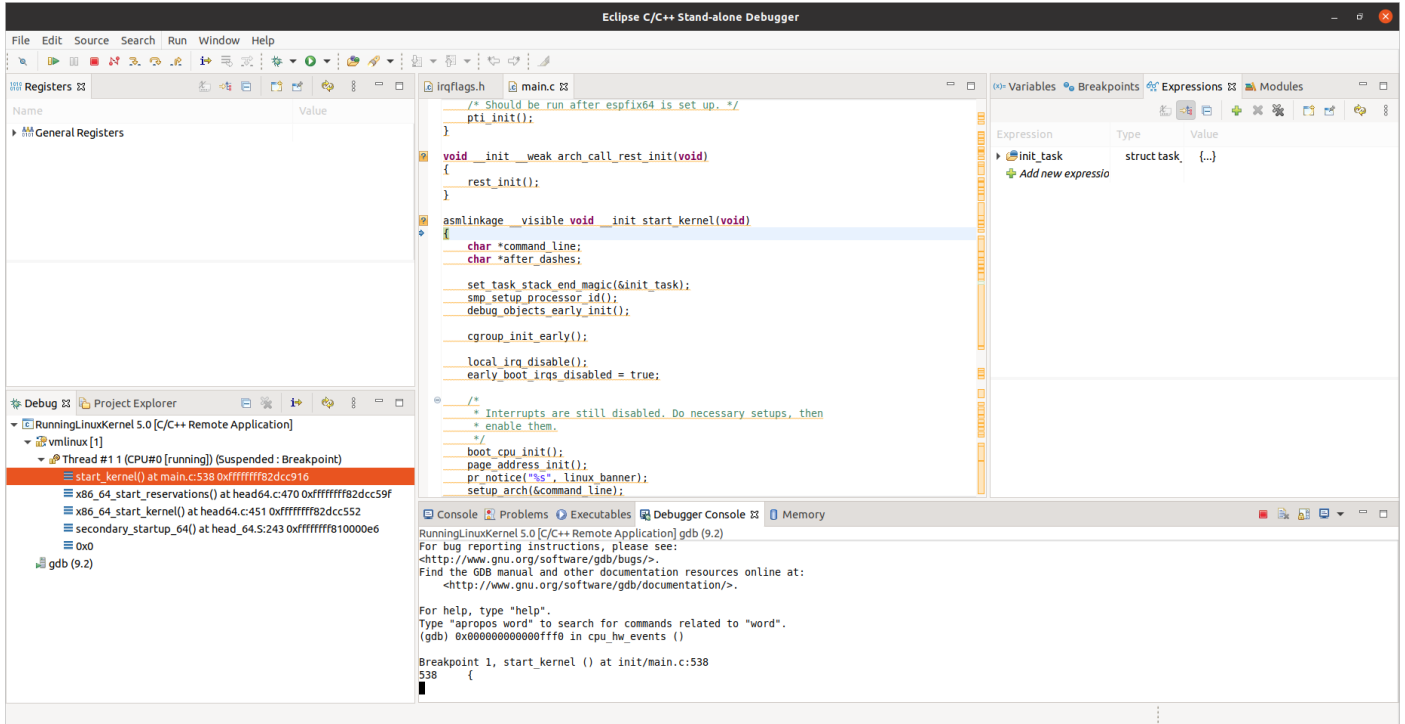


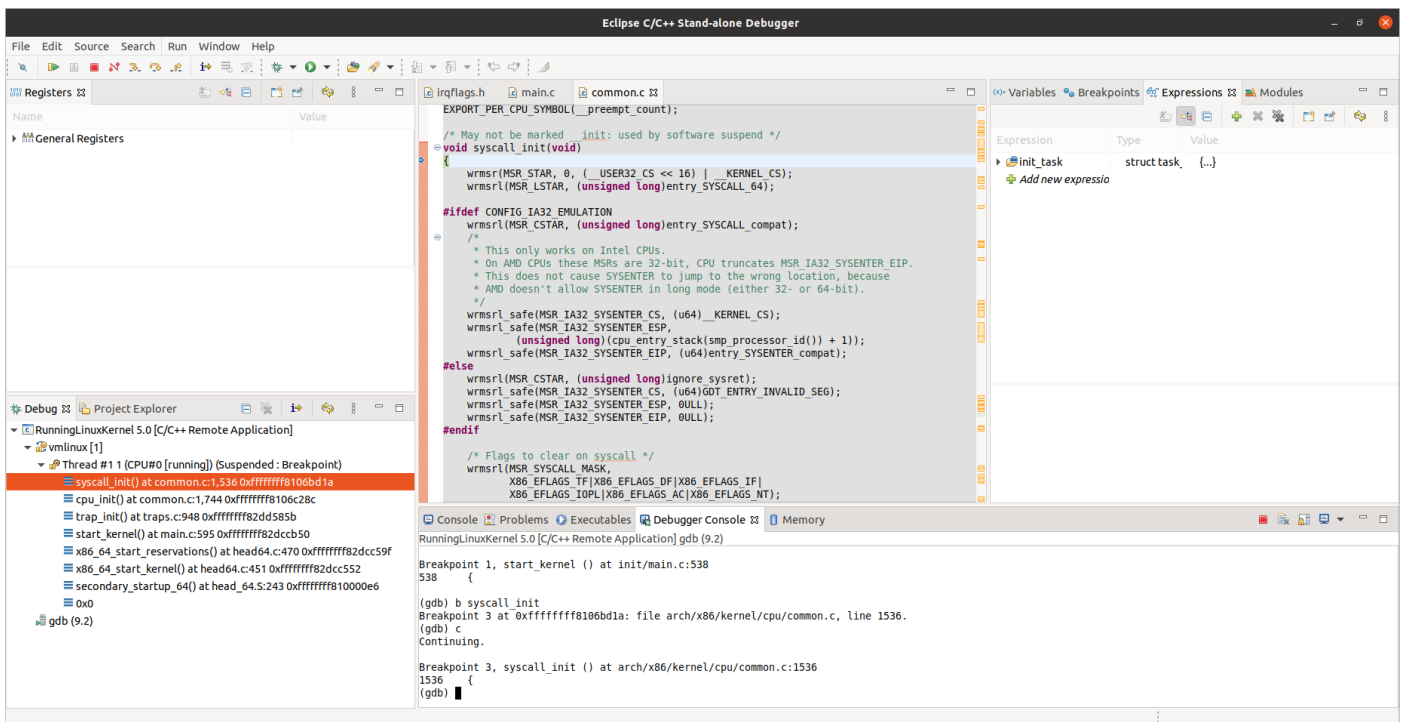
作業系統概論 hw1

學號: 408410113 姓名: 王 X 彥

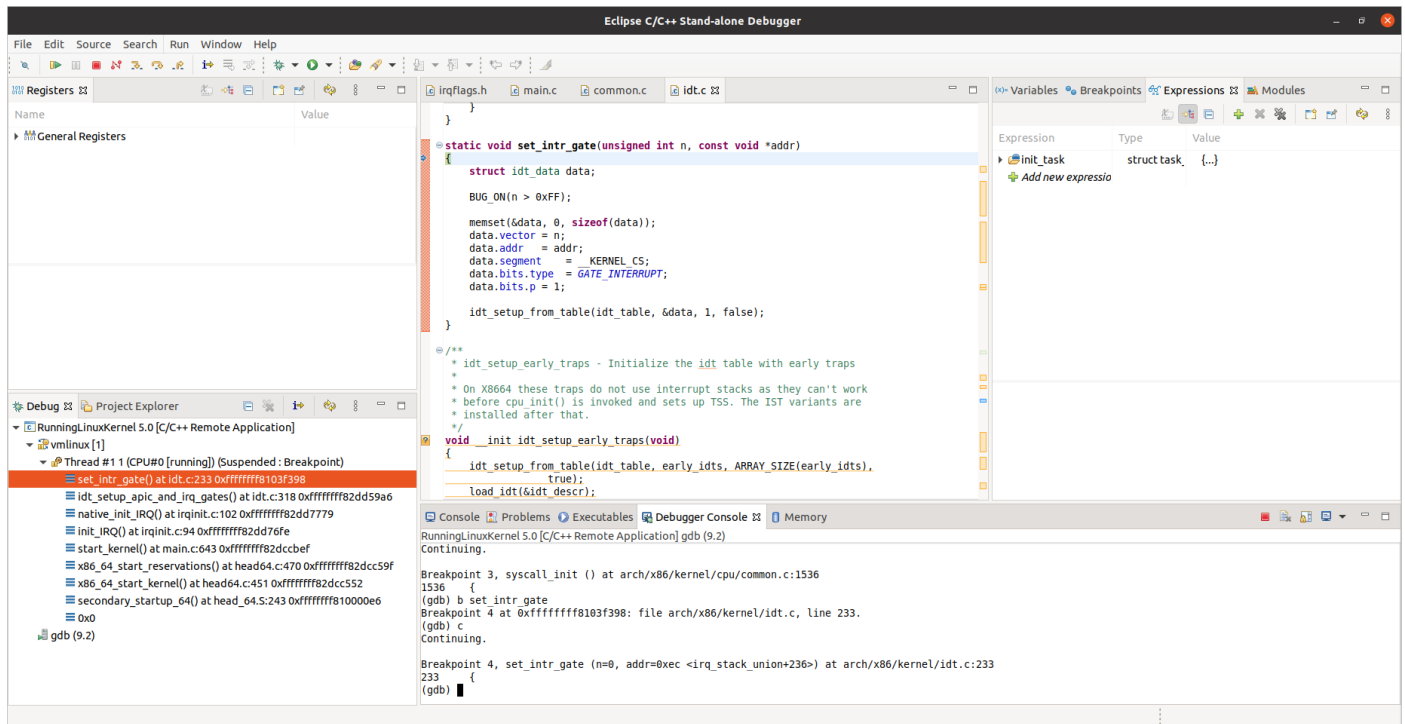
1. start_kernel



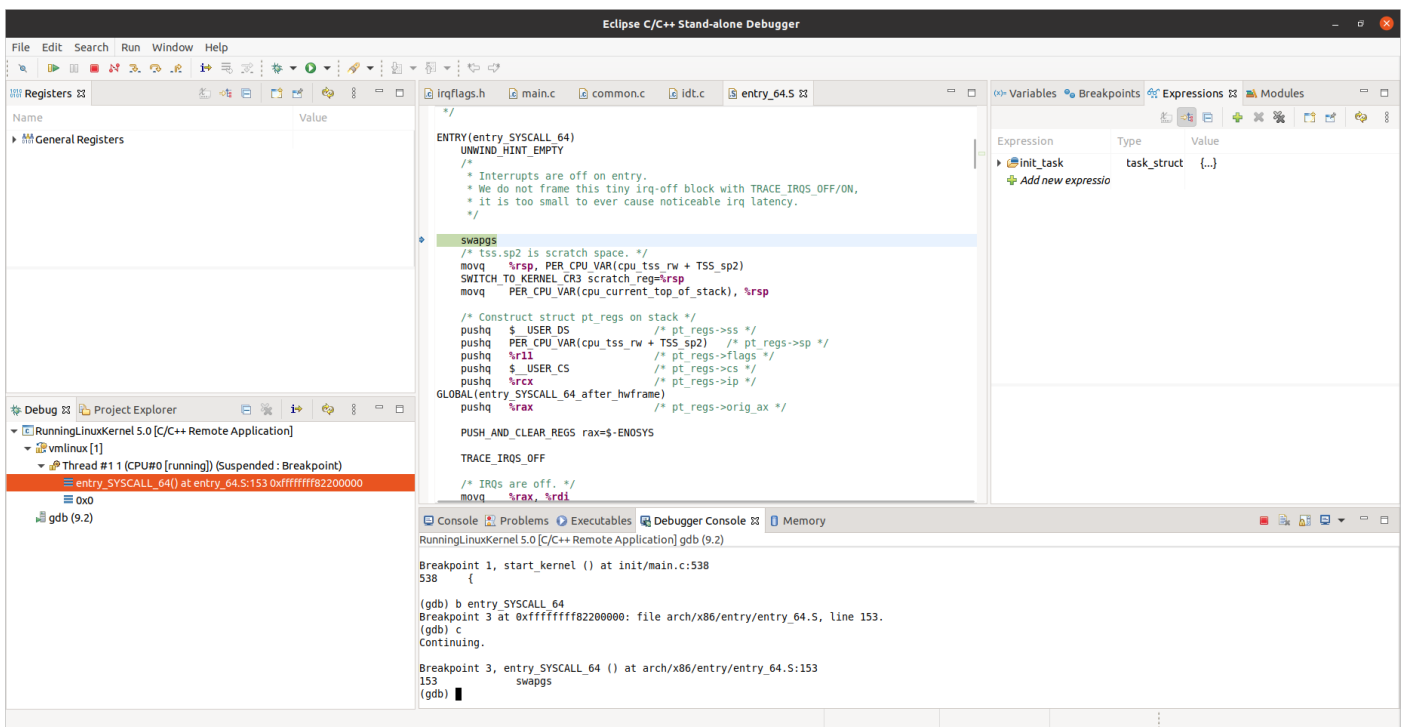
2. syscall_init



3. set_intr_gat



4. entry_SYSCALL_64



5. apic_timer_interrupt

The screenshot shows the Eclipse C/C++ Stand-alone Debugger interface. The main window displays the source code of the `apic_timer_interrupt` function in `entry_64.S`. The function is defined as follows:

```
#ifndef CONFIG SMP
apicinterrupt3 IRQ_MOVE_CLEANUP_VECTOR irq_move_cleanup_interrupt smp_irq_move_cleanup_interrupt
apicinterrupt3 REBOOT_VECTOR reboot_interrupt smp_reboot_interrupt
#endif

#ifdef CONFIG X86 UV
apicinterrupt3 UV_BAU_MESSAGE uv_bau_message_intr1 uv_bau_message_interrupt
#endif

apicinterrupt LOCAL_TIMER_VECTOR apic_timer_interrupt smp_apic_timer_interrupt
apicinterrupt X86_PLATFORM_IPI_VECTOR x86_platform_ipi smp_x86_platform_ipi

#ifdef CONFIG HAVE_KVM
apicinterrupt3 POSTED_INTR_WAKEUP_VECTOR kvm_posted_intr_wakeup_ipi smp_kvm_posted_intr_wakeup_ipi
apicinterrupt3 POSTED_INTR_NESTED_VECTOR kvm_posted_intr_nested_ipi smp_kvm_posted_intr_nested_ipi
#endif

#ifdef CONFIG X86 MCE_THRESHOLD
apicinterrupt THRESHOLD_APIC_VECTOR threshold_interrupt smp_threshold_interrupt
#endif

#ifdef CONFIG X86 MCE_AMD
apicinterrupt DEFERRED_ERROR_VECTOR deferred_error_interrupt smp_deferred_error_interrupt
#endif

#ifdef CONFIG X86 THERMAL_VECTOR
apicinterrupt THERMAL_APIC_VECTOR thermal_interrupt smp_thermal_interrupt
#endif
```

The left pane shows the registers, and the right pane shows the variable `apic_timer_interrupt` with its value `0xffffffffffffffff`. The bottom pane shows the console output, including the breakpoint hit at `entry_SYSCALL_64` and the execution of `apic_timer_interrupt`.

6. interrupt_entry

The screenshot shows the Eclipse C/C++ Stand-alone Debugger interface. The main window displays the source code of the `interrupt_entry` function in `entry_64.S`. The function is defined as follows:

```
.align 8
ENTRY(irq_entries_start)
vector=FIRST_EXTERNAL_VECTOR
.rept (FIRST_SYSTEM_VECTOR - FIRST_EXTERNAL_VECTOR)
UMWIND_HINT_IRET_REGS
pushq $(-vector+0x80) /* Note: always in signed byte range */
jmp common_interrupt
.align 8
vector=vector+1
.endr
END(irq_entries_start)

.macro DEBUG_ENTRY_ASSERT_IRQS_OFF
pushq %rax
#ifdef CONFIG_DEBUG_ENTRY
testl $X86_EFLAGS_IF, %eax
jz .Lokay_@
.Lokay_@:
popq %rax
#endif
.endm

/*
 * Enters the IRQ stack if we're not already using it. NMI-safe. Clobbers
 * flags and puts old RSP into old_rsp, and leaves all other GPRs alone.
 * Requires kernel GDBASE.
 * The invariant is that, if irq_count != -1, then the IRQ stack is in use.
 */
```

The left pane shows the registers, and the right pane shows the variable `interrupt_entry` with its value `0xffffffffffffffff`. The bottom pane shows the console output, including the breakpoint hit at `entry_SYSCALL_64` and the execution of `interrupt_entry`.

7. do_IRQ

The screenshot displays the Eclipse C/C++ Stand-alone Debugger interface. The main window shows the source code of the `do_IRQ` function in `kernel.c`. The function is responsible for handling normal device IRQs. The code includes comments about SMP cross-CPU interrupts and handlers. The function signature is `visible unsigned int __irq_entry do_IRQ(struct pt_regs *regs)`. The code defines a `struct pt_regs` and a `struct irq_desc`. It then enters a loop to handle the IRQ, including calling `ack_APIC_irq()` and `handle_irq(desc, regs)`. The code also includes a `pr_emerg_ratelimited` call for logging. The function ends with `__this_cpu_write(vector_irq[vector], VECTOR_UNUSED);`.

The left sidebar shows the Registers window with the following values:

Name	Value
rax	646
rbx	0
rcx	0
rdx	0
rsi	-2111829686
rdi	-2103427848
rbp	0x0<irq_stack_union>
rsp	0xfffff8800f603ff0
r8	0
r9	0
r10	0
r11	0
r12	0

The bottom console shows the following output:

```
RunningLinuxKernel 5.0 [C/C++ Remote Application] gdb (9.2)
Breakpoint 1, start_kernel () at init/main.c:538
538 {
(gdb) b do_IRQ
Breakpoint 3 at 0xfffff82201644: file arch/x86/kernel/irq.c, line 233.
(gdb) c
Continuing.
Breakpoint 3, do_IRQ (regs=0x0 <irq_stack_union>) at arch/x86/kernel/irq.c:233
233 {
(gdb)
```