函数内变量的处理

船说：计算机基础

1.6.4 函数内局部变量的底层处理

01 sp(栈指针）的设计可以更方便的使用内存，互不干扰

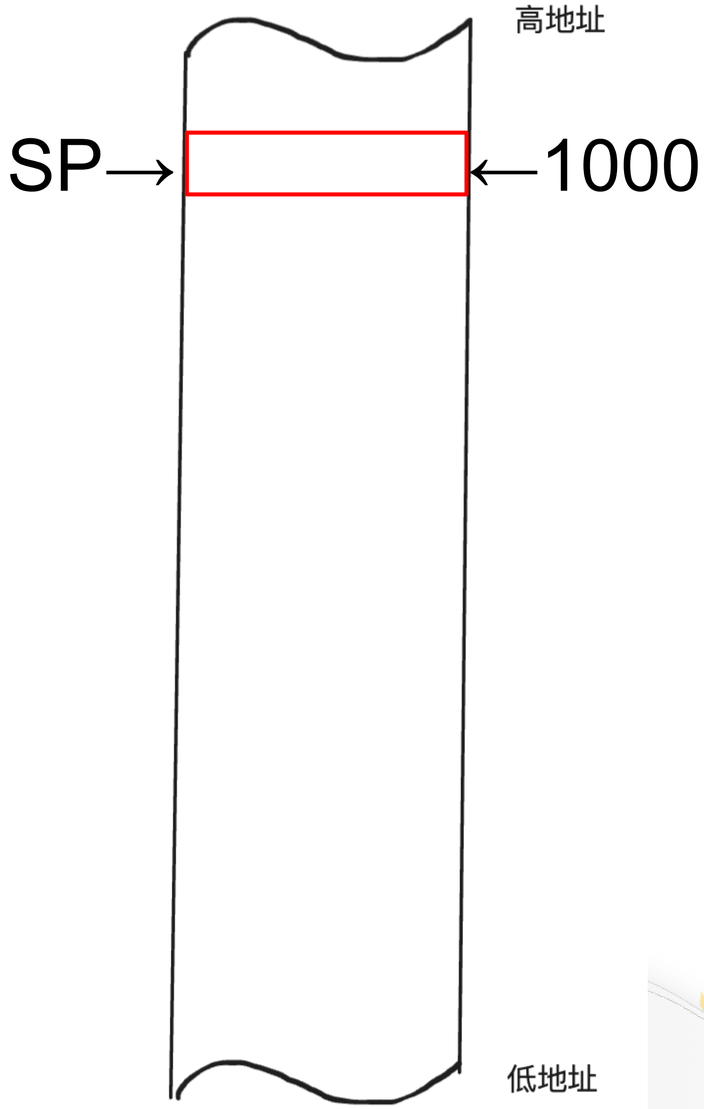
02 栈所分配的空间局部有效，离开后失效

03 全局变量的使用与注意点

1、如何更方便的使用内存

```
1   int FuncA(int x)
2   {
3       return x*x;
4   }
5   int FuncB(int x, int y)
6   {
7       return x + y;
8   }
9
10  int main()
11  {
12      FuncA(10);
13      FuncB(20,30);
14      return 0;
15  }
```

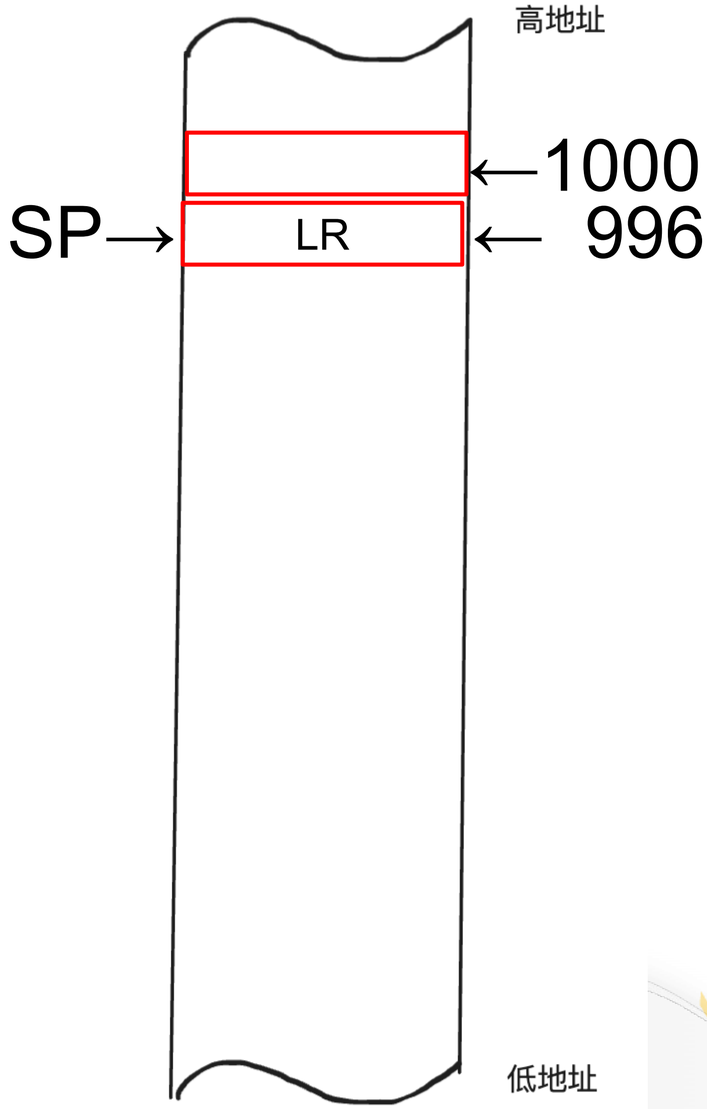
main执行前



1、如何更方便的使用内存

```
1   int FuncA(int x)
2   {
3       return x*x;
4   }
5   int FuncB(int x, int y)
6   {
7       return x + y;
8   }
9
10  int main()
11  {
12      FuncA(10);
13      FuncB(20,30);
14      return 0;
15  }
```

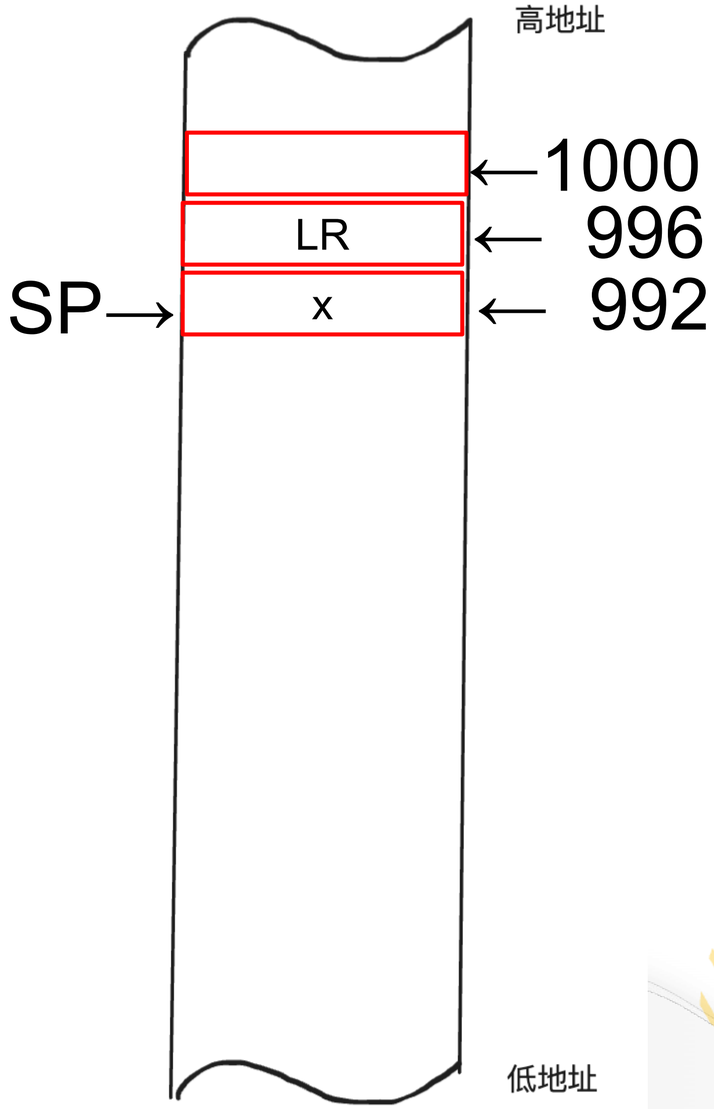
main开始执行



1、如何更方便的使用内存

```
1   int FuncA(int x)
2   {
3       return x*x;
4   }
5   int FuncB(int x, int y)
6   {
7       return x + y;
8   }
9
10  int main()
11  {
12      FuncA(10);
13      FuncB(20,30);
14      return 0;
15  }
```

main开始执行
调用FuncA

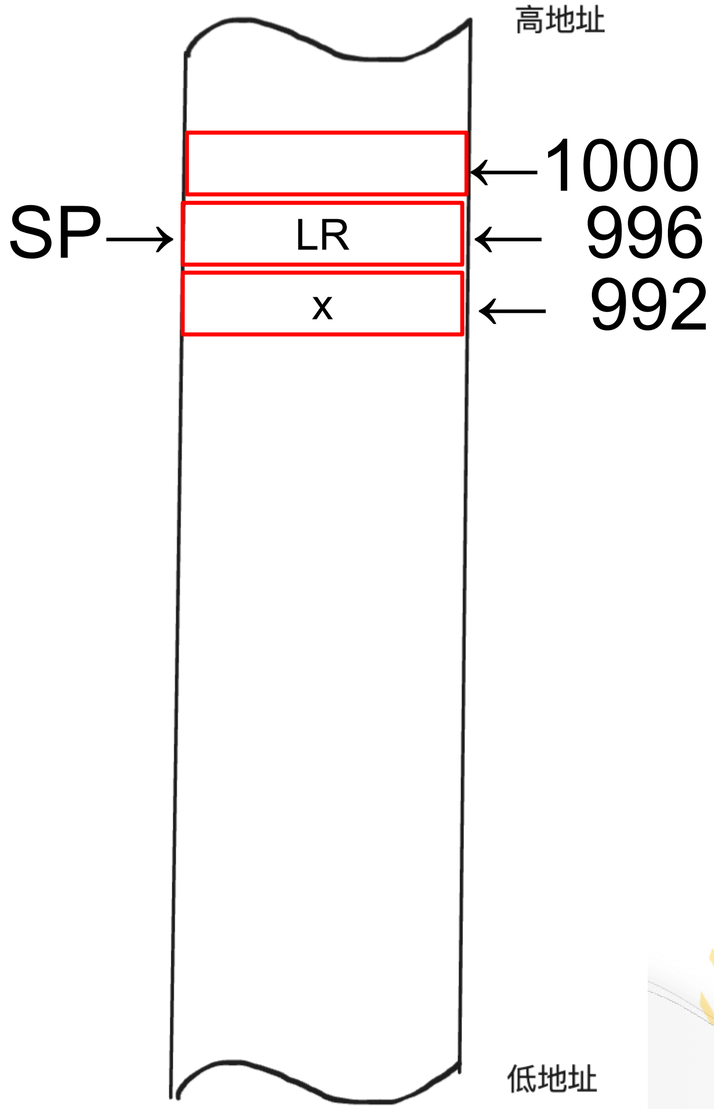


1、如何更方便的使用内存

```
1   int FuncA(int x)
2   {
3       return x*x;
4   }
5   int FuncB(int x, int y)
6   {
7       return x + y;
8   }
9
10  int main()
11  {
12      FuncA(10);
13      FuncB(20,30);
14      return 0;
15  }
```

main开始执行

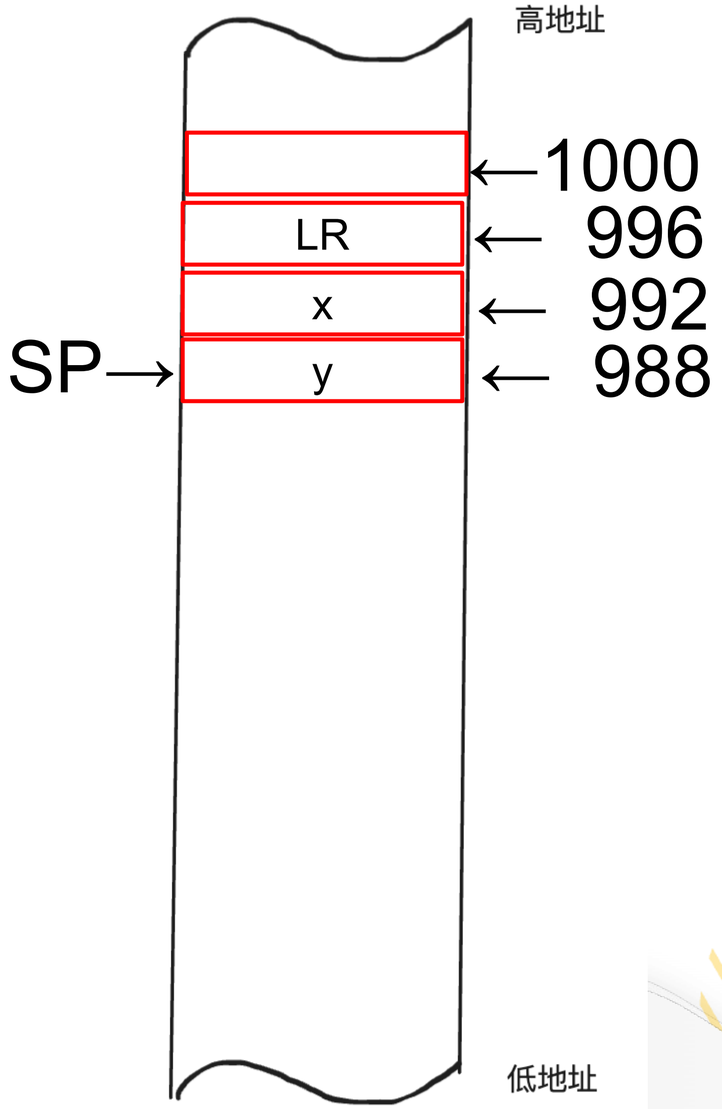
从FuncA返回



1、如何更方便的使用内存

```
1   int FuncA(int x)
2   {
3       return x*x;
4   }
5   int FuncB(int x, int y)
6   {
7       return x + y;
8   }
9
10  int main()
11  {
12      FuncA(10);
13      FuncB(20,30);
14      return 0;
15  }
```

main开始执行
调用FuncB

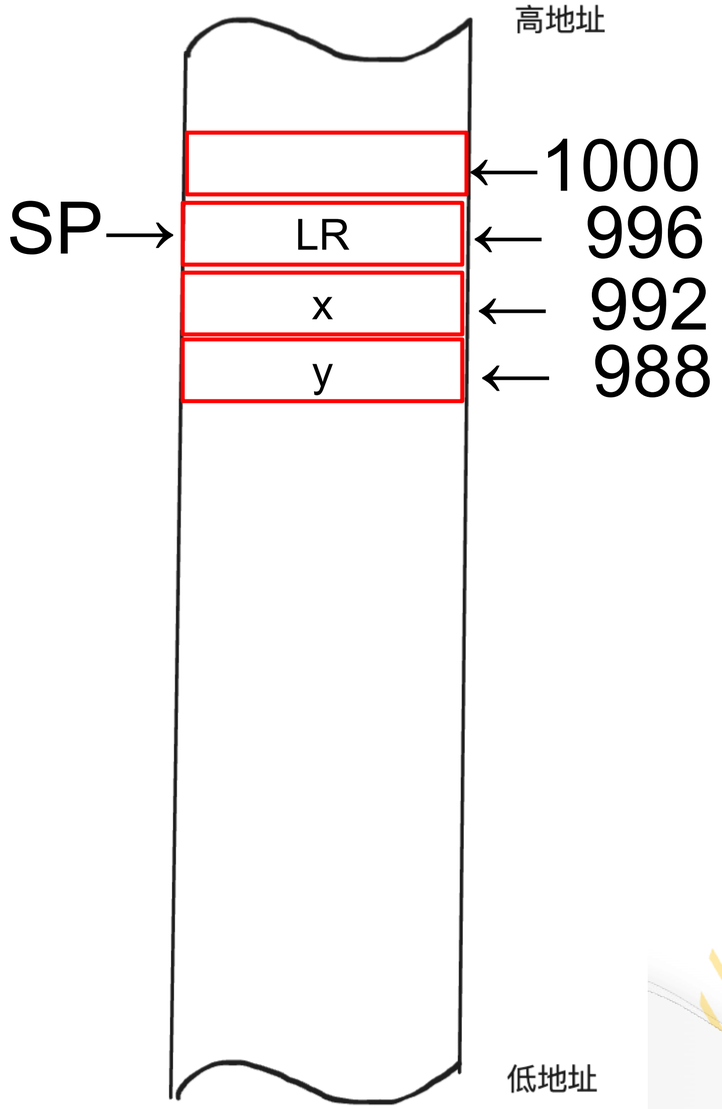


1、如何更方便的使用内存

```
1   int FuncA(int x)
2   {
3       return x*x;
4   }
5   int FuncB(int x, int y)
6   {
7       return x + y;
8   }
9
10  int main()
11  {
12      FuncA(10);
13      FuncB(20,30);
14      return 0;
15  }
```

main开始执行

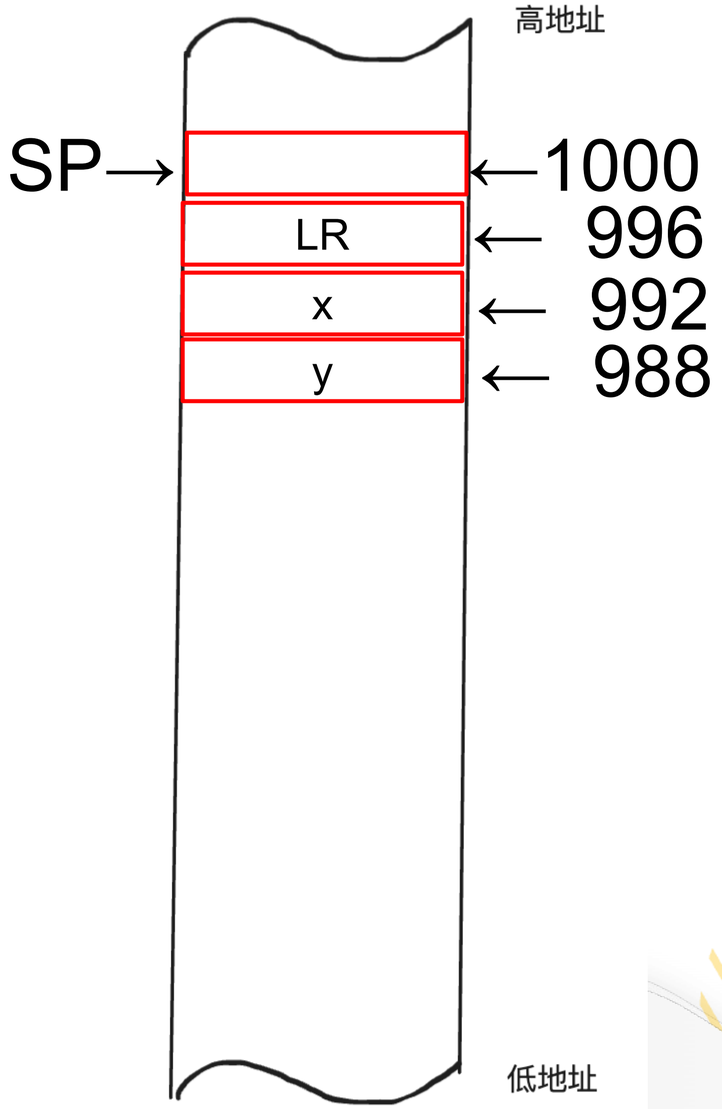
从FuncB返回



1、如何更方便的使用内存

```
1   int FuncA(int x)
2   {
3       return x*x;
4   }
5   int FuncB(int x, int y)
6   {
7       return x + y;
8   }
9
10  int main()
11  {
12      FuncA(10);
13      FuncB(20,30);
14      return 0;
15  }
```

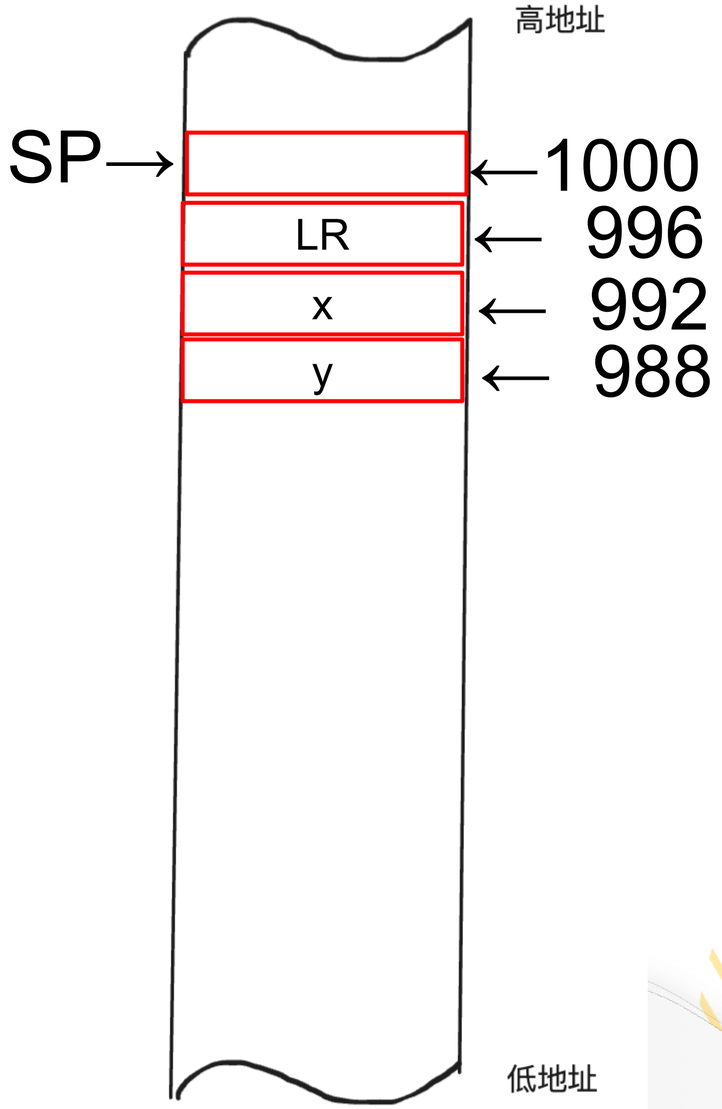
main返回



1、如何更方便的使用内存

```
1
2   int FuncA(int x)
3   {
4       int arr[3];
5       return x*x;
6   }
7   int FuncB(int x, int y)
8   {
9       return x + y;
10  }
11
12  int main()
13  {
14      FuncA(10);
15      FuncB(20,30);
16      return 0;
17  }
```

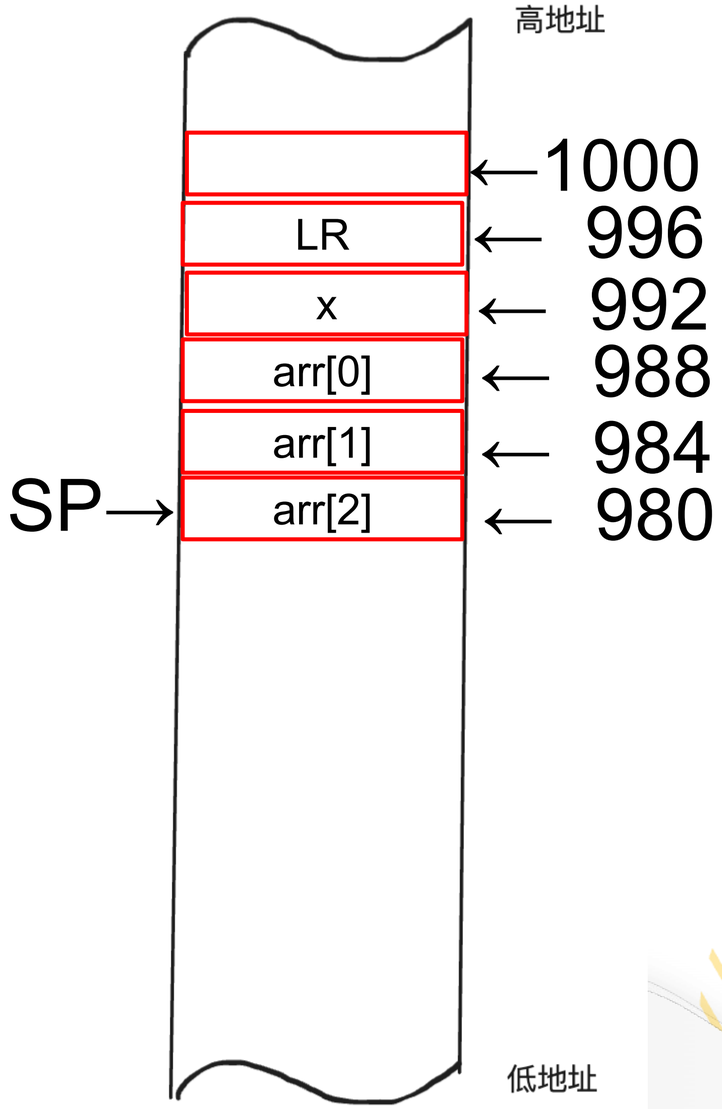
调用FuncA



1、如何更方便的使用内存

```
1
2   int FuncA(int x)
3   {
4       int arr[3];
5       return x*x;
6   }
7   int FuncB(int x, int y)
8   {
9       return x + y;
10  }
11
12  int main()
13  {
14      FuncA(10);
15      FuncB(20,30);
16      return 0;
17  }
```

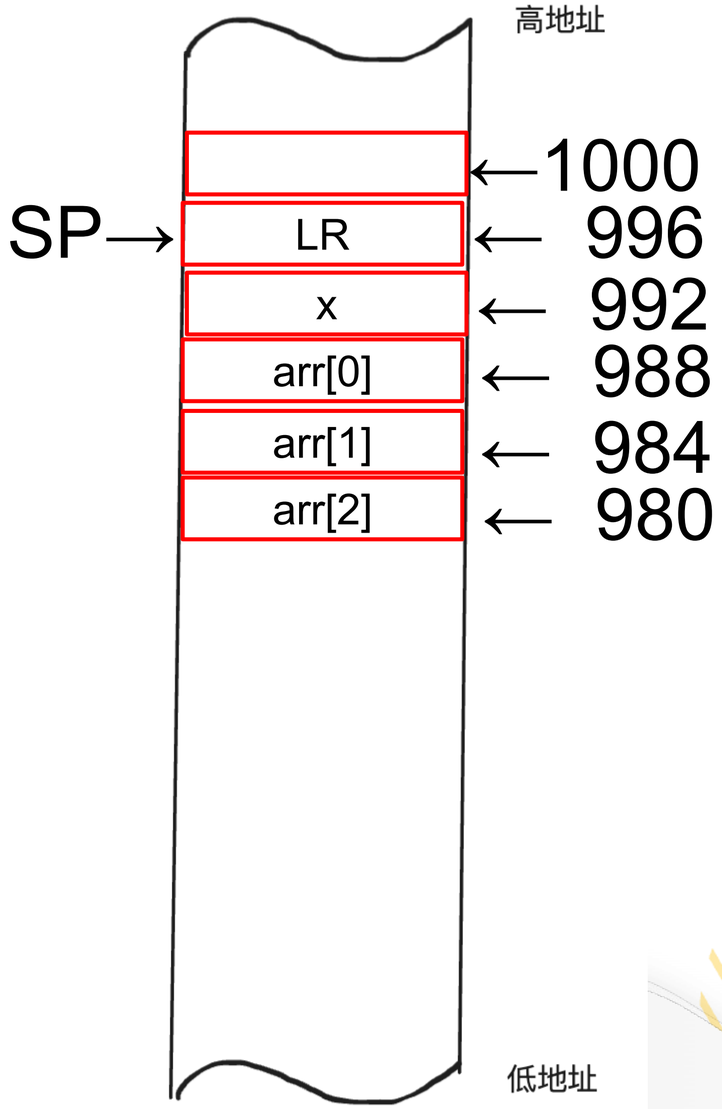
调用FuncA



1、如何更方便的使用内存

```
1
2   int FuncA(int x)
3   {
4       int arr[3];
5       return x*x;
6   }
7   int FuncB(int x, int y)
8   {
9       return x + y;
10  }
11
12  int main()
13  {
14      FuncA(10);
15      FuncB(20,30);
16      return 0;
17  }
```

FuncA返回



2、函数内的常规操作流程

01 如果需要，保存必要的寄存器内容

02 操作sp，分配需要的内容空间

03 运行函数相关功能

04 释放分配的空间

05 恢复保存过的寄存器的值，并使程序返回

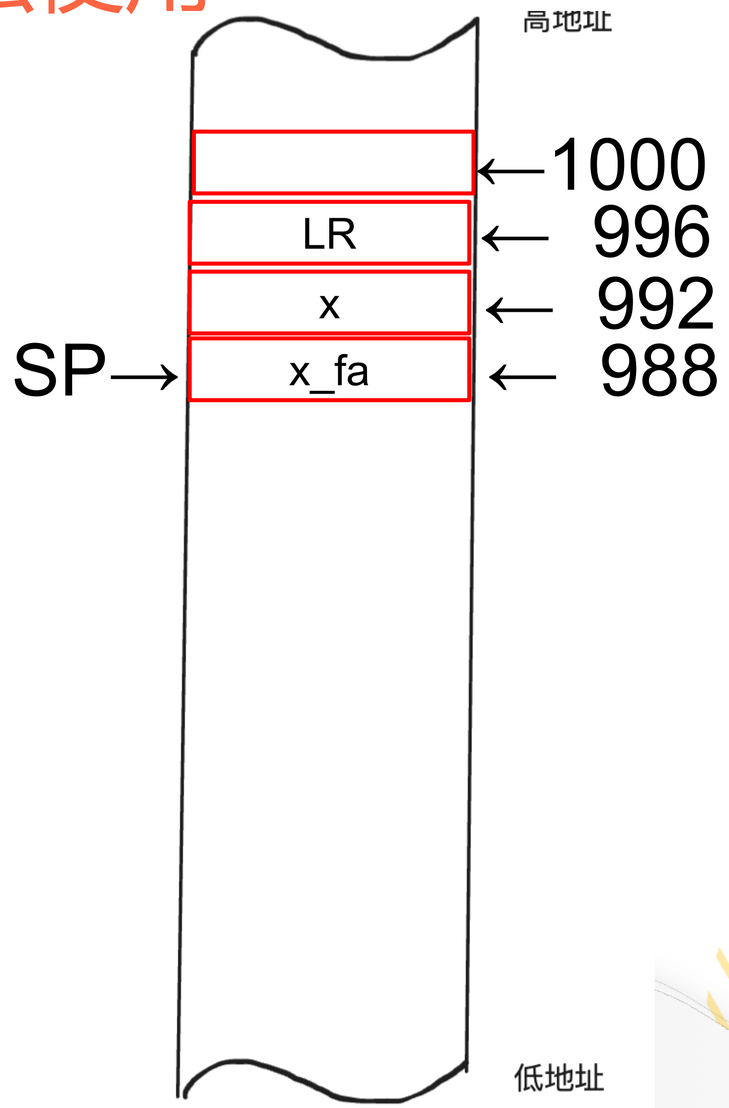
3、栈所分配的空间局部有效

01 不可以在退出后继续使用，对应的存储空间后续会有别的函数使用

4、局部变量在函数返回后无法使用

```
1   int FuncA(int x)
2   {
3       int x_fa = 100;
4       return x*x;
5   }
6   int FuncB(int x, int y)
7   {
8       x_fa = 200;
9       return x + y;
10  }
11
12  int main()
13  {
14      FuncA(10);
15      FuncB(20,30);
16      return 0;
17  }
```

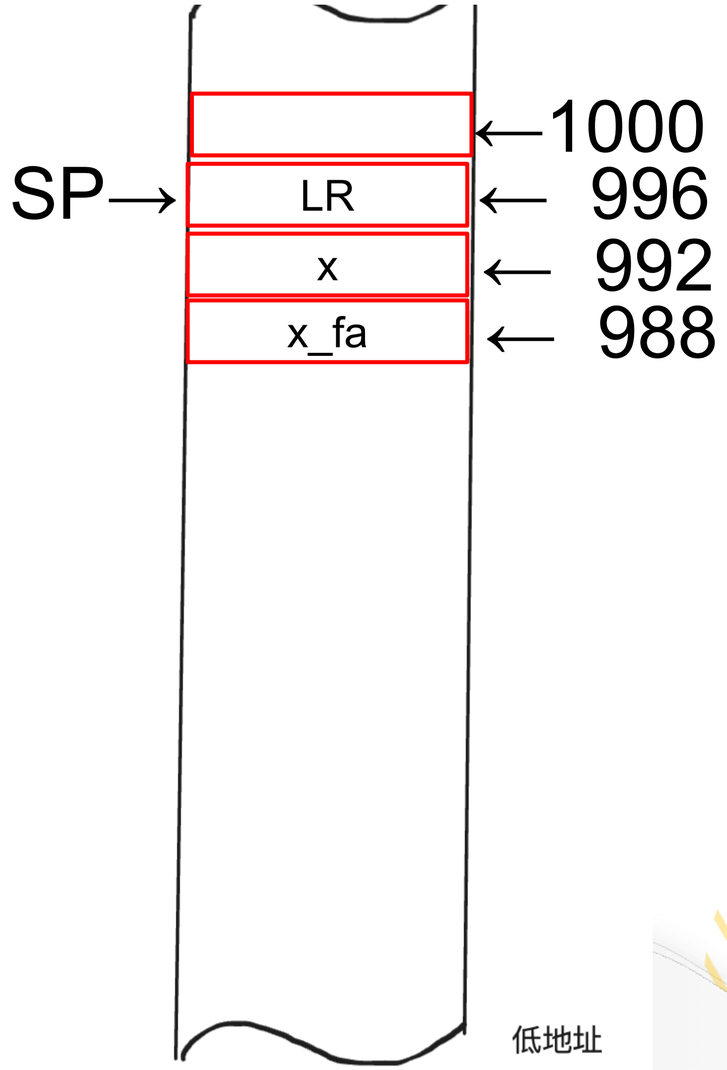
调用FuncA



4、局部变量在函数返回后无法使用

```
1   int FuncA(int x)
2   {
3       int x_fa = 100;
4       return x*x;
5   }
6   int FuncB(int x, int y)
7   {
8       x_fa = 200;
9       return x + y;
10  }
11
12  int main()
13  {
14      FuncA(10);
15      FuncB(20,30);
16      return 0;
17  }
```

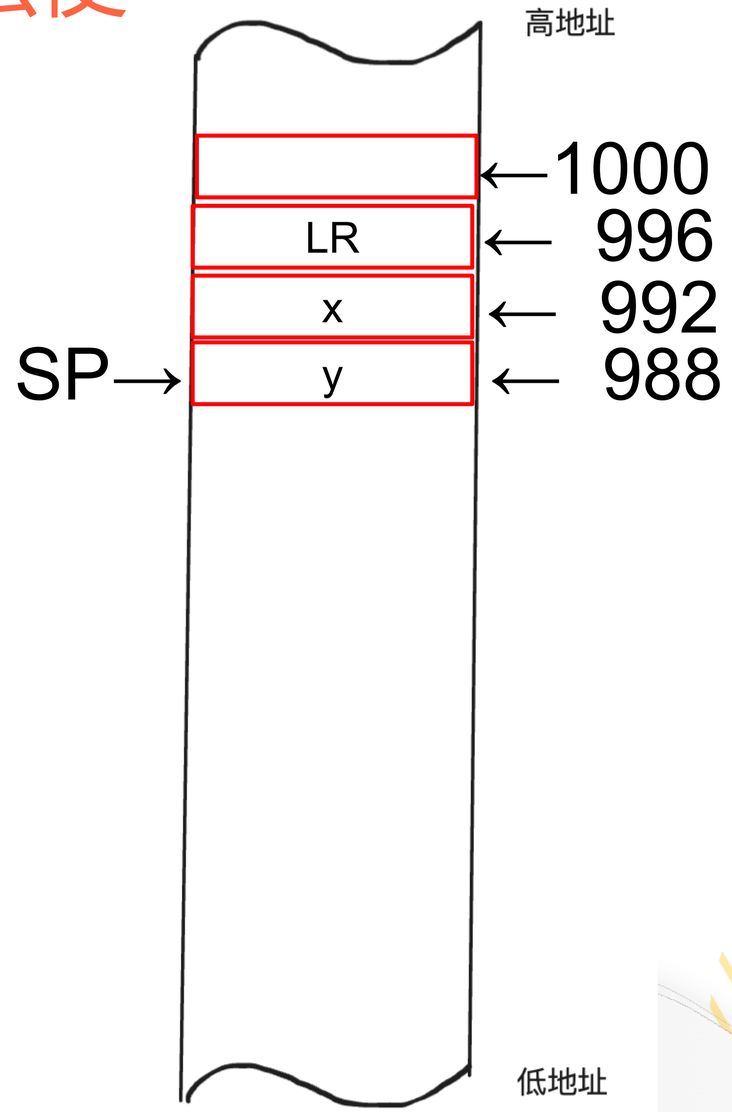
FuncA返回



4、局部变量在函数返回后无法使用

```
1   int FuncA(int x)
2   {
3       int x_fa = 100;
4       return x*x;
5   }
6   int FuncB(int x, int y)
7   {
8       x_fa = 200;
9       return x + y;
10  }
11
12  int main()
13  {
14      FuncA(10);
15      FuncB(20,30);
16      return 0;
17  }
```

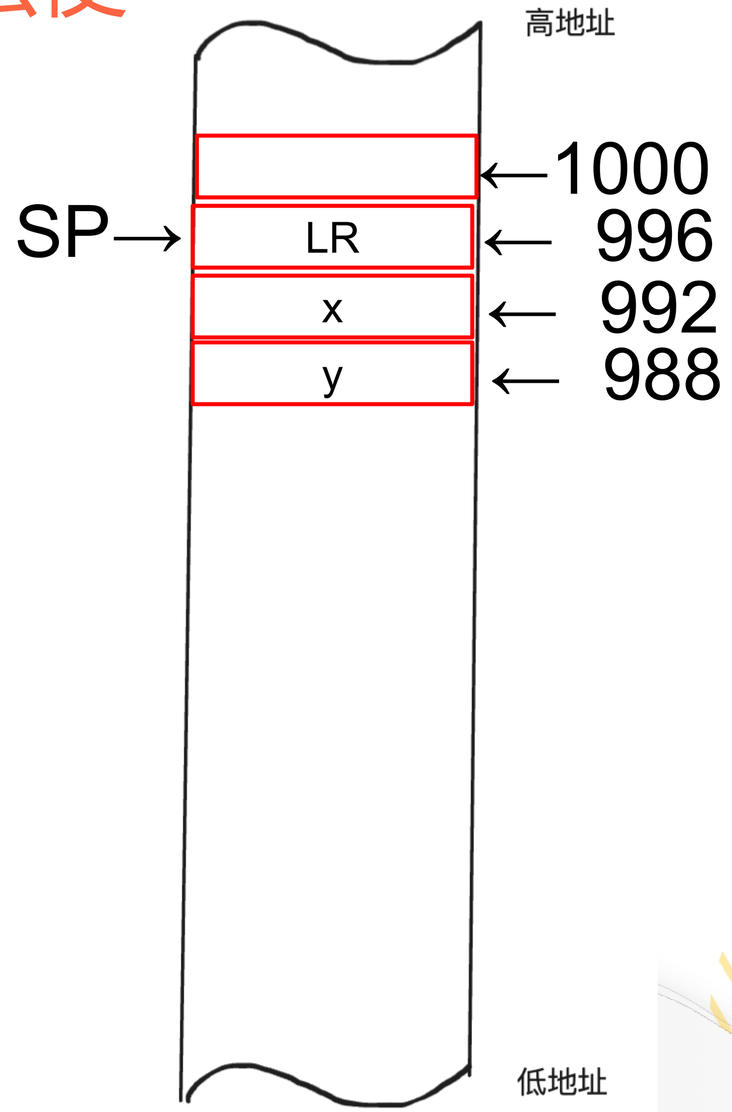
调用FuncB



4、局部变量在函数返回后无法使用

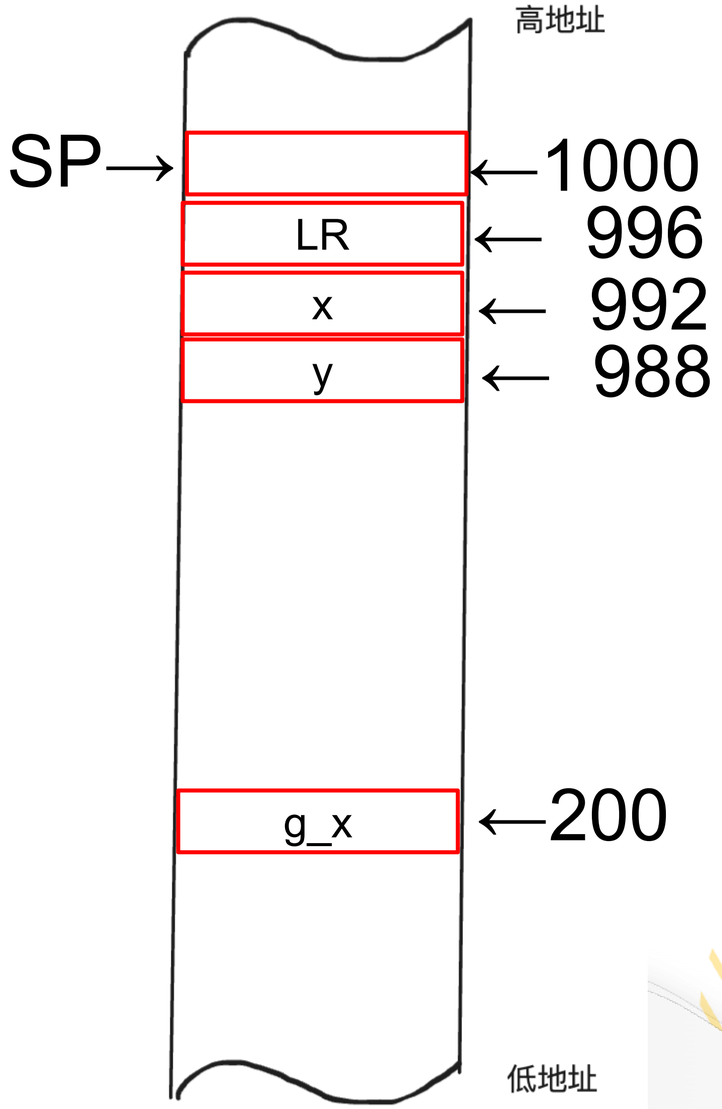
```
1   int FuncA(int x)
2   {
3       int x_fa = 100;
4       return x*x;
5   }
6   int FuncB(int x, int y)
7   {
8       x_fa = 200;
9       return x + y;
10  }
11
12  int main()
13  {
14      FuncA(10);
15      FuncB(20,30);
16      return 0;
17  }
```

FuncB返回



5、全局变量

```
1   int g_x = 100;
2   int FuncA(int x)
3   {
4       int x_fa = 100;
5       g_x = 300;
6       return x*x;
7   }
8   int FuncB(int x, int y)
9   {
10      g_x = 200;
11      return x + y;
12  }
13
14  int main()
15  {
16      g_x = 400;
17      FuncA(10);
18      FuncB(20,30);
19      return 0;
20  }
```



5、模拟器中定义全局变量

01 全局变量的定义DCD或DCB

```
G_X     DCD     1
G_Y     DCD     2,3,4,5,6
G_Z     DCB     1,2,3,4
```

6、全局变量的特点

01 全局变量的定义DCD或DCB

02 全局变量的内容会一直存在

03 全局变量所有函数都可以访问

04 全局变量太多了不方便管理

1.6.3 本节总结

1. 函数内对存储空间的使用都是通过sp进行减操作，返回时用加恢复到原来的值。
2. 也可以把sp保存在寄存器中，在函数返回前从对应的寄存器中恢复。
3. 进入函数时的sp和退出时必须一致
4. 函数内的局部变量，退出后无法再使用
5. 全部变量对应的地址空间的可见性是整个程序

欢迎参与学习

WELCOME FOR YOUR JOINING

船说：计算机基础