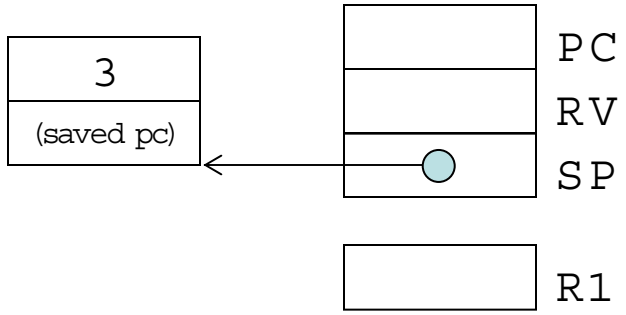
```
int fact(int n)
{
  if (n == 0) return 1;
  return n * fact(n - 1);
}
```

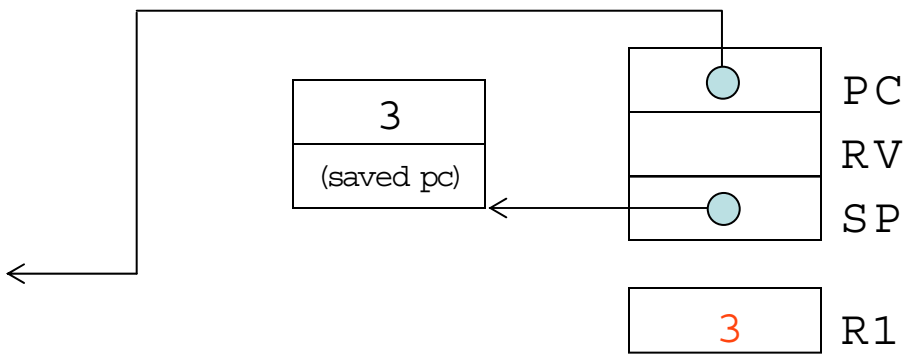
```
int fact(int n)
{
  if (n == 0) return 1;
  return n * fact(n - 1);
}
```

```
R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



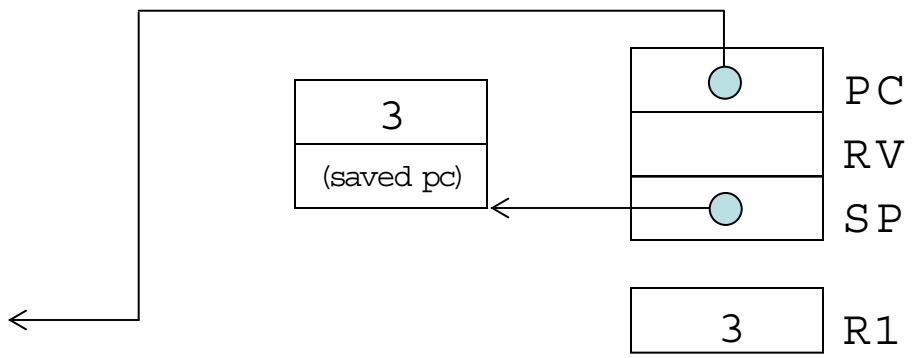
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}
```

```
R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



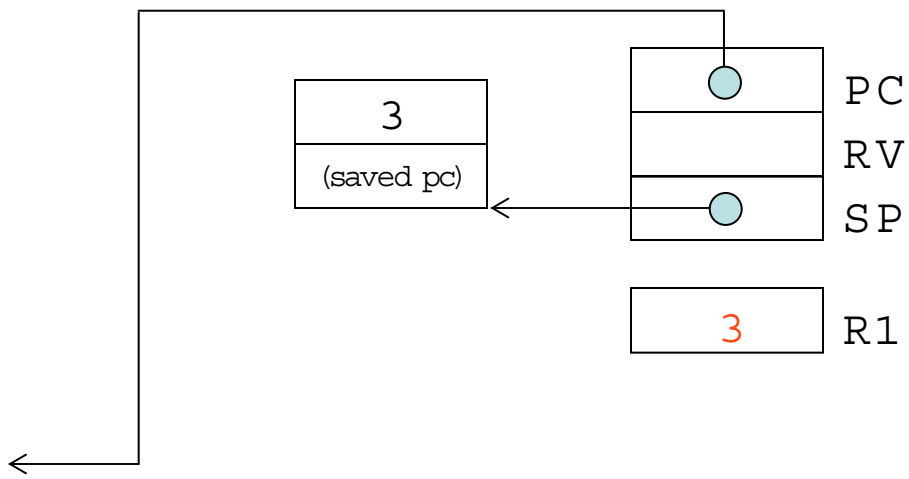
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}
```

```
R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



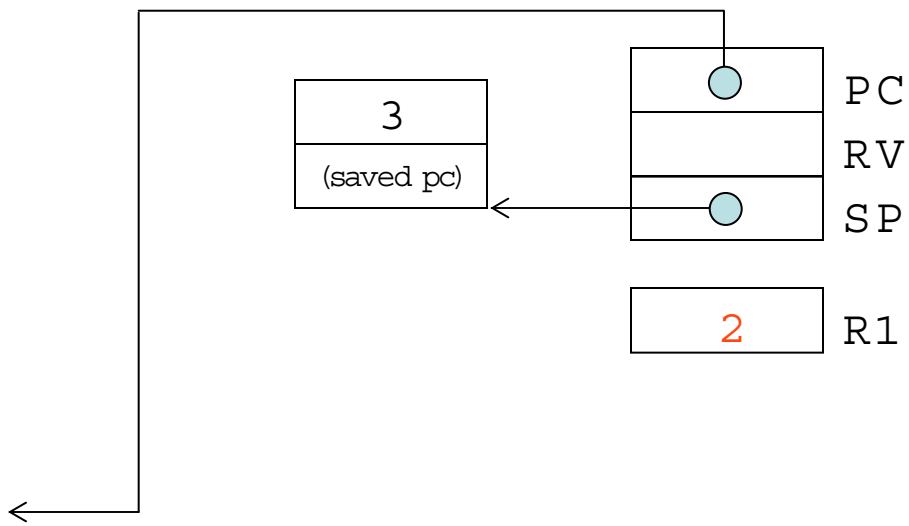
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



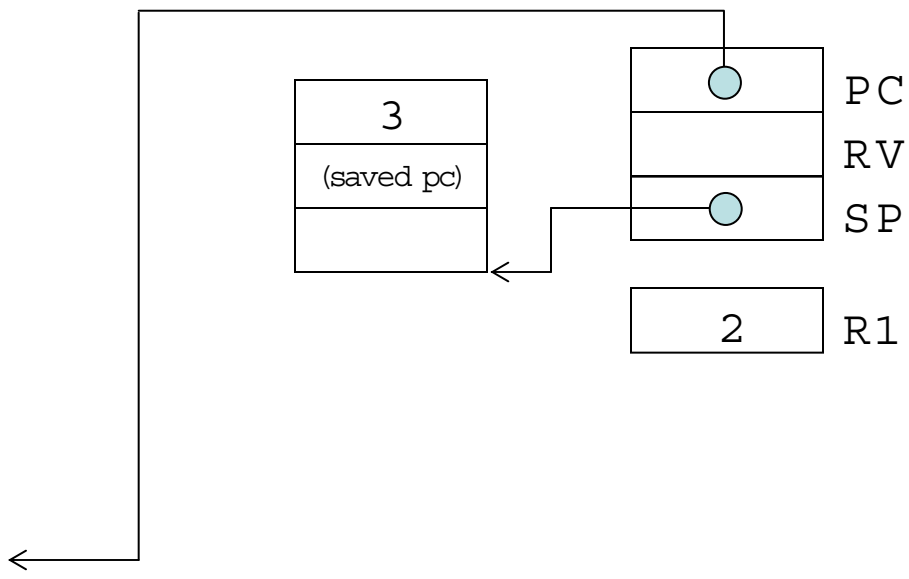
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



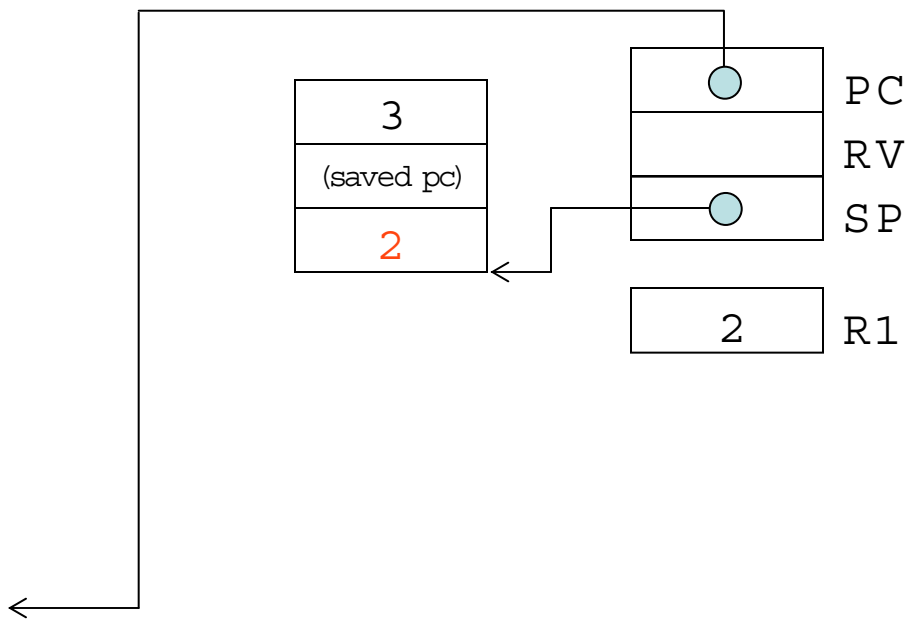
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



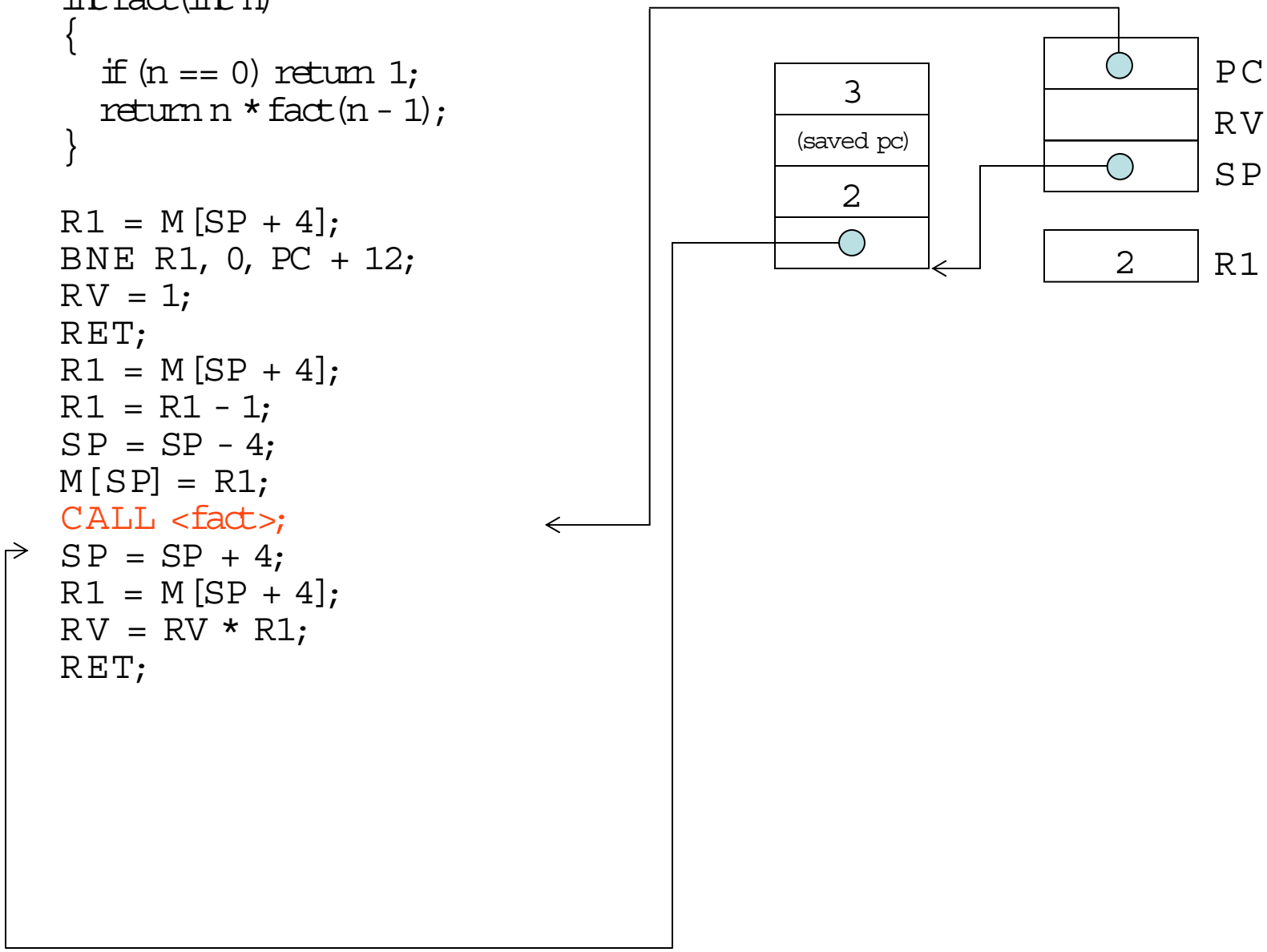
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



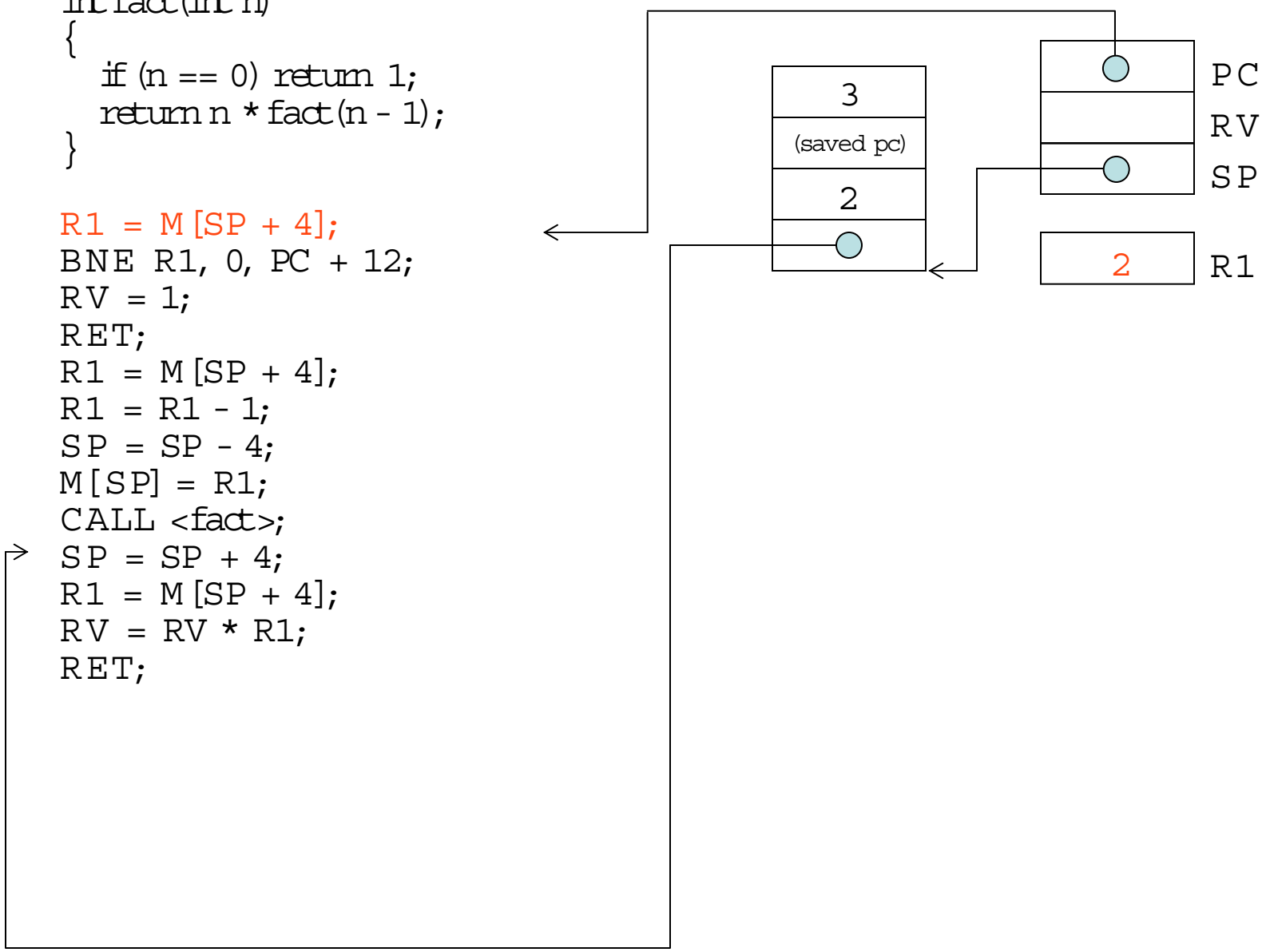
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



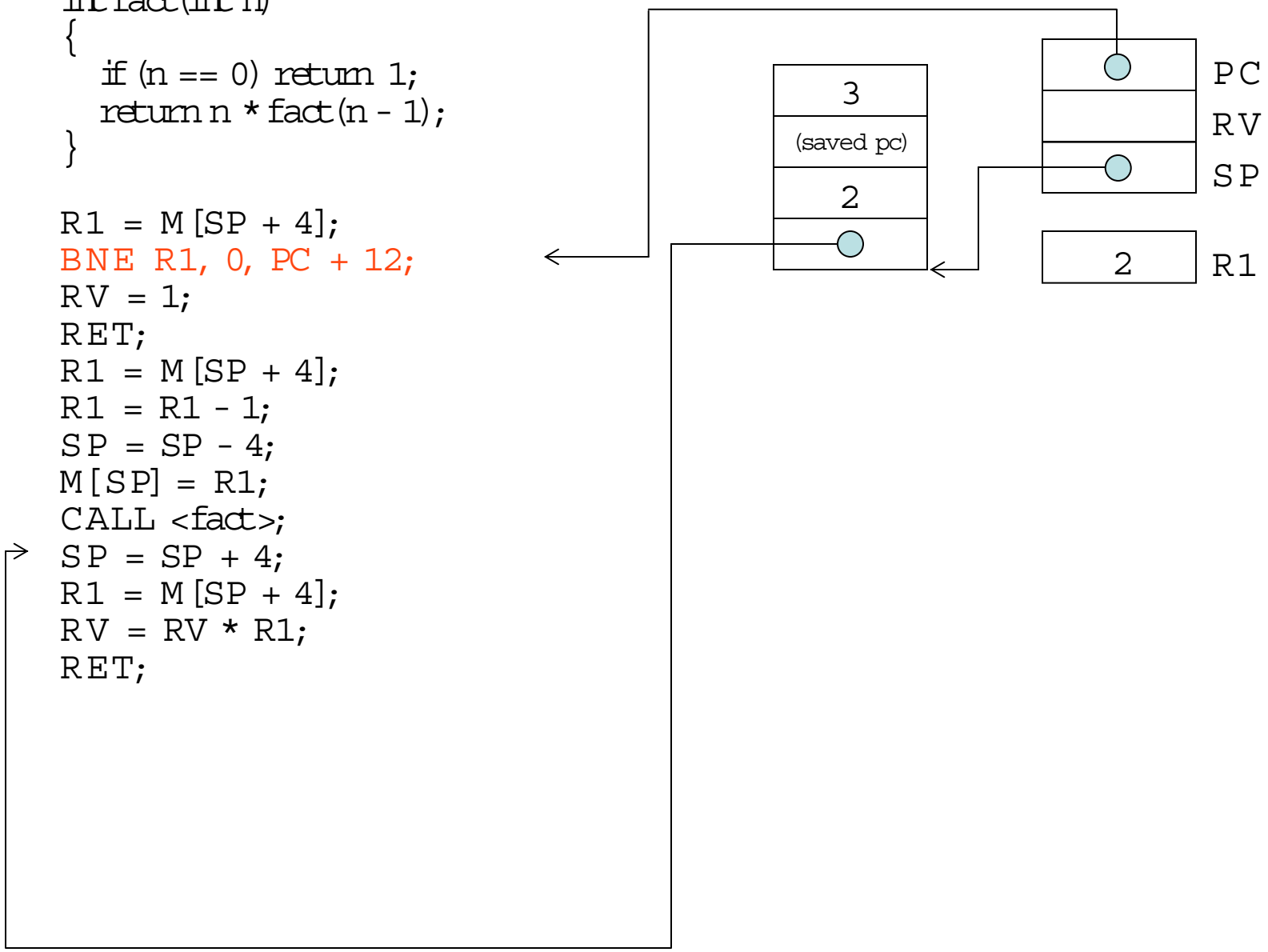
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



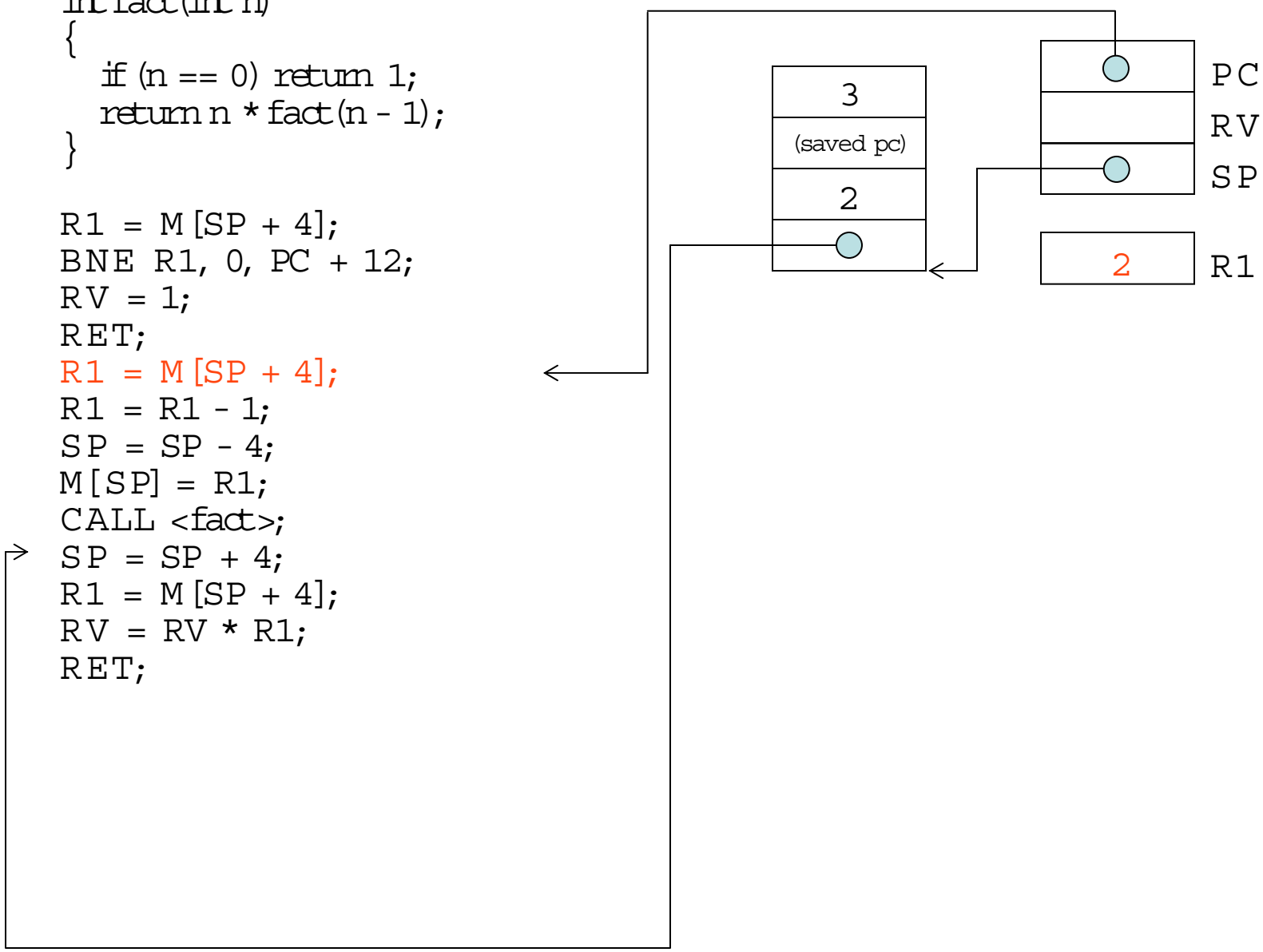
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



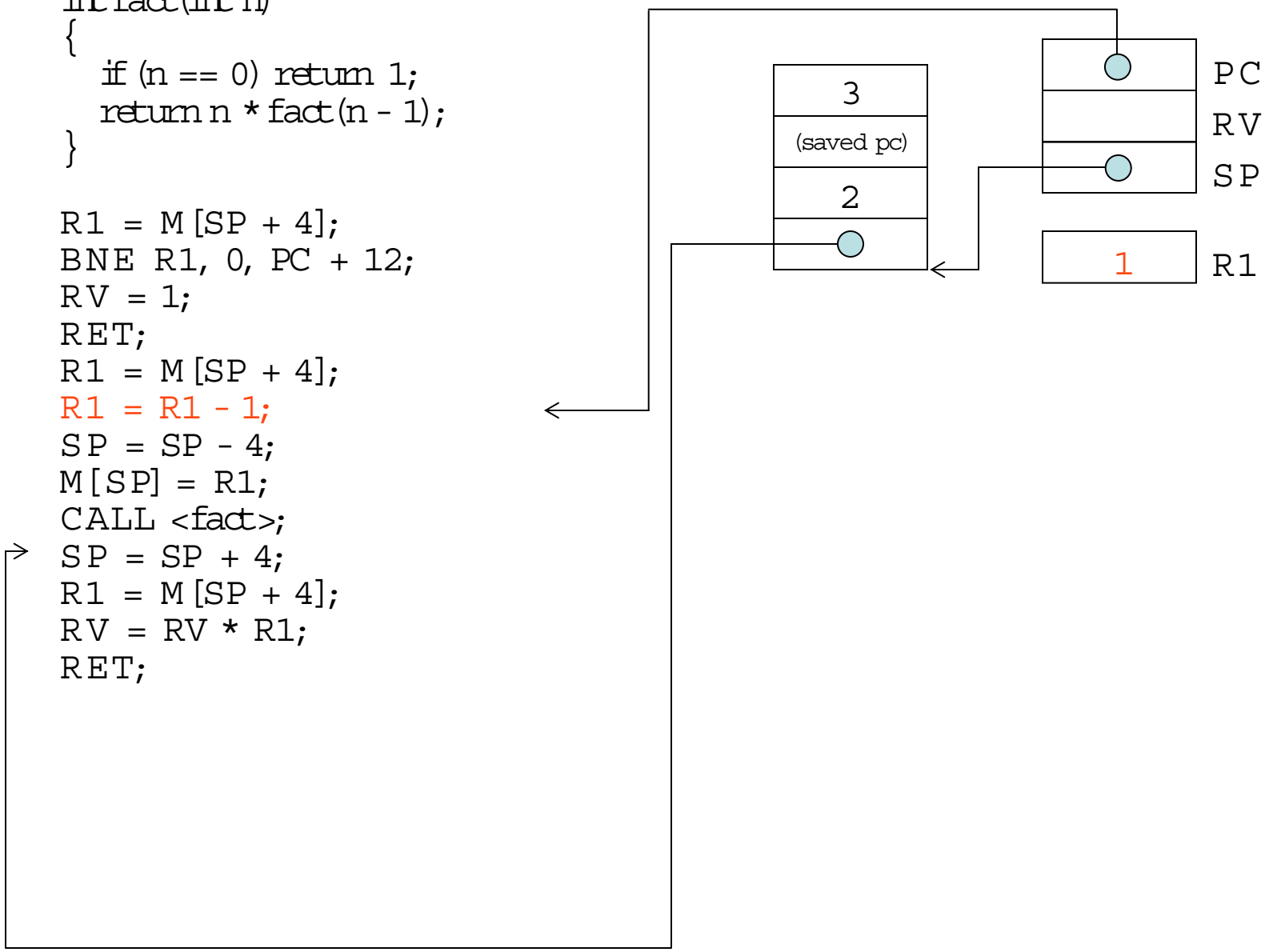
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



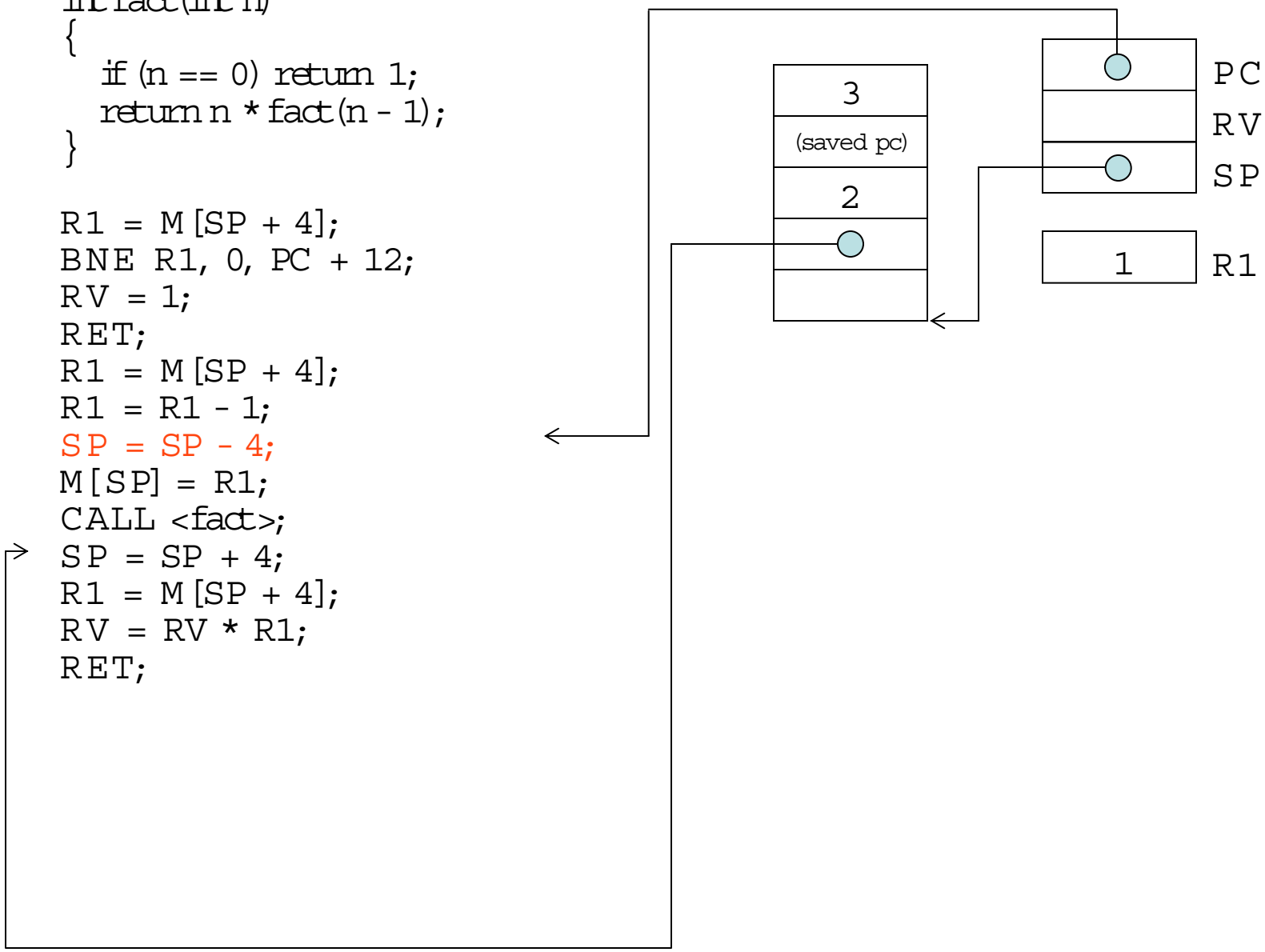
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



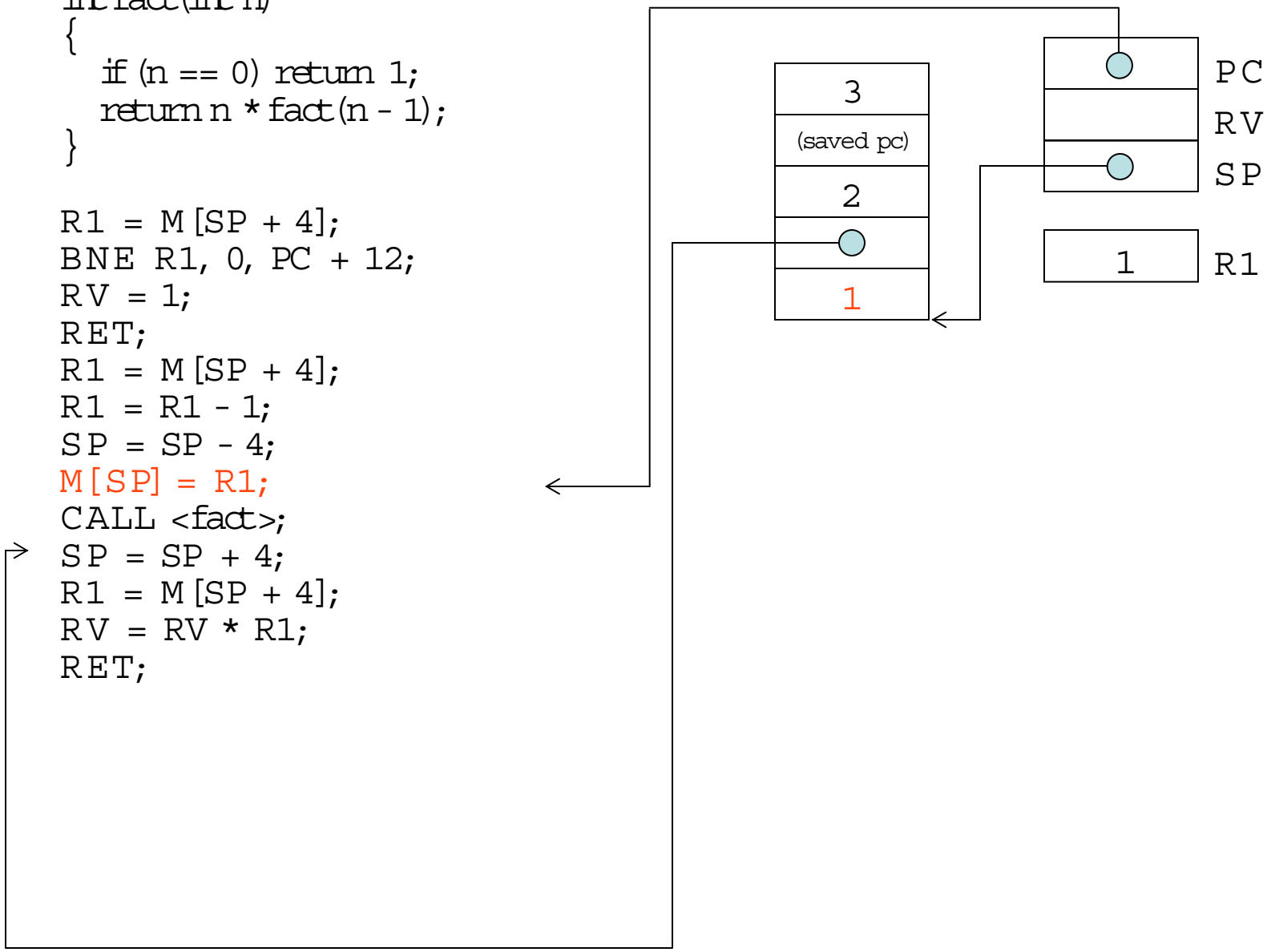
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



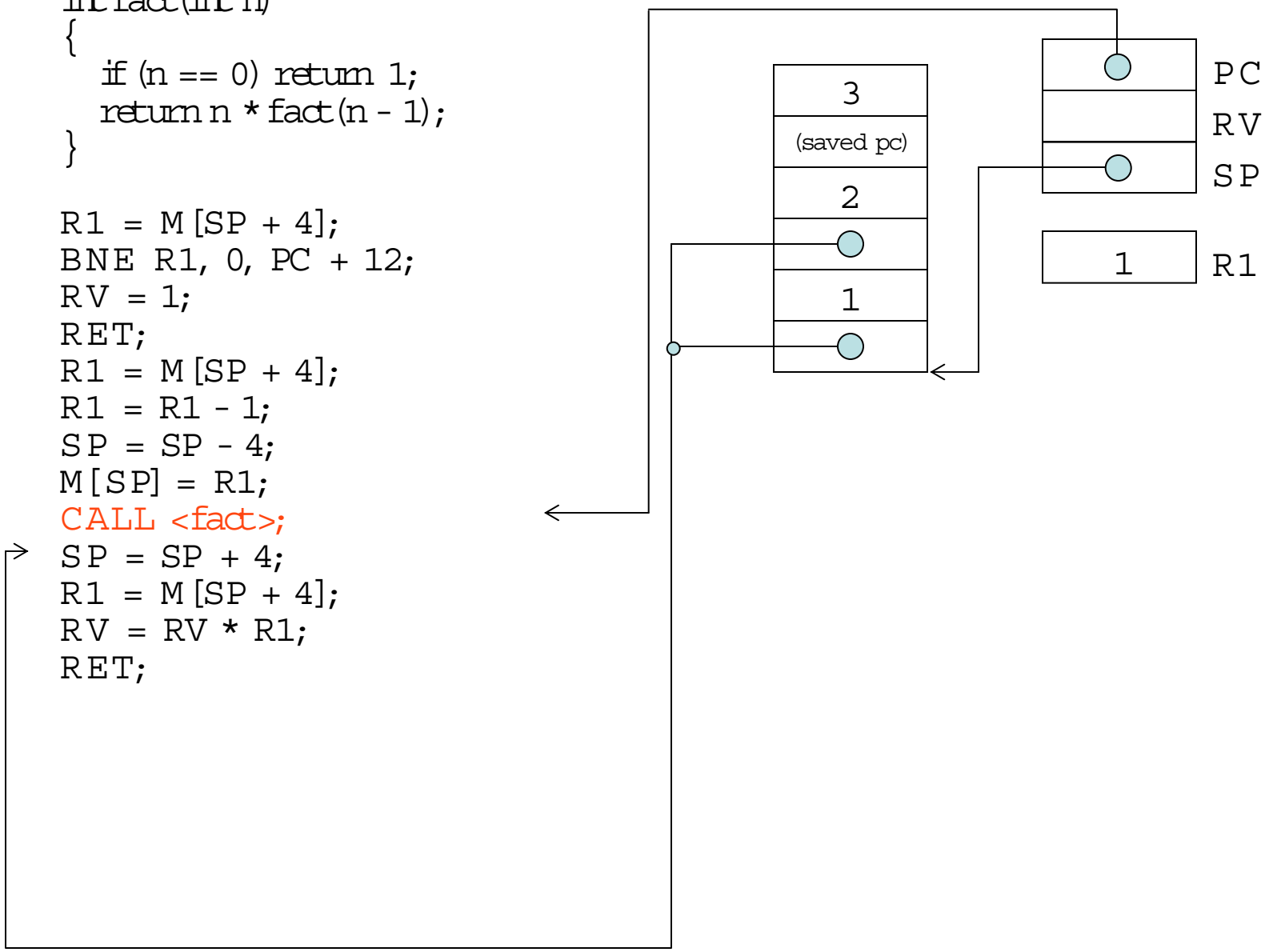
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



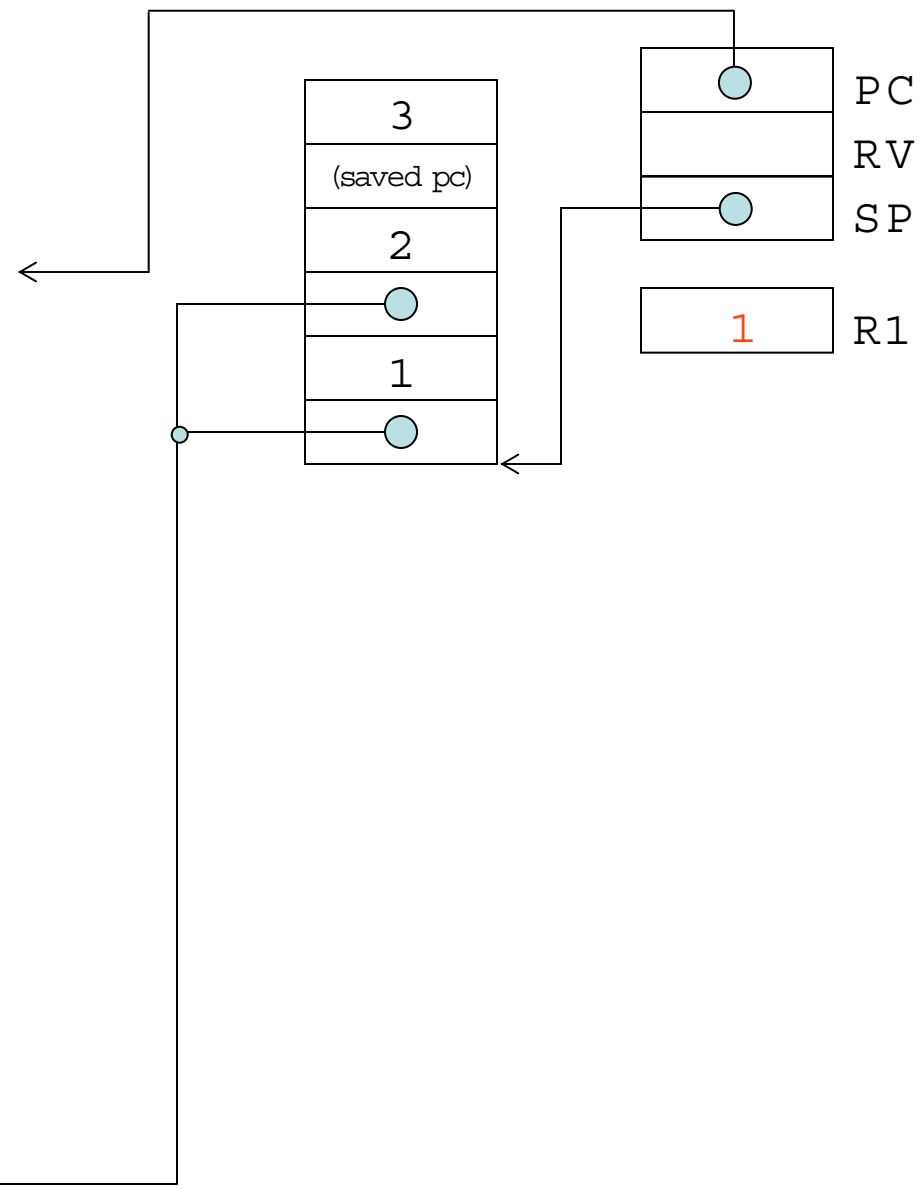
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



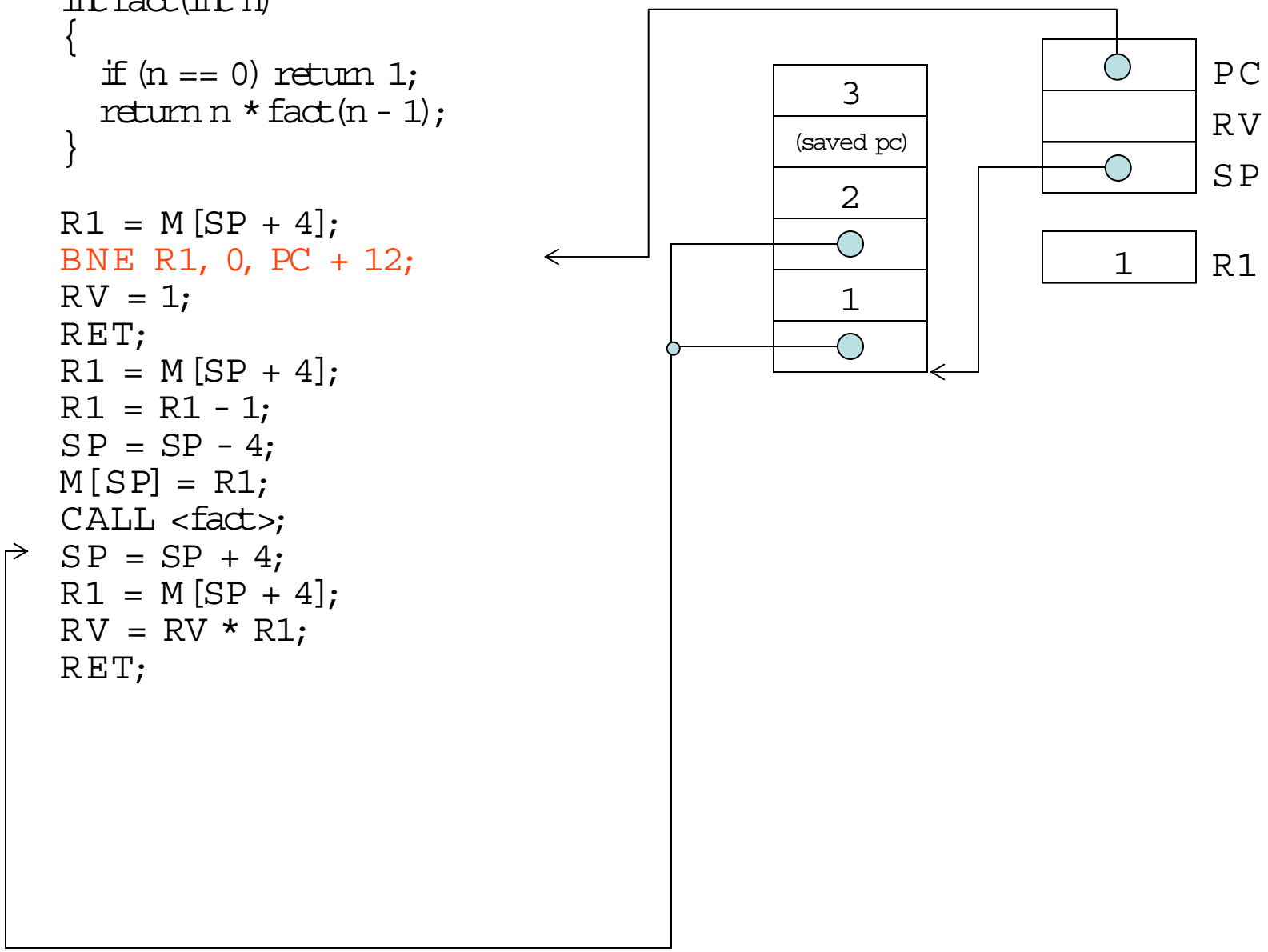
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



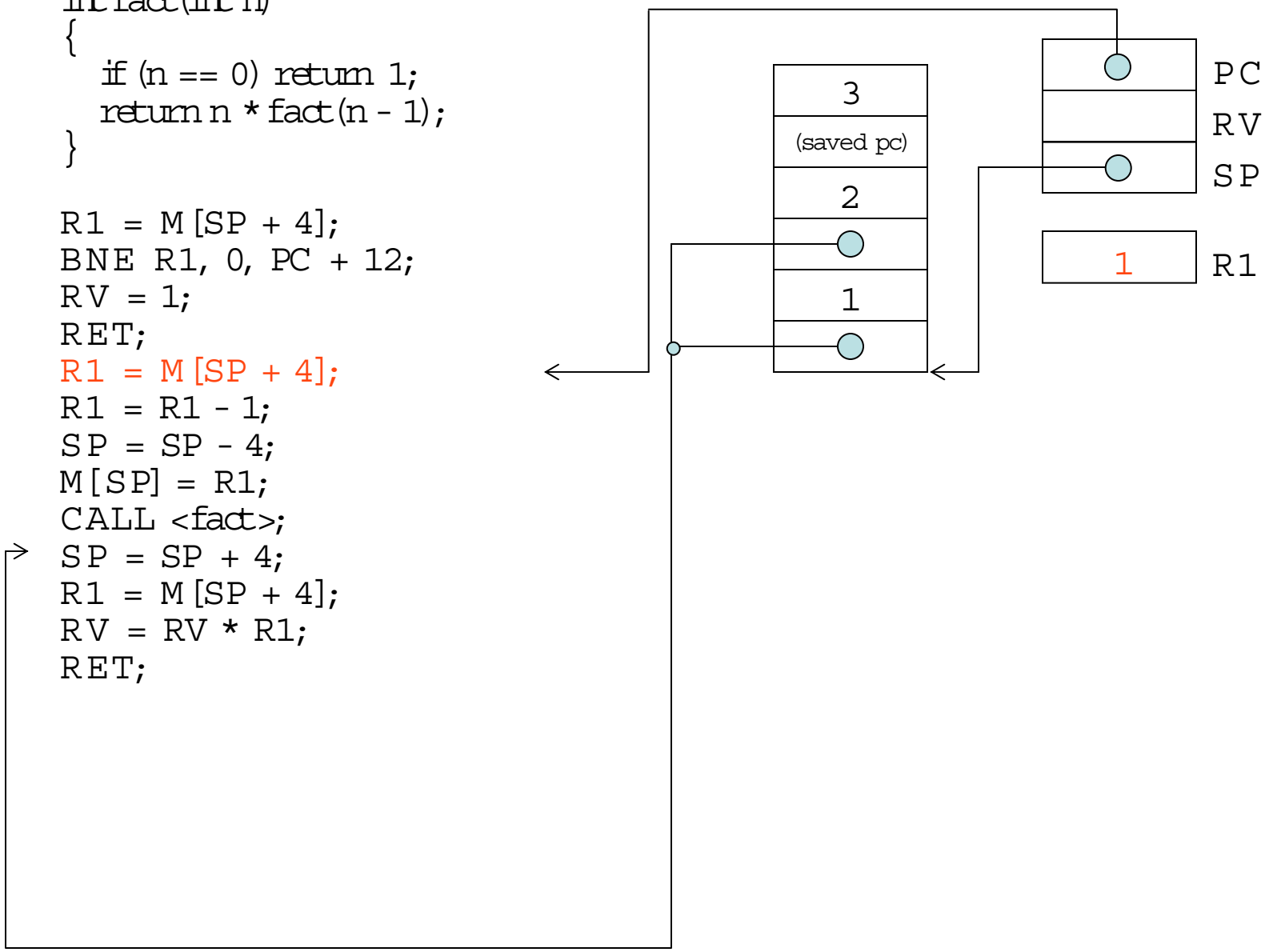
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



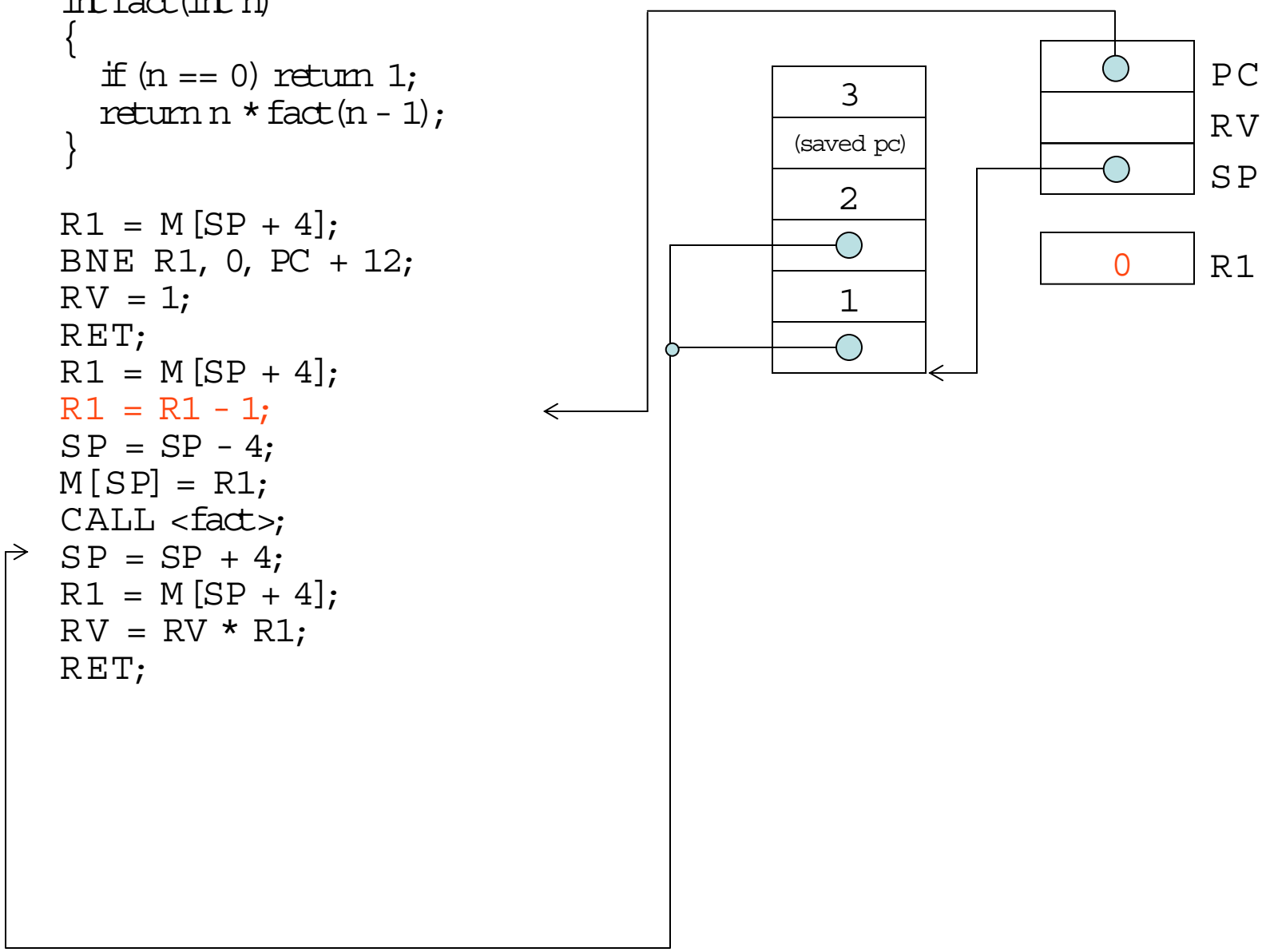
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



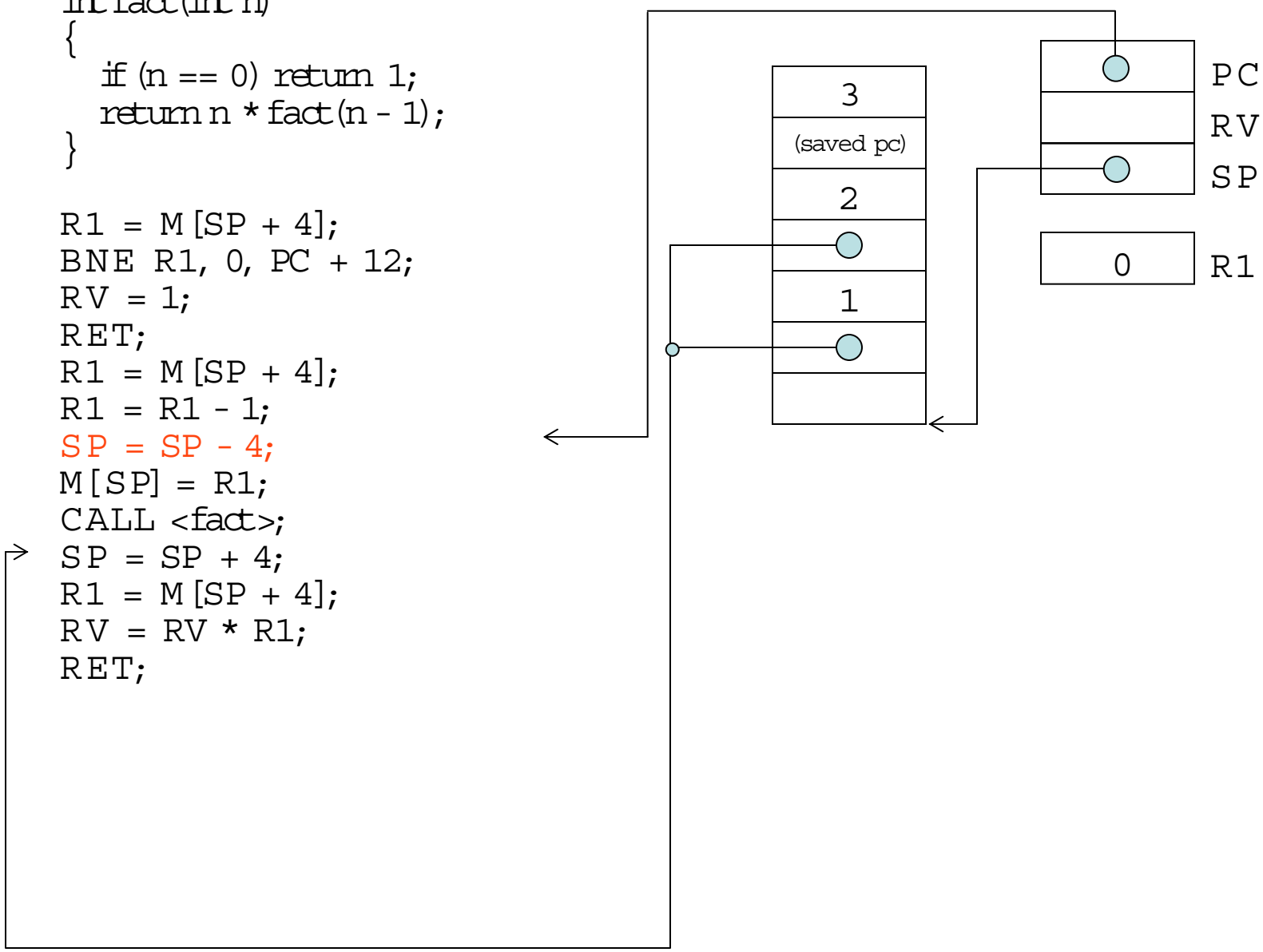
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



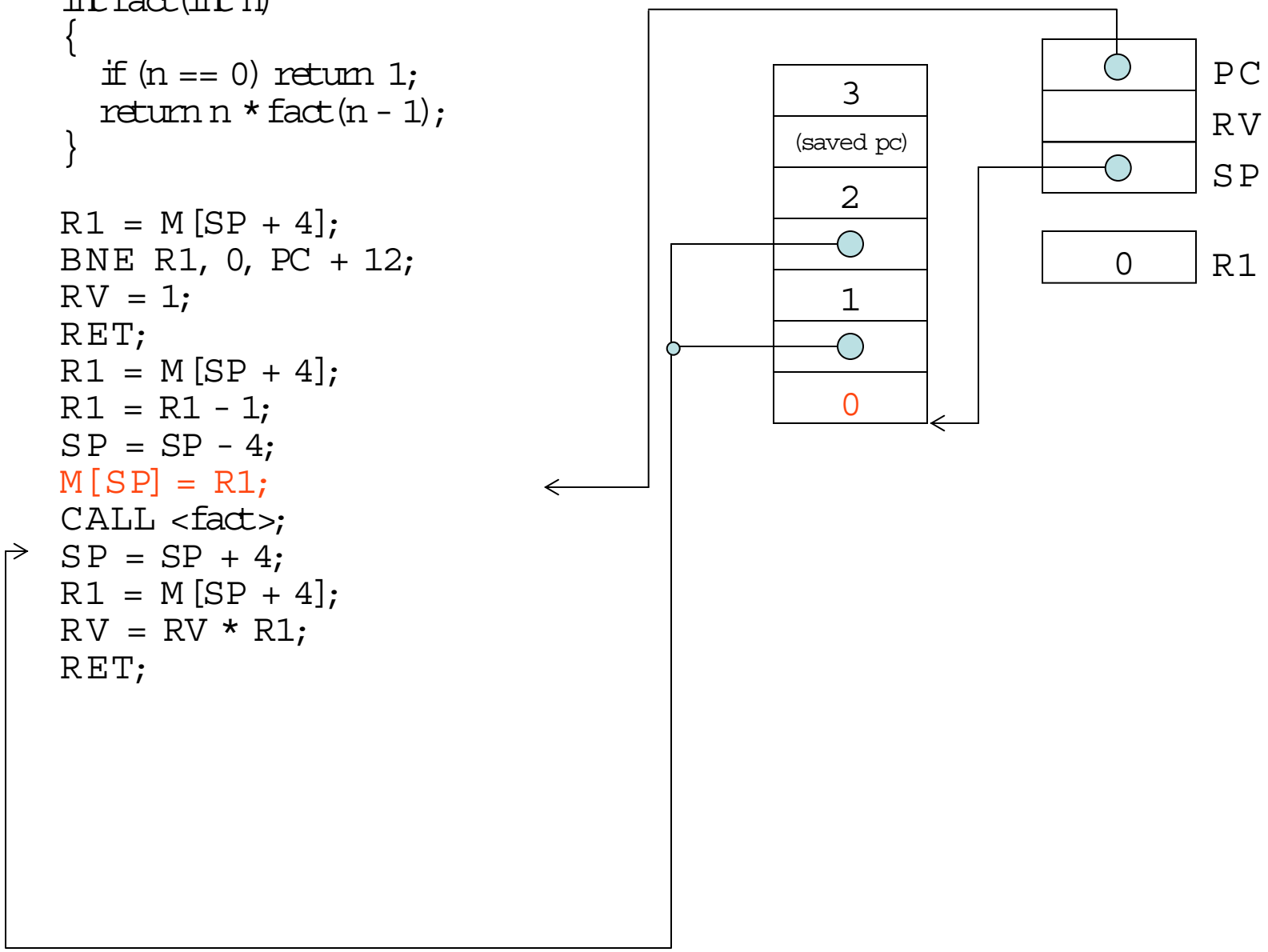
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



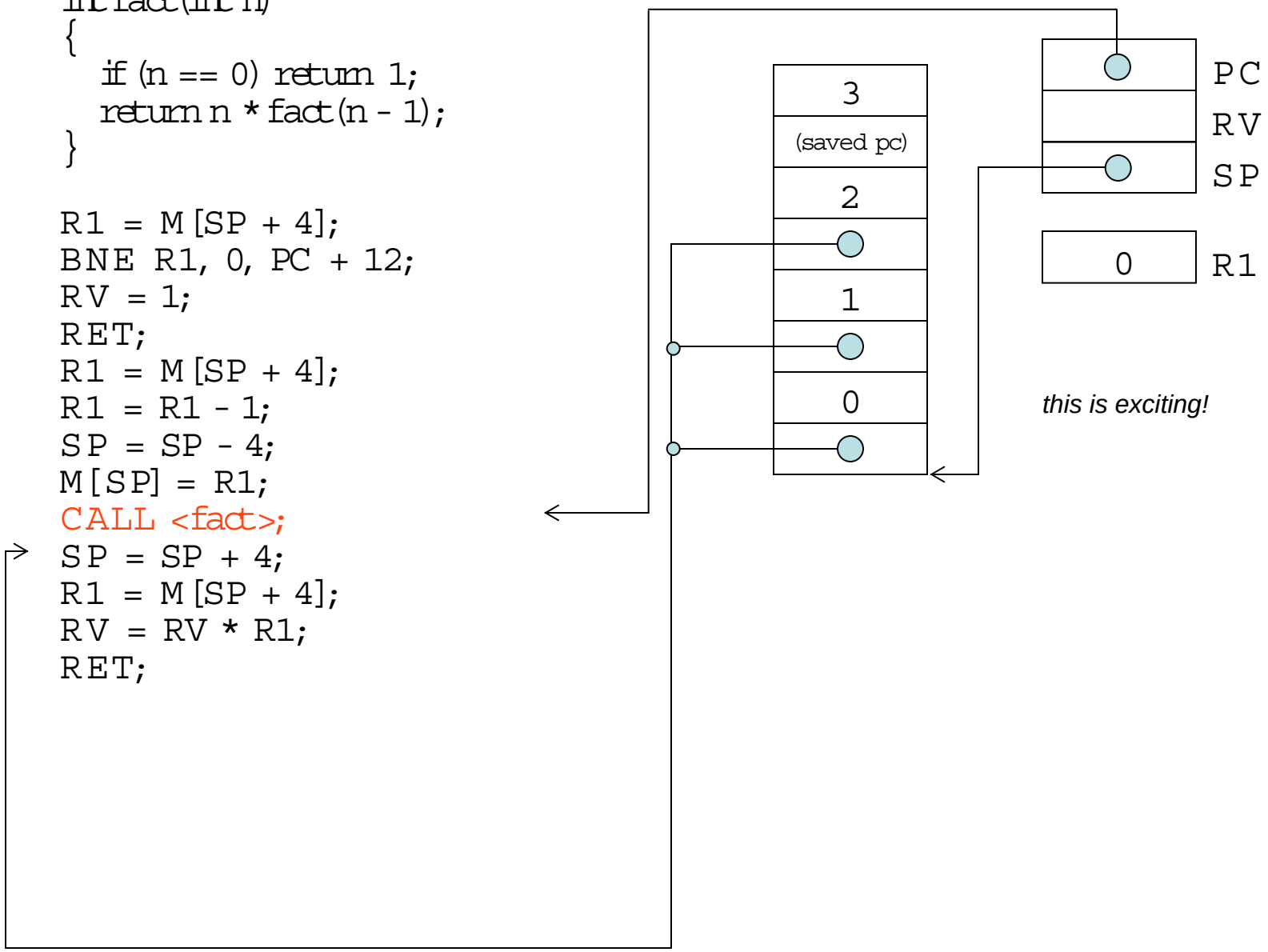
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



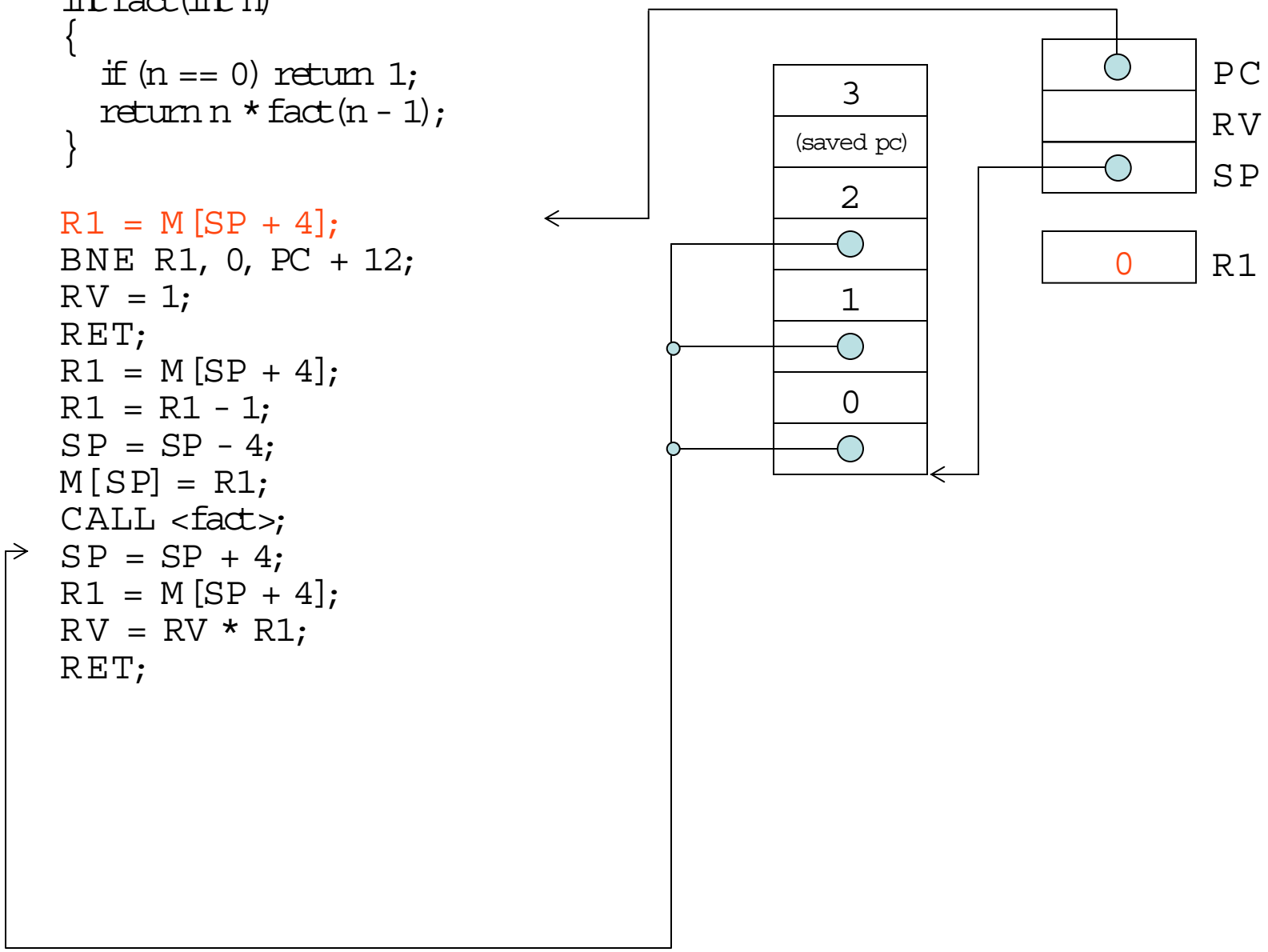
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



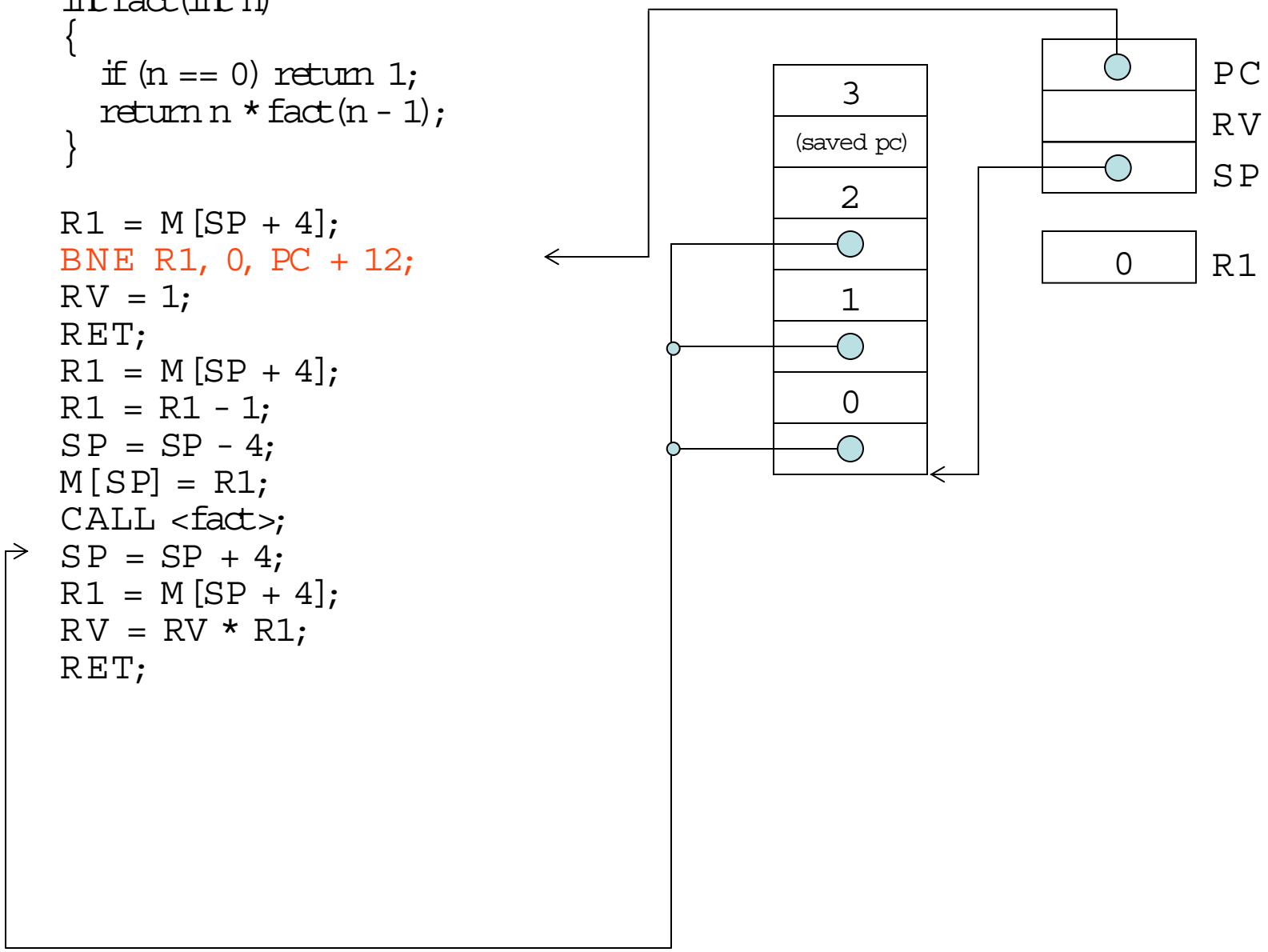
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



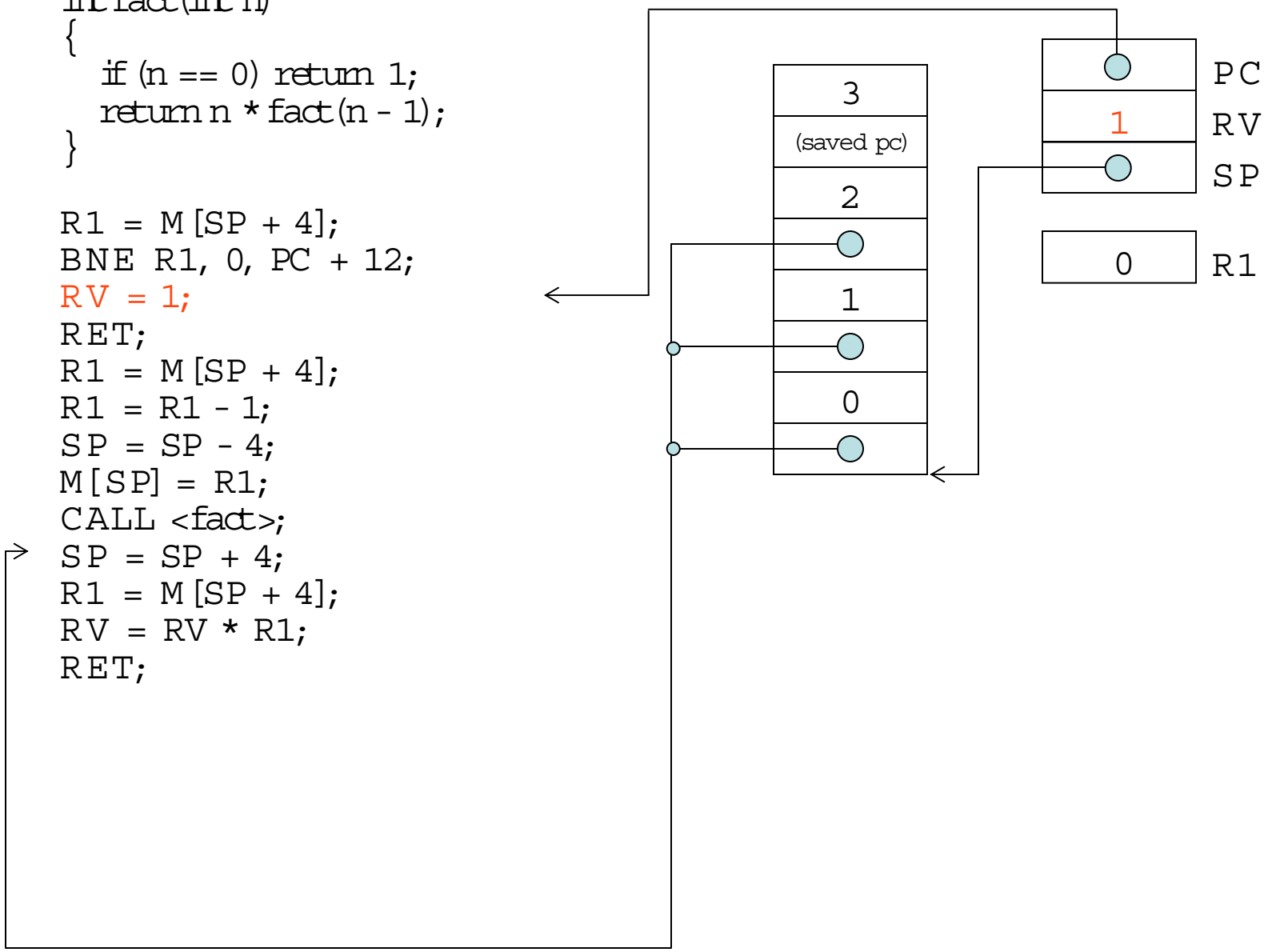
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



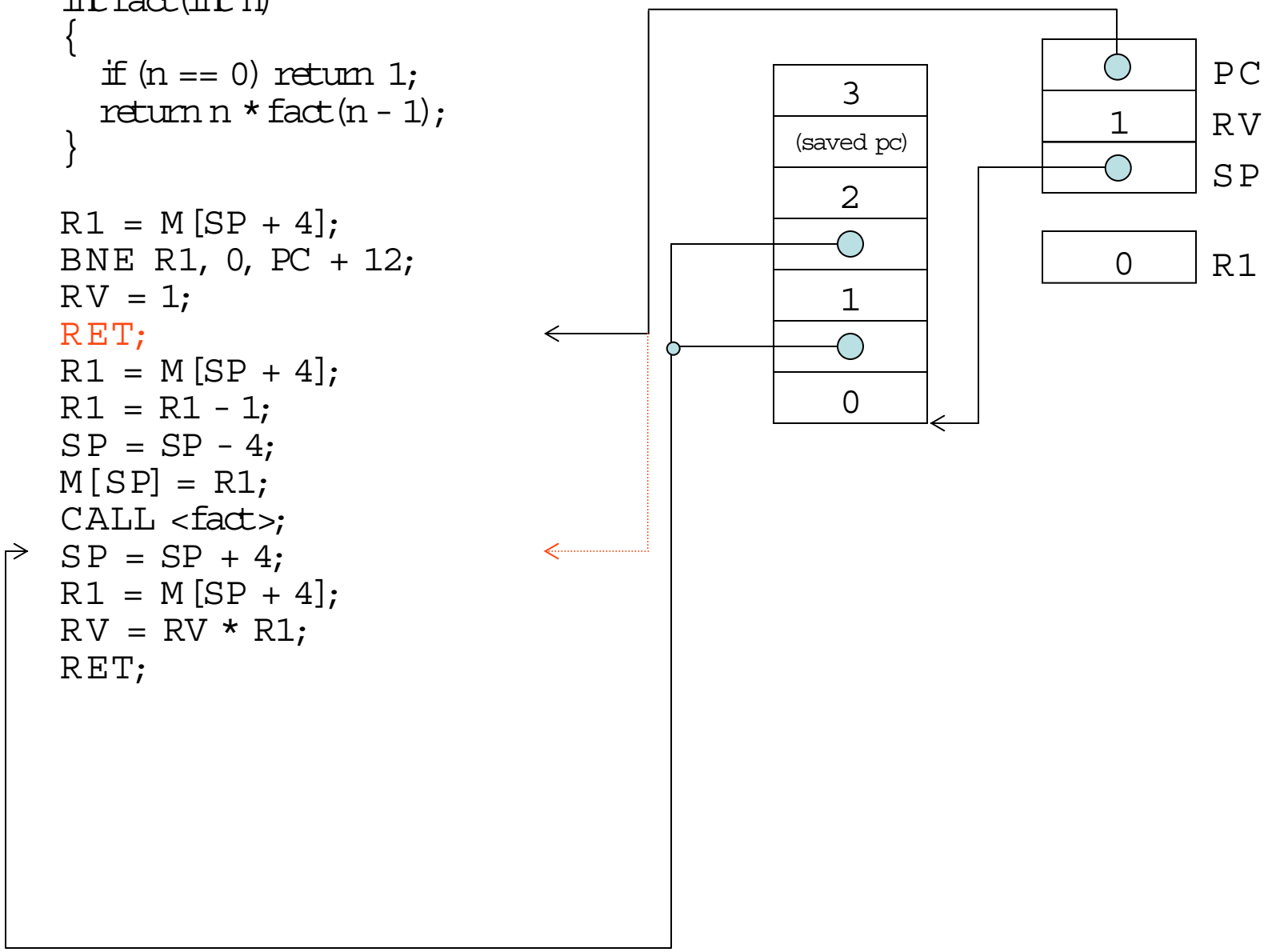
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



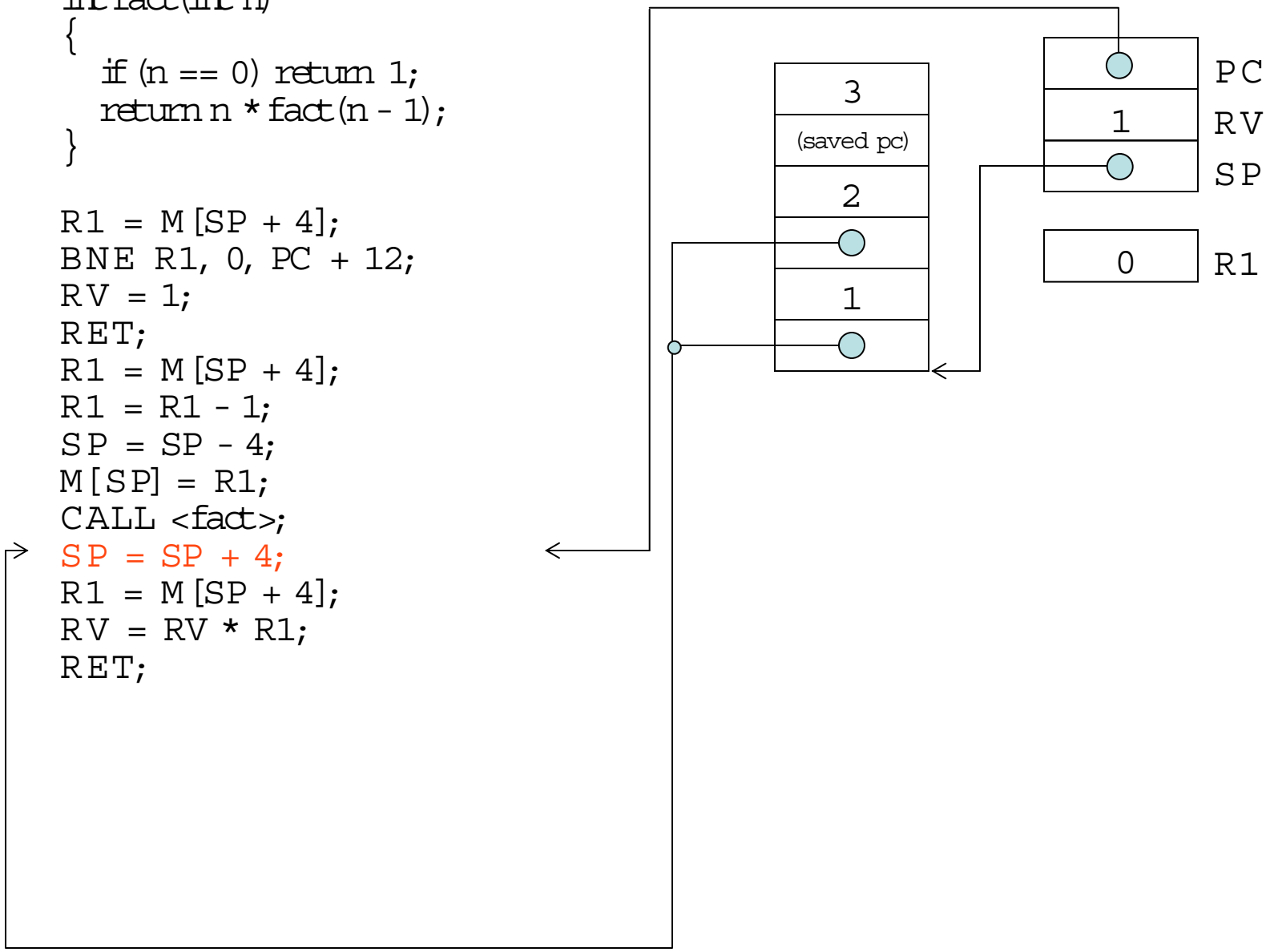
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



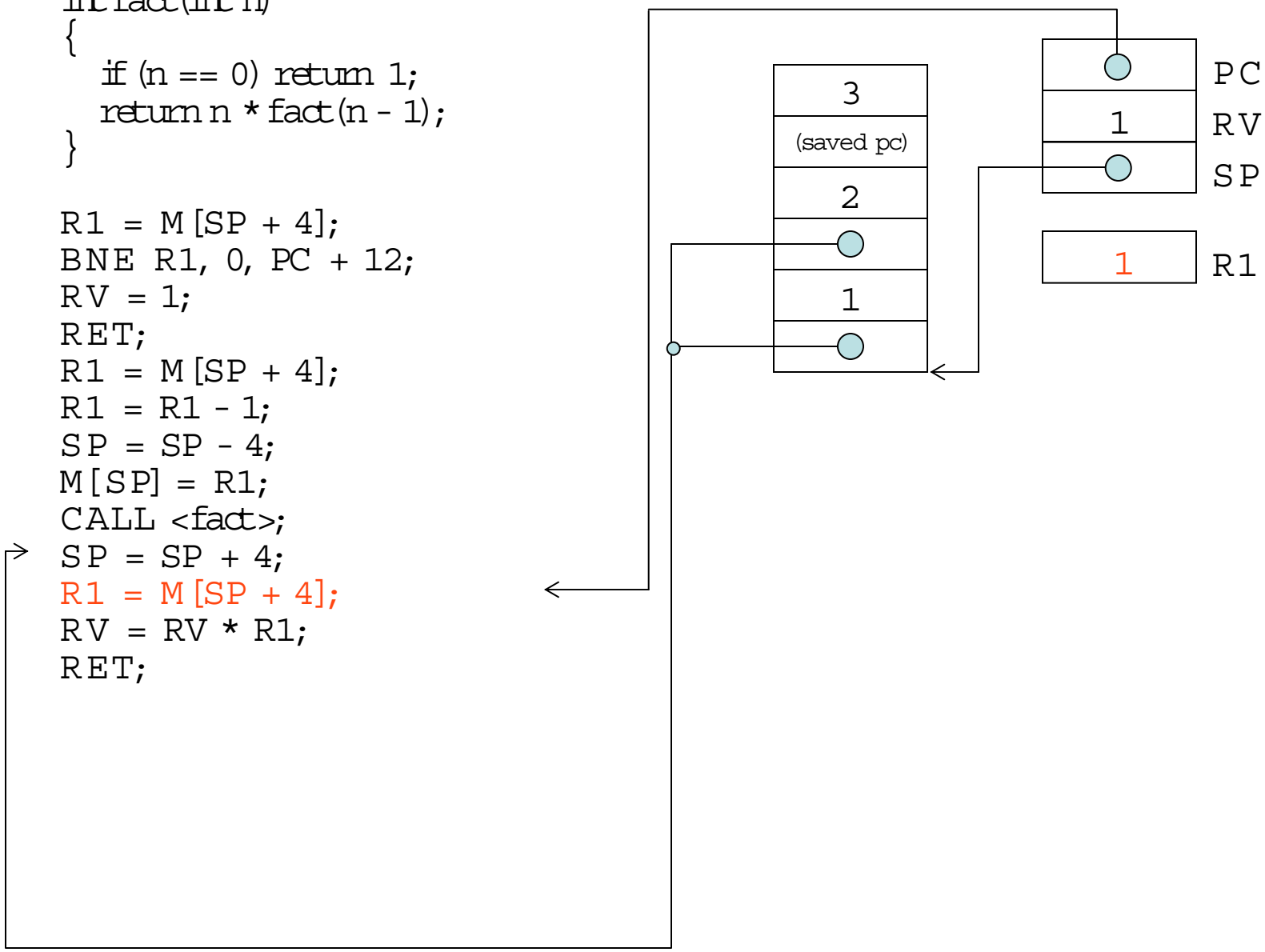
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



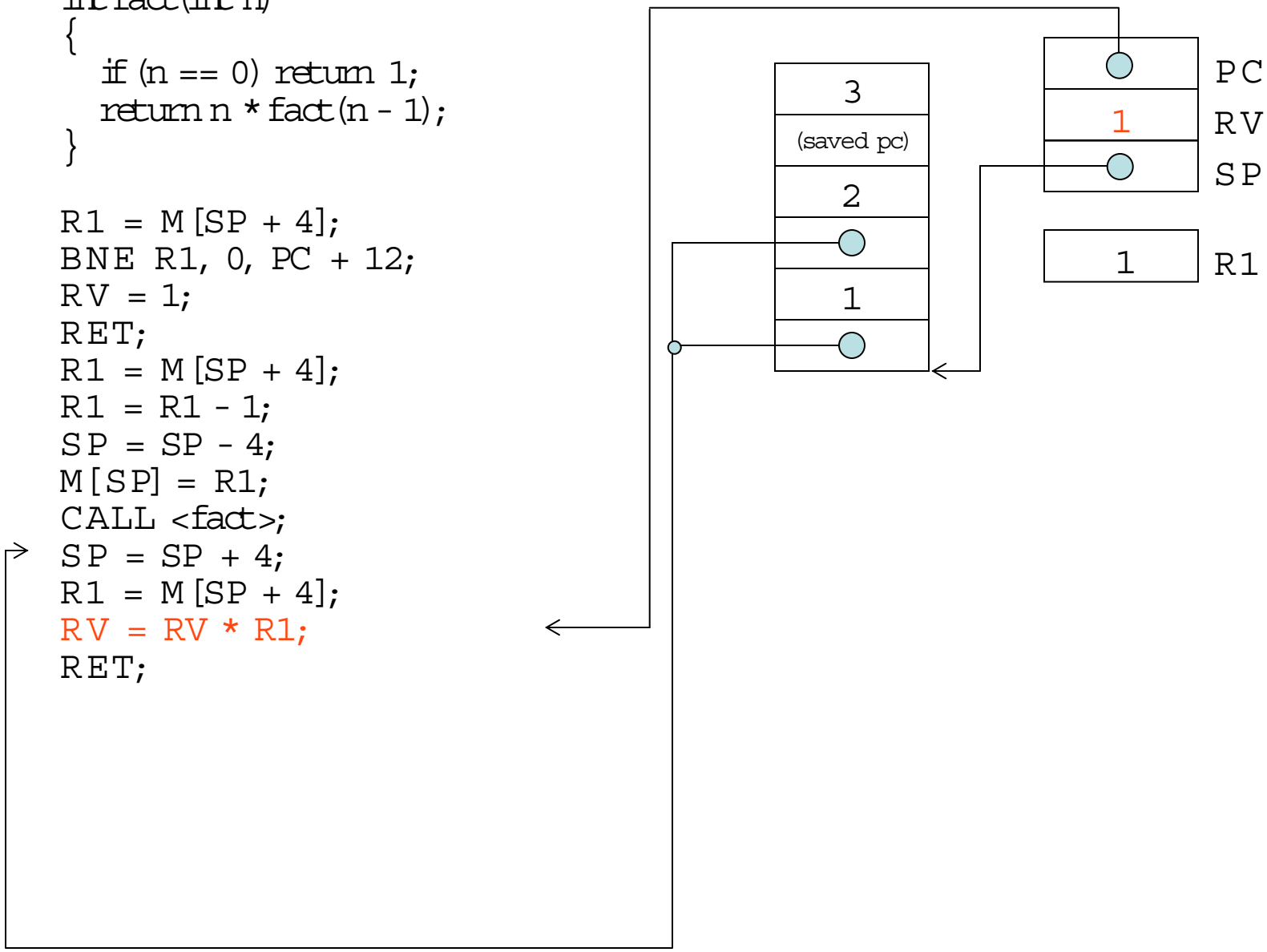
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



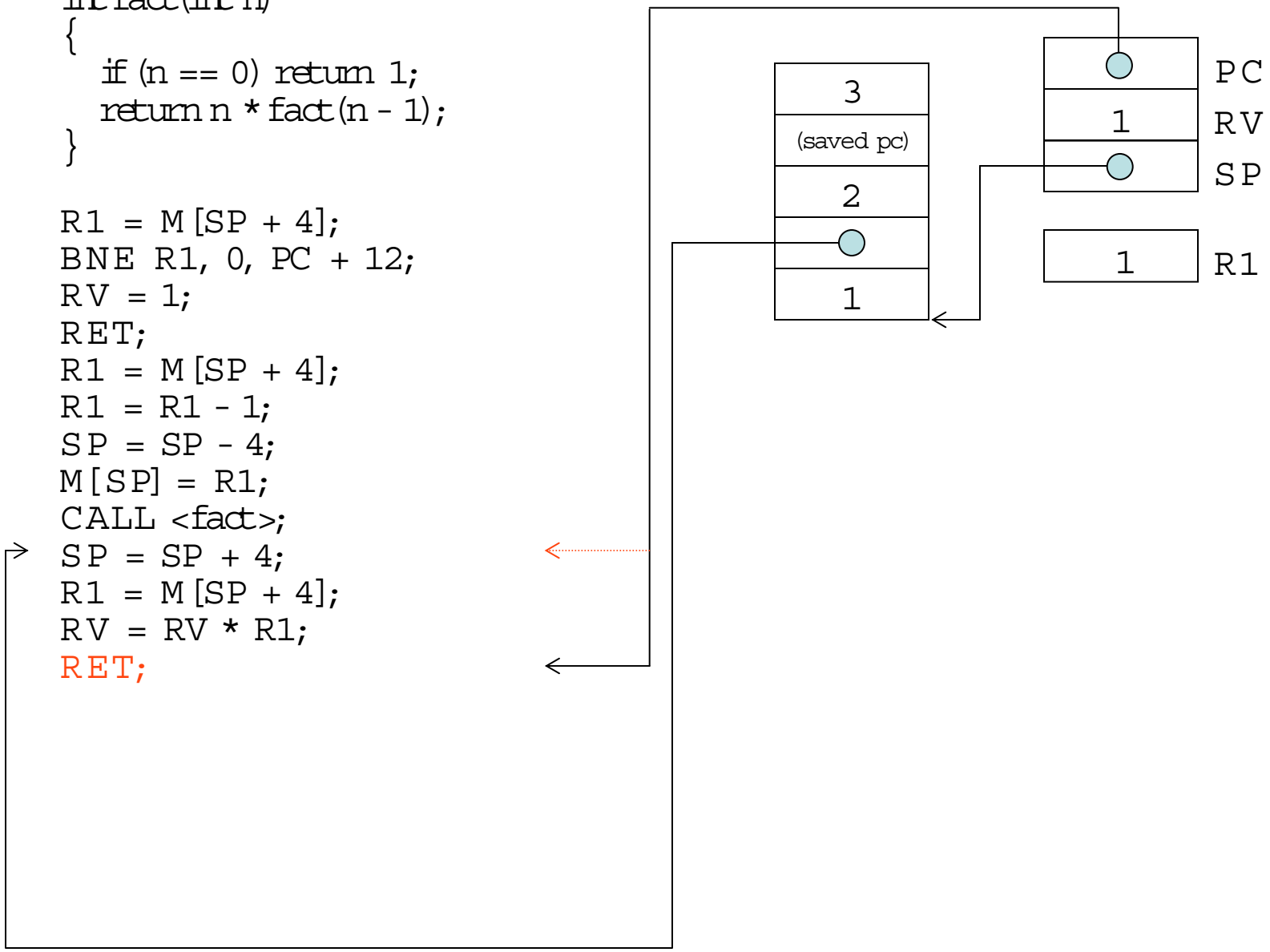
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



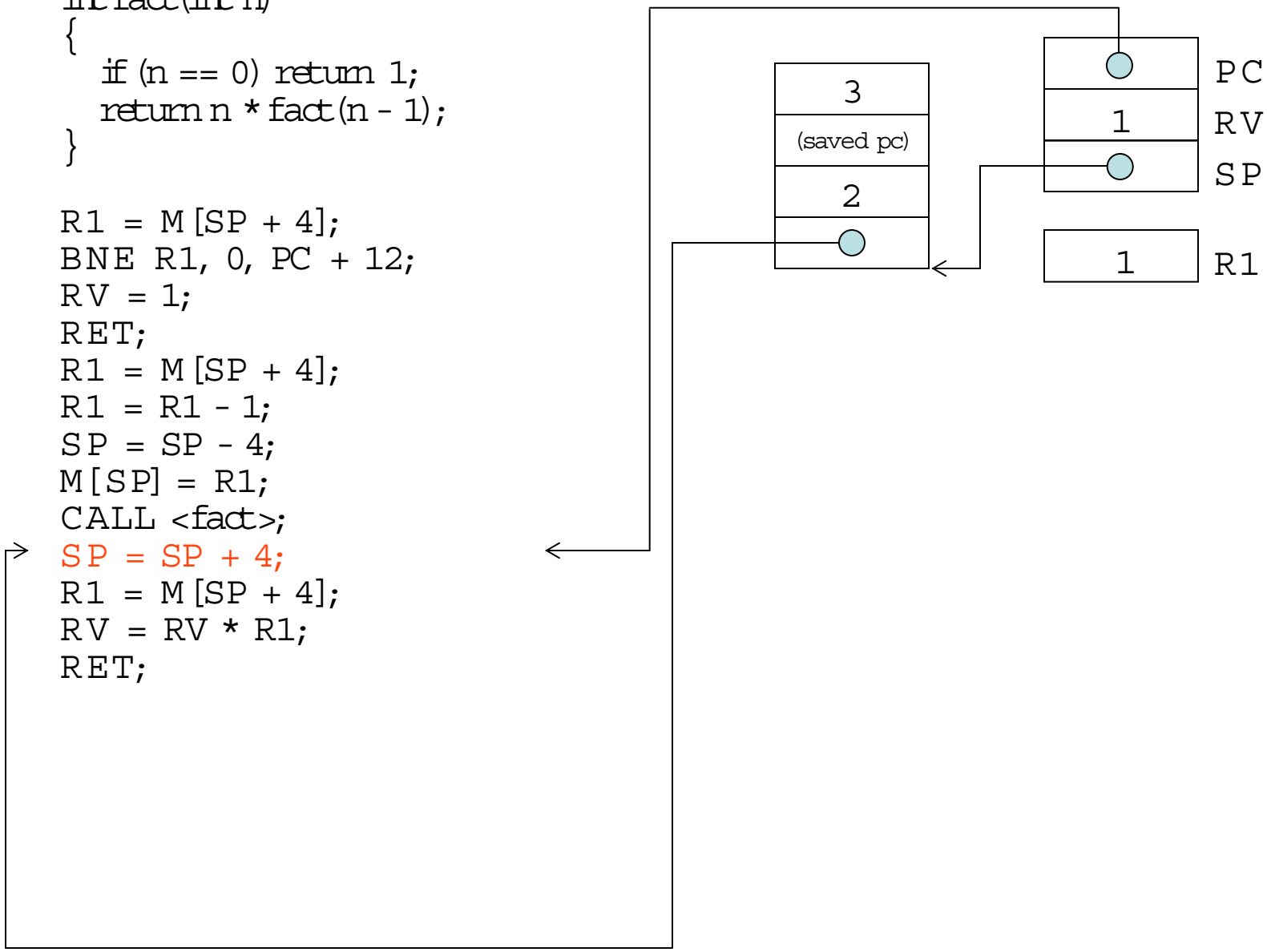
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



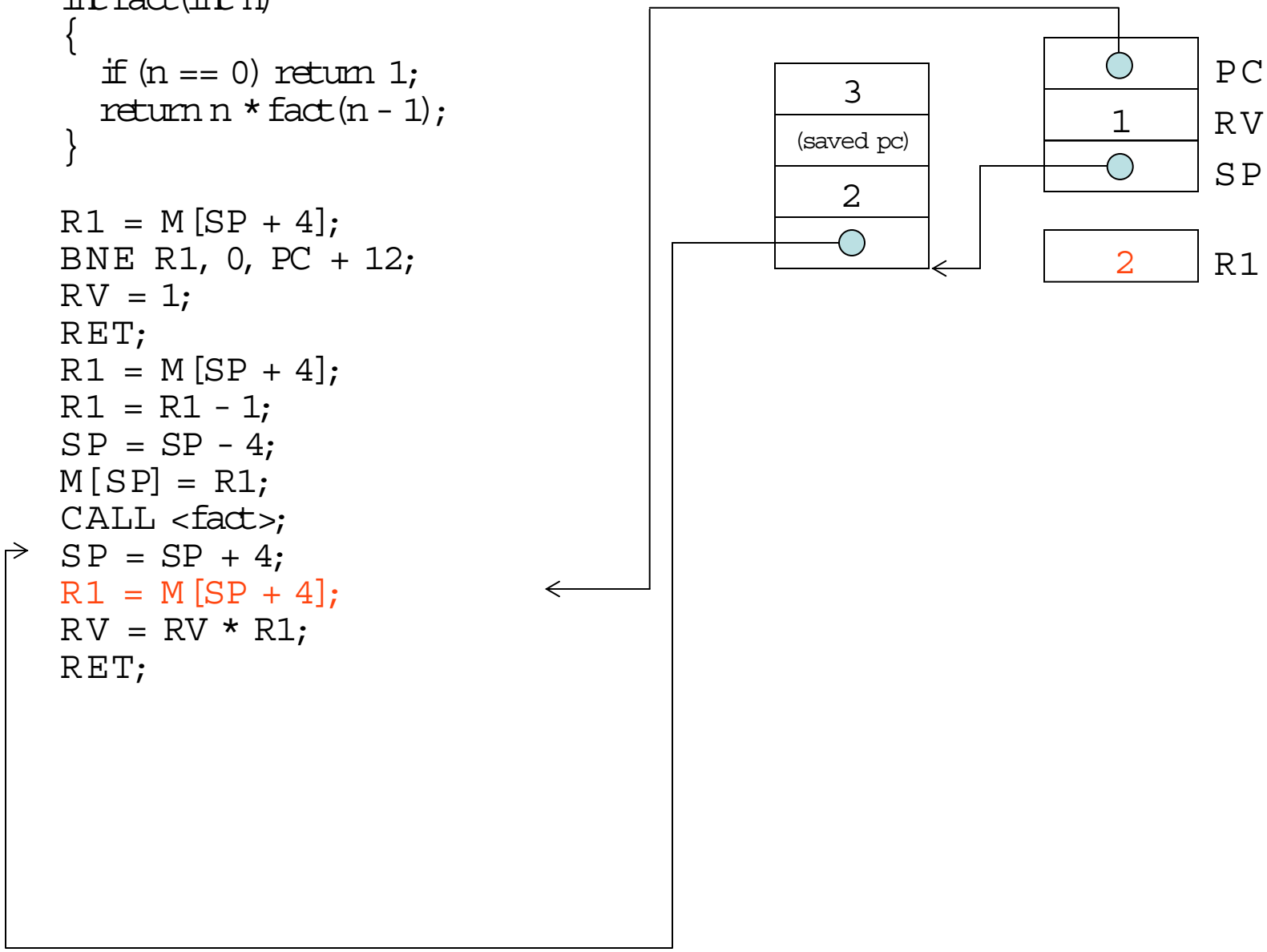
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



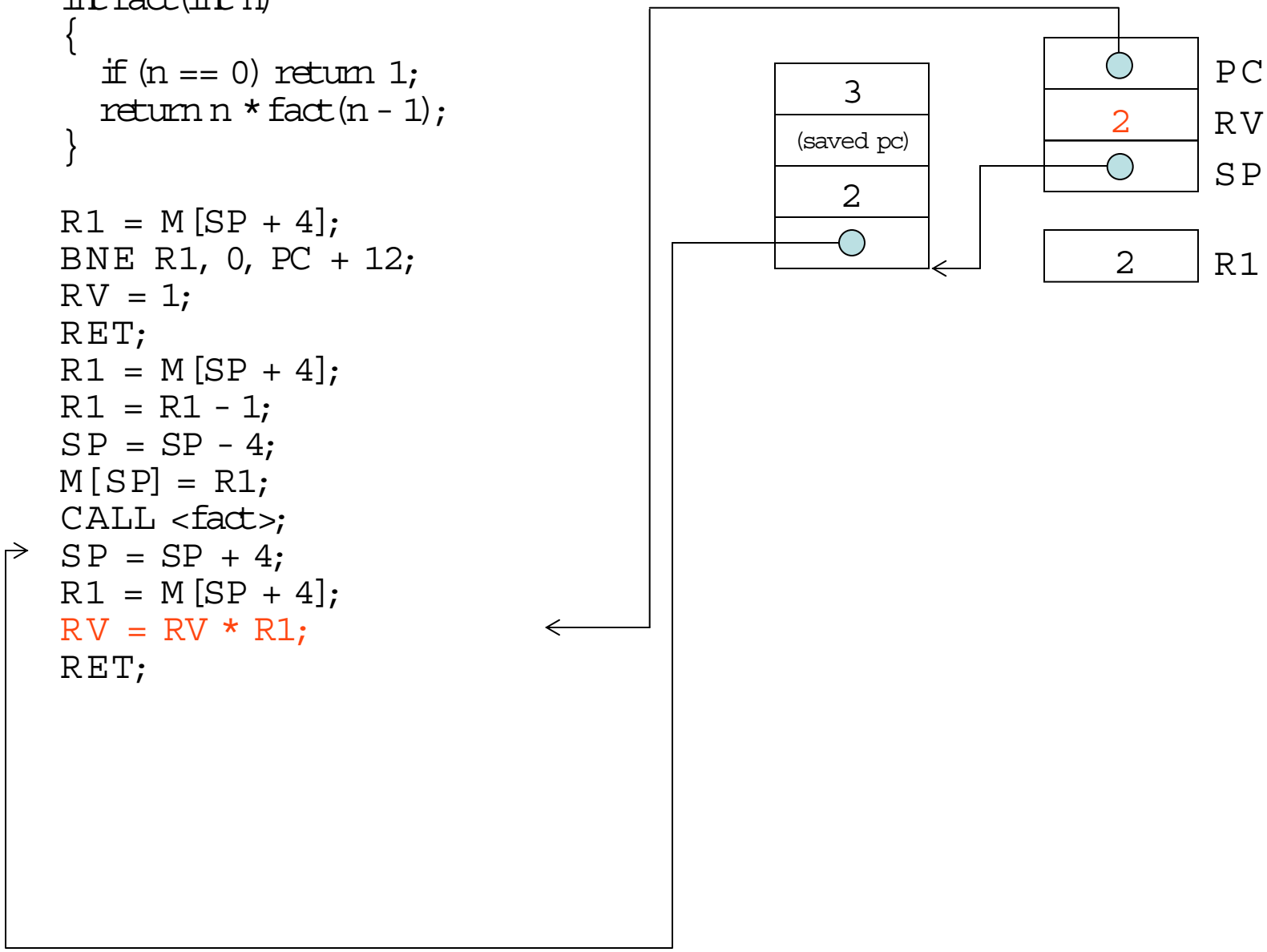
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



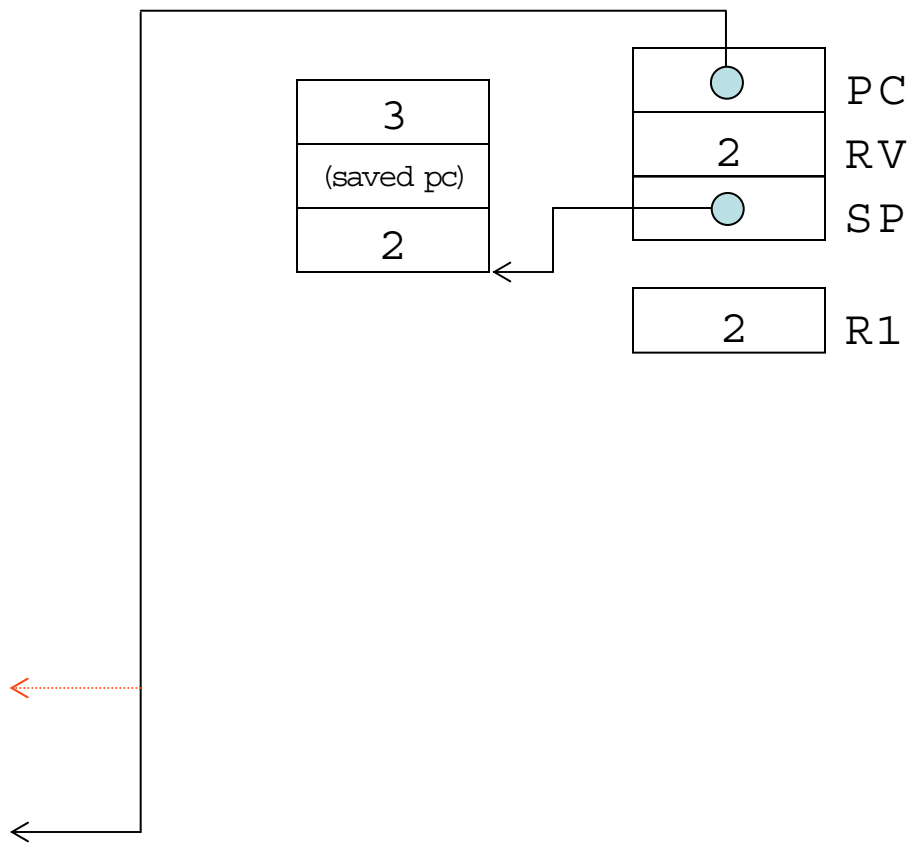
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



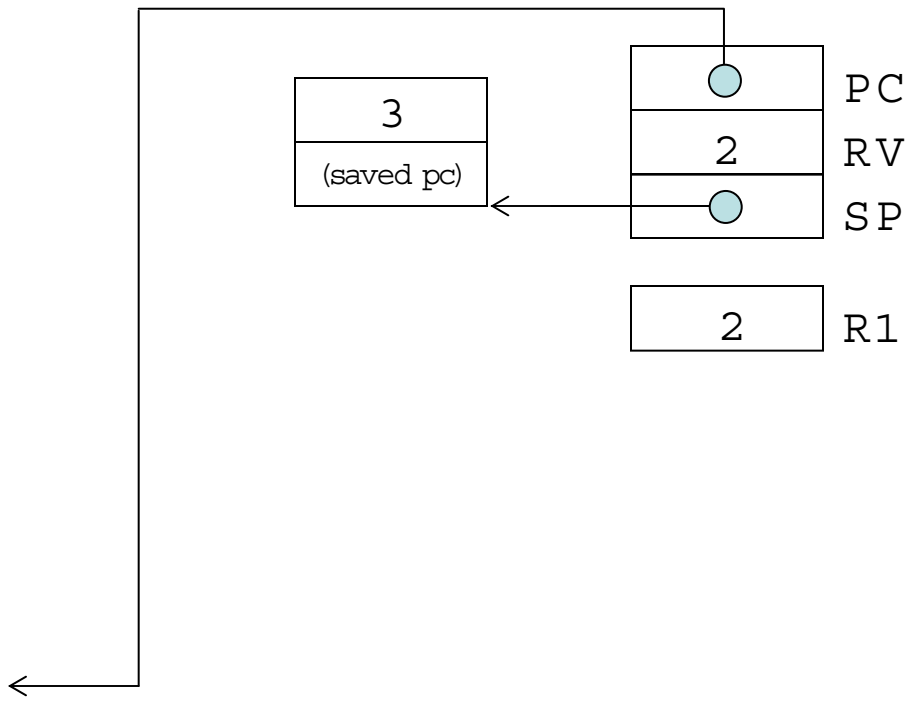
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



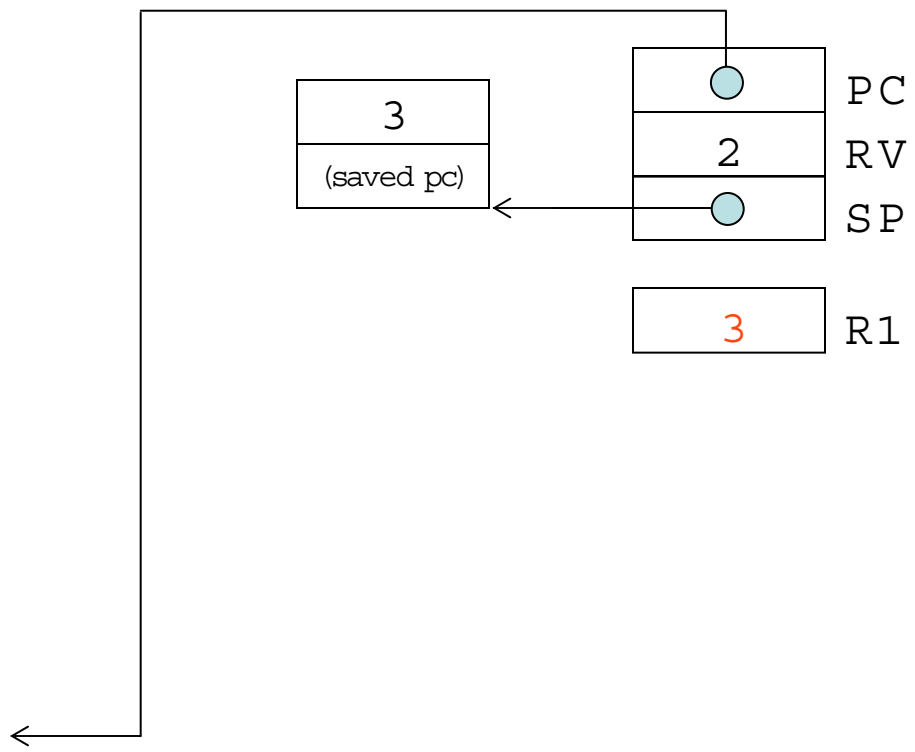
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}
```

```
R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



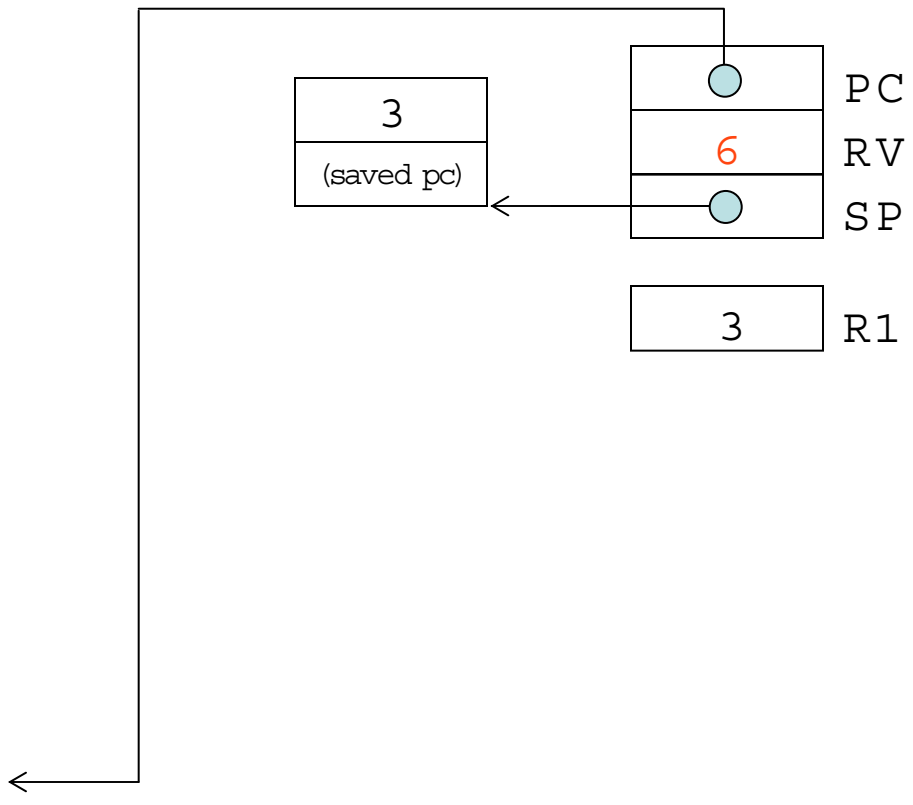
```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}
```

```
R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}

R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```



```
int fact(int n)
{
   if (n == 0) return 1;
   return n * fact(n - 1);
}
```

```
R1 = M[SP + 4];
BNE R1, 0, PC + 12;
RV = 1;
RET;
R1 = M[SP + 4];
R1 = R1 - 1;
SP = SP - 4;
M[SP] = R1;
CALL <fact>;
SP = SP + 4;
R1 = M[SP + 4];
RV = RV * R1;
RET;
```