The University of Alabama in Huntsville
Electrical and Computer Engineering Department
CPE 221 01
Test 2
April 6, 2016

This test is closed book, closed notes. You *may* use a calculator. You should have the reference packet that includes Figure 2.10 and Appendix B. You must show your work to receive full credit.

Name: _____

1. (1 point) **ld** instructions read from memory in stage _____ of the SRC pipeline.
2. (1 point) Branches that do not link complete in stage _____ of the SRC pipeline.
3. (1 point) In certain circumstances, a data dependence can become a data _____.
4. (1 point) In the microcoding described in this class, a PLA uses the _____ of an instruction to determine where the microcode for that instruction is located.
5. (1 point) A _____ capability is one that initializes a processor to a known, defined state.

6. (4 points) r2 contains a value of -720 in decimal. What is the decimal value of r1 after this instruction is executed?

```
shc r1, r2, 10
```

7. (4 points) r2 contains a value of -495 in decimal. What is the decimal value of r1 after this instruction is executed?

```
shra r1, r2, 5
```

8-12. (20 points) What are the values of the following registers and memory addresses when the program executes "brpl r31, r5" for the second time? Answer in decimal.

8. r31: _____ 9. r5: _____

10. r10: _____ 11. r3: _____

12. r4: _____

```
                .org    200
size:           .dc     10
neg_count:      .dw     1
pos_count:      .dw     1
nums:           .dc     100, 3, -1, 2, 4, 37, -100, 13, -5, 0
orig:           .org    1000
                la      r10, nums
                ld      r1, size
                la      r2, 0
                la      r3, 0
                la      r4, 0
                la      r31, done
                la      r30, loop
                la      r29, negative
loop:           sub     r5, r2, r1
                brpl    r31, r5
                shl     r5, r2, 2
                add     r5, r5, r10
                ld      r5, 0(r5)
                addi    r2, r2, 1
                brmi    r29, r5
                addi    r4, r4, 1
                br      r30
negative:       addi    r3, r3, 1
                br      r30
done:           st      r3, neg_count
                st      r4, pos_count
                stop
```

13. (10 points) Identify all of the data dependencies in the following code.

```
add     r3, r5, r4
ld      r4, 28(r1)
add     r5, r4, r3
st      r5, 100(r4)
add     r6, r1, r8
brpl    r29, r9
```

14. (2 points) If the clock frequency is 750 MHz, what is the clock period?

15. (10 points) Design a microcode sequence to implement the 1-bus SRC *lddr* instruction you created on Test 1 using the concrete RTN shown below.

address	100	101	102	700	701	702	703	704	705
Mux control									
PC_{out}									
C_{out}									
MD_{out}									
MD_{in}									
R_{out}									
$c2_{out}$									
BA_{out}									
MA_{in}									
C_{in}									
A_{in}									
R_{in}									
PC_{in}									
IR_{in}									
ADD									
OR									
Wait									
Read									
AND									
INC4									
Gra									
Grb									
Grc									
End									
Address									

Step	**RTN for the lddr Instruction**	**Control Sequence**
T0	MA ← PC: C ← PC + 4;	PC_{out}, MA_{in}, INC4, C_{in}
T1	MD ← M[MA]: PC ← C;	C_{out}, PC_{in}, Read, Wait
T2	IR ← MD;	MD_{out}, IR_{in}
T3	A ← R[rb];	Grb, R_{out}, A_{in}
T4	C ← A + R[rc];	C_{in}. R_{out}, Grc, ADD
T5	MA ← C;	MA_{in}, C_{out}
T6	MD ← M[MA];	Read, Wait
T7	R[ra] ← MD	Gra, R_{in}, End, MD_{out}

Microcode Branching Examples

Address	Mux Ctl	BrUn	BrNotZ	BrZ	BrNotN	BrN	Branch Address	Branching Action
200	00	0	0	0	0	0	ddd	None -201 next
201	01	1	0	0	0	0	ddd	To output of PLA
202	10	0	0	1	0	0	ddd	To external address if Z
203	11	0	0	0	0	1	300	To 300 if N (else 204)

16. (30 points) Complete the SRC assembly language program below so that it implements the following C++ statements.

```
const int size = 10;
int array1[size] = {100, 3, -1, 2, 4, 4, 2, -1, 3, 100};
int array2[size] = {-53, 247, 95, -7, 481, 91, -33, 1500, 29, -83};
int array3[size];
int i;
for (i = 0; i < size; i++)
  array3[i] = array1[i] + array2[i];
```

```
;
;      This program sums two arrays and puts the resulting values
;       in a third array.
;
            .org   200
size:       .dc    10
array1:     .dc    100, 3, -1, 2, 4, 4, 2, -1, 3, 100
array2:     .dc    -53, 247, 95, -7, 481, 91, -33, 1500, 29, -83
array3:     .dw    10
i:          .dc    0
orig:       .org   1000




            la     r10, array1         ; pointer to first element of array1
            la     r11, array2         ; pointer to first element of array2
            la     r12, array3        ; pointer to first element of array3
            ld     r1, size            ; holds size of arrays
            ld     r2, i              ; holds loop counter




           stop
```

17. (15 points) Trace the code fragment given through a 5-stage SRC pipeline that has forwarding from the output of stage 3 to the input of stage 3 and from the output of stage 4 to the input of stage 4. No other forwarding is available. Insert nop bubbles into the pipelines as needed to resolve any dependences. Describe how any forwarding that is happening is being accomplished. Use the table if you find it helpful, it is not required. If you don't use it, clearly show how many cycles it takes to execute this code fragment. You may not use all of the rows. If you need more rows, add them.

```
(1) shr  r3, r3, 2
(2) sub  r2, r3, r1
(3) ld   r4, 0(r3)
(4) add  r6, r4, r1
```

	IF	ID	EX	ME	WB
Cycle					
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					
19					
20					