The University of Alabama in Huntsville
Electrical & Computer Engineering Department
CPE 431 01
Test 2 Solution
Fall 2010

1. (1 point) _Fault avoidance_ is preventing fault occurrence by construction.
2. (1 point) Each disk surface of a magnetic disk is divided into concentric circles, called _tracks_.
3. (1 point) The underlying technology in flash memories is _EEPROM_
4. (1 point) The process of periodically checking status bits to see whether it is time for the next I/O operation is called _polling_.
5. (1 point) RAID 1 uses _mirroring_ to always provide two copies of all data..
6. (15 points) Virtual memory uses a page table to track the mapping of virtual addresses to physical addresses. Consider this stream of virtual addresses as seen on a system: 20717, 29637, 8849, 36965, 4493, 28674, 20517, 9179, 25632, 21111. Assume 4KB pages, 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 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 TLB, a hit in the page table, or a page fault.

TLB

Valid	Tag	Physical Page Number
1	~~11~~, ~~2~~, 5	~~12~~, ~~13~~, 11
1	7	4
1	~~3~~, ~~9~~, 2	~~6~~, ~~14~~, 13
~~0~~, 1	~~4~~, ~~5~~, ~~1~~, 6	~~9~~, ~~11~~, ~~15~~, 16

Page table

	Valid	Physical page or in disk
0	1	5
1	~~0~~, 1	~~Disk~~, 15
2	~~0~~, 1	~~Disk~~, 13
3	1	6
4	1	9
5	1	11
6	~~0~~, 1	~~Disk~~, 16
7	1	4
8	0	Disk
9	~~0~~, 1	~~Disk~~, 14
10	1	3
11	1	12

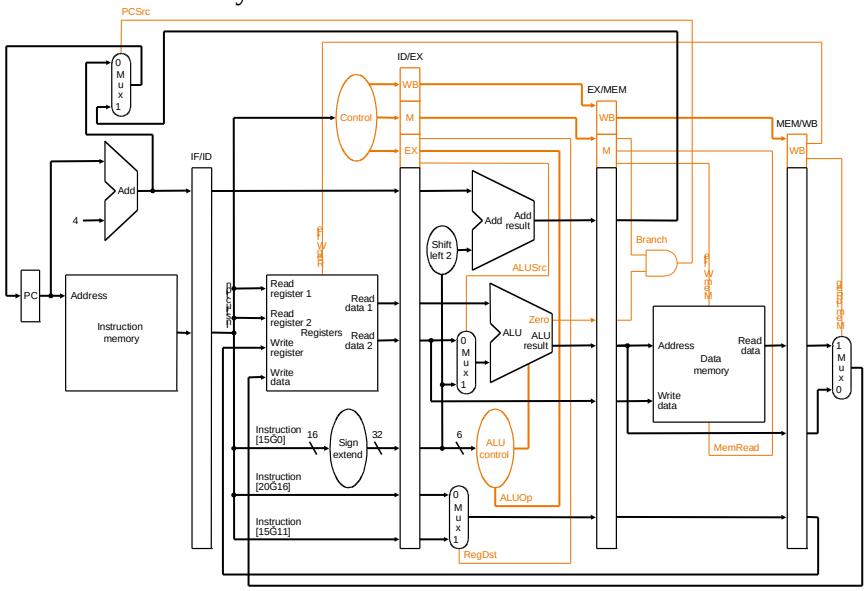
These are byte addresses. The page offset is $\log_2$ 4K = 12. The virtual page number can be obtained by dividing the byte address by 4KB (4096) and taking the integer part of the result.

	VPN	TLB?	PT?	PF?	Replacing 3 TLB?	PT?	PF?
20717	5	M	H	N	M	H	N
29637	7	H			H		
8849	2	M	M	Y	M	M	Y
36965	9	M	M	Y	M	M	Y
4493	1	M	M	Y	M	M	Y
28674	7	H			H		
20517	5	M	H	N	M	H	N
9179	2	M	H	N	M	H	N
25632	6	M	M	Y	M	M	Y
21111	5	H					

Note: When bringing in page 4 to the TLB, 11 and 3 are equivalent choices for LRU, could have evicted 11.

Valid	Tag	PPN
1	~~11~~, ~~9~~, 2	~~12~~, ~~14~~, 13
1	7	4
1	~~3~~, ~~2~~, 5	~~6~~, ~~13~~, 11
~~0~~, 1	~~4~~, ~~5~~, ~~1~~, 6	~~9~~, ~~11~~, ~~15~~. 16

7. (20 points) Consider executing the following code on a pipelined datapath like the one shown except that 1)it supports j instructions that complete in the ID stage, 2) it has forwarding from the EX/MEM pipeline register but no forwarding from the MEM/WB pipeline register, 3)it has branch completion in the ID stage. The register file does support writing in the first half cycle and reading in the second half cycle.




```
sort:     addi  $sp, $sp, -20
          sw    $ra, 16($sp)
          sw    $s3, 12($sp)
          sw    $s2, 8($sp)
          sw    $s1, 4($sp)
          sw    $s0, 0($sp)
          add   $s2, $a0, $zero
          add   $s3, $a1, $zero
          add   $s0, $zero, $zero
for1tst:  slt   $t0, $s0, $s3
          beq   $t0, $zero, exit1
          addi  $s1, $s0, -1
for2tst:  slt   $t0, $s1, $zero
          bne   $t0, $zero, exit2
          add   $t1, $s1, $s1
          add   $t1, $t1, $t1
          add   $t2, $s2, $t1
  200     ⇒lw   $t3, 0($t2)
  204       lw    $t4, 4($t2)
  208       slt   $t0, $t4, $t3
  212       beq   $t0, $zero, exit2
  216*****add   $a0, $s2, $zero*****
  220       add   $a1, $s1, $zero
            jal   swap
            addi  $s1, $s1, -1
            j     for2tst
exit2:      addi  $s0, $s0, 1
            j     for1tst
exit1:      lw    $s0, 0($sp)
            lw    $s1, 4($sp)
            lw    $s2, 8($sp)
            lw    $s3, 12($sp)
            lw    $ra, 16($sp)
            addi  $sp, $sp, 20
            jr    $ra
```

If the `lw $t3` instruction five instructions after the `for2tst` label begins executing in cycle 1 and the `beq $t0, $zero, exit2` is taken, what are the values stored in the following fields of the EX/MEM pipeline register in the 10th cycle? Show the instructions being executed in each stage of the pipeline during each cycle. Assume that before the instructions are executed, the state of the machine was as follows:

The PC has the value 200_{10}, the address of the `lw $t3` instruction
Every register has the initial value 20_{10} plus the register number.
Every memory word accessed as data has the initial value 10000_{10} plus the byte address of the word.

Fill in all of the fields, even if the current instruction in that stage is not using them.

EX/MEM.WB = 00
EX/MEM.MEM = 000
EX/MEM.EX = 0000
EX/MEM.TargetAddress = 33116
EX/MEM.Zero = 0
EX/MEM.ALUOut = 38
EX/MEM.ReadData2 = 0
EX/MEM.Write = 4

Cycle	IF	ID	EX	MEM	WB
1	lw$t3				
2	lw$t4	lw$t3			
3	slt$t0	lw$t4	lw$t3		
4	beq$t0	slt$t0	lw$t4	lw$t3	
5	beq$t0	slt$t0	bubble	lw$t4	lw$t3
6	beq$t0	slt$t0	bubble	bubble	lw$t4
7	add$a0	beq$t0	slt$t0	bubble	bubble
8	addi $s0	bubble	beq$t0	slt$t0	bubble
9	j for1tst	addi $s0	bubble	beq$t0	slt$t0
10	lw $s0	j for1tst	addi $s0	bubble	beq$t0

The instruction of interest is add $a0 at 216 that has been turned into a bubble.

8. (10 points) Consider the case of a six-deep pipeline where the branch is resolved at the end of the fourth stage for unconditional branches and at the end of the fifth stage for conditional branches. The program run on this pipeline has the following branch frequencies (as percentages of all instructions) are as follows:

Conditional branches	30%
Jumps and calls	10%
Conditional branches	60% are taken

Assuming that the CPI of the program, neglecting branch hazards, is 1.0, how much slower is the real number, when branch hazards are considered?

j				
after1	j			
after2	after1	j		
after3	after2	after1	j	
target	bubble	bubble	bubble	j

beq					
after1	beq				
after2	after1	beq			
after3	after2	after1	beq		
after4	after3	after2	after1	beq	
target	bubble	bubble	bubble	bubble	beq

The penalty for an unconditional branch being taken is 3 cycles, the penalty for a conditional branch being taken is 4 cycles. An unconditional branch being taken occurs 10% of the time. A conditional branch being taken occurs 0.3*0.6 or 18% of the time.

$CPI_{perfect}$=1.0, $CC_{perfect} = CC_{branchhazards}$, $IC_{perfect} = IC_{branchhazards}$

$CPI_{branch\ hazards} = CPI_{perfect} + CPI_{branch\ penalties} = 1.0 + 0.1*3 + (0.3*0.6*4) = 1.0 + 0.3 + 0.72 = 2.02$

$$\frac{P_{perfect}}{P_{branchhazards}} = \frac{ET_{branchhazards}}{ET_{perfect}} = \frac{IC_{perfect} \times CC_{perfect} \times CPI_{perfect}}{IC_{branchhazards} \times CC_{branchhazards} \times CPI_{branchhazards}} = \frac{CPI_{perfect}}{CPI_{branchhazards}} = \frac{2.02}{1} = 2.02$$

9. (15 points) Here is a series of address references given as word addresses: 0, 4, 16, 132, 232, 160, 1024, 30, 140, 3100, 180, 2180. Assuming a direct-mapped cache with four-word blocks and a total size of 32 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.

$32 words \times \frac{1 block}{4 words} \times \frac{1 set}{1 block} = 8 sets$, index = 3 bits, block offset = 2 bits, byte offset = 0 bits

	Tag	Index	Block Offset	
0	0000 000	000	00	miss
4	0000 000	001	00	miss
16	0000 000	100	00	miss
132	0000 100	001	00	miss
232	0000 111	010	00	miss
160	0000 101	000	00	miss
1024	0100 000	000	00	miss
30	0000 000	111	10	miss
140	0000 100	011	00	miss
3100	1100 000	111	00	miss
180	0000 101	101	00	miss
2180	1000 100	001	00	miss

Index	Tag	Data
0	~~0000 000~~,~~0000 101~~, 0100 000	~~M[0:3]~~, ~~M[160:163]~~, M[1024:1027]
1	~~0000 000~~, ~~0000 100~~, 1000 100	~~M[4:7]~~, ~~M[132:135]~~, M[2180:2183]
2	0000 110	M[232:235]
3	0000 100	M[140:143]
4	0000 000	M[16:19]
5	0000 101	M[180:M183]
6		
7	~~0000 000,~~1100 000	~~M[28:31],~~ M[3100:3103]

10. (10 points) Average and minimum times for reading and writing to storage devices are common measurements used to compare devices. Using techniques from Chapter 6, calculate values related to read and write time for a disk with the following characteristics.

Average Seek Time	RPM	Disk Transfer Rate	Controller Setup Time	Controller Transfer Rate
14 ms	7200	34 Mbytes/s	2 ms	480 Mbits/s

Calculate the average time to read or write a 4096-byte sector for the disk listed.

11. (10 points) The following code is written in MATLAB, where elements within the same column are stored contiguously. References to which variables exhibit spatial locality? A and B are both arrays of integers 8000 by 8000.

```
for J=1:8;
   for I=1:8000;
      A(I,J) = B(J,1) + A(J, I);
```

Variable	I	J	A[I][J]	B[J][1]	A[J][I]
First access			A[1][1]	B[1][1]	A[1][1]
Second access			A[2][1]	B[2][1]	A[1][2]
Spatial locality?	No	No	Yes, the elements in column A[I][1] are accessed sequentially.	Yes, the elements in column B[J][1] are accessed sequentially.	No, all elements of column 1 are stored before the first element of column 2 is stored, these two accesses are at least 8000 elements apart

12. (15 points) (a) Identify all of the data dependencies in the following code. (b) How is each data dependency either handled or not handled by forwarding? **Draw a multiple clock cycle style diagram to support your answer.**

			Dependency	How Handled
a	add	$5, $5, $4	a-b	forward from EX/MEM
b	lw	$5, 28($5)	b-c	stall, then forward from MEM/WB
c	add	$3, $4, $5	b-d	taken care of by stall
d	sw	$3, 100($5)	c-d	forward from EX/MEM
e	add	$7, $3, $4	c-e	forward from MEM/WB