From $CS107E/src/interrupts_asm.s

```
_vectors:
    ldr pc, abort_addr
    ldr pc, abort_addr
    ldr pc, abort_addr
    ldr pc, abort_addr
    ldr pc, abort_addr
    ldr pc, abort_addr
    ldr pc, interrupt_addr
    ldr pc, abort_addr

    abort_addr:       .word abort_asm
    interrupt_addr:   .word interrupt_asm

_vectors_end:
```

Symbols **_vectors** and **_vectors_end** mark region to be copied

From $CS107E/src/interrupts.c

```
void interrupts_init(void)
{
    unsigned int *dst = _RPI_INTERRUPT_VECTOR_BASE;
    unsigned int *src = &_vectors;
    unsigned int n = &_vectors_end - &_vectors;

    for (int i = 0; i < n; i++) {
        dst[i] = src[i];
    }

}
```

From $CS107E/src/interrupts_asm.s

```
interrupt_asm:
1     mov   sp, #0x8000          @ init IRQ stack
2     sub   lr, lr, #4           @ compute resume addr (in lr)
3     push  {r0-r12, lr}         @ save all registers
4     mov   r0, lr               @ pass resume addr as arg
5     bl    interrupt_dispatch   @ call C function
6     ldm   sp!, {r0-r12, pc}^   @ restore saved registers
                                   pc = resume addr (from lr)
                                   ^ change mode & restore cpsr
```

Line 1: assign banked `sp` (shared `fp` left as-is). backtrace from perspective of interrupt handler will cross from irq to svc stack (unintentional, but sort of neat)

Line 2: banked `lr` holds `pc` at time of interrupt (+8 because of pipeline)

Line 3: save all non-banked registers for simplicity and safety (could save just callee-owned registers since C function should obey conventions, but...)

Line 5: call C function `interrupt_dispatch (unsigned int arg)`

Line 6: consider why all of these actions must happen as one unit