Mirror hosted at: https://github.com/h-dew/cosc211-lab3/

3. i)

0xabcdef12

hex	position	translation	decimal	total
2	0	$0\text{x}2 * 16^0$	$2 * 1 = 2$	2882400018
1	1	$0\text{x}1 * 16^1$	$1 * 16 = 16$	
f	2	$0\text{xf} * 16^2$	$15 * 16^2 = 3840$	
e	3	$0\text{xe} * 16^3$	$14 * 16^3 = 57344$	
d	4	$0\text{xd} * 16^4$	$13 * 16^4 = 851968$	
c	5	$0\text{xc} * 16^5$	$12 * 16^5 = 12582912$	
b	6	$0\text{xb} * 16^6$	$11 * 16^6 = 184549376$	
a	7	$0\text{xa} * 16^7$	$10 * 16^7 = 2684354560$	

3. ii)

When adding these two values, it exceeds the highest possible value contained in a 32-bit register and because of this, an arithmetic overflow exception is thrown.

3. iii)

As mentioned in the previous answer, an overflow occurs.

3. iv)

The value is $0\text{xb}0000000$

3. v)

Yes, this is the desired result

3. vi)

When adding the two values in the first instruction, it exceeds the highest possible value contained in a 32-bit register and because of this, an arithmetic overflow exception is thrown.

3. vii)

As mentioned in the previous answer, an overflow occurs.

3. viii)

{0000 00} {10 000} {1 0000} {1000 0} {000 00} {10 0000}
opcode rs rt rd shamt funct

instruction format: R
opcode: 0, funct: $0x20$ (32) therefore the instruction is add.

rs: 1 0000 → 16 → $s0
rt: 1 0000 → 16 → $s0
rd: 1 0000 → 16 → $s0
shamt: 0

This operation adds the value of a register to itself and stores the result in that same register

3. ix)

```
sw $t1, 32($t2)
```

instruction: store word
instruction format: I
opcode: $0x2b$ → 10 1011

rs: $t2 → 10 → 0 1010
rt: $t1 → 9 → 0 1001
immediate: 32 → 0000 0000 0010 0000

full instruction:
{10 1011} {0 1010} {0 1001} {1000 0000 0010 0000}
opcode rs rt immediate

3. x)

`$t0 = 0xAAAAAAAA, $t1 = 0x12345678`

```
sll $t2, $t0, 44
```

This instruction shifts the bits of $t0 44 positions to the left, filling the newly created empty space with zeroes. Since the bits are shifted 44 times, more bits than the total number of bits in a register, the value of the register becomes 0.

```
addi $t2, $t2, -1
```

This instruction adds the value of -1 to $t2, effectively resulting in a subtraction due to the use of signed numbers. Because the value in \$t2 is 0, a over flow o$

3. xi)

`$t0 = 0xAAAAAAAA, $t1 = 0x12345678`

```
sll $t2, $t0, 4
```

This instruction shifts the bits of $t0 4 positions to the left, filling the newly created empty space with zeroes. The bits get shifted 4 positions to the right,$

```
addi $t2, $t2, -1
```

This instruction adds the value of -1 to $t2, effectively resulting in a subtraction due to the use of signed numbers. The value in \$t2 is currently \$0xaaaa0000$, resulting in a value of $0xaaa9ffff$ after the instruction is executed

3. xii)

`$t0 = 0xAAAAAAAA, $t1 = 0x12345678`

```
srl $t2, $t0, 3
```

This instruction shifts the bits of $t0 3 positions to the right, filling the newly created empty space with zeroes. The bits get shifted 4 positions to the righ$

```
addi $t2, $t2, 0xFFEF
```

This instruction adds the value of $0xffef$ to $t2. The value in \$t2 is currently \$0x000aaaaa$, resulting in a value of $0x000baa99$ after the instruction is executed