Institute for Automation of Complex Power Systems

E.ON Energy Research Center

Aachen, den 27.6.2019

Dr. rer. nat. Stefan Lankes
Steffen Vogel, M. Sc.

KGÜ 5 Assembler

Inhalte: Assembler, Aufbau des Objektformates
Lernziele: Assemblieren eines gegebenen Programms

Aufgabe 1: Die Erzeugung des Objektcodes

In dieser Aufgabe sollen Sie die Arbeit eines Assemblers anhand des folgenden Beispielprogramms für einen vereinfachten Rechner nachvollziehen, der dem x86 ähnlich ist. Das Befehlsformat ist gegenüber dem x86 wie folgt vereinfacht und besteht aus folgenden Elementen:

- OpCode (1 Byte)
- Register- und Adressmodusbezeichner des 1. Operanden (1 Byte, optional)
- Register- und Adressmodusbezeichner des 2. Operanden (1 Byte, optional)
- SI-Byte (Abkürzung steht für Scale und Index, 1 Byte, optional)
- Displacement (4 Bytes, optional)
- Immediate (4 Bytes, optional)

Anhand der Anweisung `mov edx, [ebx+8*ecx]` wird die Bildung des Maschinencodes erläutert. Der Opcode ist mit Hilfe der Tabelle 1 zu bestimmen und ist in diesem Fall `0C`. Für die beiden Operanden muss anschließend jeweils ein Register- und Adressmodusbezeichner bestimmt werden, die jeweils in einem Byte codiert werden. Die Bits 0-3 des Bezeichners spezifizieren den Adressmodus und sind ebenfalls aus der Tabelle 1 zu entnehmen. Der erste Operand stellt ein Register dar, so dass die Bits 0-3 den Wert `0` annehmen. Die Bits 4-7 beschreiben, welches Register als Basis für diese Operation verwendet wird und nimmt in diesem Fall den Wert `3` an, der für das Register `edx` steht. Somit besitzt der Register- und Adressmodusbezeichner des ersten Operanden den Wert `30`. Der zweite Operand verwendet als Adressierungsmodus „indirekt indiziert“, so dass die Bits 0-3 des Register- und Adressmodusbezeichners den Wert `4` annehmen. Die Bits 4 bis 7 nehmen den

Wert 1 an, der für das Register ebx steht. Somit besitzt der Register- und Adressmodusbezeichner des zweiten Operanden den Wert 14. Anschließend müssen im SI-Byte sowohl das Index-Register als auch der Skalierungsfaktor des zweiten Operanden codiert werden. Bits 0-3 stehen hierbei für das Index-Register und nehmen somit den Wert 2 an. Die Bits 4-7 beschreiben den Skalierungsfaktor und nehmen hier den Wert 8 an. Somit ergibt sich für die Anweisung `mov edx, [ebx+8*ecx]` folgender Maschinencode: `0C301482`.

```
SECTION .text
 extern printf
 extern init_bild
 global Invert
 global main

 Invert:
   push ebp
   mov ebp, esp
   push ebx
   push msg
   call printf
   add  esp, 4
   mov  eax, [ebp+16]
   imul eax, [ebp+12]
   imul eax, 3
   mov  ecx, 0
   mov  ebx, [ebp+8]

  anfang:
     cmp ecx, eax
     jge ende
     not BYTE [ebx + ecx]
     inc ecx
     jmp anfang
   ende:
     pop ebx
     pop ebp
    ret
```

```
main:
 push ebp
 mov ebp, esp

 mov eax,  HOEHE
 imul eax, BREITE
 imul eax, 3
 push eax
 push bild
 call init_bild
 add esp,8

 push BREITE
 push HOEHE
 push bild
 call Invert
 add esp, 12

 pop ebp

mov eax,1
mov ebx,0
int 0x80

SECTION .data
  bild RESB 48
  HOEHE equ 4
  BREITE equ 4

  msg db "Invertiere Bild...",13,10,0
```

a) Erläutern Sie *kurz* die Funktionsweise des Beispielprogrammes.

b) Vollziehen Sie die Phase 1 eines Assemblers wie in der Vorlesung nach. Geben Sie die Pseudooperationen und Maschinenoperationen an. Errechnen Sie die Länge der einzelnen Befehle in Maschinencode und geben Sie diese an. Schreiben Sie zu jeder Zeile den dazugehörigen Locationcounter. Generieren Sie die Symboltabelle des As-

semblers, und geben Sie die Länge der Code- und Datensektion an. Die Datensektion kann an einer beliebigen Adresse beginnen.

c) Vollziehen Sie nun die Phase 2 des Assemblers nach, indem Sie ein Objekt im vorgestellten Objektformat erzeugen. Zur Vereinfachung können alle Zahlen im „Big Endian-Format“ dargestellt werden. Außerdem seien alle Sprünge (jumps) relativ und alle Funktionsaufrufe (calls) absolut! Kodieren Sie genau drei Befehle in einem T-Datensatz (soweit möglich)! Nutzen Sie den S-Datensatz! Gehen Sie davon aus, dass der Dateiname und damit auch der Modulname „testInvert“ lautet!

Opcode

Mnemonic	OpCode
add	01
sub	02
inc	03
dec	04
push	05
pop	06
jmp	07
call	08
ret	09
cmp	0A
jge	0B
mov	0C
imul	0D
loop	0E
not	0F
int	CD

Adressmodus (4 Bits)

Adressierungs-art	Adress-modus	Beispiele
register	0	mov eax,...
immediate	1	push 2000
direkt	2	mov [2000],...
indirekt	3	mov [ebx],...
indirekt indiziert	4	mov [ebx+8*eax],...
indirekt mit displacement	5	mov [eax-4],...
indirekt indiziert mit displacement	6	mov [eax+8*ebx-4],...

Register (4 Bits)

Register	Bezeichner
eax	0
ebx	1
ecx	2
edx	3
ebp	4
esp	5
edi	6
esi	7
ax	8
al	9
ah	A
bx	B
bl	C
bh	D

Tabelle 1: Hilfstabellen zur Bestimmung des Opcodes sowie des Register- und Adressmodusbezeichners