Problem 3.2a

	Before `foo`'s `jal`		**After `bar`'s full startup**
`$fp`	prev stack		prev stack
	`$t0`		`$t0`
	`$t1`		`$t1`
`$sp`	`$t2`		`$t2`
	arg5	`$fp`	arg5
			arg3=`$a2`
			arg1=`$a0`
			`$ra`
			`$fp`
			`$s0`
		`$sp`	`$s1`

Problem 3.2b

	Before `foo`'s `jal`		**After `bar`'s prologue**		**After `bar`'s full startup**
`$fp`	prev stack		prev stack		prev stack
	`$t0`		`$t0`		`$t0`
	`$t6`		`$t6`		`$t6`
`$sp`	`$t7`		`$t7`		`$t7`
		`$fp`		`$fp`	
					arg2=`$a1`
					arg1=`$a0`
			`$ra`		`$ra`
		`$sp`	`$fp`		`$fp`
					`$s3`
				`$sp`	`$s4`

Problem 3.2c

	Before `foo`'s `jal`		**After `bar`'s prologue**		**After `bar`'s full startup**
`$fp`	prev stack		prev stack		prev stack
`$sp`	`$t2`		`$t2`		`$t2`
	arg8	`$fp`	arg8	`$fp`	arg8
	arg7		arg7		arg7
	arg6		arg6		arg6
	arg5		arg5		arg5
					arg4=`$a3`
					arg1=`$a0`
			`$ra`		`$ra`
		`$sp`	`$fp`		`$fp`
					`$s4`
					`$s5`
					`$s6`
				`$sp`	`$s7`

Problem 3.2d

	Before `foo`'s `jal`		**After `bar`'s prologue**		**After `bar`'s full startup**
`$fp`	prev stack		prev stack		prev stack
	`$t0`		`$t0`		`$t0`
	`$t5`		`$t5`		`$t5`
`$sp`	`$t6`		`$t6`		`$t6`
		`$fp`		`$fp`	
					arg2=`$a1`
			`$ra`		`$ra`
		`$sp`	`$fp`		`$fp`
					`$s0`
					`$s1`
					`$s2`
				`$sp`	`$s7`

Problem 3.3a

Registers to save: `$s0`, `$s1`, `$s3`, `$t0`, `$t1`, `$t2`

Call to make: `foo(1, 2)`

```
caller:
    # ... rest of code

    addiu   $sp,    $sp,    -12 # allocate space to save temp registers
    sw      $t0,    0($sp)      # save $t0
    sw      $t1,    4($sp)      # save $t1
    sw      $t2,    8($sp)      # save $t2

    addiu   $a0,    $zero,  1   # first arg = 1
    addiu   $a1,    $zero,  2   # second arg = 2

    jal     foo                 # call foo

    # Cleanup
    lw      $t0,    0($sp)      # restore $t0
    lw      $t1,    4($sp)      # restore $t1
    lw      $t2,    8($sp)      # restore $t2

    addiu   $sp,    $sp,    12  # deallocate space for temp registers
```

Problem 3.3b

Registers to save: `$s2`, `$s4`, `$s6`, `$t1`, `$t3`, `$t5`

Call to make: `bar(0, 0x1234, str)`

```
caller2:
    # ... rest of code

    addiu   $sp,    $sp,    -12     # allocate space to save temp registers
    sw      $t1,    0($sp)          # save $t1
    sw      $t3,    4($sp)          # save $t3
    sw      $t5,    8($sp)          # save $t5

    addiu   $a0,    $zero,  0       # first arg = 0
    addiu   $a1,    $zero,  0x1234  # second arg = 0x1234
    la      $a2,    str             # third arg = address of str (first char in str)

    jal     bar                     # call bar

    # Cleanup
    lw      $t1,    0($sp)          # restore $t1
    lw      $t3,    4($sp)          # restore $t3
    lw      $t5,    8($sp)          # restore $t5

    addiu   $sp,    $sp,    12      # deallocate space for temp registers
```

Problem 3.3c

Registers to save: `$t1`, `$t2`, `$t3`, `$s0`, `$s1`

Call to make: `fred(123, 456, 0xabc, 0xdef, 0, 10)`

```
caller3:
    # ... rest of code

    addiu   $sp,    $sp,        -16     # allocate space to save temp registers
    sw      $t1,    0($sp)              # save $t1
    sw      $t2,    4($sp)              # save $t2
    sw      $t3,    8($sp)              # save $t3
    sw      $t4,    12($sp)             # save $t4

    addiu   $a0,    $zero,      123     # first arg = 123
    addiu   $a1,    $zero,      456     # second arg = 456
    addiu   $a2,    $zero,      0xabc   # third arg = 0xabc
    addiu   $a3,    $zero,      0xdef   # fourth arg = 0xdef

    # Extra args
    addiu   $t0,    $zero,      0       # fifth arg = 0
    sw      $t0,    -8($sp)             # store extra arg 1 (arg5) two words after stack po
    addiu   $t0,    $zero,      10      # sixth arg = 10
    sw      $t0,    -4($sp)             # store extra arg 2 (arg6) one word after stack poi

    jal     fred                        # call fred

    # Cleanup
    lw      $t1,    0($sp)              # restore $t1
    lw      $t2,    4($sp)              # restore $t2
    lw      $t3,    8($sp)              # restore $t3
    lw      $t4,    12($sp)             # restore $t4

    addiu   $sp,    $sp,        16      # deallocate space for temp registers
```

Problem 3.3d

Registers to save: `$t0`, `$t1`, `$s1`, `$s3`, `$s5`, `$s7`

Call to make: `qwerty('a', 10, 'B', -2, 0xffff)`

```
caller4:
    # ... rest of code

    addiu   $sp,    $sp,        -8      # allocate space to save temp registers
    sw      $t0,    0($sp)              # save $t0
    sw      $t1,    4($sp)              # save $t1

    addi    $a0,    $zero,      'a'     # first arg = 'a' (ASCII value of 'a')
    addiu   $a1,    $zero,      10      # second arg = 10
    addi    $a2,    $zero,      'B'     # third arg = 'B' (ASCII value of 'B')
    addiu   $a3,    $zero,      -2      # fourth arg = -2

    addi    $t0,    $zero,      0xffff  # fifth arg = 0xffff
    sw      $t0,    -4($sp)             # store extra arg 1 (arg5) one word after stack poi

    jal     qwerty                      # call qwerty

    # Cleanup
    lw      $t0,    0($sp)              # restore $t0
    lw      $t1,    4($sp)              # restore $t1

    addiu   $sp,    $sp,        8       # deallocate space for temp registers
```

Problem 3.4a

Registers which will be modified: `$s0`, `$s1`, `$s3`, `$t0`, `$t1`, `$t2`

Number of Parameters: 3

Parameters to Save on Stack: `$a2`

```
callee1:
    # --- Standard Prologue ---

    addiu   $sp,    $sp,        -24 # allocate space for args 1-4, ra, and fp
    sw      $fp,    0($sp)          # save fp
    sw      $ra,    4($sp)          # save ra
    addiu   $fp,    $sp,        20  # set fp to point to top of frame

    # --- Save Registers and Args ---

    # Save args that need to be persisted
    # arg1 would go to 8($sp)
    # arg2 would go to 12($sp)
    sw      $a2,    16($sp)         # save arg3 to 16($sp)
    # arg4 would go to 20($sp)

    # Save $sX registers used by this function
    addiu   $sp,    $sp,        -12 # allocate space to save $sX registers
    sw      $s3,    0($sp)          # save s3
    sw      $s1,    4($sp)          # save s1
    sw      $s0,    8($sp)          # save s0


    # ... rest of code ...


    # --- Restore Registers ---

    # Restore $sX registers used by this function
    lw      $s3,    0($sp)          # restore s3
    lw      $s1,    4($sp)          # restore s1
    lw      $s0,    8($sp)          # restore s0
    addiu   $sp,    $sp,        12  # deallocate space for $sX registers

    # --- Standard Epilogue ---

    lw      $fp,    0($sp)          # restore fp
    lw      $ra,    4($sp)          # restore ra
    addiu   $sp,    $sp,        24  # deallocate space for args 1-4, ra, and fp
```

```
        jr      $ra                     # return
```

Problem 3.4b

Registers which will be modified: `$s2`, `$s4`, `$s6`, `$t1`, `$t3`, `$t5`

Number of Parameters: 3

Parameters to Save on Stack: None

```
callee2:
    # --- Standard Prologue ---

    addiu   $sp,    $sp,        -24 # allocate space for args 1-4, ra, and fp
    sw      $fp,    0($sp)          # save fp
    sw      $ra,    4($sp)          # save ra
    addiu   $fp,    $sp,        20  # set fp to point to top of frame

    # --- Save Registers and Args ---

    # Save args that need to be persisted
    # arg1 would go to 8($sp)
    # arg2 would go to 12($sp)
    # arg3 would go to 16($sp)
    # arg4 would go to 20($sp)

    # Save $sX registers used by this function
    addiu   $sp,    $sp,        -12 # allocate space to save $sX registers
    sw      $s6,    0($sp)          # save s6
    sw      $s4,    4($sp)          # save s4
    sw      $s2,    8($sp)          # save s2


    # ... rest of code ...


    # --- Restore Registers ---

    # Restore $sX registers used by this function
    lw      $s6,    0($sp)          # restore s6
    lw      $s4,    4($sp)          # restore s4
    lw      $s2,    8($sp)          # restore s2
    addiu   $sp,    $sp,        12  # deallocate space for $sX registers

    # --- Standard Epilogue ---

    lw      $fp,    0($sp)          # restore fp
    lw      $ra,    4($sp)          # restore ra
    addiu   $sp,    $sp,        24  # deallocate space for args 1-4, ra, and fp
```

```
        jr      $ra                     # return
```

Problem 3.4c

Registers which will be modified: `$t1`, `$t2`, `$t3`, `$s0`, `$s1`

Number of Parameters: 6

Parameters to Save on Stack: `$a0`, `$a1`, `$a2`, `$a3`

```
callee3:
    # --- Standard Prologue ---

    addiu   $sp,    $sp,        -32 # allocate space for args 1-4, ra, and fp, and two extr
    sw      $fp,    0($sp)          # save fp
    sw      $ra,    4($sp)          # save ra
    addiu   $fp,    $sp,        28  # set fp to point to top of frame

    # --- Save Registers and Args ---

    # Save args that need to be persisted
    sw      $a0,    8($sp)          # save arg1
    sw      $a1,    12($sp)         # save arg2
    sw      $a2,    16($sp)         # save arg3
    sw      $a3,    20($sp)         # save arg4

    # Save $sX registers used by this function
    addiu   $sp,    $sp,        -8  # allocate space to save $sX registers
    sw      $s1,    0($sp)          # save s1
    sw      $s0,    4($sp)          # save s0


    # ... rest of code ...


    # --- Restore Registers ---

    # Restore $sX registers used by this function
    lw      $s1,    0($sp)          # restore s1
    lw      $s0,    4($sp)          # restore s0
    addiu   $sp,    $sp,        8   # deallocate space for $sX registers

    # --- Standard Epilogue ---

    lw      $fp,    0($sp)          # restore fp
    lw      $ra,    4($sp)          # restore ra
    addiu   $sp,    $sp,        32  # deallocate space for args 1-4, ra, and fp, and two ex
    jr      $ra                     # return
```

Problem 3.4d

Registers which will be modified: `$t2`, `$t3`, `$t4`, `$t5`, `$t6`, `$s1`

Number of Parameters: 5

Parameters to Save on Stack: `$a1`

```
callee4:
    # --- Standard Prologue ---

    addiu   $sp,    $sp,        -28 # allocate space for args 1-4, ra, and fp, and one extra
    sw      $fp,    0($sp)          # save fp
    sw      $ra,    4($sp)          # save ra
    addiu   $fp,    $sp,        24  # set fp to point to top of frame

    # --- Save Registers and Args ---

    # Save args that need to be persisted
    sw      $a1,    12($sp)         # save arg2

    # Save $sX registers used by this function
    addiu   $sp,    $sp,        -4  # allocate space to save $sX registers
    sw      $s1,    0($sp)          # save s1


    # ... rest of code ...


    # --- Restore Registers ---

    # Restore $sX registers used by this function
    lw      $s1,    0($sp)          # restore s1

    # --- Standard Epilogue ---

    lw      $fp,    0($sp)          # restore fp
    lw      $ra,    4($sp)          # restore ra
    addiu   $sp,    $sp,        28  # deallocate space for args 1-4, ra, and fp, and one ext
    jr      $ra                     # return
```

Problem 4.4a

```
    add     $s0,    $s1,        $s2
```

Converted to C:

```c
int s0;       // s0
int s1 = ...; // s1
int s2 = ...; // s2

s0 = s1 + s2;
```

Problem 4.4b

```
    addi    $v0,    $zero,      1
    add     $a0,    $s0,        $zero
    syscall
```

Converted to C:

```c
int s0 = ...; // s0
printf("%d", s0);
```

Problem 4.4c

```
.data
foo:    .word   1234
bar:    .word   0

.text
    la      $t0,    foo
    la      $t1,    bar
    sw      $t0,    0($t1)
```

Converted to C:

```
int foo = 1234;
int *bar = ...;

bar = &foo;
```

Problem 4.4d

```
    beq     $s0,        $s1,        TRUE
    bne     $s2,        $zero,      TRUE
    j       FALSE

TRUE:
    add     $s3,        $zero,      $zero
    j       AFTER_IF

FALSE:
    addi    $s3,        $s3,        1

AFTER_IF:
```

Converted to C:

```c
int s0 = ...; // s0
int s1 = ...; // s1
int s2 = ...; // s2
int s3 = ...; // s3

if (s0 == s1 || s2 != 0)
{
  s3 = 0;
}
else
{
  s3++;
}
```

Problem 4.4e

```
    addi    $a0,        $zero,      123
    addi    $a1,        $zero,      456
    jal     otherFunc

    add     $t0,        $v0,        $zero
    addi    $v0,        $zero,      1
    add     $a0,        $t0,        $zero
    syscall
```

Converted to C:

```c
int otherFunc(int a0, int a1);

printf("%d", otherFunc(123, 456));
```

Problem 4.4f

```
.data
MSG:    .asciiz "Still a multiple of 4!\n"
.text
LOOP:
    andi    $t0,    $s0,        0x3
    bne     $t0,    $zero,      END_LOOP
    addi    $v0,    $zero,      4
    la      $a0,    MSG
    syscall
    srl     $s0,    $s0,        2
    j       LOOP
END_LOOP:
```

Converted to C:

```
int s0 = ...; // s0

while (s0 % 4 == 0)
{
  print("Still a multiple of 4!\n");
  s0 = s0 / 4;
}
```

Problem 4.4g

```
.data
foo:
    .word   3
caseIsImportant:
    .byte   0
    .byte   0
    .byte   0
    .byte   0
.text
    la      $s0,    foo
    lw      $s0,    0($s0)
    la      $t1,    caseIsImportant
    add     $t2,    $t1,                $s0
    lb      $t3,    0($t2)
    addi    $v0,    $zero,              11
    add     $a0,    $t3,                $zero
    syscall
```

Converted to C:

```
int foo = 3;                                // Equivalent to .word 3
char caseIsImportant[4] = {0, 0, 0, 0}; // Equivalent to .byte 0, .byte 0, .byte 0, .byte 0

// Print the character in the foo-th index of caseIsImportant
printf("%c", caseIsImportant[foo]);
```

Problem 4.4h

```
    addi    $s0,    $zero,      100
LOOP:
    slt     $t0,    $s0,        $s7
    bne     $t0,    $zero,      LOOP_END
    addi    $v0,    $zero,      1
    add     $a0,    $s0,        $zero
    syscall
    addi    $v0,    $zero,      11          # print_char('\n')
    addi    $a0,    $zero,      0xa
    syscall
    addi    $s0,    $s0,        -1
    j       LOOP
LOOP_END:
```

Converted to C:

```c
int s7 = ...; // s7

for (int s0 = 100; s0 >= s7; s0--)
{
  printf("%d\n", s0);
}
```