1 E1

Which statements are true?
Select one or more alternatives:

- ☐ The OS allows many programs to run at the same time. ✓
- ☐ The OS takes physical CPU(s) and memory and abstracts them into a more general and easy-to-use virtual form. ✓
- ☐ The OS is responsible for generating assembly code for application programs.
- ☐ The OS allows application software to access hardware resources without needing to know their technical details. ✓

2 E2

Among the options below, which two abstractions help the users to find the storage location of their data?
Select one or more alternatives:

- ☐ directory ✓
- ☐ cache
- ☐ disk
- ☐ register
- ☐ file ✓
- ☐ memory

3 E3

Hardware drivers schedule all access to hardware to avoid conflicts if multiple programs try to access the same device or resource simultaneously.

Select one alternative:

- ○ False ✓
- ○ True

4 E4

Operating system utilities are designed to analyse, configure, optimize or maintain a computer. For example, the vmstat (Virtual Memory Statistics) command returns virtual memory status information, including process states and paging activity.

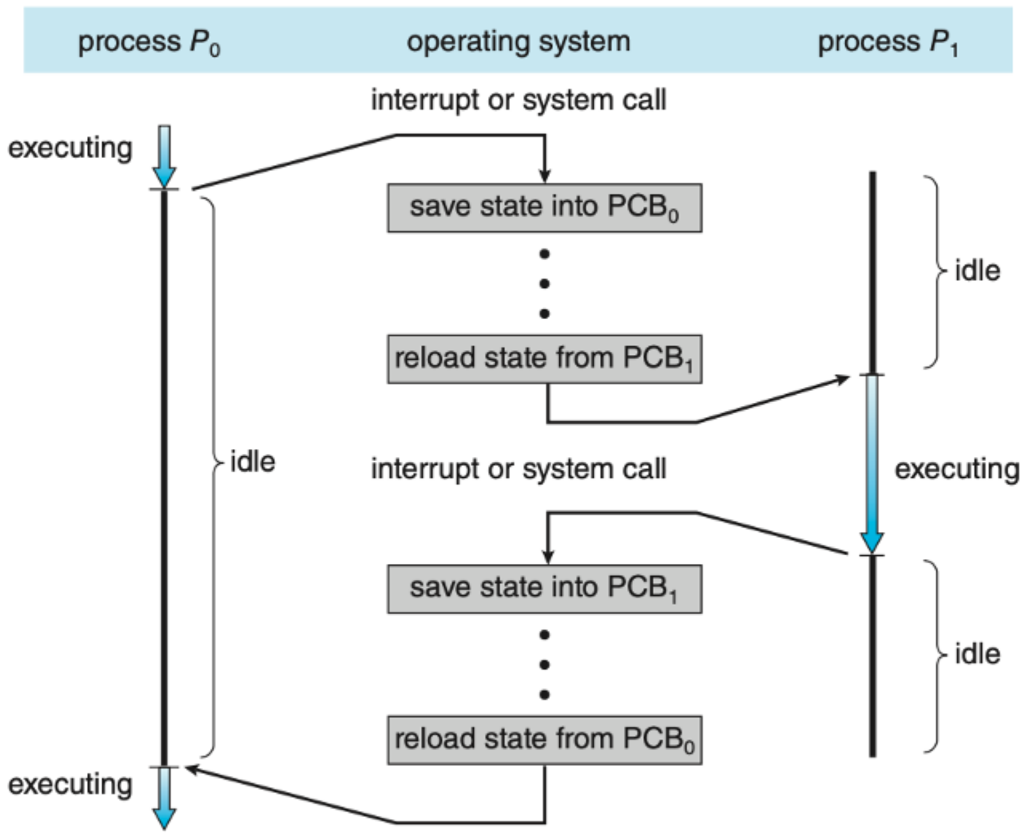
Select one alternative:

- ○ True ✓
- ○ False

5 E5

Which action of the OS does the picture below illustrates?

(context switch, CONTEXT SWITCH, Kontekstskifte, kontekstskifte, KONTEKSTSKIFTE)



6 E6

The sequence of page frames in the physical memory matches the sequence of pages in the address space.

Select one alternative:

- False ✓
- True

7 E7

What is the value of a binary semaphore when it is available?

☐ (1, one, en)

8 E8

A non-terminating recursive function call will cause which of the following problem?
Select one alternative:

- ○ memory leak
- ○ stack overflow ✓
- ○ deadlock
- ○ buffer overflow

9 E9

Compare Hard Disk Drive (HDD) with Solid State Drive (SSD). Which statements are correct?
Select one or more alternatives:

- ☐ HDDs contain mechanical moving parts which make them more noisy then SSDs. ✓
- ☐ HDDs are much cheaper, per gigabyte, than SSDs. ✓
- ☐ SSDs provide a huge performance advantage over HDDs— they're faster to start up, faster to shut down, and faster to transfer data. ✓
- ☐ SSDs are more sensitive to physical vibration than HDDs.

10 E10

The command line "*gcc -o swap swap.c*" compiles the swap*.c* file and generates the executable file *swap*. Based on the given Makefile below, write down a simpler alternative command line which produces the same result as "*gcc -o swap swap.c*", and nothing else.

```
------------------------------Makefile-----------------------------
all: process swap uppercase prime fib

clean:
   rm -f process swap uppercase prime fib

process: process.c
   gcc -o process process.c

swap: swap.c
   gcc -o swap swap.c

uppercase: uppercase.c
   gcc -o uppercase uppercase.c

prime: prime.c
   gcc -o prime prime.c

fib: fib.c
  gcc -o fib fib.c
-----------------------------------------------------------------------
```

[ ] (make swap)

11 E11

Which of the choices below are possible outputs from the execution of the following program?

```
#include <stdio.h>
#include <unistd.h>

int x = 10;

int main(){
   int rc = fork();
   if (rc == 0){
      for( ; ; ){
         sleep(1);
         printf("I am the child, x = %d\n", x);
         x++;
      }
   }else if(rc > 0){
      for( ; ; ){
         sleep(3);
         printf("I am the parent, x = %d\n", x);
         x = 12;
      }
   }
}
```

Select one or more alternatives:

- ☐ I am the child, x = 12 ✓
- ☐ I am the parent, x = 12 ✓
- ☐ I am the parent, x = 11
- ☐ I am the child, x = 10 ✓
- ☐ I am the parent, x = 10 ✓
- ☐ I am the child, x = 11 ✓

12 E12

What does this shell script below do?

```
#! /bin/bash

for x in *
do
    if [ -f $x ]
    then
        cp  $x  $x.v2
    fi
done
```

Select one alternative:

- ○ Backup all the files within the current directory ✓
- ○ Copy all the directories to a new directory
- ○ Rename all the files within the current directory
- ○ Move all the files to a new directory

13 M1

Assume that it takes one time unit for CPU to issue an I/O instruction, three time units for the I/O device to actually perform the task, and then one more time unit for CPU to complete the I/O instruction. So the total time from issuing an I/O instruction to finish the I/O instruction is 1+3+1=5 time units, in which 2 time units are spent on the CPU.

For running two processes, i.e., one process executes 1 I/O instruction and the other process executes 5 CPU instructions, compute the percentage of time that CPU is in use in the following two systems, respectively:

1. The system does NOT switch to another process while one is doing I/O, instead waiting until the process is completely finished.
2. The system switches to another process whenever one is WAITING for I/O.

Select one alternative:

○ 60%, 90%

○ 65%, 95%

○ 50%, 80%

○ 70%, 100% ✓

14 M2

Compute the average response time when running five jobs of different lengths: 3, 10, 5, 2, 20 with the Round Robin (RR) scheduler and a time-slice of 5. Assume 5 jobs arrive in the system with this order at roughly the same time.

What is the average response time of using Round Robin?
Select one alternative:

○ 4.9

○ 6.3

○ 7.8 ✓

○ 5.2

15 M3

When the physical memory space is not enough, the OS will swap pages out of memory to the swap space and swap pages into memory from the swap space. Assume a physical memory can contain at most 3 pages and the memory is empty initially. Track the memory state changes when accessing pages 1,0,2,1,4,0,2,0,1,3 one-by-one in a sequence using the Least-Recently-Used (LRU) swapping policy. What is the hit rate?

Select one alternative:

- 80%
- 50%
- 30%
- 20% ✓

The answers to the following questions can be found in the end of this document.

16 M4

Valgrind is an instrumentation framework for building dynamic analysis tools that check C and C++ programs for memory errors. Below is an output example.

```
...
==3256==  HEAP SUMMARY:
==3256==   in use at exit: 44 bytes in 1 blocks
==3256==   total heap usage: 3 allocs, 2 frees, 2,092 bytes allocated
==3256==
==3256==  LEAK SUMMARY:
==3256==    definitely lost: 44 bytes in 1 blocks
==3256==    indirectly lost: 0 bytes in 0 blocks
==3256==    possibly lost: 0 bytes in 0 blocks
==3256==    still reachable: 0 bytes in 0 blocks
==3256==    suppressed: 0 bytes in 0 blocks
...
```

(1) What kind of memory error is reported by Valgrind? (2 points)

(2) How will you fix this memory problem in the C source code? Just mention an approach is good enough. (3 points)

(The length of the answer: approximately 2 sentences)

Fill in your answer here

17 M5

Duel-mode operation

(1) When does CPU switch from user mode to kernel mode, and vice versa. (2 points)

(2) Point out one benefit of using this dual-mode operation. (3 points)

(The length of the answer: approximately 3~5 sentences)

Fill in your answer here

18 M6

A program can *change* its behaviour over time. For example, a CPU-bound program may transition to an interactive program, or vice versa.

Explain how a multi-level feedback queue (MLFQ) scheduler adjusts the priority of a program which has been **CPU-bound** for a while but now becomes **an interactive program.** (5 points)

(The length of the answer: approximately 5 sentences)

Fill in your answer here

19 A1

Below is an ×86 assembly code with AT&T syntax. Assume variable x is located at memory address 2500 and variable y is located at memory address 3000. The initial value of x is 10 and the initial value of y is 20.

```
  Thread 1              Thread 2
mov 2500, %ax         mov 3000, %bx
mov 3000, %bx         mov 2500, %ax.
add %bx, %ax          sub  %ax, %bx
mov %ax, 2500         mov %bx, 3000
```

(1) Write down the original source code in C for each thread seperately. (2 points each)

(2) What are the possible final values of x and y after both threads finish the execution? Write down the possible pairs of x and y - one for each case. For example, (x=2, y=4) and (x=2, y=7) are two different cases. (6 points for the correct pair of numbers, 4 points for the explanation of how you derive the result)

Fill in your answer here

Format

Words: 0

20 A2

Assume the physical memory size is 64KB, the address space size for the process we are looking at is 16KB, and the page size is 2KB. The format of the page table is described as the following: The high-order (left-most) bit of the page table entry is the VALID bit. If the bit is 1, the rest of the entry is the page frame number (PFN). If the bit is 0, the page is not valid. A table from hexadecimal to binary can be found in the attached file.

Page Table (from entry 0 down to the max size)

```
[ 0 ]  0x8018
[ 1 ]  0x0000
[ 2 ]  0x0000
[ 3 ]  0x800c
[ 4 ]  0x8009
[ 5 ]  0x0000
[ 6 ]  0x801d
[ 7 ]  0x8013
```

Given the page table above, answer each of the following questions to decide which of these 3 virtual addresses are valid and which are not?

Virtual Addresses :

```
VA 0x1008  --- PA or invalid ?
VA 0x33da  --- PA or invalid ?
VA 0x3a39  --- PA or invalid ?
```

(1) How many pages can the address space contain? (1 point)

(2) How many bits are required in a virtual address (VA) in order to address all of the address space? (1 point)

(3) How many top-bits of a virtual address (VA) in this case are used to indicate the page number of the page that the VA locates? (1 point)

(4) How many bits in a virtual address (VA) indicate the offset of VA in its page? (1 point)

(5) How many bits are required in a physical address (PA) in order to address all of the physical memory? (1 point)

(6) For each virtual address above, answer the following questions:

(6-1) Calculate the Page Number. (1 point each)

(6-2) Is the virtual address valid or not?

(6-2-a) If the virtual address is invalid, explain why it is invalid. (2 points each)

(6-2-b) If the virtual address is valid, what is the corresponding page frame number? (1 point each)

(6-3) Write down the corresponding physical address (PA) in the hexadecimal format.

Hint: PA is formed by the concatenation of the page frame number and the offset in the binary format. (2 points each)

Fill in your answer here

Format B I U x_2 x^2 I_x Ω Σ

Hex to Binary

0X 0	0000
0X 1	0001
0X 2	0010
0X 3	0011
0X 4	0100
0X 5	0101
0X 6	0110
0X 7	0111
0X 8	1000
0X 9	1001
0X A	1010
0X B	1011
0X C	1100
0X D	1101
0X E	1110
0X F	1111

INF113 Autumn 2022 – Exam solution

E5:
Context switch, CONTEXT SWITCH
Kontekstskifte, Kontekstskifte, KONTEKSTSKIFTE
Multiprogramming, Multiprogramming

E7:
1, one, ONE, En, EN

E10:
make swap

M4:

(1) memory leak
(2) use the free() function in C library to release or deallocate the memory blocks which are previously allocated by malloc(), calloc(), or realloc() functions.

M5:

(1) Privileged instructions can only be executed in kernel mode. A user application needs to invoke a system call handled by the OS to request a service from the OS. Then the system transitions from user mode to kernel mode to fulfil the request. When the execution of the privileged instructions is finished, the system transitions from kernel mode back to user mode.
(2) The OS can prevent user program to access the kernel directly.

M6:

(1) The MLFQ has several distinct queues, each assigned a different priority level. The processes in the topmost queue have the highest priority. If a job repeatedly relinquishes the CPU while waiting for input from the I/O devices, MLFQ will keep its priority high, as this is how an interactive process might behave. If a job uses the CPU intensively for long periods of time, MLFQ will reduce its priority. The MLFQ scheduler periodically boosts the priority of all the jobs in system, i.e., moves all the jobs in the system to the topmost queue, so, if a CPU-bound job has become interactive, the scheduler treats it properly once it has received the priority boost.

A1:

(1)

Thread 1	**Thread 2**
x = x + y	y = y - x

(2) Since **Thread 1** is the only thread which modifies the value of x, we can simplify the code for **Thread 1** to **x = 10 + y**. Similarly, since **Thread 2** is the only thread which modifies the value of y, we can simplify the code for **Thread 2** to **y = 20 - x**. Due to race conditions, there can be several execution results:

Case 1: Thread 1 reads y before y is modified. x = 10+20 = 30
Thread 2 reads x after x is modified. y = 20–30 = -10
Result: (x=30, y=-10)

Case 2: Thread 1 reads y before y is modified. x = 10+20 = 30
Thread 2 reads x before x is modified. y = 20–10 = 10
Result: (x=30, y=10)

Case 3: Thread 2 reads x before x is modified. y = 20–10 = 10
Thread 1 reads y after y is modified. x = 10+10 = 20
Result: (x=20, y=10)

They may solve the problem at the assembly code level.

A2:

(1) 16KB / 2KB = 8
(2) 16K = 2^14. So, 14 bits are required in the virtual address.
(3) 8 = 2^3. So, the top 3 bits indicate the page number.
(4) 14-3=11 bits are used as the offset
(5) 64K = 2^16. So, 16 bits are required in the physical address.
(6) VA 0x1008 = 01000000001000
The top 3 bits are 010. So, this virtual address belongs to page number 2.
The valid bit of page table entry 2 is 0. So, this virtual address is invalid.

VA 0x33da = 11001111011010
The top 3 bits are 110. So, this virtual address belongs to page number 6.
The valid bit of page table entry 6 is 1. So, this virtual address is valid.
The corresponding page frame number is 0x1d = 00011101
Concatenate page frame number 00011101 with offset 01111011010:
The physical address is 1110101111011010, which is 0xebda.

VA 0x3a39 = 11101000111001
The top 3 bits are 111. So, this virtual address belongs to page number 7.
The valid bit of page table entry 7 is 1. So, this virtual address is valid.
The corresponding page frame number is 0x13 = 00010011
Concatenate page frame number 00010011 with offset 01000111001:
The physical address is 1001101000111001, which is 0x9a39