Homework 3

Diego Rodrigues

April 1, 2024

1 Question 1

Assume the function declaration for func is "int func (int a, int b);" The code for function f is as follows:

```
Int f (int a, int b, int c, int d) {
    Return func (func (a, b), c + d)
}
```

1. Translating function f into MIPS assembly language: if you need to use registers $t0 through t7$, use the lower-number registers first.
2. Can we use the tail-call optimization in this function? If no, explain why not. If yes, what is the difference in the number of executed instructions in f with and without the optimization?
3. Right before function f returns, what do we know about contents of registers $t5$,s3, ra, andsp? Keep in mind that we know what the entire function f looks like, but for function func we only know its declaration.

1.1 Answer a

```
f:
    addi    $sp, $sp, -20
    sw      $ra, 0($sp)
    sw      $a0, 4($sp)
    sw      $a1, 8($sp)
    sw      $a2, 12($sp)
    sw      $a3, 16($sp)
    # func is assumed to preserve arguments
    jal func # func (a, b)
    move    $a0, $v0
    add     $a1, $a2, $a3
    jal func # func (func (a, b), c + d)
    lw      $a3, 16($sp)
    lw      $a2, 12($sp)
```

```
    lw      $a1, 8($sp)
    lw      $a0, 4($sp)
    lw      $ra, 0($sp)
    addi    $sp, $sp, 20
    # notice that the return value is in $v0
    jr      $ra
```

1.2 Answer b

Tail-call optimization cannot be used in this function because f does not call itself.

1.3 Answer c

Right before function f returns, we know the following about the contents of registers $t5$,s3, ra, andsp:

- $t5 : We do not know what is in$t5, but we know that it is not preserved by f.
- $s3 : We do not know what is in$s3, is the c argument of f.
- $ra : We know that$ra contains the return address of the function f.
- $sp : We know that$sp points to the top of the stack frame of the function f.

2 Question 2

Write a program in MIPS assembly language to convert an ASCII number string containing integer decimal strings, to an integer. Your program should expect register $a0 to hold the address of a null-terminated string containing some combination of the digits 0 through 9. Your program should compute the integer value equivalent to this string of digits, then place the number in register $v0. If a non-digit character appears anywhere in the string, your program should stop with the value -1 in register $v0. For example, if register $t9 points to a sequence of three bytes 50_{10} , 52_{10} , 0_{10} (the null-terminated string "24"), then when the program stops, register $v0 should contain the value 24_{10}.

```
strtoint:
    addi    $sp, $sp, -8
    sw      $ra, 0($sp)
    sw      $a0, 4($sp)
    li      $v0, 0
    li      $t0, $zero, 0x30 # ASCII '0'
    li      $t1, $zero, 0x39 # ASCII '9'
lp:
    lb      $t2, 0($a0)
    beq     $t2, $zero, done # check for null terminator
```

```
    # check non-digit
    blt     $t2, $t0, error
    bgt     $t2, $t1, error

    ## convert to integer
    sub     $t2, $t2, $t0
    mul     $v0, $v0, 10
    add     $v0, $v0, $t2
    addi    $a0, $a0, 1 # move to next character
    j       lp

error:
    li      $v0, -1
    j       done

done:
    lw      $a0, 4($sp)
    lw      $ra, 0($sp)
    addi    $sp, $sp, 8
    jr      $ra
```

3 Question 3

1. Write the MIPS assembly code that creates the 32-bit constant $00100000000000010100100100100100_2$ and stores that value to register $t1.

2. If the current value of the PC is 00000000_{16} , can you use a single jump instruction to get to the PC address as shown by the 32-bit constant in a.? Explain.

3. If the current value of the PC is 00000600_{16} , can you use a single branch instruction to get to the PC address as shown by the 32-bit constant in a.? Explain.

4. If the current value of the PC is $1FFFF000_{16}$, can you use a single branch instruction to get to the PC address as shown by the 32-bit constant in a.?

3.1 Answer a

```
lui     $t1, 0x2001
ori     $t1, $t1, 0x4924
```