Computer Sciences Department
University of Wisconsin-Madison
CS/ECE 552 – Introduction to Computer Architecture
In-Class Exercise (01/21) SOLUTION

Answers to all questions should be uploaded on Canvas

1. [2 points] Using 8 bits for each number, write the 1's complement and 2's complement binary number of the decimal numbers in the table below:

Solution:

Decimal	1's Complement	2's Complement
19	0001 0011	0001 0011
-10	1111 0101	1111 0110

2. [3 points] Perform binary arithmetic for the following numbers as directed. All the numbers are represented in 2's complement form.

(a) [1 point] Add the following 2's complement numbers. Write the final result as a 6-bit number.

```
  010110
+ 111001
  001111
```

(b) [1 point] Subtract the following 2's complement numbers. Write the final result as a 6-bit number.

```
  010011
- 010010
  000001
```

(c) [1 point] Convert the numbers for (a) and (b) into decimal numbers. Show your work.

Solution:

001111 = 8 + 4 + 2 +1 = 15
000001 = 1 = 1

3. [1 point] Convert the following 3 hexadecimal numbers to unsigned binary. A 0x before a set of symbols (from 0 to F) indicates a hexadecimal number.

Solution:

0x24 = 0010 0100
0xAA = 1010 1010
0xD7 = 1101 0111

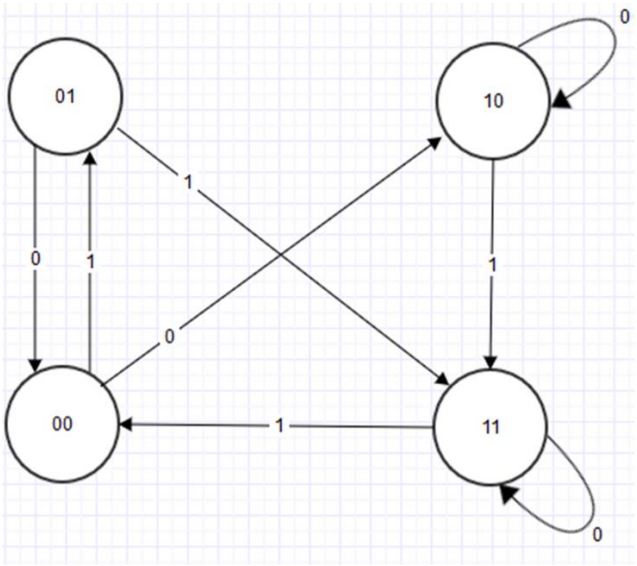
4. [1 point] Use DeMorgan's Law to reduce the number of NOT gates in the expression. Show your work for full credit.

$Z = !((!A\ AND\ !C)\ OR\ !D)$.

Solution:

((A OR C) AND D)

5. [3 points] Consider the state diagram shown in Figure 3. Each state is denoted as S_1S_0, and the values on the arrows represent the input IN, which causes the transition every clock cycle.



Solution:

S_1	S_0	IN	S_1'	S_0'
0	0	0	1	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

6. [2 points] How many cache hits and misses will occur in the following sequence of accesses to the array A, which is an array of 4 byte integers. You can assume that the cache has the following characteristics: 8 byte blocks, direct mapped, and 32 bytes in total size.

Access sequence: A[0], A[8], A[0], A[1], A[2], A[5], A[6], A[10], A[0]

Solution:

Miss, Miss, Miss, Hit, Miss, Miss, Miss, Miss, Hit

7 Misses

2 Hits

NOTE: Several people asked about the replacement policy. For a direct mapped cache, the replacement policy is somewhat irrelevant because you'll be replacing the element in your index every time. However, you could assume LRU (per index) if that helps.

7. [2 points] Fill in the following C function definition from the provided IA32 assembly code (assume that this is a Linux system):

```
int func(int a, int b) {
     A // fill in what A should be
}

func:
  push %ebp
  mov %esp, %ebp
  mov 8(%ebp), %eax
  mov 12(%ebp), %edx
  add %edx, %eax
  leave
  ret
```

Solution:

A = return a + b;