The University of Alabama in Huntsville
Electrical and Computer Engineering Department
CPE 221 01
Fall 2012
Test 2 Solution

1. (1 point).Pipelining is a technique for improving the _**throughput**_ of instruction execution.
2. (1 point) In _**horizontal**_ microcode, each control signal is represented by a bit in the microinstruction.
3. (1 point) In _**vertical**_ microcode, each distinct microinstruction is encoded as a single signal that is fanned out to the control signals that are to be asserted.
4. (1 point) _**Dynamic**_ RAM requires refreshing.
5. (1 point) The bit line _**precharge**_ operation that causes the difference between access time and cycle time

6. (4 points) r2 contains a value of 235 in decimal. What is the decimal value of r1 after this instruction is executed?

   shc r1, r2, 27

   235 = 128 + 64 + 32 + 8+ 2 +1 = 1110_1011

   r2 = `0000_0000_0000_0000_0000_0000_1110_1011`

   r1 = `0101_1000_0000_0000_0000_0000_0000_0111`
   r1 = $2^{30} + 2^{28} + 2^{27} + 2^2 + 2^1 + 2^0$ = 1476395015

7. (3 points) r2 contains a value of -5179 in decimal. What is the decimal value of r1 after this instruction is executed?

   shra r1, r2, 4

   5179 = 4096 + 1024 + 32 + 16 +8 +2 +1 = `0000_0000_0000_0000_0001_0100_0011_1011`

   r2 = -5179 = `1111_1111_1111_1111_1110_1011_1100_0101`

   r1 = `1111_1111_1111_1111_1111_1110_1011_1100`
   2s complement is `0000_0000_0000_0000_0000_0001_0100_0100` **= 256 + 64 + 4 = 324**

   r1 = -324

8. _**d**_ (2 points) Which memory is the largest? Select one.

   (a) register (b) cache memory (c) main memory (d) disk memory

9-13. (20 points) What are the values of the following registers and memory address when the program executes "`brzr r29, r4`" for the fourth time? Answer in decimal.

9. r30: **_1032_** 10. r1: **_8_** 11. r2: **_9_** 12. r3: **_32_** 13. r4: **_5_**

```
data:       .org  200
num1:       .dc   8
num2:       .dc   9
result:     .dw   1

code:       .org  1000
1000        lar   r30, again
1004        lar   r29, done
1008        ld    r1, num1
1012        ld    r2, num2
1016        sub   r3, r3, r3
1020        brzr  r29, r1
1024        brzr  r29, r2
1028        addi  r4, r2, 0
1032again:  add   r3, r3, r1
            addi  r4, r4, -1
            brzr  r29, r4
            br    r30
done:       st    r3, result
            stop
```

14. (10 points) Identify all of the data dependencies in the following code.

```
a       add   r3, r5, r4
b       ld    r4, 28(r1)
c       add   r5, r4, r2
d       st    r5, 100(r3)
e       add   r6, r5, r8
f       brpl  r29, r5
```

Instruction a produces a value of r3 that is needed by instruction d
Instruction b produces a value of r4 that is needed by instruction c
Instruction c produces a value of r5 that is needed by instructions d, e, and f

15. **_c_** (2 points) Which is **not** a register in SRC?
a. 32 general purpose registers b. a program counter
c. a cause register d. an instruction register

16. **_d_** (2 points) Suppose that R[rc] contains 0x004B 0000. When does a branch not occur?
a. c3<2..0> = 1
b. c3<2..0> = 3
c. c3<2..0> = 4
d. c3<2..0> = 5 **branch on minus, sign bit = 0**

17. **_a or e_** (2 points)This is 1-bus microarchitecture. We add new registers not found in the abstract RTN. Which is incorrect?
a. A and C registers are needed to temporarily store two operands and the result when doing ALU operations. **Correct would be one operand**
b. MA and MD registers are used as interface registers to the memory system.
c. MA contains the address of the memory operand.
d. MD is used as a buffer for outgoing and incoming values.
e. None of the above

18. **_c_** (2 points) Which is incorrect in 1-bus SRC?
a. MA only receives data from the bus and passes it on to the memory subsystem.
b. MD is bidirectional: it sends data to the memory subsystem upon a "write" and receives data from the memory system upon a "read."
c. The 4-bit register, n, stores the shift count, and in this design n is a counter that can be decremented.
d. The Cond logic employs the least significant 3 bits of the c3 field in the IR, which specify the condition to be tested in conditional branch instruction, as well as the bus contents to compute whether a given condition is true.
e. None of the above

19. **_a_** (2 points) If the maximum clock frequency is 625 MHz and we assume no safety margin, what is the minimum clock period?
a. 1.6 ns b. 800 ps c. 1.6 ps d. 800 ns

20. **__c or e__** (2 points)Which is the incorrect assumption in pipeline design?
a. The instruction set is unchanged.
b. Two separate memories are needed, one for program and the other for data.
c. Virtually all pipelined designs require a 3-port register file to allow the reading of one operand and the writing of the other in a single clock cycle. **Correct would be two operands**
d. Pipelined designs replace the buses with direct connections between registers.
e. None of the above.

21. **_b_** (2 points) Which is incorrect explaining the global state of the pipelined SRC?
a. Instruction memory is only accessed in Stage 1.
b. PC is only accessed in Stage 1.
c. General registers are read in Stage 2 and written in Stage 5.
d. Data memory is only accessed in Stage 4.

22. **_a_** (2 points) There are 9 ALU and 3 branch control signals. If we use a 4:16 decoder and a 2:4 decoder in vertical microcode, how many bits are saved compared to horizontal microcode?
a. 6 b. 8 c. 7 d. 14
Savings = (9-4) + (3 – 2) = 5 + 1 = 6

23. (10 points) Design a microcode sequence to implement the 1-bus SRC *XOR* instruction using the concrete RTN shown below.

address	100	101	102	700	701	702	703	704	705	706	707	708	709
Mux control	**00**	**00**	**01**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**11**
PC_{out}	**1**												
C_{out}		**1**			**1**		**1**		**1**		**1**		**1**
MD_{out}			**1**									**1**	
MD_{in}							**1**						
R_{out}				**1**		**1**		**1**		**1**			
$c2_{out}$													
BA_{out}													
MA_{in}	**1**												
C_{in}	**1**					**1**		**1**		**1**		**1**	
A_{in}					**1**				**1**		**1**		
R_{in}													**1**
PC_{in}		**1**											
IR_{in}			**1**										
NOT				**1**				**1**					
OR												**1**	
Wait		**1**											
Read		**1**											
AND						**1**				**1**			
INC4	**1**												
Gra													**1**
Grb				**1**						**1**			
Grc						**1**		**1**					
End													**1**
Address													**100**

Step	**RTN for the XOR 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	C ← ¬R[rb];	Grb, R_{out}, NOT C_{in}
T4	A ← C;	C_{out}, A_{in}
T5	C ← A ∧ R[rc];	Grc, R_{out}, AND, C_{in}
T6	MD ← C;	Cout, MD_{in}
T7	C ← ¬R[rc]	Grc, R_{out}, NOT, C_{in}
T8	A ← C;	C_{out}, A_{in}
T9	C ← A ∧ R[rb];	Grb, R_{out}, AND, C_{in}
T10	A ← C;	C_{out}, A_{in}
T11	C ← A ∨ MD;	MD_{out},OR, Cin
T12	R[ra] ← C;	C_{out}, Gra, R_{in}, End

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)

24. (30 points) Complete the SRC assembly language program below so that it implements the following C++ statements.

```
   int max;
   int size = 10;
   int nums[10] = {5, 3, -1, 2, 4, 37, -100, 13, -5, 0};
   max = nums[0];
   for (i = 1; i < size; i++)
      if (nums[i] > max)
         max = nums[i];
;
;     This program finds the maximum value in an array
;

            .org  200
size:       .equ  10
max:        .dw   1
array:      .dc   5, 3, -1, 2, 4, 37, -100, 13, -5, 0
orig:       .org  1000
            lar   r30, done
            lar   r29, loop
            lar   r28, inc

            lar   r10, array        ; pointer to first element of array
            la    r1, size          ; holds size of array
            ld    r2, 0(r10)        ; max = num[0]
            la    r4, 1             ; r4 = 1, index1 into array

loop:       sub   r5, r4, r1        ; Check to see whether index < size.
            brpl  r30, r5           ; If not, done.
            shl   r6, r4, 2         ; Multiply index by 4 to access entry
                                    ; in array by byte address.
            add   r6, r6, r10       ; Add index to base array pointer.
            ld    r7, 0(r6)         ; Load array[index] into r7.
            sub   r8, r7, r2
            brmi  r28, r8
            addi  r2, r7, 0
inc:        addi  r4, r4, 1
            br    r29
done:       st    r2, max
            stop
```