Betriebssysteme 1

Foliensatz C, Kap. 2 und 3

Prof. Dr. Hans-Georg Eßer

Sommersemester 2022

v3.0 – 29.04.2021

Kernel Mode

- Wechsel vom User Mode in den Kernel Mode, vgl. Skript S. 48

User Mode

- Der *User Mode* ist die Standardbetriebsart für Prozesse. Es ist nur der dem Prozess zugeteilte Speicher sichtbar, und die Nutzung privilegierter Maschinensprachebefehle, über die sich etwa die Hardware steuern ließe, ist verboten.

Kernel Mode

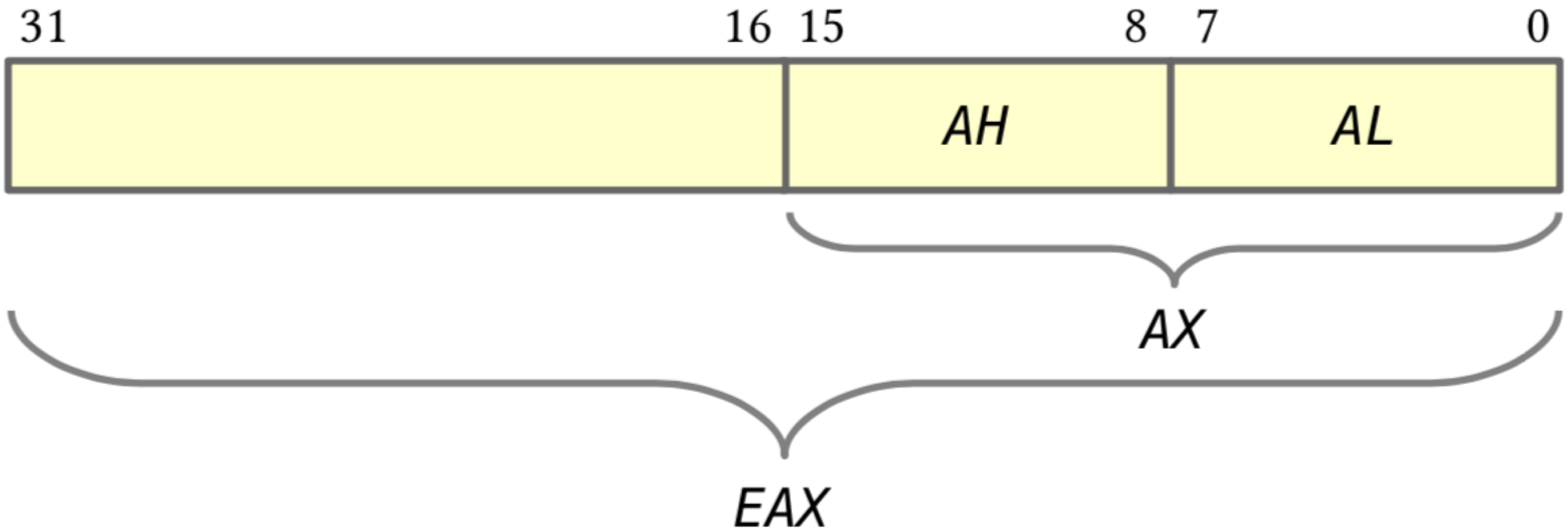
- Im *Kernel Mode* erlangt der Prozess zusätzliche Möglichkeiten: Erstens sieht er (neben seinem eigenen Speicher) nun auch den Kernel-Speicher (also Code und Daten des Betriebssystems), und zweitens sind nun die privilegierten Maschinensprachebefehle erlaubt.

Wenn ein Prozess vom User Mode in den Kernel Mode wechselt, arbeitet er also mit den Rechten des Betriebssystems und hat Vollzugriff auf die Maschine. Darum muss dieser Übergang streng kontrolliert werden – dafür sind die System Calls zuständig. Moderne

Register

- CPU-Register:
  - General-Purpose-Register
  - Intel x86: 32-Bit-Register EAX, EBX, ...

64 bit: RAX, RBX, RCX, …



System Call in Assembler

in C: write (1, msg, len);

bs1-syscall.asm

(Skript S. 48)

```
global _start                   ; für den Linker (ld)
section .text
_start:                         ; für Linker (wo gehts los)
        mov     eax, 4          ; Syscall-Nummer (__NR_write)
        mov     ebx, 1          ; file descriptor (1=stdout)
        mov     ecx, msg        ; Adresse der Nachricht
        mov     edx, len        ; Nachrichtenlänge
        int     0x80            ; Syscall ausführen
        mov     eax, 1          ; Syscall-Nummer (__NR_exit)
        int     0x80            ; Syscall ausführen

section .data
msg     db      'Hallo Welt!',0xa   ; Text
len     equ     $ - msg             ; Länge
```

1. Syscall (write)

2. Syscall (exit)

Makefile

```
bs1-syscall: bs1-syscall.asm
        nasm -f elf bs1-syscall.asm
        ld -o bs1-syscall bs1-syscall.o
```

System Call in Assembler: Praxis in der VM

- Archiv herunterladen:

```
wget http://swf.hgesser.de/bs-b1/code/bs1-syscall.zip
```

- Entpacken, mit make bauen, ausführen:

```
student@swfdebian:~/asm$ unzip bs1-syscall.zip
Archive:  bs1-syscall.zip
   creating: bs1-syscall/
  inflating: bs1-syscall/bs1-syscall.asm
  inflating: bs1-syscall/Makefile
student@swfdebian:~/asm$ cd bs1-syscall
student@swfdebian:~/asm/bs1-syscall$ make
nasm -f elf bs1-syscall.asm
ld -o bs1-syscall bs1-syscall.o
student@swfdebian:~/asm/bs1-syscall$ ./bs1-syscall
Hallo Welt!
```

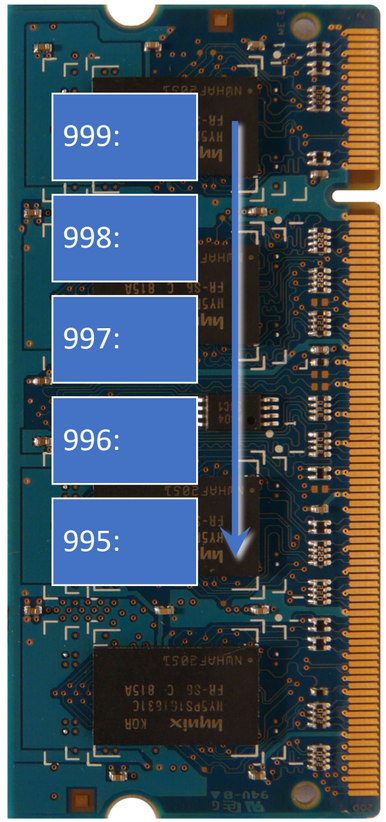
Wichtig: Unterschied „Function Call“ vs. „System Call“

- Function Call
  - Argumente auf den Stack
  - Aufruf (Call) der **Startadresse** der Funktion (schreibt Rücksprungadresse auf Stack)
  - In Funktion: Zugriff auf Argumente, Aufgabe bearbeiten
  - Rücksprung mit `ret`
- System Call
  - **Syscall-Nummer** in Register EAX (Linux, x86)
  - Argumente in Register (Linux, x86)
  - System Call (`int 0x80`, `syscall`, `sysenter` etc.) aktiviert Kernel Mode, springt an Adresse im Kernel, sichert automatisch die Rücksprungadresse
  - Im Kernel: Syscall-Nummer aus EAX auswerten, Syscall-Handler aufrufen
  - Rücksprung mit `iret`

Parameter-Übergabe via Stack oder Register

Parameter
in Registern

Parameter
auf dem Stack



Kap. 2: Lernkontrollfragen

Fragen: Skript, S. 54

10. Super System Call

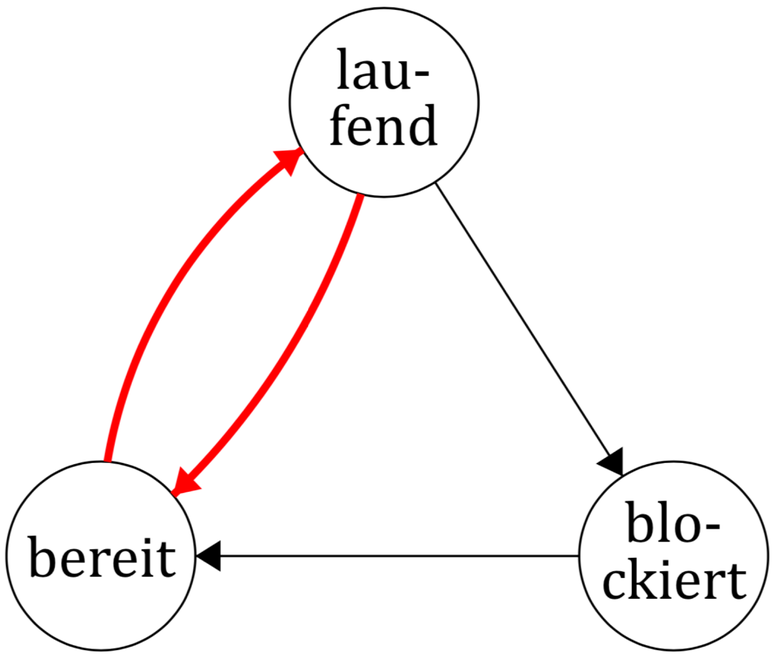
– EBX: Adresse eines Strings `functionname`
– ECX: Adresse eines Arrays mit Parametern
– Kernel sucht Adresse zu `functionname` und ruft die Funktion auf

11. Priorität

Warum Priorität nur verringern (und nicht erhöhen)?

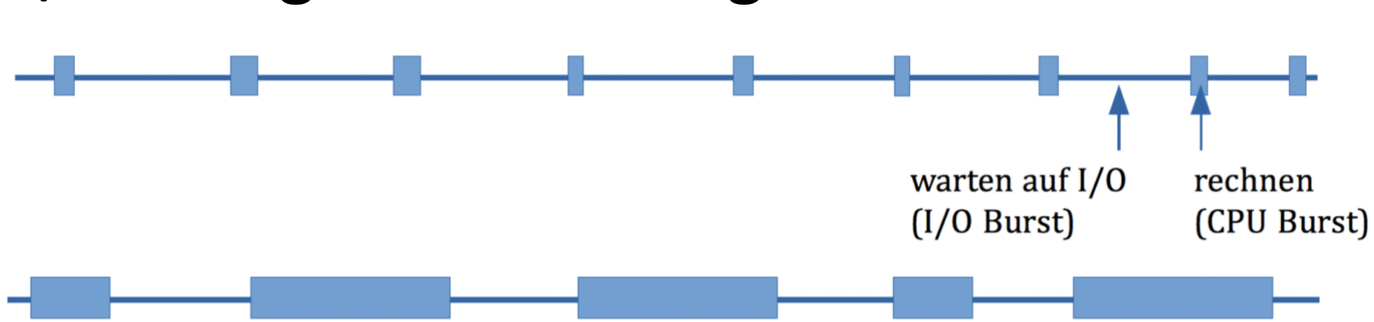
12. User Mode vs. Kernel Mode

Ist `cli` (clear interrupt flag) ein privilegierter Befehl?



Begriffe

- kooperativ vs. präemptiv
- I/O-lastig vs. CPU-lastig



Wann wird Scheduler aktiv?

- Neuer Prozess entsteht (`fork`)
- Aktiver Prozess blockiert wegen I/O-Zugriff
- Blockierter Prozess wird bereit
- Aktiver Prozess endet (`exit`)
- Prozess rechnet schon zu lange
- Interrupt (von angeschlossenem / integriertem Gerät) tritt auf

Scheduling-Ziele

- **[A1] Ausführdauer:** Wie lange läuft der Prozess insgesamt?
- **[A2] Reaktionszeit:** Wie schnell reagiert der Prozess auf Benutzerinteraktion?
- **[A3] Deadlines** einhalten
- **[A4] Vorhersehbarkeit:** Gleichartige Prozesse sollten sich auch gleichartig verhalten, was obige Punkte angeht
- **[A5] Proportionalität:** „Einfaches“ geht schnell

A = Anwendersicht

- **[S1] Durchsatz:** Anzahl der Prozesse, die pro Zeit fertig werden
- **[S2] Prozessorauslastung:** Zeit (in %), die der Prozessor aktiv war
- **[S3] Fairness:** Prozesse gleich behandeln, keiner darf „verhungern“
- **[S4] Prioritäten** beachten
- **[S5] Ressourcen** gleichmäßig einsetzen

S = Systemsicht