System-Level Programming

15 μC-System Architecture – Preface

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

What does a Compiler do?

Job of the compiler: decomposition of the program into smaller instructions that can be executed by a μController

Example 1: decomposition of an expression

```
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, ...` : “variables in memory”

`r0, r1, ...` : “variables in regsiters”

What does a Compiler do?

Job of the compiler: decomposition of the program into smaller instructions that can be executed by a μController

Example 2: decomposition of a control structure (1st part)

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

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

What does a Compiler do?

Job of the compiler: decomposition of the program into smaller instructions that can be executed by a μController

Example 2: decomposition of a control structure (2nd part)

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

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

What does a Compiler?

Job of the compiler: decomposition of the program into smaller instructions that can be executed by a μController

Example 2: decomposition of a control structure (3rd part)

```
    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:
```

What does a Compiler?

Job of the compiler: decomposition of the program into smaller instructions that can be executed by a μController

Example 2: decomposition of a control structure (3rd part)

```
    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:
```

What does a Compiler do?

Job of the compiler: decomposition of the program into smaller instructions that can be executed by a μController

- `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;`
- ...

What does a Compiler do?

Typical instructions that a μController can execute (examples):

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

...	...	

All available instructions: see manual of processor/μController.

What does a Compiler do?

Example program:

	simplified 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:`	

What does a Compiler do?

Example program:

simplified 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	...

What does an Assembler do?

Example program:

assembler code		binary 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
...	...	...

Encoding of the instructions is listed in μController's manual.

Program Counter / Instruction Pointer

Program counter (PC) or instruction pointer (IP):
The register that contains the value of the memory cell holding the instruction that has to be executed next

```
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
                   ...  ...
```

Comments

These slides

- are *important for understanding* the next lectures
  - C code is decomposed into smaller pieces by the compiler
  - smaller pieces are translated into instructions for the μController
  - instructions are encoded in binary code by the assembler
  - instructions are executed step by step by the μController, depending on the PC
- are *not relevant for the exam*