Assignment 2-7 Elaborate the actual machine code for the assembly statement 2

1. **Show the machine instruction for sll and srl**

```
[00400024] 34080008 ori $8, $0, 8                ; 8: ori     $8, $0, 0x0008
[00400028] 00084880 sll $9, $8, 2                ; 9: sll     $9, $8, 2
[0040002c] 00095082 srl $10, $9, 2               ; 10: srl     $10, $9, 2
```

Elaborate on OP Code / Registers / Function Code with the machine code

opcode	rs	rt	immediate
34	08	00	08

34080008 ori $8, $0, 8
Goal of the Instruction: Use **OR Immediate** bitwise comparison between $0 (zero) and (constant/immediate) 8. Save result into register $t0 ($8). Since we're comparing it with zero, nothing is changed in the constant.
OP code: "**34**" ORI operand (MIPS I-Type designated Op code representation)
Register source: "**08**" ($8=$t0) because that is where $t0 corresponds to.
Register target: "**00**" Register Source ($0 = zero)
Immediate: "**08**" Immediate (8) = 1000 (Stored as Binary for 8) = 0008 (Machine code uses Hexadecimal for 8)

R-Type (Register)

opcode	rs	rt ($8=$t0)	rd ($9=$t1)	shift	funct
00	0	8	4	8	0 (80?)

00084880 sll $9, $8, 2
Goal of the Instruction: Shift Left Logical. Use the value in register $t0 ($8) and shift it 2 bits places to the left, effectively multiplying the value 2^2(4). Store the result into $t1 ($9).
0000 0000 0000 1000 (8) → 0000 0000 0010 0000 (32)
OP code: "**00**" operand for sll (MIPS R-Type designated Op code representation)
Register target: "**8**" ($8=$t0) = 1000 (Binary for 8) = 0x08 (Machine code uses Hexadecimal for 8)
Register destination: "**4**" ($9=$t1) 4 corresponds to where $t1 is.
Shamt(Shift Amount): "**8**" A shift amount of 2 because 2^2 = 4, and 4x2 places = 8.
Funct: "0" MIPS R-Type function code 000 000. 0 in binary.

R-Type (Register)

opcode	rs	rt	rd	shift	funct

00	0	9	5	8	2

00095082 srl $10, $9, 2
Goal of the Instruction: Shift Right Logical. Use the value in register $t1 ($9) and shift it 2 bits places to the right, effectively dividing the value 2^2(4). Store the result into $t2 ($10).
0000 0000 0010 0000 (32) → 0000 0000 0000 1000 (8)
OP code: "**00**" operand for srl (MIPS R-Type designated Op code representation)
Register target: "**09**" ($9=$t1) = 1001 (Binary for 9) = 0x09 (Machine code uses Hexadecimal for 9)
Register destination: "**5**" ($10=$t2) 5 corresponds to where $t2 is.
Shamt(Shift Amount): "**8**" A shift amount of 2 because 2^2 = 4, and 4x2 places = 8.
Funct: "2" MIPS R-Type function code 000 010, which is 2 in binary.

2. **Show the binary values of Register $t1 after the statement sll**
   a. **Capture the screen value and show it as a decimal value**

      ```
      R9 (t1)  = 00000020
      ```

      0000 0020 = **32**
   b. **Prove this value is 2^2 of $t0**
      8 is stored into t0 as 0000 0008
      We're shifting 8 two bits to the left so effectively multiplying it by 2 (binary)^2 (bit places)
      2^2 = **4**
      8 x **4** = **32** which is 0000 0020
      So yes the value of $t1 is 2^2 of $t0

3. **Show the binary values of Register $t2 after the statement srl**
   a. **Capture the screen value and show it as a decimal value**

      ```
      R10 (t2) = 00000008
      ```

      0000 0008 = **8**
   b. **Is this value $t1 / 2^2**
      Yes
   c. **Why?**
      32 is stored into t1 as 0000 0020
      We're shifting 32 two bits to the right so effectively dividing it by 2 (binary)^2 (bit places)
      2^2 = **4**
      32 / **4** = **8** which is 0000 0008
      So yes the value of $t2 is / 2^2 of $t1

4. **Show the binary values of Register $t1 after the statement sll**
   a. **Capture the screen value and show it as decimal value**

```
R9 (t1)  = fffffffc
```

ffff fffc = **-4**

b. **Is $t1 value $t0 times by 2^2?**
   -2 is stored into t0 as ffff fffe
   We're shifting -2 two bits to the left so effectively multiplying it by 2 (binary)^2 (bit places)
   2^2 = **4 bits**
   -2 x **4** = **-8**
   So no, the value of $t1 is not 2^2 of $t0 because the t1 contains -4

   Unless, from what I've observed in the binary, it's only shifting by **1 bit place?**
   So maybe, it's actually
   2^1 not 2^2?
   So it's -2 x 2^1 = **-4**

   1111 1111 1111 1111 1111 1111 1111 1110 = fffffffe = **-2**
   After sll
   1111 1111 1111 1111 1111 1111 1111 1100 = fffffffc = **-4**

   Does it act differently because it's 2's complement and we have to **add 1**?

5. **Show the binary values of Register $t2 after the statement srl.**
   a. **Capture the screen value and show it as decimal value**

```
R10 (t2) = 7ffffffe
```

   7ffffffe = 2147483646

   b. **Isn't this value $t1 / 2^2?**
      -4 is stored into t1 as ffff fffc
      We're shifting -4 two bits to the right so effectively dividing it by 2 (binary)^2 (bit places)
      2^2 = **4 bits**
      -4 / **4** = **-1**
      So no, the value of $t2 is not / 2^2 of $t1, because $t2 contains 2147483646**.**
   c. **Why?**
      Because when srl is shifting it to the right, it fills the gap made from the shift with zeros, therefore changing the entire value, which now shows a huge positive value.
      1111 1111 1111 1111 1111 1111 1111 1100 = fffffffc = -4
      0111 1111 1111 1111 1111 1111 1111 1110 = 7ffffffe = 2147483646

6. **Replace "srl" with "sra"**
   a. **Capture the screen value and show it as decimal value**

```
R8 (t0)  = fffffffe
R9 (t1)  = fffffffc
R10 (t2) = fffffffe
```

Fffffffe = -2
Fffffffc = -4
Fffffffe = -2

b. **Isn't this value $t1 / 2^2?**
-4 is stored into t1 as ffff fffc
We're shifting -4 two bits to the right so effectively dividing it by 2 (binary)^2 (bit places)
2^2 = **4 bits**
-4 / **4** = **-1**
So no, the value of $t2 is not / 2^2 of $t1, because $t2 contains -2.

But Yes if it's by 2^1 like I mentioned earlier cause then we get -2 which is in $t1.

c. **Why?**

SRA is similar to srl but this one preserves the sign of the original number by filling the vacated bits with the original sign bits. Unlike shown in 5c.

1111 1111 1111 1111 1111 1111 1111 11**00** = fffffffc = -4
1111 1111 1111 1111 1111 1111 1111 111**0** = 7ffffffe = -2