The University of Alabama in Huntsville
Electrical & Computer Engineering Department
CPE 431 01
Test 2 Solution
Fall 2011

Show all work. You will not receive full credit for a problem if you do not show your work!

1. (3 points) The three Cs of cache misses are _compulsory_, _capacity_ and _conflict_.

2. (1 point) Getting a missing data item early from the internal resources of the pipeline is called _forwarding_.

3 (1 point) _F_(True or False) The number of sets in a two-way set associative cache is two.

4. (10 points) Multilevel caching is an important technique to overcome the limited amount of space that a first level cache can provide while still maintaining its speed. Consider a processor with the following parameters.

Base CPI, no memory stalls	Processor speed	Main memory access time	First-level cache miss rate per instruction	Second-level cache, direct-mapped speed	Global miss rate with second-level cache, direct-mapped	Second-level cache, eight-way s et associative speed	Global miss rate with second-level cache, eight-way set associative
2.0	**2.5 GHz**	**125 ns**	**4.8 %**	**15 cycles**	**2.2 %**	**25 cycles**	**1.4 %**

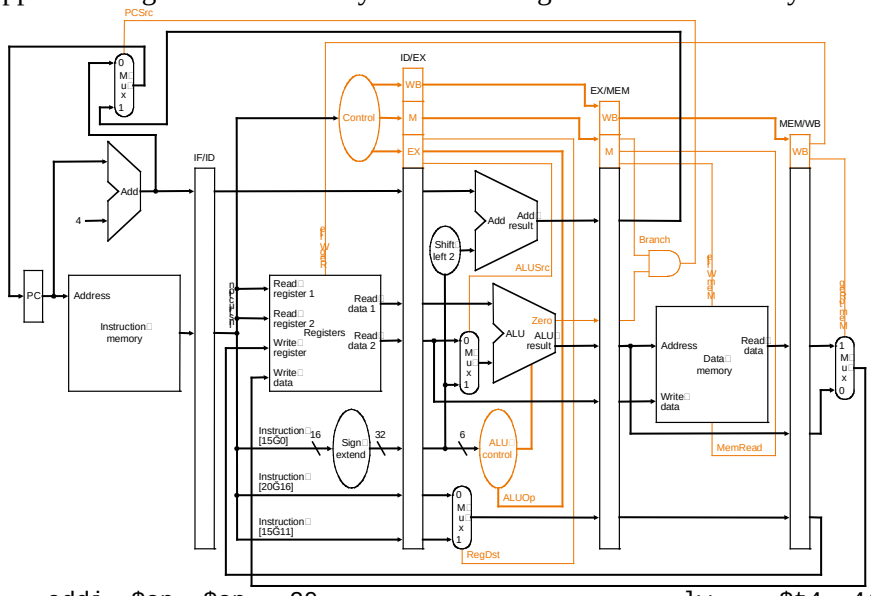
Calculate the CPI for the processor in the table using: 1) only a first-level cache, 2) a second-level direct-mapped cache, and 3) a second-level eight-way set-associative cache.

First-level cache:
a) $CPI_{\text{total L1}} = CPI_{\text{base}} + CPI_{\text{L1 hit}} + CPI_{\text{L1miss}} = 2.0 + 0 + 0.048*125*2.5 = 2.0 + 15 = 17$

Second-level direct-mapped
a) $CPI_{\text{total L1L2}} = CPI_{\text{base}} + CPI_{\text{L1 hit}} + CPI_{\text{L1missL2hit}} + CPI_{\text{L1missL2miss}}$
$= 2.0 + 0 + 0.048*15 + 0.022*125*2.5 = 2.0 + 0.72 + 6.875 = 9.60$

Second-level eight way set-associative
a) $CPI_{\text{total L1L2}} = CPI_{\text{base}} + CPI_{\text{L1 hit}} + CPI_{\text{L1missL2hit}} + CPI_{\text{L1missL2miss}}$
$= 2.0 + 0 + 0.048*25 + 0.014*125*2.5 = 2.0 + 1.2 + 4.375 = 7.58$

5. (10 points) Consider executing the following code on a pipelined datapath like the one shown (no forwarding) except that 1)it supports j instructions that complete 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
        lw    $t3, 0($t2)
        lw    $t4, 4($t2)
        slt   $t0, $t4, $t3
        beq   $t0, $zero, exit2
        add   $a0, $s2, $zero
        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 `addi $s1` instruction one instruction before the `for2tst` label begins executing in cycle 1 and the `bne $t0, $zero, exit2` is taken, what instructions are found in each of the five stages of the pipeline in the 14th 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 `addi $s1` 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.

Cycle	IF	ID	EX	MEM	WB
1	addi $s1				
2	slt $t0	addi $s1			
3	bne $t0	slt $t0	addi $s1		
4	bne $t0	slt $t0	bubble	addi $s1	
5	bne $t0	slt $t0	bubble	bubble	addi $s1
6	add $t1	bne $t0	slt $t0	bubble	bubble
7	add $t1	bne $t0	bubble	slt $t0	bubble
8	add $t1	bne $t0	bubble	bubble	slt $t0
9	add $t1	add $t1	bne $t0	bubble	bubble
10	add $t2	add $t1	add $t1	bne $t0	bubble
11	addi $s0	bubble	bubble	bubble	bne $t0
12	j for1tst	addi $s0	bubble	bubble	bubble
13	lw $s0	j for1tst	addi $s0	bubble	bubble
14	slt $t0	bubble	j for1tst	addi $s0	bubble

6. (10 points) Consider the MIPS five stage pipeline. For the base case, an instruction fetch takes 100 ps, a register read 60 ps, an ALU operation 150 ps, a data access 100 ps, and a register write 60 ps. If the register reads and write times can be shortened by 30 %, will the speedup obtained from pipelining be affected? If yes, by how much? Otherwise, why? What if the register reads and writes now take 30 % more time?

a. If the time for register file reads/writes is reduced to 42 ps, the pipeline needs not be changed because the clock cycle still needs to be 150 ps to accommodate the ALU. The register file time in this scenario is 42 ps (write) + 42 ps (read) = 84 ps, still much less than 150 ps.
b. If the time for register file reads/writes is increased to 78 ps, the pipeline must be changed because the time needed for the register file is greater than 150. The register file time in this scenario is 78 ps (write) + 78 ps (read) = 156 ps. The clock cycle must be adjusted to 156 ps.

Original and 30 % reduction

$$\frac{P_{pipe}}{P_{nonpipe}} = \frac{Timebetweeninstructions_{nonpipe}}{Timebetweeninstructions_{pipe}} = \frac{470\,ps}{150\,ps} = 3.13$$

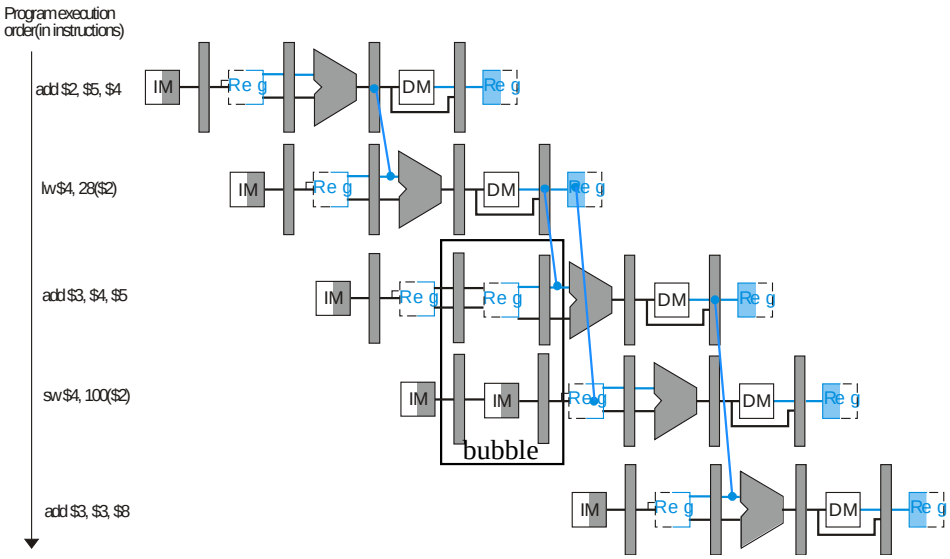
Longer register file read/write

$$\frac{P_{pipe}}{P_{nonpipe}} = \frac{Timebetweeninstructions_{nonpipe}}{Timebetweeninstructions_{pipe}} = \frac{506\,ps}{156\,ps} = 3.24$$

7. (20 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.

```
(a) add $2, $5, $4
(b) lw  $4, 28($2)
(c) add $3, $4, $5
(d) sw $4, 100($2)
(e) add$3, $3, $8
```

Dependences	Handled by Forwarding
(a) – (b)	Forward from EX/MEM
(a) – (d)	Not a hazard, no forwarding needed
(b) – (c)	Stall, then forward from MEM/WB
(b) – (d)	No forwarding necessary after stall
(c) – (e)	Forward from MEM/WB



8. (20 points) Consider the following loop executing on a MIPS pipeline. Assume that the loop always executes an even number of iterations.
a) Calculate the number of cycles it takes to execute this loop 10 times, neglecting pipeline fill cycles under the following conditions.
b) Reorder the loop to optimize it as much as possible under the following conditions.

i) The pipeline has EX/MEM forwarding only
ii) The pipeline has MEM/WB forwarding only.

Assuming one cycle branch stall.
There are three dependencies lw $1 → sw $1, add $5 → sw $1 and lw $1 → beq $1

```
Loop: lw    $1, 40($6)
      add   $5, $5, $8
      add   $6, $6, $8
      sw    $1, 20($5)
      beq   $1, $0, Loop
```

a-i)
```
Loop: lw    $1, 40($6)
      add   $5, $5, $8
      add   $6, $6, $8
      stall
      sw    $1, 20($5)
      beq   $1, $0, Loop
```
9 x 7 + 6 = 69 cycles

b-i)
```
Loop: lw    $1, 40($6)
      add   $6, $6, $8
      add   $5, $5, $8
      sw    $1, 20($5)
      beq   $1, $0, Loop
```
9 x 6 + 5 = 59 cycles

a-ii)
```
Loop: lw    $1, 40($6)
      add   $5, $5, $8
      add   $6, $6, $8
      sw    $1, 20($5)
      beq   $1, $0, Loop
```
9 x 6 + 5 = 59 cycles

b-ii)
```
Loop: lw    $1, 40($6)
      add   $5, $5, $8
      add   $6, $6, $8
      sw    $1, 20($5)
      beq   $1, $0, Loop
No reordering needed.
```
9 x 6 + 5 = 59 cycles

9. (10 points) The following code is written in MATLAB, where elements within the same column are stored contiguously. Consider each variable and answer whether it exhibits spatial locality and whether it exhibits temporal locality. A and B are both arrays of integers 8000 by 8000.

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

Variable	I	J	A(I ,J)	B(J, 1)	A(J, I)
Spatial	Unknown	Unknown	No, elements are accessed by row	Yes, elements are accessed by column	Yes, elements are accessed by column
Temporal	Yes, used 64000 times to access array elements	Yes, used 64000 times to access array elements	No, a different element is accessed each time	Yes, used 8000 times, once each 8 times	No, a different element is accessed each time

10. (15 points) Here is a series of address references given as byte addresses: 0, 4, 16, 131, 232, 160, 1024, 30, 140, 3100, 179, 2180. Assuming a two-way set associative cache with one-word blocks and a total size of 8 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.

$$8words \times \frac{1block}{1word} \times \frac{1set}{2blocks} = 4sets$$

	Index	Byte Offset	
0	000_0000_0000	00	m
4	000_0000_0000	01	m
16	000_0000_0001	00	m
131	000_0000_1000	00	m
232	000_0000_1110	10	m
160	000_0000_1010	00	m
1024	000_0100_0000	00	m
30	000_0000_0001	11	m
140	000_0000_1000	11	m
3100	000_1100_0001	11	m
179	000_0000_1011	00	m
2180	000_1000_1000	01	m

Set	Element 0	Element 1
0	MEM[0:3], MEM[128:131], MEM[1024:1027]	MEM[16:19],MEM[160:163], MEM[176:179]
1	MEM[4:7]	MEM[2180:2183]
2	MEM[232:235]	
3	MEM[28:31]], MEM[3100:3103]	MEM[140:143]