a keyboard key is pressed, the keyboard interrupt handler, console interrupt is invoked.
  - With the help of the keyboard driver, it identifies which keyboard key was pressed
  - It then pushes it into the circular buffer and control return backs to the user-space.

6 May 2020, Wednesday 7:00 PM

- Unlike hardware interrupts, which have an external device like PIC to trigger the interrupt instructions, software interrupts are invoked by the execution of commands.
- The commands can be like INT x, where x is an interrupt number generally lesser than 256 which helps in distinguishing software interrupts.
- Software interrupts are used for implementing system calls.
- In Linux, INT 128 and in xv6 INT 64 is used for system calls.
- For instance when INT 128 command is executed in the user process, the control to kernel space is invoked, thus the OS performs function using relevant input parameters.
- An example of a system call can be, execution of the print() statement.
    - When this statement is executed "stdlib" library is called and "libc" is invoked.
    - Subsequently, write (STDOUT) is invoked in the user space and this invokes an interrupt
    - The control is transferred to the handler in kernel space, which finally implements the write system call.
- How does the OS distinguish between the system calls?
    - System call numbers are used to distinguish between system calls.
        - So in xv6, before the INT 64 command is executed, a command like "mov x %eax" is executed.
        - This command moves a number x into the eax register (each system call is assigned a unique number).
        - Based on the system call number function, syscall invokes the corresponding syscall handler.
        - A general system call is represented by:
            - int system_call(resource_descriptor, parameters)
        - Typically the return type of the system calls is an integer but sometimes may differ depending on the system call.

- resource_descriptor is the target OS resource like a file, device, etc.
- The System call specific parameters are passed.
- Unlike function calls, passing parameters in system calls is typically done by:
    - Pass by Registers (for example Linux)
        - In pass by registers, if 6 or more arguments are passed, a pointer to block structure containing the argument list is also passed.
        - Note: max size of the argument is the register size.
    - pass via a user-mode stack (for example xv6)
        - In passing via user stack mode, parameters are pushed to the kernel stack
        - This kernel stack is used to create a trapframe.
        - The CPU automatically pushes some parameters in the trapframe to the kernel stack.
        - ESP and SS determine various parameters of system call.
        - ESP, the stack pointer keeps a track of the call made from the user-space.
        - eax contains the system call number, and after the kernel operation is performed it is overwritten by the value returned by the system call.
    - Pass via a designated memory region.

7 May 2020, Thursday 10:00 PM

- There can be three different common sources for exceptions:
  - Program-Error Exceptions (for example divide by 0)
  - Software generated exceptions(for example INTO overflow instruction, BOUND range exceeded)
  - Machine check Exceptions which occurs due to a hardware error (for example cache memory errors)
- There can be 3 different types of Exceptions:
  - Faults-exceptions:
    - Generally, they can be corrected and after that program can continue execution. Some examples being:
      - Divide by zero error
      - Invalid opcode
      - Device not available
  - Traps
    - Traps are reported immediately after the execution of the trapping instructions. Some examples being:
      - Breakpoint instructions
      - Overflow instructions
      - Debug instructions
  - Aborts
    - Aborts are the severe unrecoverable errors. Some examples being:
      - Double fault
        - They occur when an exception is unhandled
      - Machine Check
        - They are the internal errors in hardware like:
          - Bad memory
          - Bus errors
          - Cache errors

References:

1. http://www.it.uu.se/education/course/homepage/os/vt18/module-1/definitions/
2. http://www.it.uu.se/education/course/homepage/os/vt18/module-1/exception-and-interrupt-handling/
3. http://www.it.uu.se/education/course/homepage/os/vt18/module-1/waiting-for-keyboard-input/
4. https://www.youtube.com/watch?v=xRaApI85Zqo
5. https://www.youtube.com/watch?v=1M5qXXlap0U
6. http://www.cse.iitm.ac.in/~chester/courses/15o_os/slides/5_Interrupts.pdf

- Question 2:

Critiques on Interrupts

- Video by IIT Madras

Through the course of the thirty minutes video, the speaker covers a lot of topics, including, but not limiting to, interrupts and its types, their importance, interrupt handling, control transfer, exceptions and their types. The duration of the video is neither too long nor too short and it successfully gives the viewers a decent idea of the topic. The speaker, however, speaks blandly and the viewer is not able to create a decent connection. Throughout the video, the speaker keeps on pausing and losing his eye contact, which indicates a sense of nervousness and confusion in. The flow of the topics in the video is not smooth enough to pass the bar and anyone can easily put the blame on the speaker due to his actions as stated above.

As if the above points were not enough, the structure of the video is also not designed efficiently. To give one instance for the same, at one moment, the speaker is talking about Interrupts and its importance, and at the very next moment, he jumps to the introduction of Exceptions. The speaker then again picks up Interrupts, where he left them, and then again instantly starts talking about Exceptions. The viewer may not be able to understand either topics comprehensively due to poor structure and the flow of the video.

The speaker, however, explains some topics in a greater depth. For instance, while explaining the importance of Interrupts, the speaker illustrates the topic with the help of multiple examples. These definitely help the viewer to get a good grasp on the topic being discussed. The speaker also carries out a great amount of comparison between different types of topics like interrupts and exceptions. Through the comparisons, the speaker tries his best to explain the subtopics of both the topics at hand.

The speaker takes help of a presentation, which runs in the background. The presentations, aide the viewer to understand the topic which the speaker is trying to teach. The presentation contains visual descriptions of various topics, which help the learner to grasp things better. It also contains bullet points, which the speaker takes up and explains in great details.

Despite the use of a presentation, there are many topics that are not covered by the speaker in the depth, which is desired by the viewers. One of these topics include LAPIC. The speaker does not shed any light on APIC's and directly jump on to LAPIC. Due to this, whatever short information being provided by the speaker, a new learner cannot grasp it. Similarly different privilege levels and their uses also require more explanation, which is not covered in by the speaker.

To sum up, the video tries to cover maximum possible matter on Interrupts. While doing so, the speaker overlooks some of the key terms and concepts that are required by any learner to grasp things well.

- Video by IIT Kharagpur

Throughout the course of the thirty-five minutes video, the speaker tries to cover various aspects of interrupts and interrupt handling in sufficient detail. The duration of the video is neither too long nor too short and thus, it gives the viewers a decent idea of the topic. The video is well paced and the speaker is able to deliver the information on the topics in a very efficient manner thus the viewers don't have any difficulty in making the connection required.

In the initial few minutes of the video, we see the absence of any presentation or visual representation in the background; instead the camera focuses on the speaker's face. This is not something that is desired in this kind of educational video. Moreover, several slides are text heavy which doesn't aid much in this mode of learning. The video is also content heavy, which makes it difficult for new learners to get a good grasp on the topics being taught.

Despite the above points, as the video progresses the speaker provides an in-depth analysis of many topics in interrupt handling. The way, the speaker explains the topics are commendable. In some parts of the video, the speaker explains the topics using hand made drawings that the speaker draws on the spot. These not only aid the learner in grasping the contents, but also leave a deep remark on the audience.

The speaker also discusses multiple critical and complex topics like PSW, concept of multiple interrupts, enabling and disabling of interrupts and the needed to do it, interrupt return and service routine, nested interrupts and other relevant topics. These put a lasting effect on the learners and leave them with a never-ending hunger of wanting more to learn.

The presentation, which did not seem much impressive in the initial parts of the video, improves its quality steadily. The slides are aided by the use of multiple flowcharts that gives a very good idea to the viewers. Diagrams and visual representations, which were initially missing, also appear which improves the content quality four-folds.

To conclude, the video is great for viewers who have intermediate knowledge of the topic and want to have an in-depth understanding of it. The speaker could have further improved the quality of his lecture, had he summarized the lecture in the end, but nonetheless, this video is certainly amongst the one which will definitely help the ones who give it a watch.

- Video by IIT Delhi

Throughout the course of this hour long video, the speaker covers multiple aspects of interrupts, privilege modes, exceptions, interrupt handling and flow of control when an interrupt is invoked. The duration of the video is a bit too long and the viewers may find it difficult to watch the entire video in one go. The pace of the video is very slow which coupled by its length, doesn't attract much viewership.

The video starts with an example, explaining the flow of control for a normal task, like pressing a keyboard key. Through this example the speaker tries to explain the concept of interrupts and then tells us in technical terms what exactly they are. However, even after watching the first half of the video the user might not get a good idea of what is the need of interrupts and how exactly they work, as a majority of the topics covered during the first half are intuition based and lack technical explanation.

The speaker tries to explain many concepts like Precise exceptions, first in layman language and then come at the technical description. However, this strategy of his forces him to spend a considerable amount of time on some unrelated concepts as well. This in turn leads to reduced attention by the viewers. Due to this method of his, despite the length of the video a number of topics revolving around interrupts are left behind by the speaker.

Nonetheless, the topics that the video covers, like precise exceptions, marked instructions, privilege levels, oldPC register are structured properly and explained well with the help of examples. Moreover, the speaker constantly used a pen pointer to highlight, draw or write in video, which helped the viewers to better understand the concept that might not have been the case if only text was present.

Before introducing any concept, the speaker introduces a problem and through the course of solving the problem the speaker explains the concept/topic he wants to teach. This way the viewers can understand the underlying practical concepts well. Also, the video takes us through the example of a code by which we can understand the flow of control in

technical terms. To conclude, even though the video may be long and not cover topics in depth, it still gives a brief idea about interrupts and interrupt handling to all the viewers.

References:

1. https://nptel.ac.in/course.html
2. https://www.youtube.com/watch?v=YBl9gljYAaQ
3. https://nptel.ac.in/courses/106105163/
4. https://nptel.ac.in/courses/106/102/106102157/