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

1. (10 points) Here is a series of address references given as word addresses: 2, 3, 11, 16, 21, 13, 64, 48, 19, 11, 3, 22, 4, 27, 6, and 11. Assuming a direct mapped cache with two-word blocks and a total size of 8 words that is initially empty, (a) label each reference in the list as a hit or a miss and (b) show all entries made to the cache, including the final contents of the cache.

$$8words * \frac{1block}{2words} * \frac{1set}{1block} = 4sets$$

Block Offset - 1 bit, Index - 3 bits

Index Block Offset

2	0000 01 0	m
3	0000 01 1	h
11	0001 01 1	m
16	0010 00 0	m
21	0010 10 1	m
13	0001 10 1	m
64	1000 00 0	m
48	0110 00 0	m
19	0010 01 1	m
11	0001 01 1	m
3	0000 01 1	m
22	0010 11 0	m
4	0000 10 0	m
27	0011 01 1	m
6	0000 11 0	m
11	0001 01 1	m

0	~~16~~, ~~64~~, 48	~~17~~, ~~65~~, 49
1	~~2~~, ~~10~~, ~~18~~, ~~10~~, ~~2~~, ~~26~~, 10	~~3~~, ~~11~~, ~~19~~, ~~11~~, ~~3~~, ~~27~~, 11
2	~~20~~, ~~12~~, 4	~~21~~, ~~13~~, 5
3	~~22~~, 6	~~23~~, 7

2. (10 points) Suppose we want to use a laptop to send 100 files of 25 MB each to another computer over a 5 Mbit/sec wireless connection. The laptop battery currently holds 60,000 joules of energy. The wireless networking card alone consumes 5 watts while transmitting, while the rest of the laptop always consumes 20 watts. Before each file transfer we need 15 seconds to choose which file to send. How many complete files can we transfer before the laptop's battery runs down to zero?

Time to choose = 15 s

$$\text{Time to send} = \frac{25MB\frac{8bits}{1byte}}{5Mbits/s} = 40s$$

Energy/file = Energy while choosing * Time to choose + Energy while sending * Time to send
= 20 w * 15 s + 25 w * 40 s = 300 + 1000 = 1300 joules

Number of files = Energy/energy/file = 60,000/1300 = 46 files

3. (10 points) A secret agency wants to simultaneously monitor cellular phone conversations by sampling 2 bytes of data 4000 times each second for each conversation and bundling these samples into 1KB messages for transmission. The overhead latency is 150 μs per 1 KB message. Calculate the bandwidth required of the network to support monitoring 100 conversations.

Data required for 100 conversations = 100 * 4000 samples/s * 2 bytes/sample = 800 Kbytes/s

Time required for 100 conversations = Latency + Transmission time

$1s = 100 * 150\ \mu s + \frac{800kbytes}{x}$, where x is the bandwidth of the network in bytes/s

$1s = 15\ ms + \frac{800kbytes}{x}$

$885\ ms = \frac{800kbytes}{x}$

$x = \frac{800kbytes}{885ms} = 0.9Mbytes/s$

4. (10 points) To capture the fact that the time to access data for both hits and misses affects performance, designers often use average memory access time (AMAT) as a way to examine alternative cache designs. Average memory access time is the average time to access memory considering both hits and misses and the frequency of different accesses; it is equal to the following:

AMAT = Time for a hit + Miss rate × Miss penalty

(a) Find the AMAT for a processor with a 2 ns clock, a miss penalty of 20 clock cycles, a miss rate of 0.05 misses per reference, and a cache access time (including hit detection) of 1 clock cycle. Assume that the read and write miss penalties are the same and ignore other write stalls. (b) Suppose we can improve the miss rate to 0.03 misses per reference by doubling the cache size. This change causes the cache access time to increase to 1.2 clock cycles. Using the AMAT as a metric, determine whether this is a good trade-off.

(a) $AMAT_{orig} = 2\ ns + 0.05*40\ ns = 2\ ns + 2\ ns = 4\ ns$

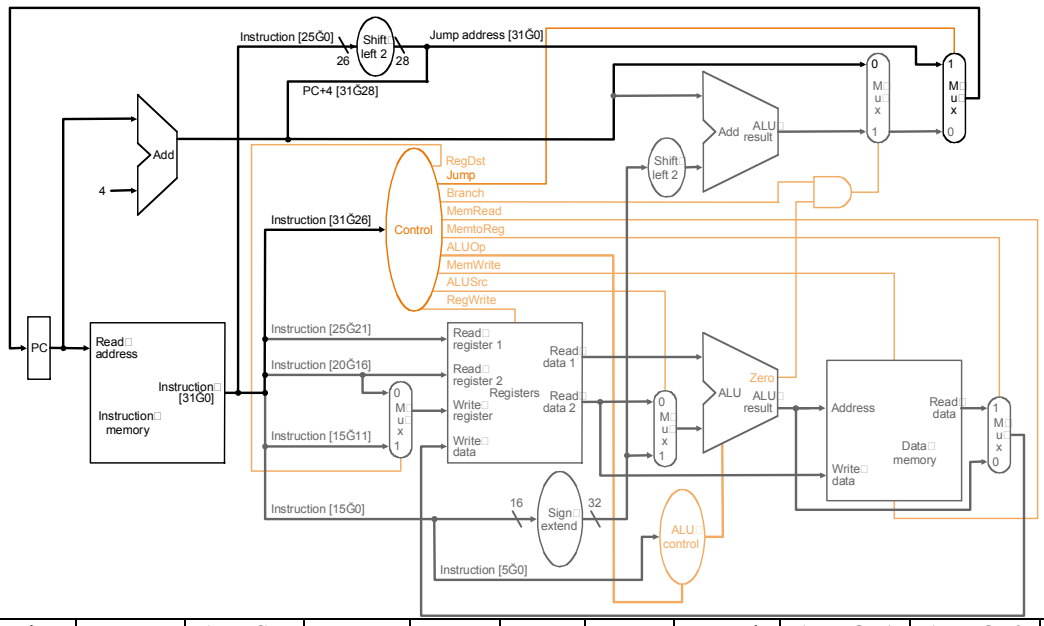
(b) $AMAT_{modified} = 1.2 * 2\ ns + 0.03 * 40\ ns = 2.4\ ns + 1.2\ ns = 3.6\ ns$

Using AMAT as the metric, the tradeoff is worthwhile.

5. (15 points) Consider the following idea: Let's modify the instruction set architecture and remove the ability to specify an offset for memory access instructions. Specifically, all load-store instructions with nonzero offsets would become pseudoinstructions and would be implemented using two instructions. For example:

```
addi $at, $t1, 104       # add the offset to a temporary
lw   $t0, $at            # new way of doing lw $t0, 104 ($t1)
```

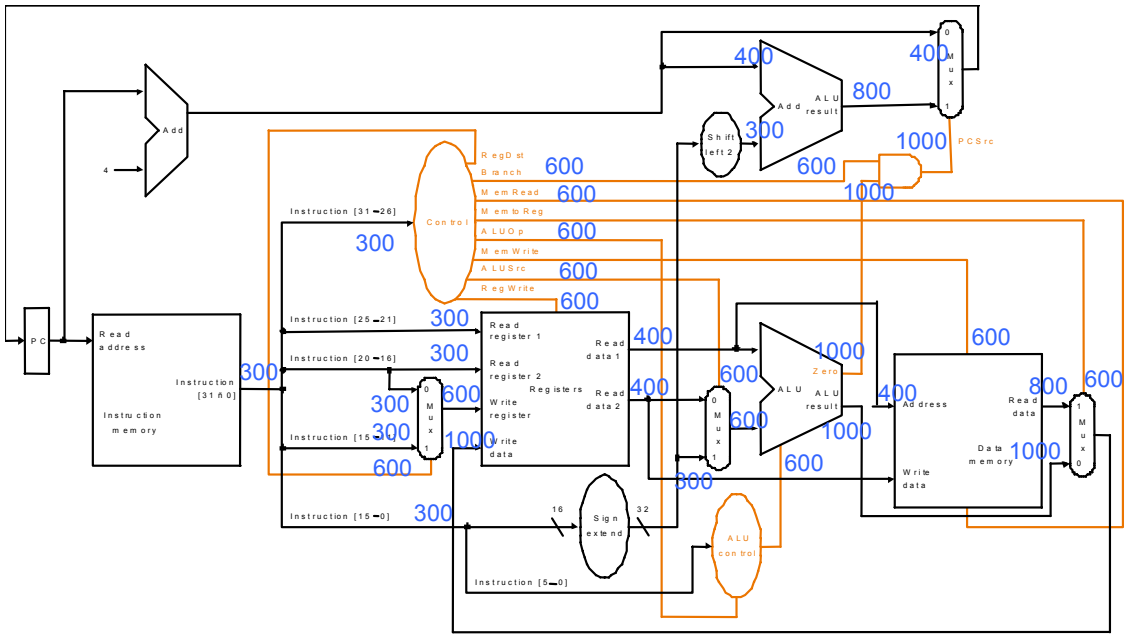
(a) (5 points) What changes would you want to make to the single-cycle datapath and control if this simplified architecture were to be used?



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	

d-don't care

(b) (10 points) If the delay for the instruction memory is 300 ps, the data memory is 200 ps, the register file delay (read or write) is 100 ps, the control (not ALU control) is 300 ps, and the ALU delay is 400 ps, what are the clock cycle times required for the original datapath and for the modified datapath? What is the highest percentage of load-store instructions with offsets that could be tolerated without total performance being degraded?



`lw` is the longest for the unmodified datapath

Consider `lw`, `sw`, R-type, `beq`, `jump` for modified datapath

CC_{lw} = 300(fetch)+max(300, 100)(control, register file read)+200(data memory read)+100(register file write) = 300 + 300 + 200 + 100 = 900 ps

CC_{sw} = 300(fetch) + max(300, 100)(control, register file read) + 200(data memory write) = 300 + 300 + 200 = 800 ps

CC_{Rtype} = 300(fetch) + max(300, 100)(control, register file read) + 400(ALU) + 100(register file write) = 300 + 300 + 400 + 100 = 1100 ps

Cc_{beq} = max((300(fetch) + max(300, 100)(control, register file read) + 400(ALU)), max(400, 300)(ADD, control) + 400 (ALU)) = max(300+300+400, 400+400) = max(1000, 800) = 1000 ps

Cc_{jump} = 300(fetch) = 300 ps

CC_{orig} = CC_{lw}= 300(fetch) + max(300, 100)(control, register file read) + 400(ALU) + 200(data memory read) + 100(register file write) = 300 + 300 + 400 + 200 + 100 = 1300 ps

CC_{mod} = max(CC_{lw}, CC_{sw}, CC_{Rtype}, Cc_{beq}, Cc_{jump}) = max(900, 800, 1100, 1000, 300) = 1100 ps

For every load-store with a nonzero offset, an additional instruction must be executed. Thus, if the fraction of instructions which is a `lw` with a nonzero offset is x, the number of modified instructions is IC_{orig} (1 + x). Remembering that CPI_{mod} = CPI_{orig} = 1

$$\frac{P_{\text{mod}}}{P_{orig}} = \frac{ET_{orig}}{ET_{\text{mod}}} = \frac{IC_{orig} * CPI_{orig} * CC_{orig}}{IC_{\text{mod}} * CPI_{\text{mod}} * CC_{\text{mod}}} = \frac{IC_{orig} * 1 * 1300\,ps}{IC_{orig}(1+x) * 1 * 1100\,ps} = 1$$

IC_{orig}*1*1.3 ns = IC_{orig}(1 + x)*1*1.1 ns

1.1818 = 1 + x and x = 0.1818, or 18.2 %

6. (10 points) Consider a pipeline for a register-memory architecture. The architecture has two instruction formats: a register-register format and a register-memory format. There is a single memory addressing mode (offset + base register). There is a set of ALU operations as follows:

ALUop← Rdest,Rsrc1,Rsrc2,offset
Rdest ← Rsrc1 ALUop Rsrc2
or Rdest ← Rsrc1 ALUop MEM[Rsrc2 + offset]
or
Rdest ← MEM[Rsrc2 + offset]
or
MEM[Rsrc2 + offset] ← Rsrc1

where the ALUop is one of the following:

Add, Subtract, And, Or(with or without offset)
Load(Rsrc1 omitted)
Store(Rdest omitted)

Rdest, Rsrc1 and Rsrc2 are registers.

Branches use a full compare of two registers and are PC-relative. Assume that this machine is pipelined so that a new instruction is started every clock cycle. The pipeline structure is

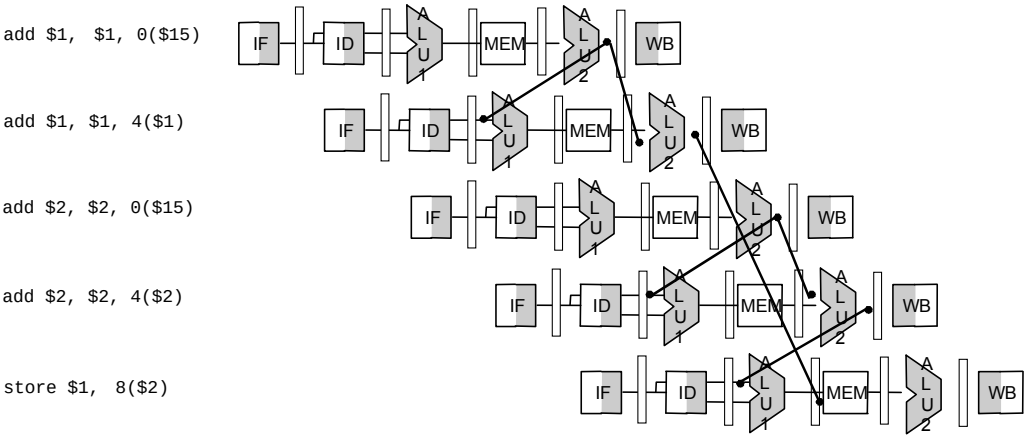
IF	RF	ALU1	MEM	ALU2	WB					
	IF	RF	ALU1	MEM	ALU2	WB				
		IF	RF	ALU1	MEM	ALU2	WB			
			IF	RF	ALU1	MEM	ALU2	WB		
				IF	RF	ALU1	MEM	ALU2	WB	
					IF	RF	ALU1	MEM	ALU2	WB

The first ALU stage is used for effective address calculation for memory references and branches. The second ALU stage is used for operations and branch comparisons. RF is both a decode and register-fetch stage. Assume that when a register read and a register write of the same register occur in the same clock cycle, the write data is forwarded. For the following code fragment:

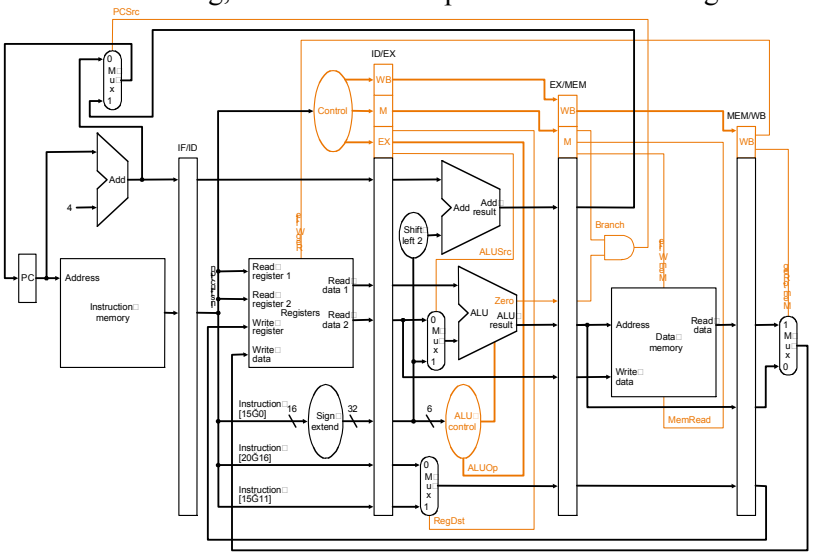
```
add   $1, $1, 0($15)
add   $1, $1, 4($1)
add   $2 ,$2, 0($15)
add   $2, $2, 4($2)
store $1, 8($2)
```

identify the data dependencies and draw a figure (using the multiple clock style) showing them as lines.

```
a  add   $1, $1, 0($15)        Dependencies
b  add   $1, $1, 4($1)         a & b, a & b
c  add   $2 ,$2, 0($15)        b & e
d  add   $2, $2, 4($2)         c & d, c & d
e  store $1, 8($2)             d & e
```



7. (5 points) Consider executing the following code on a pipelined datapath like the one shown, in which there is no forwarding, and a branch completes 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
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 $sp, $sp -20` instruction at the `sort` label begins executing in cycle 1, identify the instructions in each stage of the pipeline in cycle 7

Cycle	IF	ID	EX	MEM	WB
1	`addi $sp`				
2	`sw $ra`	`addi $sp`			
3	`sw $s3`	`sw $ra`	`addi $sp`		
4	`sw $s3`	`sw $ra`	`bubble`	`addi $sp`	
5	`sw $s3`	`sw $ra`	`bubble`	`bubble`	`addi $sp`
6	`sw $s2`	`sw $s3`	`sw $ra`	`bubble`	`bubble`
7	`sw $s1`	`sw $s2`	`sw $s3`	`sw $ra`	`bubble`

8. (5 points) A binary word with even parity and no errors will have an even 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 even parity.

   a. 01100111 P = 1
   b. 01010101 P = 0

9. (1 point) The alternative model to shared memory for communicating uses _message passing__ between processors.

10. (1 point) The coordination needed between processors operating in parallel on shared data is called __synchronization_.

11. (1 point) __Communication_ is the hardest problem.

12. (1 point) A _semiconductor__ is a substance that does not conduct electricity well but is the foundation of integrated circuits.

13. (1 point) A ___transistor__ is an on/off switch controlled by electricity.

14. (10 points) Represent 0.0703125 as a single precision floating point number.

$0.0703125_{10} = (0.0001001)_2 = 1.001 \times 2^{-4}$
exp = -4 + 127 = 123 = 01111011_2
frac = 00100…0
sign = 0

0x3D900000

15. (10 points) Show the single MIPS instruction or minimal sequence of instructions for this C statement:

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

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

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