

Last Name:

First Name:

# Computer Science C.Sc. 342

**GRADED!**

Take Home TEST No.2

*CSc or CPE*

*Submit by 11:00 PM, April 24, 2021*

1. Please hand write and sign statements affirming that you will not cheat here and in your submission:  
*"I will neither give nor receive unauthorized assistance on this TEST. I will use only one computing device to perform this TEST, I will not use cell while performing this TEST".*

2. Students should allocate no more than total 12 hours to prepare report for this TEST.

Please write

Start Time and date:

End TIME and date:

3. You can use any resources.

4. You are not allowed to communicate with fellow students or other professionals.

5. THIS IS NOT A TEAM TEST!!

You must submit:

1. Report must have OBJECTIVE section, and Table of Content with links to corresponding sections.
2. Short Video presentation, no more than 2 min.
3. Source code and project files you have used. You must include README file with instructions on how to run your examples.

The report file and video file names should be:

**YOUR\_LAST\_NAME\_TAKE\_HOME\_TEST\_2,**

**LAST\_NAME\_VIDEO\_TAKE\_HOME\_TEST\_2**

**LAST\_NAME\_SOURCE\_README\_\_TAKE\_HOME\_TEST\_2.ZIP**

Last Name:

First Name:

### ***Objective:***

The objective of this take home test is for students to

1. Run and debug a recursive function call on four different platforms: x86 Intel on Microsoft's Visual Studio, Raspberry Pi ARM processor 32 bit, MIPS on MARS Simulator, and on a 64-bit Intel processor running Linux. Display and explain all frames on stack.
2. Measure and plot the time it takes to compute Factorial (N), for N= 10, 100, 1000, 10,000.
3. **Required part :** Repeat tutorial example 1 and 2 to compute GCD(a,b) using recursive version of EUCLEDEAN algorithm for two integers  $a > 0$ ,  $b > 0$ . To refresh GCD(a,b) computation please refer to last 3 pages of this assignment.
4. **What to Submit:** report, working project files and how to use, 2 min video presentation.

**Tutorial Example of a recursive procedure** that calculates the factorial of a number and its code in both C and MIPS can be found in the textbook and is shown below.

### ***Create and explain Stack Frames for the recursive function call factorial(5)***

```
int factorial (int N)
{
    if (N==1)
        return 1;
    return (N*factorial(N-1));
}
void main()
{
    int N_fact=factorial(5);
}
```

1. **Compile and run this program in Debug mode in .NET environment.**

**For each call level display Frame on stack and write down the address on stack and value of**

- **Argument at current level**
- **local variable ( if any) at current level**
- **return address at current level**
- **EIP**
- **EBP**
- **ESP**

**You may use arrow to point a specific location on stack frame.**

**At the end of calls you should display 5 frames on the stack as shown in FIGURE 1.**

Last Name:

First Name:

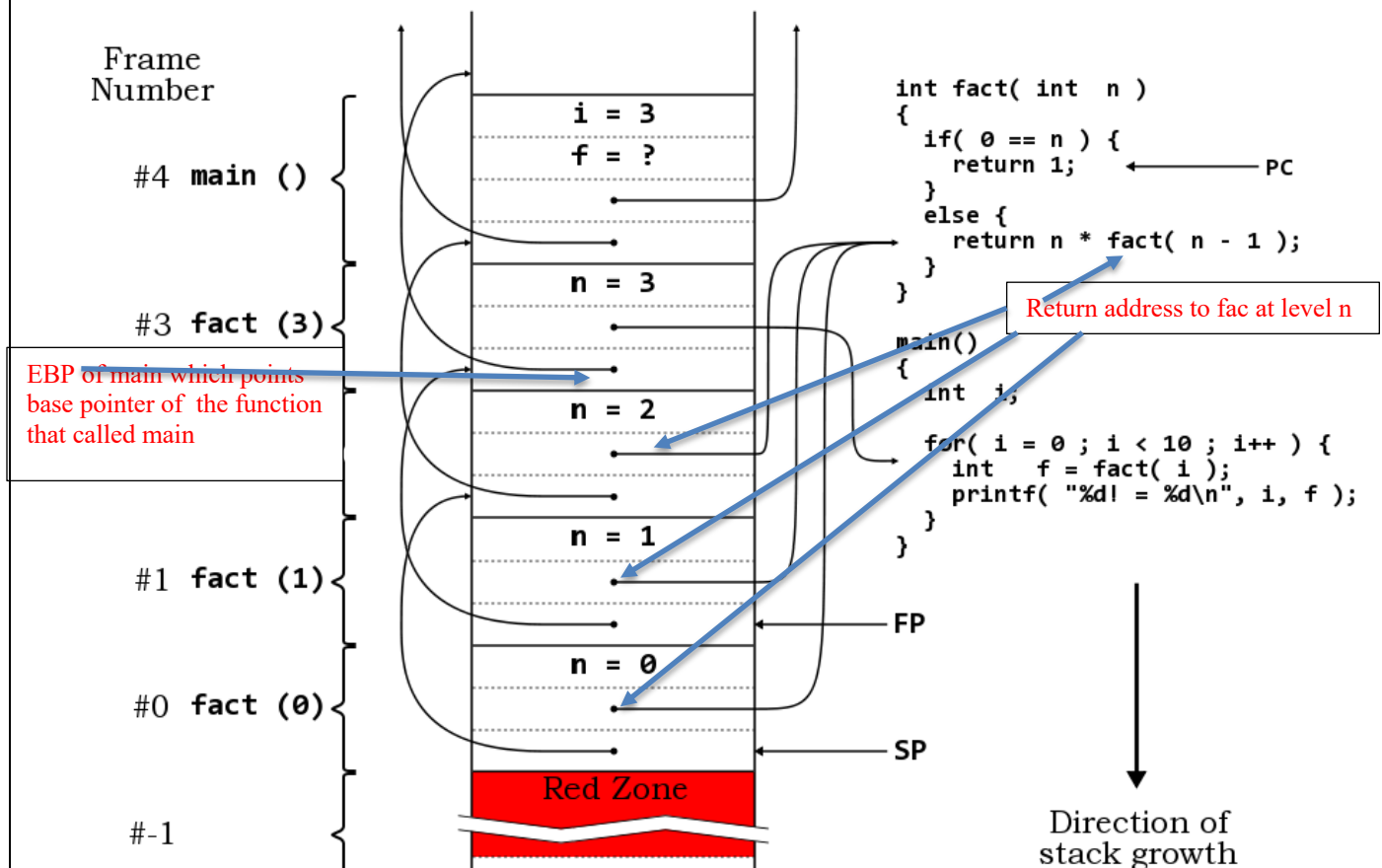


FIGURE 1. All arrows have to show labels to addresses on stack and corresponding values.

Please explain the return process – specify instructions and arguments used at each nested level when returning.

- (Optional) Create a lean version of the factorial() function. Instead of using CALL instruction ( generated by compiler), create function call using similar to JAL instruction in MIPS - save the return address and then jump to function. Do not push and pop unnecessary information on stack ( such as registers ebx, ecx, etc.) on stack.
- Please repeat Section 1 using MIPS instructions and run the program on a simulator MARS. You can use example described in the section on nested procedure calls in the textbook.
- Please repeat Section 1 using GCC, GDB in LINUX environment, and run the program in command mode using GDB. You can use example described in the section on nested procedure calls in the textbook.

Last Name:

First Name:

## Sample screenshots for X86, MS Visual Studio in Debug mode

1: int factorial(int N){

```
004013C0 55      push    ebp
004013C1 8B EC   mov     ebp,esp
004013C3 81 EC C0 00 00 00 sub     esp,0C0h
004013C9 53      push    ebx
004013CA 56      push    esi
004013CB 57      push    edi
004013CD 8D 40 FF FF lea     edi,[ebp-0C0h]
004013D2 B9 30 00 00 00 mov     ecx,30h
004013D7 B8 CC CC CC CC mov     eax,0CCCCCCCCh
004013DC F3 AB   rep stos dword ptr es:[edi]

2: if (N == 1) return 1;
004013DE 83 7D 08 01 cmp     dword ptr [N],1
004013E2 75 07   jne     factorial+2Bh (004013EBh)
004013E4 B8 01 00 00 00 mov     eax,1
004013E9 EB 13   jmp     factorial+3Eh (004013FEh)

3: return N*factorial(N - 1);
004013EB 8B 45 08 mov     eax,dword ptr [N]
004013EE 83 E8 01 sub     eax,1
004013F1 50      push    eax
004013F2 E4 FD FF FF call    factorial (004013DBh)
004013F7 83 C4 04 add     esp,4
004013FA 0F AF 45 08 imul    eax,dword ptr [N]

4: }
004013FE 5F      pop     edi
4: }
004013FF 5E      pop     esi
00401400 5B      pop     ebx
```

EAX = CCCCCCCC  
EBX = 7EFD0000  
ECX = 00000000  
EDX = 00000001  
ESI = 00000000  
EDI = 0015FAC4  
EIP = 004013DE  
ESP = 0015F9F8  
EBP = 0015FAC4  
EFL = 00000200

0x0015facc = 00000005

0x0015FA6C cc cc cc ii  
0x0015FA70 cc cc cc ii  
0x0015FA74 cc cc cc ii  
0x0015FA78 cc cc cc ii  
0x0015FA7C cc cc cc ii  
0x0015FA80 cc cc cc ii  
0x0015FA84 cc cc cc ii  
0x0015FA88 cc cc cc ii  
0x0015FA8C cc cc cc ii  
0x0015FA90 cc cc cc ii  
0x0015FA94 cc cc cc ii  
0x0015FA98 cc cc cc ii  
0x0015FA9C cc cc cc ii  
0x0015FAA0 cc cc cc ii  
0x0015FAA4 cc cc cc ii  
0x0015FAA8 cc cc cc ii  
0x0015FAAC cc cc cc ii  
0x0015FAB0 cc cc cc ii  
0x0015FAB4 cc cc cc ii  
0x0015FAB8 cc cc cc ii  
0x0015FABC cc cc cc ii  
0x0015FAC0 cc cc cc ii  
0x0015FAC4 a8 fb 15 00  
0x0015FAC8 55 14 40 00  
0x0015FACC 05 00 00 00  
0x0015FAD0 00 00 00 00  
0x0015FAD4 00 00 00 00

Memory 1

Address: 0x0015F61C

```
0x0015F61C 00 00 47 00 00 00 00 00 7f 00 00 00 14 f7 15 00 5a 3c be 77 8b 00 00 00 00 9c b1 c3 77
0x0015F638 14 f7 15 00 ce 57 c0 77 d3 3c be 77 cf c0 b9 75 40 04 00 00 00 00 00 00 00 47 00
0x0015F654 50 01 47 00 50 01 47 00 64 f7 15 00 b8 50 47 00 00 10 00 00 64 f7 15 00 01 00 00
0x0015F670 77 01 00 00 04 fe 69 0f 50 01 47 00 fe ff ff ff 00 88 47 00 cf ee 5b 0f 00 00 56 0f
0x0015F68C 01 00 00 00 7f 00 00 00 b4 f6 15 00 3c f8 15 00 00 00 00 00 e0 fd 7e cc cc cc cc
0x0015F6A8 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F6C4 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F6E0 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F6FC cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F718 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F734 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F750 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F76C 01 00 00 00 14 f9 15 00 00 00 00 00 e0 fd 7e cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F788 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F7A4 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F7C0 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F7DC cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F7F8 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F814 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F830 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F84C 00 00 00 00 e0 fd 7e cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F868 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F884 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F8A0 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F8BC cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F8D8 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F8FC cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F910 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F92C cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F948 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F964 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F980 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F99C cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F9B8 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F9D4 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015F9F0 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015FA0C cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015FA28 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015FA44 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015FA60 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015FA7C cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015FA98 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015FAB4 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
0x0015FAD0 cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc
```

factorial(1)

3c f8 15 00 f7 13 40 00

factorial(2)

14 f9 15 00 f7 13 40 00 02 00 00 00

factorial(3)

ec f9 15 00 f7 13 40 00 03 00 00 00 c4 fa 15 00 00 00 00 00 e0 fd 7e

factorial(4)

f7 13 40 00 04 00 00 00 a8 fb 15 00 00 00 00 00 e0 fd 7e

factorial(5)

a8 fb 15 00 55 14 40 00 05 00 00 00

Return address to fac  
0x004913F7

Saved EBP of  
main()

Return address during  
execution of factorial(5)

Argument during execution of  
factorial(5)

EBP of main 0x0015fba8  
Saved at ebp = 0x0015fac4 at fact(5) level

First Name:

Text Segment					Registers				
Bkpt	Address	Code	Basic	Source		Name	Number	Value	
<input type="checkbox"/>	0x00400000	0x24040005	addiu \$4,\$0,0x00000005	2: li \$a0, 5		\$zero	0	0x00000000	
<input type="checkbox"/>	0x00400004	0x0c100004	jal 0x00400010	3: jal factasm		\$at	1	0x00000000	
<input type="checkbox"/>	0x00400008	0x20500000	addi \$16,\$2,0x00000000	4: addi \$a0,\$v0,0		\$v0	2	0x00000000	
<input type="checkbox"/>	0x0040000c	0x0000000c	sycall	5: sycall	main	0x00400000			
<input type="checkbox"/>	0x00400010	0x23bdffff	addi \$29,\$29,0xffffffffff	8: addi \$sp,\$sp,-8	L1	0x00400010			
<input type="checkbox"/>	0x00400014	0x0bfef004	sw \$31,0x00000004(\$29)	9: sw \$ra, 4(\$sp)		\$a0	4	0x00000000	
<input type="checkbox"/>	0x00400018	0xafafef04	sw \$4,0x00000005(\$29)	10: sw \$a0, 0(\$sp)		\$a1	5	0x00000000	
<input type="checkbox"/>	0x0040001c	0x28800000	lwi \$5,\$4,0x00000001	11: lw \$t0, \$a0, 1		\$a2	6	0x00000000	
<input type="checkbox"/>	0x00400020	0x11000003	breq \$8,\$0,0x00000003	13: breq \$t0,\$zero,L1		\$a3	7	0x00000000	
<input type="checkbox"/>	0x00400024	0x20020001	addi \$2,\$0,0x00000001	15: addi \$v0,\$zero,1		\$t0	8	0x00000000	
<input type="checkbox"/>	0x00400028	0x23bd0008	addi \$29,\$29,0x00000008	16: addi \$sp,\$sp,8		\$t1	9	0x00000000	
<input type="checkbox"/>	0x0040002c	0x0000000e	jr \$31	17: jr \$ra		\$t2	10	0x00000000	
<input type="checkbox"/>	0x00400030	0x2084ffff	addi \$4,\$4,0xffffffffff	19: li: addi \$a0,\$a0,-1		\$t3	11	0x00000000	
<input type="checkbox"/>	0x00400034	0x0c100004	j al 0x00400010	20: jal factasm		\$t4	12	0x00000000	
<input type="checkbox"/>	0x00400038	0x8fafef04	lw \$4,0x00000000(\$29)	22: lw \$a0, 0(\$sp)		\$t5	13	0x00000000	
<input type="checkbox"/>	0x0040003c	0x0bfef004	lw \$31,0x00000004(\$29)	23: lw \$ra, 4(\$sp)		\$t6	14	0x00000000	
<input type="checkbox"/>	0x00400040	0x23bd0008	addi \$29,\$29,0x00000008	24: addi \$sp,\$sp,8		\$t7	15	0x00000000	
<input type="checkbox"/>	0x00400044	0x70b21002	muli \$2,\$4,\$2	25: mul \$v0,\$a0,\$v0		\$a0	16	0x00000000	
<input type="checkbox"/>	0x00400048	0x03be0008	jr \$31	26: jr \$ra		\$a1	17	0x00000000	
<input type="checkbox"/>						\$a2	18	0x00000000	
<input type="checkbox"/>						\$a3	19	0x00000000	
<input type="checkbox"/>						\$a4	20	0x00000000	
<input type="checkbox"/>						\$a5	21	0x00000000	
<input type="checkbox"/>						\$a6	22	0x00000000	
<input type="checkbox"/>						\$a7	23	0x00000000	
<input type="checkbox"/>						\$a8	24	0x00000000	
<input type="checkbox"/>						\$a9	25	0x00000000	
<input type="checkbox"/>						\$a0	26	0x00000000	
<input type="checkbox"/>						\$a1	27	0x00000000	
<input type="checkbox"/>						\$a2	28	0x00000000	
<input type="checkbox"/>						\$a3	29	0x00000000	
<input type="checkbox"/>						\$a4	30	0x00000000	
<input type="checkbox"/>						\$a5	31	0x00000000	
<input type="checkbox"/>						\$a6	32	0x00000000	
<input type="checkbox"/>						\$a7	33	0x00000000	
<input type="checkbox"/>						\$a8	34	0x00000000	
<input type="checkbox"/>						\$a9	35	0x00000000	
<input type="checkbox"/>						\$a0	36	0x00000000	
<input type="checkbox"/>						\$a1	37	0x00000000	
<input type="checkbox"/>						\$a2	38	0x00000000	
<input type="checkbox"/>						\$a3	39	0x00000000	
<input type="checkbox"/>						\$a4	40	0x00000000	
<input type="checkbox"/>						\$a5	41	0x00000000	
<input type="checkbox"/>						\$a6	42	0x00000000	
<input type="checkbox"/>						\$a7	43	0x00000000	
<input type="checkbox"/>						\$a8	44	0x00000000	
<input type="checkbox"/>						\$a9	45	0x00000000	
<input type="checkbox"/>						\$a0	46	0x00000000	
<input type="checkbox"/>						\$a1	47	0x00000000	



Last Name:

First Name:

## Sample screenshots for 64 bit Intel processor, GDB

```
=> 0x00000000004004f6 <+0>:      push    %rbp
0x00000000004004f7 <+1>:      mov     %rsp,%rbp
0x00000000004004fa <+4>:      sub     $0x10,%rsp
0x00000000004004fe <+8>:      mov     %edi,-0x4(%rbp)
0x0000000000400501 <+11>:     cmpl    $0x1,-0x4(%rbp)
0x0000000000400505 <+15>:     jne     0x40050e <factorial(int)+24>
0x0000000000400507 <+17>:     mov     $0x1,%eax
0x000000000040050c <+22>:     jmp     0x40051f <factorial(int)+41>
0x000000000040050e <+24>:     mov     -0x4(%rbp),%eax
0x0000000000400511 <+27>:     sub     $0x1,%eax
0x0000000000400514 <+30>:     mov     %eax,%edi
0x0000000000400516 <+32>:     callq   0x4004f6 <factorial(int)>
0x000000000040051b <+37>:     imul    -0x4(%rbp),%eax
0x000000000040051f <+41>:     leaveq  0x0000000000400520 <+42>:     retq
```

End of assembler dump.

(gdb) nexti 3

```
0x00000000004004fe      1      int factorial(int N){
```

```
1: x/i $pc
```

```
=> 0x4004fe <factorial(int)+8>: mov     %edi,-0x4(%rbp)
```

```
(gdb) printf "rbp:%x\nrsp:%x\n", $rbp, $rsp
```

```
rbp:ffffdde0
```

```
rsp:ffffddd0
```

```
(gdb) █
```

```
1: x/i $pc
=> 0x4004fe <factorial(int)+8>: mov     %edi,-0x4(%rbp)
(gdb) printf "rbp:%x\nrsp:%x\n", $rbp, $rsp
rbp:ffffdde0
rsp:ffffddd0
(gdb) nexti
2      if(N == 1) return 1;
1: x/i $pc
=> 0x400501 <factorial(int)+11>      cmpl    $0x1,-0x4(%rbp)
(gdb) x/12xw $rsp
0x7fffffffddd0: 0x00000000 0x00000000 0x00000000 0x00000001
0x7fffffffddde: 0xffffde00 0x00007fff 0x0040051b 0x00000000
0x7fffffffdd10: 0xffffde20 0x00007fff 0xffffde10 0x00000002
(gdb) p $rip
$15 = (void (*)(void)) 0x400501 <factorial(int)+11>
(gdb) █
```

Argument during factorial(1)

Return address during factorial(1)

Saved RBP of factorial(2)

END TUTORIAL EXAMPLE.

Review of GCD algorithm: