System-Level Programming

35 Organisation of Memory – Stack

J. Kleinöder, D. Lohmann, V. Sieh, P. Wägemann

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Erlangen-Nürnberg

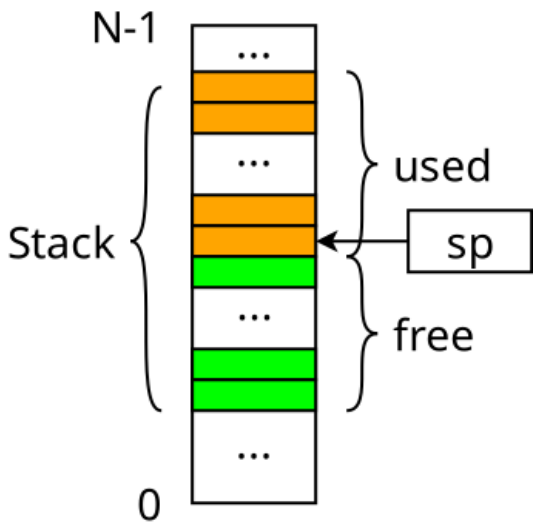
Summer Term 2024

http://sys.cs.fau.de/lehre/ss24

Dynamic Allocation of Memory – Stack

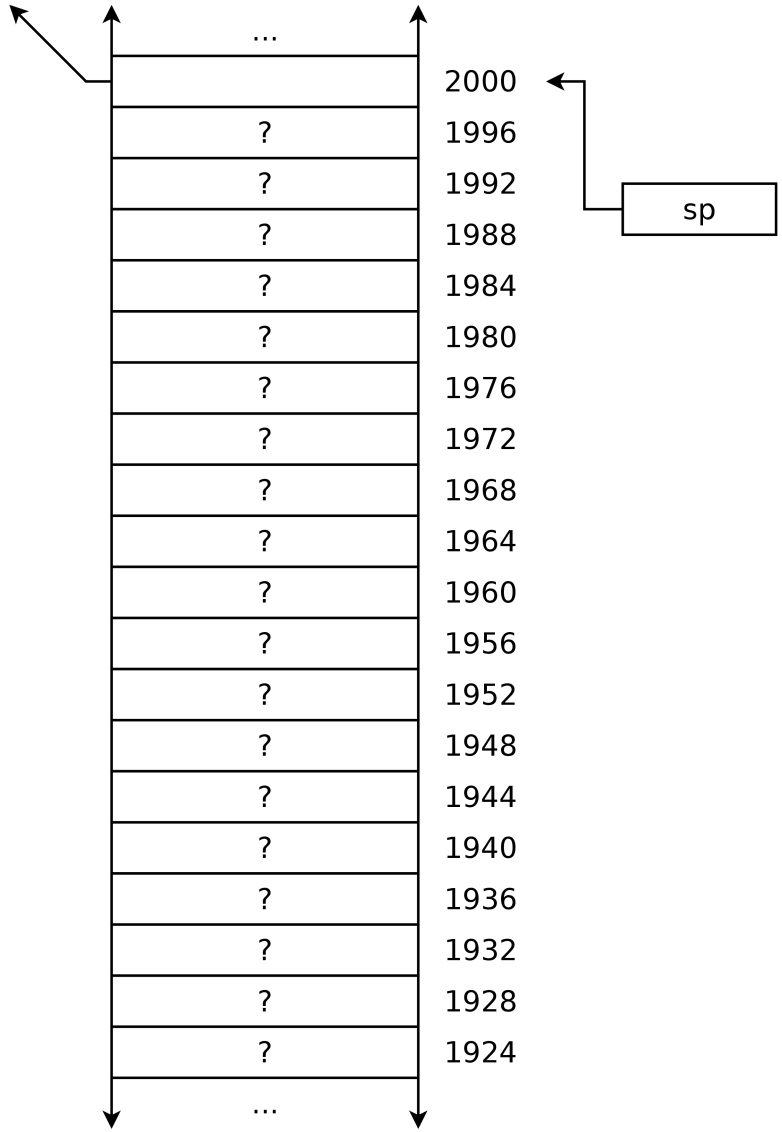
- Local variables, function parameters, and return addresses are organized by the compiler on the stack
- The stack is part of the main memory
- The processor register `sp` "**Stack Pointer**" always points to the last allocated memory on stack
- The stack "grows" "top to bottom'

  $\Rightarrow$ `sp` always points to the start of the stack's used part



Dynamic Allocation of Memory – Stack

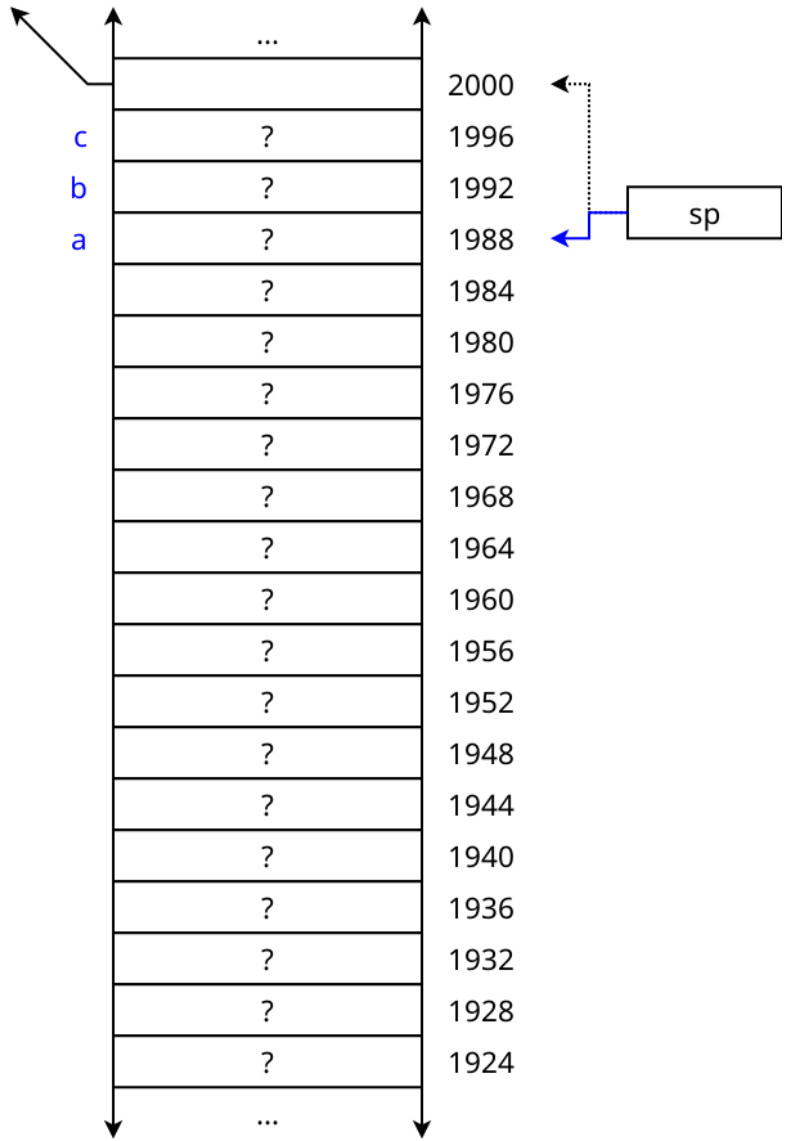
```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

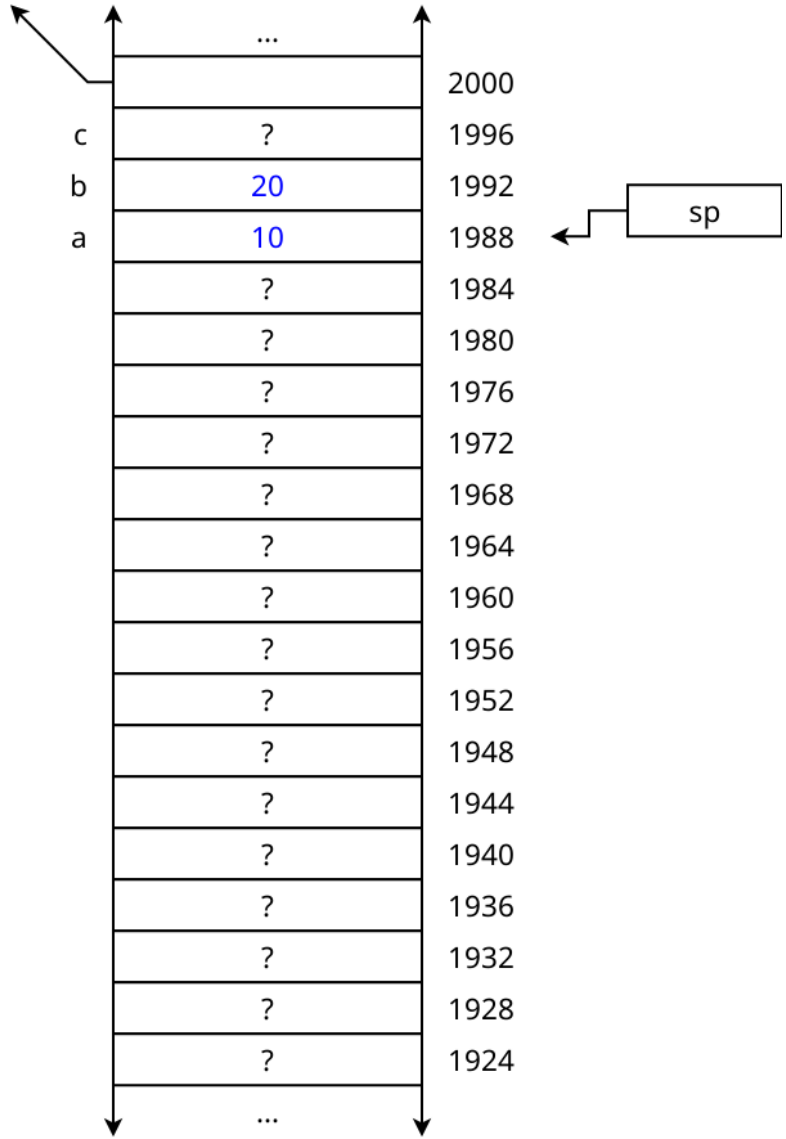
Creating a, b, c



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

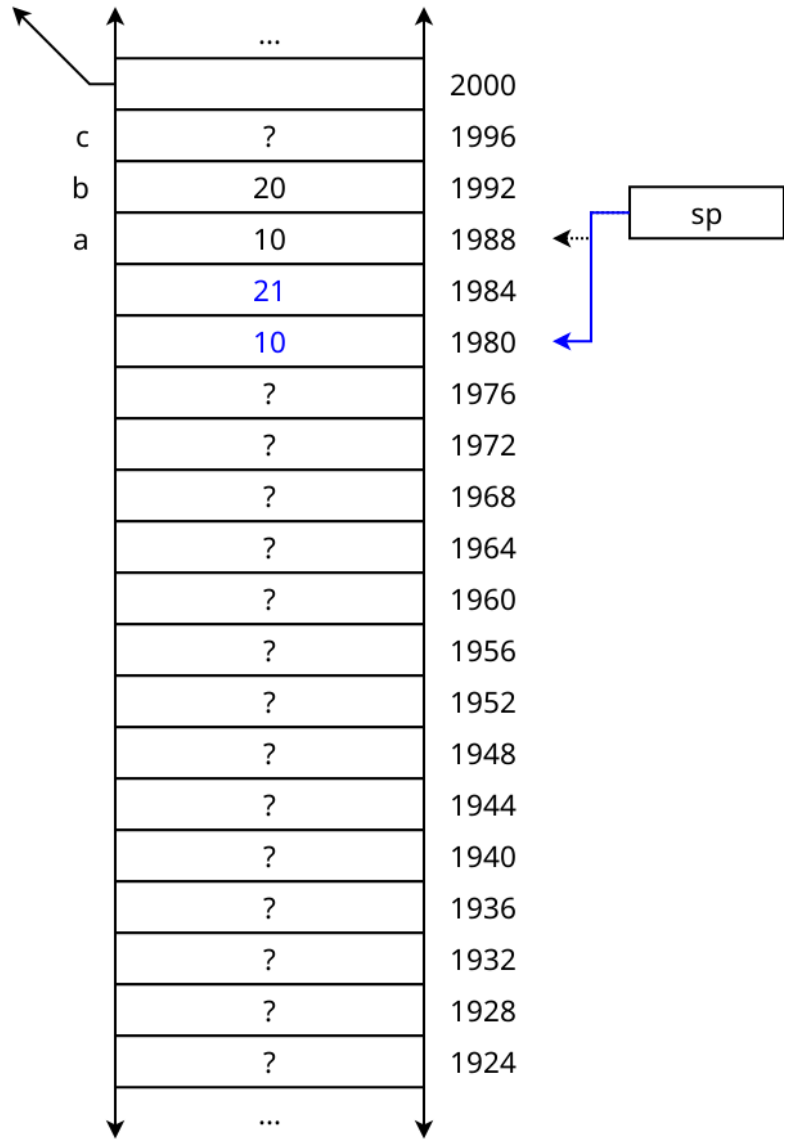
Writing of a, b



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

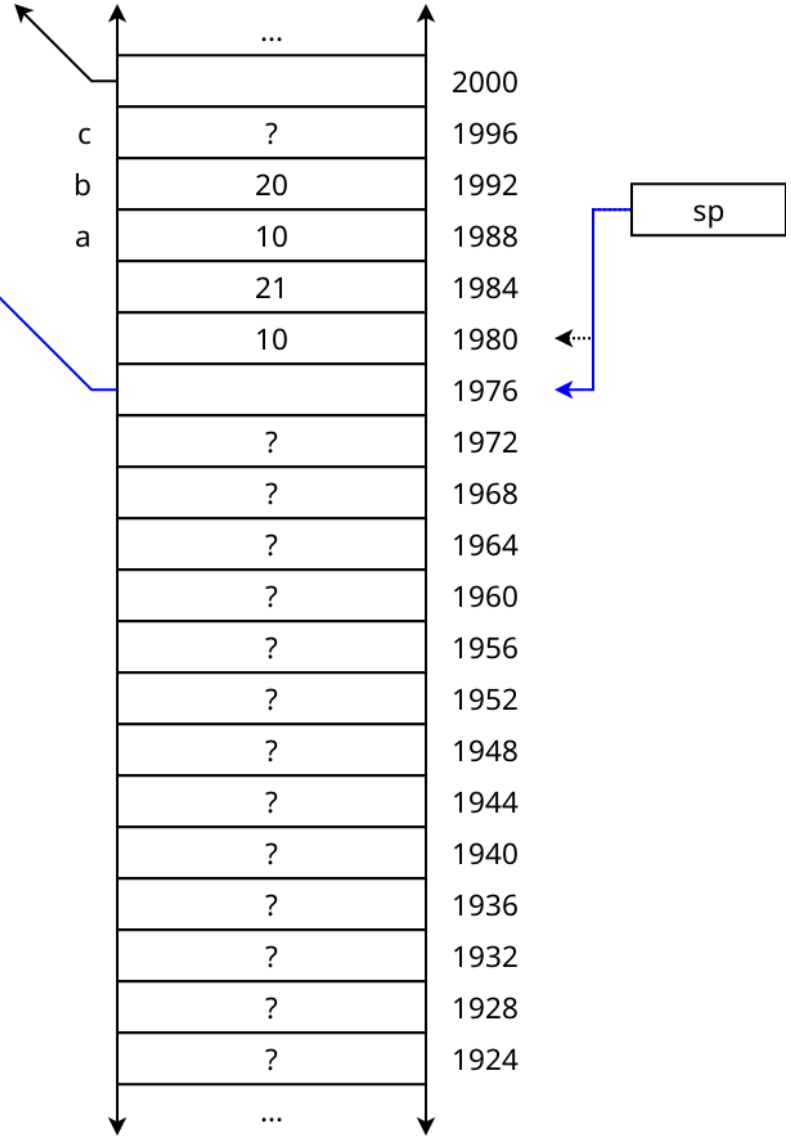
Calculating parameters



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

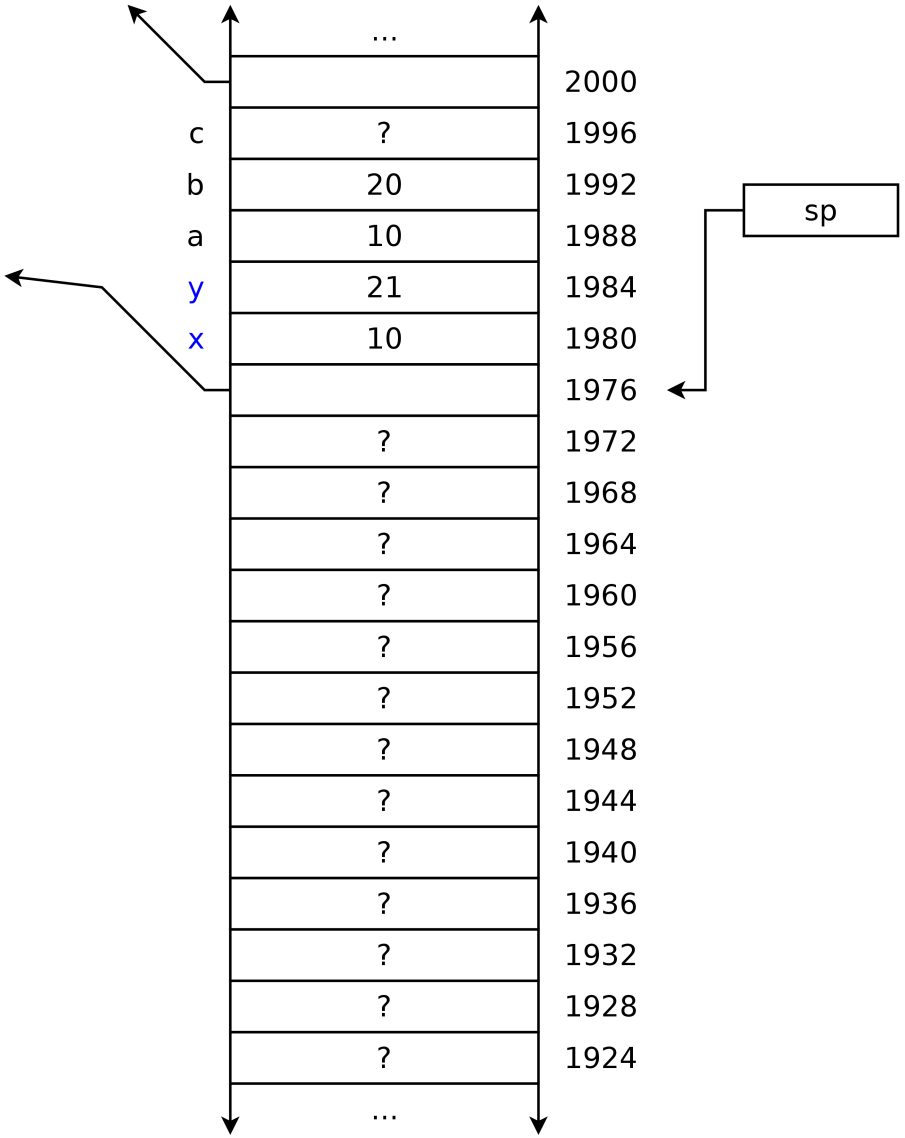
Storing the return address



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

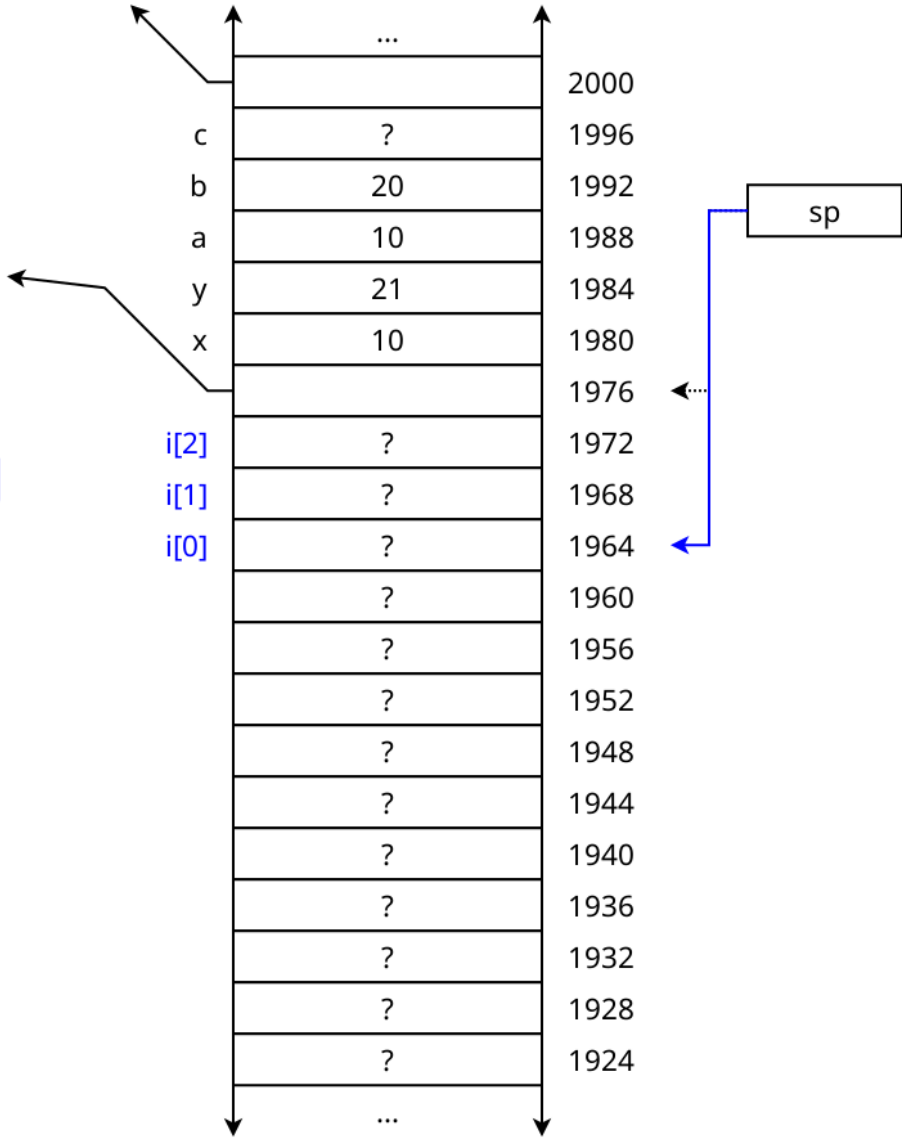
Start f1



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

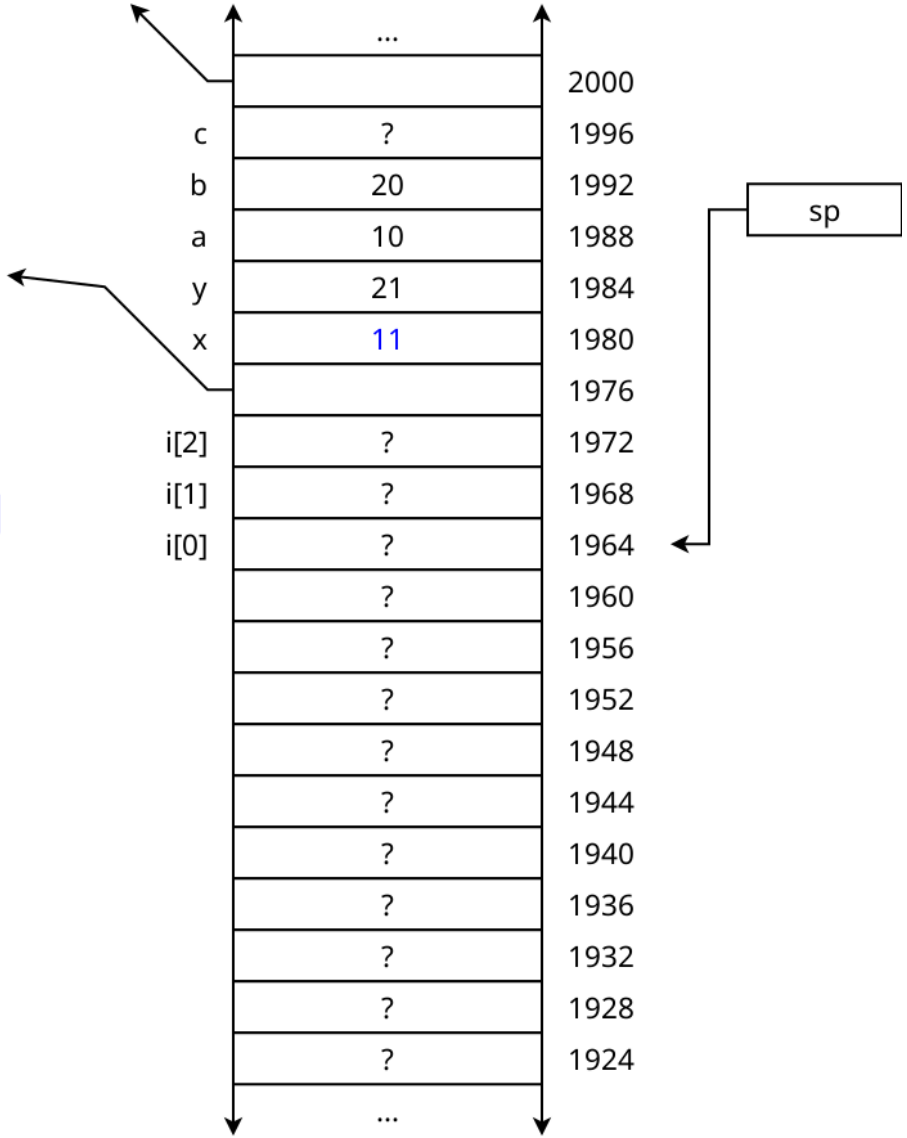
Creating i[0]...i[2]



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

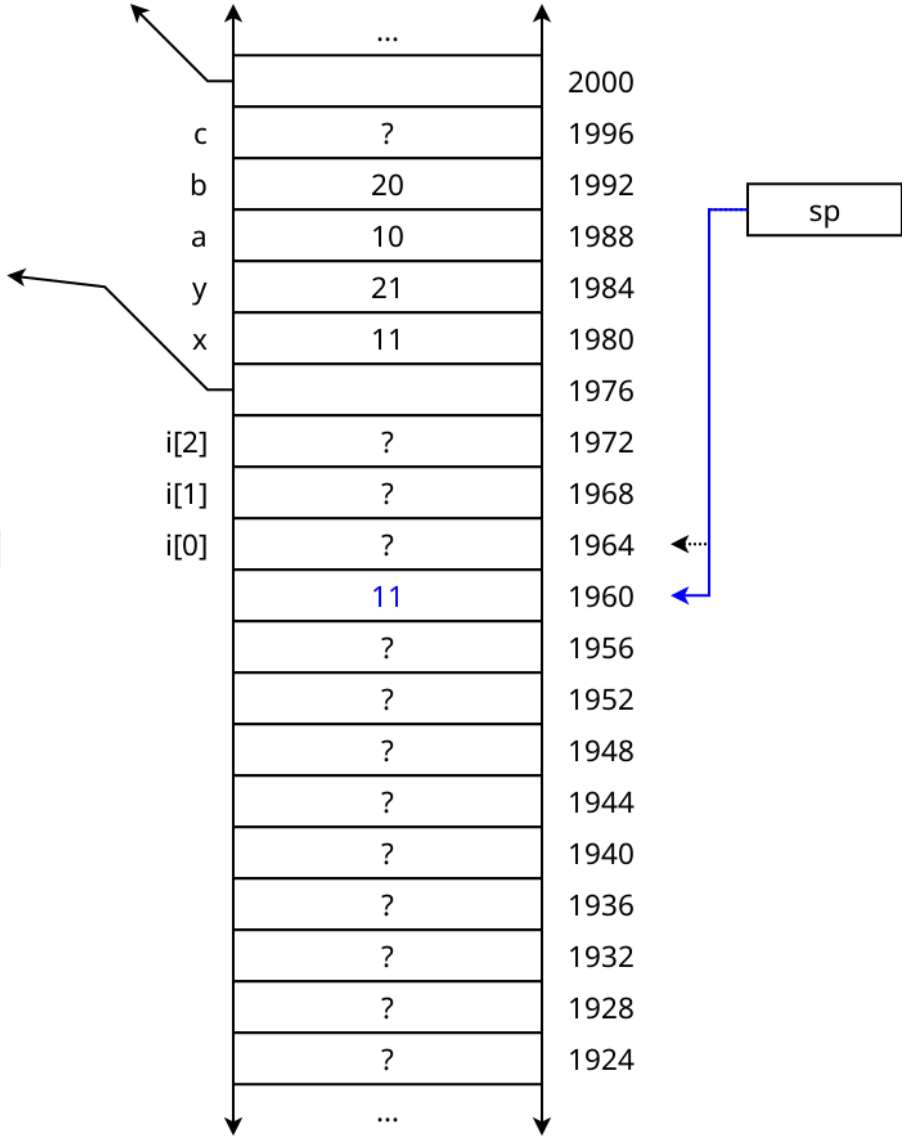
Incrementing x



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

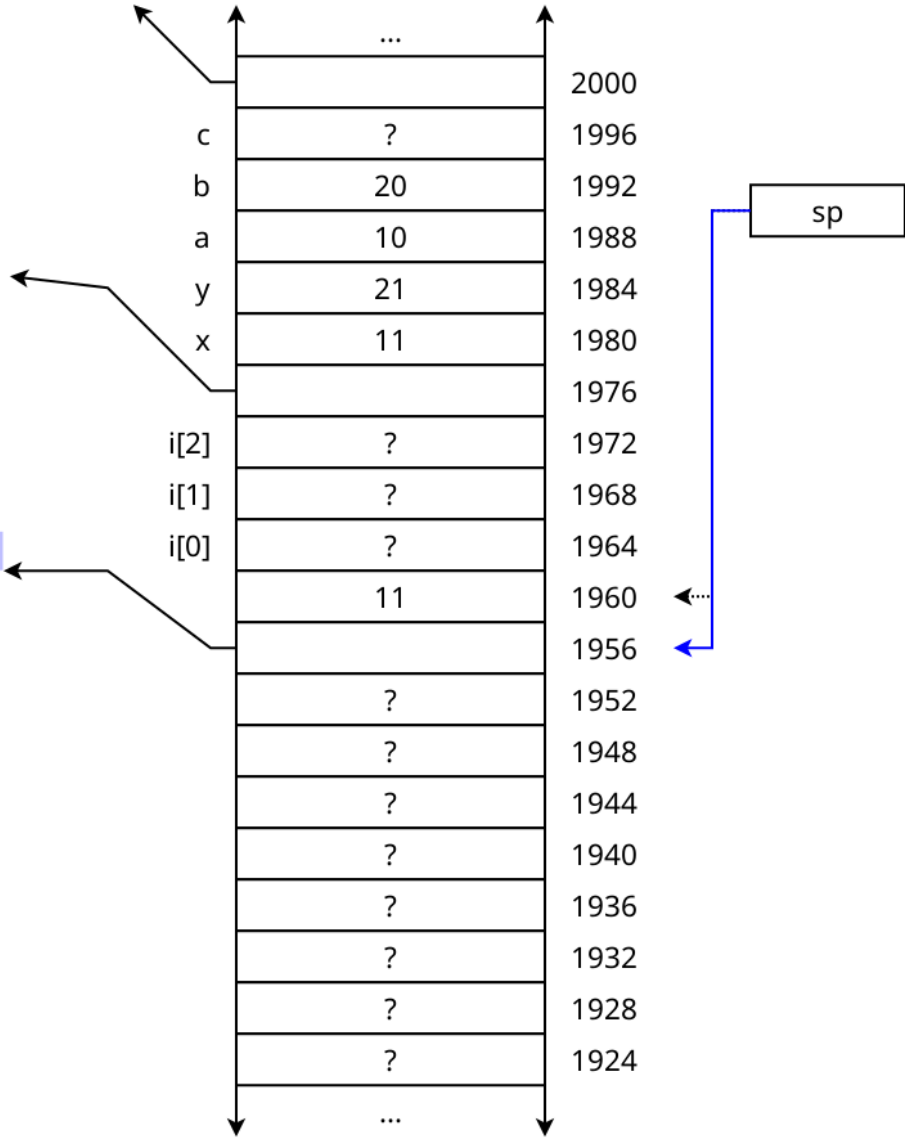
Calculation of the parameter



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

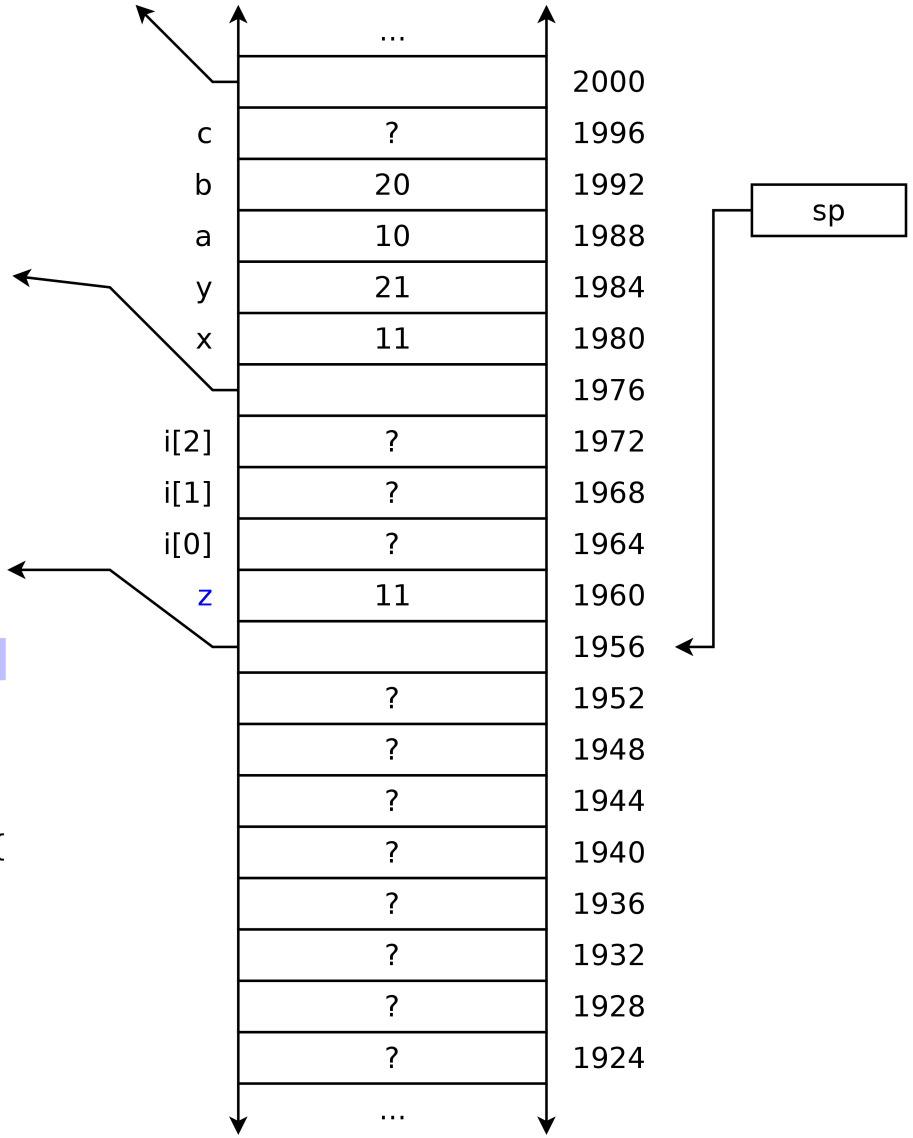
Storing the return address



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

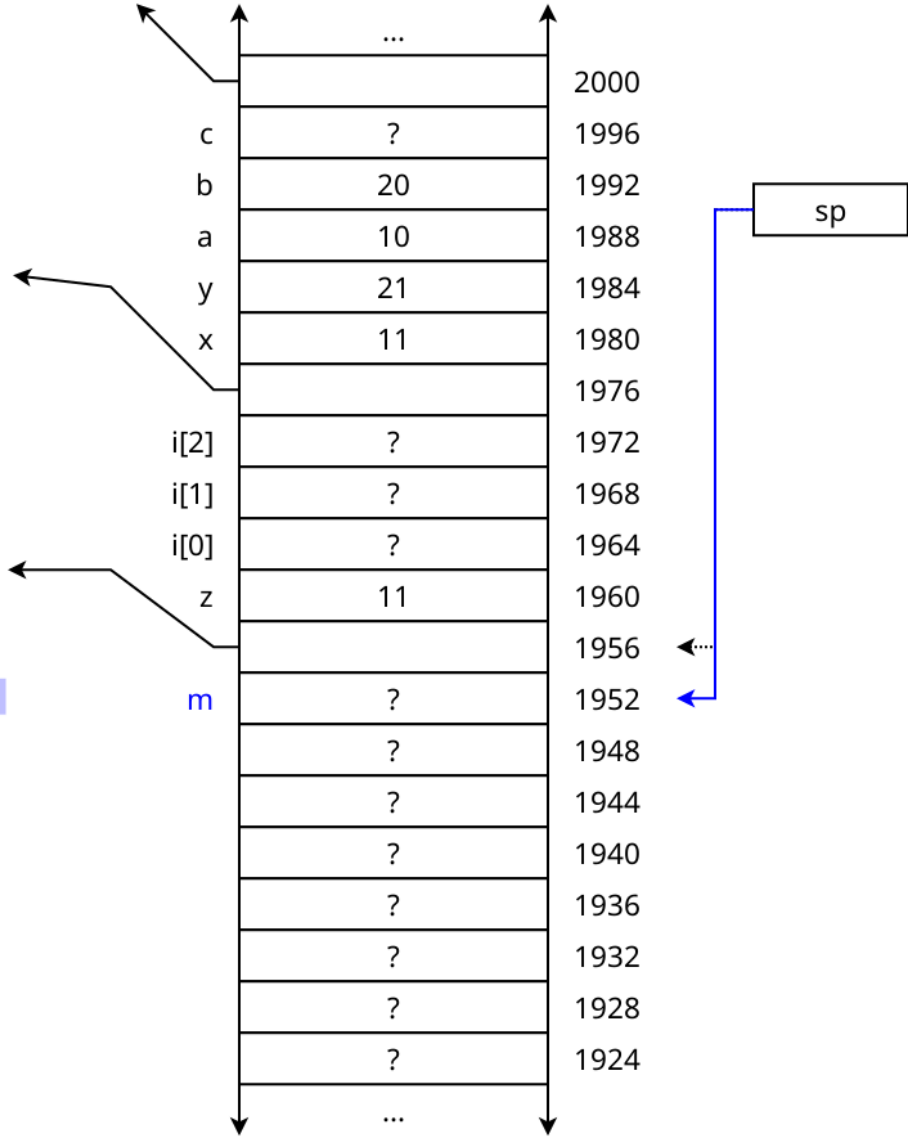
Start f2



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

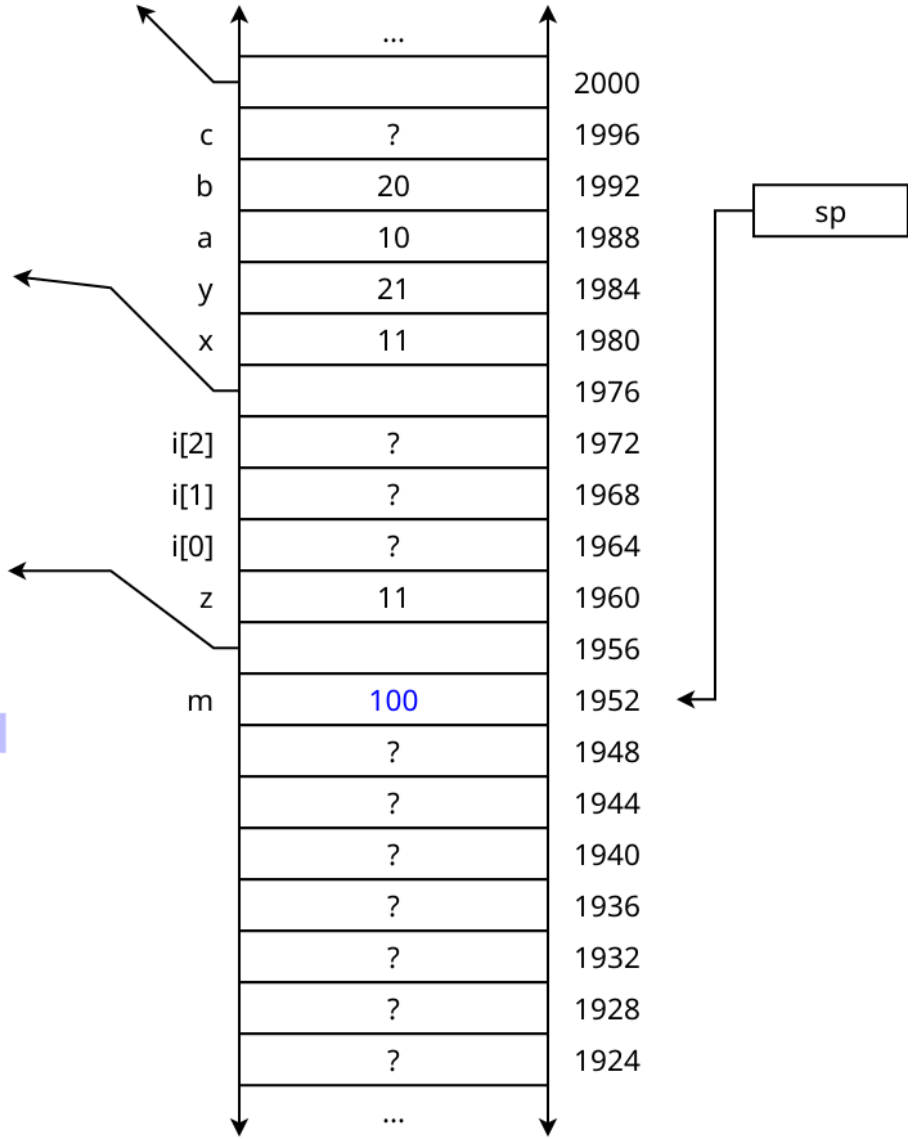
Creating m



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

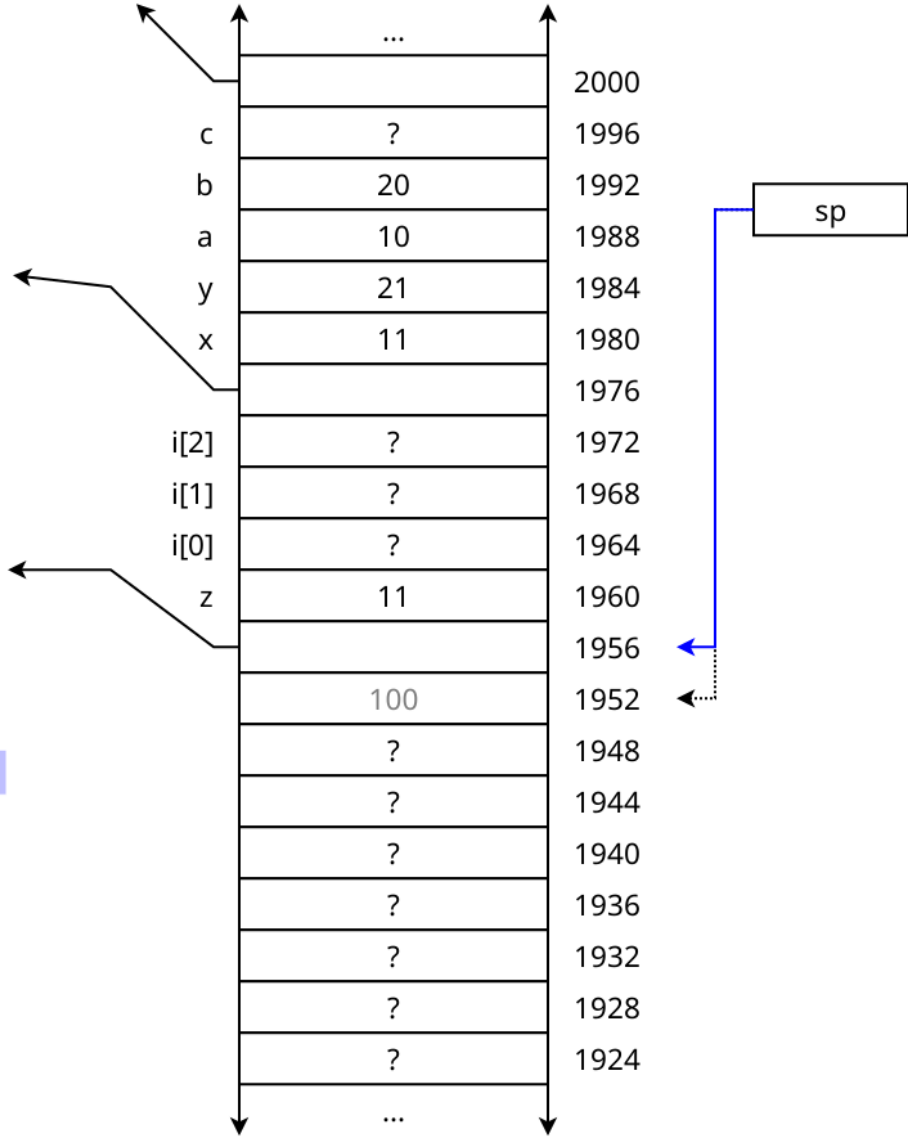
Writing of m



Dynamic Allocation of Memory – Stack

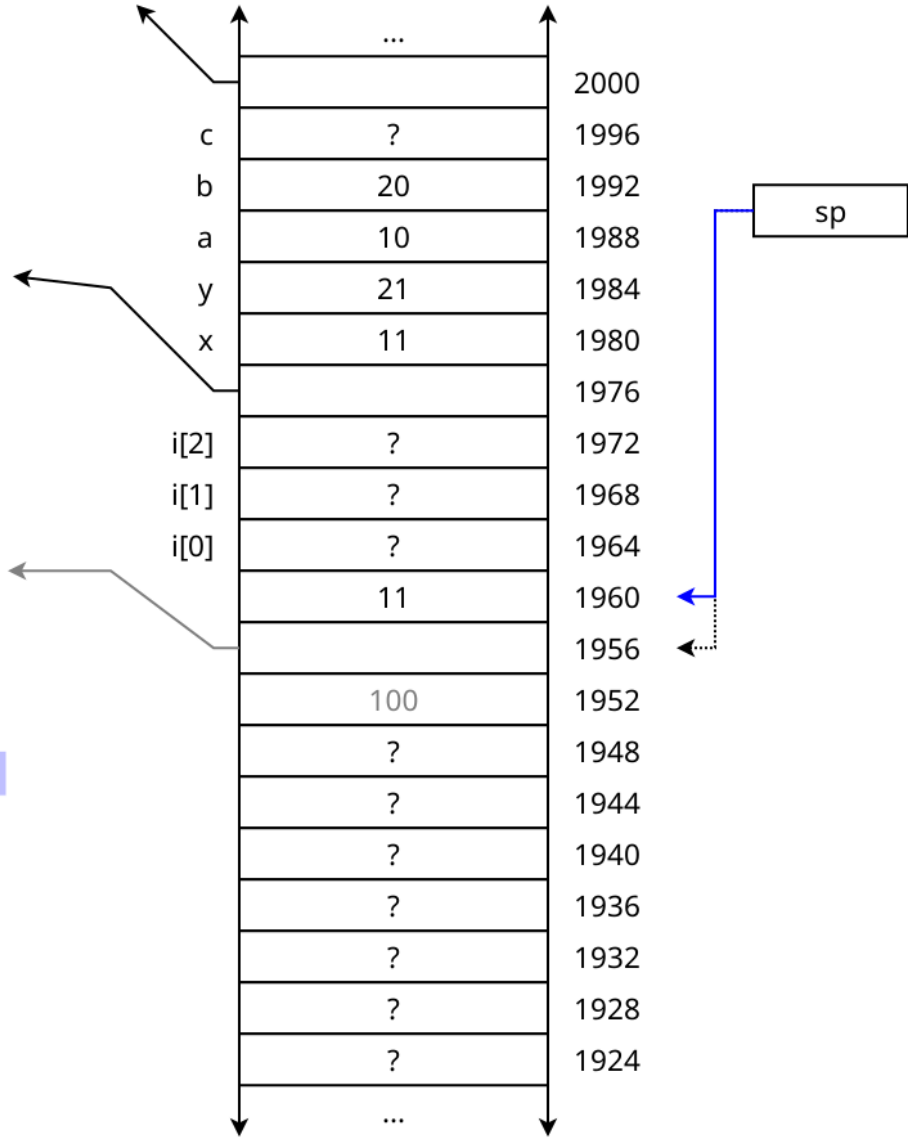
```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

Removing m



Dynamic Allocation of Memory – Stack

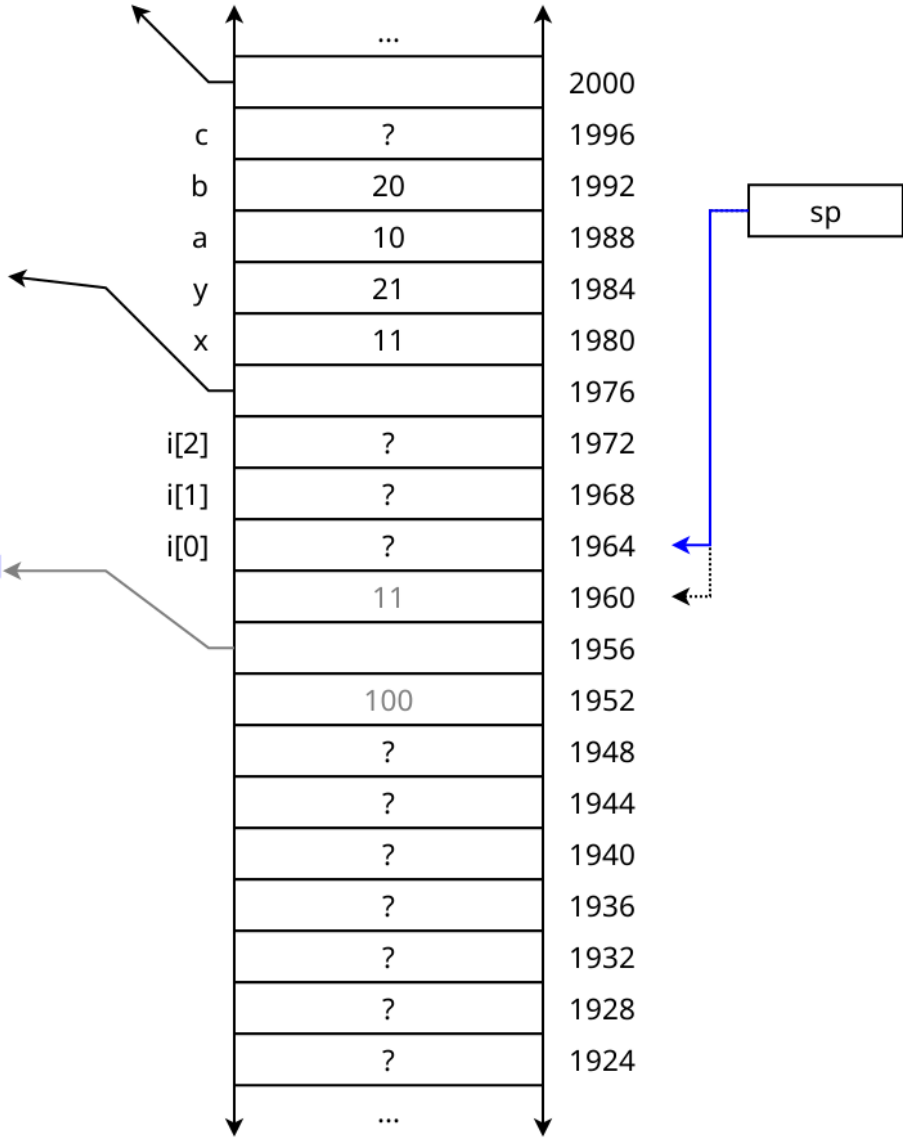
```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}

Return
```



Dynamic Allocation of Memory – Stack

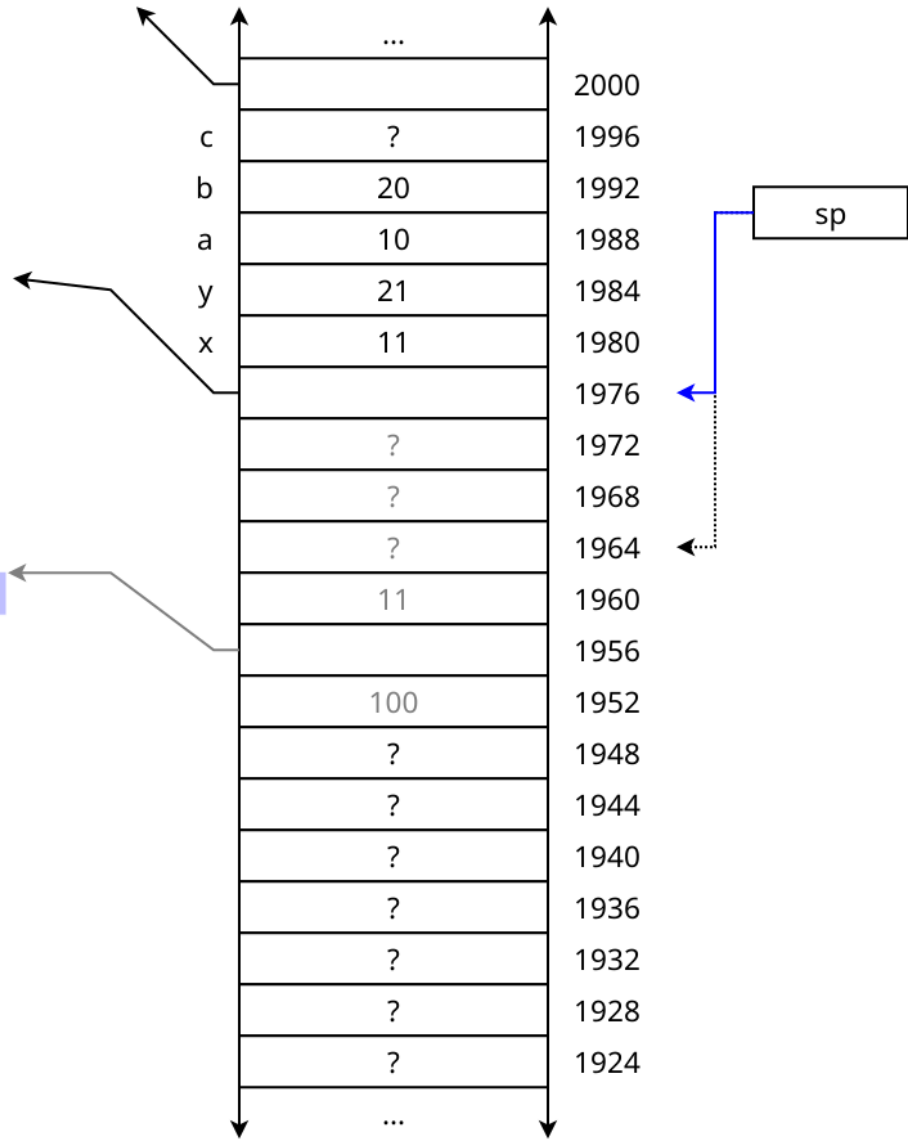
```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

Removing the parameter



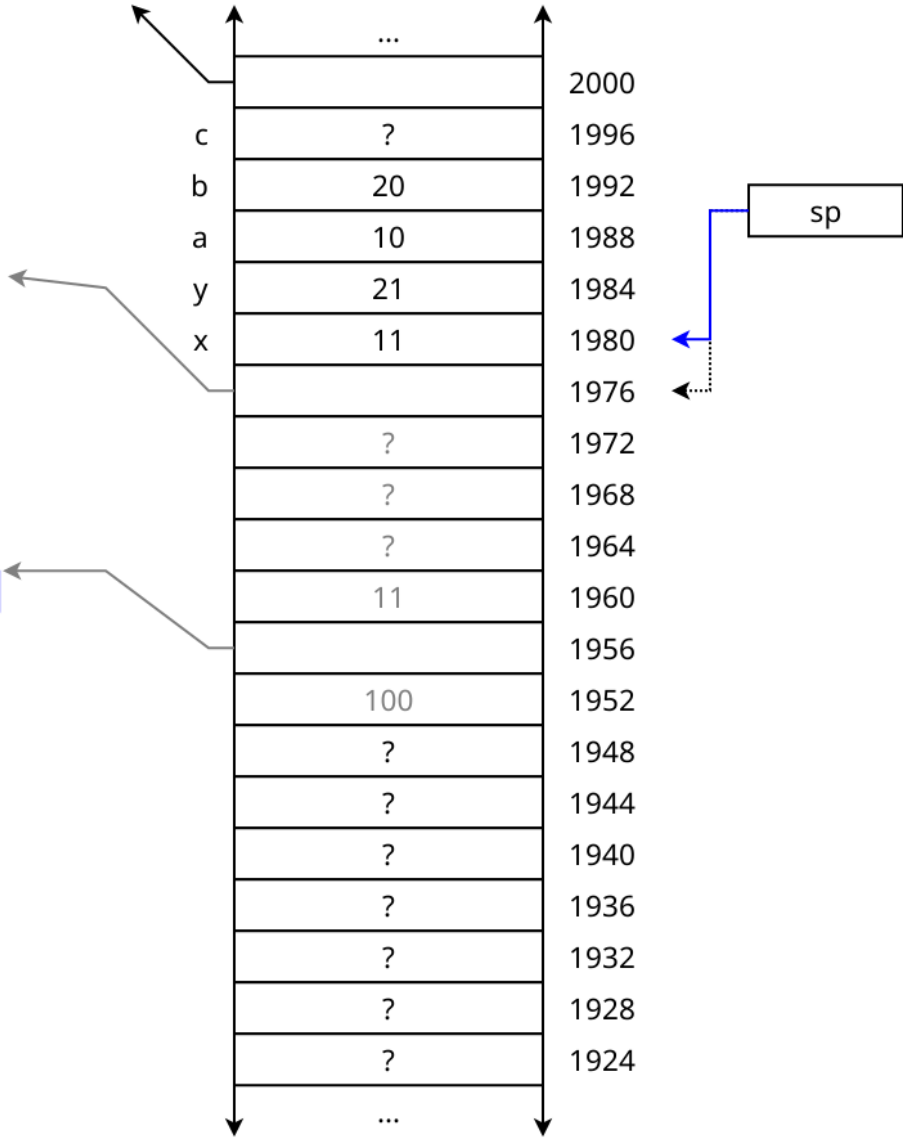
Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}

Removing i[0]...i[2]
```



Dynamic Allocation of Memory – Stack

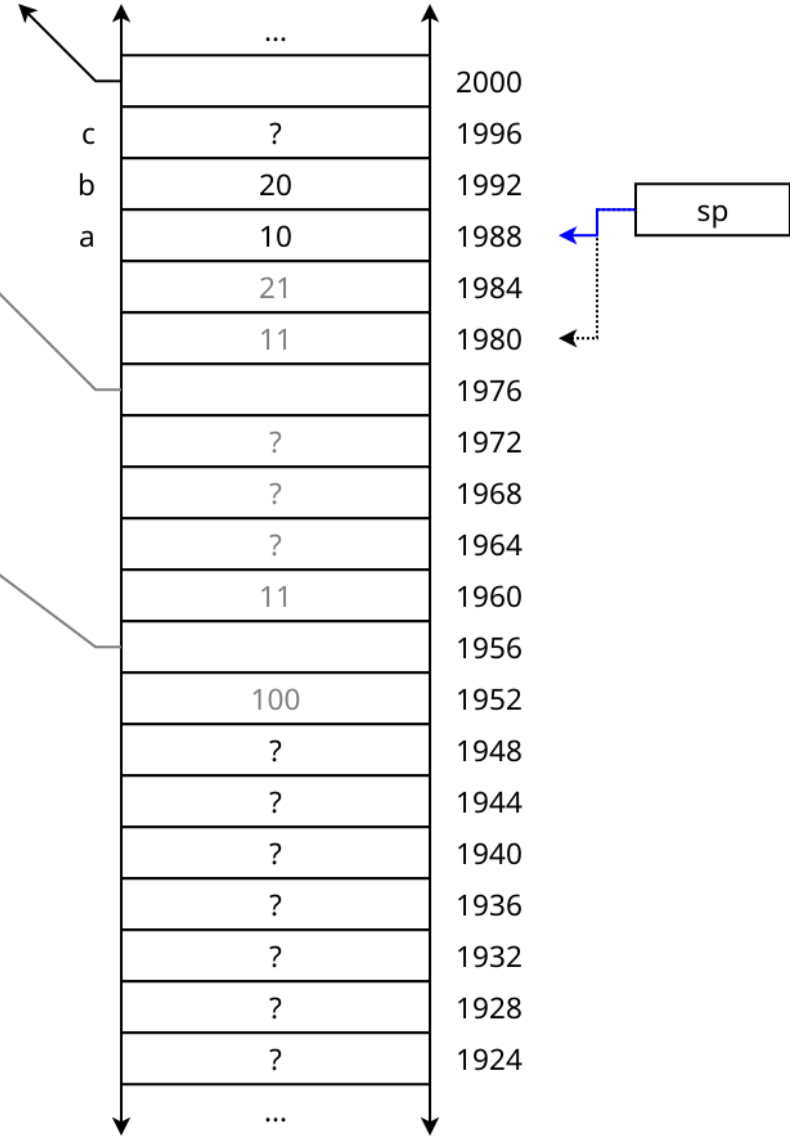
```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}

Return
```



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

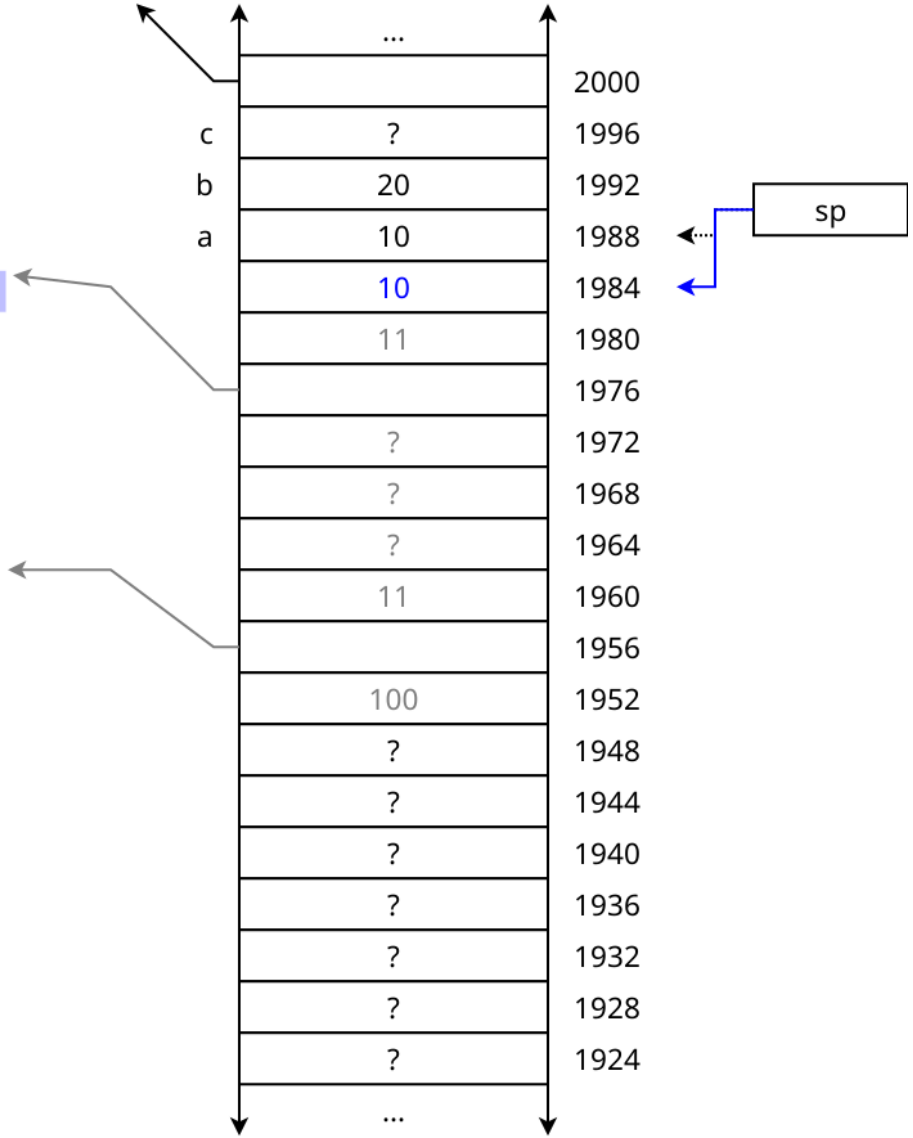
Removing the parameters



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

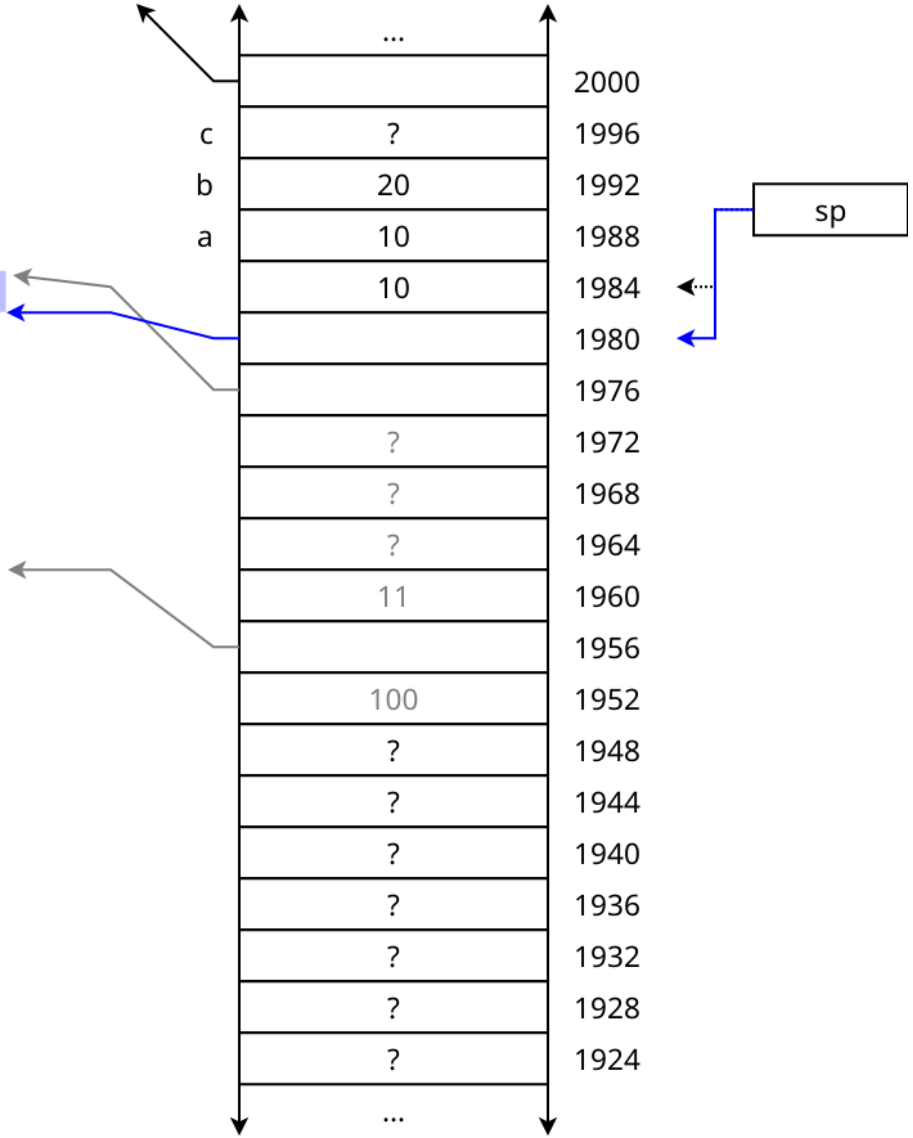
Calculating the parameter



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}
```

Storing the return address



Dynamic Allocation of Memory – Stack

```
void main(void) {
   int a, b, c;

   a = 10;
   b = 20;
   f1(a, b + 1);
   b = f3(a);
   return b;
}

void f1(int x, int y) {
   int i[3];
   x++;
   f2(x);
}

void f2(int z) {
   int m;
   m = 100;
}

int f3(int z1, int z2, int z3) {
   int m;
   return m;
}

Start f3
```