OUTCOME #4: “An ability to design and implement computer logic circuits.”

Multiple Choice – select the single most appropriate response for each question.

Note that none of the above MAY be a VALID ANSWER.

1. The five-bit sign and magnitude number, \( SM(10101)_2 \), converted to radix notation, is:
   (A) \( R (10101)_2 \)
   (B) \( R (01010)_2 \)
   (C) \( R (11010)_2 \)
   (D) \( R (11011)_2 \)
   (E) none of the above

2. When subtracting the five-bit signed numbers \( (10111)_2 \) \(-\) \( (11001)_2 \) using radix arithmetic, the result obtained is:
   (A) \( (01110)_2 \)
   (B) \( (11111)_2 \)
   (C) \( (11110)_2 \)
   (D) overflow (invalid result)
   (E) none of the above

3. The number of diagonals of full adder cells required to implement a 5x3 unsigned binary multiplier is:
   (A) 3  (B) 4  (C) 5  (D) 6  (E) none of these

4. The total number of full adder cells required to implement a 5x3 unsigned binary multiplier is:
   (A) 8
   (B) 10
   (C) 12
   (D) 24
   (E) none of the above

5. If an N-nanosecond PLD were used to generate each product component bit (using a separate macrocell for each product component bit), and implement each full adder cell (using a pair of macrocells to implement each full adder cell), the worst case propagation delay of a 5x3 unsigned binary multiplier array would be:
   (A) 7N
   (B) 8N
   (C) 9N
   (D) 10N
   (E) none of the above
The following ABEL file applies to questions 6-8:

```ABEL
MODULE cla4
TITLE '4-bit Carry Look-Ahead Adder Block'

DECLARATIONS
X0..X3, Y0..Y3 pin;  " operands
CIN pin;  " carry in (Cin)
C0..C3 pin istype 'com';  " carry equations (C3 is Cout)
S0..S3 pin istype 'com';  " sum outputs

G0 = X0&Y0;  " generate function declarations
G1 = X1&Y1;
G2 = X2&Y2;
G3 = X3&Y3;

P0 = X0$Y0;  " propagate function declarations
P1 = X1$Y1;
P2 = X2$Y2;
P3 = X3$Y3;

EQUATIONS

" carry equations
C0 = G0 # CIN&P0;
C1 = G1 # G0&P1 # CIN&P0&P1;
C2 = G2 # G1&P2 # G0&P1&P2 # CIN&P0&P1&P2;
C3 = G3 # G2&P3 # G1&P2&P3 # G0&P1&P2&P3 # CIN&P0&P1&P2&P3;

" sum equations
S0 = CIN$P0;
S1 = C0$P1;
S2 = C1$P2;
S3 = C2$P3;

END
```

6. If realized as coded above, the **number of product terms** required to implement the equation for **C1** would be:
   (A) 3  (B) 4  (C) 7  (D) 10  (E) none of the above

7. If realized as coded above, the **number of product terms** required to implement the equation for **S1** would be:
   (A) 3  (B) 4  (C) 7  (D) 10  (E) none of the above

8. If this design, as coded above, were realized in an **N-nanosecond** PLD, the worst case propagation delay (in nanoseconds) would be:
   (A) N  (B) 2N  (C) 3N  (D) 4N  (E) none of the above
The following chart applies to questions 9-11:

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_0$</th>
<th>(A)</th>
<th>$B_1$</th>
<th>$B_0$</th>
<th>(B)</th>
<th>?</th>
<th>C</th>
<th>Z</th>
<th>N</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(A) = (B)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>(A) &lt; (B)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>(A) &gt; (B)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>(A) &gt; (B)</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(A) &gt; (B)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>(A) = (B)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>(A) &gt; (B)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>(A) &gt; (B)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(A) &lt; (B)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>(A) &lt; (B)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>(A) = (B)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>(A) &lt; (B)</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>(A) &lt; (B)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>1</td>
<td>+1</td>
<td>(A) &lt; (B)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>-2</td>
<td>(A) &gt; (B)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>(A) = (B)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

9. The function for “A equals B” ($F_{A=B}$) can be expressed as:
   (A) $F_{A=B} = C$
   (B) $F_{A=B} = Z$
   (C) $F_{A=B} = N$
   (D) $F_{A=B} = V$
   (E) none of the above

10. The function for “A less than B” ($F_{A<B}$) can be expressed as:
    (A) $F_{A<B} = Z + N\cdot V + N\cdot V'$
    (B) $F_{A<B} = N\cdot V + N\cdot V'$
    (C) $F_{A<B} = N\cdot V' + N\cdot V$
    (D) $F_{A<B} = Z' + N\cdot V' + N\cdot V$
    (E) none of the above

11. The function for “A greater than or equal to B” ($F_{A\geq B}$) can be expressed as:
    (A) $F_{A\geq B} = Z + N\cdot V + N\cdot V'$
    (B) $F_{A\geq B} = N\cdot V + N\cdot V'$
    (C) $F_{A\geq B} = N\cdot V' + N\cdot V$
    (D) $F_{A\geq B} = Z' + N\cdot V' + N\cdot V$
    (E) none of the above
12. The purpose of the “correction circuit” is to:
   (A) always add (0110)_2 to the direct sum (Z3 Z2 Z1 Z0)
   (B) add (0110)_2 to the direct sum (Z3 Z2 Z1 Z0) only if Z4=1
   (C) always subtract (0110)_2 from the direct sum (Z3 Z2 Z1 Z0)
   (D) add (0110)_2 to the direct sum (Z3 Z2 Z1 Z0) only if Z4+Z3·Z2+Z3·Z1=1
   (E) none of the above

13. If X3..X0 = 0111, Y3..Y0 = 0011, and Cin = 1, the value input to the correction circuit (Z4 Z3 Z2 Z1 Z0) will be:
   (A) 0 1 0 1 0
   (B) 0 1 0 1 1
   (C) 1 0 0 0 0
   (D) 1 0 0 0 1
   (E) none of the above

14. If X3..X0 = 0111, Y3..Y0 = 0011, and Cin = 1, the value output by the correction circuit (Cout S3 S2 S1 S0) will be:
   (A) 0 1 0 1 0
   (B) 0 1 0 1 1
   (C) 1 0 0 0 0
   (D) 1 0 0 0 1
   (E) none of the above
15. As observed in Lab 13, incrementing the program counter (PC) on the same clock edge that loads the instruction register (IR) does not cause a problem because:
   (A) the memory will ignore the new address the PC places on the address bus
   (B) the output buffers in the PC will not allow the new PC value to affect the address bus until the next fetch cycle
   (C) the IR will be loaded with the value on the data bus prior to the clock edge while the contents of the PC will increment after the clock edge
   (D) the value in the PC will change in time for the correct value to be output on the address bus (and fetch the correct instruction), before the IR load occurs
   (E) none of the above

16. If a program contains more PSH instructions than POP instructions, the following is likely to occur:
   (A) stack underflow (stack collides with beginning of program space)
   (B) stack overflow (stack collides with end of program space)
   (C) program counter overflow (program counter wraps to beginning of program space)
   (D) program counter underflow (program counter wraps to end of program space)
   (E) none of the above

17. Whether or not a conditional branch is taken or not taken depends on:
   (A) the value of the program counter
   (B) the state of the condition code bits
   (C) the cycle of the state counter
   (D) the value in the accumulator
   (E) none of the above

18. The state counter in the “extended” machine’s instruction decoder and micro-sequencer needs both a synchronous reset (RST) and an asynchronous reset (ARS) because:
   (A) we want to make sure the state counter gets reset
   (B) the ARS signal allows the state counter to be reset to the “fetch” state when START is pressed, while the RST allows the state counter to be reset when the last execute cycle of an instruction is reached
   (C) the RST signal allows the state counter to be reset to the “fetch” state when START is pressed, while ARS allows the state counter to be reset when the last execute cycle of an instruction is reached
   (D) the state counter is not always clocked
   (E) none of the above

19. When a set of control signals are said to be mutually exclusive, it means that:
   (A) all the control signals may be asserted simultaneously
   (B) only one control signal may be asserted at a given instant
   (C) each control signal is dependent on the others
   (D) any combination of control signals may be asserted at a given instant
   (E) none of the above

© 2012 by D. G. Meyer / Purdue University – may not be copied or reproduced, in any form or by any means.
20. If the input control combination $AOE=0$, $ALE=1$, $ALX=0$, $ALY=0$ is applied to this circuit, the function performed will be:
   (A) ADD
   (B) SUBTRACT
   (C) LOAD
   (D) NEGATE
   (E) none of the above

21. If the input control combination $AOE=0$, $ALE=1$, $ALX=1$, $ALY=0$ is applied to this circuit, the function performed will be:
   (A) ADD
   (B) SUBTRACT
   (C) LOAD
   (D) NEGATE
   (E) none of the above

22. If the input control combination $AOE=1$, $ALE=1$, $ALX=1$, $ALY=1$ is applied to this circuit, the function (inadvertently) performed on (A) will be equivalent to a:
   (A) logical left shift
   (B) logical right shift
   (C) rotate left
   (D) rotate right
   (E) none of the above

23. When the program stops (“halts”), the value in the A register will be:
   (A) 0100 1100
   (B) 1000 0000
   (C) 0000 0000
   (D) 0100 0011
   (E) none of the above

24. When the program stops (“halts”), the condition code bits will be:
   (A) $CF = 0$, $NF = 0$, $VF = 0$, $ZF = 0$
   (B) $CF = 0$, $NF = 1$, $VF = 0$, $ZF = 0$
   (C) $CF = 1$, $NF = 1$, $VF = 1$, $ZF = 0$
   (D) $CF = 1$, $NF = 0$, $VF = 0$, $ZF = 1$
   (E) none of the above

25. When the program stops (“halts”), the value in the program counter will be:
   (A) 00000
   (B) 00110
   (C) 00111
   (D) 01111
   (E) none of the above
Figure 3 on the attached Reference Sheets applies to questions 26 through 30.

**NOTE:** If there is only one execute state, then “first execute state” = “last execute state”.

26. The following control signal is **not** asserted on a fetch cycle:
   (A) IRA  
   (B) IRL  
   (C) PCC  
   (D) POA  
   (E) none of the above

27. Assuming a **standard stack convention** (in which the SP register points to the **top stack item**), the following **SP control signal(s)** will be asserted on the **first execute cycle** of a JSR instruction:
   (A) SPI  
   (B) SPD  
   (C) SPI and SPA  
   (D) SPD and SPA  
   (E) none of the above

28. Assuming a **standard stack convention** (in which the SP register points to the **top stack item**), the following **PC control signal** will be asserted on the **last execute cycle** of a JSR instruction:
   (A) POA  
   (B) POD  
   (C) PLA  
   (D) PLD  
   (E) none of the above

29. Assuming a **standard stack convention** (in which the SP register points to the **top stack item**), the following **SP control signal(s)** will be asserted on the **first execute cycle** of an RTS instruction:
   (A) SPI  
   (B) SPD  
   (C) SPI and SPA  
   (D) SPD and SPA  
   (E) none of the above

30. Assuming a **standard stack convention** (in which the SP register points to the **top stack item**), the following **PC control signal** will be asserted on the **last execute cycle** of an RTS instruction:
   (A) POA  
   (B) POD  
   (C) PLA  
   (D) PLD  
   (E) none of the above
Figure 1. Block Diagram for Bit 3 of a Simple Computer ALU (applies to questions 20 through 22).
<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mnemonic</th>
<th>Function Performed</th>
<th>CC Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>ADD addr</td>
<td>Add contents of addr to contents of A</td>
<td>CF, NF, VF, ZF</td>
</tr>
<tr>
<td>0 0 1</td>
<td>SUB addr</td>
<td>Subtract contents of addr from contents of A</td>
<td>CF, NF, VF, ZF</td>
</tr>
<tr>
<td>0 1 0</td>
<td>LDA addr</td>
<td>Load A with contents of location addr</td>
<td>NF, ZF</td>
</tr>
<tr>
<td>0 1 1</td>
<td>XOR addr</td>
<td>XOR contents of addr with contents of A</td>
<td>NF, ZF</td>
</tr>
<tr>
<td>1 0 0</td>
<td>STA addr</td>
<td>Store contents of A at location addr</td>
<td>none</td>
</tr>
<tr>
<td>1 0 1</td>
<td>HLT</td>
<td>Halt – Stop, discontinue execution</td>
<td>none</td>
</tr>
</tbody>
</table>

Figure 2. Simple Computer Instruction Set, Block Diagram, and Memory Contents (applies to questions 23 through 25).
Instruction Set:

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Mnemonic</th>
<th>Description</th>
<th>Opcode</th>
<th>Mnemonic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>HLT</td>
<td>Stop execution</td>
<td>1 0 0</td>
<td>ADD addr</td>
<td>(A) ← (A)+(addr)</td>
</tr>
<tr>
<td>0 0 1</td>
<td>LDA addr</td>
<td>(A) ← (addr)</td>
<td>1 0 1</td>
<td>SUB addr</td>
<td>(A) ← (A)-(addr)</td>
</tr>
<tr>
<td>0 1 0</td>
<td>STA addr</td>
<td>(addr) ← (A)</td>
<td>1 1 0</td>
<td>JSR addr</td>
<td>Call subroutine at location addr</td>
</tr>
<tr>
<td>0 1 1</td>
<td>ASR</td>
<td>Arithmetic Shift Right (A)</td>
<td>1 1 1</td>
<td>RTS</td>
<td>Return from subroutine</td>
</tr>
</tbody>
</table>

ALU Function Table:

<table>
<thead>
<tr>
<th>AOE</th>
<th>ALE</th>
<th>ALX</th>
<th>ALY</th>
<th>Function</th>
<th>CF</th>
<th>ZF</th>
<th>NF</th>
<th>VF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Add</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Subtract</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Load</td>
<td>●</td>
<td>X</td>
<td>X</td>
<td>●</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Arithmetic Shift Right*</td>
<td>X</td>
<td>X</td>
<td>●</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>d</td>
<td>d</td>
<td>Output</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>d</td>
<td>d</td>
<td>&lt;none&gt;</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

X → flag affected,  ● → flag not affected
*For Arithmetic Shift Right, CF = bit shifted out of accumulator

Signal Names:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>Asynchronous Machine Reset</td>
</tr>
<tr>
<td>MSL</td>
<td>Memory Select</td>
</tr>
<tr>
<td>MOE</td>
<td>Memory Output Tri-State Enable</td>
</tr>
<tr>
<td>MWE</td>
<td>Memory Write Enable</td>
</tr>
<tr>
<td>PCC</td>
<td>Program Counter Count Enable</td>
</tr>
<tr>
<td>POA</td>
<td>Program Counter Output on Address Bus Tri-State Enable</td>
</tr>
<tr>
<td>PLA</td>
<td>Program Counter Load from Address Bus Enable</td>
</tr>
<tr>
<td>POD</td>
<td>Program Counter Output on Data Bus Tri-State Enable</td>
</tr>
<tr>
<td>PLD</td>
<td>Program Counter Load from Data Bus Enable</td>
</tr>
<tr>
<td>IRL</td>
<td>Instruction Register Load Enable</td>
</tr>
<tr>
<td>IRA</td>
<td>Instruction Register Output on Address Bus Tri-State Enable</td>
</tr>
<tr>
<td>AOE</td>
<td>A-register Output on Data Bus Tri-State Enable</td>
</tr>
<tr>
<td>ALE</td>
<td>ALU Function Enable</td>
</tr>
<tr>
<td>ALX</td>
<td>ALU Function Select Line “X”</td>
</tr>
<tr>
<td>ALY</td>
<td>ALU Function Select Line “Y”</td>
</tr>
<tr>
<td>SPI</td>
<td>Stack Pointer Increment</td>
</tr>
<tr>
<td>SPD</td>
<td>Stack Pointer Decrement</td>
</tr>
<tr>
<td>SPA</td>
<td>Stack Pointer Output on Address Bus Tri-State Enable</td>
</tr>
<tr>
<td>RST</td>
<td>Synchronous State Counter Reset</td>
</tr>
<tr>
<td>RUN</td>
<td>Machine Run Enable</td>
</tr>
</tbody>
</table>

Block Diagram:

Figure 3. Simple 8-bit Computer Instruction Set, Signal Names, and Block Diagram (applies to questions 26 through 30).