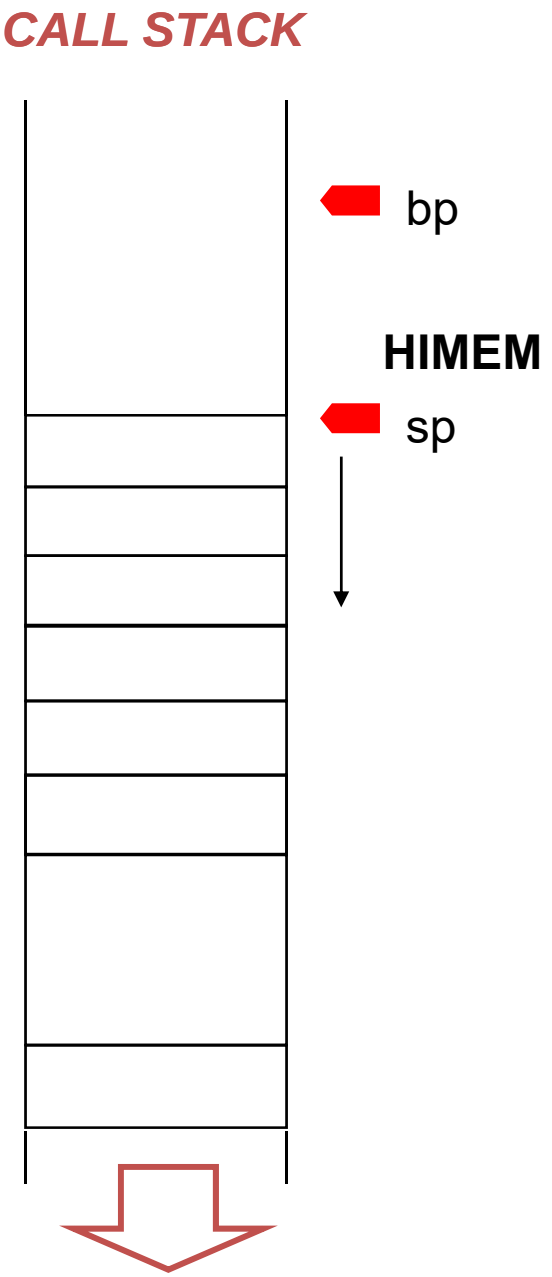
呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```



eax

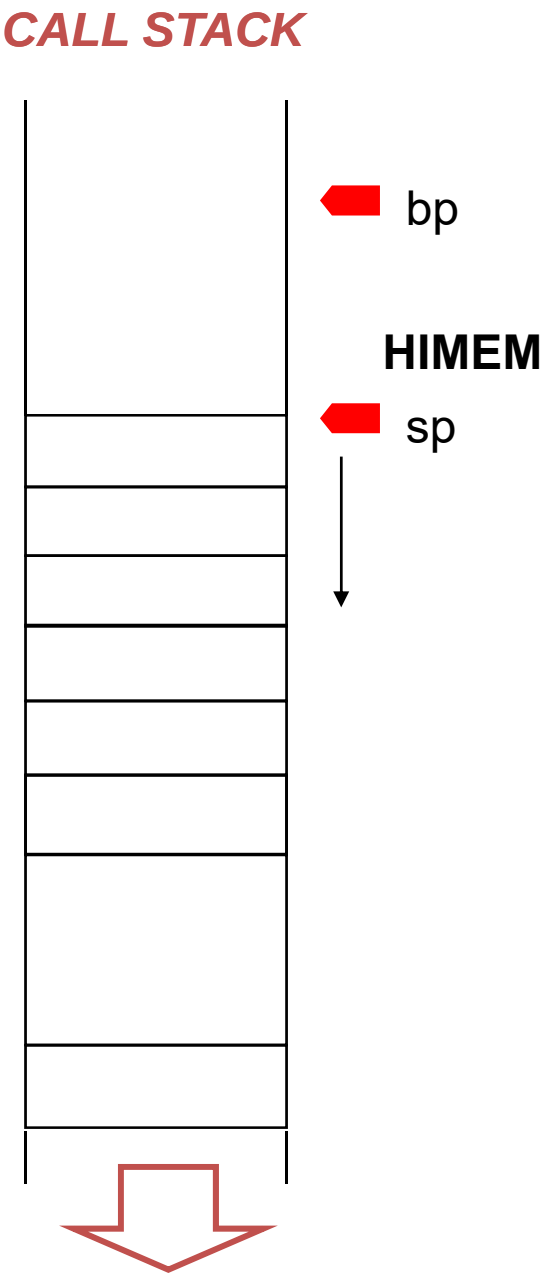
or rax for x64

呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

eax

or rax for x64

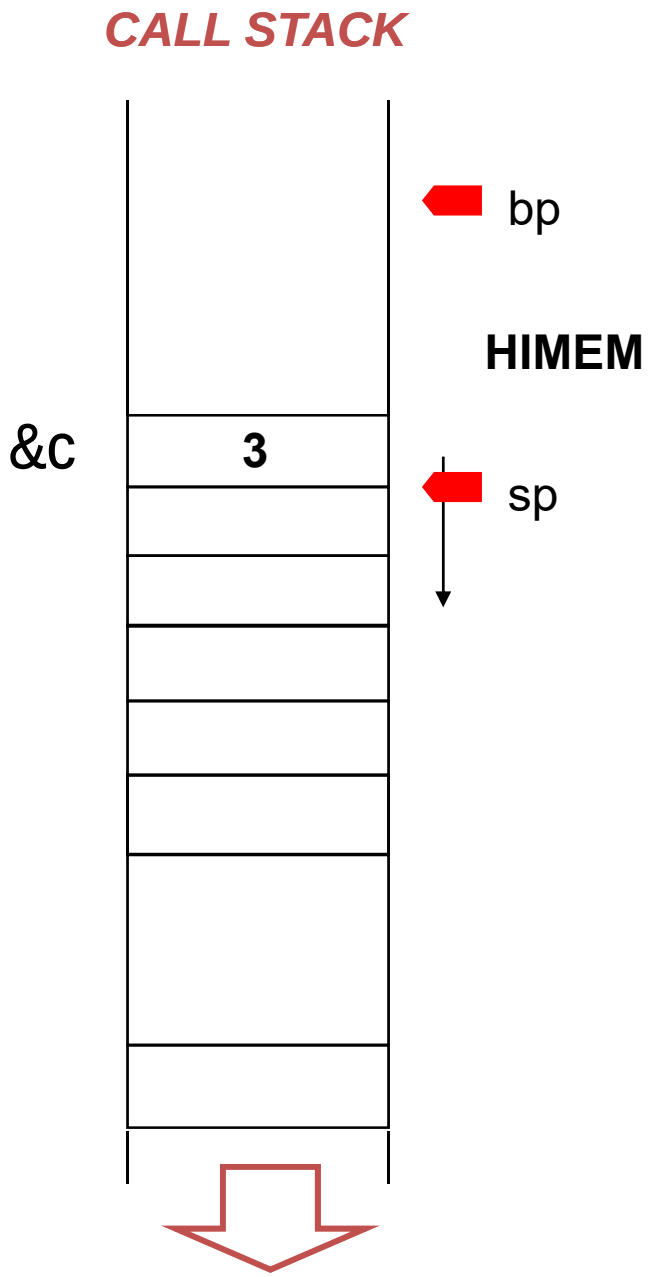


呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

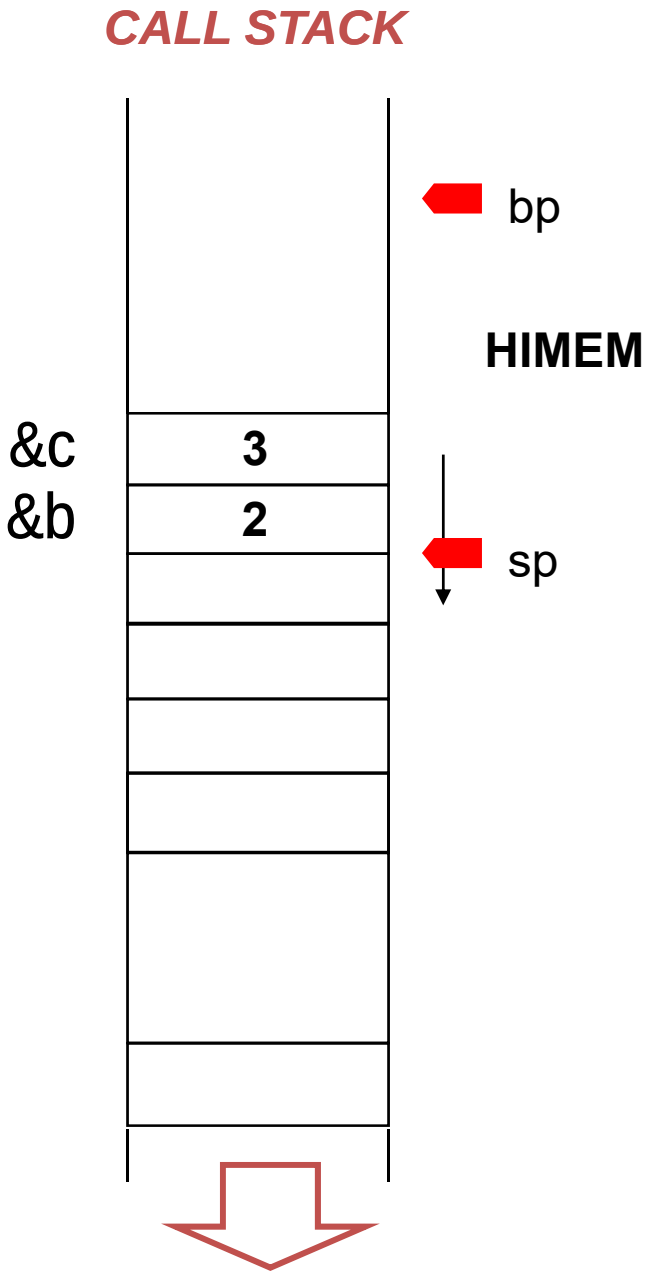
eax

or rax for x64



呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

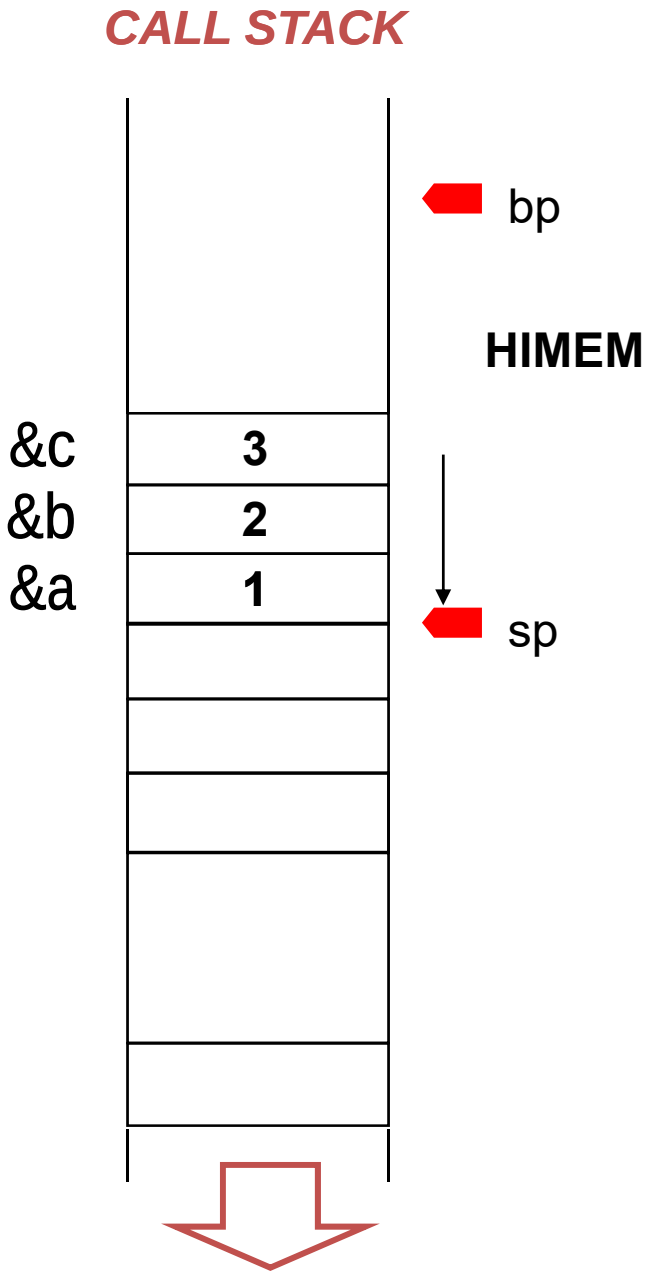


eax

or rax for x64

呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

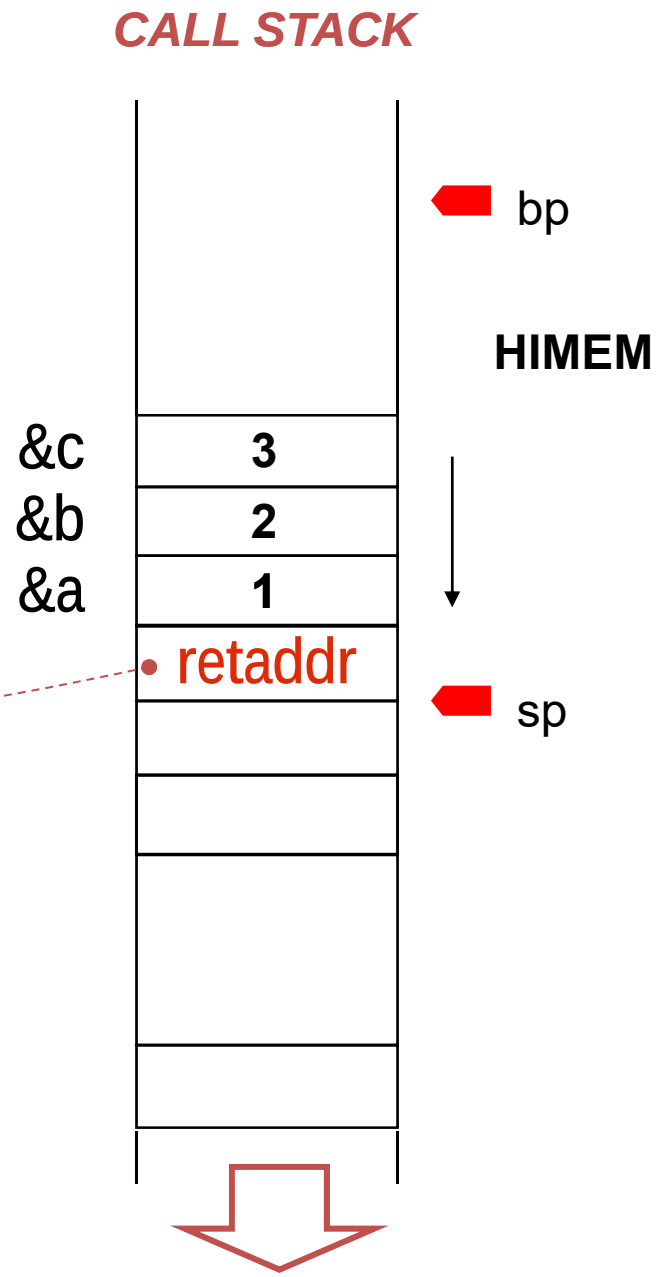


eax

or rax for x64

呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

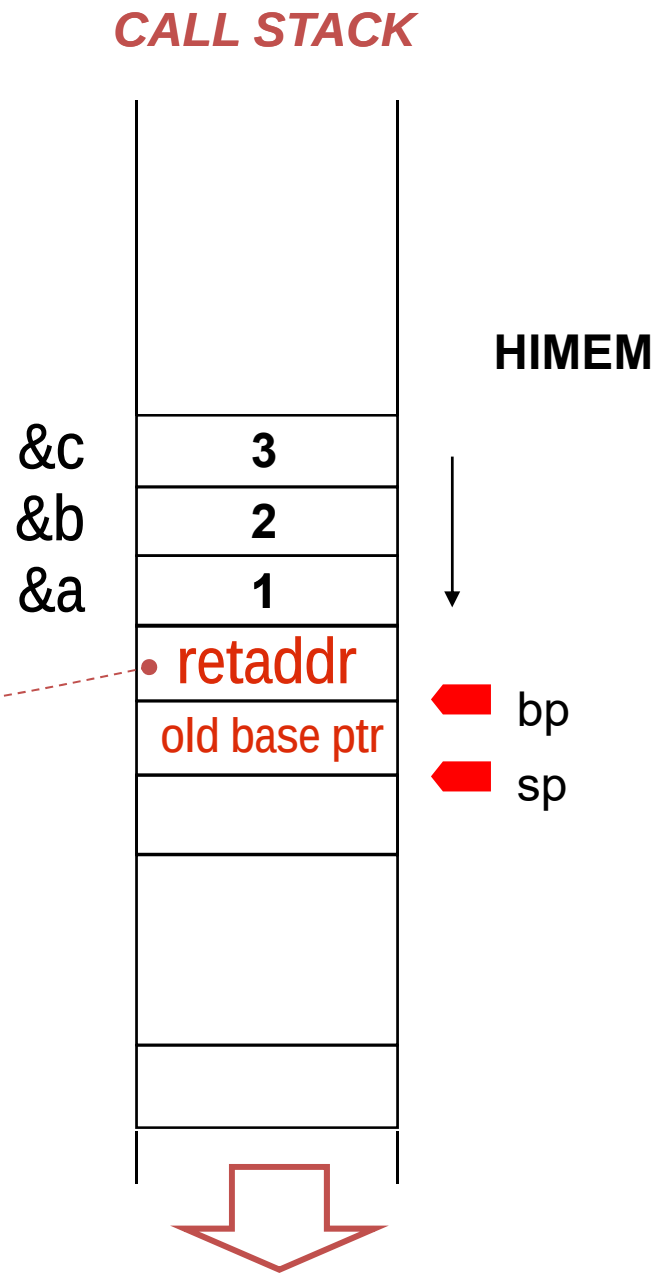


eax

or rax for x64

呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

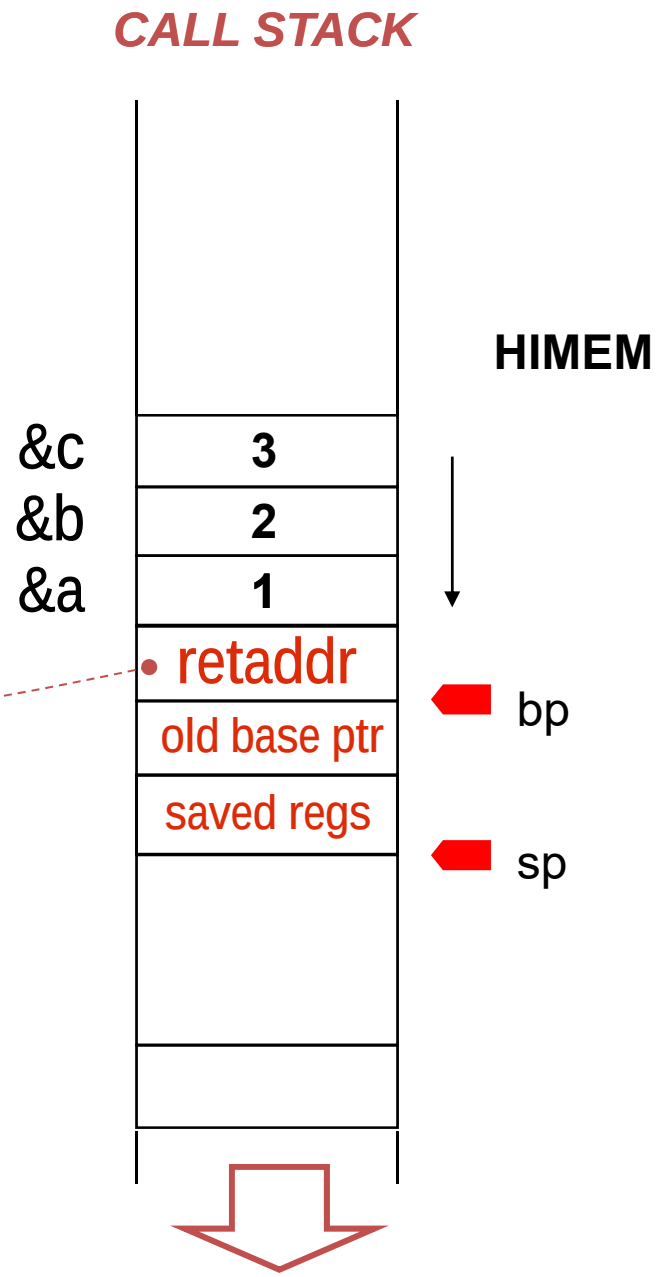


eax

or rax for x64

呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```



eax

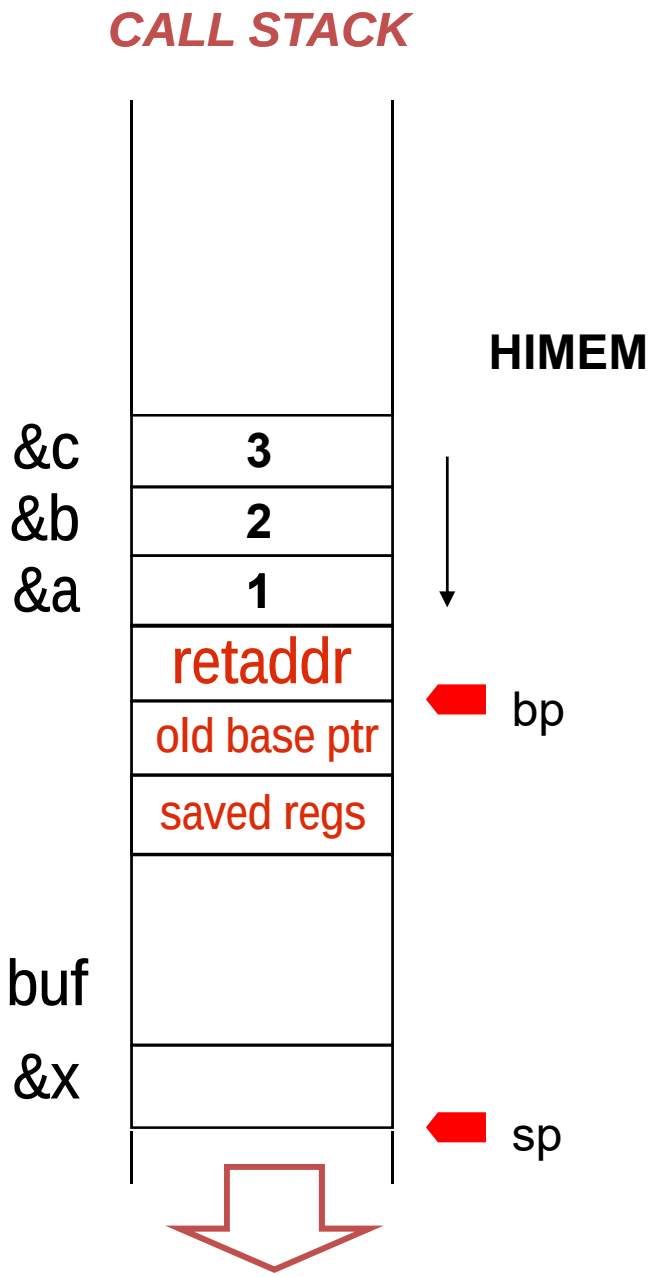
or rax for x64

呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

eax

or rax for x64

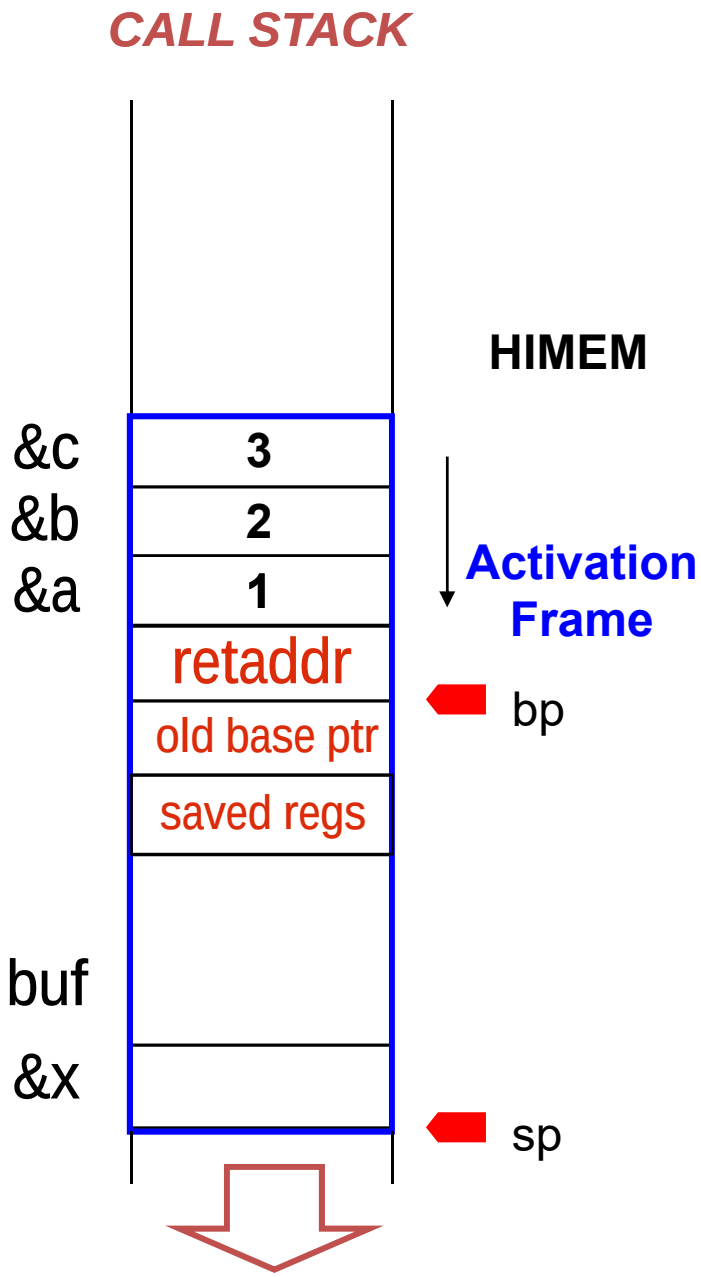


呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

eax

or rax for x64

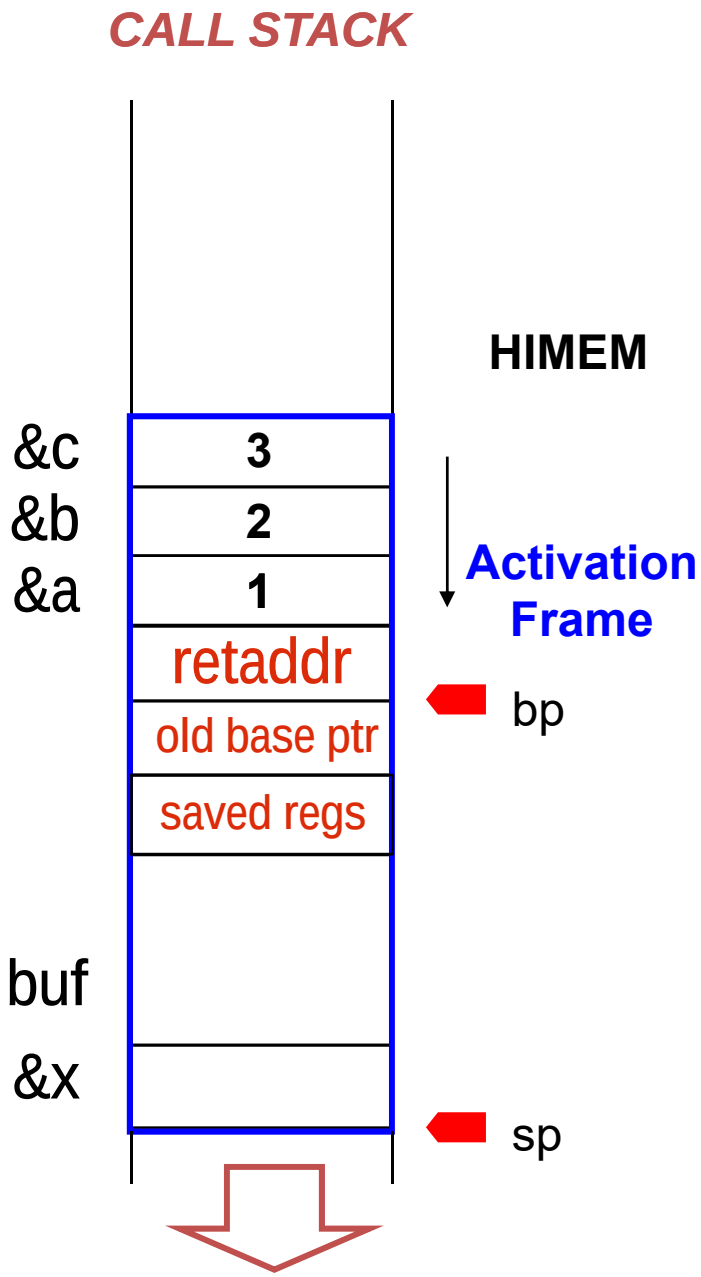


呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

eax **0**

or rax for x64

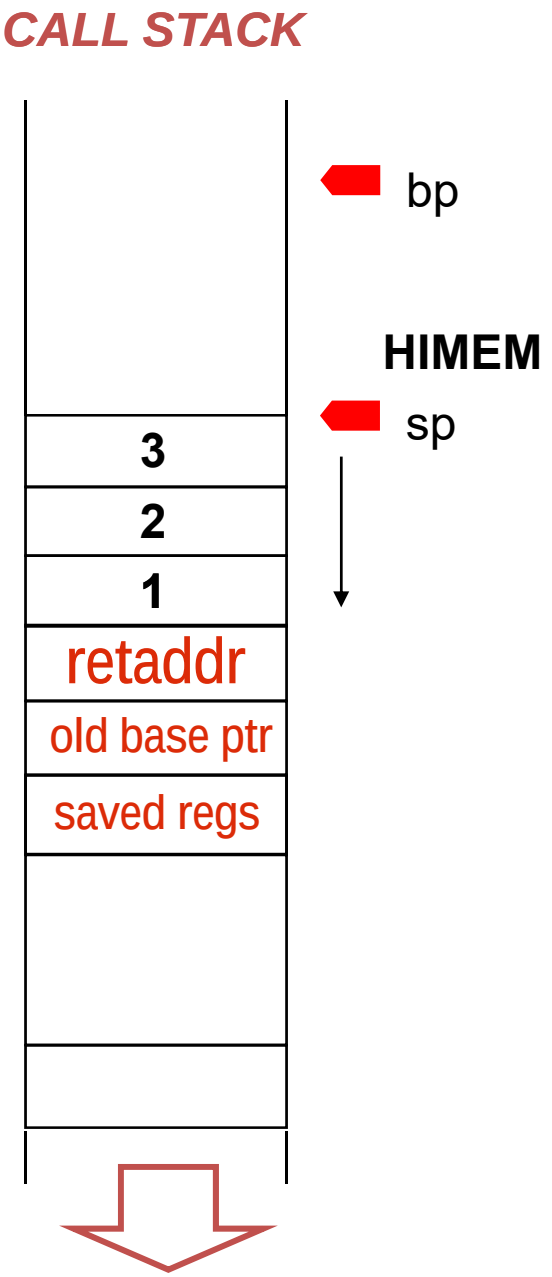


呼叫堆疊
(call stack)

```
int main() {
    int x=1, y=2, z=3, result;
    result = func(x, y, z);
    printf("%d %d %d\n", x, y, z);
}

int func(int a, int b, int c) {
    char buf[20];
    double x;
    …
    return 0;
}
```

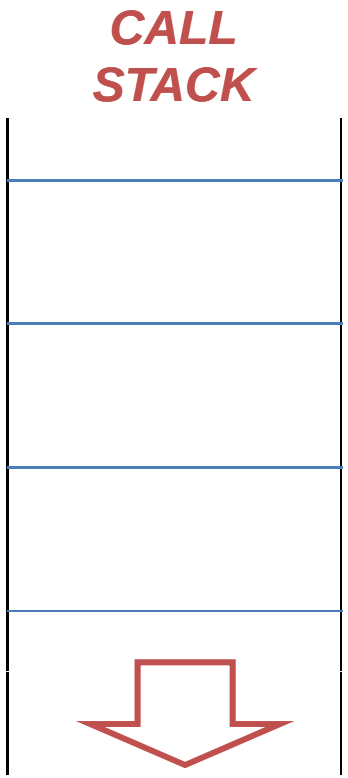
eax | 0

or rax for x64



遞迴函式呼叫

```
x = factorial(3);
printf("%d\n", x);


int factorial(int n) {
    if ((n==1)||(n==0))
        return 1;
    else
        return n*     factorial(n-1);
}
```

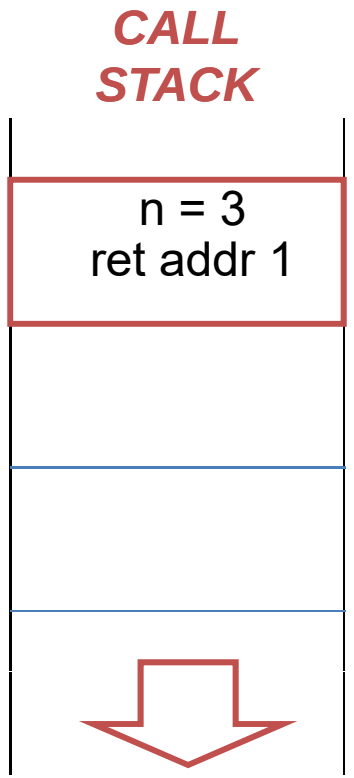


遞迴函式呼叫

factorial(3)

```
x = factorial(3);
printf("%d\n", x);


int factorial(int n) {
    if ((n==1)||(n==0))
        return 1;
    else
        return n*     factorial(n-1);
}
```

ret addr 1



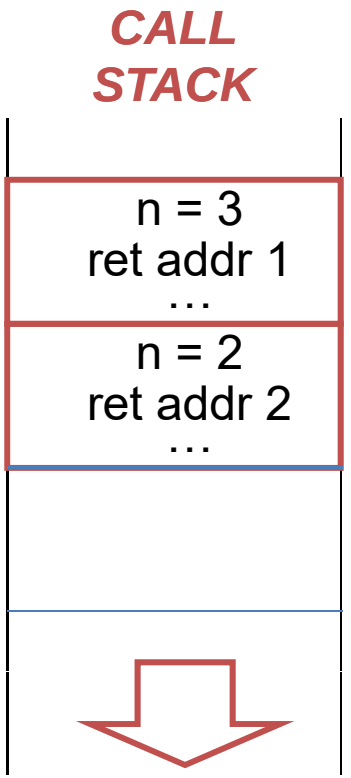
遞迴函式呼叫

factorial(3)

3*factorial(2)

```
x = factorial(3);
printf("%d\n", x);


int factorial(int n) {
    if ((n==1)||(n==0))
        return 1;
    else
        return n*      factorial(n-1);
}
```

ret addr 2



遞迴函式呼叫

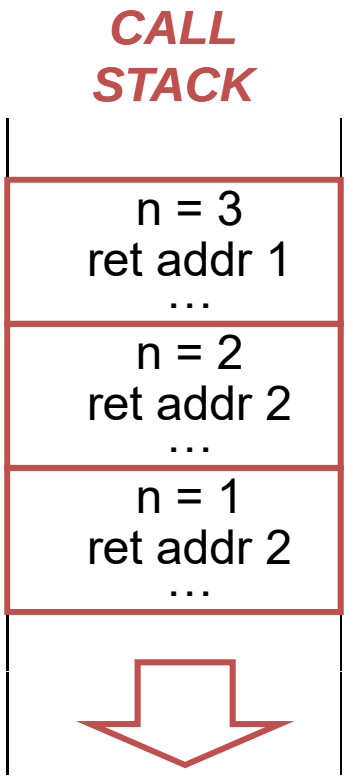
factorial(3)

3*factorial(2)

2*factorial(1)

```
x = factorial(3);
printf("%d\n", x);


int factorial(int n) {
    if ((n==1)||(n==0))
        return 1;
    else
        return n*      factorial(n-1);
}
```

ret addr 2



遞迴函式呼叫

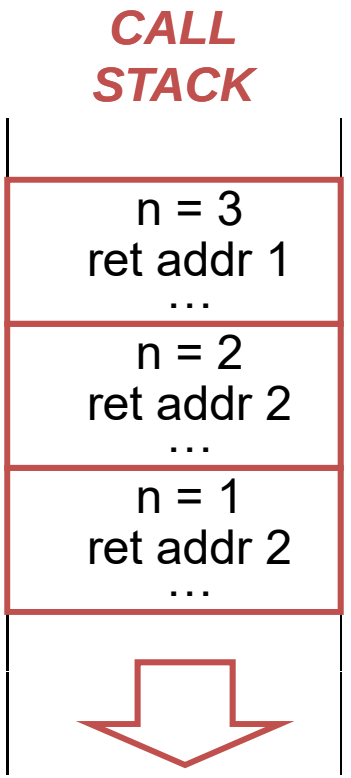
factorial(3)

3*factorial(2)

2*factorial(1)

```
x = factorial(3);
printf("%d\n", x);                    return 1


int factorial(int n) {
    if ((n==1)||(n==0))
        return 1;
    else
        return n*     factorial(n-1);
}
```

ret addr 2



遞迴函式呼叫

factorial(3)

3*factorial(2)

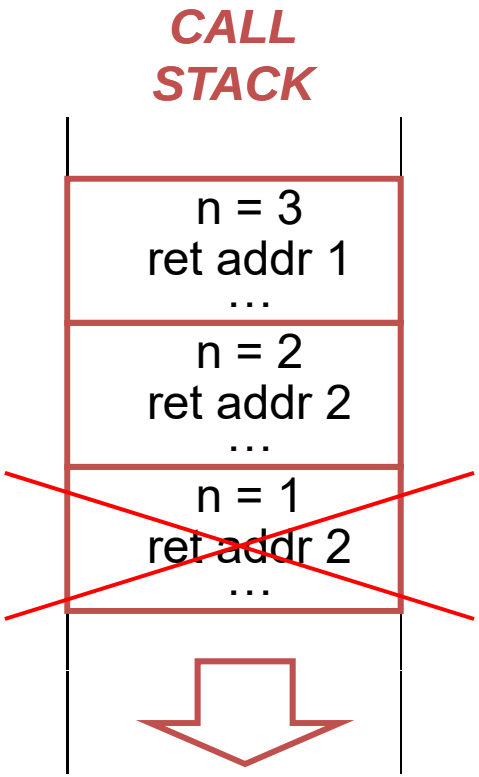
2*factorial(1)

return 2*1

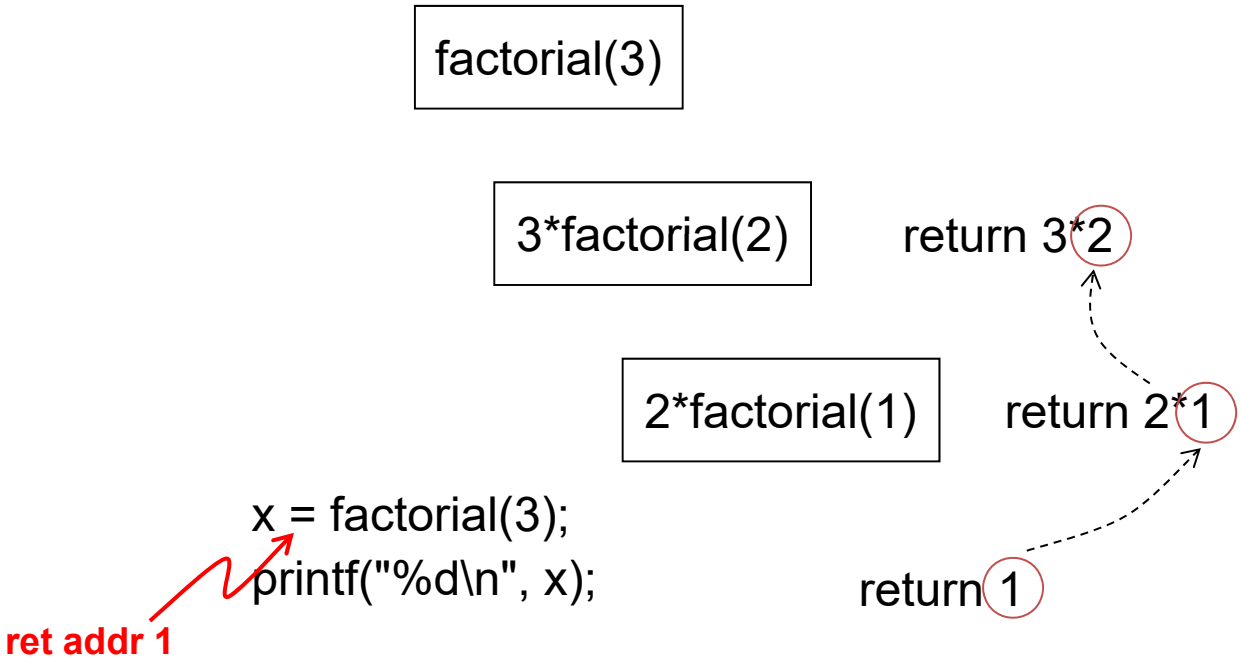
return 1

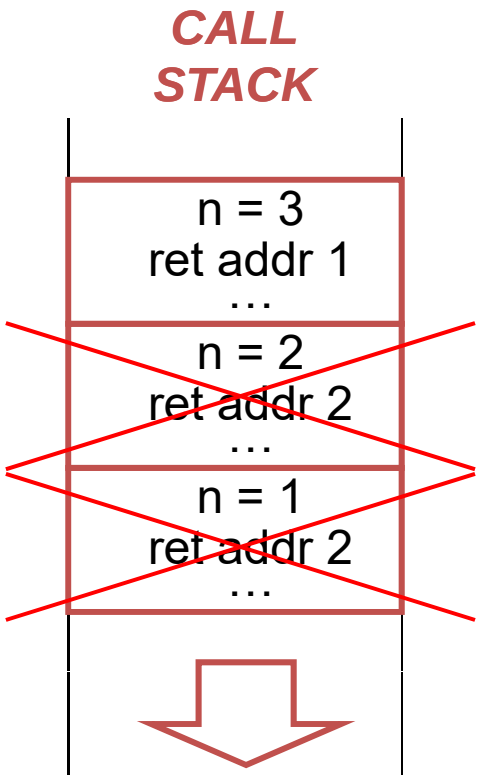
```
x = factorial(3);
printf("%d\n", x);

int factorial(int n) {
    if ((n==1)||(n==0))
        return 1;
    else
        return n*      factorial(n-1);
}
```

ret addr 2

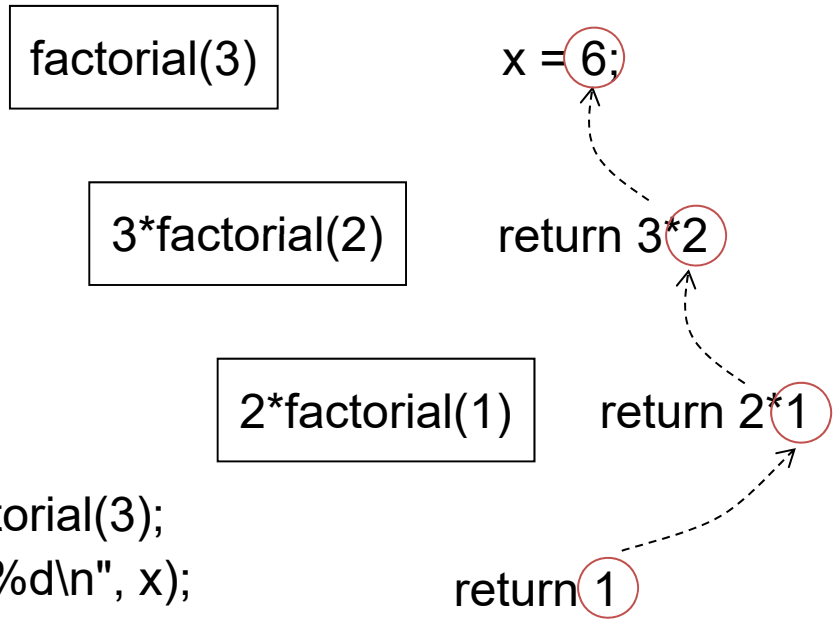


遞迴函式呼叫


factorial(3)
3*factorial(2)
return 3*2
2*factorial(1)
return 2*1
return 1
x = factorial(3);
printf("%d\n", x);
ret addr 1
int factorial(int n) {
if ((n==1)||(n==0))
return 1;
else
return n* factorial(n-1);
}


CALL STACK
n = 3
ret addr 1
…
n = 2
ret addr 2
…
n = 1
ret addr 2
…

遞迴函式呼叫




```
x = factorial(3);
printf("%d\n", x);


int factorial(int n) {
    if ((n==1)||(n==0))
        return 1;
    else
        return n*     factorial(n-1);
}
```