The University of Alabama in Huntsville
Electrical & Computer Engineering Department
CPE 431 01
Final Exam
Fall 2007

1. (10 points) Here is a series of address references given as word addresses: 1, 4, 8, 5, 20, 17, 19, 56, 9, 11, 4, 43, 5, 6, 61. Assuming a two-way set-associative cache with four-word blocks and a total size of 32 words that is initially empty, (a) label each reference in the list as a hit or a miss and (b) show the final contents of the cache. Use an LRU replacement policy.

$$32 words \times \frac{1 block}{4 words} \times \frac{1 set}{2 block} = 4 sets$$

	Tag	Index	Block Offset	Hit/Miss
1	000	00	01	Miss
4	000	01	00	Miss
8	000	10	00	Miss
5	000	01	01	Hit
20	001	01	00	Miss
17	001	00	01	Miss
19	001	00	11	Hit
9	000	10	01	Hit
56	011	10	00	Miss
11	000	10	11	Hit
4	000	01	00	Hit
43	010	10	11	Miss
5	000	01	01	Hit
6	000	01	10	Hit
61	011	11	01	Miss

0	M[0]	M[1]	M[2]	M[3]	M[16]	M[17]	M[18]	M[19]
1	M[4]	M[5]	M[6]	M[7]	M[20]	M[21]	M[22]	M[23]
2	M[8]	M[9]	M[10]	M[11]	~~M[56],~~ M[40]	~~M[57],~~ M[41]	~~M[58],~~ M[42]	~~M[59],~~ M[43]
3	M[60]	M[61]	M[62]	M[63]				

2. (10 points) What size messages would result in ATM outperforming Ethernet by a factor of ten, assuming the following latencies and bandwidths?

Characteristic	Ethernet	ATM
Bandwidth from node to network	1.125 MB/sec	10 MB/sec
Interconnect latency	15 μs	50 μs
HW latency to/from network	6 μs	6 μs
SW overhead sending to network	200 μs	207 μs
SW overhead receiving from network	241 μs	360 μs

For ATM to be ten times as fast as Ethernet, the total message time must be one-tenth that of Ethernet. We can write

$15 + 6 + 200 + 241 + \text{Transmission time}_{\text{Ethernet}} = 10 \times (50 + 6 + 208 + 360 + \text{Transmission time}_{\text{ATM}})$

462 + X/1.125 = 10(623 +X/10)
11.25(462 +X/1.125) = 11.25(6230 + (10X/10)
5197.5 + 10X = 70087.5 + 11.25X
-1.25X = 64890
X = -5192 bytes

The number of bytes can't be negative so there is no message size for which this is possible.

3. (10 points) Suppose we have a system with the following characteristics:
    1. A memory and bus system supporting block access of 4 and 16 32-bit words.
    2. A 64-bit synchronous bus clocked at 200 MHz, with each 64-bit transfer taking 1 clock cycle, and 1 clock cycle required to send an address to memory.
    3. Two clock cycles needed between each bus operation. (Assume the bus is idle before an access.)
    4. A memory access time for the first four words of 150 ns; each additional set of four words can be read in 15 ns.

    Assume that the bus and memory systems described above are used to handle disks that transfer data at 30 MB/sec. If the I/O is allowed to consume 100% of the bus and memory bandwidth, what is the maximum number of simultaneous disk transfers that can be sustained for the two block sizes?

    4 word blocks

Bus	**send**	**idle**	**2 send**	**2 idle**
Memory	**idle**	**30 cycles read**	**4 cycles idle**	

Total = 35 cycles

    For the four-word block transfers, each block takes: 1 cycle to send an address to memory, 150ns/5ns = 30 cycles to read memory, 2 cycles to send the data, 2 idle cycles between transfers, 35 cycles total. At 200 MHz, we have a bus bandwidth of (4 words * 4 bytes/word)/(35 cycles* 5 ns/cycle) = 91.43 MB/sec. The disk delivers data at 30 MB/s so 3 disks can be supported.

    16 word blocks

Bus	**send**	**idle**	**2 send**	**2 idle**	**2 send**	**2 idle**	**2 send**	**2 idle**	**2 send**	**2 idle**
Memory	**idle**	**30 cycles read**	**3 read, 1 idle**		**3 read, 1 idle**		**3 read, idle**		**4 cycles idle**	

Total = 47 cycles

    For the 16-word block transfers, each block takes: 1 cycle to send an address to memory, 150 ns or 30 cycles to read memory, 3 cycles to read the second 4 words. While the second 4 words are being read, two cycles are used to transmit the first 4 words, then one cycle idle time occurs during the read and one more idle cycle is required for the bus. This scenario is repeated two times and then the 2 cycles to send the last 4 words and the required 2 idle cycles between bus transactions occur. At 200 MHz, we have a bus bandwidth of (16 words * 4 bytes/word)/(47 cycles* 5 ns/cycle) = 272.34 MB/sec. The disk delivers data at 30 MB/s so nine disks can be supported.

4. (10 points) Consider program P, which runs on a 2 GHz machine M in 10 seconds. An optimization is made to P, replacing all instances of multiplying a value by 4 (mult X, X, 4) with two instructions that set X to X + X twice (add X, X; add X, X) Call this new optimized program P'. The CPI of a multiply instruction is 5, and the CPI of an add is 1. After recompiling, the program now runs in 8.5 seconds on machine M. How many multiplies were replaced by the new compiler?

    Let x be the number of multiplies.

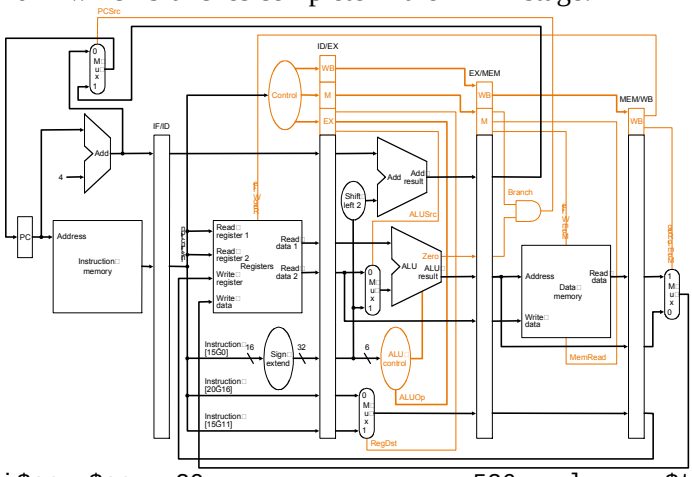
    ET_{before} = 10s, ET_{after} = 8.5s
    $CT_{before} = CT_{after}$ = 0.5 ns
    $IC_{after} = IC_{before}(1 – x + 2x)$, each multiply is replaced by two additions

    $ET_{before} = IC_{before} * CPI_{before} * CT_{before}$
    $Etafter = IC_{after} * CPI_{after} * ET_{after}$

    (1) 10 s = $(IC_{before}(1-x) * CPI_{unaff} + x*5) * 0.5$ ns
    (2) 8.5 s = $(IC_{before}(1-x) * CPI_{unaff} + 2x*1) * 0.5$ ns

    Subtracting (2) from (1) yields 1.5 s = (5x – 2x)*0.5 ns, $3*10^9 = 3$ x, x = 1,000,000,000

5. (15 points) Consider executing the following code on a pipelined datapath like the one shown, except that it has forwarding and in which branches complete in the MEM stage:




```
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
500 for1tst: slt   $t0, $s0, $s3
504       beq $t0, $zero, exit1
508       addi$s1, $s0, -1
512 for2tst: slt  $t0,$s1, $zero
516       bne $t0, $zero, exit2
520       add $t1, $s1, $s1
524       add $t1, $t1, $t1
528       add $t2, $s2, $t1
532       lw  $t3, 0($t2)
536       lw    $t4, 4($t2)
540       slt   $t0, $t4, $t3
544       beq   $t0, $zero, exit2
548       add   $a0, $s2, $zero
552       add   $a1, $s1, $zero
556       jal   swap
560       addi  $s1, $s1, -1
564       j     for2tst
568 exit2:    addi   $s0, $s0, 1
572       j     for1tst
576 exit1:    lw    $ra, 16($sp)
          lw    $s0, 0($sp)
          lw    $s1, 4($sp)
          lw    $s2, 8($sp)
          lw    $s3, 12($sp)
          addi  $sp, $sp, 20
          jr    $ra
```

If the `slt $t0` instruction at the `for1tst` label begins executing in cycle 1 and the `beq $t0, $zero, exit1` is not taken and the `bne $t0, $zero, exit2` is not taken and the `beq $t0, $zero, exit2` is taken, what are the values stored in the following fields of the ID/EX pipeline register in the 16th cycle? Assume that before the instructions are executed, the state of the machine was as follows:

The PC has the value 500_{10}, the address of `slt $t0` (at the label `for1tst`).

Every register has the initial value 20_{10} plus the register number.

Every memory word accessed as data has the initial value 2000_{10} plus the byte address of the word.

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

Cycle	IF	ID	EX	MEM	WB
1	slt $t0				
2	beq $t0	slt $t0			
3	addi $s1	beq $t0	slt $t0		
4	slt $t0	addi $s1	beq $t0	slt $t0	
5	bne $t0	slt $t0	addi $s1	beq $t0	slt $t0
6	add $t1	bne $t0	slt $t0	addi $s1	beq $t0
7	add $t1	add $t1	bne $t0	slt $t0	addi $s1
8	add $t2	add $t1	add $t1	bne $t0	slt $t0

9	lw $t3	add $t2	add $t1	add $t1	bne $t0
10	lw $t4	lw $t3	add $t2	add $t1	add $t1
11	slt $t0	lw $t4	lw $t3	add $t2	add $t1
12	beq $t0	slt $t0	lw $t4	lw $t3	add $t2
13	beq $t0	slt $t0	bubble	lw $t4	lw $t3
14	add $a0	beq $t0	slt $t0	bubble	lw $t4
15	add $a1	add $a0	beq $t0	slt $t0	bubble
16	jal swap	add $a1	add $a0	beq $t0	slt $t0

The instruction of interest is add $a0, $s2, $zero.

ID/EX.WB = 10_2
ID/EX.MEM = 000_2
ID/EX.EX = 1100_2
ID/EX.PCInc = 552_{10}
ID/EX.ReadData1 = 38_{10}
ID/EX.ReadData2 = 0_{10}
ID/EX.SignExtend = 00002020_{16}
ID/EX.WriteRt = 0_{10}
ID/EX.WriteRd = 4_{10}

6. (5 points) A binary word with odd parity and no errors will have an odd number of 1s in it. Compute the parity bit for each of the following 8-bit words so that the resulting 9-bit word has odd parity.

    a. 01100111, P = 0, leaving the total number of ones 5, an odd number
    b. 01010101, P = 1, making the total number of ones 5, an odd number

7. (1 point) The alternative model to message passing for parallel processing is __shared memory__.
8. (1 point) The term _parallel processing_ refers to a single program that runs on several processors simultaneously.
9. (1 point) A __barrier__ is a synchronization scheme in which processors wait and do not proceed until every process is ready.
10. (1 point) ___Clusters__ are networks of off-the-shelf, whole computers.
11. (1 point) A __cache coherency__ protocol maintains consistency in the value of data between several processors.
12. (10 points) Represent 52419.5625 as a single precision floating point number.

    $52419_{10} = 32768_{10} + 16384_{10} + 2048_{10} + 1024_{10} + 128_{10} + 64_{10} + 2_{10} + 1_{10}$
    $= 2^{15} + 2^{14} + 2^{11} + 2^7 + 2^6 + 2^1 + 2^0 = 1100100011000011_2$
    $0.5625_{10} = 0.1001_2$ So, $52419.5625_{10} = 1100100011000011.1001_2$
    Normalizing gives $1.10010001100001110001 \times 2^{15}$

    For single precision, we have the sign bit, an 8-bit exponent, and a 23-bit fraction. The bias is 127. 15 = Exponent-Bias, Exponent = 15 + 127 = 142.
    So, the sign bit is 0, the fraction is 1001 0001 1000 0111 0010 000, and the exponent is 10001110

13. (5 points) Consider a virtual memory system with the following properties:
    - 64-bit virtual byte address
    - 128-KB pages
    - 36-bit physical byte address

What is the total size of the page table for each process on this machine, assuming that the valid, protection, dirty, and use bits take a total of 4 bits and that all the virtual pages are in use? (Assume that disk addresses are not stored in the page table.)

For 128 Kbyte pages, that means a page offset of 17 bits. 64 –17 is 47, so there are 2^{47} virtual pages. Each entry in the page table is 36 – 17 for a page number + 6 bits of bookkeeping.

So, the size is $2^{47} * (19 + 4) = 2^{47} * 23$ bits

14. (10 points) Show the single MIPS instruction or minimal sequence of instructions for this C statement:

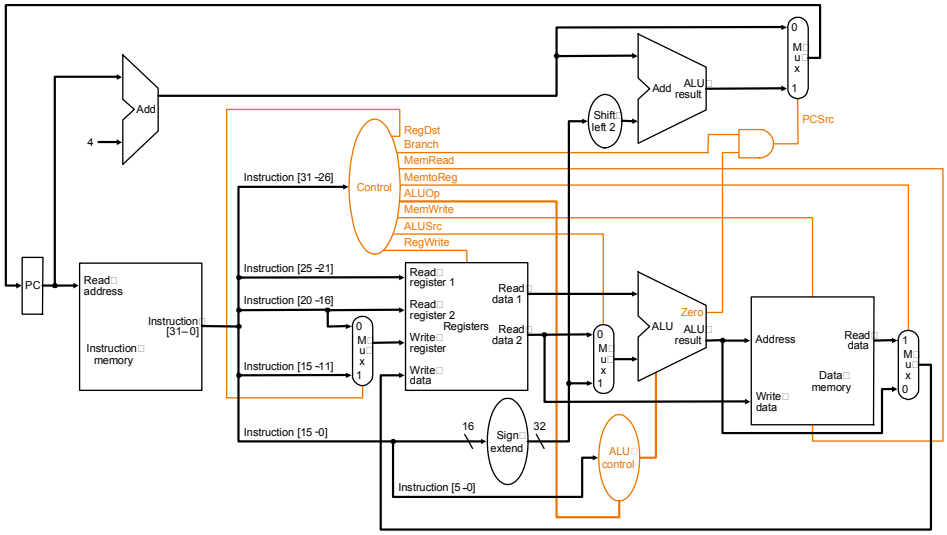
```
z[10] = x[10] + y[10];
```

Assume that array `x` (an array of 32-bit integers) has a base address of $4{,}000{,}000_{10}$ which is stored in register `$t0`, that y. (an array of 32-bit integers) has a base address of $5{,}000{,}000_{10}$ which is stored in register `$t1`, and that z (an array of 32-bit integers) has a base address of $6{,}000{,}000_{10}$ which is stored in register `$t2`.

```
lw $t3, 40($t0)
lw $t4, 40($t1)
add $t4, $t3, $t4
sw $t4, 40($t2)
```

15. (10 points) Consider each of the following stuck-at-1 faults separately: RegDst = 1, ALUSrc = 1, MemtoReg = 1, RegWrite = 1. Which instructions, if any would still work?

	RegDst = 1	ALUSrc = 1	MemtoReg = 1	RegWrite= 1
R-type	Works	Doesn't Work	Doesn't Work	Works
lw	Doesn't Work	Works	Works	Works
sw	Works	Works	Works	Doesn't Work
beq	Works	Doesn't Work	Works	Doesn't Work



Instruction	RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0
R-format	1	0	0	1	0	0	0	1	0
lw	0	1	1	1	1	0	0	0	0
sw	d	1	d	0	0	1	0	0	0
beq	d	0	d	0	0	0	1	0	1