Initials: ________ Login: ________

Total Score: ________ (to be filled out by grader)

1) Scheduling (25 pts.)

____ pts. of 4 (Graded by: ____)

a) What is "**starvation**"? (i.e., define the term) What is one example of a scheduling algorithm that may suffer from starvation?

+1 mention of process being denied CPU time

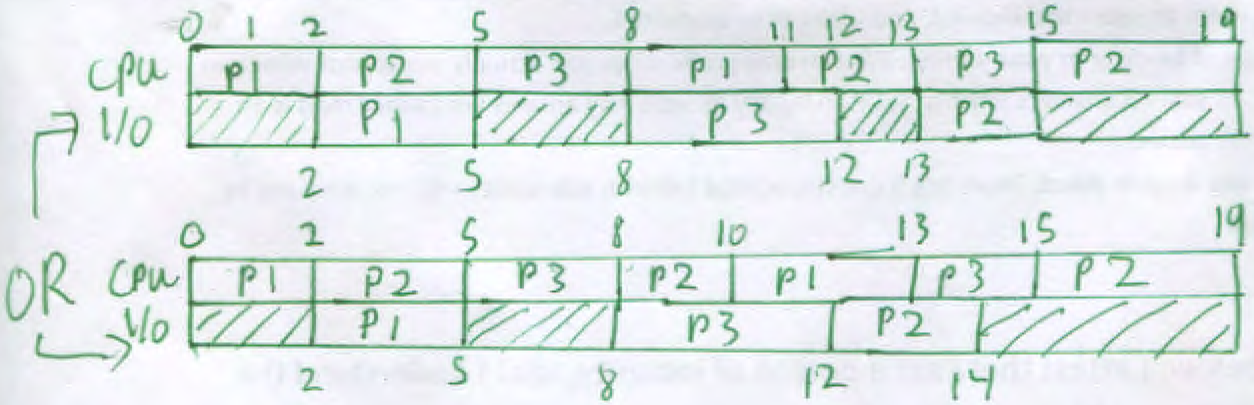
+1 mention of cause being the scheduling algorithm

+2 correct example (SJF, SRTF, Priority)

____ pts. of 10 (Graded by: ____)

b) Given the following processes that have the stated arrival times, run the stated CPU burst, followed by the stated I/O burst, and finally the stated second CPU burst, draw the CPU and I/O timelines for a **round robin** scheduler with time slice (quantum) **3 ms**.

Process	Priority	Arrival Time	CPU Burst (#1)	I/O Burst	CPU Burst (#2)
P1	1	t = 0 ms	2 ms	3 ms	3 ms
P2	2	t = 2 ms	5 ms	2 ms	4 ms
P3	3	t = 3 ms	3 ms	4 ms	2 ms



If no preemption, −10

else −1 for each incorrectly scheduled block

−1 for preempting in I/O queue

____ pts. of 4 (Graded by: _____)

c) Given the following CPU and I/O timelines, showing processes with the stated arrival times, then run the stated CPU burst, followed by the stated I/O burst, and finally the stated second CPU burst, which scheduling algorithm was used? If applicable, also state what the quantum/time slice value is.

Process	Arrival Time	CPU Burst (#1)	I/O Burst	CPU Burst (#2)
P1	t = 0 ms	5 ms	1 ms	5 ms
P2	t = 0 ms	2 ms	2 ms	2 ms
P3	t = 0 ms	4 ms	4 ms	2 ms

CPU	P2		P3				P2		P1					P3		P1				
I/O	▨	▨	P2		▨	▨	P3				▨	▨	▨	P1	▨	▨	▨	▨	▨	▨
t =	0		2		4		6		8		10			13	14	15				

+4 Shortest Job First (SJF)
(SRTF w/ quantum ≥ 5 ms = full credit)
(SRTF w/ quantum < 5 ms, or no quantum = half credit)
(SJF, but specified a quantum = half credit)

____ pts. of 3 (Graded by: _____)

d) Give two examples of scheduling algorithms that could benefit from **aging**.

+1.5 per example
- SJF
- SRTF
- Priority
- FCFS

____ pts. of 4 (Graded by: _____)

e) What is one advantage and one disadvantage of the Shortest Remaining Time First (SRTF) algorithm?

+2 for an advantage:
- no convoy effect
- lower average wait time (vs. FCFS)

+2 for a disadvantage:
- starvation

2) Virtual Memory (Paging & Segmentation) (30 pts.)

____ pts. of 3 (Graded by: _____)

a) Suppose a system has 1024 entries in each page table and 23-bit virtual addresses. How big is each page?

+1 VPN size = $\log_2(1024) = 10$ bits

+1 offset bits = addr. size − VPN size = $23 - 10 = 13$ bits

+1 page size = $2^{\text{offset-bits}} = 2^{13}$ Bytes = 2^3 KB = $\boxed{8\text{ KB}}$

($-\frac{1}{2}$ for simple math errors or wrong units)

____ pts. of 2 (Graded by: _____)

b) Suppose the same system has 65,536 (that's 64*1024) physical frames. How many total bits must the physical address have?

+1 PFN size = $\log_2(65{,}536) = 16$ bits

+1 addr. size = PFN size + offset bits = $16 + 13 = \boxed{29\text{ bits}}$

($-\frac{1}{2}$ for simple math errors)

____ pts. of 2 (Graded by: _____)

c) What term means that the processor spends more time swapping pages into and out of memory than it does running user processes?

+2 Thrashing

____ pts. of 7 (Graded by: _____)

d) Five of the following seven operations take place upon a page fault when there is no free frame in memory. Put the five correct operations in the right sequence.

1. update the page table of faulting process and frame table to reflect the changed mapping for the victim frame
2. use the frame table to find the process that owns the faulting page
3. look up if the faulting page is currently in physical memory
4. using the disk map of faulting process, load the faulting page from the disk into the victim frame
5. using the disk map of the victim process, copy the victim page to the disk (if dirty)
6. select a victim page for replacement (and the associated victim frame)
7. look up the frame table to identify the victim process and invalidate the page table entry of the victim page in the victim page table

$\boxed{6, 7, 5, 4, 1}$

+1 each correctly chosen number

+2 putting them in correct order (+1 partial credit if only one is out of order)

(Note: 1 may precede 4 or 5, but otherwise they must be in order)

____ pts. of 8 (Graded by: _____)

e) Assume physical memory has 4 frames, and each frame initially contains a page whose VPN equals the PFN (i.e., page 0 in frame 0, page 1 in frame 1, etc.). If page accesses occur in the following order, and an *exact* **Least Recently Used (LRU)** replacement policy is used, show which pages will be in which frames at the end of the sequence of memory accesses:

0, 1, 2, 3, 4, 1, 4, 5, 3, 6, 5, 1, 7, 3

PFN	VPN				final VPN
0	0	4	1		1
1	1	6	3		3
2	2	5			5
3	3	7			7

+2 each (must be in correct position)

+1 partial credit for correct pg. in wrong frame

OR

+1 partial credit if showed work and frame has correct contents except for one mistake

____ pts. of 2 (Graded by: _____)

f) In the above example (f), how much **external fragmentation** is there at the end?

+2 None
(or Zero)

____ pts. of 4 (Graded by: _____)

g) What is one advantage and one disadvantage of using segmented memory?

+2 for an advantage:
- no internal fragmentation
- program structure / organization

+2 for a disadvantage:
- experiences external fragmentation

____ pts. of 2 (Graded by: _____)

h) During the time interval t1 – t2, the following virtual page accesses are recorded for process P1. The page size is 1 KB. What is the working set size for process P1?

$\underbrace{0, 1, 2, 3, 4, 5,}_{\text{6 unique pages}}$ 0, 1, 2, 1, 2, 3, 4

+2 ∴ $\boxed{\text{6 pages}}$ or 6·1 kB = 6 KB

3) Caching (10 pts.)

____ pts. of 4 (Graded by: _____)

a) Consider the following C code:

```
int my_array[SIZE];

for (int i = 0; i < SIZE; ++i) {

    if (i < 2) {

        my_array[i] = I;

    } else {

        my_array[i] = my_array[i-1] + my_array[i-2];

    }

}
```

(i) What type of locality does the variable i exhibit?

+2 Temporal Locality

(ii) What type of locality does the array my_array exhibit?

+2 Spatial Locality

____ pts. of 6 (Graded by: _____)

b) Suppose a byte addressable memory system has a total of 8 GB of main memory. How many tag bits will a direct mapped cache with data size 256 KB (i.e., 256 KB is how much data from main memory it can hold), and block size 16 bytes need in order to support this memory?

+1 addr size $= \log_2(8\text{ GB}) = \log_2(8 \cdot 1024 \cdot 1024 \cdot 1024) = 33$ bits

*{ +1 offset bits $= \log_2(16) = 4$ bits

+2 index bits $= \log_2(\text{num lines}) = \log_2\left(\frac{256 \cdot 1024}{16}\right) = \log_2(16 \cdot 1024) = 14$ bits }

+1 (equation) tag bits = addr size − index bits − offset bits

+1 (final answer) $= 33 - 14 - 4 = \boxed{15 \text{ bits}}$

*Alternatively, may have found in one step index + offset bits $= \log_2(256\text{ KB})$ $= \underline{18 \text{ bits}}$

Points This Page:

4) Review (cumulative material) (10 pts.)

____ pts. of 2 (Graded by: _____)

a) With a dual-bus architecture that has a dual-ported register file (DPRF), how many clock cycles (minimally) are required to load the contents of two different registers into the A and B registers of the ALU?

+2 1

____ pts. of 2 (Graded by: _____)

b) Which type of program discontinuity is considered "external"?

+2 Interrupt

____ pts. of 2 (Graded by: _____)

c) What is an example of something that would cause an interrupt?

+2 for any of:
- input arrives on a device
- mouse click
- key press (on keyboard)
- network packet arrives

____ pts. of 4 (Graded by: _____)

d) What does it mean for a register to be "callee saved"?

The procedure that gets called (the "callee")
+2 must save that register
+1 if (and before) it decides to use the register
+1 and restore the saved register before returning

5) Pipelined Architecture (25 pts.)

____ pts. of 5 (Graded by: _____)

a) Explain the difference between *latency* and *throughput* in the context of a pipelined processor implementation. What are the metrics used for each (i.e., in what units are they measured)?

+2 for the first correct
+1 for each additional

• Latency is the time required to execute each instruction
• Latency is measured in average cycles per instruction (CPI)
• Throughput is how many instructions can be done in a given amount of time
• Throughput is measured in average instructions per cycle (IPC)

____ pts. of 3 (Graded by: _____)

b) Why is "flushing" necessary in order to benefit from branch prediction?

+1 • if the predicted branch is wrong,
+2 { • the partially completed instructions must not be allowed to complete (or must be aborted, etc.)

____ pts. of 2 (Graded by: _____)

c) "Read after write (RAW)" and "Write after read (WAR)" are examples of which type of hazard?

+2 Data Hazard

____ pts. of 2 (Graded by: ____)

d) Which type of hazard is created if two different pipeline stages need to use and ALU, but there is only one ALU in the processor?

+2 Structural Hazard

____ pts. of 3 (Graded by: ____)

e) Why is it important that each stage of the processor pipeline take the same number of cycles as the other stages?

+3 The slowest stage is a bottleneck that limits the speed of all stages

(or no stage can move on until the slowest completes or something similar about moving in lockstep)

____ pts. of 10 (Graded by: ____)

Note: This is probably the hardest problem on the test. We suggest you come to this at the very end after you have attempted the entire test.

f) Given the following 5 stage pipeline specifications, fill in the waterfall diagram for the code fragment shown below:

- There is data forwarding from **EX to ID/RR** and from **MEM to ID/RR**
- Write back data is available on the **next clock cycle**
    - ...meaning that ID/RR cannot read a register in the same clock cycle in which WB writes the same register (reading a different register is still OK, though)
- There is a static branch predictor that always predicts a branch as **not taken**

Assume all registers are initialized to zero at the beginning. You can stop filling in the diagram once each instruction passes through the write back stage at least once.

(continued on next page)

```
        addi  R1, R2, 0x10      ! I1
Loop:   beq   R1, R3, End       ! I2
        LW    R2, 0(R1)         ! I3
        SW    R1, 0(R2)         ! I4
        addi  R4, R2, 0x42      ! I5
        addi  R3, R3, 0x1       ! I6
        beq   R3, R3, Loop      ! I7
End:    I8
        I9
```

-3 Missing LW stall

-2 Missing WB stall

-2 For each ~~extra~~ extra NOP

-3 Not flushing

-3 Not fetching I2 after the second loop

Cycle	IF	ID/RR	EX	MEM	WB	
1	I1					
2	I2	I1				
3	I3	I2	I1			
4	I4	I3	I2	I1		
5	I5	I4	I3	I2	I1	
6	I5	I4	NOP	I3	I2	* NOP here
7	I6	I5	I4	NOP	I3	
8	I6	I5	NOP	I4	NOP	* NOP here
9	I7	I6	I5	NOP	I4	
10	I8	I7	I6	I5	NOP	
11	I9	I8	I7	I6	I5	
12	I2	NOP	NOP	I7	I6	* flush here
13	I3	I2	NOP	NOP	I7	
14						
15						
16						