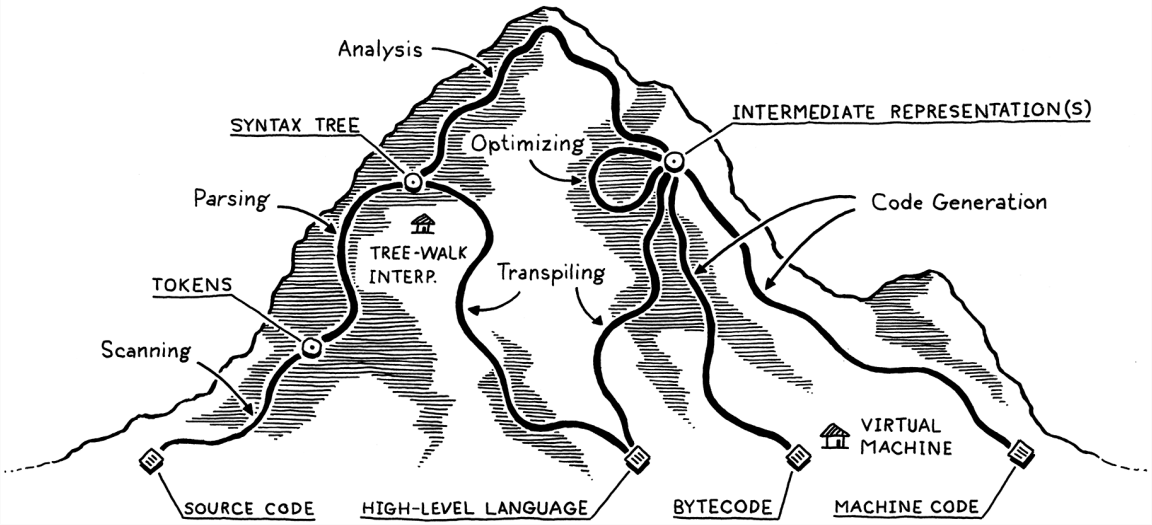
Generierung von Maschinencode (Skizze)

Carsten Gips (FH Bielefeld)

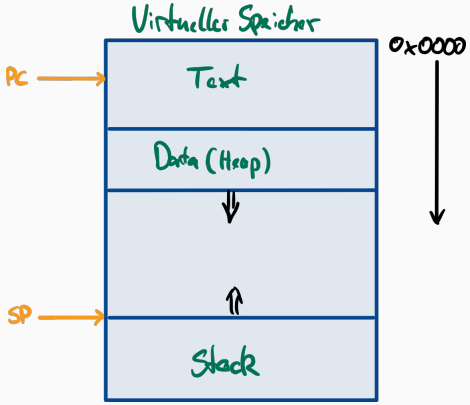
Einordnung



Quelle: "A Map of the Territory (mountain.png)" by Bob Nystrom, licensed under MIT

Prozessorarchitektur

Virtueller Speicher



Befehlszyklus (von-Neumann-Architektur)

```
op = read_next_op(pc)
decode(op)
pc += 1

args = nil
if operands_needed(op):
    args = read_operands(pc)
    pc += 1

execute(op, args)
```

Aufgaben bei der Erzeugung von Maschinen-Code

- Übersetzen des Zwischencodes in Maschinenbefehle
- Sammeln von Konstanten und Literalen am Ende vom Text-Segment

- Auflösen von Adressen:
  - Sprünge: relativ oder absolut
  - Strukturen (Arrays, Structs): Zugriff auf Elemente/Felder über Adresse
  - Zugriffe auf Konstanten oder Literalen: Zugriff auf Text-Segment

- Zuordnung der Variablen und Daten zu Registern oder Adressen
- Aufruf von Funktionen: Anlegen der *Stack-Frames*

- Aufbau des Binärformats und Linking auf der Zielmaschine (auch Betriebssystem) beachten

Übersetzen von Zwischencode in Maschinencode

```
L:  ...
    ...
    if x < v goto L
```

Übersetzen von Zwischencode in Maschinencode

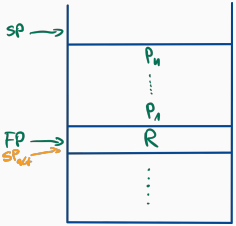
```
L:  ...
    ...
    if x < v goto L
```

```
1000: ...                ;; L
      ...
1080: ldr   r0, x        ;; R0 = x
1088: ldr   r1, v        ;; R1 = v
1096: sub   r0, r0, r1   ;; R0 = R0-R1
1108: bltz  r0, 1000     ;; if R0<0 jump to 1000 (L)
```

Aufruf von Funktionen

```
x = f(p1, ..., pn)
```

```
    FP = SP              ;; Framepointer auf aktuellen Stackpointer setzen
    Stack[SP] = R        ;; Rücksprungadresse auf Stack
    Stack[SP-4] = p1     ;; Parameter p1 auf Stack
    ...
    Stack[SP-4*n] = pn   ;; Parameter pn auf Stack
    SP = SP - 4*(n+1)    ;; Stackpointer auf nächste freie Stelle
    Goto f               ;; Setze den PC auf die Adresse von f im Textsegment
R:  x = Stack[SP+4]      ;; Hole Rückgabewert
    SP = SP + 4          ;; Stackpointer auf nächste freie Stelle
```

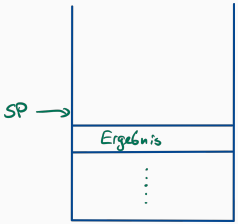


Funktionsaufruf: Prolog

```
f:  ...                 ;; Label f: hier startet die Funktion
    p1 = Stack[SP+4*n]  ;; Zugriff auf p1 (per SP)
    p1 = Stack[FP-4]    ;; Zugriff auf p1 (per FP)
    ...                 ;; hier Funktionskram
```

Funktionsaufruf: Epilog

```
SP = FP - 4              ;; Position über Rücksprungadresse auf Stack
FP = Stack[FP]           ;; Sichere Rücksprungadresse (R)
Stack[SP+4] = Ergebnis   ;; Ergebnis auf Stack (statt Rücksprung-Adresse)
Goto FP                  ;; Setze den PC auf die Rücksprung-Adresse
```



Wrap-Up

Skizze zur Erzeugung von Assembler-Code

- Relativ ähnlich wie die Erzeugung von Bytecode
- Beachtung der Eigenschaften der Zielhardware (Register, Maschinenbefehle, ...)
  - Übersetzen des Zwischencodes in Maschinenbefehle
  - Sammeln von Konstanten und Literalen am Ende vom Text-Segment
  - Auflösen von Adressen
  - Zuordnung der Variablen und Daten zu Registern oder Adressen
  - Aufruf von Funktionen: Op-Codes zum Anlegen der *Stack-Frames*

LICENSE

Exceptions

- Image “A Map of the Territory (mountain.png)” (https://github.com/munificent/craftinginterpreters/blob/master/site/image/a-map-of-the-territory/mountain.png), by Bob Nystrom, licensed under MIT
- Image “Intel i80286 arch” (https://commons.wikimedia.org/wiki/File:Intel_i80286_arch.svg), by Appaloosa, licensed under CC BY-SA 3.0