EECS 314 Test 1 Due: Wednesday February 6, 2002

Name: _____ Email: _____

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
001011 (0xb)	11101 00001 0000 0000 0000 0011 ($sp) ($at) (3)	`sltiu $at,$sp,3`
000000 (0)	0000000000 11111 00000 010000 (0) ($ra) (0) (0x10)	`mfhi $ra`

000000
(0)
	00000 01001 10001 00000 100010
	($0) ($t1) ($s1) (0) (0x22)
		`neg $s1,$t1`
		`(2's complement s1= -t1;)`
		`sub $s1,$0,$t1`

000010 (2)	0000000000 0001 1110 1011 1111 (0x1ebf) = word offset	`j 0x7afc` `(0x7afc = byte offset)`
101001 (0x29)	11110 00010 0000 0000 0000 1100 ($fp) ($v0) (12)	`sh $v0,12($fp)`
0x03e00008 000000 11111 0000 0000 0000 0000 01000 (0) ($ra) (0) (8)		`jr $ra`

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;

Instruction	pc	$ra	$s3	$s4	memory[8]	Memory[12]
`srl $s3, $s4, 3`	204		0			
`lui $s3, 0xffff`	204		0xFFFF0000			
`lh $s4, 8($s4)`	204			0x3210		
`addi $s3, $s3,–4`	204		9			
`sb $s4, 7($s4)`	204					0x76063210
`bnez $ra, 600`	600					
`j 250`	250					

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;

```
lw  $s2, 0($s3)     #s= (int) *t
```

(b) s = *w;

```
lb $s2, 0($s7)      #s = (char) *w
```

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

```
addiu $t0, $s0, 3   #$t0=x+3
addu  $t1, $t0, $t0 #$t1=(x+3)*sizeof(short) = (x+3)*2
addu  $t3, $s6, $t1 #$t3={address of d+(x+3)*sizeof(short)}
lh    $t4,  0($t3)  #$t4= (short) d[x+3]
add   $s2, $s2, $t4 #s = (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 */

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

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

```
     addi  $t0, $0, 10   #$t0 = 10
     addu  $s0, $0, $0   #(unsigned int ) x =0
L2:  bgt   $s0, $t0,L1   #if ( x > 10) goto L1
     sra   $s2, $s2,13   #(signed int)s >> 13
     addiu $s0, $s0, 1   #(unsigned int) x++;
     j L2
L1:
```

Problem 4. (25%) Translate the following code and add **comments**
Points will be taken off for assembler syntax errors.

```
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
```

→ `add $t2,$t1,$t1`

(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;
```

Decimal Location	Assembler definitions
700	fsp : .word 0
704	

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

Decimal Location	Assembler definitions
700	fcp : .word 0
704	

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

Instruction	best case	worst case
slt $t1,$s1,$s2	5	5
bne $t1,$0,L1	3	6
addi $t2,$zero,5	5	X
j L2	2	X
L1: slt $t2,$s3,$s4	X	5
beq $0,$0,L2	X	6
xori $s3,$s4,3	X	X
L2: addi $s1,$zero,10	5	5
Total Time	20	27