1. For the MIPS assembly instructions above, what is the corresponding C statements? Assume that variables f, g, h, i, and j are assigned to registers $s0, $s1, $s2, $s3, and $s4, respectively. Assume that the base address of the arrays A and B are in registers $s6 and $s7, respectively.

sll $t0, $s0, 2	$t0 = f * 4
add $t0, $s6, $t0	$t0 = &A[f]
sll $t1, $s1, 2	$t1 = g * 4
add $t1, $s7, $t1	$t1 = &B[g]
lw $s0, 0($t0)	f = A[f]
addi $t2, $t0, 4	$t2 = &A[f+1]
lw $t0, 0($t2)	t0 = A[f+1]
add $t0, $t0, $s0	t0 = A[f]+A[f+1]
sw $t0, 0($t1)	

∴ B[g] = A[f] + A[f+1];

2. Assume that registers $s0 and $s1 hold the values 0x8000000000000000 and 0xD000000000000000, respectively.

(a) what is the value of $t0 for the following assembly code?

```
add $t0, $s0, $s1
```

⇒ 0x5000 0000 0000 0000

(b) Is the result in $t0 the desired result, or has there been overflow?

⇒ overflow.

(c) For the contents of registers $s0, $s1 as specified above, what is the value of $t0 for the following assembly code?

```
sub $t0, $s0, $s1
```

⇒ 0xB000 0000 0000 0000

(d) Is the result in $t0 the desired result, or has there been overflow?

⇒ No overflow.

3. Provide the type and assembly language instruction for the following binary value : $0000\ 0010\ 0001\ 0000\ 1000\ 0000\ 0010\ 0000_{\text{two}}$. (Hint : Figure 1 may be helpful)

Name	Fields						Comments
Field size	6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	All MIPS instructions 32 bits
R-format	op	rs	rt	rd	shamt	funct	Arithmetic instruction format
I-format	op	rs	rt	address/immediate			Transfer, branch, imm. format
J-format	op	target address					Jump instruction format

(figure 1)

0000 0010 0001 0000 1000 0000 0010 0000$_{\text{two}}$

⇒ R-format, add $s0, $s0, $0

4. Provide the type, assembly language instruction, and binary representation of instruction described by the following MIPS fields :

op= 0, rs = 3 , rt = 2, rd = 3, shamt = 0, funct = 34

⇒ R-format

sub $v1, $v1, $v0

0000 0000 0110 0010 0001 1000 0010 0010

5. Consider the following MIPS loop :

```
Loop : slt   $t2, $0, $t1    //  0<t1 ?
       beq   $t2, $0, DONE   //  $t2 == 0 ?
       subi  $t1, $t1, 1
       addi  $s2, $s2, 2
       j     LOOP
DONE :
```

(a) Assume that the register $t1 is initialized to the value 10. What is the value in register $s2 assuming the $s2 is initially zero?

t1: 10 → 9 → 8 → ··· → 1 → 0 ∴ 20
s2: 0 → 2 → 4 → ··· → 18 → 20

(b) For each of the loops above, write the equivalent C code routine, Assume that the registers $s1, $s2, $t1, and $t2 are integers A, B, I, and temp, respectively.

⇒
```
while (I>0) {
    I = I - 1;
    B = B + 2;
}
```

(c) For the loops written in MIPS assembly above, assume that the register $t1 is initialized to the value N. How many MIPS instructions are executed?

⇒ 5N+2

6. Implement the following C code in MIPS assembly. Hint : Remember that the stack pointer must remain aligned on a multiple of 16.

```
int fib(int n) {
   if (n == 0)
       return 0;
   else if (n == 1)
       return 1;
   else
       return fib(n - 1) + fib(n - 2);
}
```

```
fib: addi  $sp, $sp, -16
     sw    $a0, 8($sp)      // n
     sw    $ra, 4($sp)      // return address
     sw    $s0, 0($sp)
     slti  $t0, $a0, 1      // if n<1 => $t0=1
     beq   $t0, $zero, L1   // n≥1 => L1
     add   $v0, $v0, $zero
     j     ret
L1:  slti  $t1, $a0, 2      // if n<2 => $t1=1
     beq   $t1, $zero, L2   // n≥2 => L2
     addi  $v0, $v0, 1
     j     ret
L2:  subi  $a0, $a0, 1      // n=n-1
     jal   fib
     add   $s0, $v0, $zero
     sub   $a0, $a0, 1      // n=n-2
     jal   fib
     add   $v0, $v0, $s0    // fib(n-1)+fib(n-2)
ret: lw    $s0, 0($sp)      // pop $s0
     lw    $ra, 4($sp)      // pop $ra
     lw    $a0, 8($sp)      // pop $a0
     addi  $sp, $sp, -16
     jr    $ra
```