CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

Problem	Points	Lost	Gained	Running Total	TA
1	1				
2	10				
3	13				
4	15				
5	10				
6	20				
7	16				
8	15				
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 do not discuss this test by any means (until 5 pm today)**
- **Please look through the entire test before starting. WE MEAN IT!!!**

Illegible answers are wrong answers.
Good luck!

1. (1 point, 0 min) (circle the correct choice)
If the presidential candidates had been tied for the electoral votes, one of the following could have happened under the provisions of the US constitution:

a) Barack Obama, President & Mitt Romney, Vice President
b) Mitt Romney, President & Barack Obama, Vice President
c) Mitt Romney, President & Joe Biden, Vice President
d) Barack Obama, President & Paul Ryan, Vice President
e) None of the above
f) "I'm tired of Bronco Bamma and Mitt Romney"
(http://www.youtube.com/watch?feature=player_embedded&v=OjrthOPLAKM)

CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

Pipelining

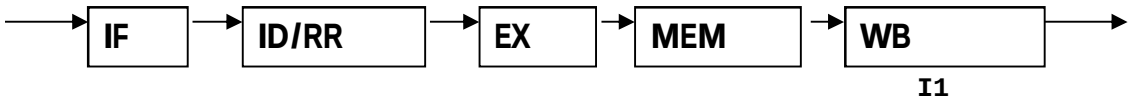
2. (10 points, 5 min)

(a) (2 points)

Consider the following two instructions (I2 immediately follows I1 in the original program) in flight in a pipelined processor **with register forwarding**:

I1: R1 ⟵ Memory[R2+offset]; load R1 with contents of memory at R2+offset
I2: R4 ⟵ R1 + R5

The state of the pipeline is:



At what stage of the pipeline is I2?
I2 is in EX stage (load instruction I1 can forward only in MEM stage) (+2)

(b) (4 points)

Consider the following two instructions (I2 immediately follows I1 in the original program) in flight in a pipelined processor **with register forwarding**:

I1: R1 ⟵ R2 + R3
I2: R1 ⟵ R4 + R5
I3: R6 ⟵ R1 + R2

The state of the pipeline:

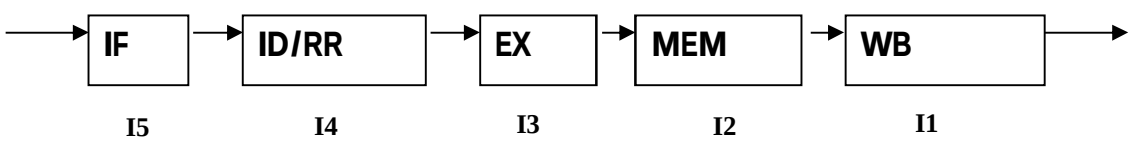
At what stages of the pipeline are I2 and I3?
I2 is in ID/RR (WAW conflict with I1) (+2)
I3 is in IF (+2)
NOPs in EX and MEM stages

(c) (4 points)

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

I1: R1 ⟵ R2 + R3
I2: R4 ⟵ R5 + R6
I3: R7 ⟵ R8 + R9
I4: R6 ⟵ R1 + R4
I5: R3 ⟵ R6 + R7

The state of the pipeline is:



CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

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 above state of the pipeline.

-0.5 for each incorrect entry

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

'0' in all other B + RP spots for all registers

3. (13 points, 5 min)

(a) (2 points)
With conservative handling of branches in a 5-stage pipeline discussed in the class (i.e., stop new instructions from entering the pipeline until the outcome of the branch is known), there will be **at most (circle the right choice):**

1. 1 bubble in the pipeline
2. 2 bubbles in the pipeline **(+2)**
3. 3 bubbles in the pipeline
4. 0 bubbles in the pipeline

(b) (4 points)
If a pipelined processor supports a 1-slot delayed branch (i.e., the instruction immediately following the branch instruction will ALWAYS be executed), the compiler has to do two things to ensure **correct execution** of the user programs and **no performance loss** due to branches:

1. **add NOP instructions after each branch instruction for correct execution** **(+2)**

2. **try and replace NOP instructions that it compiled in by useful instructions that are unrelated to the outcome of the branches in the delay slots** **(+2)**

CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

(c) (2 points)
What purpose does the Brach target buffer (BTB) perform in a pipelined processor?

Helps remember the outcomes of the branch instructions encountered during execution so that it can predict the outcomes in future visits to the same branch instructions.

(+2, all or nothing)

(d) (3 points)
Given the following code fragment:

```
PC     Instruction
1000   BEQ R1, R2, 1200; if R1 = R2 go to 1200
……
1500   BEQ R1, R2, 1800;
……
2000   BEQ R1, R2, 5000;
```

When the code is executed the following happens:

- branch at PC=1000 is taken
- branch at PC=1500 is taken
- branch at PC=2000 is not taken

Show the contents of the BTB

PC of branch	target of branch	taken/not
1000	1200	T
1500	1800	T
2000	5000	N

(-1 for each incorrect row; -2 for missing column)

(e) (2 points)
In a pipelined processor, one of the following will be an acceptable course of action to deal with external interrupts **(Select one correct choice)**:

1. The currently running process is terminated.
2. The processor stops sending new instructions into the pipeline, sending NOPs through the pipeline instead, allows the current instructions in flight to complete, and then enters the INT state. **(+2)**
3. The processor immediately terminates all in-flight instructions and enters into the INT state.
4. The processor ignores the interrupt until the current process completes execution.

CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

Process Scheduling

4. (15 points, 5 min)

(a) (2 points) (**Select one correct choice**)
A process differs from a program in that

1. Process is a machine language representation of a program.
2. Process is a program in execution. **(+2)**
3. Process is a program that executes correctly.
4. Process is a special kind of program that is part of the operating system.
5. There is no difference.

(b) (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 **(+2)**

(c) (3 points) (**Answer True/False with justification**)
It is impossible to implement a preemptive scheduling algorithm without timer interrupts.

False. Preemption can be added to any scheduling algorithm. Essentially, the "priority" per the specific scheduling algorithm needs to be re-evaluated anytime a process joins the ready queue to decide whether or not to pre-empt the currently running process. For e.g., with FCFS, if the process rejoining the ready queue has an earlier arrival time (this is the "priority" criterion for FCFS) than the currently running process, then pre-empt the currently running process.

(+1 for "False"; +2 for good explanation)

(d) (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 **(+2)**
3. The user stack
4. The heap space of the process

(e) (3 points)
Explain the "convoy" effect with FCFS schedule with an example.

In FCFS scheduling discipline, it is possible for several processes with short CPU bursts to get "stuck" behind an earlier arriving long running process. Thus the shorter jobs will appear like a military convoy behind the long running process as they move from the CPU queue to the I/O queue for their entire execution in the system. (All or nothing)

CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

(f) (3 points)
Explain the "starvation" effect in SJF with an example.

If the current job mix has a number of processes with short CPU bursts, then a lone long running process may continually be denied time on the CPU since there may always be a shorter job to be scheduled on the processor. This is the "starvation" effect with SJF schedule.
(All or nothing)

5. (10 points, 5 min)
Answer the following questions with respect to the **Linux** scheduler.

(a) (**Answer True/False with justification**) The scheduling priority for a task remains unchanged for its lifetime in Linux.

False. The scheduler takes a "carrot" and "stick" approach to promote processes that do short CPU bursts between I/O calls (a characteristic of interactive jobs) to higher priority levels; and to demote processes that do long CPU bursts between I/O calls (non-interactive jobs) to lower priority levels.
(+1 for "False"; +1 for good explanation)

(b) What happens when the time quantum for the current task expires?

After saving the TCB, the task's TCB is placed in the "Expired" array at the right priority level. (+1)

Next highest prioity task from the "Active" array scheduled on the processor. (+1)

(c) What happens when the current task makes a blocking I/O call?

After saving the TCB and noting its remaining time quantum, the task's TCB is put aside awaiting I/O completion, (+1)

and the scheduler picks the next highest runnable task from the "Active" array to schedule on the CPU. (+1)

(d) What happens when a blocking I/O finishes for a task?

The task's TCB is placed in the "Active" array at the right priority level (with its remaining unused time quantum at the point of the I/O call).

(+2)

(e) What happens when the scheduler runs out of tasks to run in the "Active" array?

Simply flip the "Active" and "Expired" array pointers and continue with the scheduling algorithm.
(+2)

CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

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

6. (20 points, 10 min)

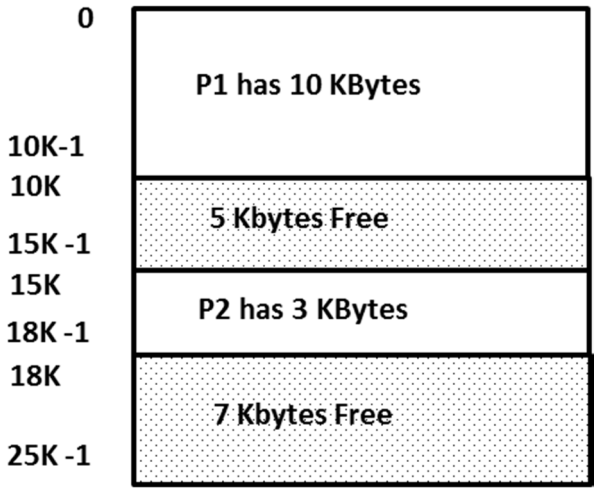
(a) (2 points) (**Select one correct choice**)
The **minimal** additional hardware support needed for dynamic relocation in LC-2200 is

1. Fence register
2. Bounds registers
3. Base plus limit registers **(+2)**
4. LC-2200 is cool as is

(b) (2 points) (**Select one correct choice**)
With page-based memory management scheme there can be

1. No external fragmentation **(+2)**
2. No internal fragmentation
3. No fragmentation
4. Both internal and external fragmentation

(c) (10 points)
Consider a **variable size partition** memory management scheme implemented for a byte-addressed machine. The total available memory is 25 Kbytes. The current allocation looks as follows:



(i) Show the contents of the allocation table that reflects the above memory allocation:

Start address	Size	Process
0	10K	P1
10K	5K	FREE
15K	3K	P2
18K	7K	FREE

(-1 for each incorrect row)

(-2 for each missing column)

CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

(ii) P2 completes execution and releases its allocated memory. Show the contents of the allocation table after P2 has completed its execution.

Start address	Size	Process
0	10K	P1
10K	15K	FREE

(-2 for each incorrect row; -2 for each missing column)

(d) (3 points)
Virtual address is 64 bits; pagesize 8 Kbytes; How many entries are there in the page table?

Number of bits in page offset = $\log_2$(pagesize) = $\log_2$(8K)= 13 (+1)
Number of bits to represent the VPN = 64-13 = 51 (+1)
Number of entries in the page table = $2^{VPN} = 2^{51}$ (+1)

(e) (3 points)
For the same memory system as in (d), the physical address is 40 bits. How many physical page frames does the memory system have?

Number of bits to represent the physical frame number (PFN)
= 40 – 13 = 27 (+2)
Number of physical frames = $2^{PFN} = 2^{27}$ (+1)

Demand paging, Working set, page replacement

7. (16 points, 10 min)
(a) (3 points)
The frame table is a data structure of the memory manager for reverse lookup, i.e., given a page frame, the frametable gives the <Process-ID, VPN> that is currently hosted in that page frame. What role does the frametable perform in the memory management?

If a victim page frame is currently in use by another process (say P2), then the memory manager has to first invalidate the page table entry for the particular VPN of that process (P2) that is currently mapped to this physical frame. To accomplish this, the memory manager needs to do a reverse lookup to know the <PID, VPN> that the victim PFN is currently hosting. The frametable is the data structure that facilitates this reverse lookup.

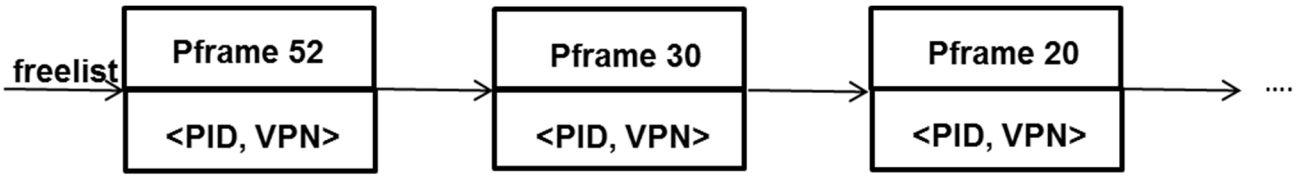
(+3 for good explanation)

CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

(b) (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)

(c) (2 points)(**Select one correct choice**)
The reference bit for supporting an approximate LRU page replacement scheme is implemented by associating

1. One bit per page table entry **(+2)**
2. One bit per memory location
3. One bit per physical page frame
4. One bit per process
5. One bit for the entire physical memory
6. One bit for the entire page table

(d) (2 points)
What is meant by the term "second chance" in the page replacement scheme that goes by that name (aka "clock" algorithm)?

If the page replacement algorithm chooses a PFN as a victim, then the memory manager consults the PTE that is currently using this pframe (by using a reverse lookup). If the reference bit in the PTE is set then this page has been recently referenced. In this case, the memory manager gives the associated virtual page (that this PTE pertains to) a "second chance" to remain mapped to the pframe that was a potential victim. It simply resets the reference bit for this PTE, and does NOT choose this pframe as a victim.
(-1 if ref-bit in PTE not mentioned)

CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

(e) (6 points)
During the time interval t1 - t2, the following virtual page accesses are recorded for the processes P1 and P2.

P1: 0, 0, 0, 23, 4, 0, 0, 4, 4, 2, 0
P2: 0, 0, 3, 1, 2, 0, 20, 30, 20

(i) What is the **working set** for P1 in this time interval?

WS_{P1} = {0, 2, 4, 23} **(+2)**
WSS_{P1} = 4

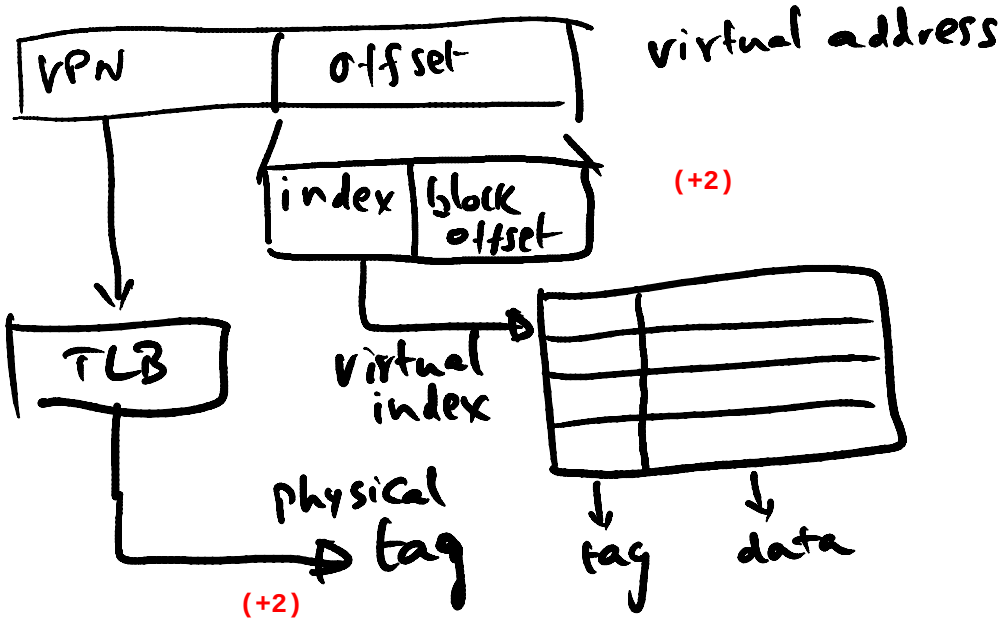
(ii) What is the working set for P2 in this time interval?

WS_{P2} = {0, 1, 2, 3, 20, 30} **(+2)**
WSS_{P2} = 6

(iii) What is the total **memory pressure** on the system during this interval?

The total memory pressure = WSS_{P1} + WSS_{P2} = 4 + 6 = 10 **(+2)**

TLB and Processor Cache (Note: K = 1024)
8. (15 points, 10 min)
(a) (4 points)
In a virtually indexed, physically tagged cache, using a figure
(i) explain how the index into the cache is generated
(ii) explain how the tag is generated (from the CPU's virtual address) for comparison with the tag contained in the cache.



CS 2200 Fall 2012 Test 2 Solution

Prism ID:_____

Name:_____Kishore_____ GTID#: 9_____

(b) (5 points)
A pipelined processor has an average CPI of 1.2 not considering memory effects. On an average each instruction has an I-cache miss of 0.9%, and a D-cache miss of 0.4%. The miss penalty is 100 cycles. What is the effective CPI taking into account memory stalls?

Memory stalls due to I-cache misses = 0.9/100 * 100 = 0.9 cycles **(+1)**
Memory stalls due to D-cache misses = 0.4/100 * 100 = 0.4 cycles **(+1)**

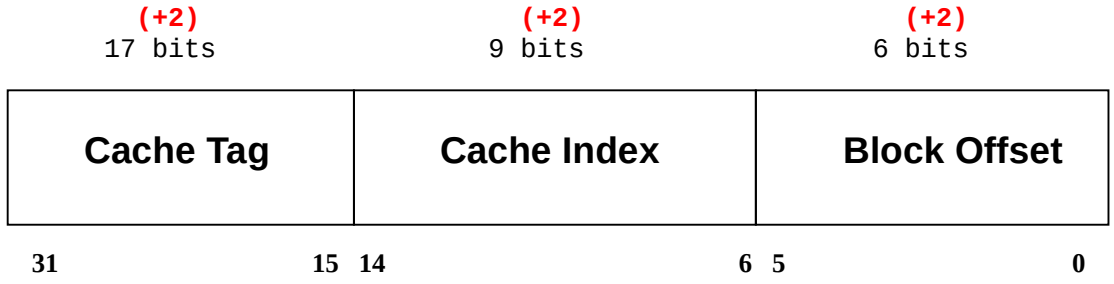
CPI with memory stalls = Base Average CPI + Memory stalls
= 1.2 + 0.9 + 0.4
= 2.5 **(+3)**

(c) (6 points)

Consider a direct mapped cache for a byte addressed processor.

- Data size of cache = 32 KB.
- CPU address = 32 bits
- Memory word = 4 bytes.
- Cache block size = 64 bytes.
- Write policy is Write-through at the granularity of individual words.
- Cache replacement policy = LRU

The memory address is interpreted as follows:



Fill in the blanks above to indicate how the memory address is interpreted for looking up the cache.