CS 2200 Spring 2018 Test 2A – soln

Name:_________________________ Email-id: ____________

Problem	(Points, Minutes)	Lost	Gained	TA
0	(1, 0min)			
1	(10, 5min)			
2	(15, 10min)			
3	(15, 10min)			
4	(16, 10min)			
5	(13, 10min)			
6	(13, 10min)			
7	(17, 15min)			
Total	(100, 70min)			

- **You may ask for clarification but you are ultimately responsible for the answer you write on the paper.**
- **Illegible answers are wrong answers.**
- **SHOW YOUR WORK FOR PARTIAL CREDIT.**
- **Answers to True/False questions WITHOUT any justification get no credit.**
- **Please do not discuss this test by any means.**
- **Please look through the entire test before starting. WE MEAN IT!!!**

Good luck!

0. (1 point, 0 min) (circle one; you get a point regardless of correct/incorrect answer)
Bhraamari asana is another name for
(a) Laugh yoga
(b) Super brain yoga
(c) Tree pose
[✓ (d) Bee sound while shutting out visual and auditory inputs]
(e) Don’t know what you are talking about

CS 2200 Spring 2018 Test 2A – soln

Name:__________________________ Email-id: _____________

Performance

1. (10 points, 5 min)

An architecture has three types of instructions that have the following CPI:

Type	CPI
A	2
B	7
C	3

An architect determines that he can reduce the CPI for B to 5, with no change to the CPIs of the other two instruction types, but with an increase in the clock speed of the processor. What is the maximum permissible increase in clock cycle time that will make this architectural change still worthwhile? Assume that all the workloads that execute on this processor use 20% of A, 20% of B, and 60% of C types of instructions.

CPI of original machine = 0.2 * 2 + 0.2 * 7 + 0.6 * 3 (+1 for each term)
= 0.4+1.4+1.8
= 3.6

CPI of proposed machine = 0.2 * 2 + 0.2 * 5 + 0.6 * 3 (+1 for each term)
= 0.4+1.0+1.8
= 3.2

Ex. Time of Original = 3.6*C0
Ex. Time of New = 3.2 * C1
Ex. Time of New $\leq$ Ex. Time of Old (+4)
3.2 C1 $\leq$ 3.6 C0
C1 $\leq$ 3.6/3.2 C0

Max increase in clock cycle time $\leq$ (3.6/3.2 – 1) * 100 % +1 bonus
$\leq$ 12.5%

(bonus for subtracting 1, did not require simplification)

CS 2200 Spring 2018 Test 2A – soln

Name:___________________________ Email-id: _____________

Pipelining

2. (Total 15 points, 10 min)

(a) (3 points)(Answer True/False with justification)
"Delayed branch technique stops issuing new instructions into the pipeline until the outcome of the branch is determined."

False. (+1)
Some number of instructions (equal to the delay slots) following the branch always get executed. (+ 2)

(b) Assume the pipeline has busy bit (B), Read Pending signal (RP), and register forwarding in answering this question. Given the following sequence of instructions:

I1: add R1, R2, R3; R1 <- R2 + R3
I2: add R1, R3, R4; R1 <- R3 + R4
I3: sub R5, R1, R7; R5 <- R1 + R7

(i) (2 points) what potential hazards exist in the above instruction sequence?

WAW between I1 and I2 (+1)
RAW between I2 and I3 (+1)

(ii) (6 points)
Show the passage of these instructions through the 5-stage pipeline by filling in the table below

Cycle Number	IF	ID/RR	EX	MEM	WB
1	I_1				
2	I_2	I_1			
3	I_3	I_2	I_1		
4	I_3	I_2	NOP	I_1	
5	I_3	I_2	NOP	NOP	I_1
6	I_3	I_2	NOP	NOP	NOP
7		I_3	I_2	NOP	NOP
8			I_3	I_2	NOP
9				I_3	I_2
10					I_3
11					
12					
13					
14					
15					

-2 for general conceptual error, such as NOPs inserted in wrong stage

(+1)
(+1)
(+1)
(+1)
(+1)
(+1)

No double jeopardy

(c) Recall that the branch target buffer (BTB) records the past history of the outcomes of a given branch instruction execution to make a decision on future branch prediction. Given the contents of the branch target buffer as shown below for the currently executing program:

PC of branch instr.	**Past history (left is most recent)**	**Target of branch**
1000	NNTTTTNTTT	1400
1500	TTTTTTNNNN	2000
2000	TNNNNNNNNT	4000

(T: branch taken; N: branch not taken)

(i) (2 points) Branch instruction at address 1000 enters the pipeline. From which memory address will the next instruction be fetched and fed into the pipeline?

1400 (predictor uses the majority of the outcomes from previous visits to the branch instruction in making this decision)

(all or nothing)

(ii) (2 points) (Answer True/False with justification)
"The strategy in (i) will always eliminate branch induced bubbles in the pipeline."

False. (+1)
Prediction can go wrong. (+1)

Process Scheduling

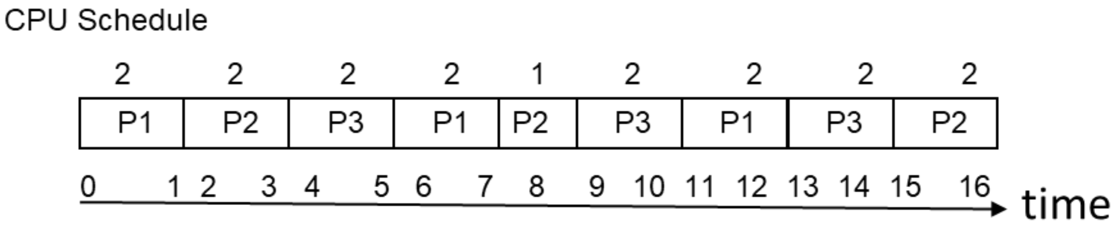
3. (15 points, 10 min)

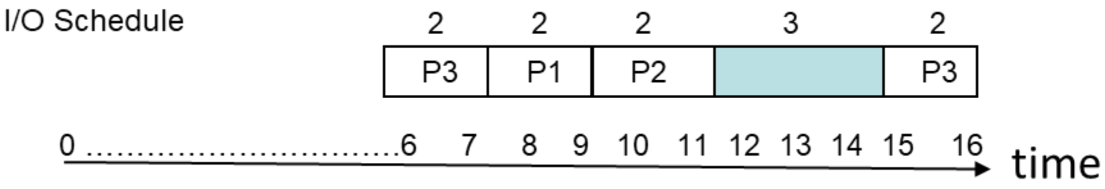
(a) (5 points)

Explain (**with 3 or 4 concise bullets**) the purpose of Process Context Block (PCB)

- Save volatile (general purpose registers) state of the process **(+2)**
- Save PC and SP **(+2 something about resuming execution)**
- Hold process metadata (memory occupancy, scheduling priority, etc.) **(+1 any mention of metadata, ethier for scheduling or VM)**

(b) (6 points)
Given the following schedule for three processes (starts at time = 0):
Note P1 completes at t = 12 (i.e., it takes a total of 13 time units to complete), P2 completes at t = 16, and P3 completes at t = 16. Numbers above the boxes are the lengths of the schedules. Numbers below the boxes are the linear progression of time.





(i) (4 points) What is the average waiting time experienced by the processes in the above schedule?

WT = turnaround time – execution time
P1: 12-8 = 4 **(or accept 13-8 = 5)**
P2: 16-7 = 9 **(or accept 17-7 = 10)**
P3: 16-10 = 6 **(or accept 17-10 = 7)**
Average wait time = (4+9+6)/3 = 19/3 ⇒ (+4)
(or accept (5 + 10 + 7)/3 = 22/3)

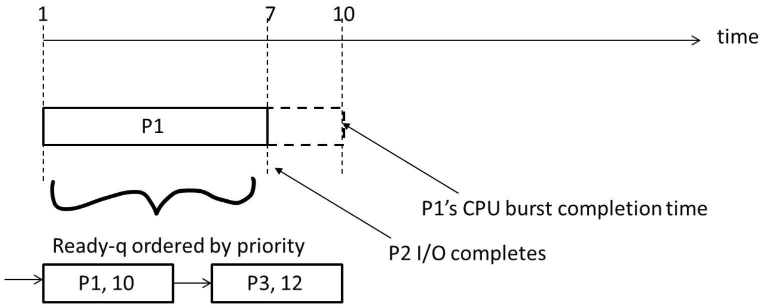
If final answer incorrect
+1 for each term above

(ii) (2 points) What is the average throughput of the system?

Average throughput = 3/16 (all or nothing)

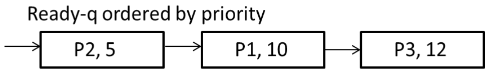
(or accept 3/16)

(c) (4 points)
The scheduling discipline used is **priority (lower number higher priority) with preemption.** Given the timeline below:



- P1 (priority = 10) is executing currently; (Ready queue is as shown above)
- P2 (priority = 5) completes I/O at time = 7

(i) (2 points) Show the new Ready queue (with all the processes) at time 7



(all or nothing)

(ii) (2 points) Explain which process will run next on the CPU, and why?

P2. (+1)

P1 will be preempted and P2 will get to run since it is higher priority. (+1)

CS 2200 Spring 2018 Test 2A – soln

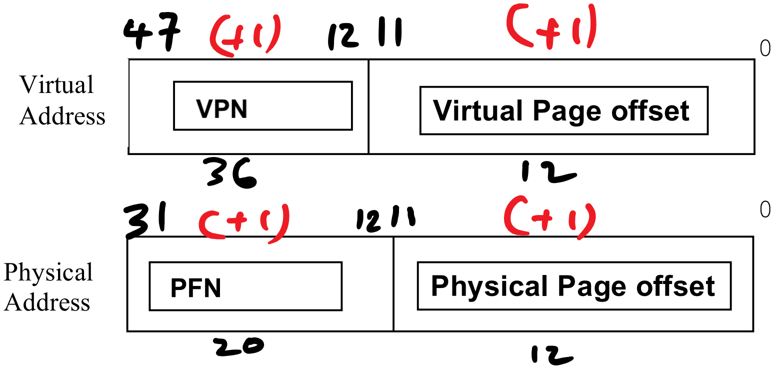
Name:__________________________ Email-id: ____________

Virtual Memory (Note: K = 1024), Working set

4. (16 points, 10 min)

(a) Consider a memory system with 48-bit virtual addresses and 32-bit physical addresses. The page size is 4 KB.

(i) (4 points) Assuming little-endian notation show the bit positions occupied by the VPN, Virtual Page Offset, PFN, and the Physical Page Offset in the figures below.



no double jeopardy

(ii) (2 points) How many entries are there in the page table?

2^{36} (all or nothing)

(iii) (2 points) How many page frames are there in the memory system?

2^{20} (all or nothing)

(b) During the time interval t1 - t2, the following virtual page accesses are recorded for the three processes P1, P2, and P3, respectively.

```
P1: 0,  10,  1,  0,  1,  2, 10, 2, 1, 1, 0, 25
P2: 0, 100, 101, 102, 103, 0, 101, 102, 104, 0, 105
P3: 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5, 0, 1, 2, 3, 4, 5
```

(i) (6 points) What is the working set for each of the above three processes for this time interval?

P1: {0, 1, 2, 10, 25} => 5 pages (+2)

P2: {0, 100, 101, 102, 103, 104, 105} => 7 pages (+2)

P3: {0, 1, 2, 3, 4, 5} => 6 pages (+2)

(ii)) (2 points) What is the cumulative memory pressure on the system during this interval?

Cumulative memory pressure = sum of all the three working sets = (5+7+6) * pagesize = 18*4K = 72 KB

(+1) (+1)

Demand paging, page replacement

5. (13 points, 10 min)

(a) (2 points) (Answer True/False with justification)
"With page-based virtual memory system there is no internal fragmentation."

False. (+1)
Internal fragmentation still possible if the entire page is not used. (+1)

(b) (2 points) A 32-bit architecture, has a 16-bit page offset and 16 bit VPN. If the VPN = 0x0000 maps to PFN = 0x0FFE, show the **full 32-bit** physical address corresponding to the virtual address 0x000035FE.

0x0FFE35FE (all or nothing)

(c) (2 points) The clock algorithm you implemented in your project as an approximation to LRU algorithm is referred to as "second chance replacement" algorithm. Why?

FIFO replacement candidate is spared from eviction (i.e., given a second chance to stay in memory) if its associated reference bit is set.

(d) (2 points)The current content of the TLB is as shown below (U – user entries; K – kernel entries)

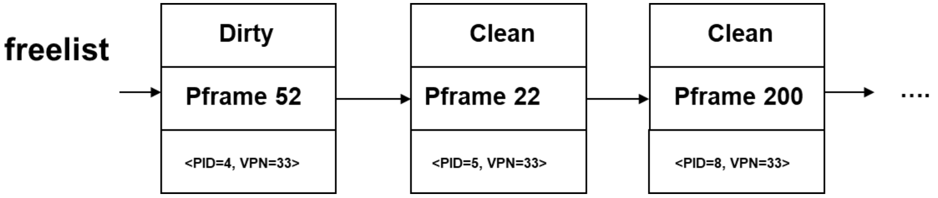
USER/KERNEL	VPN	PFN	VALID/INVALID
U	0	122	~~V~~ I
U	XX	XX	I
U	10	152	~~V~~ I
U	11	170	~~V~~ I
K	0	10	V
K	1	11	V
K	3	15	V
K	XX	XX	I

-1 each missing change
-1 extraneous changes

-1 if rows scratched out or stated that TLB is "flushed" without explaining what that entails.

Mark in the above table changes to the TLB upon a context switch

(e) (5 points) The freelist of page frames maintained by the memory management system looks as follows:



Process with PID=5 incurs a page fault on virtual page=33. What actions will be taken by the memory manager? Show your answers by modifying the data structures below:

PAGE TABLES

P4 (PID=4)

VPN	V	PFN
0	V	0
1	V	53
• • •		
33	I	
• • •		

P5 (PID=5)

VPN	V	PFN
0	V	199
1	V	1
• • •		
33	~~I~~ V	22 (+2)

P8 (PID=8)

VPN	V	PFN
0	V	23
1	V	400
• • •		
33	I	
• • •		

FRAME TABLE

PFN	<PID, VPN>
0	<P4, 0>
1	<P5, 1>
• • •	
22	~~Un-assingned~~ <P5, 33> (+3)
23	<P8, 0>
• • •	
52	Un-assigned
53	<P4, 1>
• • •	
199	<P5, 0>
200	Un-assigned
• • •	

(No credit for each if extraneous changes)

CS 2200 Spring 2018 Test 2A – soln

Name:___________________________ Email-id: _____________

Cache Design, Execution time (Note: K = 1024; M = 1024*1024)

6. (Total 13 points, 10 min)

(a) (3 points) Associate definitions below (A, B, C) with the type of miss choosing from

compulsory miss,
conflict miss,
capacity miss.

A. Miss incurred when the cache is full ________ **capacity** __________ (+1)

B. Miss incurred since memory location accessed for the first time by CPU ________ **compulsory** ________ (+1)

C. Miss incurred due to limited associativity even though the cache is not full ________ **conflict** __________ (+1)

(b) (10 points) In a pipelined processor

- **average CPI = 1.3** without accounting for memory stalls.
- I-Cache has a **hit rate** of 98%
- D-Cache has a **hit rate** of 99%
- Memory reference instructions account for 30% of all the instructions executed
- Out of these memory reference instructions 80% are loads and 20% are stores.
- Read-miss penalty (for instruction or data) is 100 cycles
- Write-miss penalty is 5 cycles.

Compute the effective CPI of the processor accounting for the memory stalls.

Effective CPI = Average CPI + Memory Stalls (+1)
Memory Stalls = I-cache miss penalty + D-cache miss penalty (+1)
I-cache miss penalty = I-Cache miss rate * read-miss penalty (+2)
= 0.02 * 100 = 2
D-cache miss penalty = fraction of memory reference instructions * D-cache miss rate * (fraction of loads * read-miss penalty + fraction of stores * write-miss penalty)
= 0.3 * 0.01 * (0.8 * 100 + 0.2 * 5)
= 0.003 * (81)
= 0.243
Effective CPI = 1.3+ 2 + 0.243 = 3.543 (+2)

+2 D-cache miss rate * fraction of mem instrs.

+2 fraction of LW and SW accounted for

Partial credit as above if final answer incorrect.
No double jeopardy

```
7. (Total 17 points, 15 min)
(a) (2 points)
How is the principle of spatial locality exploited in cache design?
```

By having a larger block size (than the granularity of CPU access) in the cache

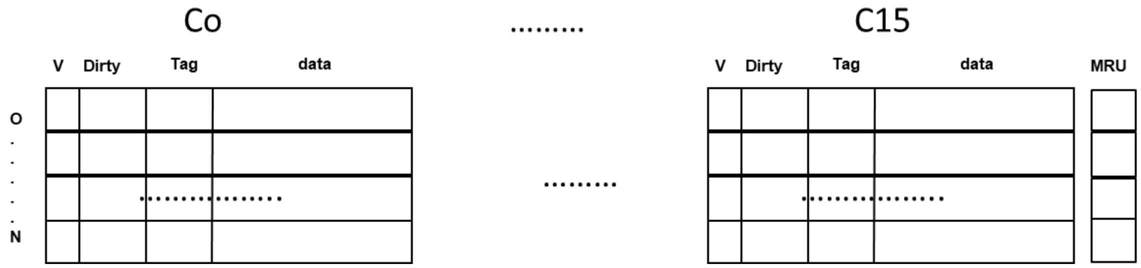
+2

```
(b) (2 points)
How is the principle of temporal locality exploited in cache design?
```

By using the LRU principle for block eviction in set-associative cache design

+2 **(also accepted any discussion of set associativity)**

```
(c) (13 points)
Consider a 16-way set associative cache with the following parameters:
```



- Cache size (i.e, the amount of actual data it can hold) of **1 Mbytes.**
- **32-bit byte-addressable** memory.
- Each memory word contains **4 bytes**
- cache block size is **64 bytes.**
- **write-back** policy. **One dirty bit per word.**
- **one valid bit** per block.
- **MRU field** that records the most recently cache for that cacheline

```
(i) (3 points) How big is N in the above picture (show your work)?
```

Total number of cache blocks = cache size/block size

= 1 Mbyte/64 bytes = 2^{14} (+1)

Number of lines in each cache = total number of cache blocks/associativity

= $2^{14}/2^{4}$ **(+2 for blocks/associativity)**

= 2^{10}

= 1024

N = 1023

(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**

As shown above, the memory address is partitioned into **b** - the number of bits for the block offset; **n** - the number of bits for the cache index; and **t** - the number of bits for the tag. What are the values of **t**, **n**, and **b** (show your work)?

$b = \log_2 64 = 6$ (+2)

$n = \log_2(\text{cachelines}) = \log_2 2^{10} = 10$ (+2)

$t = 32 - (6+n) = 32 - 16 = 16$ (+2)

No double jeopardy

(iii) (4 points) How many meta-data bits are there totally for the entire cache as shown in the above picture? (show your work)
[Note: Remember that meta-data refers to all the other stuff (besides the data itself) that goes into cache.]

MRU bits for the entire organization = $\log_2$associativity $\log_2 16 = 4$ (+1)

For each of the 16 parallel caches (C0 to C15):
Valid bit per line = 1
Dirty bits per line = 16
Tag bits per line = 16
] +2
(-1 if only a single dirty bit per block)

Meta data for the entire cache organization = (MRU bits + (associativity * valid+dirty+tag)) * number of rows] +1
= (4 + 16 * (1+16+16)) * 1024
= 544,768 bits

Acronyms

IF - instruction fetch (fetch into IR and increment PC)

ID/RR - instruction decode/register read (read register contents)

EX - execute (perform arithmetic/logic/address computation - maybe)

MEM - memory (fetch/store memory operand - maybe)

WB - write back to register file (may be)

TLB - Translation Look-aside Buffer

PTE - Page Table Entry

PFN - Physical Frame Number

VPN - Virtual Page Number

RLT - Reverse Lookup Table

PCB - Process Control Block

FCFS – First Come First Served

SJF – Shortest Job First

SRTF – Shortest Remaining Time First

Name	Notation	Units	Description
CPU Utilization	-	**%**	**Percentage of time the CPU is busy**
Throughput	**n/T**	**Jobs/s**	**System-centric metric quantifying the number of jobs *n* executed in time interval *T***
Avg. Turnaround time (t_{avg})	$(t_1+t_2+\ldots+t_n)/n$	**Secs**	**System-centric metric quantifying the average time it takes for a job to complete**
Avg. Waiting time (w_{avg})	$(w_1+w_2+\ldots+w_n)/n$	**Secs**	**System-centric metric quantifying the average waiting time that a job experiences**
Response time	t_i	**Secs**	**User-centric metric quantifying the turnaround time for a specific job *I***
Variance in Response time	$E[(t_i - t_{avg})^2]$	**Secs^2**	**User-centric metric that quantifies the statistical variance of the actual response time (t_i) experienced by a process (P_i) from the expected value (t_{avg})**
Starvation	-	-	**User-centric qualitative metric that signifies denial of service to a particular process or a set of processes due to some intrinsic property of the scheduler**
Convoy effect	-	-	**User-centric qualitative metric that results in a detrimental effect to some set of processes due to some intrinsic property of the scheduler**

Name	Notation	Units	Comment
Memory footprint	-	Bytes	Total space occupied by the program in memory
Execution time	$(\sum CPI_j)$ * clock cycle time, where $1 \leq j \leq n$	Seconds	Running time of the program that executes *n* instructions
Arithmetic mean	$(E_1+E_2+\ldots+E_p)/p$	Seconds	Average of execution times of constituent *p* benchmark programs
Weighted Arithmetic mean	$(f_1*E_1+f_2*E_2+\ldots+f_p*E_p)$	Seconds	Weighted average of execution times of constituent *p* benchmark programs
Geometric mean	p^{th} root $(E_1*E_2*\ldots*E_{p)}$	Seconds	p^{th} root of the product of execution times of *p* programs that constitute the benchmark
Harmonic mean	$1/(\ ((1/E_1)+(1/E_2)+\ldots+(1/E_p)\ )/p\ )$	Seconds	Arithmetic mean of the reciprocals of the execution times of the constituent *p* benchmark programs
Static instruction frequency		%	Occurrence of instruction *i* in compiled code
Dynamic instruction frequency		%	Occurrence of instruction *i* in executed code
Speedup (M_A over M_B)	E_B/E_A	Number	Speedup of Machine A over B
Speedup (improvement)	E_{Before}/E_{After}	Number	Speedup After improvement
Improvement in Exec time	$(E_{old}-E_{new})/E_{old}$	Number	New Vs. old
Amdahl's law	$Time_{after}$ $= Time_{unaffected} + Time_{affected}/x$	Seconds	*x* is amount of improvement

Execution time = N * CPI_{Avg} * cycle time

Execution time = N * CPI_{eff} * cycle time

CPI_{eff} = CPI_{Avg} + Memory-$stalls_{Avg}$

Execution time = N * (CPI_{Avg} + M-$stalls_{Avg}$) * cycle time

Memory-$stalls_{Avg}$ = misses per $instruction_{Avg}$ * miss-$penalty_{Avg}$

Total memory stalls = N * Memory-$stalls_{Avg}$