Name: _____

Problem 1 (18%). Assemble the following machine instructions into **binary**, use spaces to separate fields, and **registers** in their symbolic form ($ra NOT $31). Assume absolute jump addresses.

Field 1	Fields 2 and etc	instruction
100011	00011 11111 0000000000001100	`lw $ra, 12($v1)`
001000	10010 11001 0000000011111010	`addi $t9, $s2, 0xFA`
000000	00000 11101 00100 11100 000000	`sll $a0,$sp,0x1C`
000000	01011 00000 00101 00000 100000	`move $a1,$t3`
000011	00000000000000000000010100	`jal $20`
000000	1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0	`add $t0,$s2,$t0`

Problem 2 (7%).
Assume each part is **independent**. Assume absolute jump & branch addresses (no pc relative).
Fill in only registers that changed!
What is the value of the register or memory contents **after** the execution of the instruction.

Assume pc = 1000; $t1=12; $t2=3; $ra=50; memory[12]=0x17761453;

instruction	pc	$ra	$t1	memory[12]
`lw $ra, 12($0)`	1004	0x17761453		
`addi $t1, $t2, 0x01`	1004		4	
`sll $t1,$t2,2`	1004		12	
`clear $t1`	1004		0	
`jal $20`	20	1004		
`bge $t1,$t2,40`	40			
`ble $t1,$t2,40`	1008			

see problem 3.10 pseudoinstructions.

```
slt  $at,$t2,$t1
beq  $at, $zero, 40
```

Problem 3. (25%) Translate the following C code into MIPS. Please comment your code.
Assume **x** is $s0; **y** is $s1; **p** is $a0 and points to integers; **s** is $a1 and points to unsigned char.
Points will be taken off for assembler syntax errors.

(a) x = (x + (y – 3)) & 0xFF;

```
        addi    $t0, $s1, –3
        add     $t0, $s0, $t1
        andi    $s0, $t0, 0xFF
```

(b) x = (y > 3)? x + 1 : x – 1;

```
        addi    $t0, $zero, 3
        slt     $t1, $t0, $s1       # if ((3 < y )== 0) goto L1
        beq     $t1, $zero, L1
        addi    $s0, $s0, 1         # then
        j       L2
L1:     addi    $s0, $s0, –1        # else
L2:
```

(c) for(x=1; x; x++) { y++; }

```
        addi    $s0, $zero, 1       # x = 1;
L2:     beq     $s0, $zero, L1      # if (x = = 0) goto L1
        addi    $s1, $s1, 1         # y++ sames y=y+1
        addi    $s0, $s0, 1         # x++
        j       L2;
L1:
```

(d) p[5] = *p + y + *s; /*note: same as *(p+5) = *(p+0) + y + *(s + 0) */

```
        lbu     $t0,0($a1)      # *s   (points to unsigned char)
        add     $t1,$t0,$s1     # + y
        lw      $t2,0($a0)      # *p   (points to int)
        add     $t3, $t2, $t1   # *p + y + *s
        addi    $t4,$zero,20    # 5 * sizeof(int) = 5*4 = 20
        add     $t5,$t4,$a0     # address of p[5]
        sw      $t3,0($t5)      # *(p+5) =
```

(e) p[x+3] = s[x+2] | 0x01 /* note: same as *(p+x+3) = *(s+x+2) | 0x01 */

```
addi    $t0,$s0,2           # x + 2
add     $t1,$a1,$t0         # (s + x + 2)
lbu     $t2,0($t1)          # = *(s + x + 2)
ori     $t3,$t2,0x01
addi    $t4,$s0,3           # x+3
add     $t5,$t4,$t4         # 2*(x+2)
add     $t6,$t5,$t5         # 4*(x+2)
add     $t7,$a0,$t6         # p + x + 2
sw      $t3,0($t7)          # *(p+x+2) = $t3
```

```
# alternate solution
addi    $t0,$s0,2
add     $t1,$a1,$t0
lbu     $t2,0($t1)
ori     $t3,$t2,0x01
addi    $t4,$s0,3
add     $t5,$s0,$s0
add     $t6,$t5,$t5
add     $t7,$a0,$t6
sw      $t3,12($t7)
```

Problem 4. (25%) Translate the following code and add **comments**

```
void rgcd(int x, int y)  {
     register int  w=0, z=1;

     if (y = = 0)  { return x; }
     z = x + y;
     w = rgcd(y, z);
     return w + z;
}
```

(a) Write the prolog

```
          .text                  # read appendix A.6 and A
rgcd:
     add  $sp,$sp –12            # allocate space
     sw   $ra,8($sp)
     sw   $s1,4($sp)
     addi $s1,$zero,1            # z = 1;
     sw   $s0,0($sp)
     add  $s0,$zero,$zero        # w = 0
```

(b) Write the body

```
     bne  $a1,$zero,rgcd2        # if (y != 0) goto rgcd2
     add  $v0,$a0,$zero          # return x
     j    rgcd_epilog
rgcd2:
     add  $s1,$a0,$a1            # z=x + y
     add  $a0,$a1,$zero          # $a0 = y
     add  $a1,$s1,$zero          # $a1 = z
     jal  rgcd
     add  $s0,$v0,$zero          # w = return
     add  $v0,$s0,$s1            # return = w+z
```

(c) Write the epilog

```
rgcd_epilog:                     # not require to retore $a0, $a1
     lw   $s0,0($sp)             # restore caller's $s0
     lw   $s1,4($sp)             # restore caller's $s1
     lw   $ra,8(sp$)
     addi $sp,$sp,12             #free space
     jr   $ra
```

Problem 5. (10%) Translate the following global variables

(a) char x[6] = “hello”;

```
            .data
x:          .byte     ‘h’,’e’,’l’,’l’,’o’,0
```

(b)
```
struct treenode {
        struct treenode *left, *right;
        char     symbol[4];
} root;
```

```
                .data
root:
root_left:          .word     0
root_right:         .word     0
root_symbol:        .byte     0,0,0,0
```

Problem 6. (15%) Given the following instruction sequence in the table below.

Assume the (alu instructions are 6 clocks); (loads 8 clocks); (stores 7 clocks); (jumps 2 clocks); (branches 5 fall through/8 for branch);

(a) Show the **best** case timing path through the code showing annotations and total.
(b) Show the **worst** case timing path through the code showing annotations and total.

(c) What values will make this code execute the worst case?
$s0 = 0 and $s1 = 3

instruction	best case	worst case	
bne $s0,$zero,L2	8 branch	5 fall through	5 fall
addi $t0,$zero,3		6 alu	6 alu
bne $s1,$t0,L2		~~5 fall through~~	8 branch
addi $s1,$s1,1		~~6 alu~~	
j L1		~~2 jump~~	
L2: addi $s1,$zero,10	6 alu		6 alu
L1:			
Time	14 clocks	24 clocks	25 clocks[1] (correct)

[1]correct answer due to a quick thinking student