1A

```
Sll $t0, 16($s6), 2      // $t0 = $A[4]*4
Lw $t0, 16($s6)          // $t0 = A[4]
Sll $t1, 8($s6), 2       // $t0 = $A[2]*4
Lw $t1, 8($s6)                  // $t1 = A[2]
Add $t0, $t0, $t1        // $t0 = A[4]*4 + A[2]*4
Lw  $t1, 12($s6)         // $t1 = A[3]
Sub $t0, $t0, $t1        // $t0 = (A[4]*4 + A[2]*4) - A[3]
Sll $t0, $t0, 2          // $t0 = $t0*4
Add $t0, $t0, $s7        // $t0 =  ((A[4]*4 + A[2]*4) - A[3])* 4) + B[0]
Lw $t1, 0($t0)                  // $t1 =  $t0
Add $s0, $s1, $t1        // $s0 = $s1 + $t1
```

Final c instruction is

A = b + (A[4]*4 + A[2]*4 - A[3])*4 + B[0]

—----------

1B

```
Lw $t0, 16($s6)          // $t0 = A[4]
Lw $t1, 8($s6)                  // $t1 = A[2]
Add $t0, $t0, $t1        // $t0 = $t0 + $t1
Sll $t0, $t0, 2          // $t0 = $t0*4 (t0 = (A[4]+A[2])*4
Lw  $t1, 12($s6)         // $t1 = A[3]
Sub $t0, $t0, $t1        // $t0 = (A[4] + A[2])*4 - A[3]
Sll $t0, $t0, 2          // $t0 = $t0*4
Add $t0, $t0, $s7        // $t0 =  ((A[4] + A[2])*4 - A[3])* 4) + B[0]
Add $s0, $s1, $t0        // $s0 = $s1 + $t0 =  b + ((A[4] + A[2])*4 - A[3])*4 + B[0]
```

—----------

1C

```
lw $t0, 4($s6)           // $t0 = A[1]
lw $t1, 8($s6)           // $t1 = A[2]
add $t0, $t0, $t1        // $t0 = A[1] + A[2] = $t0 + $t1
sw $t0, 0($s7)           // $s7 = B[0] = A[1] + A[2]
lw $t1, 0($s7)           // $t1 = B[0] = A[1] + A[2]
add $t1, $t1, $t0        // $t1 = $t1 + $t0 = A[1] + A[2] + A[1] + A[2] = 2A[1] +2A[2]
sub $s0,  $s1, $t1       // $s0 = a = b - 2A[1] -2A[2]
```

Final c instruction is
a = b - 2*(A[1] + A[2])

1D

B[f]
```
Sll $t0, $s0, 2          // offset f by 4*
Add $t0, $t0, $s7        // creating pointer to B[f]
Lw $t2, 0($t0)                  // $t2 = B[f]
```

B[i-j]
```
Sub $t1, $s3, $s4        // $t1 = i-j
Sll $t1, $t1, 2          // offset i-j by 4*
Add $t1, t1, $s7         // creating pointer to B[i-j]
Lw $t3, 0($t1)                  // $t3 = B[i-j]
```

A[2] = B[i-j] - B[f]
```
Sub $t2, $t3, $t2        // $t2 = B[i-j] - B[f]
Sw $t2, 8($s6)                  // A[2] = B[i-j] - B[f]
```

—----------

2

toLower:
```
Lw $t0, 0($a0)                  // loading the first character into $t0
Beq $t0, $zero END  // if the character equal 00 then jump to end
Addi $t1, $zero, 65             // loading with immediate value to check
Blt $t0, $t1, LeaveAlone        // if $t0 is less than 65 skip it
Addi $t1, $zero, 90             // loading with immediate value to check
Bgt $t0, $t1, Leave Alone       // if $t0 is greater than 90 leave it alone
Add $t0 , $t0, 32        // converting the character from uppercase into lowercase
```

```
Sw $t0 , 0($a0)                    // store the new character into the string

LeaveAlone:
Addi $a0, $a0, 4        //incrementing array address to next character
J toLower

END:
```

3

Assuming $a0 holds address for array

```
Lw $t0, $zero // $t0 = 0 $t0 will hold the running sum
Lw $t1, $zero // $t1 = 0 $t1 is holding the variable 1
Sum:
Bgt $t1, 9 End          // if i value is greater than 9 or equals ten exit the loop
Add $t0, $t0, $a0       //adding the value in a[i] into the sum
Addi $t1, $t1, 4        //incrementing the value of i by 1
Add $a0, $a0, $t1       // incrementing the address of the array by 1 to point to next element in
array
J Sum

End:
```

—----------

4a

00000010 01010011 10100000 00100001

000000 10010 10011 10100 00000 100001

Op rs rt rd shamt funct

000000 op code means R instruction

Rs Source Register Number is register 18 or $s2

Rt Source Register 2 number is register 19 or $s3

Rd Register destination number is register 20 or $s4

Shamt is the shift amount in this case 0

Funct is the function in this case 100001 is addu

Final instruction is

Addu $s4, $s3, $s2

4B

—---

Or $t0, $t1, $s4

Op 000000
Rs 8 01000
Rt 9 or 01001
Rd 20 or 10100
Shamt 0 or 00000
Funct 100101
Final R instruction
00000001000010011010000000100101
4C
Op = 0 = 000000
Rs = 7 = 00111 = $a3
Rt = 4 = 00100 = $a0
Rd = 10 = 01010 = $t2
Shamt = 0 = 00000
Funct = 36 = 100100 = And
Final MIPS instruction is
And $t2, $a0, $a3

—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----—-----
—-----
5A
Sll $t2, $t0, 6
$t2 = 0b10101010101010101010101010000000
$t2 = 0xAAAAAA80

And $t2, $t2, $t1
$t2 = 28220880
$t2 = 00101000001000100000100010000000

—---

5B
Sll $t2, $t0, 4
$t2 = 0xAAAAAAA0

$t2 = 0b10101010101010101010101010100000

Andi $t2, $t2, 123456
$t2 = 2863434976

—----------

5C
Srl $t2, $t0, 3
$t2 = 0x15555555
$t2 = 0b00010101010101010101010101010101

Andi $t2, $t2, 0xFFEF
$t2 = 0x5545w