RecursiveCalls

April 25, 2023

1 Recursive function calls MIPS assembly

Compile this piece of C code:

```
int fib(int n)
{
    if(n < 0)
    {
        cout << "n should be greater than -1" << endl;
        return -1;
    }
    if(n == 0) return 0;
    if(n < 3) return 1;
    return fib(n-1)+fib(n-2);
}

int main() {
    cout << "Fibonacci of 10 is " << fib(10) << endl;
    return 0;
}
```

First of all, I've transfered the code to be a little bit easier:

```
int fib(int n)
{
    if(n < 0)
    {
        cout << "n should be greater than -1" << endl;
        return -1;
    }
    if(n == 0) return 0;
    if(n < 3) return 1;

    int f = fib(n - 1);
    int s = fib(n - 2);
    int result = f + s;

    return result;
}
```

```
int main() {
    cout << "Fibonacci of 10 is " << fib(10) << endl;
    return 0;
}
```

Assumptions:

- All registers' values are zero by the start of the program.

```
    .text
main:
addi $t7, $zero, -1
move $t2, $t7    # n to $t2

# Call function to get fibonnacci #n
move $a0, $t2
move $v0, $t2
jal fib          # call fib (n)
move $t3,$v0     # result is in $t3

# Output message and n
la $a0,result
li $v0,4
syscall

move $a0,$t3
li $v0,1
syscall

la $a0,endl
li $v0,4
syscall

# End program
li $v0,10
syscall

fib:
# Compute and return fibonacci number
slt $s0, $a0, $zero
beq $s0, 1, neg_one
beqz $a0, zero    #if n=0 return 0
beq $a0, 1, one   #if n=1 return 1
beq $a0, 2, one   #if n=2 return 1

#Calling fib(n-1)
sub $sp,$sp,4   #storing return address on stack
```

```
sw $ra,0($sp)

sub $a0,$a0,1   #n-1
jal fib     #fib(n-1)
add $a0,$a0,1

lw $ra,0($sp)   #restoring return address from stack
add $sp,$sp,4


sub $sp,$sp,4   #Push return value to stack
sw $v0,0($sp)
#Calling fib(n-2)
sub $sp,$sp,4   #storing return address on stack
sw $ra,0($sp)

sub $a0,$a0,2   #n-2
jal fib     #fib(n-2)
add $a0,$a0,2

lw $ra,0($sp)   #restoring return address from stack
add $sp,$sp,4
#---------------
lw $s7,0($sp)   #Pop return value from stack
add $sp,$sp,4

add $v0,$v0,$s7 # f(n - 2)+fib(n-1)
jr $ra # decrement/next in stack

zero:
li $v0,0
jr $ra
one:
li $v0,1
jr $ra
neg_one:
la $a0, less_than_zero
li $v0,4
syscall
li $v0, -1
jr $ra

.data
less_than_zero: .asciiz "n should be greater than -1\n"
result: .asciiz "Fibonacci of 10 is "
endl: .asciiz "\n"
```