

CSE 30321 – Lecture 07-08 – In Class Example Handout

Part H: Recursive Function Calls

```

int fact(int n) {
    if (n<1)
        return(1);
    else
        return(n*fact(n-1));
}

```

Part A:

Let's consider how we might use the stack to support these nested calls. We'll also make use of the frame pointer (\$fp).

Code Section #	Address	Label	MIPS Instruction	Comments
1	0	Fact:	subi \$sp, \$sp, 12	Make room for 3 pieces of data on the stack; \$fp, \$sp, and 1 local argument
	4		sw 8(\$sp), \$ra	If \$sp = 88, M(88 + 8) ← value of \$ra
	8		sw 4(\$sp), \$fp	If \$sp = 88, M(88 + 4) ← value of \$fp
	12		subi \$fp, \$fp, 12	Update the frame pointer
2	16		bgtz \$a0, L2	If N > 0 (i.e. not < 1) we're not done → we assume N is in \$a0
4	20		addi \$v0, \$0, 1	We eventually finish and want to return 1, therefore put 1 in return register
	24		j L1	Jump to return code
3	28	L2:	sw \$a0, 0(\$fp)	Save argument N to stack (we'll need it when we return)
	32		subi \$a0, \$a0, 1	Decrement N (N = N - 1), put result in \$a0
	36		jal Fact	Call Factorial() again
6	40		lw \$t0, 0(\$f0)	Load N (saved at *** to stack)
	44		mult \$v0, \$v0, \$t0	Store result in \$v0
5	48	L1:	lw \$ra, 8(\$sp)	Restore return address
	52		lw \$fp, 4(\$sp)	Restore frame pointer
	56		addi \$sp, \$sp, 12	Pop stack
	60		jr \$ra	Return from factorial

Part B: PLACE COMPLETED FACTORIAL UP ON THE SCREEN.

Let's walk through this code (and memory) assuming that we call the factorial function with the number 2. For your reference, I've included tables below that you can use to keep track of both instruction execution *and* the contents of memory. We'll assume that \$sp is initially 112 and that \$fp is 124.

Code Trace:

1 st Call to Factorial		2 nd Call to Factorial		3 rd Call to Factorial	
Addr	What Happens	Addr	What Happens	Addr	What Happens
0	\$sp = \$sp-12; \$sp ← 100	0	\$sp = \$sp-12; \$sp ← 88	0	\$sp = \$sp-12; \$sp ← 76
4	M(100+8) = M(108) ← \$ra	4	M(96) ← \$ra (\$ra=40)	4	M(84) ← \$ra (\$ra=40)
8	M(100+4) = M(104) ← \$fp	8	M(92) ← \$fp (\$fp=112)	8	M(80) ← \$fp (\$fp=100)
12	\$fp = \$fp-12; \$fp ← 112	12	\$fp = \$fp-12; \$fp ← 100	12	\$fp = \$fp-12; \$fp ← 88
16	2 is greater than 0	16	1 is greater than 0	16	0 is NOT greater than 0
28	M(\$fp / 112) ← N (store #)	28	M(\$fp / 100) ← N (store #)		(start to return)
32	N = N-1 (new arg = 1)	32	N = N-1 (new arg = 0)		
36	jal Fact (\$ra = 40 ₁₀)	36	jal Fact (\$ra = 40 ₁₀)		

Return from 3 rd Call		Return from 2 nd Call		Return form 1 st Call	
Addr	What Happens	Addr	What Happens	Addr	What Happens
24	j L1	40	lw \$t0, 0(\$fp); \$t0 ← M(100); \$t0 ← 1	40	lw \$t0, 0(\$fp); \$t0 ← M(112); \$t0 ← 2
48	\$ra ← M(\$sp+8) ← M(84) \$ra ← 40	44	\$v0 ← 1x1 \$v0 = return address reg.	44	\$v0 ← 1x2 \$v0 ← \$v0 x \$t0
52	\$fp ← M(\$sp+4) ← M(80) \$fp ← 100	48	\$ra ← M(\$sp+8) ← M(96) \$ra ← 40	48	\$ra ← M(\$sp+8) ← M(108) \$ra ← factorial caller RA
56	\$sp = 76+12; \$sp ← 88 (pop stack)	52	\$fp ← M(\$sp+4) ← M(92) \$fp ← 112	52	\$fp ← M(\$sp+4) ← M(104) \$fp ← factorial caller FP
60	jr \$ra implies that PC ← 40	56	\$sp ← 88 + 12 = 100	56	\$sp ← 100 + 12 = 112
		60	jr \$ra makes: PC ← 40	60	jr \$ra (PC + 4 of fact caller)

Memory Contents: (Assume main() calls function which calls factorial.)

Memory Address	Before 1 st Fact Call	During 1 st Fact Call	During 2 nd Fact Call	During 3 rd Fact Call	Return from 3 rd	Return from 2 nd	Return from 1 st
76				Current \$sp			
80				Saved \$fp from prior call (100)			
84				Saved \$ra of fact (40)			
88			Current \$sp	Current \$fp N never stored	\$sp 3 rd fact call out		
92			Saved \$fp from prior call (112)				
96			Saved \$ra of fact (40)				
100		Current \$sp	Current \$fp N = 1			\$sp 2 nd fact call out	
104		Saved \$fp of function calling fact (124)					
108		Saved \$ra of function calling fact					
112	Current \$sp	Current \$fp N = 2					\$sp 1 st fact call out
116	Saved \$fp of main						
120	Saved \$ra of main						
124	Current \$fp						

