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

1. (1 point) _**Hot swapping**_ is replacing a hardware component while the system is running
2. (1 point) _**Clusters**_ are networks of off-the-shelf, whole computers.
1. (1 point) For _**combinational**_ elements, their outputs depend only on the current inputs.
4. (1 point) A _**fully associative**_ cache structure is one in which a block can be placed in any location in the cache.
5. (1 point) _**Instructions**_ are individual commands to a computer.

6. (10 points) Consider the following portions of two different programs running at the same time on three processors in a symmetric multicore processor (SMP). Assume that before this code is run, w is 3, x is 5 and y and z are 1. w, x, y, and z are type `int`.

   Core 1: `y = 5/z;`
   Core 2: `x = (x + y)/w + 1;`
   Core 3: `z = w*x + y;`

	Resulting Values			
Possible Sequences	**w**	**x**	**y**	**z**
	3	**5**	**1**	**1**
1, 2, 3	**3**	**4**	**5**	**17**
1, 3, 2	**3**	**4**	**5**	**16**
2, 1, 3	**3**	**3**	**5**	**14**
2, 3, 1	**3**	**3**	**0**	**10**
3, 1, 2	**3**	**2**	**0**	**16**
3, 2, 1	**3**	**3**	**0**	**16**

What are all the possible resulting values of w, x, y, and z? Show all possible interleavings of instructions and the resulting values of w, x, y, and z.

7. (5 points) Sometimes software optimization can dramatically improve the performance of a computer system. Assume that a CPU can perform a multiplication operation in 10 ns, and a subtraction operation in 1 ns. (a) How long will it take for the CPU to calculate the result of d = a × b – a × c? (b) Could you optimize the equation so that it will take less time? (c) If so, what is the resulting speedup?

a) d= a × b – a × c requires two multiplications and one subtraction,
Time = 2*10 ns + 1 ns = 21 ns.

b) It can be rewritten d = a × (b – c).
As such, it requires one subtraction and then one multiplication, Time = 1 ns + 10 ns = 11ns
Speedup = ET_{old}/ET_{new} = 21 ns/11 ns = 1.9

8. (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. 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 entire history of the cache

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

	Tag	Index	Block Offset	
1	0000 0000	00	01	miss
4	0000 0000	01	00	miss
8	0000 0000	10	00	miss
5	0000 0000	01	01	hit
20	0000 0001	01	00	miss
17	0000 0001	00	01	
19	0000 0001	00	11	
56	0000 0011	10	00	
9	0000 0000	10	01	
11	0000 0000	10	11	
4	0000 0000	01	00	
43	0000 0010	10	11	
5	0000 0000	01	01	
6	0000 0000	01	10	

Index	Tag	Data	Tag	Data
0	0x0000 .. 00	M[0:3]		
1	0x0000 .. 00	M[4:7]	0x0000 .. 01	~~M[20:23]~~
2	0x0000 .. 00	~~M[8:11]~~		
3				

9. (10 points) Given your understanding of PC-relative addressing, explain why an assembler might have problems directly implementing the branch instruction in the following code sequence:

```
here:     beq   $s0, $s2, there
…
there     add   $s0, $s0, $s0
```

Show how the assembler might rewrite this code sequence to solve these problems.

If `here` and `there` are more than $\pm 2^{16}$ words apart, beq cannot be used.
It can be rewritten as

```
here:     bne $s0, $s2, nobranch
          j there
nobranch: next statement
```

10. (5 points) What is the value of the following 32 bits if they represent a single precision floating point number?

0xDB348880 = `1 10110110 01101001000100010000000`
Sign bit = 1, Exponent = 10110110 = 182, Mantissa = 01101001000100010000000
$(-1)^1 \times (1 + 2^{-2} + 2^{-3} + 2^{-5} + 2^{-8} + 2^{-12} + 2^{-16}) \times 2^{(182 - 127)}$
$-1 \times (1.410415649) \times 2^{55}$
$-5.081557915 \ 10^{16}$

11. (10 points) A secret agency simultaneously monitors cellular phone conversations and multiplexes the data onto a network with a bandwidth of 5 MB/sec and an overhead latency of 150 μs per 1 KB message. Calculate the transmission time per message and determine whether there is sufficient bandwidth to support this application. Assume that the phone conversation data consists of 2 bytes sampled at a rate of 4 KHz.

$$\frac{2bytes}{1sample} \times \frac{4000samples}{1\text{sec}} = 8000\frac{bytes}{\text{sec}}$$

In one second, Transmission = 8000 bytes(150 μs/1024 bytes) + 8000 bytes/(5 MB/sec) = 1.172 ms + 1.6 ms = 2.772 ms of transmission time, ∴ there is sufficient bandwidth.

12. (10 points) Assume you are configuring a Sun Fire x4150 server and assume that this configuration contains four processors. Determine whether configurations of 4, 8, and 16 disks present an I/O bottleneck.

Program Instructions Per I/O Operation	OS Instructions Per I/O Operation	Workload (KB reads)	Processor Speed (Instructions/Second)
500,000	100,000	32	1 Billion

The seek time for the disks is 2.9 ms. Consider sequential reads and writes. The disks rotate at 15,000 RPM and have a sustained transfer rate of 112 MB/s. Assume that the bandwidth of all other elements is sufficient to sustain the I/O rate of the processors and the disks.

Disk Operations = 112 MB/s/(32 KB/I/O) = 3500 I/Os/sec

4 disks = 14000 I/Os/sec
8 disks = 28000 I/Os/sec
16 disks = 56000 I/Os/sec

Processors: 4 processors × (1 billion instructions/sec)/(600,000 instructions/I/O) = 6666.7 I/Os/sec

All of the disk configurations can handle 6666.7 I/Os/sec, so none of them present a bottleneck.

13. (10 points) There are several parameters that have an impact on the overall size of a page table. Listed below are several key page table parameters.

Virtual address size	Page size	Page table entry size
32 bits	4 KB	4 bytes

Calculate the total page table size for a system running five applications that utilize half of the memory available.

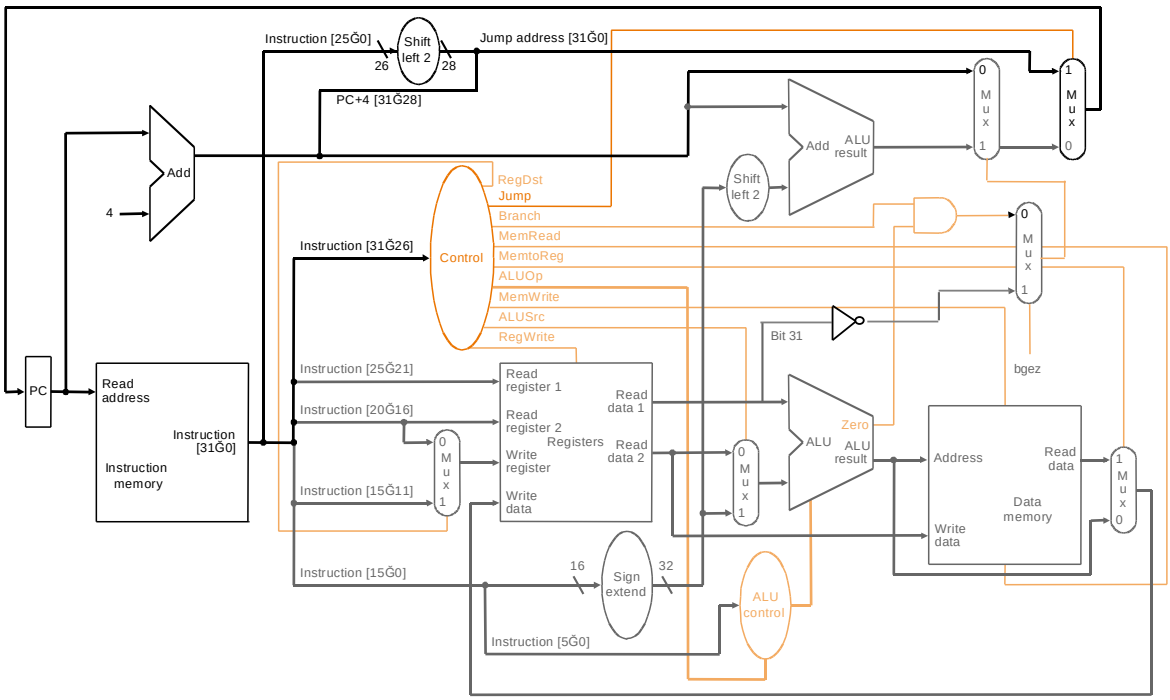
#page table entries for total address space = $2^{32}/2^{12} = 2^{20}$
Size of page table = (# entries * size of entry)/2 = (2^{20} * 4 bytes)/2 = 2 MB
Total page table size = page table size * number of applications = 2 MB * 5 = 10 MB

14. (10 points) Add the instruction `bgez` (branch on greater than or equal to zero) to the single-cycle datapath shown in the figure below. The `begz` instruction is defined below. Add any necessary datapath elements and control signals and show the necessary additions to the table of control signals given.

`begz rs, label`

if (rs >= 0)
PC ← PC + 4 + 4*offset
else
PC ← PC + 4

1	rs	1	offset



Instruction	RegDst	ALUSrc	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUOp1	ALUOp0	bgez
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**
bgez	**d**	**d**	**d**	**0**	**0**	**0**	**d**	**d**	**d**	**1**

d-don't care

15. (15 points) Unroll the following code so that three iterations of the loop are done at once. Assume the loop index is a multiple of 3 (i.e., $10 is a multiple of twelve):

```
Loop: lw    $2, 0($10)
      sub   $4, $2, $3
      addi  $10, $10, 4
      sw    $4, 0($10)
      bne   $10, $30, Loop
```

Schedule this code for fast execution on a MIPS pipeline that has EX/MEM pipelining only. Assume initially $10 is 0 and $30 is 480 and that branches are resolved in the ID stage. How does the unrolled, scheduled code compare against the original code in terms of total execution time?

```
Loop: lw    $2, 0($10)
      stall
      stall
      addi  $10, $10, 4
      sub   $4, $2, $3
      stall
      stall
      sw    $4, 0($10)
      bne   $10, $30, Loop
      stall
```

Total cycles = 119 iterations * 10 cycles + 1 iteration * 9 cycles = 1,199 cycles

```
Loop:    lw    $2, 0($10)
         addi  $10, $10, 12
         stall
         sub   $4, $2, $3
         sub   $5, $4, $3
         sub   $6, $5, $3
         sw    $4, -8($10)
         sw    $5, -4($10)
         sw    $6, 0($10)
         bne   $10, $30 Loop
         stall
```

Total cycles = 39 iterations * 11 cycles + 1 iteration * 10 cycles = 439 cycles

Speedup = 1,199 cycles/439 cycles = 2.73