Exception & ISR

Purpose

Learn how exceptions works in Linux.

Steps

Observe the Initialization of Exception

1. `git clone --depth=1 https://github.com/raspberrypi/linux -b rpi-4.9.y`
2. Add `printk("Initialize traps\n");` and `trap_init();` to the `start_kernel` function in `/init/main.c`.
3. Add `printk("arm: system call handler initialization -> see assembly code\n");` to the function `trap_init` in `/arch/arm/kernel/traps.c`.

Add a New System Call

1. Add `CALL(sys_newsyscall)` to `/arch/arm/kernel/calls.S`
2. Add `#define __NR_newsyscall (__NR_SYSCALL_BASE+INCREMENT)`
3. Add `newsyscall.c` to `/arch/arm/kernel/` with appropriate code.

```
#include<linux/linkage.h>
#include <linux/kernel.h>

asmlinkage void sys_newsyscall(int a, char *b){
        printk("system yee …\n");
        printk("int1:%d, staring:%s in kernel\n", a, b);
}
```

4. Add `asmlinkage void sys_newsyscall(int a, char* b);` to `/include/linux/syscalls.h )`
5. In `/arch/arm/kernel/Makefile`, append `newsyscall.o` to

```
obj-y := elf.o entry-common.o irq.o opcodes.o \
process.o ptrace.o reboot.o return_address.o \
setup.o signal.o sigreturn_codes.o \
stacktrace.o sys_arm.o time.o traps.o
```

6. `cd linux`
7. `export PATH=$PATH:$HOME/WORK/crossgcc2/bin`
8. `make ARCH=arm bcm2709_defconfig`
9. `make -j 7 ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- bzImage`
10. Replace the zImage on Raspbarry Pi with `/arch/arm/kernel/zImage`

How System Call Works

1. Create a `hello.c` with

```
#include <stdio.h>

int main (void) {
```

```
                printf("hello world~\n");
                return 0;
        }
```

2. `$HOME/WORK/crossgcc2/bin/arm-linux-gnueabihf-gcc -static hello.c -o hello`
3. `arm-linux-gnueabihf-objdump -d hello > assembly`

What does `asmlinkage` do?

`asmlinkage` tells GCC to get the arguments for the function from the CPU's stack instead of the registers.