Gliederung der Vorlesung

1. Einleitung
2. Power Management
3. Parallele I/O
4. Serielle Schnittstelle
   - I^2C
   - USART
5. Timer Bausteine
6. Interrupt Handling
7. Signalverarbeitung
8. Regelungen
9. Software Interrupt
10. Leitungscodes
11. Weiterführende Themen

Software Interrupt – Aufgaben I

- Aufgabe des Software Interrupt
- Der Software Interrupt ist eine benutzerdefinierte, synchrone Programmunterbrechung um aus dem User Mode in den privilegierten Mode zu gelangen.
- Der privilegierte Mode wird of auch Supervisor Mode genannt.
- Im privilegierten Mode können privilegierte Operationen ausgeführt werden, die im User Mode nicht erlaubt sind
- Der privilegierte Mode wird oft genutzt, um darin ein Betriebssystem laufen zu lassen.
- Die Hardware bietet so den für das Betriebssystem nötigen Schutz vor Anwendungen im User Mode.

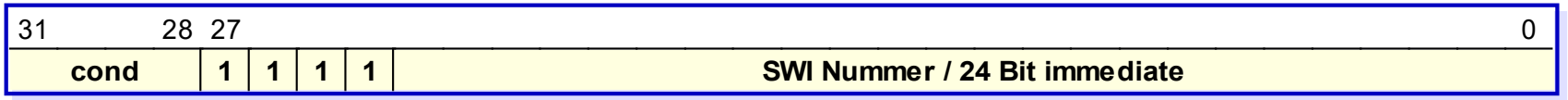
Software Interrupt – Aufgaben II

- In vielen Systemen ist der Zugriff auf den Speicher-Bereich, in dem die Peripheriebausteine für I/O eingeblendet sind, nur in einem privilegierten Mode (Privileged Mode oder Handler Mode) erlaubt.
- Der Schreibzugriff auf Teile des Prozessor-Statusworts ist geschützt und kann nur in einem privilegierten Mode ausgeführt werden.
- Ein Wechsel des Mode der CPU oder das Sperren von Interrupts ist daher im User Mode nicht möglich,
- Der Speicherbereich, in dem sich die Vektortabelle des Prozessors befindet ist in vielen Systemen geschützt und kann im User Mode nicht beschrieben werden,

Software Interrupt

- Eine Memory Protection Unit (MPU) oder Memory Management Unit MMU ermöglicht die Unterteilung des Speicherbereichs in geschützte Bereiche bzw. erlaubt die Unterteilung in Seiten mit virtuellen Adressen.
- Der Zugriff auf eine MPU oder MMU (falls vorhanden) ist nur im privilegierten Mode möglich.
- Nicht alle Mikrocontroller unterstützen einen Software Interrupt:
  - **Aufruf Cortex M3**
    - **Instruktion: SVC #3 ; SVC: Supervisor Call**
  - **Aufruf ARM7TDMI**
    - **Instruktion: SWI #3 ; SWI: Software Interrupt**
  - Cortex M0+
    - Unterstützt keinen Softwareinterrupt mit einem dedizierten Maschinenbefehl

SWI – Syntax ARM7TDMI



- Syntax:
  - **swi{<cond>} <SWI Nummer>**

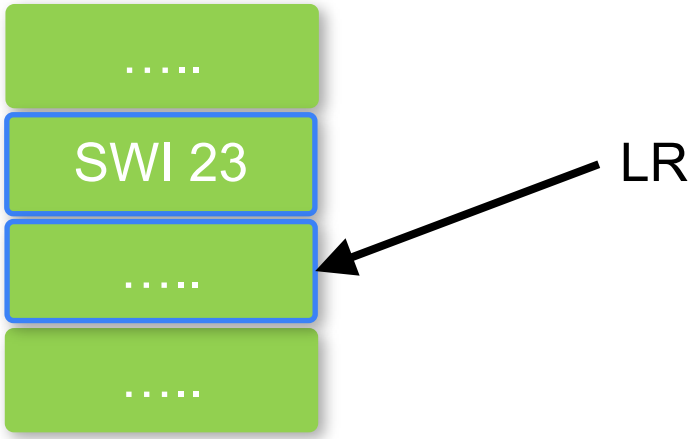
- Der SWI Interrupt Handler kann die SWI Nummer lesen und damit entscheiden, welche Operation ausgeführt werden soll

- Der Software Interrupt setzt LR_SVC auf PC-4; das ist der Befehl nach dem Softwareinterrupt. Durch einen Zugriff auf LR-4 kann man daher den Befehlscode des SWI in ein Register laden und die SWI Nummer extrahieren

SWI – Zugriff ARM7TDMI

Zugriff auf die Nummer des SWI Befehls

- ldr r0, [lr, #-4] @ Befehlscode wird geladen
- bic r0, r0, #0xff000000 @ Obere 8 Bits werden maskiert



SWI – ein einfacher SWI Handler in ASM

```
    .global SWIHandler
    .text
SWIHandler:
    stmfd   sp!, {lr}           @ Retten von Ruecksprungadresse
    ldr     ip, [lr, #-4]       @ Laden des Programmcodes
    bic     ip, ip, #0xff000000 @ Ausmaskieren der SWI Nummer
    ldr     lr, =SWIJumpTable
    ldr     ip, [lr, ip, LSL #2] @ Laden der Funktionsadresse nach ip
    mov     lr,  pc             @ indirekter Unterprogrammaufruf
    mov     pc, ip
    ldmfd   sp!, {pc}^
SWIJumpTable:
    .word   putchar
    .word   getchar
    .word   init_ser
.end
```

SWI – ein einfacher SWI Handler in C

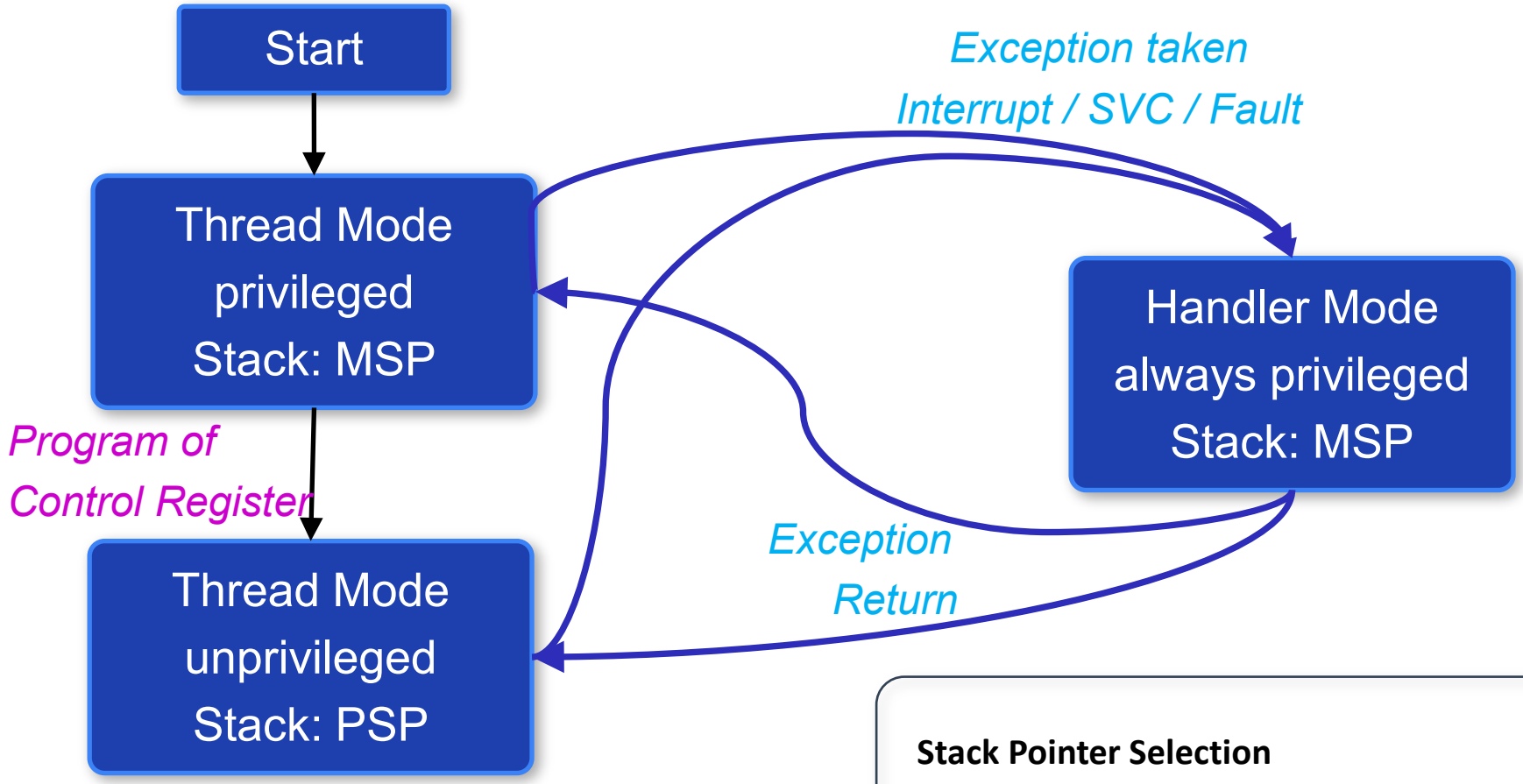
```
void SWIHandler () __attribute__ ((interrupt ("SWI")));

register int *reg_14 asm ("r14");  // link register (SVC) is mapped in register bank

void SWIHandler() {
    switch( *(reg_14 - 1) & 0x00FFFFFF) // Masking of the top 8 Bits and branching
                                        // depending on SWI number
    {
    case 0:
            function0(); break;
    case 1:
            function1(); break;
    case 2:
            function2(); break;
    }
}
```

```
int main(void) {
   ....
   asm("swi 0");
   ....
}
```

ARM Cortex-M3 — Operating Modes



Stack Pointer Selection

MSP – Main Stack Pointer

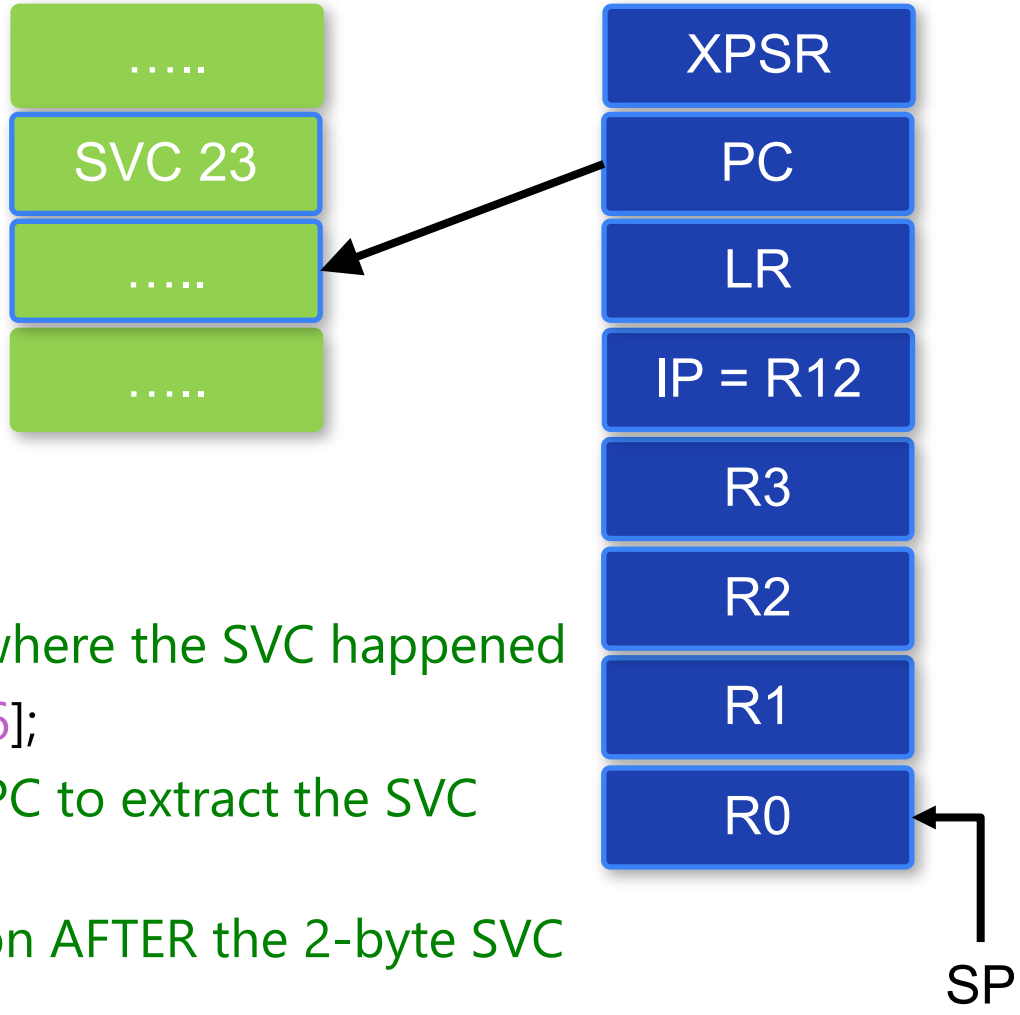
Handler Mode, Thread Priv. (default)

PSP – Process Stack Pointer

Thread Mode (Unprivileged / RTOS Tasks)

ARM Cortex-M3 – Eintritt Interrupt SVC

Ermitteln der SVC Nummer:




```
// 1. Get the Program Counter (PC) where the SVC happened
unsigned int stacked_pc = svc_args[6];
// 2. Read the byte right before the PC to extract the SVC number
// stacked_pc points to the instruction AFTER the 2-byte SVC instruction
unsigned char svc_number = ((unsigned char *)stacked_pc)[-2];
```

SWI – Beispiel: Linux Kernel-Entry

Suchen Sie auf https://www.kernel.org/

Ansichten: Release (z.B. stable) browse -> tree

Verzeichnis: arch/arm/kernel

In folgender Datei findet man den SWI-Handler.

entry-common.S

https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/arch/arm/kernel/entry-common.S