NESTED (RECURSIVE) FUNCTION IN RISC-V ISA

Dr Sparsh Mittal

Nested functions

- For nested call, caller needs to save on the stack:
  - Its return address
  - Any arguments and temporaries needed after the call
- Restore from the stack after the call

Convert this to RISC-V

- C code:

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

  - Argument n is initially stored in x10
  - Final result should be stored in x10
  - “jr x1” is same as “jalr x0 x1 0”.

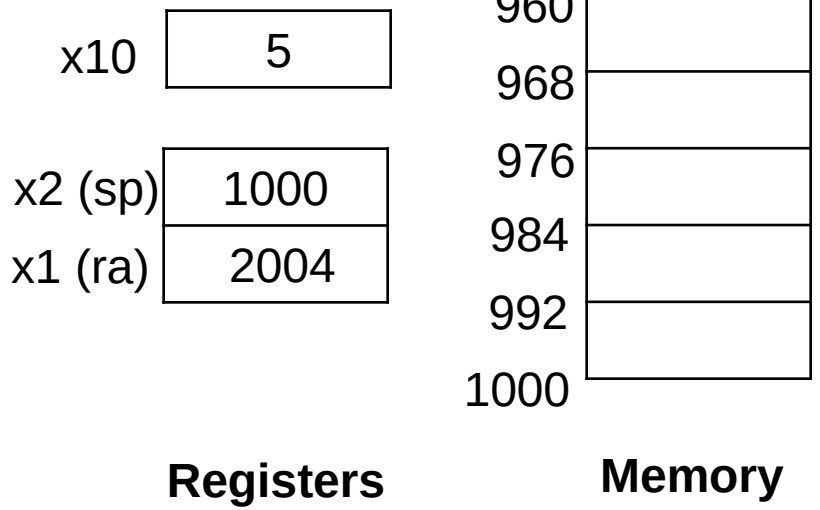
Example instruction	Instruction name	Meaning
jal x1,offset	Jump and link	Regs[x1]←PC+4; PC←PC + (offset<<1)
jalr x1,x2,offset	Jump and link register	Regs[x1]←PC+4; PC←Regs[x2]+offset

Example

```
j main
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            call fact(n-1)
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

Example

```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        call fact(n-1)
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```



Example

```
fact:
    addi sp,sp,-16
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1
    bge  x5,x0,L1
    addi x10,x0,1
    addi sp,sp,16
    jr x1
L1: addi x10,x10,-1
    jal  x1,fact
    addi x6,x10,0
    ld   x10,0(sp)
    ld   x1,8(sp)
    addi sp,sp,16
    mul  x10,x10,x6
    jr x1
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

Save return address and n on stack

x5 = n - 1

if n >= 0, go to L1

Else, set return value to 1

Pop stack, don't bother restoring values

Return

n = n - 1

call fact(n-1)

move result of fact(n - 1) to x6

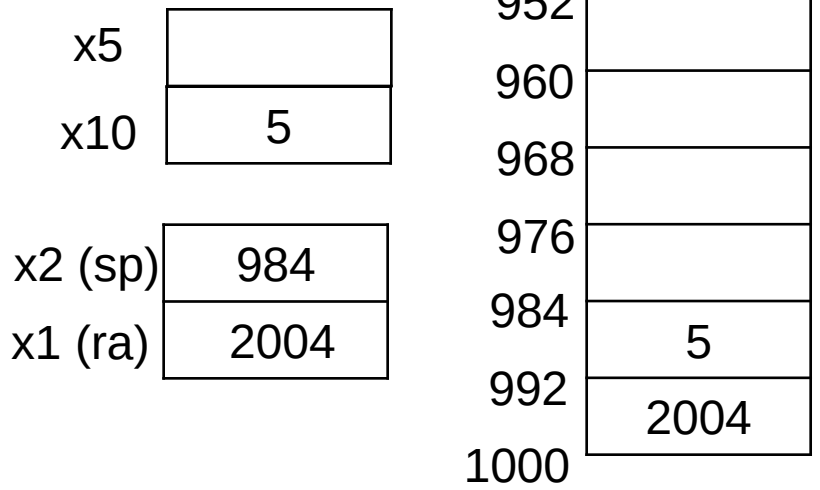
Restore caller's n

Restore caller's return address

Pop stack

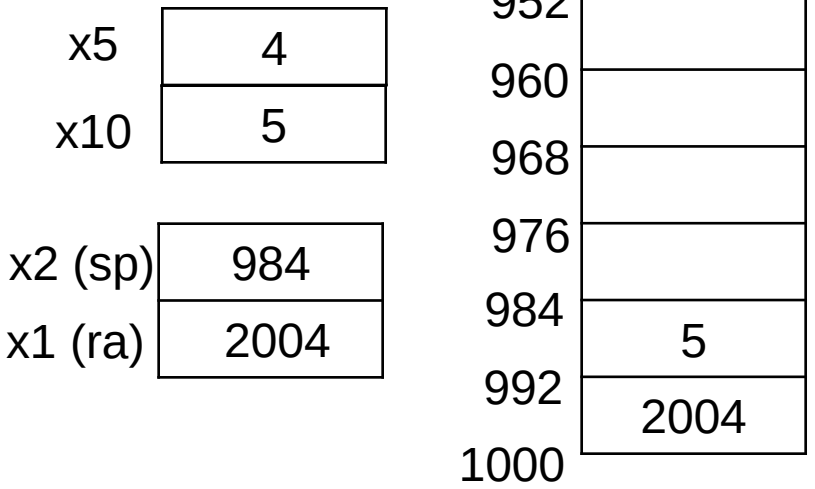
return n * fact(n-1)

return



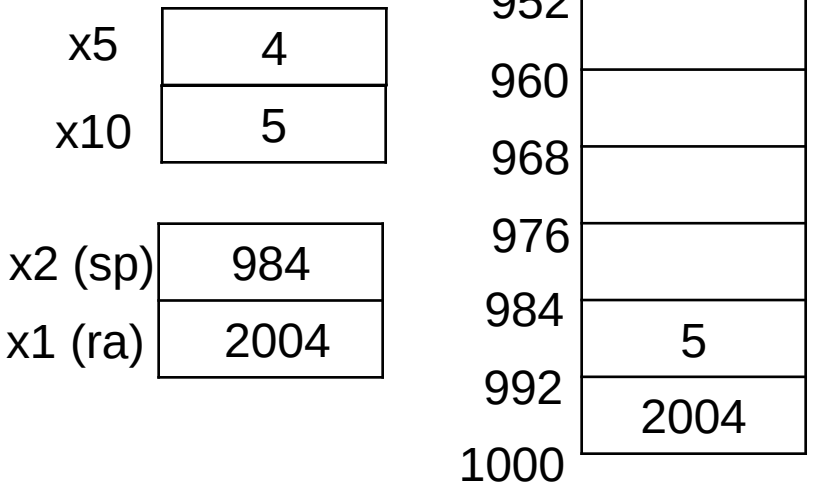
Example

```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            call fact(n-1)
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```



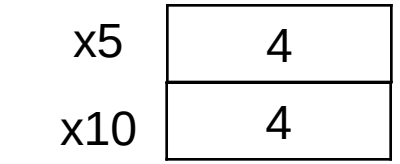
Example

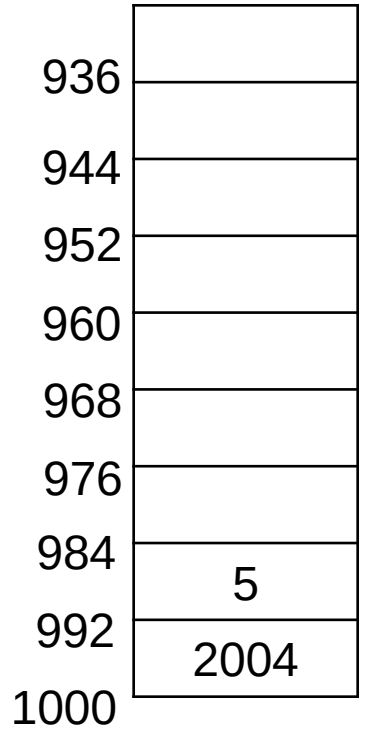
```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            call fact(n-1)
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```



Example

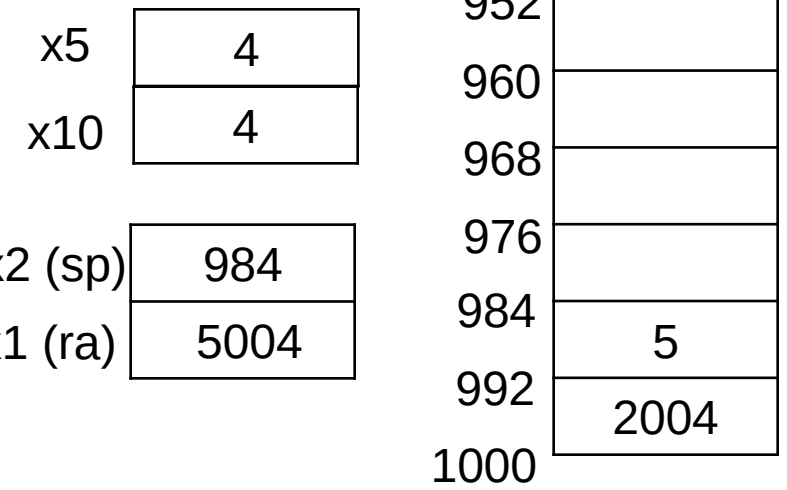
```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            call fact(n-1)
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```





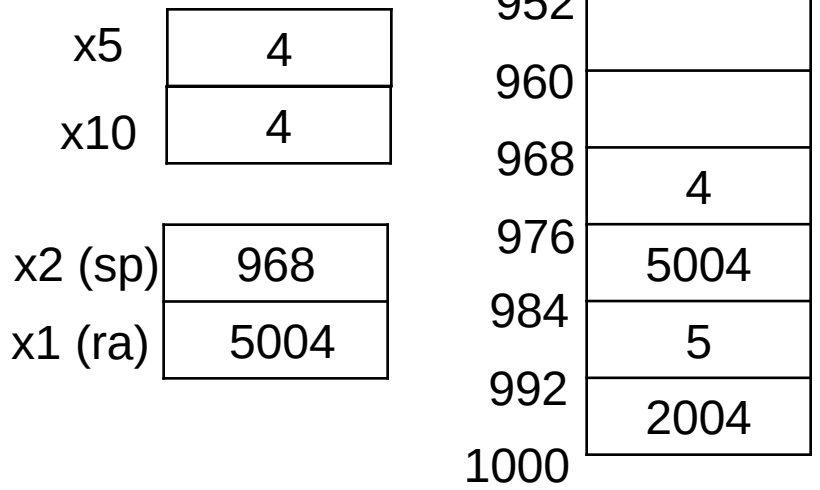
Example

```
fact:
    addi sp,sp,-16      <-- Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```



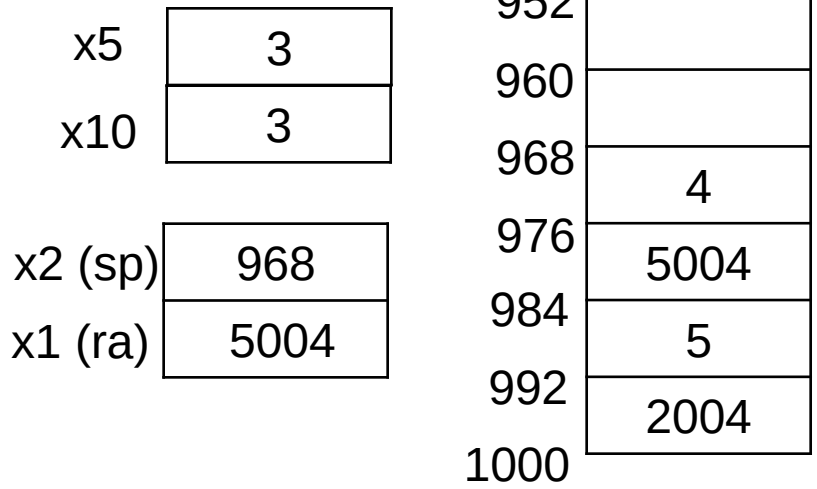
Example

```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            /*Address = 5000*/
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```



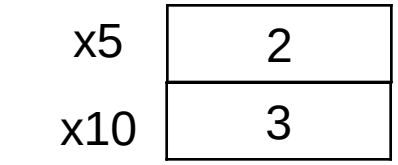
Example

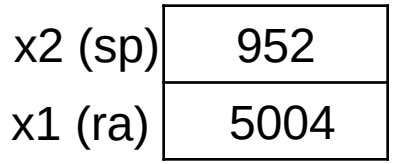
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

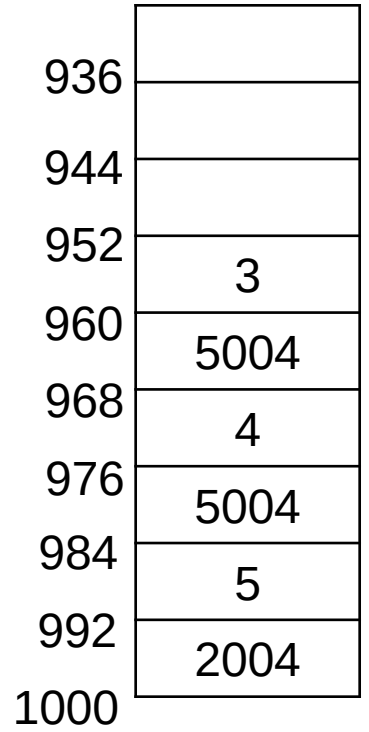


Example

```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            /*Address = 5000*/
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

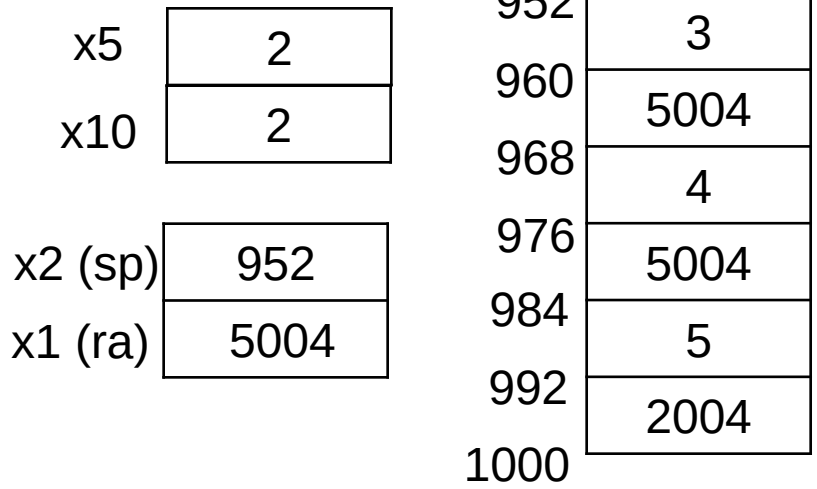






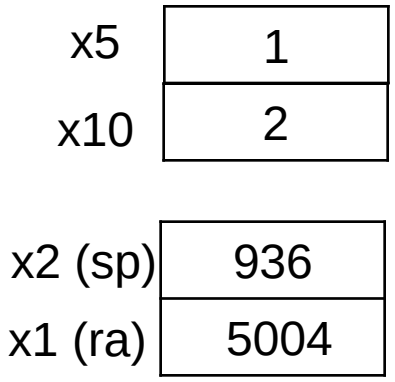
Example

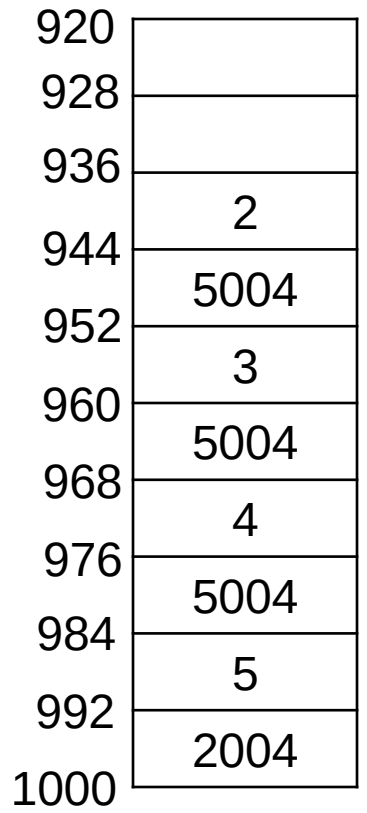
```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            /*Address = 5000*/
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```



Example

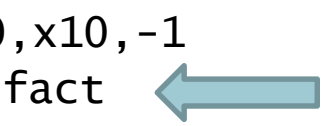
```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            /*Address = 5000*/
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

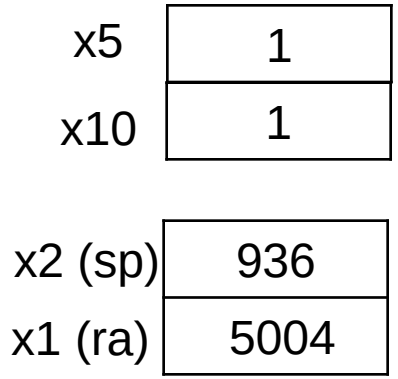


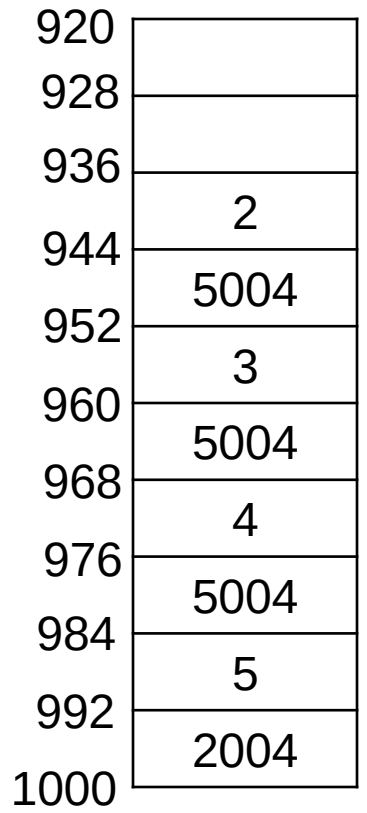


Example

```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

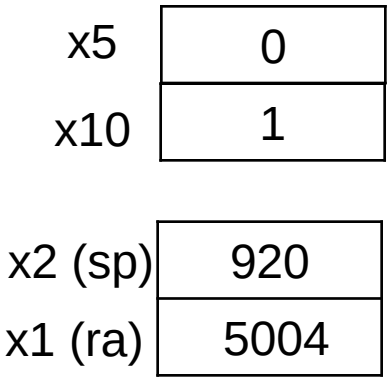


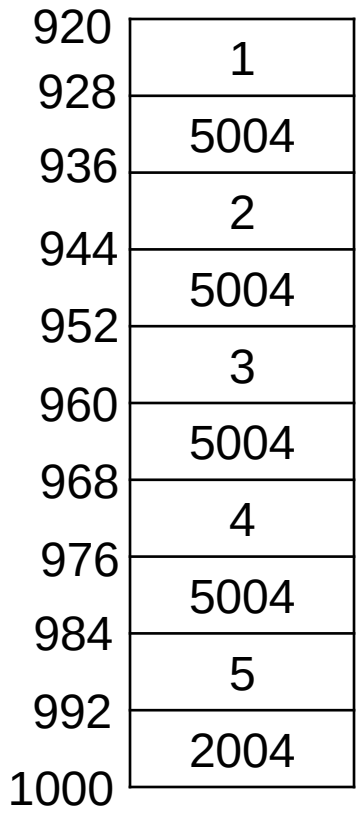




Example

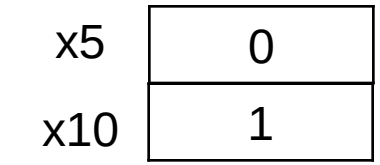
```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            /*Address = 5000*/
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

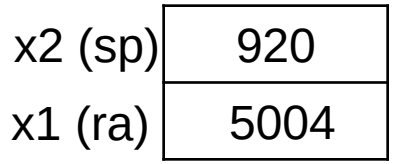


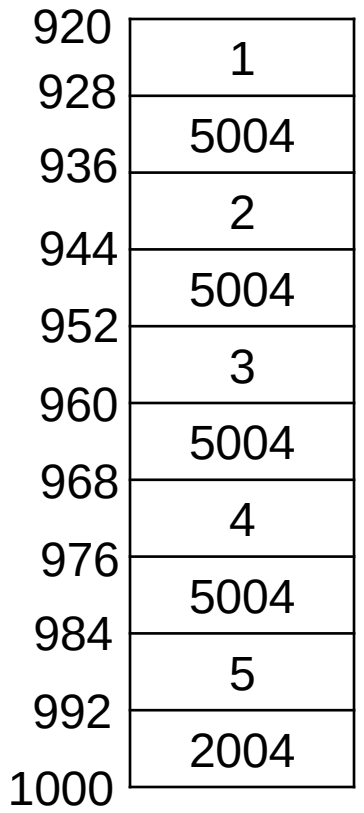


Example

```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            /*Address = 5000*/
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

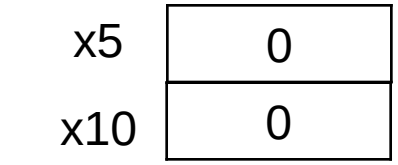


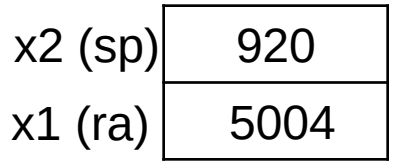


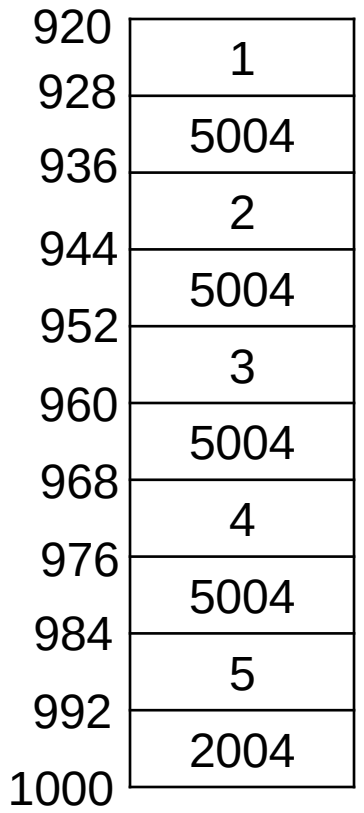


Example

```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            /*Address = 5000*/
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

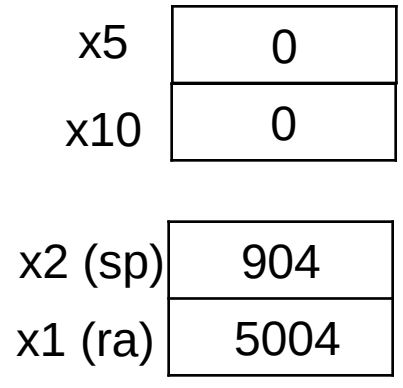


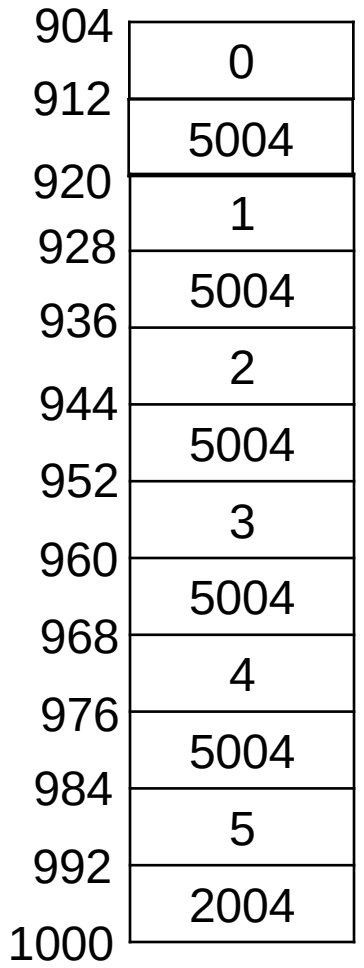




Example

```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            /*Address = 5000*/
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```





Example

```
fact:
    addi sp,sp,-16
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1
    bge  x5,x0,L1
    addi x10,x0,1
    addi sp,sp,16
    jr x1
L1: addi x10,x10,-1
    jal  x1,fact
    addi x6,x10,0
    ld   x10,0(sp)
    ld   x1,8(sp)
    addi sp,sp,16
    mul  x10,x10,x6
    jr x1
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

Save return address and n on stack

x5 = n - 1

if n >= 0, go to L1

Else, set return value to 1

Pop stack, don't bother restoring values

Return

n = n - 1

/*Address = 5000*/

move result of fact(n - 1) to x6

Restore caller's n

Restore caller's return address

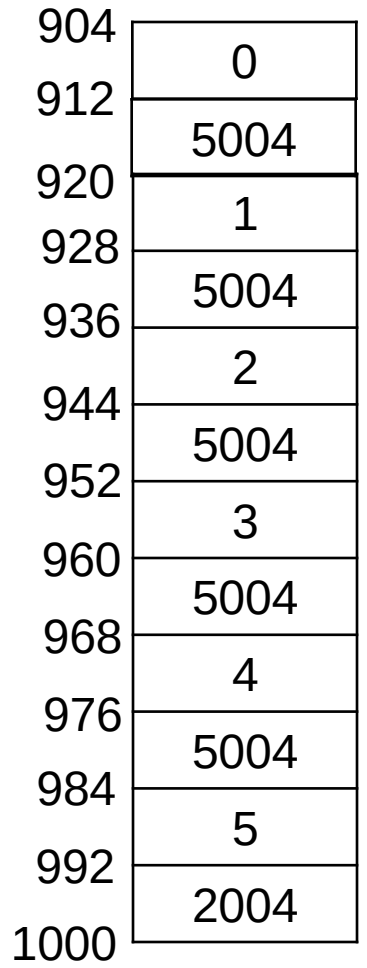
Pop stack

return n * fact(n-1)

return

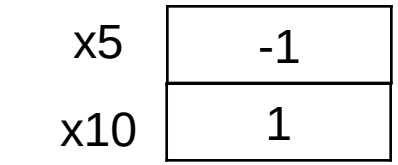
x5	-1
x10	0

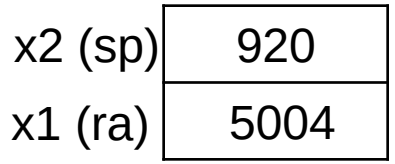
x2 (sp)	904
x1 (ra)	5004

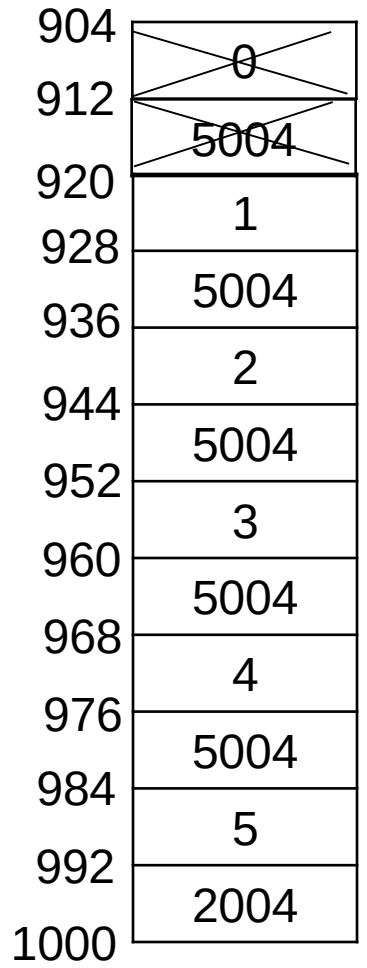


Example

```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

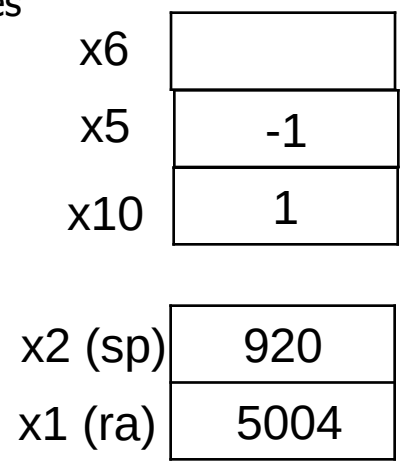


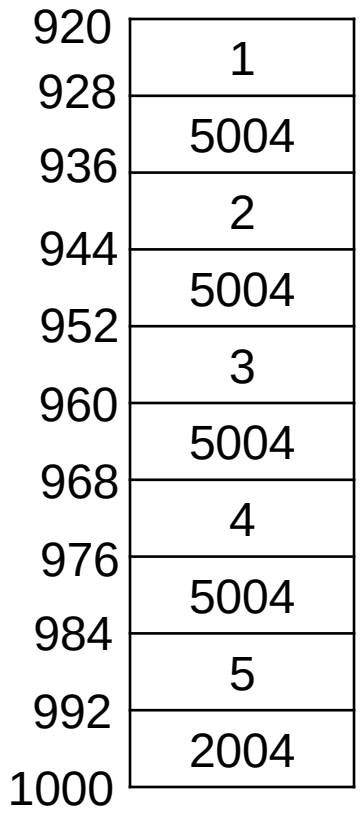




Example

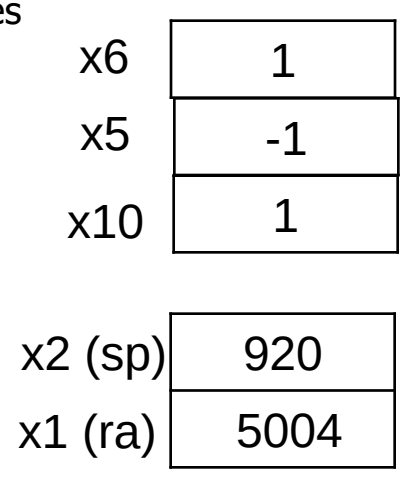
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

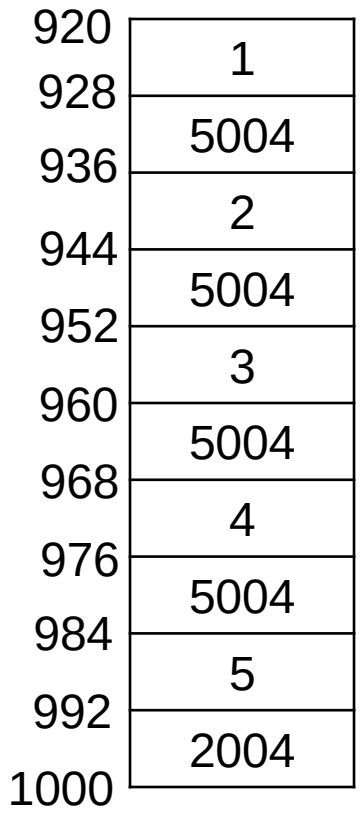




Example

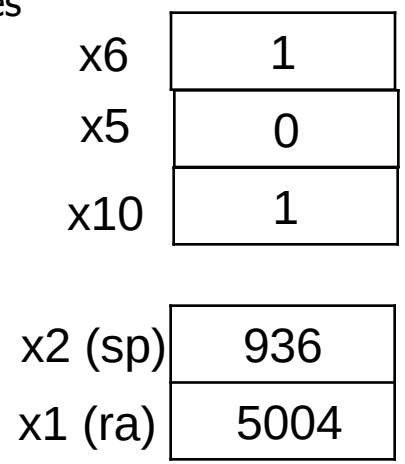
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

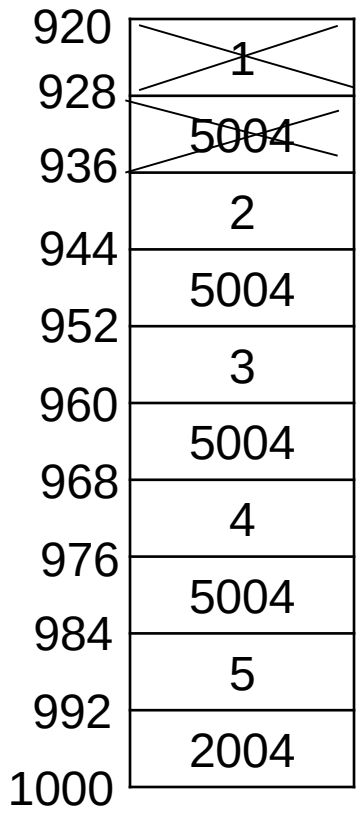




Example

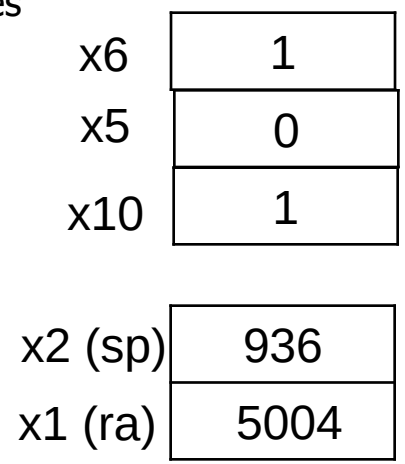
```
fact:
    addi sp,sp,-16          Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1          x5 = n - 1
    bge  x5,x0,L1           if n >= 0, go to L1
    addi x10,x0,1           Else, set return value to 1
    addi sp,sp,16           Pop stack, don't bother restoring values
    jr x1                   Return
L1: addi x10,x10,-1         n = n - 1
    jal  x1,fact            /*Address = 5000*/
    addi x6,x10,0           move result of fact(n - 1) to x6
    ld   x10,0(sp)          Restore caller's n
    ld   x1,8(sp)           Restore caller's return address
    addi sp,sp,16           Pop stack
    mul  x10,x10,x6         return n * fact(n-1)
    jr x1                   return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

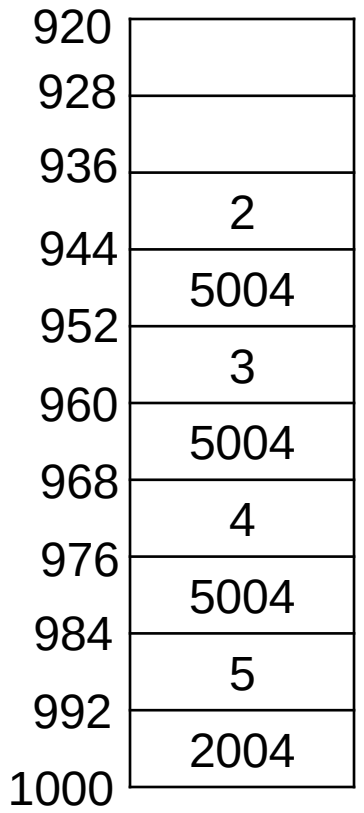




Example

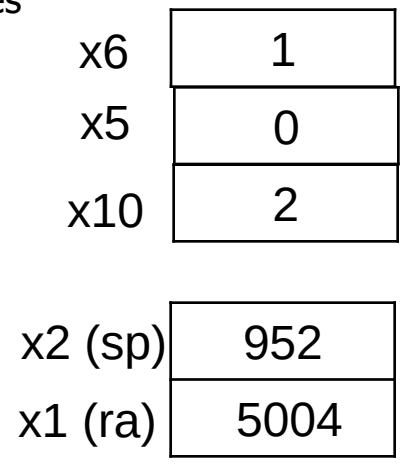
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

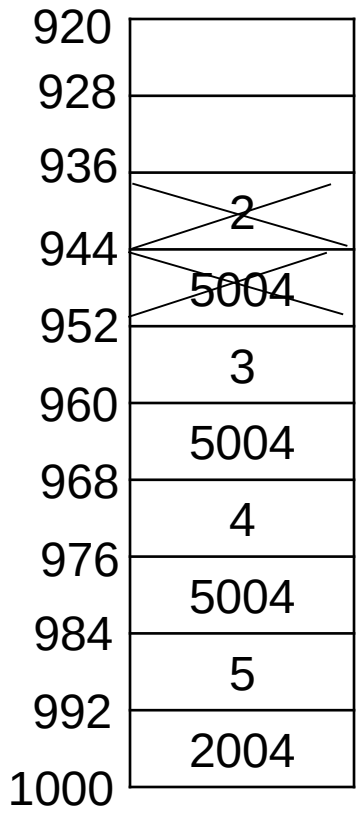




Example

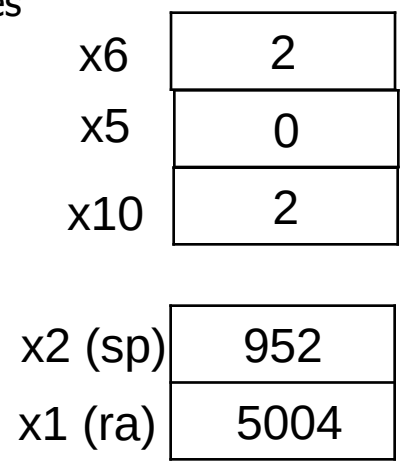
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

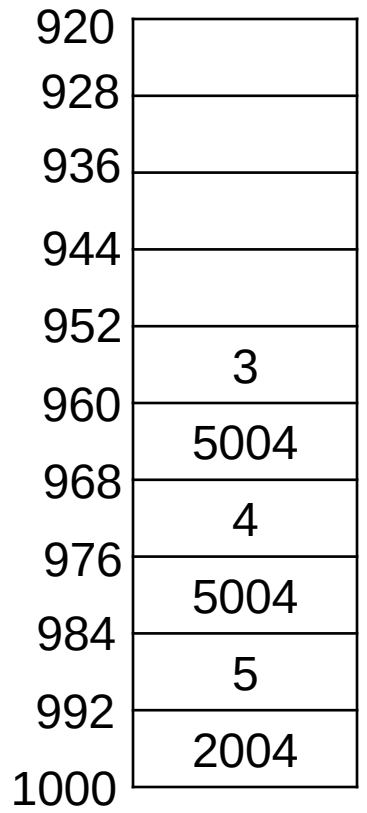




Example

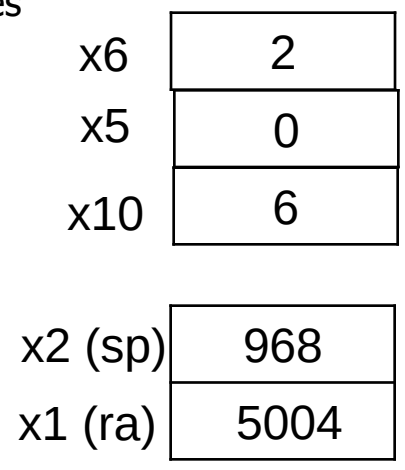
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

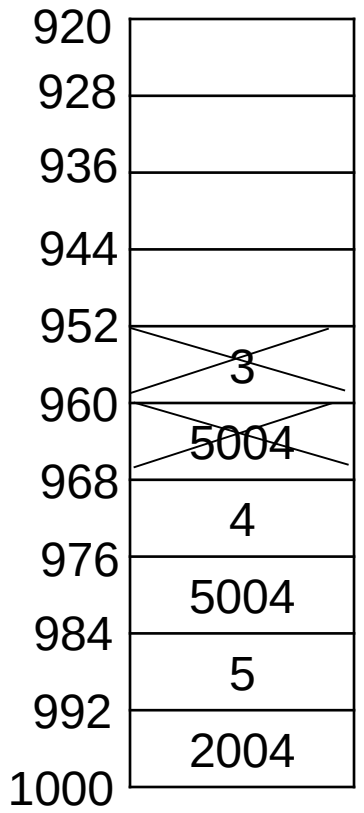




Example

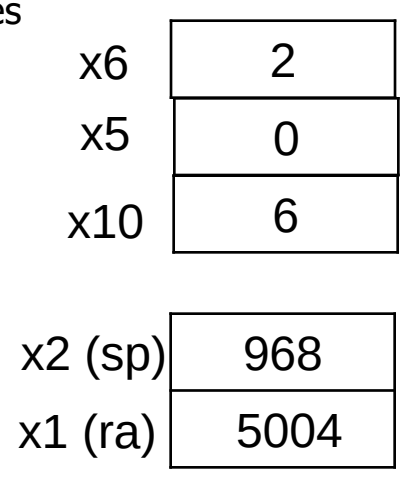
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

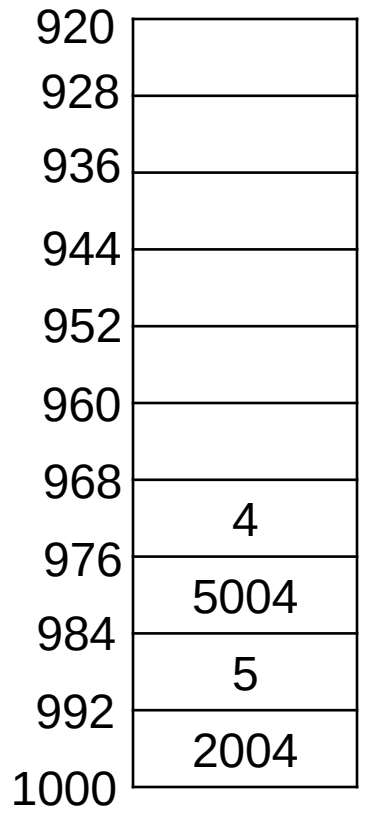




Example

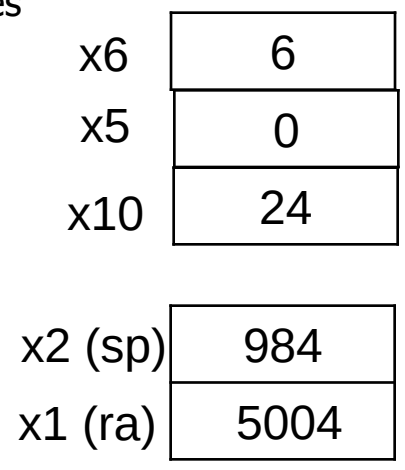
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

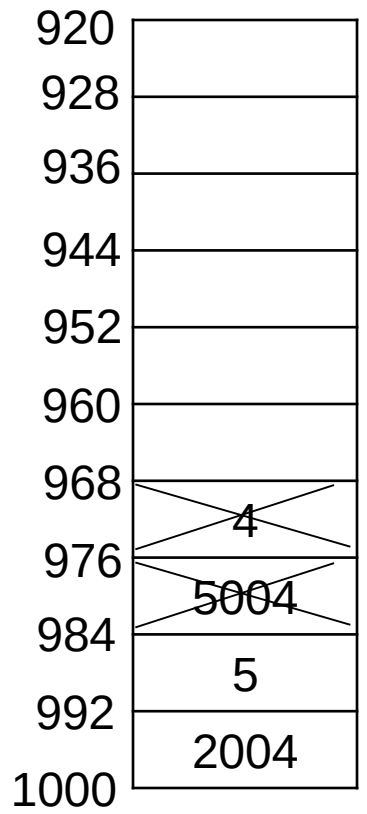




Example

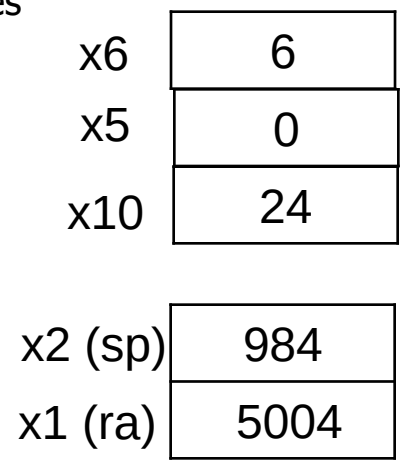
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

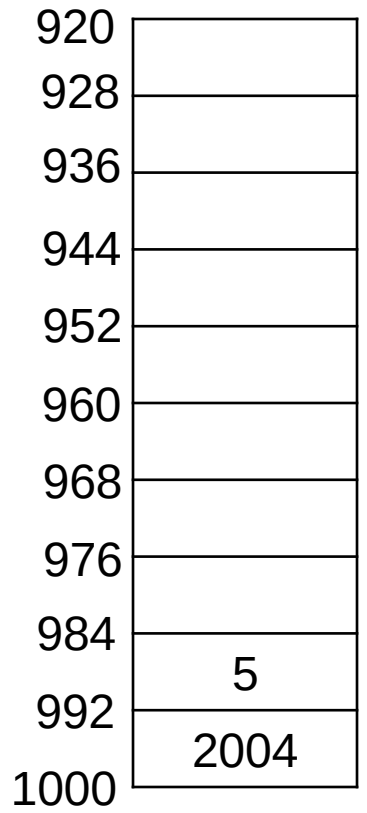




Example

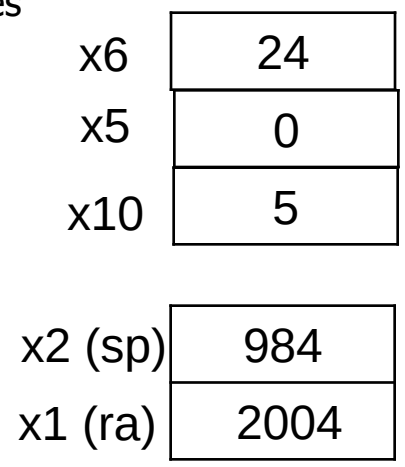
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

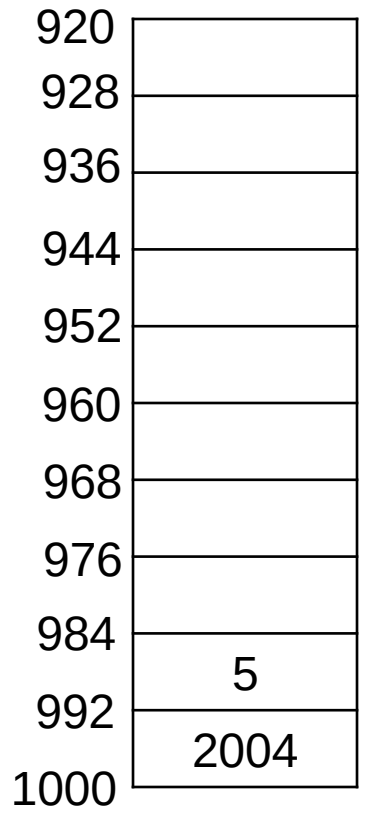




Example

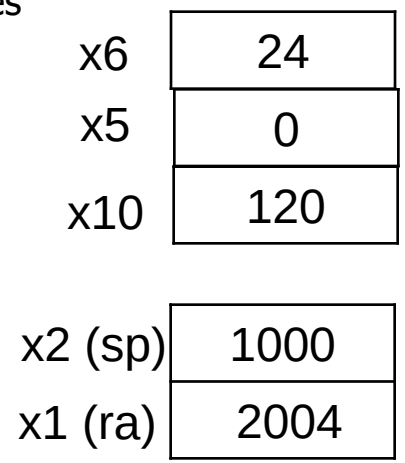
```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```

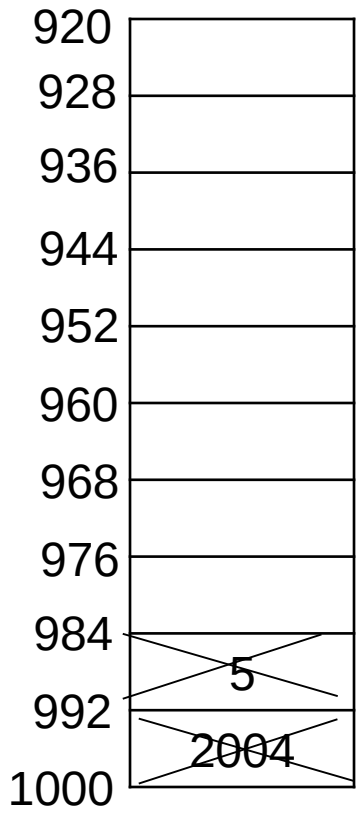




Example

```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```





Example

```
fact:
    addi sp,sp,-16      Save return address and n on stack
    sd   x1,8(sp)
    sd   x10,0(sp)
    addi x5,x10,-1      x5 = n - 1
    bge  x5,x0,L1       if n >= 0, go to L1
    addi x10,x0,1       Else, set return value to 1
    addi sp,sp,16       Pop stack, don't bother restoring values
    jr x1               Return
L1: addi x10,x10,-1     n = n - 1
    jal  x1,fact        /*Address = 5000*/
    addi x6,x10,0       move result of fact(n - 1) to x6
    ld   x10,0(sp)      Restore caller's n
    ld   x1,8(sp)       Restore caller's return address
    addi sp,sp,16       Pop stack
    mul  x10,x10,x6     return n * fact(n-1)
    jr x1               return
main:
    addi x10, x0, 5
    jal  x1, fact #addr=2000
```