

J Programme, Prozesse und Speicher

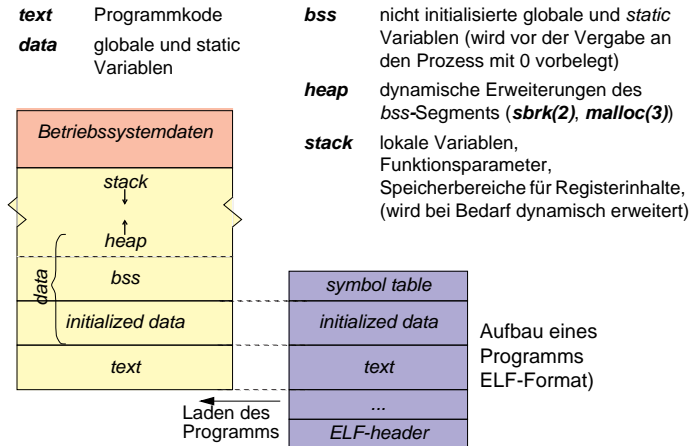
J Programme, Prozesse und Speicher

- **Programm:** Folge von Anweisungen (hinterlegt beispielsweise als ausführbare Datei auf dem Hintergrundspeicher)
- **Prozess:** Betriebssystemkonzept
 - Programm, das sich in Ausführung befindet, und seine Daten (Beachte: ein Programm kann sich mehrfach in Ausführung befinden)
 - eine konkrete Ausführungsumgebung für ein Programm Speicher, Rechte, Verwaltungsinformation (verbrauchte Rechenzeit,...),...
- jeder Prozess bekommt einen eigenen virtuellen Adressraum zur Verfügung gestellt
 - eigener (virtueller) Speicherbereich von 0 bis 2 GB (oder mehr bis 4 GB)
 - Datenbereiche von verschiedenen Prozessen und Betriebssystem sind gegeneinander geschützt
 - Datentransfer zwischen Prozessen nur durch Vermittlung des Betriebssystems möglich

SPIC

J.1 Speicherorganisation eines Prozesses

J.1 Speicherorganisation eines Prozesses



SPIC

J.1 Speicherorganisation eines Prozesses (2)

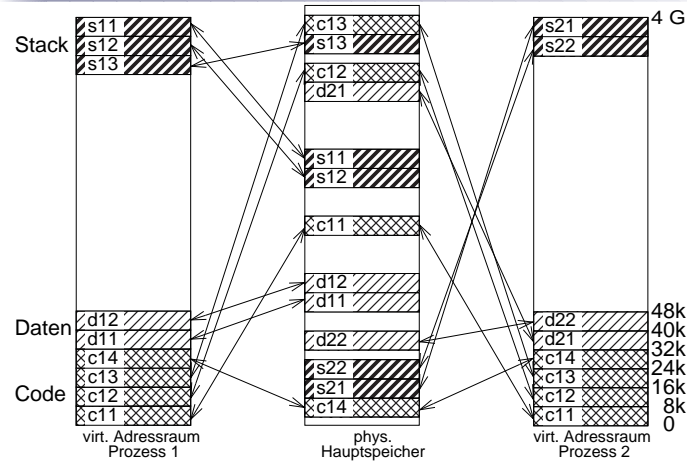
J.1 Speicherorganisation eines Prozesses

- Abbildung des virtuellen Adressraums in den realen Hauptspeicher durch Seitenadressierung (**Paging**)
 - Adreßraum ist in kleine (4 oder 8 kB) Stücke unterteilt (**Seiten**)
 - jede Seite wird über eine Tabelle in ein entsprechendes Stück des Hauptspeichers (**Kachel**) abgebildet
 - bei jedem Speicherzugriff wird die virtuelle Adresse in die entsprechende physikalische Adresse umgerechnet (spezielle Hardware: Memory Management Unit - MMU)
 - zu jeder Seite sind Zugriffsrechte vermerkt (nur lesen, lesen+schreiben, Maschinenbefehle ausführen)
 - eine Seite kann bei Speichermangel von Betriebssystem auf Festplatte ausgelagert werden und bei Bedarf automatisch wieder eingelagert werden

SPIC

J.1 Speicherorganisation eines Prozesses(3)

J.1 Speicherorganisation eines Prozesses



SPIC

J.2 Stackaufbau eines Prozesses

1 Prinzip

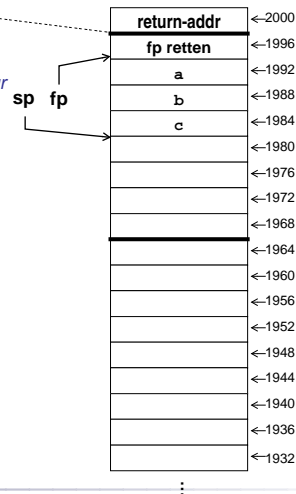
- für jede Funktion wird ein **Stack-Frame** angelegt, in dem
 - lokale Variablen der Funktion
 - Aufrufparameter an weitere Funktionen
 - Registerbelegung der Funktion während des Aufrufs weiterer Funktionen
 gespeichert werden
- Stackorganisation ist abhängig von
 - Prozessor
 - Compiler und
 - Betriebssystem
- Beispiele aus einem UNIX auf Intel-Prozessor (typisch für CISC)
 - RISC-Prozessoren mit Registerfiles gehen teilweise anders vor!

SPIC

2 ■ Stack mehrerer Funktionsaufrufe

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

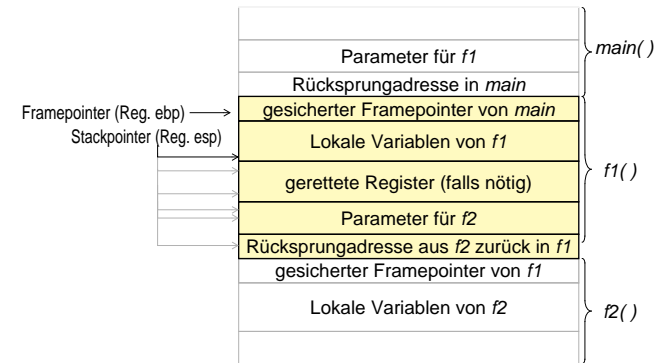
Stack-Frame für main erstellen
&a = fp-4
&b = fp-8
&c = fp-12



SPIC

2 Beispiel

■ Aufbau eines Stack-Frames (Funktionen main(), f1(), f2())



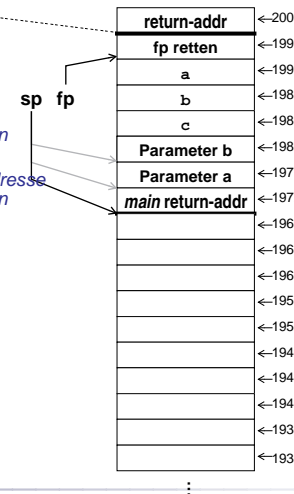
► Achtung: architekturabhängige Optimierungen können zu Padding (Füllbytes) führen!

SPIC

2 ■ Stack mehrerer Funktionsaufrufe

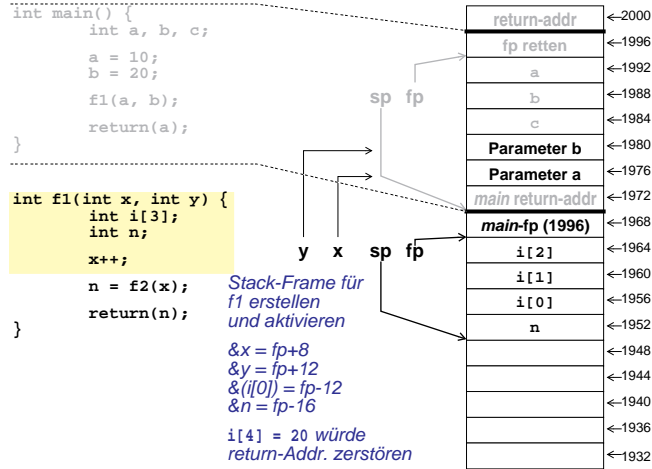
```
int main() {
    int a, b, c;
    a = 10;
    b = 20;
    f1(a, b);
    return(a);
}
```

Parameter auf Stack legen
Bei Aufruf Rücksprungadresse auf Stack legen

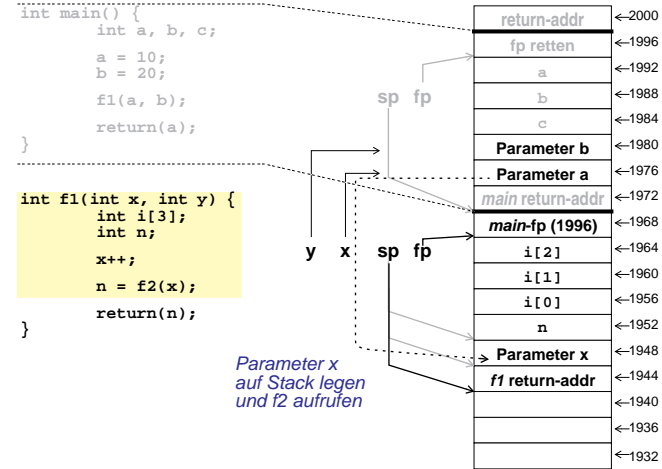


SPIC

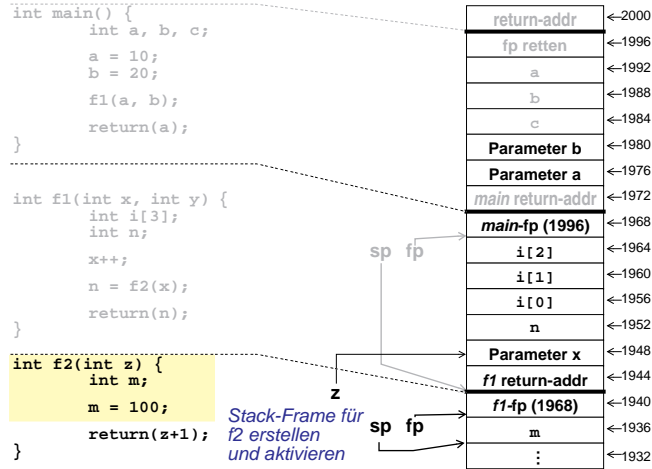
2 ■ Stack mehrerer Funktionsaufrufe



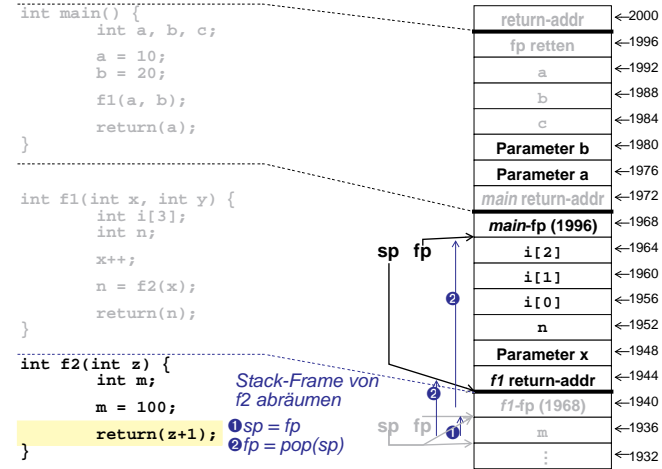
2 ■ Stack mehrerer Funktionsaufrufe



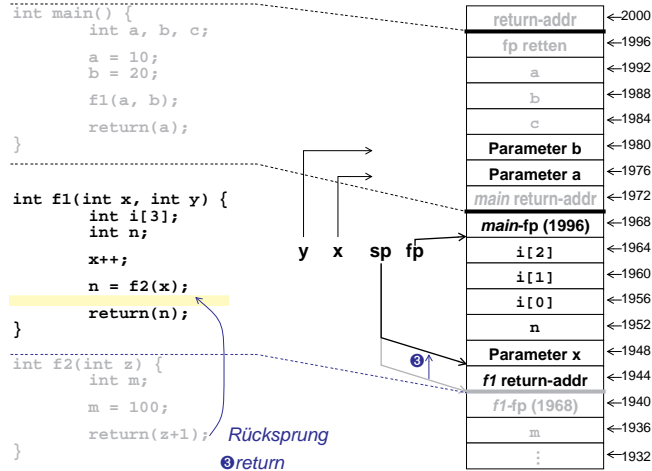
2 ■ Stack mehrerer Funktionsaufrufe



2 ■ Stack mehrerer Funktionsaufrufe

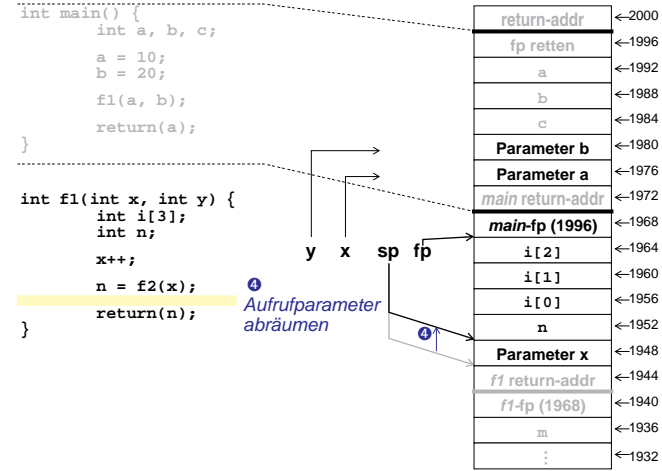


2 ■ Stack mehrerer Funktionsaufrufe



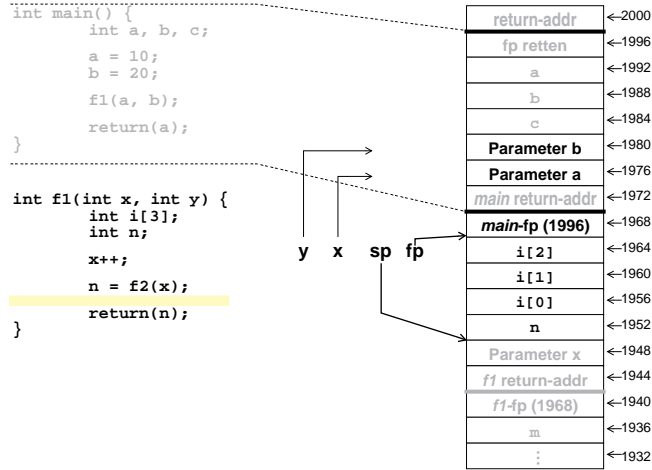
SPIC

2 ■ Stack mehrerer Funktionsaufrufe



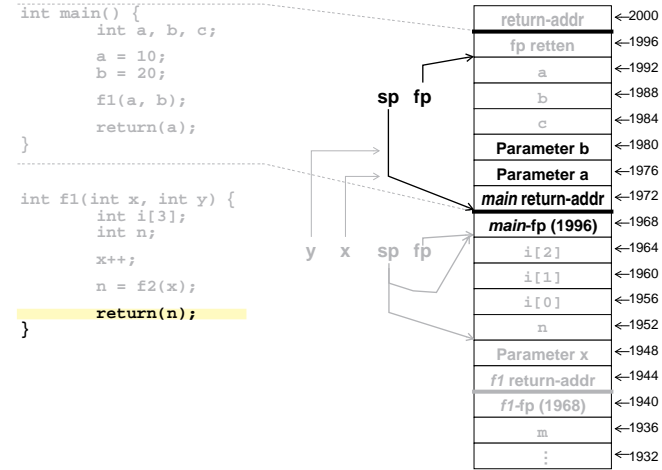
SPIC

2 ■ Stack mehrerer Funktionsaufrufe



SPIC

2 ■ Stack mehrerer Funktionsaufrufe

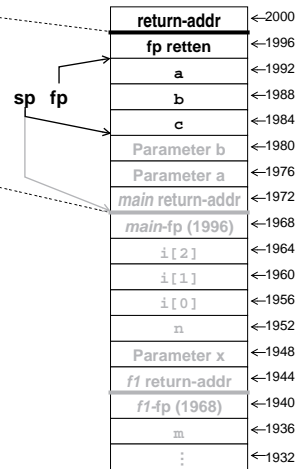


SPIC

2 ■ Stack mehrerer Funktionsaufrufe

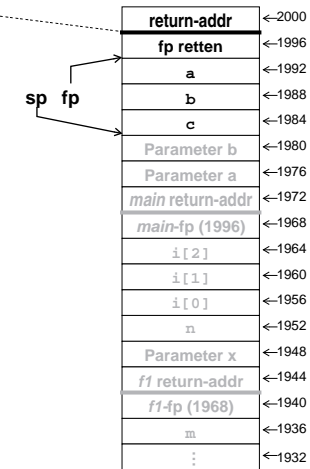
```
int main() {
    int a, b, c;
    a = 10;
    b = 20;
    f1(a, b);
    return(a);
}

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



2 ■ Stack mehrerer Funktionsaufrufe

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



2 ■ Stack mehrerer Funktionsaufrufe

```
int main() {
    int a, b, c;
    a = 10;
    b = 20;
    f1(a, b);
    f3(4, 5, 6);
}
```

was wäre, wenn man nach f1 jetzt eine Funktion f3 aufrufen würde?

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

