CS 2200 Spring 2008 Test 2 - Solution

Name:_____Kishore_____GT Number: gt_____

Please indicate your GT number in the grid below so we have some chance of being able to read it.

G																								
T																								
0	1	2	3	4	5	6	7	8	9	0	e	g	h											
0	1	2	3	4	5	6	7	8	9	0														
0	1	2	3	4	5	6	7	8	9	0														
0	1	2	3	4	5	6	7	8	9	0														
A	b	c	D	E	f	G	h	i	j	k	L	m	n	o	p	q	r	s	t	u	v	w	x	y	z

Problem	Points	Lost	Gained	Running Total	TA
1	1				
2	12				
3	20				
4	9				
5	8				
6	10				
7	15				
8	10				
9	15				
Total	100				

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

Illegible answers are wrong answers.
Good luck!

1. (1 point, 0 min)
How many characters are in your professor's last name?
(a) 7 (b) 12 (c) 9 (d) 10 (e) 13

CS 2200 Spring 2008 Test 2 - Solution

Name:_____Kishore_____GT Number: gt_____

Pipelined processor
2. (12 points, 10 minutes)

(a) (2 points) (select one correct choice)

Branch target buffer is
(a)An area of memory reserved for branch instructions
(b)A hardware device that keeps the outcome and target addresses of recent branches encountered during the program execution
(c)A hardware device that is pre-loaded before the program starts with the expected outcome and the target addresses of the branches in the program
(d)An extra stage in the pipeline for efficient handling of control hazards
(e)None of the above
(f)All of the above

(b) (2 points)(select one correct choice)

Structural hazard in a pipeline occurs due to
(a)Branch instructions in the program
(b)Load instructions in the program
(c)Data dependencies in the program
(d)Hardware limitations in the datapath
(e)None of the above
(f)All of the above

(c) (2 points)(select one correct choice)

RAW (read after write) hazard can be overcome by
(a)Using a branch target buffer
(b)Forwarding a register value from the execution stage to the register reading stage
(c)Forwarding a register value from the register reading stage to the execution stage
(d)Additional ALU in the execution stage
(e)Pipelining the processor
(f)Clever instruction-set design

CS 2200 Spring 2008 Test 2 - Solution

Name:_____Kishore_____GT Number: gt_____

(d) (6 points)

Itemize the steps taken in hardware from the time an interrupt occurs to the time the processor starts executing the handler code in a **pipelined** processor. (An English description is sufficient.)

1. **Allow instructions already in the pipeline (useful instructions) to complete execution**
2. **Stop fetching new instructions**
3. **Start sending NOPs from the fetch stage into the pipeline**
4. **Once all the useful instructions have completed execution, record the address where the program needs to be resumed in the PC (this will be memory address of last completed useful instruction + 1)**
5. **GO to INT state; sent INTA; receive vector; disable interrupts**
6. **Save current mode in system stack; change mode to kernel mode;(OK if they don't say this)**
7. **Store PC in $k0; Retrieve handler address using vector; Load PC; resume pipeline execution**

Process scheduling
3. (20 points, 15 mins)

(a) (2 points) (select one correct choice)

The following attributes are likely to be found in the PCB of a process
1. General Purpose Registers that are visible to the instruction set
2. Program counter and the register that represents the stack pointer
3. Pointer to the page table of the process
4. Priority information
5. Internal registers in the datapath of the processor
6. {1, 2, 3, 4, 5}
7. {1, 2, 3}
8. {1, 2, 3, 4}
9. None of the above

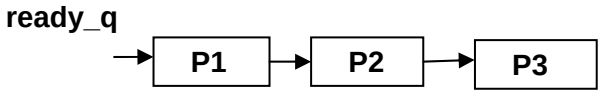
CS 2200 Spring 2008 Test 2 - Solution

Name:_____Kishore_____GT Number: gt_____

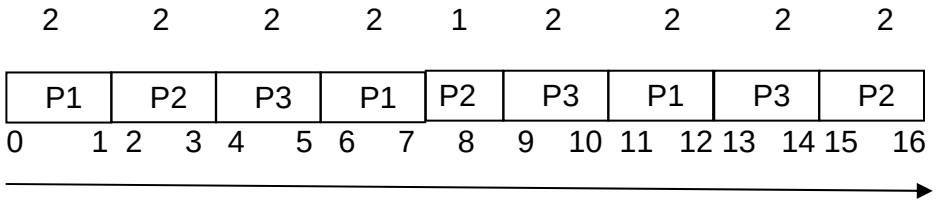
(b)
This problem uses **round-robin schedule** with a **time quantum = 2**.
Consider three processes with CPU and I/O bursts as shown in the table below:

	CPU	I/O	CPU	I/O	
P1	4	2	2		**P1 is done**
P2	3	2	2		**P2 is done**
P3	2	2	4	2	**P3 is done**

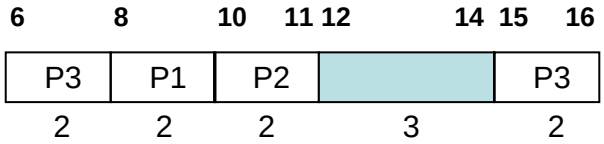
(i) (10 points)
Show the CPU and I/O timelines that result with round-robin scheduling from t=0 until all three processes exit the system.



CPU Schedule (Round Robin, timeslice = 2)



I/O Schedule



(ii)(5 points)
What is the waiting time for each process?

Wait time = (response time – useful work done either on the CPU or I/O)
Wait time for P1 = (13 – 8) = 5
Wait time for P2 = (17 – 7) = 10
Wait time for P3 = (17 – 10) = 7

(iii) (3 points)
What is the average throughput of the system?

Throughput of the system
= number of completed processes/total execution time
= 3/17 processes/unit-time

Virtual Memory and Physical Memory
4. (9 points, 10 mins)

CS 2200 Spring 2008 Test 2 - Solution

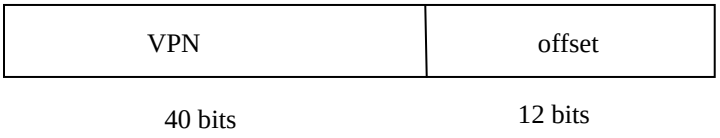
Name:_____Kishore_____GT Number: gt_____

Consider a memory system with **52-bit virtual addresses** and **32-bit physical addresses**. The page size is **4 KB**.

a) (4 points)
Show the layout of the virtual and physical addresses.

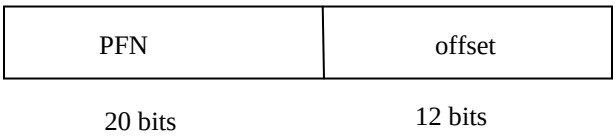
Number of bits for page offset = $\log_2 4096 = 12$
Number of bits for virtual page number (VPN)
= number of bits in virtual address – page offset bits
= 52-12
= 40 bits

Layout of virtual address:



Number of bits for physical frame number
= number of bits in physical address – page offset bits
= 32-12

= 20
Layout of physical address:



b) (3 points)
How many entries are in the page table?

Number of page table entries = $2^{(\text{number of bits in VPN})} = 2^{40}$

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

Number of page frames = $2^{(\text{number of bits in PFN})} = 2^{20}$

Working Set
5. (8 points, 5 mins)

Define the following terms with respect to memory management

CS 2200 Spring 2008 Test 2 - Solution

Name:_____Kishore_____GT Number: gt_____

a) Thrashing

Cumulative memory requirement of the processes currently competing for time on the CPU far exceeds the total physical memory of the system leading to a state where the system is spending most of the time swapping pages in and out of the physical memory to the secondary storage

b) Working set

The *exact set* of virtual memory pages that a process needs to allow its execution to proceed without incurring page faults

c) Working set size

The *minimum number* of page frames that a process needs to allow its execution to proceed without incurring page faults

d) Memory pressure

The *cumulative number* of page frames needed by all the processes currently competing for time on the CPU in the system. This is the same as the sum of the working sets of all the processes currently competing for time on the CPU in the system.

Total memory pressure = $\sum_{i=1}^{i=n} WSSi$

Demand paging and Page replacement
6. (10 points, 5 mins)

(a) (2 points) (select one correct choice)

To implement paging the minimum additional hardware needed in the CPU data path

CS 2200 Spring 2008 Test 2 - Solution

Name:_____Kishore_____GT Number: gt_____

1. One Page table implemented in hardware
2. Multiple page table (one per process) implemented in hardware
3. One Page Table Base Register (PTBR)
4. Multiple PTBR (one per process)
5. None of the above

(b) (2 points) (select one correct choice)

With demand paging, a page fault
1. Occurs only in a non-pipelined processor
2. Can occur in any stage of a pipelined processor
3. Can occur only in the stages that need to access memory
4. Can never occur once the process is scheduled to run

(c) (6 points)

Five of the following seven operations take place upon a page fault when there is no free frame in memory.
1. use the frame table to find the process that owns the faulting page
2. using the disk map of faulting process, load the faulting page from the disk into the victim frame
3. select a victim page for replacement (and the associated victim frame)
4. update the page table of faulting process and frame table to reflect the changed mapping for the victim frame
5. using the disk map of the victim process, copy the victim page to the disk (if dirty)
6. 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
7. look up if the faulting page is currently in physical memory

i. Put the five correct operations in the right sequence

Sequence of steps in order:
3; 6; 5; 2; 4
(Note: 4 can precede 5 or 2 but the other operations have to occur in the order shown)

ii. Identify the two incorrect operations.

Incorrect operations :
1, 7

Caches
7. (15 points, 12 min)

(a) (6 points)
Given the following code fragment:

a[0] = 0; **(1)**

CS 2200 Spring 2008 Test 2 - Solution

Name:_____Kishore_____GT Number: gt_____

```
a[1] = 1;                                   (2)
for (i = 2; i < 100; i++) {
      a[i] = a[i-1] + a[i-2];               (3)
}
```

Answer true/false with justification:

(i) The above code fragment exploits spatial locality
True. Assuming the array is aligned on cache block boundary and the block size is 4 array elements, write access to a[0] (step 1) will result in a write miss. However, this write miss will bring in a[1], a[2], a[3] into the cache. Thus the accesses to a[1], a[2], and a[3] (in steps 2 and 3) do not result in write misses.

(ii) The above code fragment exploits temporal locality
True. In the loop, every iteration uses the previous two recently generated values. For e.g., a[3] uses a[2] and a[1].

(b) (6 points)
Consider the following memory hierarchy:

- L1 cache: Access time = 3ns; hit rate = 98%
- L2 cache: Access time = 6ns; hit rate = 90%
- Main memory: Access time = 100ns

Compute the effective memory access time (EMAT).

$EMAT_i = T_i + m_i * EMAT_{i+1}$
$EMAT_{L2} = T_{L2}+(1-h_2)*T_m$
$= 6 + (1-0.9)*100$
$= 16$ ns
$EMAT_{L1} = T_{L1}+(1-h_1)*EMAT_{L2}$
$= 3 + (1-.98) * 16$
$= 3 + 0.32$
$= 3.32$ ns
$EMAT = EMAT_{L1} = 3.32$ ns

(c) (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 miss**
B. Miss incurred since memory location accessed for the first time by CPU **compulsory miss**
C. Miss incurred due to limited associativity even though the cache is not full **conflict miss**

CS 2200 Spring 2008 Test 2 - Solution

Name:_____Kishore_____GT Number: gt_____

Effect of Memory Hierarchy on Memory Stalls in a Pipeline
8. (10 points, 10 mins)
Consider a pipelined processor:

- I-Cache hit rate = 95%.
- D-Cache hit rate = 98%.
- Assume that memory reference instructions account for 10% of all the instructions executed. Out of these 80% are loads and 20% are stores.
- Read-miss penalty = 25 cycles.
- Write-miss penalty = 4 cycles.

(a) (4 points)
What is the average number of memory stalls (in clock cycles) experienced per instruction due to I-cache misses?

Average Memory stalls due to I-cache misses per instruction
= I-cache miss rate * read-miss penalty
= (1-0.95) * 25
= 1.25 cycles

(b) (6 points)
What is the average number of memory stalls (in clock cycles) experienced per instruction due to D-cache missies?

Average Memory stalls due to D-cache read misses per instruction
= D-Cache miss rate *
fraction of instructions that are memory reference *
fraction of memory reference instructions that are load *
read-miss penalty
= (1-0.98) * 0.1 * 0.8 * 25
= 0.04 cycles

Average Memory stalls due to D-cache write misses per instruction
= D-Cache miss rate *
fraction of instructions that are memory reference *
fraction of memory reference instructions that are stores *
write-miss penalty
= (1-0.98) * 0.1 * 0.2 * 4
= 0.0016 cycles

Average memory stalls due to D-cache misses per instruction =
Average Memory stalls due to D-cache read misses per instruction
+
Average Memory stalls due to D-cache write misses per instruction
= 0.04 + 0.0016 = 0.0416 cycle

Cache design
9. (15 points, 13 mins)
Consider a 8-way set-associative cache.

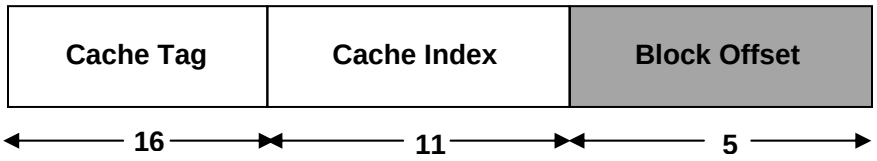
- Total data size of cache = 512 KB.
- CPU generates 32-bit byte-addressable memory addresses.
- Each memory word consists of 4 bytes.
- The block size is 32 bytes.
- The cache has one valid bit per block.
- The cache uses write-back policy with one dirty bit per word.

CS 2200 Spring 2008 Test 2 - Solution

Name:_____Kishore_____GT Number: gt_____

(a) (5 points)
Show how the CPU interprets the memory address (i.e., which bits are used as the cache index, which bits are used as the tag, and which bits are used as the offset into the block?).

Number of bits in block offset = $\log_2$blocksize = $\log_2$32 = 5 bits
Since the cache is 8-way set associative, number of lines in cache
= size of cache / (block size * associativity)
= (512 KB) / ((32 * 8) bytes)
= 2048 cache-lines
Number of index bits needed to access the cache
= $\log_2$number-of-cachelines = $\log_2$2048 = 11 bits
Number of tag bits
= total number of address bits –
(number of index bits + number of offset bits)
= 32 – (11+5) = 16 bits



(b) (10 points)
Compute the amount of meta data in each cache line (show partial work to get any credit)

Number of words in a data block = blocksize/wordsize = 32/4 = 8
Meta data in each data block of the cache line:
Tag = 16 bits
Valid = 1 bit
Dirty bits = 8 bits (1 per word in a data block)

Total = (16+1+8) = 25 bits
8 such data blocks in each cache line (8-way set associative).

Meta data per cache line (since it is 8-way):
8 * meta-data in each data block
= 8 * 25 = 100 bits