CS 2200 Fall 2013 Test 2

Prism ID:____________

Name:__________Kishore________ GTID#: 9_____________

Problem	Points	Lost	Gained	Running Total	TA
1	1				
2	15				
3	10				
4	14				
5	10				
6	10				
7	20				
8	20				
Total	100				

- **You may ask for clarification but you are ultimately responsible for the answer you write on the paper.**
- **Illegible answers are wrong answers.**
- **Please look through the entire test before starting. WE MEAN IT!!!**

Illegible answers are wrong answers.
Good luck!

1. (1 point, 0 min) (circle one; you get a point regardless of correct/incorrect answer)
"They have the attention span of slightly moronic woodland creatures."
(a) Said President Obama about Republicans
(b) Said Linux creator Linus Torvalds about code developers
(c) Said German Chancellor Angela Merkel about the NSA
(d) Said the President of Georgia Tech about UGA students

Pipelining

CS 2200 Fall 2013 Test 2

Prism ID:____________

Name:__________Kishore_______ GTID#: 9_____________

2. (15 points, 10 min)

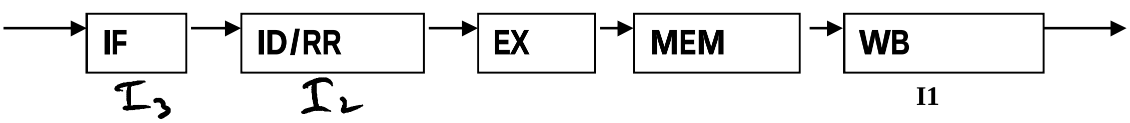
(a) (2 points)Structural hazard in a pipeline occurs due to (circle the correct choice)

(i) Branch instructions in the program
(ii) Load instructions in the program
(iii) Data dependencies in the program
(iv) Hardware limitations in the datapath

(b) Consider the following three consecutive instructions in a program in flight in a pipelined processor with **register forwarding**:

I1: R1 ← R2 + R3
I2: R1 ← R3 + R4
I3: R5 ← R1 + R7

The state of the pipeline is:



(i) (2 points) At what stage of the pipeline are **I2** and **I3?**

I2 in ID/RR **(+1)**
I3 in IF **(+1)**

(ii) (3 points) Why are **I2 and I3** where you have shown them to be?

I2 notices that the B bit is set for R1 indicating a WAW hazard. So it waits for that condition to go away in ID/RR stage. **(+2)**

I3 is waiting behind I2 in IF **(+1)**

CS 2200 Fall 2013 Test 2

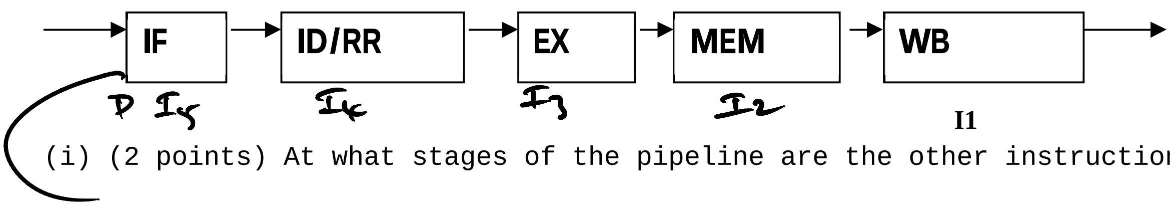
Prism ID:____________

Name:_________Kishore_______ GTID#: 9____________

(c) Consider the following instructions in flight in a pipelined processor **with register forwarding**:

```
I1:  R0 ←—— R2 + R3
I2:  R3 ←—— R0 + R1
I3:  R7 ←—— R3 + R9
I4:  R8 ←—— R0 + R7
I5:  R7 ←—— R6 + R8
```

The partial state of the pipeline is:



(i) (2 points) At what stages of the pipeline are the other instructions?

(+0.5 for each)

(ii) (6 points) Recall that associated with each element of the register file are a busy bit (B) and a read pending signal (RP) as shown below. **Fill in the state of the B and RP bits** in the register file given the state of the pipeline as you have it in your answer for part (i).

(Note on the answer key: RP is a signal that is asserted ONLY by the instruction in ID/RR stage; Therefore, AT MOST 2 registers will have the RP signal asserted).

I4 needs to read R0 and R7 and both are being written by earlier instructions that are in flight in the pipeline. So RP for R0 and R7 will be 1.

-1 for each incorrect entry

		B	RP
R0		1	1
R1			
R2			
R3		1	
R4			
R5			
R6			
R7		1	1
R8			
R9			

'0' everywhere else

CS 2200 Fall 2013 Test 2

Prism ID:____________

Name:__________Kishore________ GTID#: 9____________

3. (10 points, 7 min)

(a) Given the following sequence of instructions:

```
        BEQ       L1
        ADD
        LW
        ...
L1      NAND
        SW
```

The processor uses **branch prediction** assuming the **sequential path to be the probable one.**

In the actual execution of the above code snippet, **the prediction turns out to false.**

(i) (8 points) Fill in the "waterfall diagram" below showing the progress of the above instructions through the pipeline stages.

Cycle Number	IF	ID/RR	EX	MEM	WB
1	BEQ	-	-	-	-
2	ADD	BEQ	-	-	-
3	LW	ADD	BEQ	-	-
4	NAND	NOP	NOP	BEQ	-
5	SW	NAND	NOP	NOP	BEQ
6	-	SW	NAND	NOP	NOP
7	-	-	SW	NAND	NOP
8			-	SW	NAND
9				-	SW

(+1 for each correct diagonal for BEQ, NAND, SW)
(+2 for correct diagonal for ADD and LW)
(+1 if the whole thing finished in 9 cycles)

(ii) (2 points)What is the observed CPI for the instructions that actually get executed (BEQ, NAND, SW)?

Observed CPI = 9/3 = 3 **(+2)**

(all or nothing)

CS 2200 Fall 2013 Test 2

Prism ID:____________

Name:_________Kishore_______ GTID#: 9____________

Process Scheduling

4. (14 points, 5 min)

(a) (2 points) (**Select one correct choice**)

One of the following is **NOT** part of the state of a running program

1. General Purpose Registers that are visible to the instruction set
2. Program counter and the register that represents the stack pointer
3. Layout of the program in memory
4. Priority information
5. Internal registers in the datapath of the processor

(b) (4 points)

What is the difference between a "scheduler" and a "dispatcher"?

Scheduler picks the next process to run commensurate with the scheduling policy (FCFS, RR, etc.) **(+2)**

Dispatcher loads the processor registers (GPRs, PC, SP, PTBR) with the "state" of the chosen process (by the scheduler) from the PCB. **(+2)**

(c) (2 points) (**Select one correct choice**)

A preemptive scheduling algorithm requires

1. A trap instruction
2. An external interrupt
3. The currently running process to terminate
4. The currently running process to make an I/O request

(c) (2 points) (**Select one correct choice**)

Upon context switch, the scheduler saves the volatile state of the current process in

1. The system stack
2. The PCB for that process
3. The user stack
4. The heap space of the process

(d) (2 points) (**Select one correct choice**)

I want to know how good the system is for executing my gaming application. The metric that will help me determine this best is:

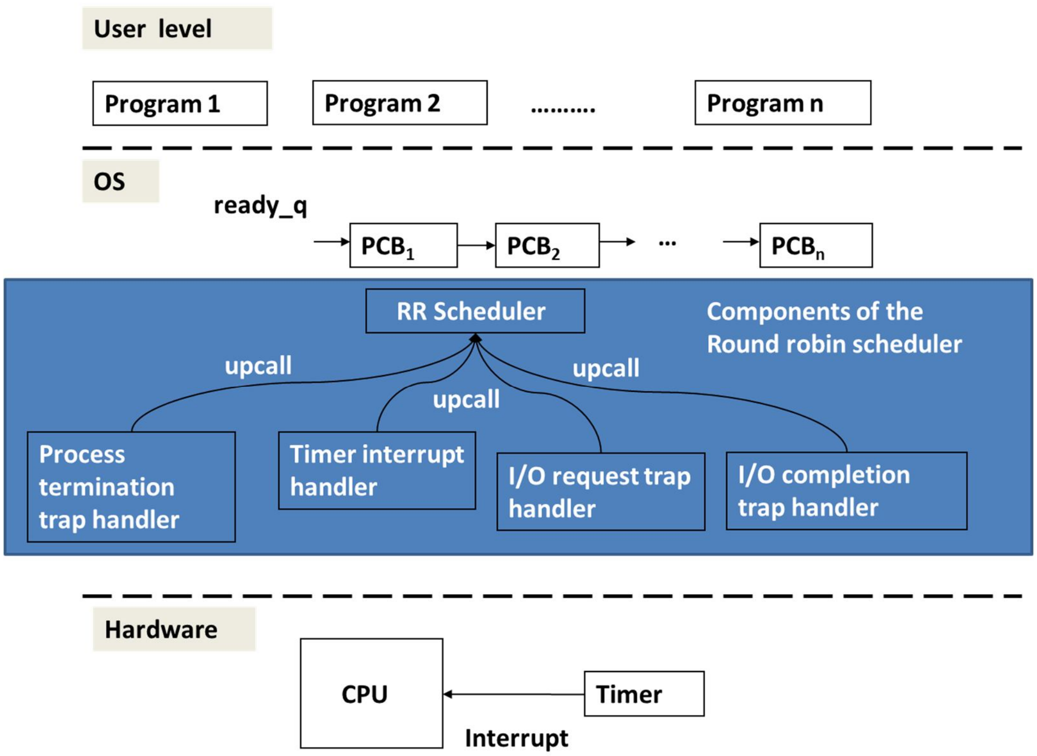
1. Throughput
2. Average waiting time
3. Average turnaround time
4. CPU Utilization
5. Response time

(e) (2 points) (**Select one correct choice**)

One of the following attributes is common to SJF and priority schedulers:

1. They are both fair
2. Both of them could lead to "starvation"
3. They both favor short running processes
4. They both lead to the best average waiting time for processes
5. They both suffer from convoy effect

5. (10 points, 5 min)
Given the picture of the RR scheduler:



Fill in functionally the work to be done in each of the components of the RR scheduling framework (your work on the next page).

The level of detail expected for the “work to be done” is (“get head of the ready_q”; “dispatch selected PCB on the processor”; “upcall to RR Scheduler”; “save context of running process in PCB”; etc.).

CS 2200 Fall 2013 Test 2

Prism ID:____________

Name:_________Kishore_______ GTID#: 9____________

Scheduler:
get head of ready queue;
set timer;
dispatch;

(+2)

Timer interrupt handler:
save context in PCB;
upcall to scheduler;

(+2)

I/O request trap:
save context in PCB;
move PCB to I/O queue;
upcall to scheduler;

(+2)

I/O completion interrupt handler:
save context in PCB;
move PCB of I/O completed process to ready queue;
upcall to scheduler;

(+2)

Process termination trap handler:
Free PCB;
upcall to scheduler;

(+2)

(-0.5 for each missed “work to be done”)

CS 2200 Fall 2013 Test 2

Prism ID:____________

Name:_________Kishore_______ GTID#: 9____________

Memory Management and Virtual Memory (Note: K = 1024)

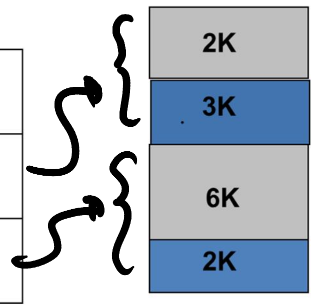
6. (10 points, 5 min)

(a) Consider the following allocation table with fixed-size partition memory allocation.

Allocation table

Occupied bit	Partition Size	Process
1	5K	P2 (need 2K)
1	8K	P1 (need 6K)

Memory



(i) (2 points) How much is the internal fragmentation?

5 K **(+2)**

(ii) (2 points) How much is the external fragmentation?

0 **(+2)**

(b) (3 points)
Virtual address is 32 bits; pagesize 4 Kbytes; How many entries are there in the page table?

Number of bits in page offset = $\log_2 4K$ = 12 **(+1)**
Number of bits in VPN = (VA size – page offset size)= 20 **(+1)**
Number of entries in page table = 2^{VPN} = 2^{20} **(+1)**

(c) (3 points)
For the same memory system as in (b), the physical address is 24 bits. How many physical page frames does the memory system have?

Number of bits in PFN = (PA size – page offset size)= 12 **(+2)**
Number of physical frames = 2^{PFN} = 2^{12} **(+1)**

CS 2200 Fall 2013 Test 2

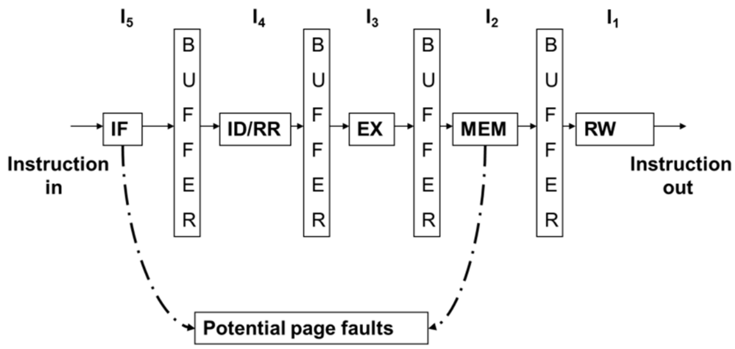
Prism ID:____________

Name:__________Kishore________ GTID#: 9_____________

Demand paging, Working set, page replacement

7. (20 points, 15 min)

(a) Consider a 5-stage piplelined processor, which uses demand-paged virtual memory management. **Instruction I2 incurs a page fault.**



(i) (2 points) What will happen to the instructions in flight in the processor?

I1 allowed to complete **(+1)**
I3, I4, and I5 squashed **(+1)**

(ii) (2 points) At which instruction will the program be resumed?

The program will resume execution from I2. **(+2)**

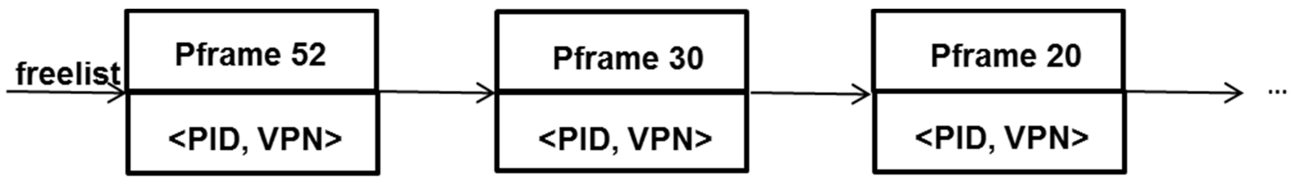
(iii) (2 points) What additional information is needed to be carried along in the buffers between the stages for demand paging to work in a pipelined processor?

The PC value corresponding to every instruction has to be sent along from buffer to buffer. That's the way the OS will know where to restart the process that incurred the page fault. **(+2)**

(b) (2 points)
In page-based virtual memory system, what information is in the PCB that identifies to the OS the memory space occupied by the process?

PTBR (The page table base register) field. This points to the memory region where the PT for a given process is kept by the OS. **(+2)**

(c) (3 points)
The freelist (as shown below) is a data structure of the memory manager that contains the pool of page frames that are available for allocation to satisfy page faults.



Each entry in the freelist, shows the page frame number that is available for allocation as well as the reverse mapping, i.e., the process-ID and the VPN of that process that was hosted in this page frame. What is the purpose of this reverse mapping for a page frame that is in the freelist?

The freelist is a convenience for the memory manager to handle a page fault immediately (without having to run a page replacement algorithm to secure a free physical frame). However, the pframes in the freelist have not yet been allocated to any other process and therefore they contain the contents of the virtual pages of the processes that previously had them as part of their memory footprint in the physical memory. In the above list, let's say pframe 52 used to belong to <PID=P2, VPN=48>. Suppose the next time P2 is run on the CPU, it page faults on VPN = 48. In that case, there is no need to do I/O; simply remap pframe 52 in the page table for P2 at VPN = 48. Thus, the purpose of the reverse mapping is an optimization to avoid I/O (which is expensive) if the missing virtual page for a faulting process is already present in the freelist of pframes. **(+3 for a good explanation)**

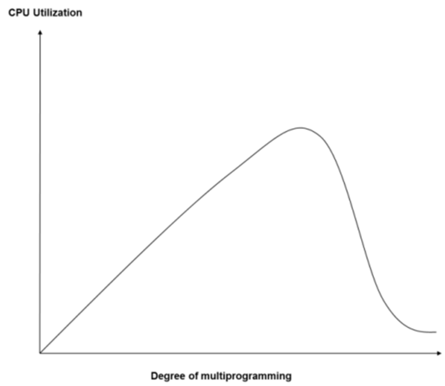
(d) (3 points)
Why is it infeasible to implement a true LRU algorithm for page replacement in the operating system?

To implement True LRU, we need the order of memory accesses. A way to do this is to use a push down stack which contains at the top of the stack the MRU page frame, and at the bottom the LRU page frame. On every memory access this stack has to be updated to bring the referred page frame to the top. Who has to do this? Of course, the hardware since you will be unhappy if the OS gets in the middle of every one of your loads and stores! How big is this stack? As big as the number of page frames! This makes it infeasible to implement in hardware, which is why true LRU is infeasible.
(+3 for a good explanation)

(e) (2 points)
What is the difference between a simple FIFO page replacement algorithm and the “second chance” page replacement scheme(aka “clock” algorithm)?

In Clock the FIFO candidate gets a second chance (meaning it will not be chosen as a victim) if its reference bit is set. **(+2)**

(f)



(i) (2 points) In the picture above, why does the CPU utilization increase as we increase the degree of multiprogramming?

Normally, since processes flip between computing and I/O activities, having more processes ready to go in physical memory allows the OS to keep the processor utilization high. This is the reason why as we increase the degree of multiprogramming the utilization goes up (for a while) **(+2)**

(ii) (2 points) Why does CPU utilization start decreasing beyond a certain point?

However, note that if a process does not have its working set in memory it will page fault to bring it in. When the degree multiprogramming exceeds a threshold, none of the processes have their working set in memory, so they are constantly page faulting…this is "implicit I/O" and so the processor utilization starts dropping. This phenomenon is called "thashing." **(+2)**

TLB and Processor Cache (Note: K = 1024)

8. (20 points, 10 min)

(a) (4 points)

A TLB is a small table implemented in hardware to speed up memory access in a virtual memory system. Explain the principle behind the concept of a TLB.

Spatial locality: **(+1)**

- Sequentiality of instructions in a program
- Sequentiality of complex data structures such as arrays and structs

Temporal locality: **(+1)**

- Behavior of programs that make them re-visit the same instructions and or data structures repeatedly over time.

Stash away recent translations from the page table (in physical memory) in a small high speed "memory" (similar to registers in terms of hardware implementation) close the processor. **(+1)**

Subsequent translations become faster since we don't have to go to physical memory (i.e., page table) to do the translation. **(+1)**

(b) (4 points)

CS 2200 Fall 2013 Test 2

Prism ID:____________

Name:__________Kishore________ GTID#: 9_____________

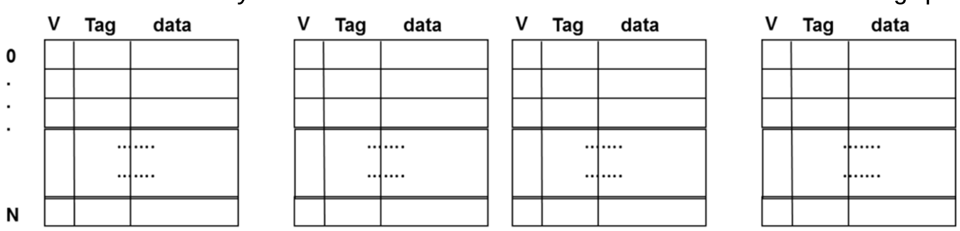
What is motivation for implementing the cache closest to the CPU as a “virtually indexed and physically tagged” cache?

- Since the caches use physical address, we have to do address translation to convert the virtual address to physical. **(+1)**
- This brings the TLB lookup into the critical path for cache access and increases the latency and therefore increases the CPU cycle time. **(+1)**
- By using the bits of the virtual address that remains unchanged in the translation as “index” into the cache, we can do the address translation (i.e., TLB lookup) **in parallel** with the cache lookup. **(+2)**

(c) (2 points)
Why are the caches not implemented using the virtual address for both index and tag?

We will have to flush out the cache on every context switch. **(+2)**

(d)
Consider a 4-way set associative cache with the following parameters:



- Cache size (i.e, the amount of actual data it can hold) of **128 Kbytes**
- **32-bit byte-addressable** memory
- Each memory word contains **4 bytes**
- cache block size is **16 bytes**.
- **write-through** policy.
- **one valid bit** per block.

(i) (4 points) How big is **N** in the above picture (show your work)?

Number of blocks in the cache = cache size / blocksize = 128K/16 bytes = 8K **(+2)**

N the number of cache lines = number of blocks/associativity = 8K/4 = 2048 **(+2)**

CS 2200 Fall 2013 Test 2

Prism ID:____________

Name:_________Kishore_______ GTID#: 9____________

(ii) (6 points) The 32-bit memory address is split into block offset, tag, and index as shown below:

Cache Tag	Cache Index	Block Offset
t	**n**	**b**

What are the values of **t**, **n**, and **b** (show your work)?

b = $\log_2$blocksize = 4 bits **(+2)**

n = $\log_2$N = 11 bits **(+2)**

t = size of address – (b + n) = 17 bits **(+2)**