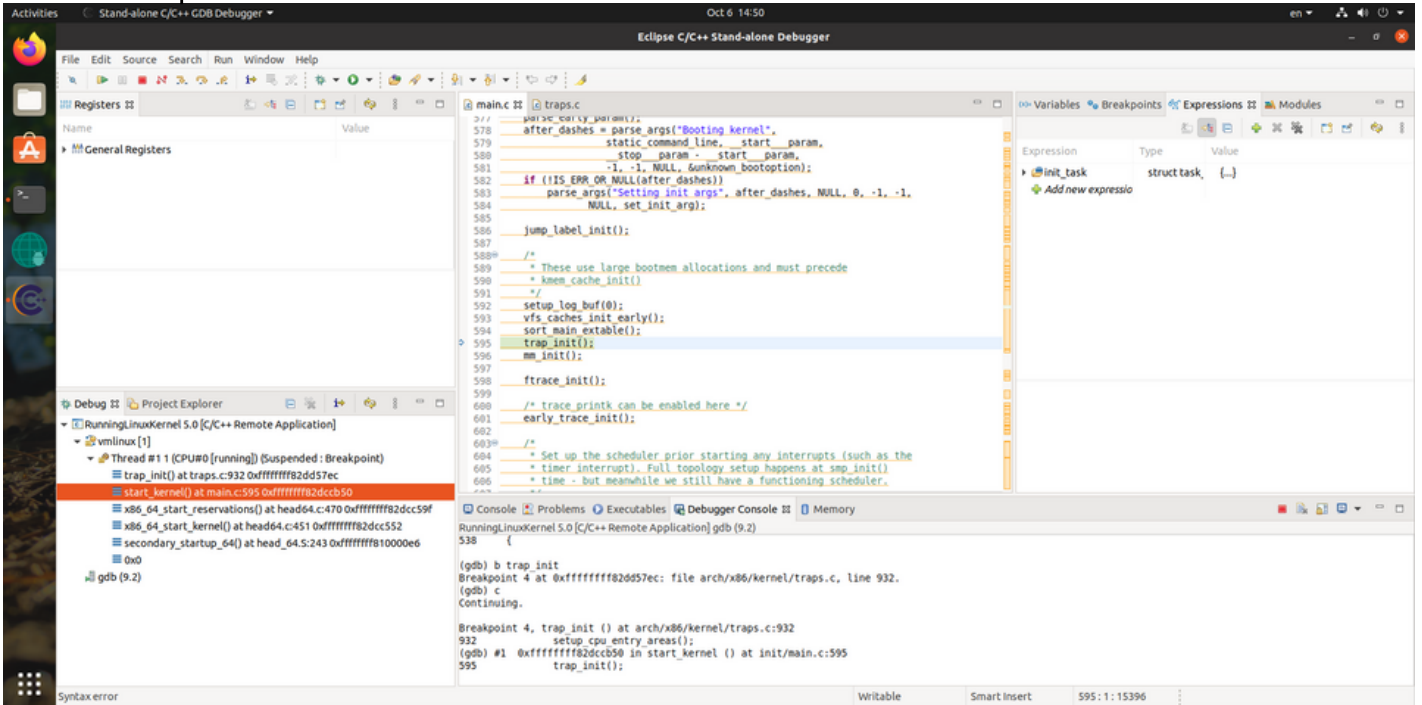


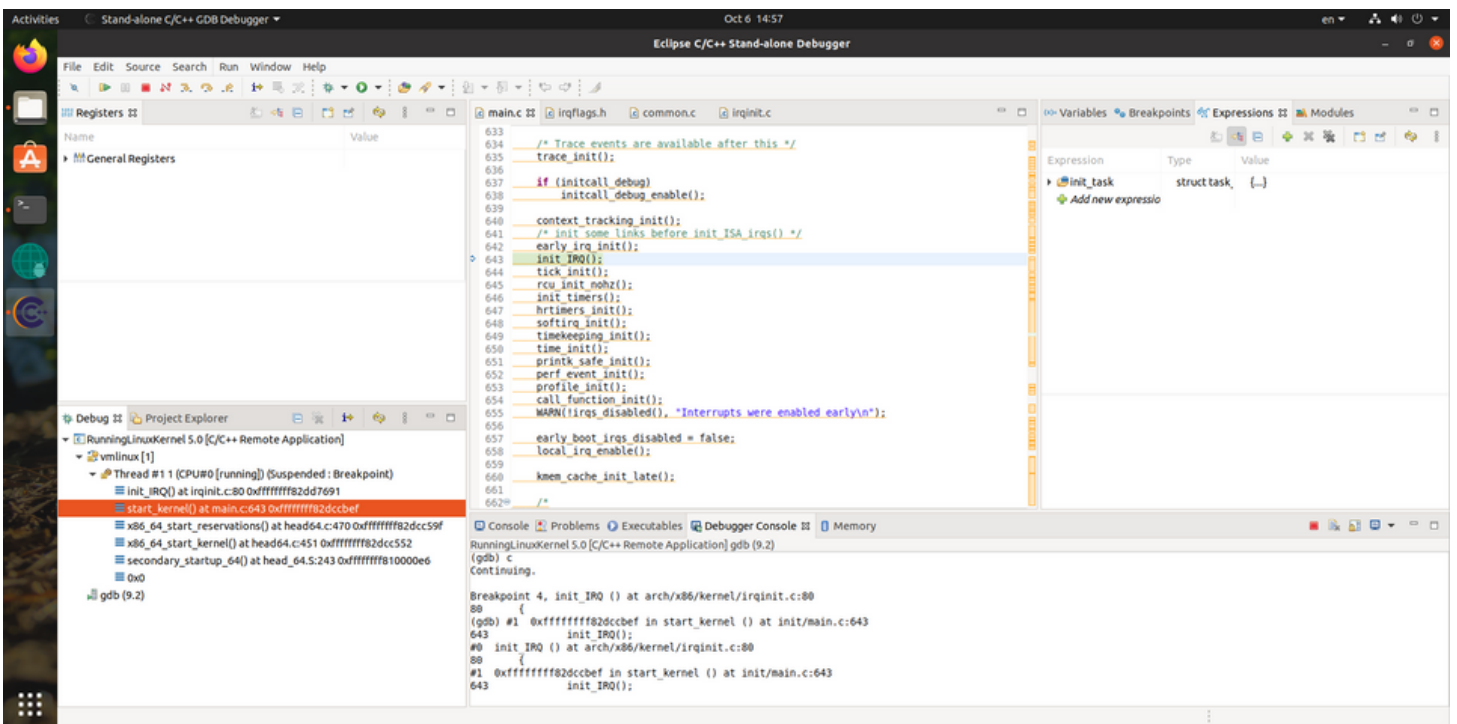
# HW2

408410094葉X勛

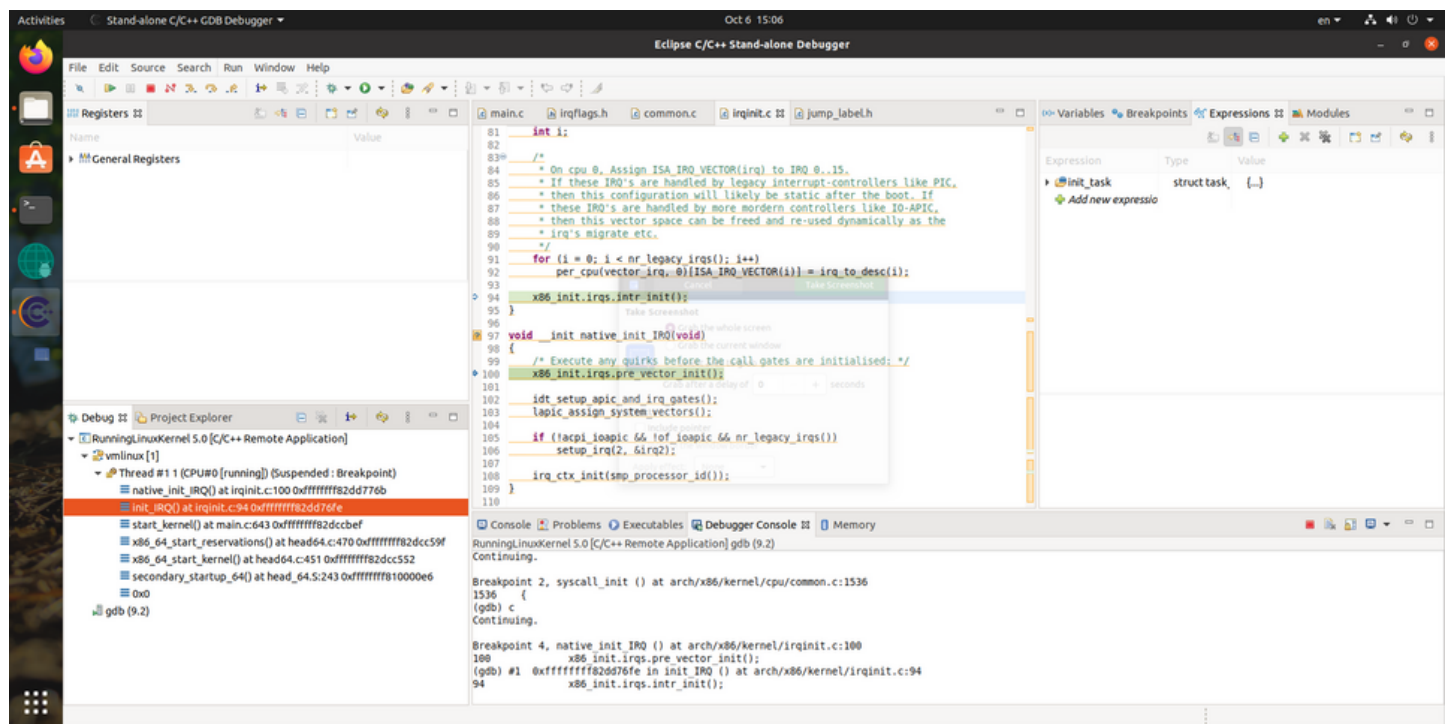
## 1.b trap\_init



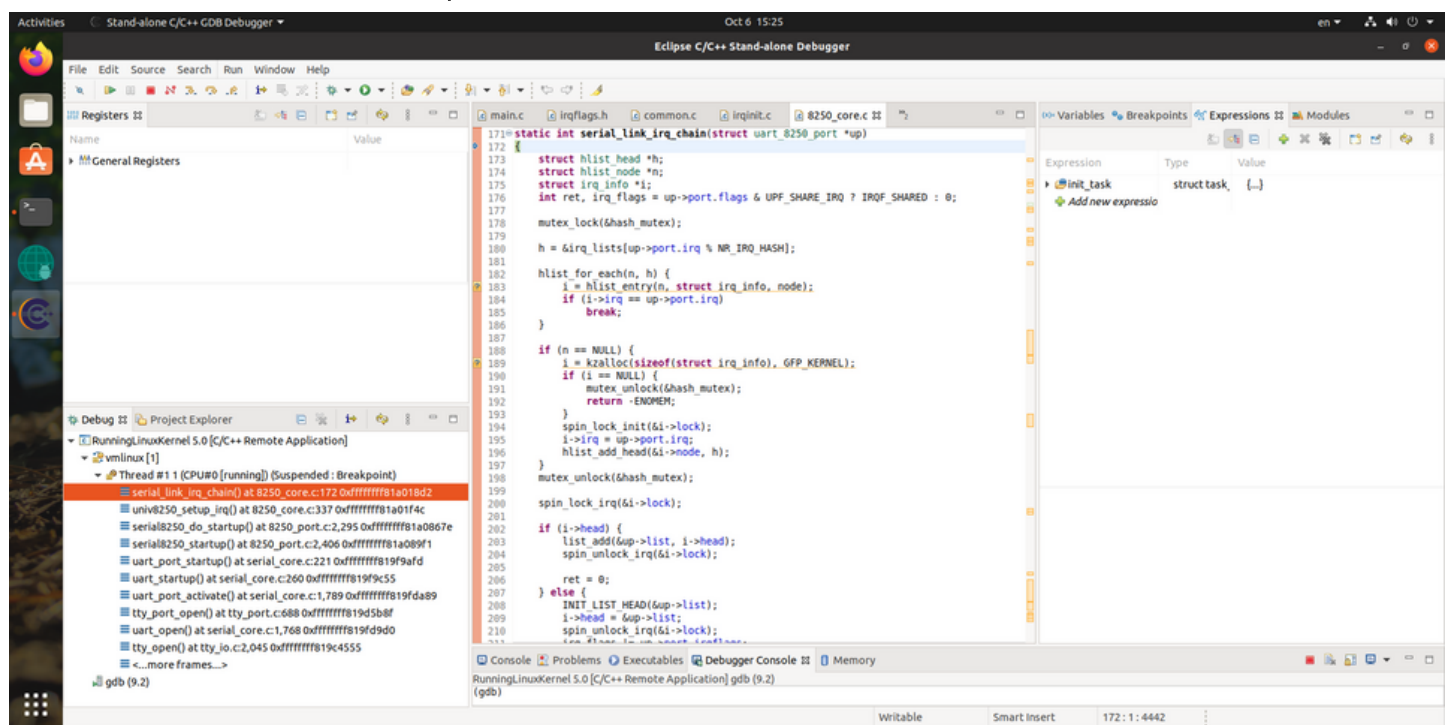
## 2.b init\_IRQ



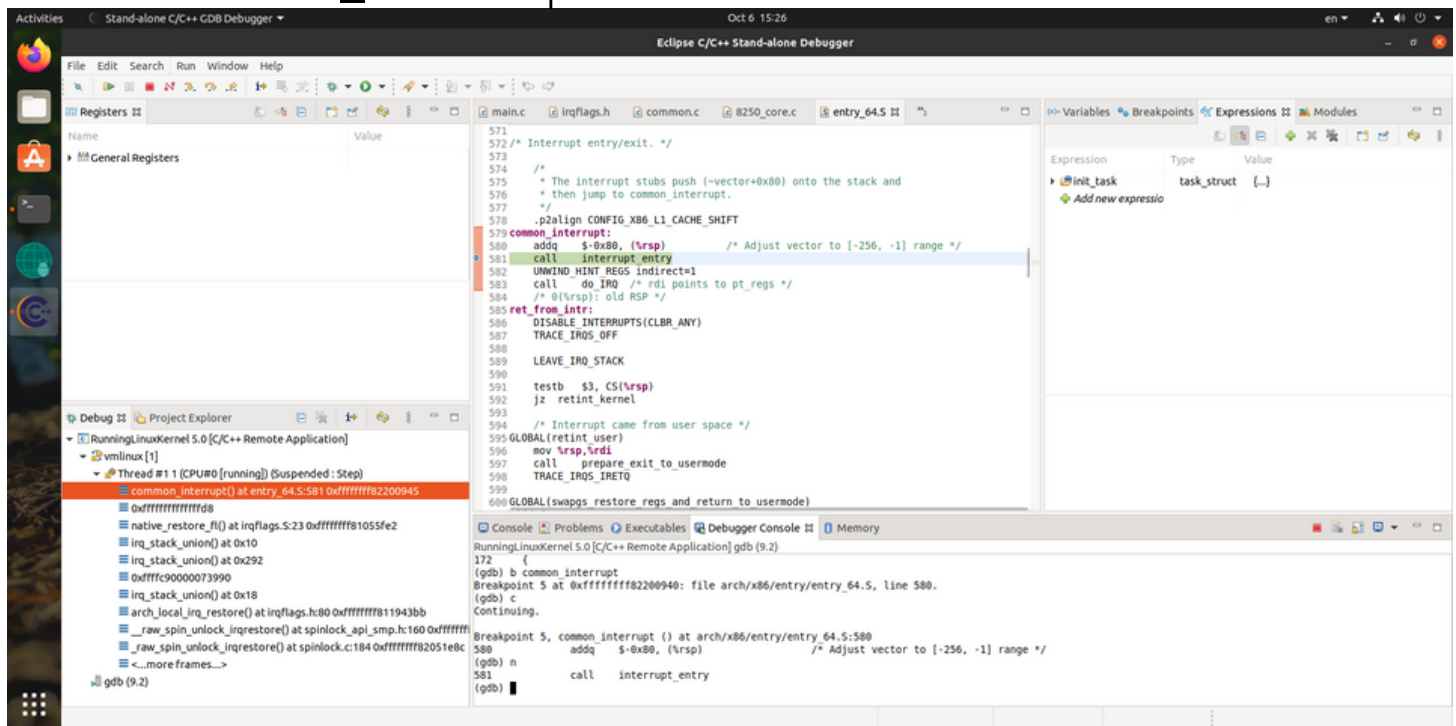
## 3.b native\_init\_IRQ



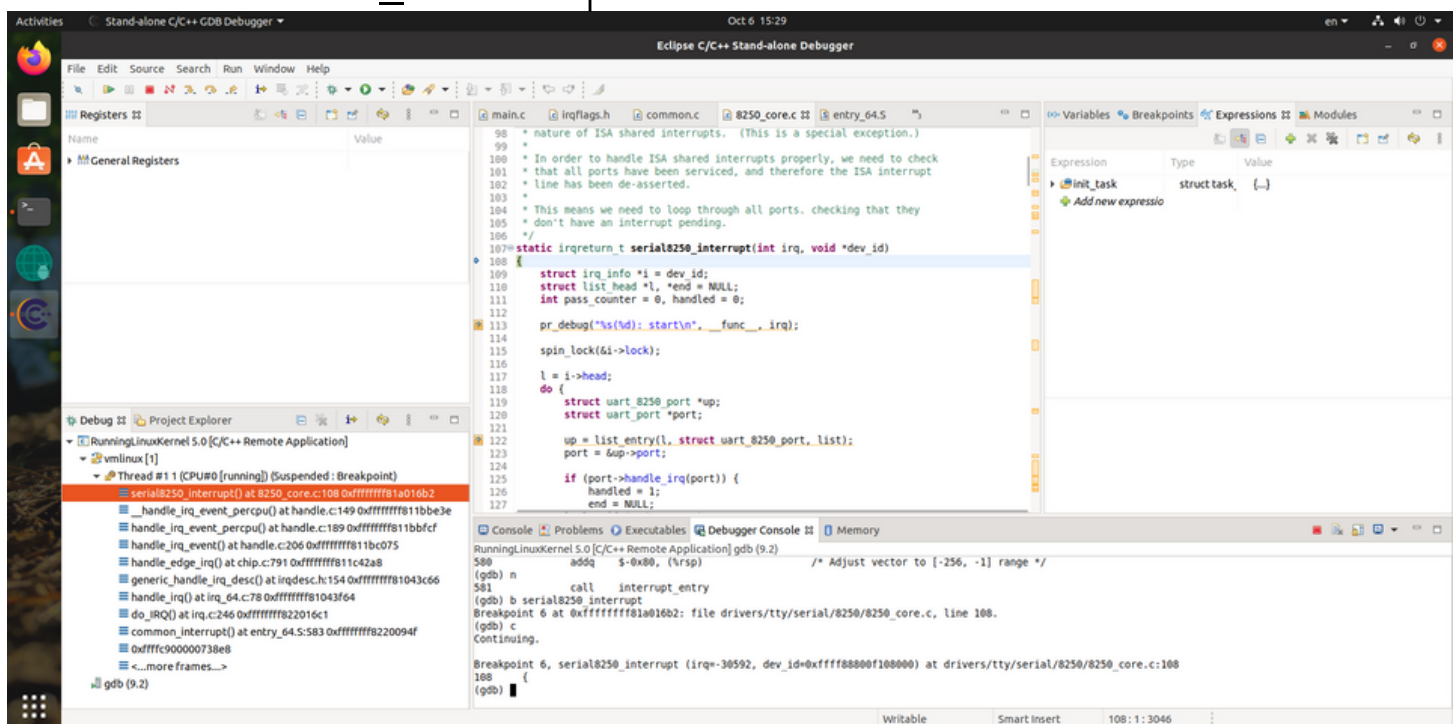
## 4.b serial\_link\_irq\_chain



## 5.b common\_interrupt



## 6.b serial8250\_interrupt



The screenshot displays the Eclipse C/C++ Stand-alone Debugger interface. The top bar indicates the application is running on Oct 6 15:44. The main window is divided into several panes:

- Registers:** Shows the General Registers.
- Debug:** Contains the Project Explorer, showing the running application structure, including Thread #1 [CPU#0 [running]] (Suspended: Breakpoint) and the start\_kernel() function at main.c:538.
- Source Code:** Displays the main.c file, showing the initialization of the kernel, including the setup of the command line and the initialization of the kernel.
- Disassembly:** Shows the assembly code for the irq\_entries\_start function, including instructions like pushq, jmpq, and nop.
- Console:** Displays the GDB output, including the running application name (RunningLinuxKernel 5.0 [C/C++ Remote Application]) and the GDB version (gdb (9.2)).

The screenshot shows the Eclipse C/C++ IDE with the Stand-alone Debugger. The main editor displays the assembly code for 'entry\_64.S'. The left sidebar shows the 'Registers' window with 'General Registers' and the 'Project Explorer' showing the file structure. The bottom panel shows the 'Debugger Console' with GDB output.

**Registers Window:**

Name	Value
General Registers	

**Project Explorer:**

- RunningLinuxKernel 5.0 [C/C++ Remote Application]
  - vmlinux [1]
    - Thread #1 [CPU#0 [running]] (Suspended: Step)
      - irq\_entries\_start() at entry\_64.S:375 0xffffffff8220024a
        - irq\_stack\_union() at 0x58
        - native\_restore\_fli() at irqflags.h:80 0xffffffff811943bb
        - irq\_stack\_union() at 0x10
        - irq\_stack\_union() at 0x292
        - 0xffffc90000073990
        - irq\_stack\_union() at 0x18
        - arch\_local\_irq\_restore() at irqflags.h:80 0xffffffff811943bb
        - \_raw\_spin\_unlock\_irqrestore() at spinlock\_api\_smp.h:160 0xffffffff82051ebc
        - \_raw\_spin\_unlock\_irqrestore() at spinlock.c:184 0xffffffff82051ebc
        - ...more frames...

**Debugger Console:**

```
RunningLinuxKernel 5.0 [C/C++ Remote Application] gdb (9.2)
Breakpoint 2, syscall_init () at arch/x86/kernel/cpu/common.c:1536
1536 {
(gdb) c
Continuing.

Breakpoint 4, 0xffffffff82200248 in irq_entries_start () at arch/x86/entry/entry_64.S:375
375 .endbr
(gdb) ni
0xffffffff8220024a in irq_entries_start () at arch/x86/entry/entry_64.S:375
375 .endbr
(gdb) c
```

**Assembly Code (entry\_64.S):**

```
365 /*
366  * align 8
367  * ENTRY(irq_entries_start)
368  * vector=FIRST_EXTERNAL_VECTOR
369  * .rept (FIRST_SYSTEM_VECTOR - FIRST_EXTERNAL_VECTOR)
370  * UNWIND_HINT_IRET_REGS
371  * pushq $(-vector+0x80) /* Note: always in signed byte range */
372  * jmp common_interrupt
373  * align 8
374  * vector=vector+1
375  * .endbr
376  * END(irq_entries_start)
377  */
378 #macro DEBUG_ENTRY_ASSERT IRQS_OFF
379 #ifndef CONFIG_DEBUG_ENTRY
380 pushq %rax
381 SAVE_FLAGS(CLR_RAX)
382 testl $X86_EFLAGS_IF, %eax
383 jz .Lokay_v@
384 ud2
385 .Lokay_v@:
386 popq %rax
387 #endif
388 .endm
389
390 /*
391  * Enters the IRQ stack if we're not already using it. NMI-safe. Clobbers
392  * %rflags and puts old RSP into old_rsp, and leaves all other GPRs alone.
393  * Requires kernel GSBASE.
394  */
```

**Variables Window:**

Variable	Value
irq_entries_start	0xffffffff82200940 <common_interrupt>



## 問題2

ANS:中斷分為兩部分，第一部分為CPU內建的中斷，又稱「software interrupt」或「trap」或是「exception」。此類型的中斷在設定時Linux在start\_kernel中會先呼叫trap\_init，將內部中斷初始化，然後將CPU內部中斷的中斷處理函數寫入到中斷向量表(IVT)，而在此設定的中斷共有19個，其餘的內部中斷(20~31)會被Intel保留起來，日後做其他的用途。而第二部分為外部中斷，此類中斷在設定時會由init\_IRQ一路呼叫到set\_intr\_gate。

## 問題3.

ANS:以serial port的中斷為例，他是第四號外部中斷，而在irq\_entries\_start中為「0x58」。之後呼叫common\_interrupt後會先將所有暫存器存入堆疊中，其資料型態為pt\_regs，並將serial port在irq\_entries\_start中的編號放入pt\_regs的orig\_ax。之後呼叫do\_IRQ後，就會根據orig\_ax內存放的中斷編號去呼叫相對應的函數，提供服務。