

Typical Questions for Experiment 5

Checkpoint 1

- What does the `swi` instruction do?
- Does the `swi` instruction jump directly to the label `swi_handler`, in the one step?
- What is the address of the label `ev08`?
- What do the `mrs` and `msr` instructions do? Why are they used? Why is using a read-modify-write cycle better than just using `msr` by itself?
- What is the value of CPSR at the “`mov r1, #value2`” instruction in *flash-v1.s*?
- At what points in the program does CPSR change?
- What does the `.section` directive do? What is the address of the label `swi_handler`?

Checkpoint 2

- What would happen if the “`movs pc, lr`” instruction was replaced with “`mov pc, lr`”? Would the program still work (ie, would the LEDs still flash)? If so, why is it a bad idea to use `mov`?
- What would the software interrupt return if a value of 5 was passed in R0? What is the path taken by the processor?
- Why is it a good idea to use separate public/private header files?

Checkpoint 3

- What does the “`ldr pc, [ip, r0, lsl #2]`” instruction do?
- Why is it a good idea to use jump tables to handle `swi` functions?

Checkpoint 4

- What do the `strneb` and `tst` instructions do? Why is `tst` used instead of `cmp`?
- What happens if either S2 or S3 is pressed? If neither are pressed? If both are pressed?
- Is there any moment when all six LEDs are turned on at the same time? Why or why not?

Checkpoint 5

- What do the first three lines at the label `init` do? Why are they necessary? What value is stored using the `strb` instruction, and at what address?
- What is the value of CPSR at “`b main`”? During the execution of the IRQ handler?
- What happens if the `bic` and `strb` instructions are removed from `irq_handler`?
- What happens to the trivial software counter in R0 in `main`? Why?

Checkpoint 6

- What are the values of CPSR during the execution of `init`?
- What is the value of SP on entry to `irq_handler`? What values are stored in memory (and in what order) by the `stmfd` instruction?
- What happens to the trivial software counter in R0 in `main`? Why?

Checkpoint 7

- Your working source code should speak for itself: it should be well-organised and well-commented.

Checkpoint 8 (No Credit)

- How do you include the instruction “`div10 Rd, Rm, #10`”, binary encoding “`0xFEE(Rm)(Rd)00A`” in your assembly code, without getting an assembler syntax error?

Checkpoint 9 (Extra Credit × 5)

- What is state of a process?
- How do you save the state of the suspended process on its own stack?
- How much time did you spend doing this task?