Lab 14, interrupts, part 3

By ADMIN | *Published:* APRIL 16, 2013

Now we can study the C part of handling interrupts.
But before we start remember that we need to install stack pointers to appropriate modes:

```
01 void
02 trapinit(void)
03 {
04     Vpage0 *vpage0;
05     /* set up the exception vectors */
06     vpage0 = (Vpage0*)HVECTORS;
07     memmove(vpage0->vectors, vectors, sizeof(vpage0->vectors));
08     memmove(vpage0->vtable,  vtable,  sizeof(vpage0->vtable));
09
10     setr13(PsrMfiq, m->fiqstack+nelem(m->fiqstack));
11     setr13(PsrMirq, m->irqstack+nelem(m->irqstack));
12     setr13(PsrMabt, m->abtstack+nelem(m->abtstack));
13     setr13(PsrMund, m->undstack+nelem(m->undstack));
14 }
15
16 TEXT setr13(SB), $-4
17     MOVW    4(FP), R1
18     MOVW    CPSR, R2
19     BIC     $PsrMask, R2, R3
20     ORR     R0, R3
21     MOVW    R3, CPSR        /* switch to new mode */
22     MOVW    SP, R0          /* return old sp */
23     MOVW    R1, SP          /* install new one */
24     MOVW    R2, CPSR        /* switch back to old mode */
25     RET
```

Have a look at *trap(Ureg *)*.
First worth to check that kernel stack is not overflow, otherwise we have no choice than “deat screen”:

```
01 int rem, itype;
02
03 if(up != nil)
04     rem = ((char*)ureg)-up->kstack;
05 else rem = ((char*)ureg)-(char*)m->stack;
06
07 if(ureg->type != PsrMfiq && rem < 256) {
08     panic("trap %d stack bytes remaining (%s), "\
09           "up=#%8.8lux ureg=#%8.8lux pc=#%8.8ux"
10           ,rem, up?up->text:"", up, ureg, ureg->pc);
11     for(;;);
12 }
```

(Later we can use *delay()* and *reboot()* in such case to prevent frozing)

Adjust *ureg->pc*

```
01 itype = ureg->type;
02 /*  All interrupts/exceptions should be resumed at ureg->pc-4,
03     except for Data Abort which resumes at ureg->pc-8. */
04 if(itype == PsrMabt+1)
05     ureg->pc -= 8;
06 else ureg->pc -= 4;
07
08 if(up){
```

Categories

- Blog
- Boost
- C++
- Cryptography
- Embedding
- Hybrids
- Inferno OS
- MacAppStore
- Misc
- Models
- Projects
- PyQt
- PySide
- Qt
- QtSpeech
- Raspberry Pi
- Research
- Ru
- TogMeg
- Trac
- TTS
- Tutorial
- Undo
- Web

```
    if(up){
        up->pc = ureg->pc;
        up->dbgreg = ureg;
    }
```

Now we can have a *switch* of type. Break from switch will mean kernel panic:

```
        default:
    faultpanic:
            setpanic();
            dumpregs(ureg);
            panic("exception %uX %s\n", ureg->type, trapname(ureg->type));
            break;
        }
```

All cases:

```
        switch(itype) {
        case PsrMirq:
            t = m->ticks;    /* CPU time per proc */
            up = nil;       /* no process at interrupt level */

            //todo:irq(ureg);

            up = m->proc;
            preemption(m->ticks - t);
            break;

        case PsrMund:
            panic("Undefined instruction");
            if(*(ulong*)ureg->pc == BREAK && breakhandler) {
                int s;
                Proc *p;

                p = up;
                s = breakhandler(ureg, p);

                if(s == BrkSched) {
                    p->preempted = 0;
                    sched();
                } else if(s == BrkNoSched) {
                    /* stop it being preempted until next instruction */
                    p->preempted = 1;
                    if(up)
                        up->dbgreg = 0;
                    return;
                }
                break;
            }
            if(up == nil) goto faultpanic;
            spllo();
            if(waserror()) {
                if(waslo(ureg->psr) && up->type == Interp)
                    disfault(ureg, up->env->errstr);
                setpanic();
                dumpregs(ureg);
                panic("%s", up->env->errstr);
            }
    //      if(!fpiarm(ureg)) {
    //          dumpregs(ureg);
    //          sys_trap_error(ureg->type);
    //      }
            poperror();
            break;

        case PsrMsvc: /* Jump through 0 or SWI */
            if(waslo(ureg->psr) && up && up->type == Interp) {
                spllo();
                dumpregs(ureg);
                sys_trap_error(ureg->type);
            }
            setpanic();
            dumpregs(ureg);
            panic("SVC/SWI exception");
            break;

        case PsrMabt: /* Prefetch abort */
            if(catchdbg && catchdbg(ureg, 0))
                break;
            /* FALL THROUGH */
        case PsrMabt+1: /* Data abort */
            if(waslo(ureg->psr) && up && up->type == Interp) {
                spllo();
```

```
66              faultarm(ureg);
67          }
68          print("Data Abort\n");
69          /* FALL THROUGH */
```

Check booting that nothing is broken:

```
1  ## Starting application at 0x00008000 ...
2  Entered main() at 00009074 with SP=00002FE8
3  Clearing Mach:  00002000-00002060
4  Clearing edata: 00065708-0006C830
5  Conf: top=134217728, npage0=32659, ialloc=26750976, nproc=735
6
7  ARM 0 MHz id 410fb767
8  Inferno OS Fourth Edition (20121207) Vita Nuova
9  to inifinite loop
```

FILES:

fns.h
dat.h
intr.s
trap.c
dump.c
main.c
armv6.s

This entry was posted in *Blog*, *Inferno OS*, *Raspberry Pi*, *Research*. Bookmark the *permalink*. *Post a comment* or leave a trackback: *Trackback URL*.

« *Lab 13, interrupts, part 2*

Lab 15, Eve, Hello World from Limbo! »