

Systemnahe Programmierung in C

35 Speicherorganisation – Stack

J. Kleinöder, D. Lohmann, V. Sieh

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Erlangen-Nürnberg

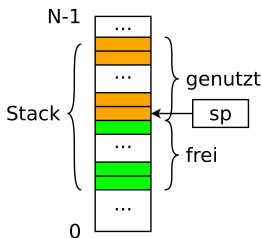
Sommersemester 2024

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



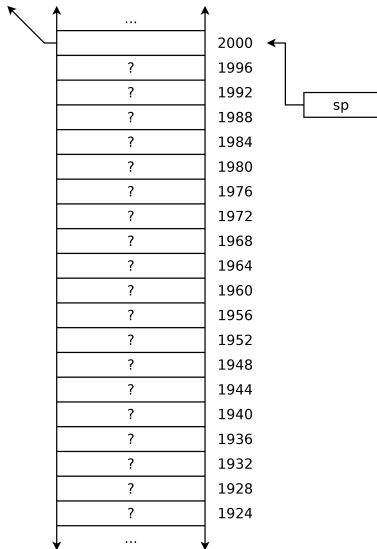
Dynamische Speicherallokation – Stack

- Lokale Variablen, Funktionsparameter und Rücksprungadressen werden vom Übersetzer auf dem **Stack** (Stapel, Keller) verwaltet
- Stack ist Teil des normalen Hauptspeichers
- Prozessorregister **sp** „**Stack Pointer**“ zeigt immer auf das zuletzt abgelegte Datum (architekturabhängig)
- Stack „wächst“ „von oben nach unten“ (architekturabhängig)
=> **sp** zeigt damit immer auf den Anfang des genutzten Teil des Stacks



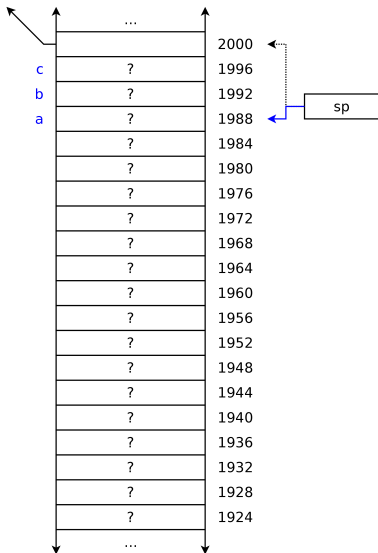
Dynamische Speicherallokation – Stack

```
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;  
}
```



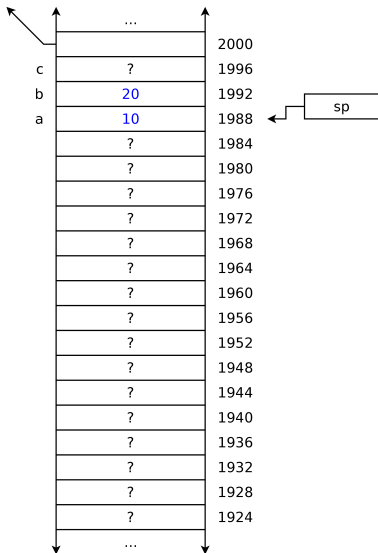
Dynamische Speicherallokation – Stack

```
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;  
}  
  
Anlegen von a, b, c
```



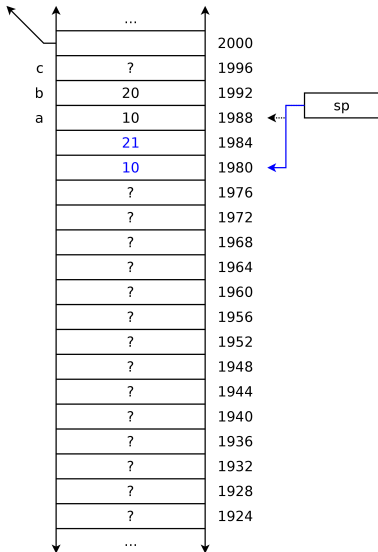
Dynamische Speicherallokation – Stack

```
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;  
}  
  
Schreiben von a, b
```

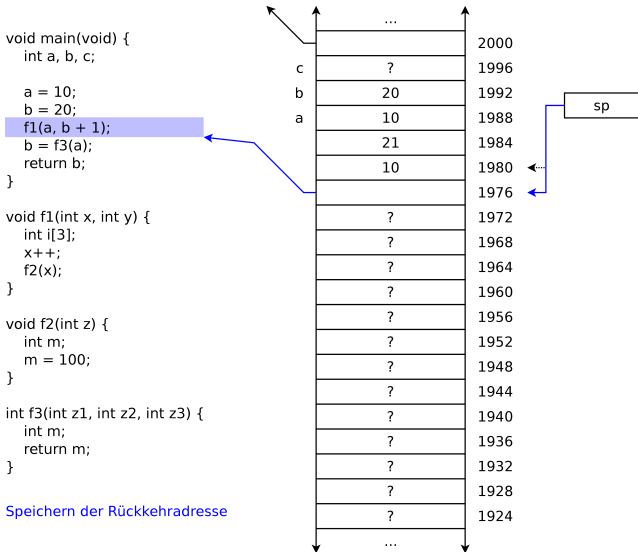


Dynamische Speicherallokation – Stack

```
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;  
}  
  
Berechnen der Parameter
```



Dynamische Speicherallokation – Stack



Dynamische Speicherallokation – Stack

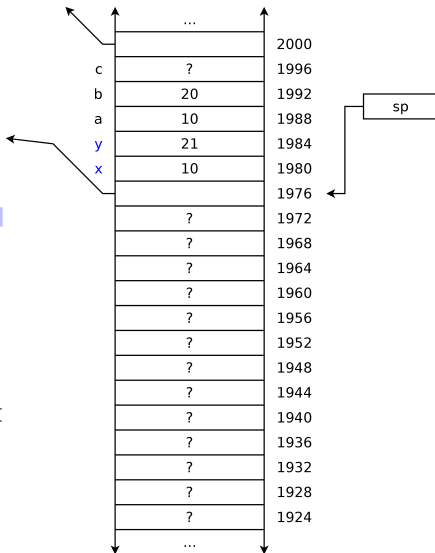
```
void main(void) {  
    int a, b, c;  
  
    a = 10;  
    b = 20;  
    f1(a, b + 1);  
    b = f3(a);  
    return b;  
}
```

```
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;  
}
```

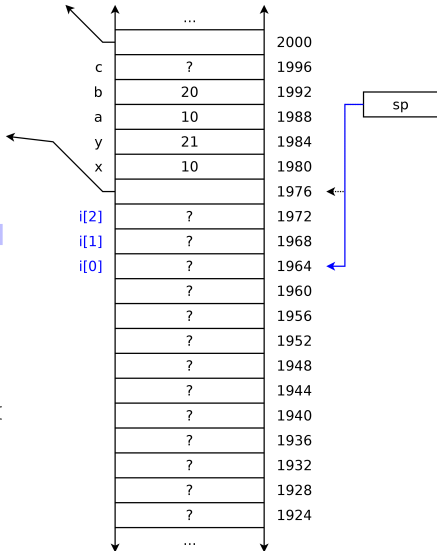
Start f1



Dynamische Speicherallokation – 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;  
}
```

Anlegen von i[0]...i[2]



Dynamische Speicherallokation – Stack

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

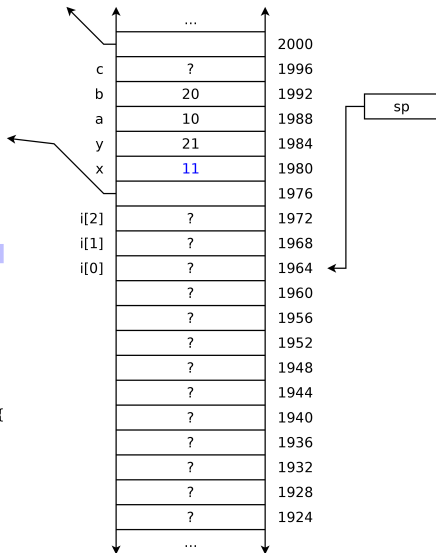
    a = 10;
    b = 20;
    f1(a, b + 1);
    b = f3(a);
    return b;
}

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;
}
```

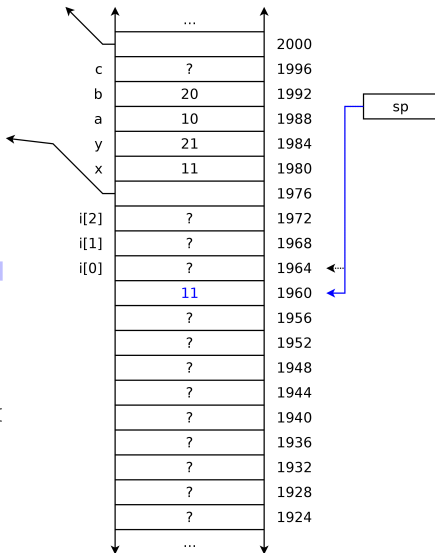
Inkrementieren von x



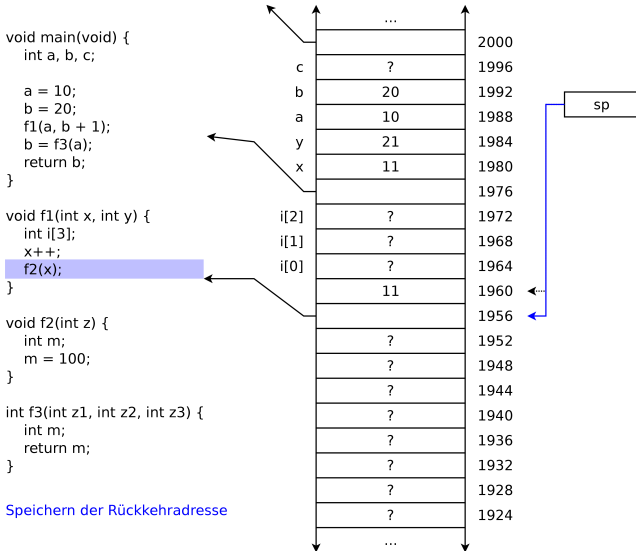
Dynamische Speicherallokation – Stack

```
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;  
}
```

Berechnen des Parameters

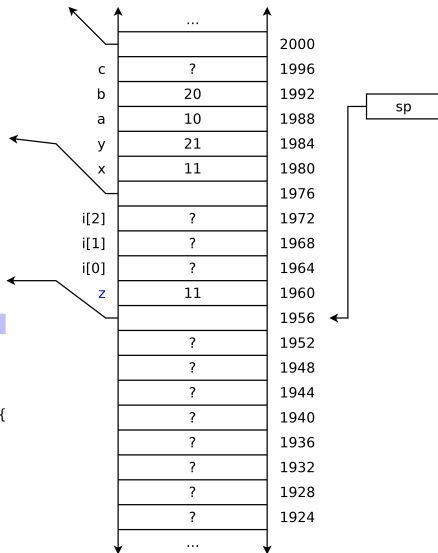


Dynamische Speicherallokation – Stack



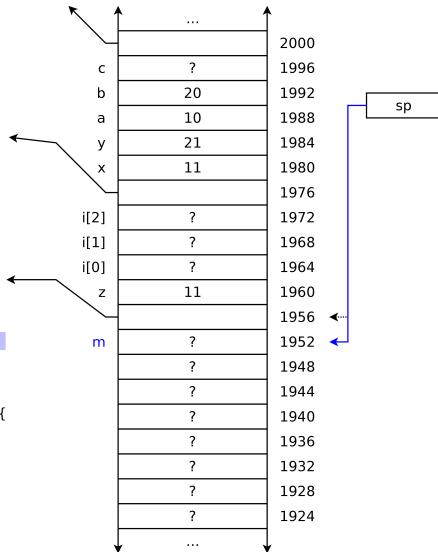
Dynamische Speicherallokation – Stack

```
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
```



Dynamische Speicherallokation – 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;  
}  
  
Anlegen von m
```



Dynamische Speicherallokation – Stack

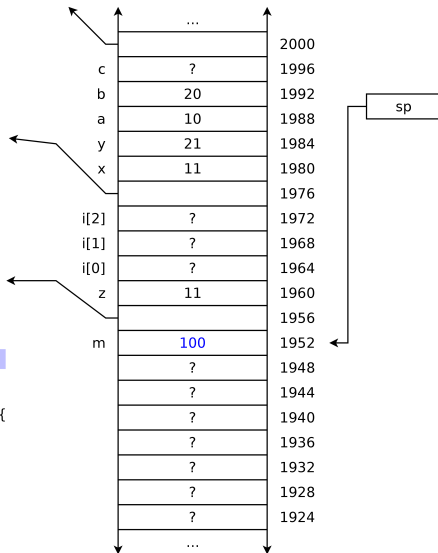
```
void main(void) {  
    int a, b, c;  
  
    a = 10;  
    b = 20;  
    f1(a, b + 1);  
    b = f3(a);  
    return b;  
}
```

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

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

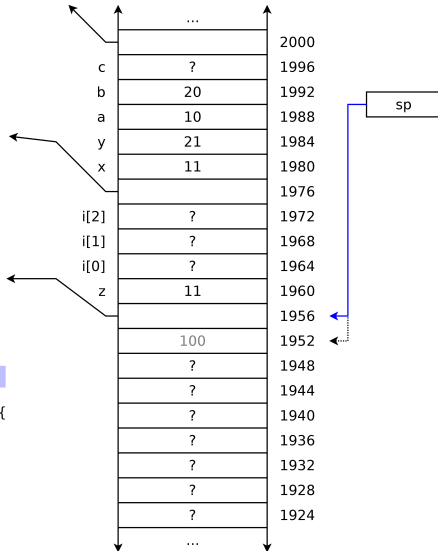
```
int f3(int z1, int z2, int z3) {  
    int m;  
    return m;  
}
```

Schreiben von m



Dynamische Speicherallokation – 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;  
}  
  
Entfernen von m
```



Dynamische Speicherallokation – Stack

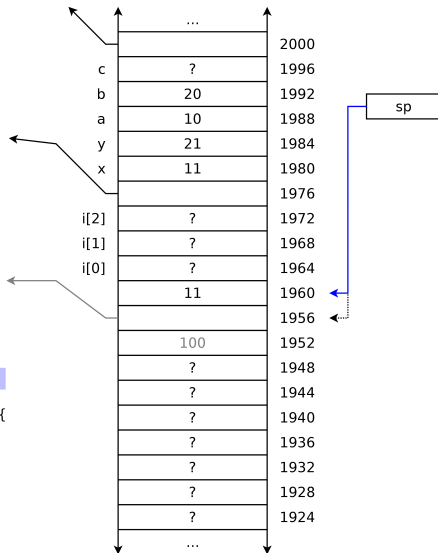
```
void main(void) {  
    int a, b, c;  
  
    a = 10;  
    b = 20;  
    f1(a, b + 1);  
    b = f3(a);  
    return b;  
}
```

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

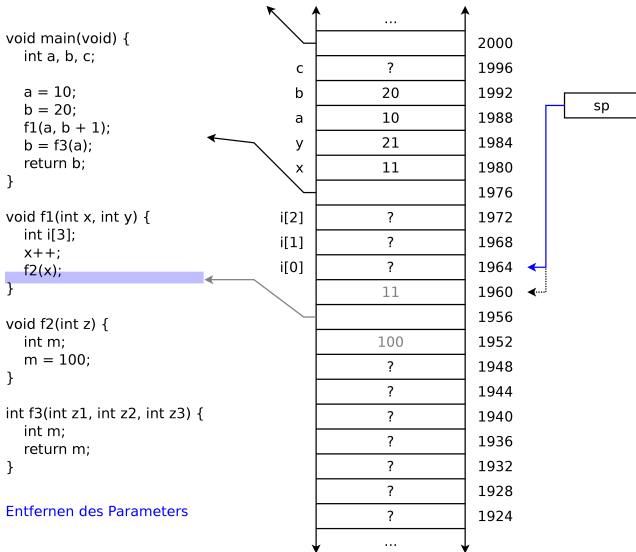
```
void f2(int z) {  
    int m;  
    m = 100;  
}
```

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

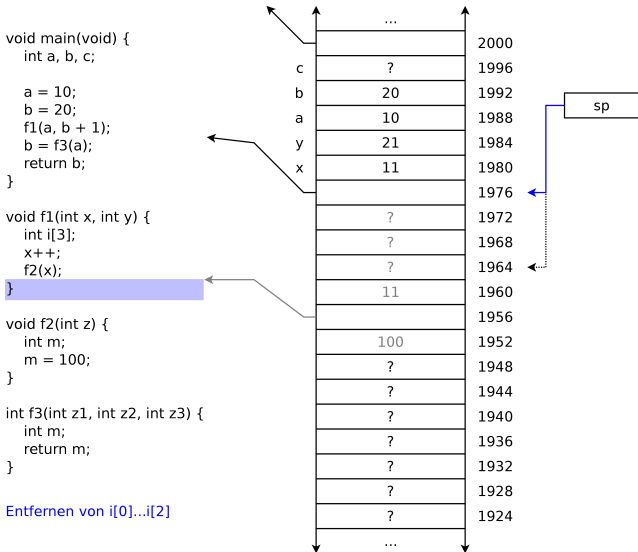
Rücksprung



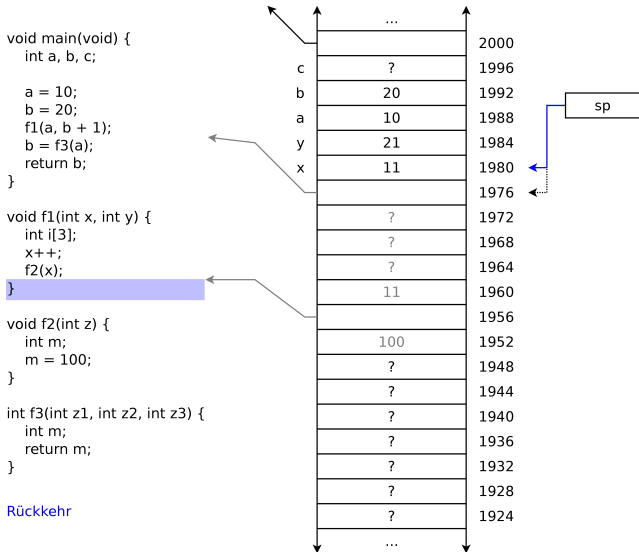
Dynamische Speicherallokation – Stack



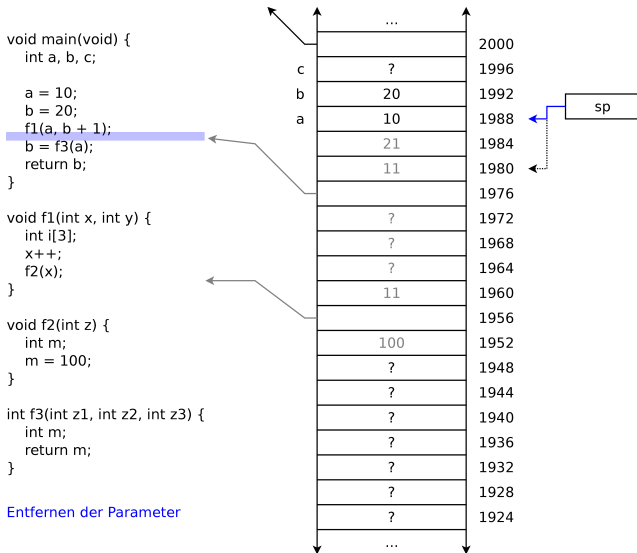
Dynamische Speicherallokation – Stack



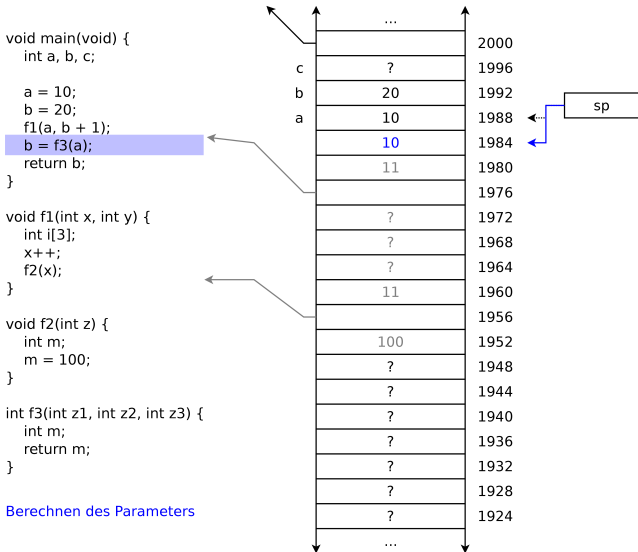
Dynamische Speicherallokation – Stack



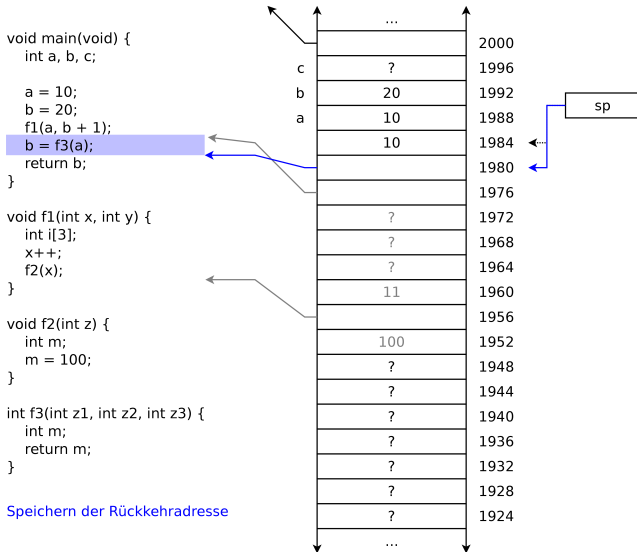
Dynamische Speicherallokation – Stack



Dynamische Speicherallokation – Stack



Dynamische Speicherallokation – Stack



Dynamische Speicherallokation – 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

