Name: ____________________________ Table: __________

Run \LaTeX\ again to produce the table
1. For each question below, give a short answer - a few words or symbols, maybe a sentence or two.

(a) (10 points) Give the instructions which will perform each of the following tasks:

Add registers $s0$ and $s1$, and put the sum into $s2$:

**Solution:** `add $s2, $s0, $s1`

Subtract three from register $t3$, and store it back into the same register:

**Solution:** `addi $t3, $t3, -3`

Read the two least-significant bits of $s7$, and store them into $t0$:

**Solution:** `andi $t0, $s7, 0x3`

Read a word from memory into $t0$; the address of the word is currently stored in the register $t1$:

**Solution:** `lw $t0, 0($t1)`

Load the address of the variable `foo`, and place it into the register $s4$:

**Solution:** `la $s4, foo`

(b) (10 points) Each of the following instructions is invalid. Explain what is wrong with each one.

`sub $s3, 10, $t0`

**Solution:** Two things (student will get credit if they mention either one):

- **SUB** doesn’t allow for a constant
- All I-format instructions have the constant as the 2nd input, no the first.

`andi $t0, 0xdeadbeef`

**Solution:** This has a 32-bit constant; I-format instructions only allow 16-bit constants.

(c) (5 points) Suppose that you have an integer stored in $s0$. Give the MIPS code to print that integer. (You don’t have to print anything other than the integer - not even a trailing newline!)

**Solution:**

```
addi $v0, $zero, 1
add $a0, $s0, $zero
syscall
```
2. (20 points) Assume that you have the variables `tom`, `dick`, `harry` loaded into registers `$t0`, `$t1`, `$t2` respectively.

Implement the following C code, including all required labels. (Represent "...TRUE_CODE..." and "...FALSE_CODE..." with similar lines in your assembly.)

```c
if (tom < dick && dick < harry)
{
    ...TRUE_CODE...
}
else
{
    ...FALSE_CODE...
}
```

Solution:

```assembly
slt $t8, $t0,$t1 # t8 = (tom < dick)
beq $t8,$zero, FALSE
slt $t8, $t1,$t2 # t8 = (dick < harry)
beq $t8,$zero, FALSE

# if we get here, then both conditions are TRUE.

...TRUE_CODE...
j AFTER

FALSE:

...FALSE_CODE...

AFTER:
```
3. (25 points) Assume that the following C variables are in memory (none are in registers, yet).

```c
int asdf = ...;
int qwerty = ...;
int uiop = ...;
```

Write MIPS code which implements the following C code. When the C code writes to a variable, make sure to update the related register in memory!

```c
asdf *= 3;
qwerty = uiop+asdf;
```

**Special Limitations:**

- You may modify any `tX` register, but **do not modify any `sX` register**.
- You may use `la` for each variable once - but do not use it more than once per variable.

(Comments are not required for this code - but they will help you get partial credit.)

```
Solution:

```
4. This question assumes some MIPS code (on the last page of this exam). The code sets up memory locations asdf, jkl, qwerty, zxcv, harold. The code then loads the values of some of these variables into the indicated MIPS registers. In answering these questions, you can assume this code has already been executed, and that the value of some of the variables are already in the indicated registers.

**Special Limitations:**

- Each question is independent of the other questions - that is, assume that the program has started over from scratch each time.
- You may need to read from memory - but **do not** write to memory unless specifically instructed.
- Do not modify any sX register, unless specifically instructed.

See the last page for the list of allowable instructions.

(a) (15 points) Put asdf+123-qwerty into $t3.

**Solution:**

```
addi $t0, $s0,123  # t0 = asdf+123
la $t1, qwerty    # t1 = &qwerty
lw $t1, 0($t1)   # t1 = qwerty
sub $t3, $t0,$t1 # t3 = asdf+123-qwerty
```

(b) (15 points) If (asdf == harold), then set $s7 to 1; otherwise, set it to zero.

**Solution:**

```
bne $s0,$s2, FALSE
    addi $s7, $zero,1
j AFTER
FALSE:
    addi $s7, $zero,0
AFTER:
```
# values are hidden so that you can’t hardcode the answers!
.data
asdf: .word xxx
jkl: .word xxx
qwerty: .word xxx
zxcv: .word xxx
harold: .word xxx

.text
main:
    # set $s0 = asdf
    la $s0, asdf
    lw $s0, 0($s0)

    # set $s1 = zxcv
    la $s1, zxcv
    lw $s1, 0($s1)

    # set $s2 = harold
    la $s2, harold
    lw $s2, 0($s2)

Allowable Instructions
When writing MIPS assembly, the only instructions that you are allowed to use (so far) are:

- add, addi, sub
- beq, bne, j
- slt, slti
- and, andi, or, ori, nor, nori, xor, xori
- sll, srl, sra
- lw, lh, lb, sw, sh, sb
- la
- syscall