Problem Set 1: Week 1 ~ 7

Student ID: Student Name:

1. Define the operating system, and describe the main purpose of an operating system?

An operating system is a software between application and hardware. The main purpose of an operating system is to make sure the computer system operates correctly (preventing misusage of system) and efficiently (maximizing system performance) in an easy-to-use manner (virtualizing system resources in easy-to-use form.)

2. Represent the decimal number 1.625 to the hexadecimal number in IEEE 754 standard single precision format.

Since 1.625 = 1 + 0.625 = 1 + 0.5 + 0.125, the decimal number 1.625 is represented in IEEE 754 standard single precision format is 0 01111111 10100000000000000000000. So, its hexadecimal representation is 0x3FD00000.

3. In RISC-V assembly, write an assembly language version of the following C code segment:

```
for (i = 0; i < 98; i++) {
    C[i] = A[i+1] – A[i] + B[i+2]
}
```

The start addresses of arrays `A`, `B`, and `C` are stored in `x18`, `x19`, `x20`, respectively. The local variable `i` is stored in `x5`.

```
addi x5, x0, 0       # x5: i = 0
addi x6, x0, 98      # x6: 98
loop:
    bge x5, x6, end  # if (i >= 98) goto end
    slli x7, x5,     # x7: 4 * i
    add x8, x18, x7  # x8: address of A[i]
    lw x9, 4(x8)     # x9: A[i+1]
    lw x10, 0(x8)    # x10: A[i]
    sub x9, x9, x10  # x9: A[i+1] - A[i]
    add x8, x19, x7  # x8: address of B[i]
    lw x10, 8(x8)    # x10: B[i+2]
    add x9, x9, x10  # x9: A[i+1] - A[i] + B[i+2]
    add x8, x20, x7  # x8: address of C[i]
    sw x10, 0(x8)    # c[i] = A[i+1] - A[i] + B[i+2]
    addi x5, x5, 1   # i++
    jal x0, loop     # goto loop
end:
```

4. Describe each of the four segments of the memory layout of process: text, data, heap, and stack.

Text: Memory spaces for machine-level (binary) instructions
Data: Memory spaces for static variables
Heap: Memory spaces for dynamically allocated variables
Stack: Memory spaces for local variables and values in registers

5. True or False

A. The unix `exec` system call creates a new process. F
B. Processes may use the same virtual address but will actually access different physical addresses. T
C. In a one-level paging scheme, a single page table and a single TLB are shared by all processes. F

D. FCFS has throughput at least as good as RR. T

At the very least, RR incurs the same number of context switches as FCFS. In reality, RR will incur many more context switches, resulting in context switching overhead and suboptimal cache performance. (Throughput is the number of processes that are completed per time unit. However, since we did not cover this metric in our class, the problem 5.D is not on the midterm exam.)

6. How many processes are created?

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

int main(void)
{
    fork();
    fork();
    fork();
    return 0;
}
```

8 (including the parent process)

7. Explain the difference between preemptive and nonpreemptive scheduling.

Preemptive: Scheduling in which processes are involuntarily moved from the running state (e.g., by a timer signaling the kernel to allow the next thread to run).

Nonpreemptive: Scheduling in which, once a core has been allocated to a thread, the thread keeps the CPU until it releases the CPU either by terminating or by switching to the waiting state.

8. Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use nonpreemptive scheduling, and base all decisions on the information you have at the time the decision must be made.

Process	Arrival Time	Burst Time
P1	0.0	8
P2	0.4	4
P3	1.0	1

A. What is the average turnaround time for these processes with FCFS scheduling algorithm?

	0.0	0.4	1.0	8.0	12.0	13.0	Turnaround time
P1	Run	Run	Run	Complete			8.0 – 0.0 = 8.0
P2		Ready	Ready	Run	Complete		12.0 – 0.4 = 11.6
P3			Ready	Ready	Run	Complete	13.0 – 1.0 = 12.0

The average turnaround time is 10.53.

B. What is the average turnaround time for these processes with SJF scheduling algorithm?

	0.0	0.4	1.0	8.0	9.0	13.0	Turnaround time
P1	Run	Run	Run	Complete			8.0 – 0.0 = 8.0
P2		Ready	Ready	Ready	Run	Complete	13.0 – 0.4 = 12.6
P3			Ready	Run	Complete		9.0 – 1.0 = 8.0

The average turnaround time is 9.53.

9. Consider a virtual address space of 256 pages with a 4 KB page size, mapped onto a physical memory of 64 frames.

A. How many bits are required in the virtual address? 20 bits (8 bits for VPN, 12 bits for offset)

B. How many bits are required in the physical address? 18 bits (6 bits for PPN, 12 bits for offset)

10. Consider an operating system that use hardware support for paging to provide virtual memory to applications.

A. Explain how space and time overheads arise from use of paging.

In a page-based virtual memory system, memory is divided into fixed-sized units called pages, which are mapped to corresponding virtual addresses. The operating system uses page tables to keep track of the mapping between physical and virtual addresses. However, page tables leads a significant overhead in terms of memory usage and lookup time. For example, assume a 32-bit system using 4 KB pages. Since the number of bits required for offset in this system is 12 bits (4KB), the number of pages that can be expressed with 20 bits is 2^{20}. Thus, a page table can have a total of 2^{20} page table entries (PTEs). If the PTE is 4 bytes, the total size of the page table is 4 MB. Because page tables exist for each process, page tables arise lots of space overhead. Also, since these page tables can only be stored in main memory or disk due to their size, paging arises lots of time overhead by increasing the number of memory reference (or disk reference).

B. Explain how the Translation Lookaside Buffer (TLB) mitigates the time overheads.

TLB is a cache, which holds recently referenced page table entries, located on the processor itself. Note that there is no need to access the page table in memory when a page reference to an entry in the TLB occurs (TLB hit). The memory access is only required when a reference not in the TLB occurs (TLB miss). We define the average memory access time as

$$\bar{T}_{\text{memory}} = T_{\text{TLB}} + P_{\text{miss}} \times T_{\text{memory}}$$

where T_{TLB} is a TLB lookup time, T_{memory} is a memory access time, P_{miss} is a TLB miss ratio. Thanks to the principle of locality, P_{miss} is usually low. Thus, TLB can reduce the time overhead by paging ($\bar{T}_{\text{memory}} < T_{\text{memory}}$)

C. Explain how the multi-level page table mitigate the space overheads.

Multilevel page decreases space overhead by using a tree data structure. Since multi-level page table do not actually hold PTEs for unused pages, it can reduce the overall size of the page table. However, there is a space-time trade off. The number of memory reference increase by the depth of tree.