What is your name?: ___________________________(4 points)

There are two sections:
   I. Short Questions . . . . . . . . .30 points; (15 questions, 2 points each)
   II. Short Sections of Code . . .66 points; (11 questions, 6 points each)
--------------------
   96 points + 4 for name = 100 points total

This test is worth 15% of your final grade. This test is open book and open notes. You have 60 minutes.

I. True/False: (2 pts. each)

T F 1. In the course we have been studying MIPS assembly language programming, where MIPS stands for Millions of Instructions Per Second.

T F 2. There is at least one MIPS assembler instruction that can directly store a result of an ALU calculation into memory.

T F 3. In addition to the 32 MIPS register file registers having mnemonic names such as $t3, the registers can be referred to by their number, starting from $1 up to and including $32.

T F 4. Usage conventions for register use (what registers are used for what purposes) are strictly enforced by the assembler.

T F 5. The value part (in this case 0xffff) of the instruction
   li $t3, 0xffff
   can actually be a 32-bit value.

T F 6. A carry-out at the most significant bit after an addition of two signed numbers always indicates overflow.

T F 7. A carry-out at the most significant bit after an addition of two unsigned numbers always indicates overflow.

T F 8. Overflow will never occur if two signed numbers of different signs are added.

T F 9. Even if they are not being used as parameters, it is necessary to always leave room on the stack for $a0 through $a3.

T F 10. In generating code, compilers can translate a multi-dimensional array into a row-major, single-dimension array, where a mapping function is implemented to index into the correct array element.

T F 11. Instructions can be treated as data in an assembler program.

T F 12. Memory-mapped I/O means that a copy of an I/O device memory is kept in main memory.

T F 13. The MIPS pipeline has a separate instruction cache and data cache because the two caches store values in different formats.
14. The trap handler discussed in class is an example of reentrant code since it can be called by more than one process.

15. Consider the code segment:

```assembly
.data
first: .ascii "One\n"
second: .ascii "Two\n"
third: .ascii "Three\n"
done: .asciiz "Done."
.text
main:
  la $a0, first
  li $v0, 4
  syscall
  la $a0, done
  li $v0, 4
  syscall

The output of running this program segment is:
One
Done.
```

III. Short Answer: (6 points each)
Where there are multiple steps, please circle your final answer. You must show your work for credit.

1. Consider the following instruction:
   ```assembly
   srl $a0,$t5,3
   ```
   Write this instruction in hexadecimal.

2. After running a program using PCSpim, the logfile is saved and examined. You notice that the instruction
   ```assembly
   bltz $v0 0x5c
   ```
is translated into hex as:
   ```assembly
   0x04400017
   ```
   Where did the rightmost two hex digits (17) come from?
3. In translating a high-level language if-then statement such as

   if ($a0 > 5) then ...

   the resulting assembler code uses a `blez` statement rather than `bgt`. Why is this?

4. An assembler program could have the following statement in the `.data` segment:

   ```
   table: .word first second third fourth
   ```

   where first second third and fourth are labels of subroutines elsewhere in the program. The program can then use this as a jumptable to select the appropriate subroutine to run. How is it that these labels get translated into actual addresses?

5. If the $at register suddenly "broke" on a machine, what wouldn’t we be able to do anymore?
6. When using PCSpim, assume your program has a .data segment that includes the line:
   .asciiz "Exit"
   After running the program, assume you save the log file. How would the text "Exit" be encoded as hex
   in the logfile, assuming it starts on a word boundary?

7. Assuming the least-significant bit in a register is bit 0, write the code needed to print out only bit 7 on the
   console.

8. Assume your assembler program calls subprogram $B$, which calls subprogram $C$, which recursively calls
   subprogram $C$ again. Additionally assume these subprogram calls are storing parameters on the stack.
   Must $ra be stored on the stack for subprogram $C$? Why or why not?

9. Why do some compilers generate code that uses $fp$? What would be the problem if $fp$ were not used?
10. Consider the following section of code:

```
addi $a0, $a0, 4  # line 1
srl  $a0, $a0, 2  # line 2
sll  $a0, $a0, 2  # line 3
li   $v0, 9
syscall
```

a) What is the purpose of the first 3 lines?

b) In what context is this used?

11. Is a nop instruction always needed after a branch when running on a pipelined machine? Explain why or why not.
What is your name?: ____________________________ (4 points)

There are two sections:

I. Short Questions . . . . . . . . . 30 points; (15 questions, 2 points each)
II. Short Sections of Code . . .66 points; (11 questions, 6 points each)

---------------
96 points + 4 for name = 100 points total

This test is worth 15% of your final grade. This test is open book and open notes. You have 60 minutes.

I. True/False: (2 pts. each)

T  F  1. In the course we have been studying MIPS assembly language programming, where MIPS stands for Millions of Instructions Per Second.

T  F  2. There is at least one MIPS assembler instruction that can directly store a result of an ALU calculation into memory.

T  F  3. In addition to the 32 MIPS register file registers having mnemonic names such as $t3, the registers can be referred to by their number, starting from $1 up to and including $32.

T  F  4. Usage conventions for register use (what registers are used for what purposes) are strictly enforced by the assembler.

T  F  5. The value part (in this case 0xffff) of the instruction
   li $t3, 0xffff
   can actually be a 32-bit value.

T  F  6. A carry-out at the most significant bit after an addition of two signed numbers always indicates overflow.

T  F  7. A carry-out at the most significant bit after an addition of two unsigned numbers always indicates overflow.

T  F  8. Overflow will never occur if two signed numbers of different signs are added.

T  F  9. Even if they are not being used as parameters, it is necessary to always leave room on the stack for $a0 through $a3.

T  F  10. In generating code, compilers can translate a multi-dimensional array into a row-major, single-dimension array, where a mapping function is implemented to index into the correct array element.

T  F  11. Instructions can be treated as data in an assembler program.

T  F  12. Memory-mapped I/O means that a copy of an I/O device memory is kept in main memory.

T  F  13. The MIPS pipeline has a separate instruction cache and data cache because the two caches store values in different formats.
T  F  14. The trap handler discussed in class is an example of reentrant code since it can be called by more than one process.

T  F  15. Consider the code segment:

```assembly
.data
    first: .ascii "One\n"
    second: .ascii "Two\n"
    third: .ascii "Three\n"
    done: .asciiz "Done."

.text
    main:
        la $a0, first
        li $v0, 4
        syscall
        la $a0, done
        li $v0, 4
        syscall
```

The output of running this program segment is:

One
Done.

III. Short Answer: (6 points each)
Where there are multiple steps, please circle your final answer. You must show your work for credit.

1. Consider the following instruction:
   `srl $a0,$t5,3`
   Write this instruction in hexadecimal.

2. After running a program using PCSpim, the logfile is saved and examined. You notice that the instruction
   `bltz $v0 0x5c`
   is translated into hex as:
   `0x04400017`
   Where did the rightmost two hex digits (17) come from?
3. In translating a high-level language if-then statement such as

\[
\text{if ($a_0 > 5) then ...}
\]

the resulting assembler code uses a `blez` statement rather than `bgt`. Why is this?

4. An assembler program could have the following statement in the `.data` segment:

\[
\text{table: .word first second third fourth}
\]

where first second third and fourth are labels of subroutines elsewhere in the program. The program can then use this as a jumptable to select the appropriate subroutine to run. How is it that these labels get translated into actual addresses?

5. If the $at register suddenly "broke" on a machine, what wouldn’t we be able to do anymore?
6. When using PCSpim, assume your program has a .data segment that includes the line:
   .asciiz "Exit"
   After running the program, assume you save the log file. How would the text "Exit" be encoded as hex
   in the logfile, assuming it starts on a word boundary?

7. Assuming the least-significant bit in a register is bit 0, write the code needed to print out only bit 7 on the
   console.

8. Assume your assembler program calls subprogram B, which calls subprogram C, which recursively calls
   subprogram C again. Additionally assume these subprogram calls are storing parameters on the stack.
   Must $ra be stored on the stack for subprogram C? Why or why not?

9. Why do some compilers generate code that uses $fp? What would be the problem if $fp were not used?
10. Consider the following section of code:

```
addi $a0, $a0, 4  # line 1
srl  $a0, $a0, 2  # line 2
sll  $a0, $a0, 2  # line 3
li   $v0, 9
syscall
```

a) What is the purpose of the first 3 lines?

b) In what context is this used?

11. Is a nop instruction always needed after a branch when running on a pipelined machine? Explain why or why not.