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

<table>
<thead>
<tr>
<th>Field 1</th>
<th>Fields 2 and etc</th>
<th>instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>001011</td>
<td>11101 00001 0000 0000 0011</td>
<td>sltiu $at,$sp,3</td>
</tr>
<tr>
<td>000000</td>
<td>0000000000 11111 0000 010000</td>
<td>mfhi $ra</td>
</tr>
<tr>
<td>000000</td>
<td>0000 01001 10001 0000 100010</td>
<td>neg $s1,$t1</td>
</tr>
<tr>
<td></td>
<td>(0)</td>
<td>(2’s complement $s1= -t1;)</td>
</tr>
<tr>
<td></td>
<td>($ra)</td>
<td>sub $s1,$0,$t1</td>
</tr>
<tr>
<td>000010</td>
<td>0000000000 0011 1110 1011 1111</td>
<td>j 0x7afc</td>
</tr>
<tr>
<td>(2)</td>
<td>(0x1ebf)</td>
<td>(0x7afc = byte offset)</td>
</tr>
<tr>
<td>101001</td>
<td>11110 00010 0000 0000 0000 1100</td>
<td>sh $v0,12($fp)</td>
</tr>
<tr>
<td>(0x29)</td>
<td>($fp)</td>
<td></td>
</tr>
<tr>
<td>0x03e00008</td>
<td>00000 11111 0000 0000 0000 01000</td>
<td>jr $ra</td>
</tr>
<tr>
<td>(0)</td>
<td>($ra)</td>
<td>(0)</td>
</tr>
</tbody>
</table>

**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 = 200; $s3=13; $s4=6; $ra=250; memory[8]=0xfedcba98; memory[12]=0x76543210;

<table>
<thead>
<tr>
<th>Instruction</th>
<th>pc</th>
<th>$ra</th>
<th>$s3</th>
<th>$s4</th>
<th>memory[8]</th>
<th>Memory[12]</th>
</tr>
</thead>
<tbody>
<tr>
<td>srl $s3, $s4, 3</td>
<td>204</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lui $s3, 0xffff</td>
<td>204</td>
<td>0xffff</td>
<td>0xFFFF0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lh $s4, 8($s4)</td>
<td>204</td>
<td>0x3210</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>addi $s3, $s3,-4</td>
<td>204</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sb $s4, 7($s4)</td>
<td>204</td>
<td>0x76063210</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bnez $ra, 600</td>
<td>600</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>j 250</td>
<td>250</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 3. (25%) Translate the following C code into MIPS. Please comment your code. Assume signed unless defined otherwise. The value \( x \) is \$s0; the value \( y \) is \$s1; the value \( s \) is \$s2; \( t \) is \$s3; the address of \( r \) is in \$s4; \( p \) is \$s5; the address of \( d \) is in \$s6, \( w \) is \$s7.

Points will be taken off for assembler syntax errors.

register unsigned int x, y; register int s, *t; struct { int a[3]; short b; } r, *p; short d[5]; char *w;

(a) \( s = *t; \)

\[
\text{lw } \$s2, 0(\$s3) \quad \#s = (\text{int}) *t
\]

(b) \( s = *w; \)

\[
\text{lb } \$s2, 0(\$s7) \quad \#s = (\text{char}) *w
\]

(c) \( s += d[x + 3]; \)

\[
\text{addiu } \$t0, \$s0, 3 \quad \#$t0=x+3
\]
\[
\text{addu } \$t1, \$t0, \$t0 \quad \#$t1=(x+3)\ast\text{sizeof(short)} = (x+3)\ast2
\]
\[
\text{addu } \$t3, \$s6, \$t1 \quad \#$t3=\{\text{address of } d+(x+3)\ast\text{sizeof(short)}\}
\]
\[
\text{lh } \$t4, 0(\$t3) \quad \#$t4= (\text{short}) d[x+3]
\]
\[
\text{add } \$s2, \$s2, \$t4 \quad \#s = (\text{signed int}) \{s + d[x+3]\}
\]

(d) \(*(d + 3) = s + r.b; /* offsets within struct 0 : a[0], 4: a[1], 8:a[2], 12 : b */\)

\[
\text{lh } \$t0, 12(\$s4) \quad \#$t0= (\text{short}) r.b
\]
\[
\text{add } \$t1, \$s2, \$t0 \quad \#$t1= (\text{signed int}) \{s+r.b\}
\]
\[
\text{sh } \$t1, 6(\$s6) \quad \#*(d+3)= (\text{short})d[3]\{6=3\ast\text{sizeof(short)}\}
\]

(e) for(x=0; x <= 10; x++) { s = (s >> 13); } /* or {s >>= 13} */

\[
\text{addi } \$t0, \$0, 10 \quad \#$t0 = 10
\]
\[
\text{addu } \$s0, \$0, \$0 \quad \#(\text{unsigned int}) x =0
\]
L2: \[
\text{bgt } \$s0, \$t0, L1 \quad \#\text{if ( } x > 10 \text{) goto L1}
\]
\[
\text{sra } \$s2, \$s2, 13 \quad \#(\text{signed int})s >> 13
\]
\[
\text{addiu } \$s0, \$s0, 1 \quad \#(\text{unsigned int}) x++;
\]
\[
\text{j } L2
\]
L1:
Problem 4. (25%) Translate the following code and add comments
Points will be taken off for assembler syntax errors.

```c
int unicodestrlen(short *s) {
    register int n = 0;

    for(i=0; s[i] != 0; i++) {n++;}

    return n;
}
```

(a) Write the prolog

```
#prolog is empty because:
• $t0..$t4 registers are only used (by convention these registers are not required to be saved).
• Function does not call another function, therefore no need to save $ra.
```

(b) Write the body (hint: write the body first; then write the prolog)

```
addi $t0,$0,0    #n=0;
addi $t1,$0,$0    #i=0;
L2: sll $t2,$t1,1 #$t2 = i * sizeof(short)
    add $t3,$a0,$t2 #$t3 = address(s + i)
    lh $t4,0($t3) #$t4 = *(s+i) = s[i]
    beq $t4,$0,L1
    addi $t0,$t0,1 #n++;
    addi $t1,$t1,1 #i++;
    j L2
L1: addi $v0,$t0,0 #return n
```

(c) Write the epilog

```
END: jr $ra    #return
```

Alternative Solution

```
addi $v0, $0, 0
L2: lh $t4, 0($a0)
    beq $t4, $0, END
    addi $v0, $0, 1
    addi $a0, $0, 2 /*2 = sizeof(short)*/
    j L2
END: jr $ra
```
Problem 5. (10%) Translate the following global variables and assign the location counter beginning at 700
(a) class xxstring {
    char argv;
    short montab[2];
    short (*daytab)[13];
    void (*strcpy)();
    class xxstring **next;
} *fsp;

(b) class ccstring {
    char argv;
    short montab[2];
    short (*daytab)[13];
    void (*strcpy)();
    class ccstring **next;
} *fcp;

Problem 6. (15%) Given the following instruction sequence in the table below.
Assume the (alu and slt instructions are 5 clocks); (loads 10 clocks); (stores 20 clocks); (jumps 2 clocks); (branches 3 fall through/6 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 for $s1,$s2,$s3,$s4 will make this code execute the worst case?
    $s1 < $s2 ; $s3,$s4 any values

<table>
<thead>
<tr>
<th>Instruction</th>
<th>best case</th>
<th>worst case</th>
</tr>
</thead>
<tbody>
<tr>
<td>slt $t1,$s1,$s2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>bne $t1,$0,L1</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>addi $t2,$zero,5</td>
<td>5</td>
<td>X</td>
</tr>
<tr>
<td>j L2</td>
<td>2</td>
<td>X</td>
</tr>
<tr>
<td>L1: slt $t2,$s3,$s4</td>
<td>X</td>
<td>5</td>
</tr>
<tr>
<td>beq $0,$0,L2</td>
<td>X</td>
<td>6</td>
</tr>
<tr>
<td>xori $s3,$s4,3</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>L2: addi $s1,$zero,10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Total Time</td>
<td>20</td>
<td>27</td>
</tr>
</tbody>
</table>