# Microprocessors with Security
# Final Exam, Fall 2014

**Name: _Solutions_**

**Note: No Explanations, No Credits!**

1. Interrupts **(25 points)**

   a. Cortex-A9 (ARM) **physically** accepts 2 kinds of interrupt inputs. What are those? How are these 2 interrupts different and why? (5 points)

- nIRQ: Normal interrupt
- nFIQ: Fast interrupt. Context switch is fast since it have dedicated registers.

   b. Explain **what** is done by H/W (Cortex-A9) upon an entry to the interrupt handling? (8 points)

- Return address is saved to r14 in the interrupt mode.
- CPSR is saved to SPSR of the interrupt mode.
- CPSR' I is set to 1 to mask the further interrupt.
- CPSR mode is changed to interrupt mode.
- PC is changed to 0x18

   c. **What instruction** would you use to return from the interrupt and **why**? (5 points)

**subs pc, r14, #4** or **ltm sp, {pc}^**
where sp contains the memory address of the stack.

It restores both PC and CPSR from r14_irq and SPSR_irq, respectively.

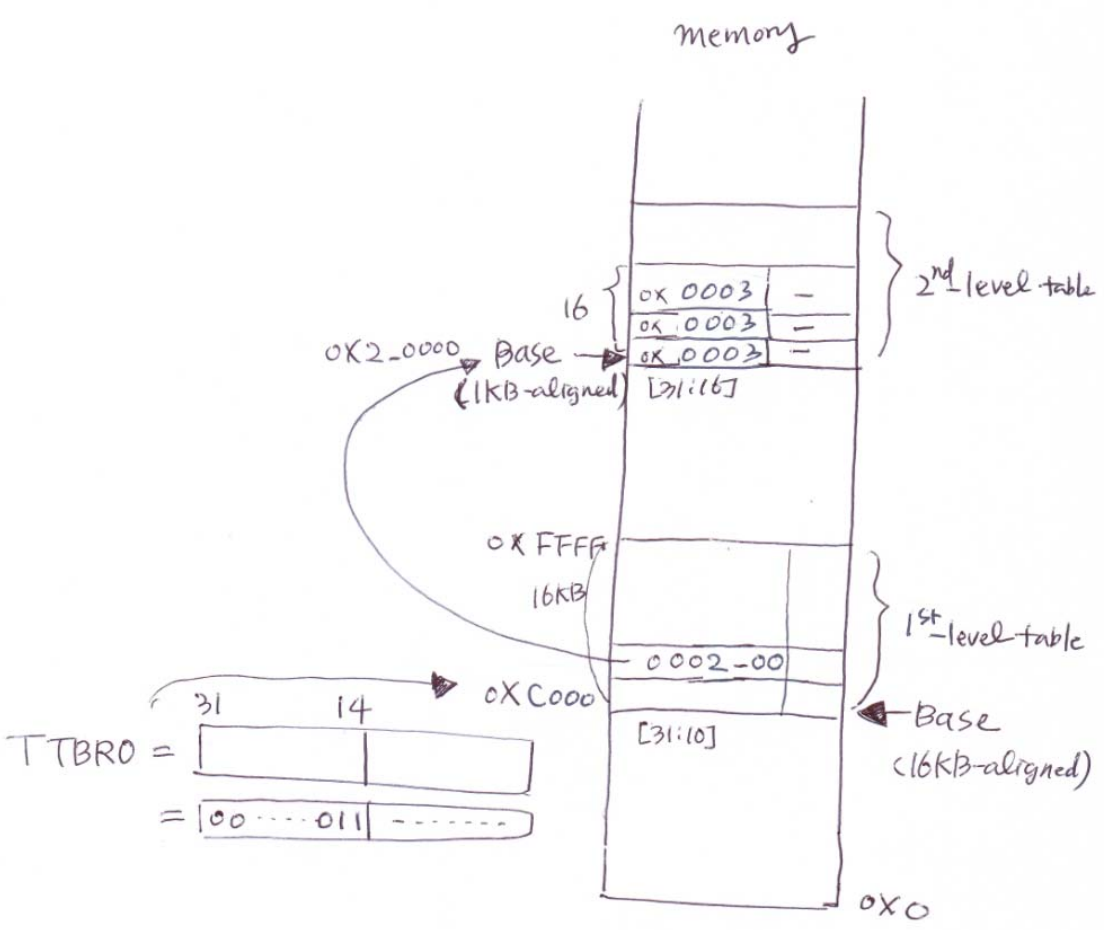
   d. What is the difference between the following 2 instructions? When would you use the instructions? (7 points)

| | Detailed operations | Usage case |
|---|---|---|
| `mov pc, lr;` | pc ← lr | Return from a function |
| `movs pc, lr;` | pc ← lr<br>cpsr ← spsr | Return from exception or interrupt |

2. The SVC (Supervisor Call, previously SWI) instruction is typically used to implement system calls in OS. The diverse system calls in OS are differentiated by the 24-bit immediate field in the SVC instruction. Write an ARM assembly code that extracts the immediate field after jumping to the ISR of SVC. **(10 points)**

```
ldr  r10, [lr, #-4];
BIC  r10, r10, #0xff000000
```

3. You want to map a 64KB virtual page from 0x0010_0000 to a physical page from 0x0030_0000. Draw page tables in memory below as detailed as possible. Focus only on memory addresses when you create page tables (ignore the other bit fields). Specify the base locations of page tables in the figure as well. Elaborate why you chose the base locations. What value would you program to the TTBR register? (Refer to the page table entry information in the next page) **(25 points)**



### Short-descriptor translation table first-level descriptor formats

Each entry in the first-level table describes the mapping of the associated 1MB MVA range.

Figure B3-4 shows the possible first-level descriptor formats.

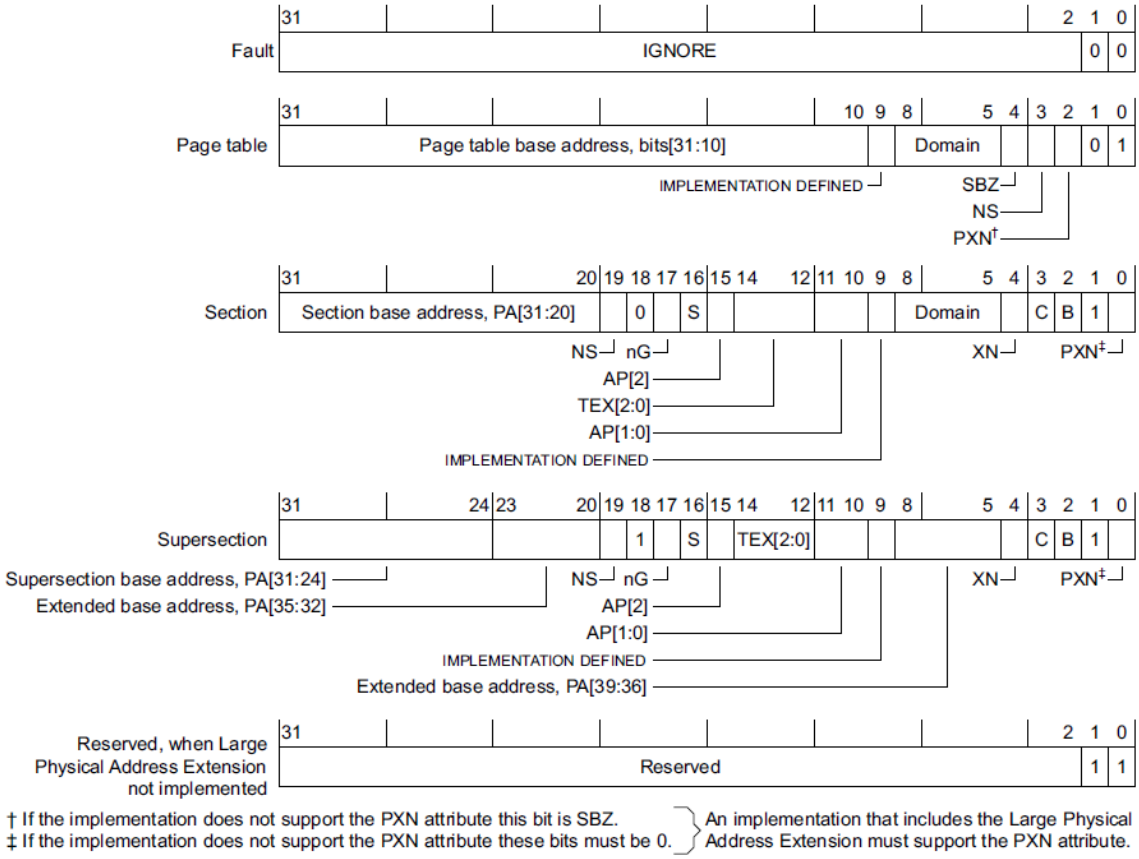



**Figure B3-4 Short-descriptor first-level descriptor formats**

### Short-descriptor translation table second-level descriptor formats

Figure B3-5 shows the possible formats of a second-level descriptor.

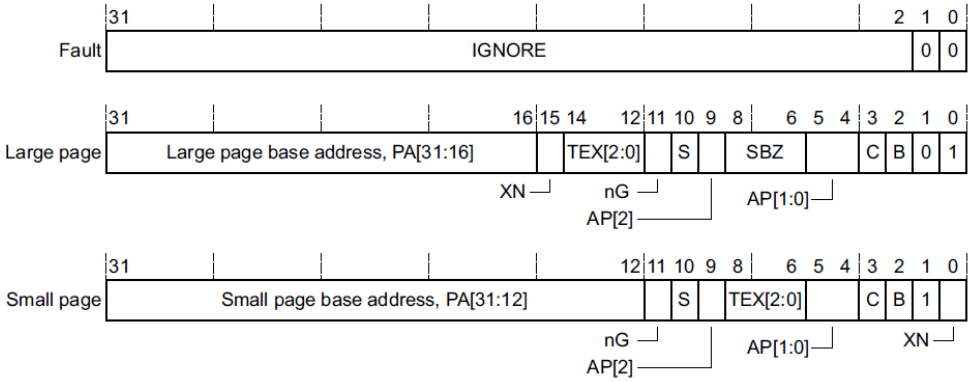
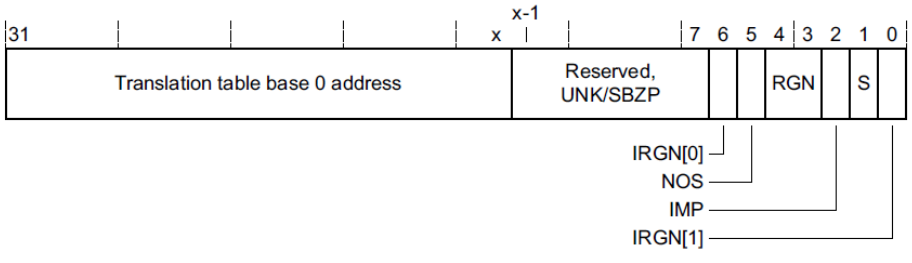



**Figure B3-5 Short-descriptor second-level descriptor formats**

In an implementation that includes the Multiprocessing Extensions, the 32-bit TTBR0 bit assignments are:



4. TrustZone **(20 points)**

   a. There are 2 ways to enter into the Monitor mode in TrustZone. Explain as detailed as possible. **(5 points)**

- SMC (Secure Monitor Call)
- FIQ, IRQ or Abort exceptions: SCR register (EA, FIQ, IRQ) should be set appropriately.

   b. Write an assembly code that sets the base addresses of 3 interrupt vector tables to VBARs and MVBAR for secure world, normal world, and monitor mode. Note that the program should include the world-switching code. You are allowed to use pseudo code, in case you don't remember the exact syntax of MCR and MRC instructions **(15 points).**

| Secure World Code | Monitor Mode Code | Normal World Code |
|---|---|---|
| `cyfd_secure_vector:`<br>`b cyfd_secure_reset`<br>`b cyfd_secure_undefined`<br>`b cyfd_secure_software_interrupt`<br>`b cyfd_secure_prefetch`<br>`b cyfd_secure_data`<br>`b cyfd_secure_not_used`<br>`b cyfd_secure_irq`<br>`b cyfd_secure_fiq`<br><br>`cyfd_secure_reset:`<br><br>`// Set VBAR (Vector Base Address Register) to my own interrupt vectors`<br>`LDR r0,=cyfd_secure_vector`<br>`MCR p15, 0, r0, c12, c0, 0`<br><br>`// Set MVBAR (Monitor Vector Base Address Register) to my own interrupt vectors`<br>`LDR r0, =cyfd_monitor_vector`<br>`MCR p15, 0, r0, c12, c0, 1`<br><br>`.arch_extension sec`<br>`// smc will take you to SMC_handler in monitor mode`<br>`smc #0` | `cyfd_monitor_vector:`<br>`nop`<br>`nop`<br>`b SMC_handler`<br>`nop`<br>`nop`<br>`nop`<br>`b cyfd_monitor_irq`<br>`b cyfd_monitor_fiq`<br><br>`SMC_handler:`<br><br>`// SPSR'mode <- Supervisor`<br>`ldr lr, =cyfd_normal_reset`<br>`msr spsr_cxsf, #MODE_SVP`<br><br>`//-----------------------`<br>`// Switch to Normal world`<br>`// -----------------------`<br>`// Read SCR`<br>`// Set NS bit`<br>`// Write SCR`<br><br>`MRC p15, 0, r4, c1, c1, 0`<br>`ORR r4, #NS_BIT`<br>`MCR p15, 0, r4, c1, c1, 0`<br>`movs pc, lr` | `cyfd_normal_vector:`<br>`b cyfd_normal_reset`<br>`b cyfd_normal_undefined`<br>`b cyfd_normal_software_interrupt`<br>`b cyfd_normal_prefetch`<br>`b cyfd_normal_data`<br>`b cyfd_normal_not_used`<br>`b cyfd_normal_irq`<br>`b cyfd_normal_fiq`<br><br>`cyfd_normal_reset:`<br><br>`// Set VBAR (Vector Base Address Register) to my own interrupt vectors`<br><br>`LDR r0, =cyfd_normal_vector`<br>`MCR p15, 0, r0, c12, c0, 0` |

5. In the worst case, what would happen in L1 caches and L1 TLBs of Cortex-A9 when executing the following 4 instructions? Assume that the cache line size is 8 words (=32 bytes), and the page size is 4KB **(20 points)**

| Address: Instructions | Worst case scenario | |
|---|---|---|
| | L1 Caches (I$ and D$) | TLB |
| 0x0FFC: sub r0, r1, r2 | I$ miss (8-word line fill) | ITLB miss |
| 0x1000: mov r10, #0xA004 | I$ miss (8-word line fill) | ITLB miss |
| 0x1004: ldr r2, [r10, #0] | D$ miss (dirty line replacement and 8-word line fill) | DTLB miss (dirty bit update to page table) |
| 0x1008: ldm sp!, {r0-r12} | 3 D$ misses, depending on sp. For each miss, dirty line replacement and 8-word line fill | 2 DTLB misses (dirty bit update to page table) |