Fibonacci (attempted stack pointer movement)

| Assembly | Instruction (in Binary) | Instruction (in Hex) |
| --- | --- | --- |
| main: addi $a0, $zero 5 #set n=5 | 000011 00000 00100 0000000000000101 | 0c040005 |
| jal fib #function call | 000111 00000100000000000000000010 | 1c100002 |
| fib: addi $sp, $sp, -12 #make room on stack | 000011 11101 11101 1111111111110100 | 0fbdfff4 |
| sw $ra, 8($sp) #push $ra | 000000 00000 01000 0001000000100000 | 0bbf0008 |
| sw $s0, 4($sp) #push $s0 | 000010 11101 10000 0000000000000100 | 0bb00004 |
| sw $a0, 0($sp) #push $a0 (N) | 000010 11101 00100 0000000000000000 | 0ba40000 |
| beq $a0, $0, next #if n>0, test if n=1 | 000100 00100 00000 0000000000000010 | 10800002 |
| j test2 | 000110 00000100000000000000001010 | 1810000a |
| next: add $v0, $0, $0 #else fib(0) = 0 | 000000 00000 00000 00010 00000 100000 | 00001020 |
| j rtn | 000110 00000100000000000000010101 | 18100015 |
| test2: addi $t0, $0, 1 #if n=1, rtn 1 | 000011 00000 01000 0000000000000001 | 0c080001 |
| beq $t0, $a0, cont | 000100 01000 00100 0000000000000001 | 11040001 |
| j gen #else if n!=1 and n!=0 | 000110 00000100000000000000001111 | 1810000f |
| cont: add $v0, $0, $t0 #v0 = t0 = 1, | 000000 00000010000001000000100000 | 00081020 |
| j rtn #rtn 1 from v0 | 000110 00000100000000000000010101 | 18100015 |
| addi $a0, $a0, -1 #set n = n-1 | 000011 00100001001111111111111111 | 0c84ffff |
| jal fib #call fib(n-1) | 000111 00000100000000000000000010 | 1c100002 |
| add $s0, $v0, $0 #store fib(n-1) | 000000 00010 00000 10000 00000 100000 | 00408020 |
| sub $a0, $a0, 1 #n=n-2 | 000011 00100001001111111111111111 | 0c84ffff |
| jal fib #call fib(n-2) | 000111 00000100000000000000000010 | 1c100002 |
| add $v0, $v0, $s0 #fib(n-1) + fib(n-2) | 000000 00010 10000 00010 00000 100000 | 00501020 |
| rtn: lw $a0, 0($sp) #pop $a0 | 000001 11101001000000000000000000 | 07a40000 |
| lw $s0, 4($sp) #pop $s0 | 000001 11101100000000000000000100 | 07b00004 |
| lw $ra, 8($sp) #pop $ra | 000001 11101111110000000000001000 | 07bf0008 |
| addi $sp, $sp, 12 #restore sp | 000011 11101111010000000000001100 | 0fbd000c |
| jr $ra #return | 001000 11111111110000000000000000 | 23ff0000 |

Leaf (attempted stack pointer movement)

| Assembly | Instruction (In Binary) | Instruction (in Hex) |
| --- | --- | --- |
| main: addi $a0, $zero, 4 #set $a0 to 4 | 000011 00000 00100 0000000000000100 | 0c040004 |
| addi $a1, $zero, 4 #set $a1 to 4 | 000011 00000 00101 0000000000000100 | 0c050004 |
| addi $a2, $zero, 2 #set $a2 to 2 | 000011 00000 00110 0000000000000010 | 0C060002 |
| addi $a3, $zero, 5 #set $a3 to 5 | 000011 00000 00111 0000000000000101 | 0C070005 |
| jal leaf #leaf function call | 000111 00000100000000000000000101 | 1c100005 |
| leaf: Addi $sp, $sp, -4 #adjust stack pointer | 000011 11101 11101 1111111111111100 | 0fbdfffc |
| sw, $s0, 0($sp) #save address of $sp | 000010 11101 10000 0000000000000000 | 0bb00000 |
| add , $t0, $a0, $a1 #t0 = a0+a1 = 8 | 000000 00100 00101 01000 00000 100000 | 00854020 |
| add, $t1, $a2, $a3 #t1 = a2+a3 = 7 | 000000 00110 00111 01001 00000 100000 | 00c74820 |
| sub, $s0, $t0, $t1 #s0 = t0-t1 = 1 | 000000 01000 01001 10000 00000 100010 | 01098022 |
| add, $v0, $s0, $zero #v0 = s0 = 1 | 000000 10000 00000 00010 00000 100000 | 02001020 |
| lw, $s0, 0($sp) #reload stack pointer | 000001 11101 10000 0000000000000000 | 07B00000 |
| addi, $sp, $sp, 4 #restore stack pointer | 000011 11101 11101 0000000000000100 | 0FBD0004 |
| jr $ra #return | 001000 11111 11111 0000000000000000 | 23FF0000 |

Stack Pointer’

Fibonacci (Not Stackpointer)

| Assembly | Instruction (In Binary) | Instruction (in Hex) |
| --- | --- | --- |
| Main: addi $t1, $zero, 1 | 000011 00000 01001 0000000000000001 | 0c090001 |
| addi $s1, $zero, 1 | 000011 00000 10001 0000000000000001 | 0c110001 |
| addi $t2, $zero, 2 | 000011 00000 01010 0000000000000010 | 0C0A0002 |
| addi $t3, $zero, 9 | 000011 00000 01011  0000000000001001 | 0C0B0009 |
| Loop: beq $t2, $t3, finish | 000100 01011 01010 0000000000000110 | 116A0006 |
| add $s2, $t1, $s1 | 000000 01001 10001 10010 00000 100000 | 01319020 |
| add $t1, $zero, $s1 | 000000 10001 00000 01001 00000 100000 | 02204820 |
| add $s1, $zero, $s2 | 000000 00000 10010 10001 00000 100000 | 00128820 |
| addi $t2, $t2, 1 | 000011 01010 01010 0000000000000000 | 0d4a0001 |
| j loop | 000110 00000100000000000000000100 | 18100004 |
| finish : sw $s1, 0($zero) | 000010 00000 10001 0000000000000000 | 08110000 |
| lw $t1, 0($zero) | 000001 00000 01001  0000000000000000 | 04090000 |

Leaf.exe (No stack pointer)

| Assembly | Instruction (In Binary) | Instruction (in Hex) |
| --- | --- | --- |
| main: addi $a0, $zero, 4 #set $a0 to 4 | 000011 00000 00100 0000000000000100 | 0c040004 |
| addi $a1, $zero, 4 #set $a1 to 4 | 000011 00000 00101 0000000000000100 | 0c050004 |
| addi $a2, $zero, 2 #set $a2 to 2 | 000011 00000 00110 0000000000000010 | 0C060002 |
| addi $a3, $zero, 5 #set $a3 to 5 | 000011 00000 00111 0000000000000101 | 0C070005 |
| add , $t0, $a0, $a1 #t0 = a0+a1 = 8 | 000000 00100 00101 01000 00000 100000 | 00854020 |
| add, $t1, $a2, $a3 #t1 = a2+a3 = 7 | 000000 00110 00111 01001 00000 100000 | 00c74820 |
| sub, $s0, $t0, $t1 #s0 = t0-t1 = 1 | 000000 01000 01001 10000 00000 100010 | 01098022 |
| add, $v0, $s0, $zero #v0 = s0 = 1 | 000000 10000 00000 00010 00000 100000 | 02001020 |