

**Module: R2: Intro to RISC-V Assembly**  
**Section: CALL Task: Programming Task**

## Bare Metal Assembly on Spike -

<https://github.com/ImNomanCR7/fa21-lab-starter/blob/main/R2%20-%20Intro%20to%20RISCV/Final%20Programming%20Task/test.s>

### Programming Task

---

#### 1. RISC-V Assembly Code:

##### ■ Code Snippet:

```
.globl factorial

.text
# start code here
main:
    la t0, num      # t0 = address of num
    lw a0, 0(t0)    # a0 = num
    jal factorial
    j exit

factorial:
    addi sp, sp, -8
    sw ra, 0(sp)
    sw a0, 4(sp)
    li t1, 1
    bgt a0, t1, else
    li a0, 1
    lw ra, 0(sp)
    addi sp, sp, 8
    jr ra

else:
    addi a0, a0, -1
    jal factorial
    lw t2, 4(sp)
    lw ra, 0(sp)
    addi sp, sp, 8
    mul a0, t2, a0
    jr ra

exit:
    add a1, a0, x0
    li a0, 1
```

```

    ecall
    li a0, 10
    ecall

# end code here

write_tohost:

li x1, 1

sw x1, tohost, t5

j write_tohost

.data

# start data section here
num: .word 8
result: .word 1
# end data section here

.align 12
.section ".tohost","aw",@progbits;
.align 4; .global tohost; tohost: .dword 0;
.align 4; .global fromhost; fromhost: .dword 0;

```

### ■ Linker Script:

```

#Author: Noman Rafiq
#Dated: July 3, 2024

```

#### SECTIONS

```

{
    . = 0x80000000;

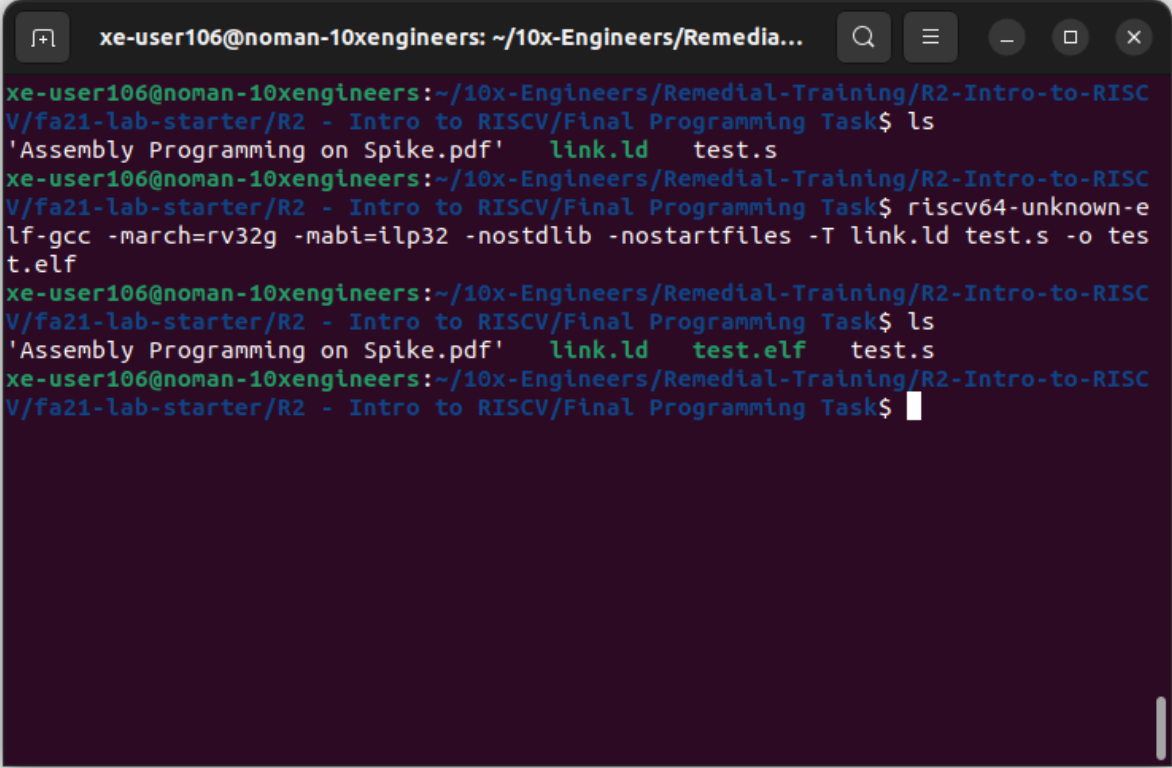
    .text : { *(.text) }
    . = 0x80001000;
    .data : { *(.data) }
    _end = .;

}

```

Run this command:

```
riscv64-unknown-elf-gcc -march=rv32g -mabi=ilp32 -nostdlib -nostartfiles  
-T link.ld test.s -o test.elf
```



```
xe-user106@noman-10xengineers: ~/10x-Engineers/Remedia...  
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R2-Intro-to-RISC  
V/fa21-lab-starter/R2 - Intro to RISC/Final Programming Task$ ls  
'Assembly Programming on Spike.pdf' link.ld test.s  
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R2-Intro-to-RISC  
V/fa21-lab-starter/R2 - Intro to RISC/Final Programming Task$ riscv64-unknown-e  
lf-gcc -march=rv32g -mabi=ilp32 -nostdlib -nostartfiles -T link.ld test.s -o tes  
t.elf  
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R2-Intro-to-RISC  
V/fa21-lab-starter/R2 - Intro to RISC/Final Programming Task$ ls  
'Assembly Programming on Spike.pdf' link.ld test.elf test.s  
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R2-Intro-to-RISC  
V/fa21-lab-starter/R2 - Intro to RISC/Final Programming Task$
```

In order to check dis-assembly file, use this command:

```
riscv64-unknown-elf-objdump -D test.elf
```

In the screenshot below, the dis-assembly verifies that out **.text** section started from 0x80000000 and the **.data** section started from 0x80001000

```

xe-user106@noman-10xengineers: ~/10x-Engineers/Remedial-Training/R2-Intro-to-RISC-V/fa21-lab-starter/R2 - Intro to RISC-V/Final Programming Task
Disassembly of section .text:
00000000 <main>:
00000000: 00001297      auipc    t0,0x1
00000004: 00028293      mv      t0,t0
00000008: 0002a503      lw      a0,0(t0) # 80001000 <num>
0000000c: 000000ef      jal     80000014 <factorial>
00000010: 0440006f      j       80000054 <exit>

00000014 <factorial>:
00000014: ff810113      addi    sp,sp,-8
00000018: 00112023      sw      ra,0(sp)
0000001c: 00a12223      sw      a0,4(sp)
00000020: 00100313      li      t1,1
00000024: 00a34a63      blt     t1,a0,80000038 <else>
00000028: 00100513      li      a0,1
0000002c: 000120a3      lw      ra,0(sp)
00000030: 00010113      addi    sp,sp,8
00000034: 00008067      ret

00000038 <else>:
00000038: fff50513      addi    a0,a0,-1
0000003c: fd9ff0ef      jal     80000014 <factorial>
00000040: 00a12383      lw      t2,4(sp)
00000044: 000120a3      lw      ra,0(sp)
00000048: 00010113      addi    sp,sp,8
0000004c: 02a38533      mul     a0,t2,a0
00000050: 00008067      ret

00000054 <exit>:
00000054: 000505b3      add     a1,a0,zero
00000058: 00100513      li      a0,1
0000005c: 00000073      ecall
00000060: 00a00513      li      a0,10
00000064: 00000073      ecall

00000068 <write_tohost>:
00000068: 00100093      li      ra,1
0000006c: 00002f17      auipc    t5,0x2
00000070: f81f2a23      sw      ra,-108(t5) # 80002000 <_end>
00000074: ff5ff06f      j       80000068 <write_tohost>

Disassembly of section .data:
00001000 <num>:
00001000: 0008              .lnsn    2, 0x0008
...

00001004 <result>:
00001004: 0001              .lnsn    2, 0x0001
...

Disassembly of section .tohost:
80002000 <tohost>:
...

```

Now, we can manually check the register to verify the output. We can use the following command to go to the debug mode.

```
spike -d --log-commits --isa=rv32gc $(which pk) test.elf
```

```

xe-user106@noman-10xengineers: ~/10x-Engineers/Remedia...
12: 3376      .insn 2, 0x3376
14: 6932      .insn 2, 0x6932
16: 7032      .insn 2, 0x7032
18: 5f31      .insn 2, 0x5f31
1a: 326d      .insn 2, 0x326d
1c: 3070      .insn 2, 0x3070
1e: 615f 7032 5f31 .insn 6, 0x5f317032615f
24: 3266      .insn 2, 0x3266
26: 3270      .insn 2, 0x3270
28: 645f 7032 5f32 .insn 6, 0x5f327032645f
2e: 697a      .insn 2, 0x697a
30: 32727363   bgeu tp,t2,356 <main-0x7ffffcaa>
34: 3070      .insn 2, 0x3070
36: 7a5f 6669 6e65 .insn 6, 0x6e6566697a5f
3c: 32696563   bltu s2,t1,366 <main-0x7ffffc9a>
40: 3070      .insn 2, 0x3070
42: 7a5f 6d6d 6c75 .insn 6, 0x6c756d6d7a5f
48: 7031      .insn 2, 0x7031
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R2-Intro-to-RISC
V/fa21-lab-starter/R2xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Trai
ning/R2-Intro-to-RISC
V/fa21-lab-starter/R2 - Intro to RISC/Final Programming Task$ spike -d --log-co
mmits --isa=rv32gc $(which pk) test.elf
(spike)

```

Now, we can manually check the register to verify the output. We can use the following command to go to the debug mode.

```
rs
```

```

xe-user106@noman-10xengineers: ~/10x-Engineers/Remedia...
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
core 0: exception trap_instruction_access_fault, epc 0x00000000
core 0: tval 0x00000000
(spike) rs
(spike)

```

We'll add a breakpoint to verify the output of our program. The output for **8! = 40320**, hence the output should be 40320. In our dis-assembly file, we can see that the function is returning the **a0** (factorial) value at PC = 0x80000010, hence we'll add a break point to our execution until this **PC**.

```

80000000 <main>:
80000000:      00001297      auipc    t0,0x1
80000004:      00028293      mv      t0,t0
80000008:      0002a503      lw      a0,0(t0) # 80001000 <num>
8000000c:      008000ef      jal     80000014 <factorial>
80000010:      0440006f      j       80000054 <exit>

80000014 <factorial>:
80000014:      ff810113      addi    sp,sp,-8
80000018:      00112023      sw      ra,0(sp)
8000001c:      00a12223      sw      a0,4(sp)
80000020:      00100313      li      t1,1
80000024:      00a34a63      blt     t1,a0,80000038 <else>
80000028:      00100513      li      a0,1
8000002c:      00012083      lw      ra,0(sp)

```

Adding a breakpoint at 0x80000010.

```

26: 3270          .insn 2, 0x3270
28: 645f 7032 5f32 .insn 6, 0x5f327032645f
2e: 697a          .insn 2, 0x697a
30: 32727363      bgeu  tp,t2,356 <main-0x7ffffcaa>
34: 3070          .insn 2, 0x3070
36: 7a5f 6669 6e65 .insn 6, 0x6e6566697a5f
3c: 32696563      bltu  s2,t1,366 <main-0x7ffffc9a>
40: 3070          .insn 2, 0x3070
42: 7a5f 6d6d 6c75 .insn 6, 0x6c756d6d7a5f
48: 7031          .insn 2, 0x7031
4a: 0030          .insn 2, 0x0030
xe-user106@noman-10xengineers:~/10x-Engineers/Remedial-Training/R2-Intro-to-RISC-V/fa21-lab-starter/R2 - Intro to RISCV/Final Programming Task$ spike -d --mmio --isa=rv32gc $(which pk) test.elf
(spike) until pc 0 80000010

```

After that, we'll view the contents of the registers to check values, use the following command:

```
reg 0 a0
```

```

core 0: 3 0x8000004c (0x02a38533) x10 0x000002d0
core 0: 3 0x80000050 (0x00008067)
core 0: 3 0x80000040 (0x00412383) x7 0x00000007 mem 0xfffffffff4
core 0: 3 0x80000044 (0x00012083) x1 0x80000040 mem 0xfffffffff0
core 0: 3 0x80000048 (0x00810113) x2 0xfffffffff8
core 0: 3 0x8000004c (0x02a38533) x10 0x000013b0
core 0: 3 0x80000050 (0x00008067)
core 0: 3 0x80000040 (0x00412383) x7 0x00000008 mem 0xfffffffffc
core 0: 3 0x80000044 (0x00012083) x1 0x80000010 mem 0xfffffffff8
core 0: 3 0x80000048 (0x00810113) x2 0x00000000
core 0: 3 0x8000004c (0x02a38533) x10 0x00009d80
core 0: 3 0x80000050 (0x00008067)
(spike) reg 0 a0
0x00009d80
(spike)

```

The value at this location is 0x00009d80 which is equal to 40320 in Decimal. We can convert the value back to decimal to check for any wrong value. The value of the a0 in our case is 0x00009d80 which is equal to 40320 (decimal) i.e. factorial for number 8 (8! = 40320).

### 3. Conclusion:

We have successfully compiled and executed the assembly code with custom linker script using RISC-V GNU toolchain for 32-bit architecture, without any

warnings or errors.