1) 4 points (2 points each). Convert 8-bit (2-digit) 2’s complement hex values:

\[ \text{41} \] -29

\[ \text{0x29} \] \[ \text{0xE3} \]

2) 4 points. Convert 8-bit 2’s complement hex to decimal:

\[ \text{0x3A} \] \[ 58 \]

\[ \text{0xE5} \] \[ -27 \]

3) 4 points. Give the truth table for this transistor-level diagram

\[ \begin{array}{ccc|c}
A & B & \text{Out} \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array} \]

(This is an AND gate)

4) 4 points. Create a truth table for the following circuit.

5) 4 points. Prof. Madden likes Sesame bagels, but does not like Raisin bagels. He will be Happy if the coffee shop has Sesame bagels, and does not have Raisin bagels. Or, if it’s the Weekend, he’s always happy no matter what. Given inputs of S, R, W, sketch a small circuit using Boolean logic gates to determine if he is H.

Truth table is OK here too
6) 4 points. Simplify the Boolean equation \( Y = (ABC + A\bar{B}C)(B+C)(A) \)

\[
(B+\bar{B})(AC)(B+C)A \rightarrow (AAC)(B+C) \rightarrow ABC + AC \rightarrow AC
\]

7) 4 points. You need a 2-input OR gate, but you only have an AND gate and some inverters. Use your AND gate and inverters to make an OR gate.

DeMorgan’s Law!

8) 4 points. Show the truth table for the following finite state machine. S1 S0 is the current state, NS1 NS0 is the next state. The input is T

<table>
<thead>
<tr>
<th>T</th>
<th>S1</th>
<th>S0</th>
<th>NS1</th>
<th>NS0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>0</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>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

9) 6 points. Construct a Karnaugh map, and show the simplified circuit for the following truth table.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<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>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>x</td>
</tr>
</tbody>
</table>

Top wraps around. Gates are \( !B \) \( !C \) and \( BC \)

4 points Karnaugh, 2 points equation or circuit
For the following questions, assume that the **data segment starts at 0x10010000**, and any **code starts at 0x40000000**. The MIPS assembly language sometimes converts a single machine instruction into two simpler instructions; go ahead and assume that this does not happen, and each instruction is only 32 bits.

```assembly
.data
a:    .word 29
b:    .word 11
c:    .word 6
d:    .word 48
.word 0
```

10) 4 points. What is in $ra, when you get to the beq?

```assembly
addi $a0, $0, 55
addi $a1, $a0, 23
jal cosine
beq $a3, $a4, skip
```

11) 2 points. What is in a1, after this instruction?

```assembly
lw $a1, b
```

12) 2 points. What is in a1, after this instruction?

```assembly
la $a1, b
```

13) 10 points. Write a MIPS assembly subroutine “maxArray”. You are passed a pointer to an integer array (that has all positive numbers, with zero used to terminate the array). Your subroutine should find the maximum (largest positive) number in the array, and return that.

```assembly
maxArray:
    li $v0, 0
loop: lw $t0, 0($a0)
    beq $t0, $0, done
    blt $t0, $v0, skip
    move $v0, $t0
skip: addi $a0, $a0, 4
    j loop
done:
    jr $ra
```

Could switch to bgt, compare other way. 5 points for the loop working correctly, 5 points for compare correct, 2 points min if something is at least partially right.
14) 4 points. Write MIPS assembly to add 8 to a value (passed as a pointer). The C code looks like this. This is a subroutine.

```c
add_eight(int *x)
{
    *x = *x + 8;
}
```

```assembly
add_eight:
    lw $t0, 0($a0)
    addi $t0, $t0, 8
    sw $t0, 0($a0)
    jr $ra
```

15) 8 points. Write MIPS assembly for sub1, a subroutine that calls two other subroutines. The C code looks like this. This is also a subroutine. sub2 and sub3 are in a library.

```c
int sub1()
{
    sub2(10);
    sub3(30);
    return 6;
}
```

```assembly
sub1: addi $sp, $sp, -4
    sw $ra, 0($sp)
    li $a0, 10
    jal sub2
    li $a0, 30
    jal sub3
    li $v0, 5
    lw $ra, 0($sp)
    addi $sp $sp, 4
    jr $ra
```

Q15: 6 points for stack pointer work, 2 points for other stuff

16) 4 points. You have a pointer in register a0 to the C structure below. Write MIPS code for the line of C.

```c
struct
{
    int x;
    int y;
    int z;
} examObj;
```

```assembly
lw $t0, 0($a0)
addi $t0, $t0, 41
sw $t0, 4($a0)
```
17) 2 points. Name two types of hazards you might see on a pipelined microprocessor.

control and data

18) 4 points. Modify the code below so that it could run faster on a pipelined microprocessor.

```
add $a0, $a3, $a4
add $a0, $a1, $a0
lui $a3, 58
```

Hazard is on first two lines, with $a0.
Moving lui up one step would fix things

19) 2 points. Suppose you have a pipelined machine. Each cycle is 300ps, and there are five stages in the pipeline. How long does it take to complete one instruction?

\[ 5 \times 300 = 1500 \text{ps} \]

20) 2 points. Other than rearranging lines of code, what's a way that you can fix a hazard?

Insert a no-op would be the most likely answer.
If they've got another good idea, points...

21) 4 points. You have a direct mapped cache with 4k entries. Your program makes 2000 memory access, with 1200 of them being hits. The processor is pipelined, has 6 stages, and each cycle is 50ps. Sesame bagels are delicious. What is the miss rate?

\[ 800 \text{ misses} \rightarrow 800/2000 \text{ or } 4/10, .4, 40\% \]
22) 8 points. You have a simple direct-mapped cache with 8 blocks, each block being one word (just like the example in the slides and textbook). You run the following code; what is the hit rate for the memory accesses?

```assembly
addi $t0, $0, 10
loop: beq $t0, $0, done
lw $t1, 0x4($0)
lw $t3, 0x8($0)
addi $t0, $t0, -1
j loop
done:
```

Loop runs 10 times, first pass is miss, after that, all hits.
18/20, or 90%, .9

23) 4 points. We have this page table. What physical location does virtual memory address 0x5B44 map to?

```
0x1B44
```

24) 2 points. What does the abbreviation TLB stand for?

**Translation look aside buffer**

25) 4 points. Bonus question. You’re writing a travel guide. What is the best thing about Binghamton?

The beach (or palm trees, the sunny sandy beach...). Yes, the is wrong. We don’t have a beach, which is why it’s funny. The exam is out of 104, so these are 4 extra points.