ninjastorms

Episode II

Was hatten wir vor?

- **Ziel:** eigenen präemptiven RT-Scheduler bauen und RT-Demo vorführen
- **Aufgaben:**
  - Task-Konzept einführen
  - Zustand eines Tasks speichern/wiederherstellen
  - Rudimentäres Memory Management
  - Debugging mittel Emulation (qemu)
  - Timer-Interrupts zur Präemption
  - Interrupt Service Routine
  - Scheduling-Algorithmus

Was hatten wir vor?

- **Ziel:** eigenen präemptiven RT-Scheduler bauen und RT-Demo vorführen
- **Aufgaben:**
  - Task-Konzept einführen
  - Zustand eines Tasks speichern/wiederherstellen
  - Rudimentäres Memory-Management
  - Debugging mit Emulation (qemu)
  - Timer-Interrupt zur Präemption
  - Interrupt Service Routine
  - Scheduling-Algorithmus

Check!

Real-Time Demo™

Scheduler: API

```
49 int ev3ninja_main (void)
50 {
54   add_task(&controller_task);
55   start_scheduler();
```

```
10 void func_task_a(void) {
11   while(1) {
12     control_motor();
13   }
14 }
15
16 void func_task_b(void) {
17   while(1) {
18     lightshow();
19   }
20 }
21
22 void controller task(void) {
23   ...
47 }
```

```
add_task(func_task_a);
[…]
add_task(func_task_b);
```

Scheduler: Funktionsweise (1)

Hinzufügen von Tasks:

- `add_task()` weist STACK_BASE zu und ruft `init_task()`
- `init_task()` initialisiert `task_struct` und nullt es

```
1 typedef struct task {
2         unsigned int reg[16];
3         unsigned int cpsr;
4 } task_t;
```

- `task_struct` wird in den Ringpuffer eingefügt
- MAX_TASK_NUMBER zurzeit 16

Scheduler: Funktionsweise (2)

Start des Schedulers:

```
122 void start_scheduler() {
123   if (!isRunning) {
124     current_task = ring_buffer_remove();
125     isRunning = 1;
126     stop_timer();
127     init_interrupt_handling();
128     init_timer();
129     load_current_task_state();
130   }
131 }
```

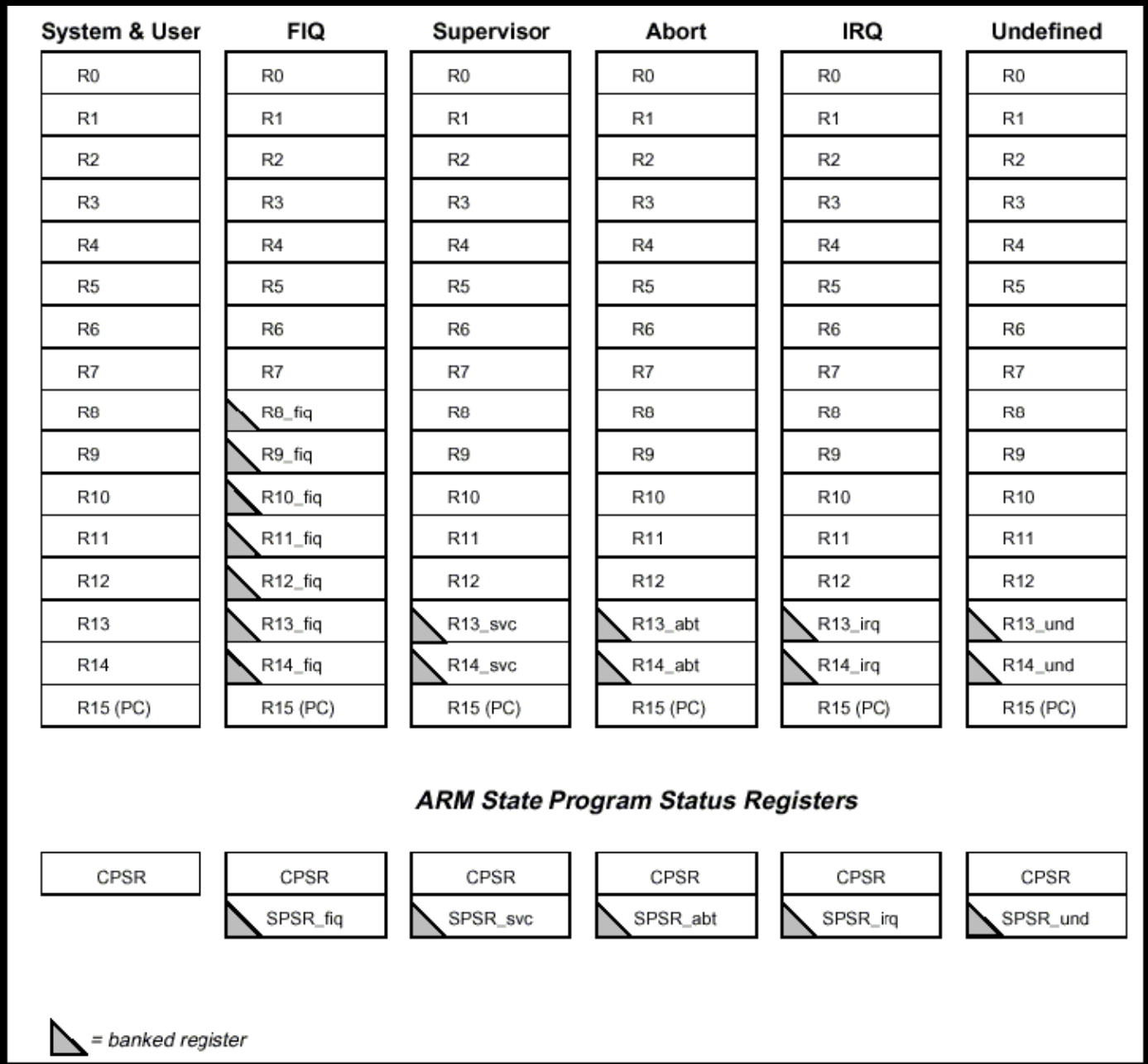
Scheduler: Funktionsweise (3)

```
16 irq_handler:
17   push  {r0-r12, lr}
18
19   bl  save_current_task_state
20   bl  schedule
21
22   // clear interrupt @ timer
23   ldr r0, =TIMER0_INTCTLSTAT_ASM
24   mov r1, CLEARTIMER34_ASM
25   str r1, [r0]
26
27   // clear interrupt @ interrupt controller
28   ldr r0, =AINTC_SECR1_ASM
29   mov r1, T64P0_TINT34_ASM
30   str r1, [r0]
31
32   pop  {r0-r12, lr}
33
34   b  load_current_task_state
```

Scheduler: Funktionsweise (4)

Kontextwechsel:

- Register im Memory abspeichern ohne Register zu überschreiben
  –> Register pushen und anschließend in den Memory schieben
- Aber: Was ist mit **Banked Registers?**


System & User
FIQ
Supervisor
Abort
IRQ
Undefined
R0
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15 (PC)
R8_fiq
R9_fiq
R10_fiq
R11_fiq
R12_fiq
R13_fiq
R14_fiq
R13_svc
R14_svc
R13_abt
R14_abt
R13_irq
R14_irq
R13_und
R14_und
ARM State Program Status Registers
CPSR
SPSR_fiq
SPSR_svc
SPSR_abt
SPSR_irq
SPSR_und
= banked register

Scheduler: Funktionsweise (4)

Kontextwechsel:

- Register im Memory abspeichern ohne Register zu überschreiben
  –> Register pushen und anschließend in den Memory schieben

- Aber: Was ist mit **Banked Registers?**

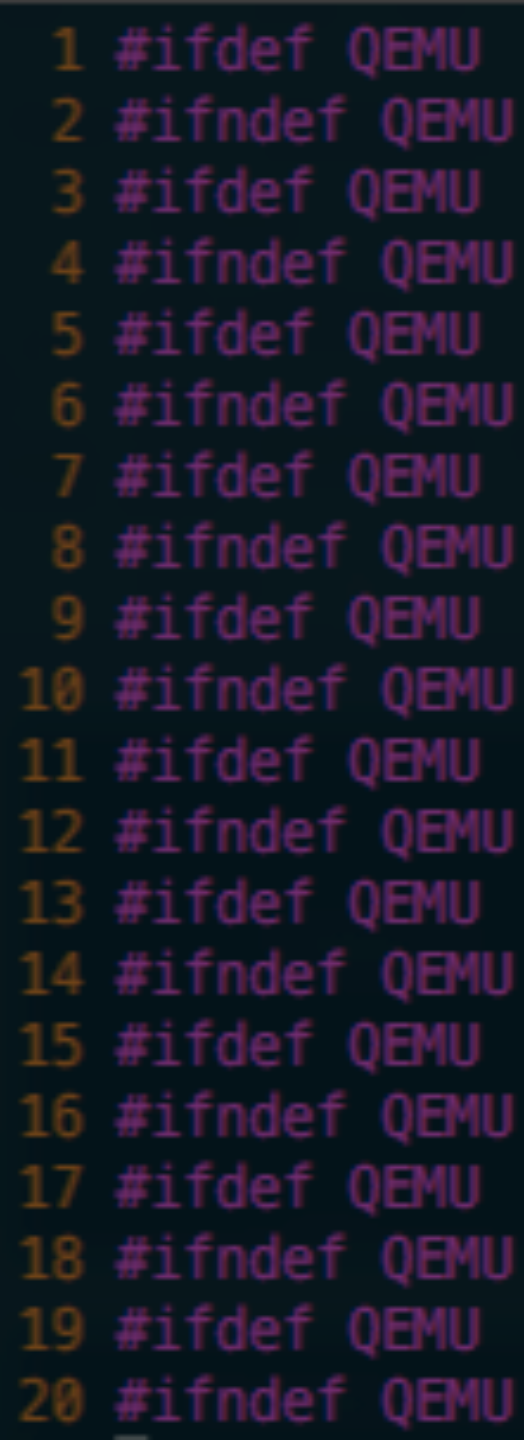
```
50    // save user mode sp and lr to the struct
51    stm  r0, {sp, lr}^
```

EMULATION

- **Board**: versatilepb
- **CPU**: ARM926EJ-S
- **Timer**: ARM SP804GDB
- **Int. Controller**: ARM PL190
- GDB-Stub zum Debugging
- Memory beginnt bei `0x0`

Realität

- **Board**: TI omapl138
- **CPU**: ARM926EJ-S
- **Timer**: 64-Bit Timer Plus
- **Int. Controller**: AINTC
- UART / printf-Debugging
- Memory beginnt bei `0xFFFF0000`



Was haben wir noch vor?

- Architektur-spezifischen Code besser kapseln (DaVinci vs. qemu)
- Dokumentation

Future Work

- Memory Management (Heap?)
- Deadlines, Prioritäten, weitere Scheduling-Algorithmen

ninjastorms

Episode II

Bonus-Menü

`load_current_task_state()`	8 Zeilen Code, ARM Assembly
`save_current_task_state()`	22 Zeilen Code, ARM Assembly
`schedule()`	2 Zeilen Code, C
Christians `.vimrc`	50 Zeilen

Bonus: schedule()

```
77 void schedule(void) {
78   ring_buffer_insert(current_task);
79   current_task = ring_buffer_remove();
80 }
```

Bonus: save_current_task_state()

```
37 save_current_task_state:
38   // save r0-r12
39   ldr  r4, =current_task // load current_task
40   ldr  r0, [r4]          // dereference current_task, to get the task_struct
41   mov  r2, sp            // save sp to iterate over it
42   mov  r3, #13           // iteration count; r0-r12 is 13 registers, so while i<13
43 save_registers_loop:
44   ldr  r1, [r2], #4      // load saved register from stack
45   str  r1, [r0], #4      // save it to the struct
46   subs r3, #1            // i--
47   bne  save_registers_loop
48   // save sp, lr
49   stm  r0, {sp, lr}^   // save user mode sp and lr to the struct
50   // save pc
51   ldr  r1, [r2]          // load saved lr from stack
52   sub  r1, #4            // because lr is the old pc+4
53   str  r1, [r0, #8]      // save r1 to the pc field in the task_struct
54   // save cpsr
55   mrs  r1, spsr          // save cpsr of the current task (in spsr atm)
56   str  r1, [r0, #12]     // r0 still points to sp, so skip sp, lr, pc (i.e. 12)
57
58   bx   lr
```

Bonus: `load_current_task_state()`

```
62 load_current_task_state:
63   ldr  r4, =current_task // load current_task
64   ldr  r0, [r4]          // dereference current_task, to get the task_struct
65   ldr  r5, [r0, #64]     // load cpsr from task_struct to r5 (i.e. task_struct+64)
66   msr  spsr, r5          // copy r5 to spsr
67   add  lr, r0, #60       // load address of saved pc into lr
68   ldm  r0, {r0-r14}^     // load saved registers into user mode registers ((do not) trust the caret!)
69   ldm  lr, {pc}^         // return to loaded task and restore cpsr from spsr
```

Bonus: Christians .vimrc

```
set nocompatible              " be iMproved, required
filetype off                  " required

set rtp+=/Users/Christian/Library/python2.7/site-packages/powerline/bindings/vim/

" Always show statusline
set laststatus=2
"
" " Use 256 colours (Use this setting only if your terminal supports 256
" colours)
set t_Co=256

" set the runtime path to include Vundle and initialize
set rtp+=~/.vim/bundle/Vundle.vim

call vundle#begin()

Plugin 'gmarik/Vundle.vim'

Plugin 'altercation/vim-colors-solarized'

" Plugin 'vim-scripts/LanguageTool'
" Plugin 'moll/vim-bbye'
" Plugin 'othree/javascript-libraries-syntax.vim'
" Plugin 'scrooloose/nerdtree'
" Plugin 'godlygeek/tabular'
" Plugin 'bling/vim-airline'
" Plugin 'tpope/vim-fugitive'
" Plugin 'terryma/vim-multiple-cursors'
" Plugin 'mbbill/undotree'
" Plugin 'tpope/vim-repeat'
Plugin 'powerline/powerline', {'rtp': 'powerline/bindings/vim/'}
" Plugin 'edkolev/tmuxline.vim'
" Plugin 'svermeulen/vim-easyclip'
" Plugin 'dhruvasagar/vim-table-mode'

call vundle#end()            " required

syntax on
set number


" colorscheme solarized
" set background=dark
" let g:solarized_termcolors = 256
" colorscheme solarized

filetype plugin indent on    " required

" autocmd vimenter * NERDTree
```