COMP22712 – Exercise Terminal Window

You should submit this exercise for assessment and feedback through blackboard. The submission should include code using a stack to nest procedure calls to print text message on the screen.

A legible, appropriately commented source file will be appreciated and credited appropriately.

Submit this exercise as a single ASCII file

Add the following header information to your file:

```
Exercise Terminal Window
your name # first last
your student ID
your email # we may provide feedback by email if BB fails
```

If you use multiple files, start each file in your submitted file with:

```
##### file your_filename.s #####
```

COMP22712 – System Calls (SVC)

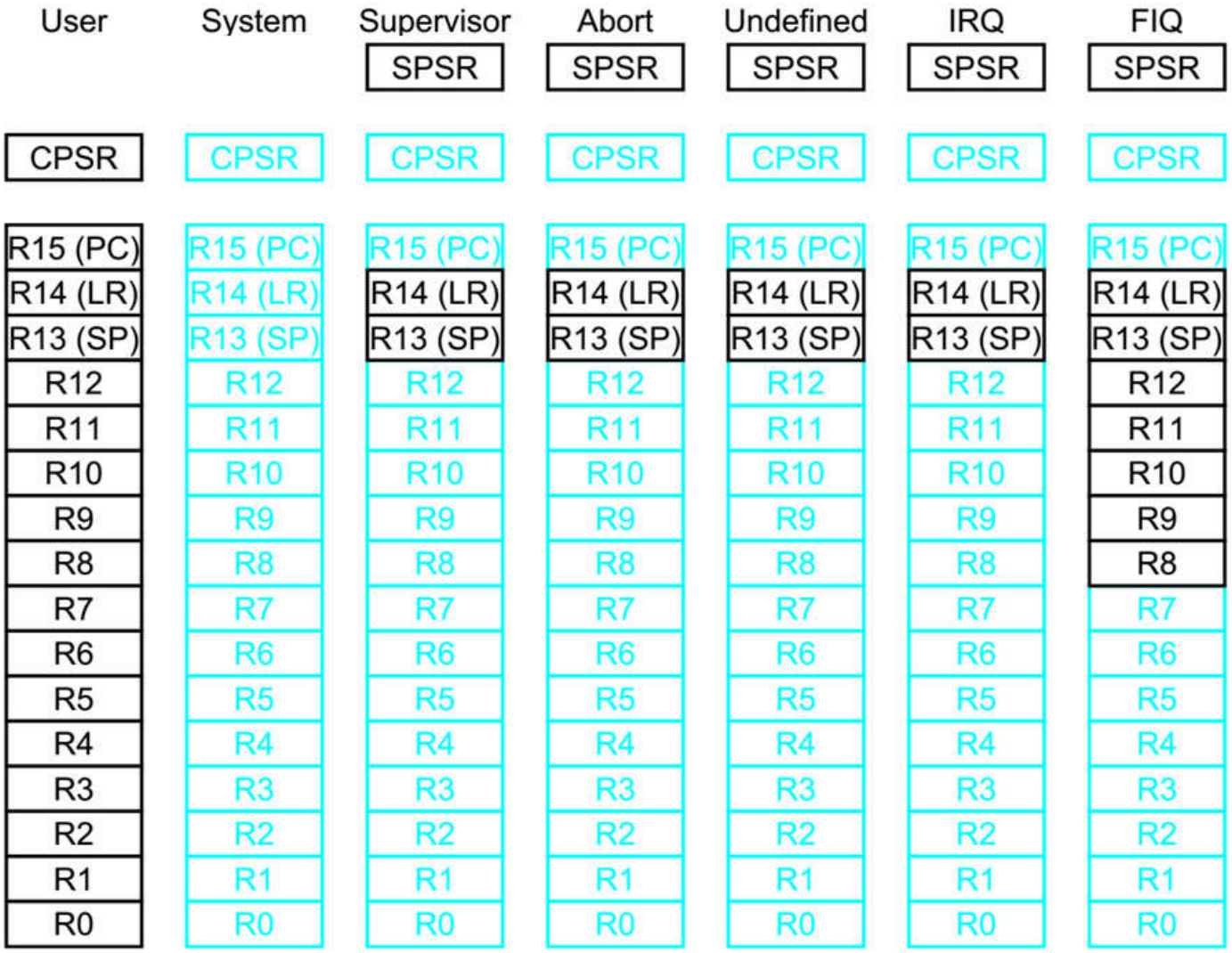
What belongs where?

- **Operating system (previleged mode)**
  - **Everything that deals with hardware (I/O, memory, files, network, ...)**
  - **Interprocess communication, semaphores, ...**
- **Libraries (user mode)**
  - **Functionality that is used by many programs (GUI elements, compression, encryption, ...)**
  - **May call operating system functions**
- **User program (user mode)**
  - **Application-specific code**
  - **May use operating system and library functions**

Understanding ARM Context Switching

What do we have to do for carrying out a context switch in ARM when an SVC is fired?

- **Save working registers used by SVC handler to memory (typically on stack)**
- **Save registers in „task control block“**
- **R0...R12 are easy, but what about the others?**



COMP22712 – System Calls (SVC)

The SVC instruction (SVC xxxx) has a 24~bit field which is ignored by the processor.

This can be used to indicate the type of SVC.

To determine the contents of this field it is necessary to read the op. code from the service routine.

E F x x x x x x

To read the instruction use:

```
LDR     R14, [LR, #-4]   ; Read SVC instruction
BIC     R14, R14, #&FF000000 ; Mask off opcode
```

There are a few SVCs defined already in the simulation environment and we use SVCs above 0x100 for our lab!

COMP22712 – System Calls (SVC)

The ARM vector table

Exception	Mode	Vector (address)	Link made (LR)
Reset	Supervisor	00000000	None
Undefined instruction	Undefined	00000004	R14_und = undef. instr. + 4
SVC	Supervisor	00000008	R14_svc = SVC instr. + 4
Prefetch abort	Abort	0000000C	R14_abt = aborted instr. + 4
Data abort	Abort	00000010	R14_abt = aborted instr. + 8
—	—	00000014	—
IRQ	IRQ	00000018	R14_irq = interrupted instr. + 4
FIQ	FIQ	0000001C	R14_fiq = interrupted instr. + 4

Abbreviation	Mode	CPSR code (Binary)	CPSR code (Hex)
USR	User	1 0000	10
FIQ	Fast Interrupt	1 0001	11
IRQ	Interrupt	1 0010	12
SVC	Supervisor	1 0011	13
ABT	Abort	1 0111	17
UND	Undefined	1 1011	1B
SYS	System	1 1111	1F

```
MRS     R0, CPSR        ; Get current CPSR
BIC     R0, R0, #&0F    ; Clear low order bits
MSR     CPSR_c, R0      ; Rewrite CPSR
NOP                     ; Bug fix on some ARMs
```

COMP22712 – System Calls (SVC)

In an ARM, calling an SVC like

```
MOV     R1, R2          ; some random instruction
SVC     &42             ; call the SVC with argument
ADD     R1, R2, R2      ; some random instruction after SVC
```

has the following behavior:

- **The current status CPSR is saved to the supervisor SPSR**
- **The mode is switched to supervisor mode**
- **Normal (IRQ) interrupts are disabled**
- **ARM mode is entered (if not already in use)**
  (you can call an SVC from ARM (32-bit) and Thumb (16-bit) mode)
- **The address of the following instruction (i.e. the return address) is saved into the link register (LR or R14); note this is R14_SVC**
  → Similar to nested BLs, nested SVCs require stacking SPSR_SVC
- **The PC is altered to jump to address 00000008**

COMP22712 – System Calls (SVC)

To return from an SVC requires two steps:

- **Set PC to return address**
- **Restore supervisor status register to current status register (copy SPSR to CPSR)**

This has to be done atomic!

- **What happens if you first change status register?**
- **What hapens if you first change PC?**

We have support for this:

```
MOVS    PC, LR
```

- **The S is normally used to set flags (like in ADDS)**
- **Here it is used to copy SPSR to CPSR while changing the PC**
- **Note: this works only if the target register is the PC (you can see this as an polimorphic instruction)**

COMP22712 – SVC entry

We learned how we get the 24-bit parameter passed with SVC xxxxxx

We can use that to branch to a specific SVC handler:

```
SVC_entry       CMP     R0, #Max_SVC    ; Check upper limit
                BHI     SVC_unknown     ;
                CMP     R0, #0          ;
                BEQ     SVC_0           ;
                CMP     R0, #1          ;
                BEQ     SVC_1           ;
                ...
```

A more elegant way is this one (we will learn another version soon):

```
SVC_entry       CMP     R0, #Max_SVC    ; Check upper limit
                BHI     SVC_unknown     ;
XYZ             ADD     R0, PC, R0, LSL #2  ; Calc. table address
                LDR     PC, [R0, #0]    ; #0? - see below

Jump_table      DEFW    SVC_0           ;
                DEFW    SVC_1           ;
                ...
```

COMP22712 – System Calls (SVC)

MRS & MSR are special move instructions for the status register

We can use this to change mode (e.g. To go to IRQ mode)

```
MRS     R0, CPSR          ; Read current status
BIC     R0, R0, #&1F      ; Clear mode field
ORR     R0, R0, #&12      ; Append IRQ mode
MSR     CPSR_c, R0        ; Update CPSR
```

the _c is not needed here, but it changes only the command byte (8 LSBs) has

Starting a user program:

```
MOV     R14, #&D0         ; User mode, no ints.
MSR     SPSR, R14         ;
ADR     R14, User_code_start
MOVS    PC, R14           ; 'Return' to user code
```