start.s

```
_start:
    mov r0, #0xD2     @ mode = interrupt
    msr cpsr_c, r0
    mov sp, #0x8000

    mov r0, #0xD3     @ mode = supervisor
    msr cpsr_c, r0
    mov sp, #0x8000000
    mov fp, #0

    bl _cstart
```

cstart.c

```
void _cstart() {
    int *bss = &__bss_start__;
    int *bss_end = &__bss_end__;

    while (bss < bss_end) { // Zero BSS section
        *bss++ = 0;
    }

    // Copy interrupt vector table to proper location
    int *vectorsdst = _RPI_INTERRUPT_VECTOR_BASE;
    int *vectors = &_vectors;
    int *vectors_end = &_vectors_end;
    while (vectors < vectors_end) {
        *vectorsdst++ = *vectors++;
    }

    main();
}
```

vectors.s

```
_vectors:
    ldr pc, _abort_asm
    ldr pc, _abort_asm
    ldr pc, _abort_asm
    ldr pc, _abort_asm
    ldr pc, _abort_asm
    ldr pc, _abort_asm
    ldr pc, _interrupt_asm
    ldr pc, _abort_asm

    _abort_asm:         .word abort_asm
    _interrupt_asm:     .word interrupt_asm
_vectors_end:
```

```
interrupt_asm:
    sub   lr, lr, #4                @ compute return add
    push  {r0-r3, r12, lr}          @ save registers
    mov   r0, lr                    @ pass old pc as arg
    bl    interrupt_vector          @ call C function
    ldm   sp!, {r0-r3, r12, pc}^    @ return, ^ to change
                                      mode + restore cpsr

abort_asm:
    b abort_asm
```