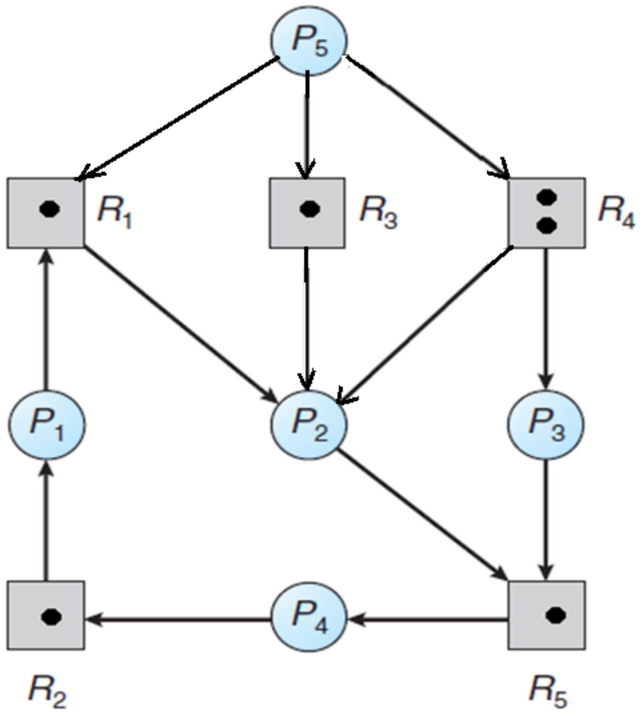
Assumption University
Vincent Mary School of Engineering, Science and Technology
CSX3008(OS) Quiz II (60 marks) 2/2024 KEY

<u>A non-programmable calculator is allowed, and This is a closed-book examination</u>.

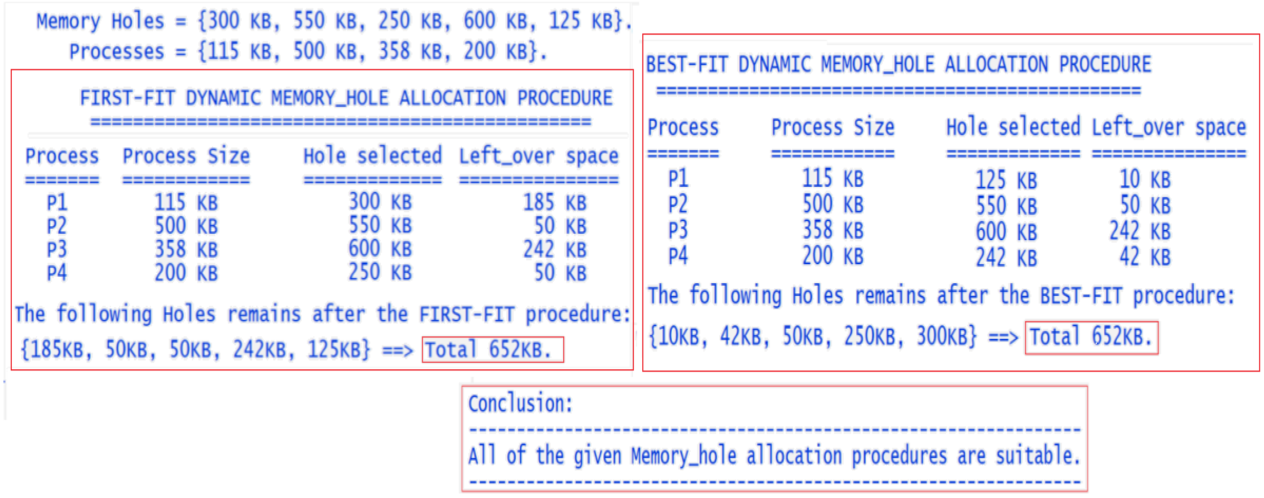
1. (2 points) Define the term **deadlock**. What are the **four conditions** that can cause a deadlock in a system? **Find this answer from the corresponding lecture slide**
2. (2 points) Consider the **resource allocation graph** (RAG) shown in the **Figure** below, attempt the **graph reduction** method, and determine whether a **deadlock** exists or not (where R_1, R_2, R_3, R_4, and R_5 are resources with their units, and P_1, P_2, P_3, P_4, and P_5 are processes). **Deadlock exists because the RAG cannot be reduced.**



(5 points) Consider the process snapshot of a system shown in **Figure 2** (where P_0, P_1, P_2, P_3, and P_4 are processes and A, B, C, and D are resources), and the resources have **13**, **11**, **6**, and **8** units respectively. If the system is in a ***safe state***, then show its process safe state sequence. Otherwise, show the sequence of failed and completed processes (if any). Finally, show all of your computation steps. **Safe state sequence: <P2, P3, P4, P0, P1> OR <P2, P3, P4, P1, P0>** **<u>Its computation steps are not showing here; find them yourself.</u> (you need to show computation steps to get a full score).**

	Max				*Allocation*			
	A	*B*	*C*	*D*	*A*	*B*	*C*	*D*
P_0	5	6	1	7	3	0	1	3
P_1	3	2	3	4	2	2	0	0
P_2	3	2	2	1	2	0	2	0
P_3	4	6	1	2	0	5	1	1
P_4	6	3	2	5	3	1	0	2

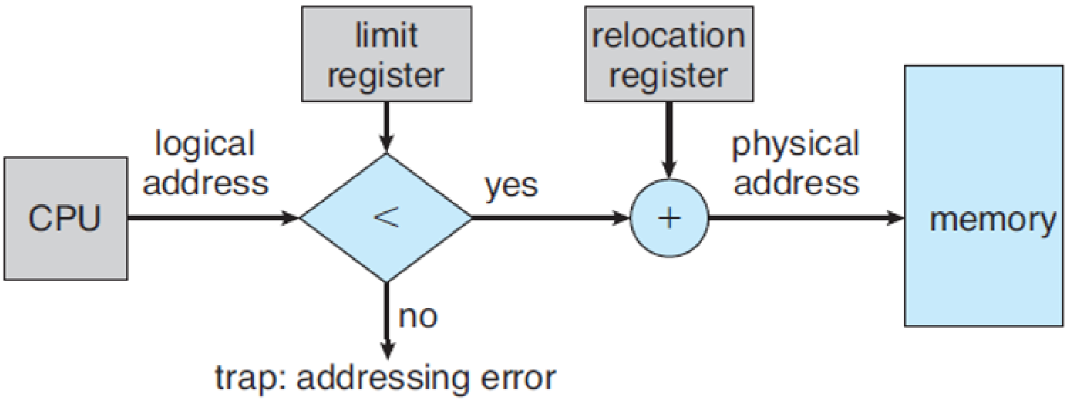
3. (2 points) What is *address binding*? Describe how the address binding supports a user application. **Address binding: The OS translates the logical address (disk address) of the user's executable file to the physical memory address (main memory address) to load it into the main memory, becoming a process. The CPU only executes the process.**
4. (2 points) Describe *demand paging* and *dynamic loading*. **Get this answer from the related lecture slide**
5. (4 points) Given **five** memory holes of sizes 300 KB, 550 KB, 250 KB, 600 KB, and 125 KB (in order), how would the *first-fit* and *best-fit* dynamic processes allocate algorithms allocate **four** processes of size 115 KB, 500 KB, 358 KB, and 200 KB (in order) into the given holes? Rank the algorithms by comparing the total hole space generated after their hole allocation.



```
Memory Holes = {300 KB, 550 KB, 250 KB, 600 KB, 125 KB}.
   Processes = {115 KB, 500 KB, 358 KB, 200 KB}.

   FIRST-FIT DYNAMIC MEMORY_HOLE ALLOCATION PROCEDURE
   ===============================================
Process  Process Size     Hole selected  Left_over space
=======  ============     =============  ===============
  P1        115 KB           300 KB          185 KB
  P2        500 KB           550 KB           50 KB
  P3        358 KB           600 KB          242 KB
  P4        200 KB           250 KB           50 KB
The following Holes remains after the FIRST-FIT procedure:
{185KB, 50KB, 50KB, 242KB, 125KB} ==> Total 652KB.

BEST-FIT DYNAMIC MEMORY_HOLE ALLOCATION PROCEDURE
===============================================
Process     Process Size     Hole selected  Left_over space
=======     ============     =============  ===============
  P1           115 KB           125 KB          10 KB
  P2           500 KB           550 KB          50 KB
  P3           358 KB           600 KB         242 KB
  P4           200 KB           242 KB          42 KB
The following Holes remains after the BEST-FIT procedure:
{10KB, 42KB, 50KB, 250KB, 300KB} ==> Total 652KB.

Conclusion:
-------------------------------------------------------------
All of the given Memory_hole allocation procedures are suitable.
-------------------------------------------------------------
```

6. (2 points) Consider the following statement: "*The entire pages of a process are never loaded into physical memory for execution.*" What is this memory scheme? Describe.
The demand paging scheme reduces a process's primary/main memory space usage and saves memory space.

7. (3 points) Consider the logical address to the physical address translation algorithm shown in the **Figure** below. (a). (1.5 points) Assume that the values of ***logical address***, ***limit register***, and ***relocation*** (***base***) register are **800**, **800**, and **2000**, respectively. Show the physical address of the given ***logical address***. **Trap** (b). (1.5 points) Assume the ***physical address***, ***relocation register***, and ***limit register*** values are **3800**, **3000**, and **900**. Show its logical address. **800**



8. (3 points) Consider the **segment table** shown in the **Table** below. Based on the ***segment table,*** calculate the **physical addresses** (in *decimal*) of the following **logical addresses** (*segment no*, *offset*): a)[1 point] 0, 580. **899** b) [1 point] 2, 599. **1818** c) [1 point] 3, 299. **Trap**

Segment	Base	Limit
0	319	600
1	919	300
2	1219	600
3	1819	299

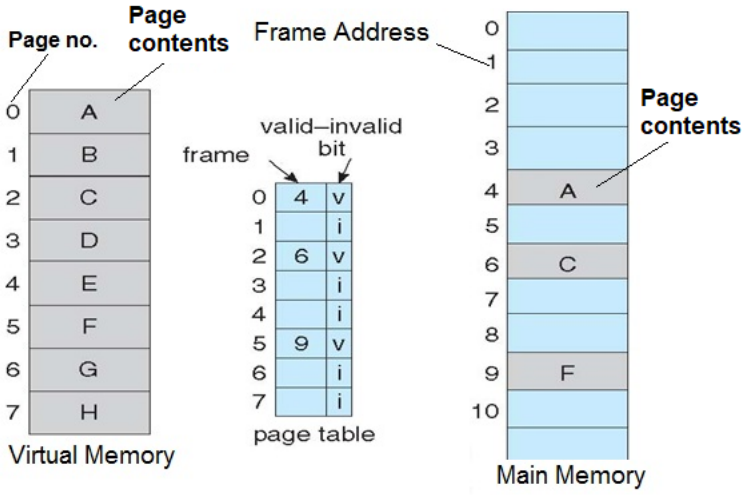
9. (2 points) Consider a computer with a 32-bit virtual address (VA) and a 4k-bit page size. *How many entries are required in a page table*?

10. (3 points) The **table below** shows a section of a paging system's **page Table.** Each ***virtual address*** (**VA**) has an **8-bit page number**. **PA = (Page_frame x VA size) + offset**

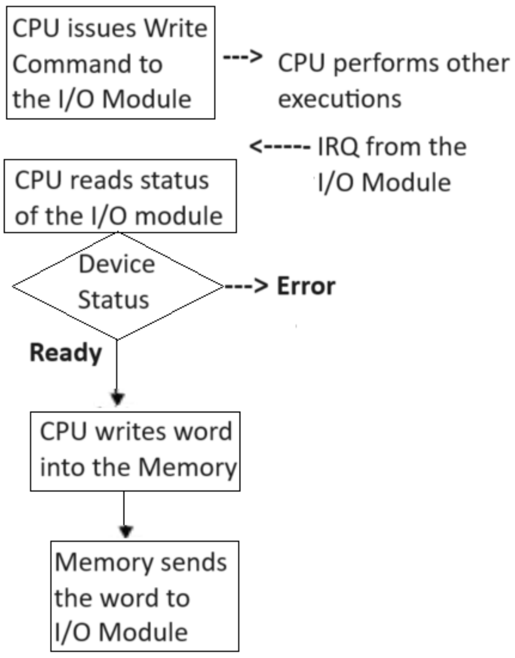
Page No.	v/i bit	Page Frame
0	v	A
1	v	7
2	v	0
3	v	B
---	-	--
E	v	8
F	v	C

Based on the page table, convert the following **virtual addresses** to their equivalent **physical addresses** (**PA**) in decimal: (a). [1 point] 311_{16} **149**(b). [1 point] $E07_{16}$ **103**(c). [1 point] $F8C_{16}$ **284**

11. (6 points) Assume that a user process under execution has **five** pages, its page reference string is "**213425125312513**", and the system has **three memory frames** for the page allocation. Show the performance of **optimal**, **FIFO**, and **LRU** page replacement algorithms regarding their number of **page fault** resolutions (show all your page allocation steps for each algorithm).
Optimal: 4 PF resolved, LRU: 9 PF resolved, and FIFO: 9 PF resolved
The page allocation steps for each algorithm are not shown here; use the steps in the lecture slide.
12. (7 points) A direct virtual memory (VM) system is shown in the **Figure** below. Based on the VM: (a). [1 point] Why does the page table need the ***valid-invalid*** bit? **The v/i bit indicates the availability of pages in the memory frames and helps identify a page fault.** (b). [2 points] Assume **page no. 4** is selected for the subsequent execution. Briefly describe how the OS handles the situation. **The situation is a page fault, and the PFHR of the OS identifies the missed page from the VM, loads it into the available memory frames, and finally updates the page table.** (c). [2 points] Is any page replacement algorithm needed to solve the issue in (b)? Why? **From the VM scenario, there is no need to execute a page replacement algorithm for the missed page because there are free frame spaces in the main memory** (d) [2 points] Suppose there are **1M rows** in the page table. ***How many pages are in the VM***? **1M**



13. (2 points) Describe how an operating system distinguishes various I/O devices of a computer system. **The installed device driver helps the OS identify an I/O device during a system call by the user application.**
14. (2 points) Describe the difference between a *system call* and an *I/O controller request*. **A *system call* is a software interrupt caused mainly by a user application to OS to initiate an I/O operation. However, the *I/O controller request* is a hardware interrupt towards the CPU, resulting from the system call.**
15. (4 points) Identify the nature of the **I/O operation** shown in the **Figure** below. Why?


CPU issues Write Command to the I/O Module
---> CPU performs other executions
<----- IRQ from the I/O Module
CPU reads status of the I/O module
Device Status
---> Error
Ready
CPU writes word into the Memory
Memory sends the word to I/O Module

This is an Interrupt-driven I/O operation because the CPU does not need to wait for the status of the I/O device from its status register after issuing an I/O command. The CPU can do its other executions until the corresponding I/O controller interrupts it by asserting an IRQ (into the CPU's IRQ input). Once the device status is available (either Ready or Error), the I/O controller informs the CPU via an interrupt; the CPU can either accept or reject the interrupt based on the priority of the I/O device.

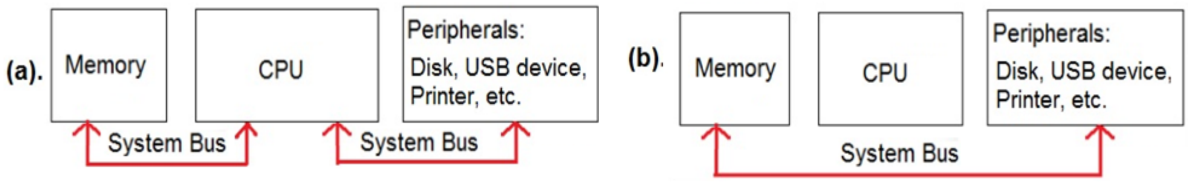
16. (3 points) Describe how the OS supports the **Direct I/O** (I/O Polling) operation. What is its drawback? **Once the device is Ready: a). The running process sends an I/O request to the OS via a system call, hoping that the OS provides some I/O services (e.g., read a file in the Disk) b). The OS receives the I/O request; it blocks the requested process, which means the CPU is released from the process for the I/O operation to be completed. c). The I/O subsystem within the OS (kernel) handles the I/O request. d). The I/O subsystem passes the request to the device driver. e). The device driver sets the related I/O commands based on the request. f). The device controller is ready to support the I/O device. g). The OS can allocate the CPU resources for the I/O operation. Drawback: if the device is not Ready, it will waste CPU time.**

    These are the complete steps; you don't need to show all these steps, which means you can answer in a summarized form.

17. (3 points) Two types of **I/O** operations are shown in **Figures** below **(a** and **b)**. Based on the figure, answer the following:

    a) (1.5 points) Identify the nature of the **I/O** operation in **Figure (a)**. **programmed I/O or direct I/O**

    b) (1.5 points) Identify the nature of the **I/O** operation in **Figure (b)**. **CPU is not involved in data transmission; this is a Direct Memory Access (DMA) I/O operation.**



18. (3 points) Briefly describe how the OS handles the **direct memory access (DMA) I/O** operation. In which situation is the DMA I/O operation preferable? **Once the system call is caused by the instructions related to the DMA operation: a). The DMA controller collects the status of its I/O device and interrupts the CPU as DRQ (DMA Request) to initiate data transfer. b). If the CPU is willing to accept the DRQ from the DMA controller, it sends a DACK (DMA Acknowledgement) signal back to the DMA controller. c). It causes the DMA controller to seize or borrow (called cycle stealing) the system bus (*address bus, data bus*, and *control bus*) from the CPU. d). DMA controller places the desired memory address on the address bus for data transfer between the DMA I/O device and the memory. In the meantime, the DMA controller cancels the DRQ signal, and the CPU cancels the DACK signal to indicate that the CPU is free from data transfer.**

    Most data-transferring devices, such as USB flash drives, Bluetooth devices, HD, SSD cards, etc., have built-in DMA controllers.

    These are the complete steps; you don't need to show all these steps, which means you can answer in a summarized form.

================== **End of Quiz II** ==================