COMP - Intro to Compilers II

* This form will record your name, please fill your name.

1. Consider the following C function and the output assembly code by gcc -O0 for MIPS:

```
int sum(int A[], int N) {
  int i, sum = 0;
  for(i=0; i<N; i++) {
    sum = sum + A[i];
  }
  return sum;
}

//sum(int*, (//sum(int*,) int):
1.      blez    $5,$L4
2.      sll     $5,$5,2
3.      addu    $5,$4,$5
4.      move    $2,$0
  $L3:
5.      lw      $3,0($4)
6.      addiu   $4,$4,4
7.      bne     $5,$4,$L3
8.      addu    $2,$2,$3
9.      j       $31
9.      nop
  $L4:
10.      j       $31
11.      move    $2,$0
```

2. Indicate the register the compiler assigned to each variable:
(1 Point)

	$2	$3	$4	$5	none
sum	◯	◯	◯	◯	◯
A	◯	◯	◯	◯	◯
i	◯	◯	◯	◯	◯
N	◯	◯	◯	◯	◯

3. Why did the compiler output an instruction after each j (jump) instruction?
(1 Point)

- ◯ because the target MIPS processor has a branch delay slot of 1
- ◯ because the branch may not be taken and in that case the instruction following j is executed

4. The compiler optimized the code considering that the loop executes at least 1 iteration
(1 Point)

- ◯ TRUE
- ◯ FALSE

5. The implementation of the FOR loop is similar to
(1 Point)

- ◯ a do...while
- ◯ a while

6. Consider the following C function and the output assembly code by gcc -O0 for MIPS:

```
#define N 1000
int sum(int A[]) {
  int i, sum = 0;
  for(i=0; i<N; i++) {
    sum = sum + A[i];
  }
  return sum;
}
```

```
//sum(int*) (//sum(int*)):
1.      addiu   $5,$4,4000
2.      move    $2,$0
    $L2:
3.      lw      $3,0($4)
4.      addiu   $4,$4,4
5.      bne     $5,$4,$L2
6.      addu    $2,$2,$3
7.      j       $31
8.      nop
```

7. Indicate the register the compiler assigned to each variable:
(1 Point)

	$2	$3	$4	$5	none
sum	◯	◯	◯	◯	◯
A	◯	◯	◯	◯	◯
i	◯	◯	◯	◯	◯

8. The compiler optimized the code considering that the loop executes at least 1 iteration
(1 Point)

◯ TRUE

◯ FALSE

9. The implementation of the FOR loop is similar to
(1 Point)

- ◯ a do...while
- ◯ a while