Homework 2

Problem 1: Array Copying

PseudoCode

```
for (i = 0; i < 100; i +=1){
B[i]=A[i] + 4;
}
```

a)

We need to add the +4 portion of the pseudocode to the revised lecture coded. This will not add an extra instruction inside of the loop, as lw already took an offset paremeter, it was previously 0.

```
li s4, 100

loop: #critical section here
    slli t0, s1, 2 # t0 = i * 4
    add t2, t0, s2 # compute addr of A[i]

    lw t1, 4(t2)   # This line changed, offset now 4

    add t3, t0, s3 # compute addr of B[i]
    sw t1, 0(t3)         # Saves word into B[i]
    addi s1, s1, 1 # i++
test: bne s1, s4, loop # Does the loop if i < 100

# 7 lines total in loop
```

b)

We can roll 4 opperations into the loop, saving the slli instruction (t0 = i * 4), addi (s1, s1, 1), and the test being run for 3 opperations. By rolling 4 itterations of the loop inside the loop, we decrease the number of instructions used for the last 3 opperations, meaning we save 3 opperations, 3 times, or save using 9 instructions over 4 loops. An importaint thing to note is that since these instructions are be executed in groups of 4, this moves sets of 4 exclusively now, but for the full 100 lines, and uses less instructions than its previous counterpart.

```
li s4, 100

loop:
```

```
    slli t0, s1, 2 # t0 = i * 4

    add t2, t0, s2 # compute addr of A[i]
    lw t1, 4(t2)   # This line changed, offset now 4
    add t3, t0, s3 # compute addr of B[i]
    sw t1, 0(t3)         # Saves word into B[i]

    addi t2, 1     # compute addr A[i + 1] by adding 1 to A[i]
    lw t1, 4(t2)   # This line changed, offset now 4
    addi t3, 1     # compute addr B[i + 1] by adding 1 to A[i]
    sw t1, 0(t3)         # Saves word into B[i]

    addi t2, 1     # compute addr A[i + 2] by adding 1 to A[i]
    lw t1, 4(t2)   # This line changed, offset now 4
    addi t3, 1     # compute addr B[i + 2]by adding 1 to B[i]
    sw t1, 0(t3)         # Saves word into B[i + 2]

    addi t2, 1     # compute addr A[i + 3] by adding 1 to A[i]
    lw t1, 4(t2)   # This line changed, offset now 4
    addi t3, 1     # compute addr B[i + 3] by adding 1 to B[i]
    sw t1, 0(t3)         # Saves word into B[i + 3]

    addi s1, s1, 4 # i + 4
test: bne s1, s4, loop # Does the loop if i < 100

# 19 lines total in loop
```

Problem 2: A 2D Array

```
li s1, 0    # i = 0
li s2, 0    # j = 0
li s3, 16   # istop = 16
li s4, 8   # jstop = 8
li s5, 256 #var for 256

Fori: #loop run when i < 16

Forj:
 # T[i][j] = 256 * i + j;
add t0, s1, s2 # i + j = t0
slli t0, t0, 8 # (i+j) * 256
sw t0, s9, t0 # store the word with offset
```

```
addi s2, 1              # j++
blt s2, s4,   Forj      # If j < 8
beq x0, x0, EndForj    # Else go to EndForj

EndForj:

btl s1, s3, Fori      # If i < 16 goto Fori
beq x0, x0, EndFori    # Else go to EndFori

EndFori:
```

HW Questions:

1. Problem 1a. The new instruction added in the example is **addi**. Enter an instruction name (the mnemonic) like 'add'.
2. Problem 1a. The number of executed instruction in each iteration is **8**.
3. Problem 1a. The number of iteration is **100**.
4. Problem 1a. The total number of executed instructions is **802**.
5. Problem 1b. The number of SLLI instructions in the code is **1**.
6. Problem 1b. The number of LW instructions in the code is **4**.
7. Problem 1b. The number of iteration is **25**.
8. Problem 1b. The total number of executed instructions is **427**.
9. Problem 2. What instruction is used to perform '* 256'? Enter an instruction name (the mnemonic) like 'add'. **slli**
10. Problem 2. If T starts from address 1000, the address of T[10][4] is **1000+32*10* + *4* = 1336**.
11. Problem 2. If T starts from address 1000, the address of T[15][7] is? **1000 + 32*15* + *47* = 1508**
12. Problem 2. The number of words in T before of T[i][j] can be computed with a * i + j, where a is **8**
13. Problem 2. The displacement of T[i][j] relative to the starting address of T can be computed with x * i + y * j, where x is **32**.
14. Problem 2. The displacement of T[i][j] relative to the starting address of T can be computed with x * i + y * j, where y is **4**.
15. Problem 2. The number of SW instruction executed is **16*8=128**.