The University of Alabama in Huntsville
Electrical & Computer Engineering Department
CPE 431 01
Final Exam Solution
Fall 2004

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 two-word blocks and a total size of 16 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.

$16 words * \frac{1 block}{2 words} * \frac{1 set}{2 blocks} = 4 sets$ Block Offset - 1 bit, Index - 2 bits

	tag, index, block offset				
1	000 00 1	m	9	001 00 1	m
4	000 10 0	m	11	001 01 1	h
8	001 00 0	m	4	000 10 0	h
5	000 10 1	h	43	101 01 1	m
20	010 10 0	m	5	000 10 1	h
17	010 00 1	m	6	000 11 0	m
19	010 01 1	m	61	111 10 1	m
56	111 00 0	m			

0	~~0~~, 16	~~1~~, 17	~~8~~, 56	~~9~~, 57
1	~~18~~, 42	~~19~~, 43	10	11
2	4	5	~~20~~, 60	~~21~~, 61
3	6	7		

2. (15 points) A program repeatedly performs a three-step process: It reads in a 16-KB block of data from disk, does some processing on that data, and then writes out the result as another 16-KB block elsewhere on the disk. Each block is contiguous and randomly located on a single track on the disk. The disk drive rotates at 19200 RPM, has an average seek time of 8 ms, and has a transfer rate of 128 MB/sec. The controller overhead is 2 ms. No other program is using the disk or processor, and there is no overlapping of disk operation with processing. The processing step takes 50 million clock cycles, and the clock rate is 1 GHz. What is the overall speed of the system in blocks processed per second?

Disk Access = seek + rotational latency + controller + transfer
= 8 ms + 0.5 rotations * 1 minute/ 19200 rotations * 60 s/min + 2 ms + 16 KB/128MB/s
= 8 ms + 1.56 ms + 2 ms + 0.1 ms
= 11.7 ms

Since each block processed involves two accesses (read and write), the disk component of the time is 23.4 ms per block processed. The (non-overlapped) computation takes 50 million clock cycles at 1 GHz, or another 50 ms. Thus, the total time to process one block is 73.4 ms, and the number of blocks processed per second is simply 1/0.0734 = 13.6.

3. (10 points) Consider a virtual memory system with the following properties:
   - 64-bit virtual byte address
   - 64-KB pages
   - 48-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.)

$$\text{\# of virtual pages} = \frac{2^{64}}{2^{16}} = 2^{48}$$

#bits in physical page address = 48-16 = 32
#total bits = $2^{48} \cdot (32+4) = 2^{48} \cdot 36$

4. (15 points) Consider a write-back cache used for a processor with a bus and memory system as described in the book on page 665 (assume that writes require the same amount of time as reads). The time to transfer 8-word blocks is 49 cycles and the time to transfer 16-word blocks is 57 cycles. Additionally, the following performance measurements have been made:

   The cache miss rate is 0.05 misses per instruction for block sizes of 8 words.
   The cache miss rate is 0.03 misses per instruction for block sizes of 16 words.
   For either block size, 40% of the misses require a write-back operation, while the other 60% require only a read.

Assuming that the processor is stalled for the duration of a miss (including the write-back time if a write-back is needed), find the number of cycles per instruction that are spent handling cache misses for each block size. (Hint: First compute the miss penalty.)

The approach is the same for both block sizes, though the miss rates and miss penalties vary. For a miss which doesn't require a write-back, the penalty is one block transfer. For a miss requiring a write-back, the penalty is two block transfers (one to do the write-back, one to get the new block). So, the number of cycles per instruction spent handling cache misses = miss ratio*(0.4*2*block transfer time + 0.6*block transfer time)

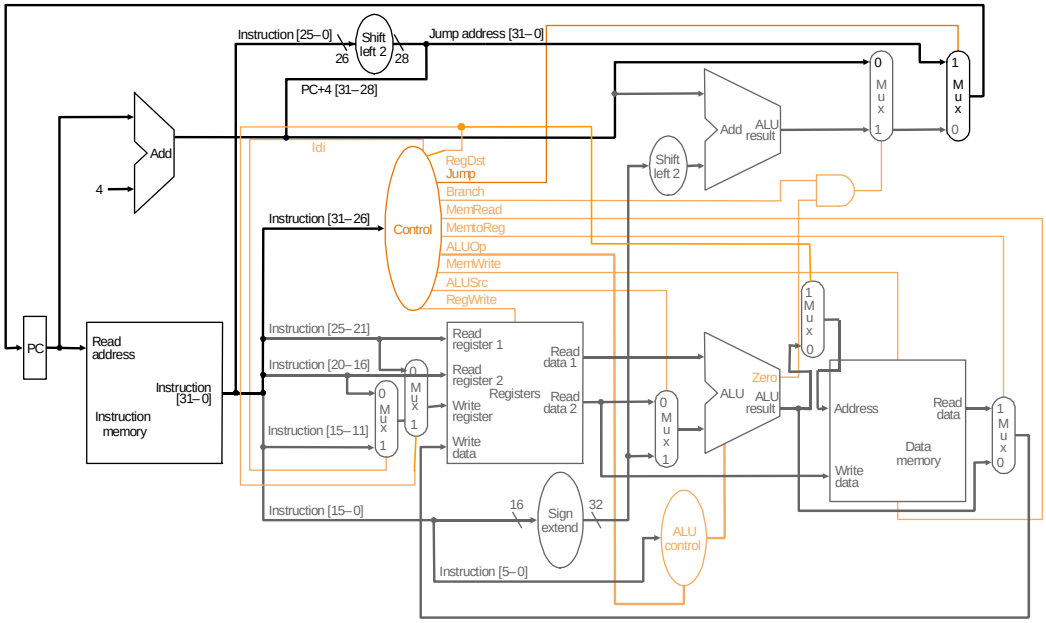
8 words Cache miss CPI = 0.05*(0.4*2*49 + 0.6*49) = 0.05*(1.4*49) = 3.43
16 words Cache miss CPI = 0.03*(0.4*2*57 + 0.6*57) = 0.03*(1.4*57) = 2.39

5. (10 points) Add the instruction `ldi` (load immediate) to the single-cycle datapath shown in the figure below. The `ldi` instruction loads a 32-bit immediate value from the memory location following the instruction address. Add any necessary datapaths and control signals and show the necessary additions to the table of control signals given.

`ldi` $t0 $t0 = Mem[PC+4]

2	rt	1	0

The only solution that makes sense in the single cycle datapath is one in which the immediate value is fetched from the data memory. This solution doesn't really make sense, the immediate value resides in the instruction memory. Fetching it from the instruction memory would require multiple cycles.



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

d-don't care

6. (5 points) Show the single MIPS instruction or minimal sequence of instructions for this C statement:

```
x[10] = x[11] + c;
```

Assume that `c` corresponds to register `$t0` and the array `x` has a base address of $4{,}000{,}000_{10}$ which is stored in register `$t1`.

```
lw    $t2, 44($t1)
add   $t2, $t2, $t0
sw    $t2, 40($t1)
```

7. (10 points) The MicroBlaze embedded soft core is a reduced instruction set computer (RISC) optimized for implementation in Xilinx field programmable gate arrays (FPGAs). It has a 32-bit instruction word with three operands and two addressing modes. MicroBlaze instructions are either Type A or Type B. The instruction formats for each are given below:

Bits	0-5	6-10	11-15	16-20	20-31
Type A	opcode	Rd	Ra	Rb	00000000000
Type B	opcode	Rd	Ra	Immediate	
				16-31	

For arithmetic/logical instructions, we have:

Rd ← Ra op Rb, where op is +, -, AND, OR, etc. `add Rd, Ra, Rb`

For loads:

Rd ← MEM[Ra + Rb] `lw Rd, Ra, Rb`

or Rd ← MEM[Ra + Immediate] `lw Rd, Ra, Imm`

For stores:

MEM[Ra + Rb] ← Rd	`sw Rd, Ra, Rb`
MEM[Ra + Immediate] ← Rd	`sw Rd, Ra, Imm`

For beq:

PC ← Rb if Ra = 0	`beq Ra, Rb`

The MicroBlaze has a 3 stage pipeline as follows:

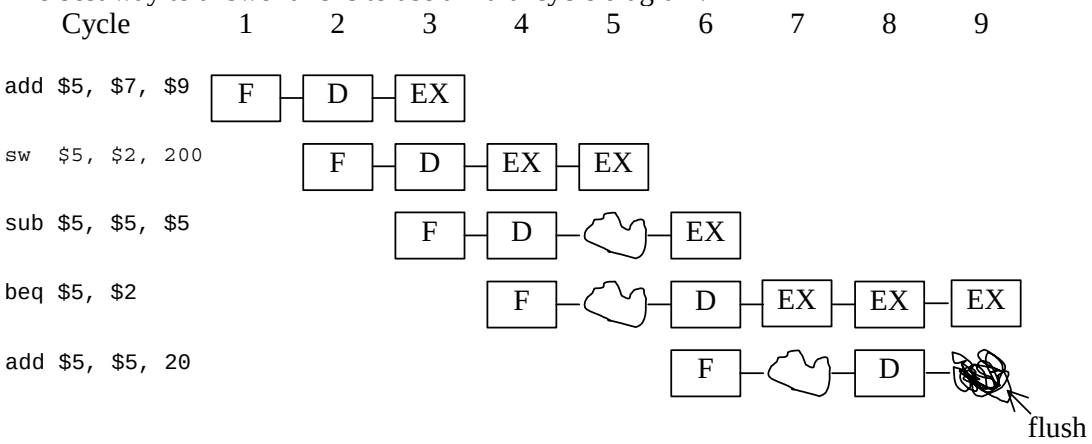
Fetch	Decode	Execute

The register file is read in the Decode state and the ALU and memory operations take place in the Execute stage. Even without data hazards, the pipeline has stalls. Loads and stores require an additional cycle to complete (a total of 4) and beq requires two extra cycles to complete (a total of 5) if taken. If the beq is not taken, no stalls are required.

For the following MicroBlaze code, how many cycles will it take to execute?

```
add  $5, $7, $9
sw   $5, $2, 200
sub  $5, $5, $5
beq  $5, $2
add  $5, $5, 20
```

The best way to answer this is to use a multi-cycle diagram.



Another way to look at it is:

Cycle	Fetch	Decode	Execute
1	`add $5`		
2	`sw $5`	`add $5`	
3	`sub $5`	`sw $5`	`add $5`
4	`beq $5`	`sub $5`	`sw $5`
5	`beq $5`	`sub $5`	`sw $5`
6	`add $5`	`beq $5`	`sub $5`
7	after 1	`add $5`	`beq $5`
8	after 1	`add $5`	`beq $5`
9	after 1	`add $5`	`beq $5`
10	target	`bubble`	`bubble`

Either way, it takes 9 cycles.

8. (5 points) What number does the two's complement binary number represent:
   $1111\ 1110\ 0000\ 1100_2$?

$-2^{15}+2^{14}+2^{13}+2^{12}+2^{11}+2^{10}+2^{9}+2^{3}+2^{2}$= -32768+16384+8192+4096+2048+1024+512+8+4 = -500

9. (10 points) Consider program P, which runs on a 1 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 4, and the CPI of an add is 1. After recompiling, the program now runs in 9 seconds on machine M. How many multiplies were replaced by the new compiler?

ET_{new} = 9s, ET_{orig}= 10s, amount of improvement = 2 (4 cycles for one mult vs. 2 1 cycle adds)
$ET_{new} = ET_{unaffected} + ET_{affected}$/amount of improvement
$ET_{unaffected} = ET_{orig} - ET_{affected}$ = 10 s - $ET_{affected}$

Substituting,

9 s = 10 s - $ET_{affected} + ET_{affected}/2$
0.5* $ET_{affected}$ = 1 s
$ET_{affected}$ = 2 s

Number of multiplies = $ETaffected * \frac{cycles}{s} * \frac{1mult}{4cycles} = 2s * \frac{1\times10^{9}\,cycles}{s} * \frac{1mult}{4cycles} = 500\times10^{6}$

10. (1 point) The alternative model to shared memory for communicating uses __message passing__ between processors.

11. (1 point) The coordination needed between processors operating in parallel on shared data is called ____synchronization_.

12. (1 point) ___Communication_ is the hardest problem.

13. (1 point) A rectangular component that results from dicing a wafer is called a _die_.

14. (1 point) A byte consists of __eight_ bits.

15. (5 points) An alternative scheme to branch prediction to reduce the cost of control hazards is called delayed branch. In a delayed branch, the execution cycle with a branch delay of length 1 is
branch instruction
sequential successor
branch target if taken

The sequential successor is in the branch-delay slot. That is, the seqential successor follows the branch in the program but is actually executed before the branch is taken. The job of the software is to make the successor instruction valid and useful. For the following code, rewrite it for delayed branch and in the presence of the forwarding unit in the text that operates in the EX stage of the pipeline.

```
Loop:   lw      $10, 0($1)
        add     $4, $10, $2
        sw      $4, 0($1)
        subi    $1,$1, 4
        bnez    $1, Loop
```

```
Loop:   lw      $10, 0($1)
        subi    $1, $1, 4
        add     $4, $10, $2
        bnez    $1, Loop
        sw      $4, 4($1)
```