Systemnahe Programmierung in C (SPiC)

15 μC-Systemarchitektur – Vorbemerkungen

Jürgen Kleinöder, Daniel Lohmann, Volkmar Sieh

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Sommersemester 2022

http://sys.cs.fau.de/lehre/SS22/spic

Was tut ein Compiler?

Aufgabe des Compilers: Zerlegung des Programms in kleine, vom μ-Controller ausführbare Befehle

Beispiel 1: Zerlegung eines Ausdrucks

```
int a, b, c, d;

a = b + c * abs(d - 1);
```

```
int r0, r1, r2, r3;
int a, b, c, d;

r0 = b;
r1 = c;
r3 = d;
r3 -= 1;
r2 = abs(r3);
r1 *= r2;
r0 += r1;
a = r0;
```

`a, b, ...` : „Speichervariablen"

`r0, r1, ...` : „Registervariablen"

Was tut ein Compiler?

Aufgabe des Compilers: Zerlegung des Programms in kleine, vom μ-Controller ausführbare Befehle

Beispiel 2: Zerlegung einer Kontrollstruktur (1. Schritt)

```
if (n != 0) {
    for (i = 0; i != 10; i++) {
        output();
    }
}
```

```
if (n != 0) {
    i = 0;
    while (i != 10) {
        output();
        i++;
    }
}
```


Was tut ein Compiler?

Aufgabe des Compilers: Zerlegung des Programms in kleine, vom μ-Controller ausführbare Befehle

Beispiel 2: Zerlegung einer Kontrollstruktur (2. Schritt)

```
if (n != 0) {
    i = 0;
    while (i != 10) {
        output();
        i++;
    }
}
```

```
if (n != 0) {
    i = 0;
    goto test;
loop:
    output();
    i++;
test:
    if (i != 10) goto loop;
}
```


Was tut ein Compiler?

Aufgabe des Compilers: Zerlegung des Programms in kleine, vom μ-Controller ausführbare Befehle

Beispiel 2: Zerlegung einer Kontrollstruktur (3. Schritt)

```
if (n != 0) {
    i = 0;
    goto test;
loop:
    output();
    i++;
test:
    if (i != 10) goto loop;
}
```

```
    if (n == 0) goto endif;
    i = 0;
    goto test;
loop:
    output();
    i++;
test:
    if (i != 10) goto loop;
endif:
```


Was tut ein Compiler?

Aufgabe des Compilers: Zerlegung des Programms in kleine, vom μ-Controller ausführbare Befehle

Beispiel 2: Zerlegung einer Kontrollstruktur (3. Schritt)

```
    if (n == 0) goto endif;
    i = 0;
    goto test;
loop:
    output();
    i++;
test:
    if (i != 10) goto loop;
endif:
```

```
    r0 = n;
    if (r0 == 0) goto endif;
    r0 = 0;
    i = r0;
    goto test;
loop:
    output();
    r0 = i;
    r0++;
    i = r0;
test:
    r0 = i;
    if (r0 != 10) goto loop;
endif:
```


Was tut ein Compiler?

Aufgabe des Compilers: Zerlegung des Programms in kleine, vom μ-Controller ausführbare Befehle:

- `rN = const;`
- `rN = var;`
- `rN op= const;`
- `rN op= rN;`
- `rN = func(...);`
- `var = rN;`
- `goto label;`
- `if (rN op const) goto label;`
- `if (rN op rM) goto label;`
- `return rN;`
- ...

Was tut ein Compiler?

Typische, vom μ-Controller ausführbare Befehle (Beispiele):

C-Code	Mnemonic	
`rN++;`	`inc rN`	increment
`rN--;`	`dec rN`	decrement
`rN = const;`	`ldi rN, const`	load immediate
`rN = var;`	`ld rN, var`	load
`rN += const;`	`addi rN, const`	add immediate
`rN -= const;`	`subi rN, const`	subtract immediate
`rN += rM;`	`add rN, rM`	add
`rN -= rM;`	`sub rN, rM`	sub
`rN = func();`	`call func`	call function
`var = rN;`	`st var, rN`	store
`goto label;`	`jmp label`	jump
`if (rN == rM) goto label;`	`cmp rN, rM` `beq label`	compare branch if equal
...	...	

Vorhandene Befehle siehe Handbuch zum Prozessor/μ-Controller.

Was tut ein Compiler?

Beispielprogramm:

vereinfachter C-Code	Assembler-Code
`r0 = n;`	`ld r0, n`
`if (r0 == 0) goto endif;`	`cmpi r0, 0`
	`beq endif`
`r0 = 0;`	`ldi r0, 0`
`i = r0;`	`st i, r0`
`goto test;`	`jmp test`
`loop: output();`	`loop: call output`
`r0 = i;`	`ld r0, i`
`r0++;`	`inc r0`
`i = r0;`	`st i, r0`
`test: r0 = i;`	`test: ld r0, i`
`if (r0 != 10) goto loop;`	`cmpi r0, 10`
	`bneq loop`
`endif:`	`endif:`

Was tut ein Compiler?

Beispielprogramm:

vereinfachter C-Code		Assembler-Code	
	uint8_t n;	10	
	uint8_t i;	11	
	...	...	
	r0 = n;	20	ld r0, 10
	if (r0 == 0) goto endif;	21	cmpi r0, 0
		22	beq 33
	r0 = 0;	23	ldi r0, 0
	i = r0;	24	st 11, r0
	goto test;	25	jmp 30
loop:	output();	26	call 70
	r0 = i;	27	ld r0, 11
	r0++;	28	inc r0
	i = r0;	29	st 11, r0
test:	r0 = i;	30	ld r0, 11
	if (r0 != 10) goto loop;	31	cmpi r0, 10
		32	bneq 26
endif:		33	
	...	...	...
	output(...)	70	...

Was tut ein Assembler?

Beispielprogramm:

Assembler-Code		Binär-Code
...		...
20	ld r0, 10	0a4f
21	cmpi r0, 0	a77f
22	beq 33	77bc
23	ldi r0, 0	87ee
24	st 11, r0	7439
25	jmp 30	30af
26	call 70	dd33
27	ld r0, 11	75ca
28	inc r0	9e88
29	st 11, r0	11f2
30	ld r0, 11	ad8f
31	cmpi r0, 10	54e1
32	bneq 26	98e4
...	...	...

Codierung der Befehle siehe Handbuch zum Prozessor/μ-Controller.

Program Counter / Instruction Pointer

Program Counter (PC) oder Instruction Pointer (IP):
Register, das die Nummer der Speicherzelle enthält, die den nächsten auszuführenden Befehl enthält

```
PC = 24
```

```
                ...
                21   cmpi r0, 0
                22   beq 33
                23   ldi r0, 0
        PC -->  24   st 11, r0
                25   jmp 30
                26   call 70
                27   ld r0, 11
                ...  ...
```


Bemerkungen

Diese Folien

- sind *wichtig für das Verständnis* der nächsten Vorlesungen
  - C-Code wird vom C-Compiler in kleinere Einheiten zerlegt
  - kleinere Einheiten können in Befehle für den μ-Controller übersetzt werden
  - Befehle werden vom Assembler in binären Code übersetzt
  - Befehle werden vom μ-Controller gemäß dem Program Counter Schritt-für-Schritt abgearbeitet
- sind *nicht Prüfungs-relevant*