The University of Alabama in Huntsville
ECE Department
CPE 431 01
Test 2 Solution
Fall 2019

1. (1 point) _**Spatial**_ locality states that items close to items referenced recently will be referenced.
2. (1 point) Miss is to cache as _**page fault**_ is to physical memory.
3. (1 point) _**Non-volatile**_ storage retains information when power is removed.
4. (1 point) A _**structural**_ hazard occurs when a planned instruction cannot execute in the proper clock cycle because the hardware does not support the combination of instructions that are set to execute.
5. (1 point) An _**exception**_ is an unscheduled event that disrupts program execution.
6. (10 points) The following code is written in MATLAB, where elements within the same column are stored contiguously. Do `A(j, i)` and `A(i, j)` exhibit spatial locality? temporal locality? Explain your answers. A, B, and C are all arrays of integers 8000 by 8000.

```
for i = 1:8000
  for j = 1:8000
      A(i,j ) = B(j, 1) + A(j, i) + C(1, i);
```

Variable	Locality Type	Exhibited?	Explanation
`A(j, i)`	Spatial	Yes	Access pattern is A(1,1), A(2, 1), A(3, 1) .., so elements are accessed from the same column
`A(j, i)`	Temporal	No	Different address each time
`A(i, j)`	Spatial	No	Access pattern is A(1,1), A(1, 2), A(1, 3) .., so elements are accesses from different columns
`A(i, j)`	Temporal	No	Different address each time

7. (20 points) Here is a series of address references given as byte addresses: 118, 483, 2069, 321, 368, 1505, 812, 2832, 373, 1411, 511, 122, 690, 4820, 1714, 1508, 2070, 1080. Assuming a two-way set associative-mapped cache with 2-word blocks and a total size of 32 32-bit words that is initially empty and uses LRU, (a) label each reference in the list as a hit or a miss and (b) show the entire history of the cache, including tag and data.
32 words x 1 block/2 words x 1 set/2 blocks = 8 sets, 3 bits index, 1 bit block offset, 2 bits byte offset

Byte Address (Decimal)	Byte Address (Hexadecimal)	Byte Address (Binary) **Index**	Hit/Miss
118	76	0000 0000 01 110 1 10	Miss
483	1E3	0000 0001 11 100 0 11	Miss
2069	815	0000 1000 00 010 1 01	Miss
321	141	0000 0001 01 000 0 01	Miss
368	170	0000 0001 01 110 0 00	Miss
1505	5E1	0000 0101 11 100 0 01	Miss
812	32C	0000 0011 00 101 1 00	Miss
2832	B10	0000 1011 00 010 0 00	Miss
373	175	0000 0001 01 110 1 01	Hit
1411	583	0000 0101 10 000 0 11	Miss
511	1FF	0000 0001 11 111 1 11	Miss
122	7A	0000 0000 01 111 0 10	Miss
690	2B2	0000 0010 10 110 0 10	Miss
1508	5E4	0000 0101 11 100 1 00	Hit
2070	816	0000 1000 00 010 1 10	Hit
1080	438	0000 0100 00 111 0 00	Miss

Index	Tag	Data	Tag	Data
0	5	M[320:327]	22	M[1408:1415]
1				
2	44	M[2832:2839]	32	M[2064:2071]
3				
4	7	M[480:487]	23	~~M[1504:1511]~~
5	12	M[808:815]		
6	5	M[368:375]	~~1~~, 10	~~M[112:119]~~, M[688:695]
7	1	M[120:127]	~~7~~, 16	~~M[504:511]~~, M[1080:1087]

8. (20 points) Consider the execution of the following loop in a statically scheduled superscalar processor that has full forwarding. Additionally, any branches are resolved in the EX stage.

```
Loop:  lw   $t0, 0($s1)
       lw   $t4, 0($s2)
       mul  $t0, $t0, $t4
       add  $t0, $t3, $t0
       addi $s1, $s1, -8
       addi $s2, $s2, -8
       bne  $s1, $zero, Loop
```

a. (15 points) Unroll this loop so that three iterations of it are done at once and schedule it for maximum performance on a 2-issue static superscalar processor that has one slot for R-type/branch instructions and one slot for lw/sw instructions. Assume that the loop always executes a number of iterations that is a multiple of 3. You can use registers **$10** through **$20** when changing the code to eliminate dependences.

b. (5 points) Calculate the number of cycles for the original and for the unrolled, scheduled code and the speedup of unrolled, scheduled compared to the original.

Cycle	R-type/branch	lw/sw
1	addi $s1, $s1, -24	lw $t0, 0($s1)
2	addi $s2, $s2, -24	lw $t4, 0($s2)
3		lw $10, 16($s1)
4	mul $t0, $t0, $t4	lw $11, 16($s2)
5	add $t0, $s3, $t0	lw $12, 8($s1)
6	mul $10, $10, $11	lw $13, 8($s2)
7	add $10, $t3, $10	
8	mul $12, $12, $13	
9	add $12, $t3, $12	
10	bne $s1, $zero, Loop	

$CC_{original}$ = 3 iterations *(7 instructions @ 1 cycle each + 1 data hazard stall + 2 cycle branch stall) = 30 cycles
$CC_{unrolled,scheduled}$ = 10 instructions + 2 cycle branch stall = 12
Speedup = $CC_{original}/CC_{unrolled,scheduled}$ = 30/12 = 2.5

8. (10 points) Consider an SEC code that protects 8 bit words with 4 parity bits. If we read the value 0x294, is there an error? If so, correct the error.

```
          111
123456789012
001010010100
```

C1 = XOR(1, 3, 5, 7, 9 , 11) = XOR(0, 1, 1, 0, 0, 0) = 0
C2 = XOR(2, 3, 6, 7, 10, 11) = XOR(0, 1, 0, 0, 1, 0) = 0
C4 = XOR(4, 5, 6, 7, 12) = XOR(0, 1, 0, 0, 0) = 1
C8 = XOR(8, 9, 10, 11, 12) = XOR(1, 0, 1, 0, 0,) = 0
Bit 4 is in error, correct value 0x394

10. (15 points) Virtual memory uses a page table to track the mapping of virtual addresses to physical addresses. The following table is a stream of virtual addresses as seen on a system. Assume 16 KiB pages, byte addressing, a four-entry fully associative TLB, and true LRU replacement. If pages must be brought in from disk, increment the next largest page number. Given the address stream, and the shown initial state of the TLB and page table, show the final state of the system. Also list for each reference if it is a hit in the page table, or a page fault.

TLB

Valid	Tag	Physical Page Number
1	~~11~~, 1	~~12~~, 13
1	7	4
1	3	6
~~0~~, 1	~~4~~, ~~5~~, 4	9, 11, 9

Page table

VPN	Valid	Physical page or in disk
0	1	5
1	~~0~~, 1	~~Disk~~, 13
2	0	Disk
3	1	6
4	1	9
5	1	11
6	0	Disk
7	1	4
8	0	Disk
9	0	Disk
10	1	3
11	1	12
12	0	Disk

Address	VPN Address/ Page Size	TLB Hit/Miss	Page Table Hit/Miss	Page Fault Y/N
92,158	5	Miss	Hit	N
52,250	3	Hit	-	N
120,000	7	Hit	-	N
119,200	7	Hit	-	N
28,156	1	Miss	Miss	Y
73,004	4	Miss	Hit	N
63,998	3	Hit	-	N
20,000	1	Hit	-	

11. (10 points) Assume that the following MIPS code is executed on a pipelined processor with a 5-stage pipeline, full forwarding, and a predict-not-taken branch predictor:

```
            lw    r2, 0(r1)
label1:     beq   r2, r0, label2     # not taken once, then taken
            lw    r3, 0(r2)
            beq   r3, r0, label1     #taken
            add   r1, r3, r1
label2:     sw    r1, 0(r2)
```

Draw the pipeline execution diagram for this code, assuming there are no delay slots and that branches execute in the EX stage.

Cycle	IF	ID	EX	MEM	WB
1	lw r2				
2	beq r2	lw r2			
3	lw r3	beq r2	lw r2		
4	lw r3	beq r2	bubble	lw r2	
5	beq r3	lw r3	beq r2	bubble	lw r2
6	add r1	beq r3	lw r3	beq r2	bubble
7	add r1	beq r3	bubble	lw r3	beq r2
8	sw r1	add r1	beq r3	bubble	lw r3
9	beq r2	bubble	bubble	beq r3	bubble
10	lw r3	beq r2	bubble	bubble	beq r3
11	beq r3	lw r3	beq r2	bubble	bubble
12	sw r1	bubble	bubble	beq r2	bubble
13		sw r1	bubble	bubble	beq r2
14			sw r1	bubble	bubble
15				sw r1	bubble
16					sw r1

12. (10 points) Calculate the total number of bits required for a cache with the parameters listed below, assuming a 32-bit address and byte addressability. Include valid and dirty bits.

Cache Data Size: 256 KiB
Cache Block Size: 8 32-bit words
Cache Set Associativity: 8-way

256 KiB x 1 word/4 bytes x 1 block/8 words x 1 set/8 blocks = 1024 sets, 10 index bits
Byte Offset – 2 bits, Block Offset – 3 bits, Tag = 32 – 10 – 2 – 3 = 17 bits
Number of bits = #sets * #entries/set *(Tag + Data + Valid + Dirty) = 1024 * 8 * (17 + 256 + 2) = 2,252,800 bits