Task Switch in TachyOS (ARMv7m)

> Context switch exploits exception exit / entry behavior on target architecture.

Start initial task

1. exception entry stack is prepared for new task

prepared exception entry

Registers	Initial Value	Stack Top Pointer (Task Context)
R4	0	*
R5 ~ R11	0	
xPSR	THUMB (1<<24)	
Return Address	address of runtime init routine for task	
R0 - R4, LR	DONT CARE	

2. enter handler mode (or supervisor mode) using `SVC`
3. set `PSP` with stack top address from the initial task
4. pop `PSP` by amount of saved context `R4-R11` (size of 32 bytes)

> Note : task context, which is not actually used, is inserted into the stack only for maximization of code reuse. By doing this, we don't have to differentiate the way of stack preparation between `start_task()` and `switch_task()`

Switching Task

1. When exception has occurred, the hardware pushes exception entry context`R0 - R3, R12, R14(LR), Return Address, xPSR` into process stack `PSP` ahead of jump to any exception handler

2. if switch is required before returning from exception handler, additional exception entry context is inserted into process stack (just beneath original exception entry frame) to mimick hardware to jump switch function on exception return.

    1. decrement `PSP` with size of exception entry stack
    2. fill function parameters for switch function (which is standard C function) into `R0 - R4` within the prepared exception entry stack
    3. set return address for the exception entry stack as a function entry of switch function (standard C function)
    4. set xPSR as thumb mode (assuming ARM interworking not used here)

3. Return from exception

4. handler will restore callee-saved registers

> Note : core registers `R4-R11` will be preserved by callee, that mean the exception handler implemented as pure C function which complies AAPCS should restore the values of callee-saved-registers ahead of return

5. then H/W will jump to the switch function with popping up the exception entry context from process stack `PSP`, which is pretty much same to calling general C function with argugment
6. switch function push callee saved registers into stack `PSP`

Switch function (Cortex-M4)

```
void tch_port_switch(uwaddr_t nth,uwaddr_t cth)
{
    asm volatile(
#if FEATURE_FLOAT > 0
            "vpush {s16-s31}\n"
#endif
            "push {r4-r11}\n"       ///< save callee-saved registers
            "str sp,[%0]\n"         ///< save stack pointer within TCB

            "ldr sp,[%1]\n"         ///< load stack pointer value from TCB
of next task
            "pop {r4-r11}\n"        ///< restore callee-saved registers
#if FEATURE_FLOAT > 0
            "vpop {s16-s31}\n"
#endif
            "ldr r0,=%2\n"          ///< set svc handler argument
            "svc #0" : : "r"(&((tch_thread_kheader*) cth)->ctx),"r"(&
((tch_thread_kheader*) nth)->ctx),"i"(SV_EXIT_FROM_SWITCH) :
"r4","r5","r6","r8","r9","r10", "r11", "lr" );
}
```

Exception entry stack frame on switch

Fields	Role	Origin
R0 ~ R3	function arguments	fake exception entry
R4 ~ R11	local variables	from original exception entry (callee saved registers)
R12	Intra-Procedure-call scratch	from fake exception entry
R14	Link Register	from fake exception entry
Return	return address	from fake exception entry
xPSR	instruction mode for return address (ARM / THUMB)	from fake exception entry