1. (20%) **Integer Representations and Memory Address of Array Elements.**

   1) Write the hexadecimal representation of the word
      $0101\ 1001\ 1101\ 0111\ 1110\ 1111\ 0011\ 1001_{two}$

      0x59D7EF39

   2) What is the binary word (32-bit) of the decimal number -9 in two's complement?

      11111 1111 1111 1111 1111 1111 1111 0111

   3) For the half-word
      $1111\ 1111\ 1111\ 1010_{two}$
      in two's complement. What is the word (32-bit) representation of same number? what decimal (base 10) number does it represent.

      1111 1111 1111 1111 1111 1111 1111 1010

      -6

   4) For a given two-dimensional array in C as follows
      `int A[8][16];`
      If the address of `A[1][4]` is `0x0FFA0040`, what is the memory address of `A[3][6];`

      The offset from A[1][4] to A[3][6] is (3-1)*16 + (6-4) elements, which is 0x22. The number of bytes of 0x22 elements is 0x22 * 4 = 0x88

      Thus the address of A[3][6] is 0x0FFA0040 + 0x88 = 0x0FFA00C8

Name:_____ Student #:_____

2. (10%) **Compile C to MIPS.** Assume that the variables `f, g, i`, and `j` are assigned to registers `$s0, $s1, $s2,` and `$s3` respectively. Assume that the base address of array `A` is in register `$s4` and each element is a 4-byte word. You can use temp registers `$t0, $t2, $t3, $t4 …`

   1) What is the corresponding MIPS assembly code for the C statement

```
f = g + i – j * 8;
```

```
sll  $t0, $s3, 3        # $t0 now has j * 8
add $t1, $s1, $s2       # $t1 now has g + i
sub $s0, $t1, $t0       # $s0 (f) now has g + i – j * 8
```

   2) What is the corresponding MIPS assembly code for the C statement

```
A[4] = A[6] + g;
```

```
lw $t0, 24($s4)         # A[6] is now in $t0
add $t1, $t0, $s1       # $t1 now has A[6] + g
sw $t1, 16($s4)         # A[4] now has A[6] + g
```

If the answer provide calculation of the offset of A[4] and A[6] and the solution is correct, 5 points bonus should be given

```
addi $t0, $zero, 6      # load 6 to $t0
sll $t0, $t0, 2         # $t0 now has 6*4
add $t0, $t0, $s4       # $t0 now has the address of A[6]
lw $t0, 0($t0)          # $t0 now has A[6]
add $t1, $t0, $s1       # $t1 now has A[6] + g

addi $t0, $zero, 4      # load 6 to $t0
sll $t0, $t0, 2         # $t0 now has 4*4
add $t0, $t0, $s4       # $t0 now has the address of A[4]

sw $t1, 0($t0)          # A[4] now has A[6] + g
```

3. (10%) **Logical and Branch Instructions.**

   1) Suppose the register `$t0` contains a (hexadecimal) value `0x0000004A`, what is the (hexadecimal or decimal) value of `$t2` after the following instructions:

```
sll $t0, $t0, 4
addi $t2, $t0, 8
```

0x0000004A: after sll 4, $t0 has 0x4A0; after addi, $t2 has 0x4A8 or 1192

   2) Suppose the register `$t0` contains a (hexadecimal) value `0x0000004A`, what is the (hexadecimal or decimal) value of `$t2` after the following instructions:

```
        addi $t1, $zero, 64
        bne  $t0, $t1, Else
        andi $t2, $t0, 1
        j Done
Else:   ori $t2, $t0, 1
Done:   ……
```

$t1 has 64, which is 0x40 and it is not equal to $t0 (0x4A), thus Else branch is taken and ori is executed. The ori instruction perform bitwise or operation of 0x4A and 1. It is 0100 1010 | 1 = 0100 1011 = 0x4B. Thus $t2 contains 0x4B or 75.

Name:_____ Student #:_____

4. (20%) **Compile C to MIPS.**

Using the MIPS implementation of `clear1` function given in the note as reference to convert the following C code to MIPS assembly. The variables `i` and `size` are assigned to registers `$t0` and `$a1` respectively. The base address of array `array` is in register `$a0` and each element is a 4-byte word. Feel free to use other registers such as `$t1` to `$t7` and `$s0` to `$s7`.

```
int array[], int size;
int i;
for (i=1; i<size-1; i+=1) {
   array[i] = array[i-1] + array[i] + array[i+1];
}
```

```
        addi $t0, $zero, 1    # i = 1
        addi $a1, $a1, -1     # size = size - 1;
loop1:  sll $t1,$t0,2     # $t1 = i * 4
        add $t2,$a0,$t1   # $t2 = &array[i]
        lw  $t7, 0($t2)   # $t7 has array[i]
        lw  $t6, -4($t2)  # $t6 has array[i-1]
        lw  $t5, 4($t2)   # $t5 has array[i+1]
        add $t6,$t6,$t7   # $t6 has array[i] + array[i-1]
        add $t5,$t5,$t6   # $t5 has array[i]+
                          # array[i-1]+array[i+1]
        sw $t5, 0($t2)    # array[i] now has the result
        addi $t0,$t0,1    # i = i + 1
        slt $t3,$t0,$a1   # $t3 = (i < size - 1)
        bne $t3,$zero,loop1 # if (…) goto loop1
```

In this answer, we use -4 and +4 as offset for the address of array[i-1] and array[i+1] elements with regards to the address of array[i]. If in the solution, the addresses for array[i], array[i-1] and array[i+1] are calculated separately, they are correct as well.

Name:_____ Student #:_____

5. (20%) **Performance.**

Assume processors P1, P2, P3 have same instruction set.

	P1	P2	P3
Clock Rate	4 GHz	5 GHz	2 GHz
CPI	3	2	5

1) Which processor has the best performance?
2) **For processor P1**, if it can execute a program in 10 seconds, find the number of instructions and number of cycles for this processor.
3) **For processor P2,** we are trying to reduce the execution time by 30% but it leads to an increase of 20% in the CPI. What should be the new clock rate of this processor?

The solution needs the formula CPUTime = InstructionCount * CPI * CycleTime = InstructionCount * CPI / ClockRate

For 1), for the same program, instruction count is the same.
Thus CPUTime(P1) = IC * 3/4 = 0.75 * IC
CPUTime(P2) = IC * 2/5 = 0.4 * IC
CPUTime(P3) = IC * 5/2 = 2.5 * IC
Thus processor P2 has the best performance.

For 2), Put the given parameters in the CPUTime formula, we have
$10 = IC * 3 / 4 * 10^{-9}$,
Thus $IC = 10 * 10^9 * 4/3 = 1.33 * 10^{10}$ number of instructions.
Number of cycles = $IC * CPI = 4 * 10^{10}$ cycles

For 3), CPUTime(P2) = 0.4* IC. IC = CPUTime(P2)/0.4

For new P2, we reduce the execution time by 30% (CPUTime(P2) * 0.7) and increase CPI by 20% (2*1.2), thus we have the following formula:

CPUTime(NewP2) = IC * NewCPI / NewClockRate

NewClockRate(P2) = IC * NewCPI/CPUTime(NewP2)
= CPUTime(P2)/0.4 * 2*1.2/(CPUTime(P2) * 0.7)
= 1.2 * 5 / 0.7 = 8.6GHz

Name:_____ Student #:_____

6. (20%) **Performance.**
Consider a program with four classes of instructions A, B, C and D. Refer to following table for the data.

	A	B	C	D
CPI	1	2	5	3
% of instructions	20%	20%	50%	10%

1) Can we achieve a 2x overall speedup by only improving the CPI of instruction C? If so, please calculate the new CPI of the instruction C. Show all the steps.

CPUTime = IC * CPI * CycleTime, since we only improve CPI, CycleTime does not change and IC stays the same.

CPUTime = IC * (0.2 * 1 + 0.2 * 2 + 0.5 * 5 + 0.1 * 3) * CycleTime = IC * 3.4 * CycleTime

To achieve 2x speedup, NewCPUTime should be CPUTime/2, which should be IC*1.7*CycleTime

NewCPUTime = IC * (0.2*1 + 0.2*2 + 0.5*NewCPI(C) + 0.1*3) * CycleTime = IC * 1.7 * CycleTime.

Thus NewCPI(C) = 1.6, we can achieve 2x speedup by improving CPI for C to 1.6