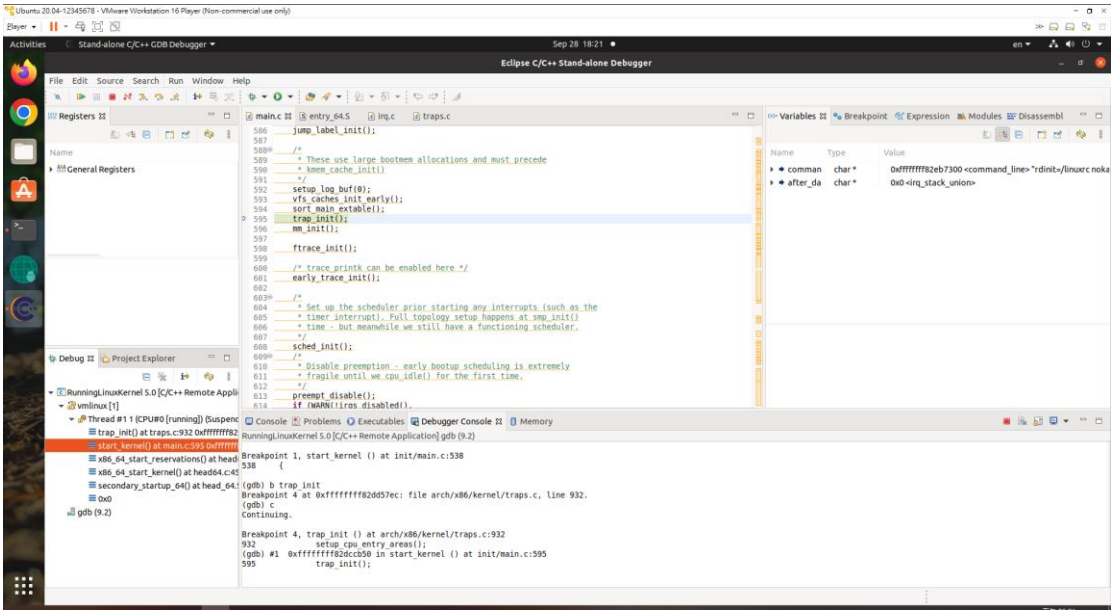
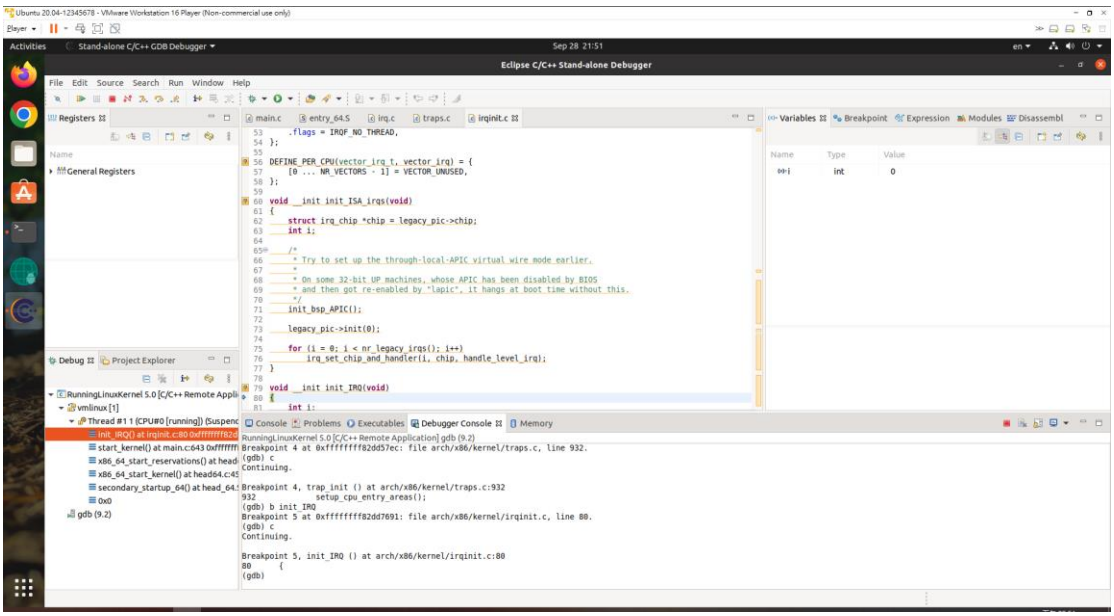
HW2 觀察中斷

系級：資工二　學號：409410114　姓名：周述君

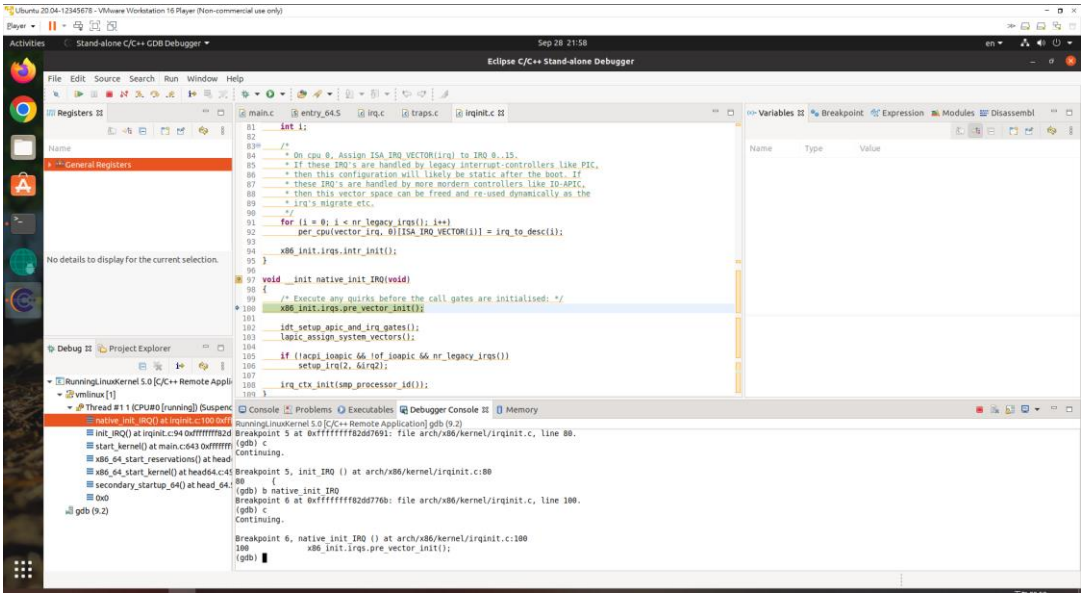
1. 一份簡單的報告，請將題目所說的八個中斷點予以截圖
   - 截圖 1. b trap_init



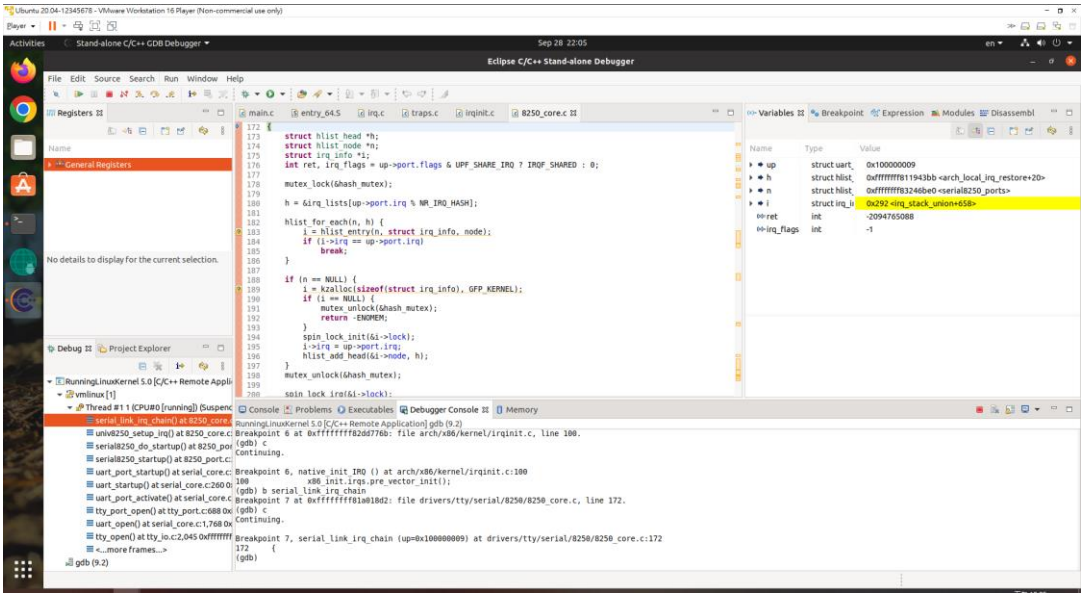
   - 截圖 2. B init_IRQ



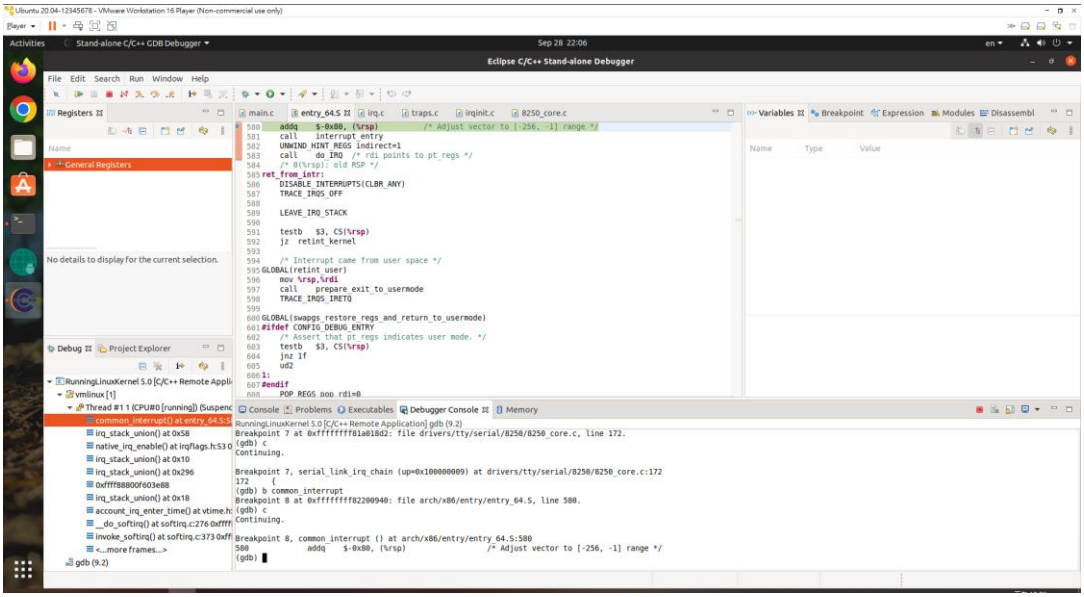
- 截圖 3. B native_init_IR



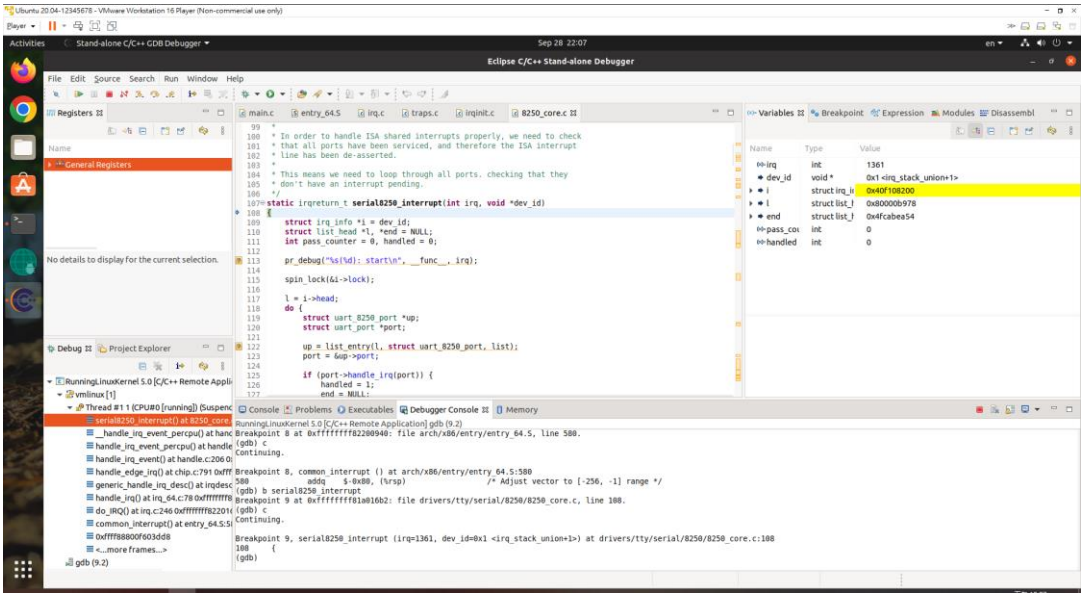
- 截圖 4.b serial_link_irq_chain



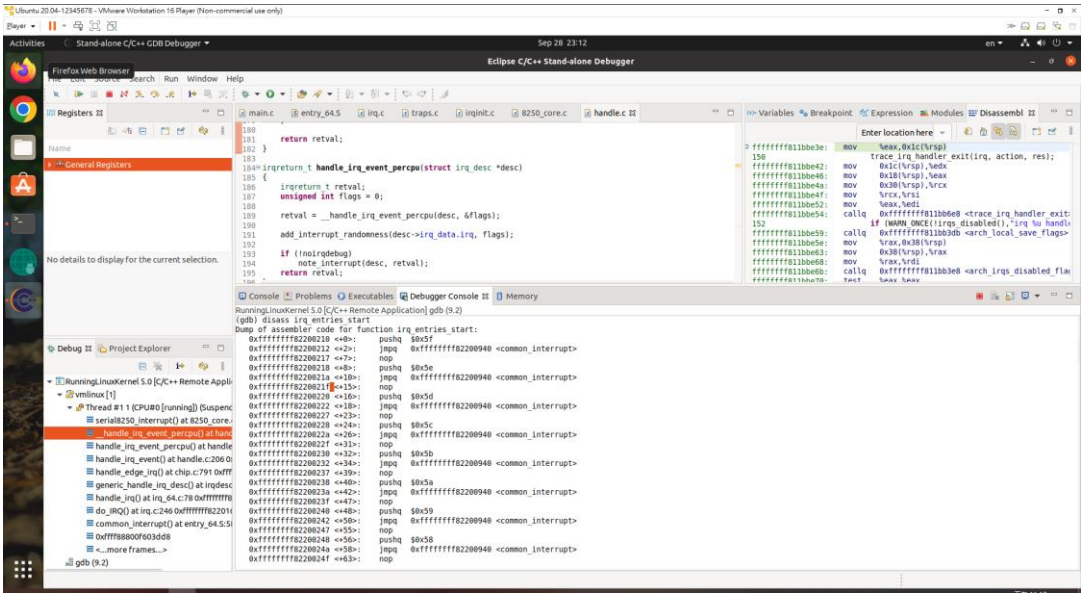
- 截圖 5.b common_interrupt



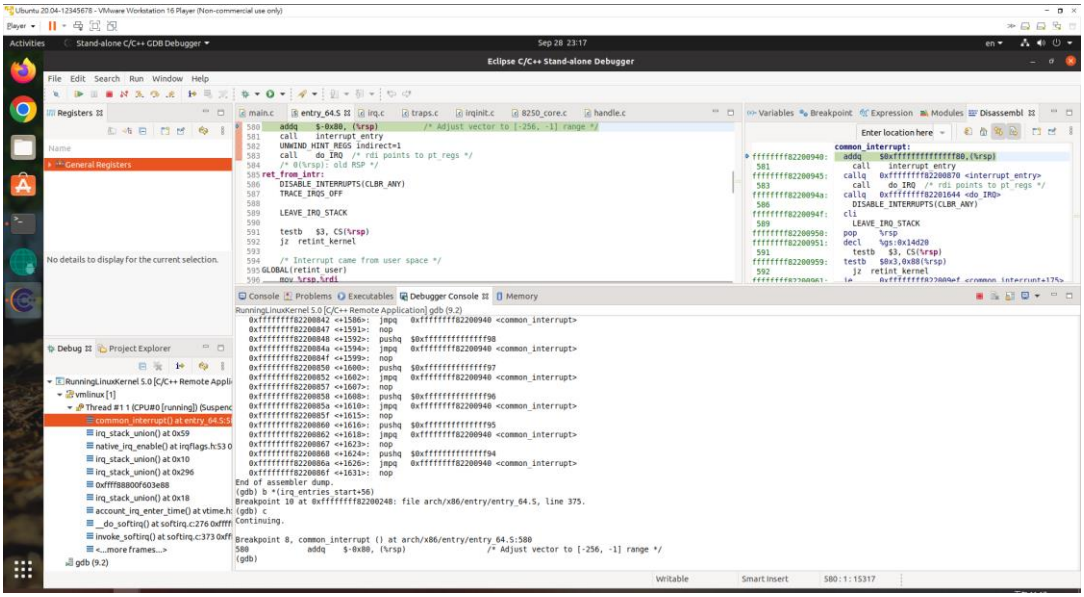
- 截圖 6.b serial8250_interrupt



- 截圖 7.disass irq_entries_start



- 截圖 8.b*(irq_entries_start+56)



2. 在（問題 1.）的報告中，說明 Linux 如何設定中斷向量

Linux 中斷向量表分為內部和外部中斷

CPU 內建的中斷事件又稱 software interrupt 或 trap，linux 在 start_kernel 中會先呼叫 trap_init 來初始化處理器的 trap，再將 CPU 內部中斷的中段處理函數寫入中斷向量表。

外部中斷的部分是由 start_kernel 呼叫 init_IRQ 來初始化 16 個一般外部中斷陣列，再呼叫 x86_init.irqs.intr_init()建立外部中斷向量表，直到呼叫 set_intr_gate 為止。

```
Dump of assembler code for function irq_entries_start:
   0xffffffff82200210 <+0>:     pushq  $0x5f
   0xffffffff82200212 <+2>:     jmpq   0xffffffff82200940 <common_interrupt>
   0xffffffff82200217 <+7>:     nop
   0xffffffff82200218 <+8>:     pushq  $0x5e
   0xffffffff8220021a <+10>:    jmpq   0xffffffff82200940 <common_interrupt>
   0xffffffff8220021f <+15>:    nop
```

3. 在（問題 1.）的報告中，說明 Linux 如何從中斷向量的組合語言部分（interrupt service routine，這裡只討論外部中斷）跳躍到特定的中斷函數
以 serial port 為例，他是第四號外部中斷(從/proc/irq/4 可知)

```
0xffffffff82200248 <+56>:    pushq   $0x58
0xffffffff8220024a <+58>:    jmpq    0xffffffff82200940 <common_interrupt>
0xffffffff8220024f <+63>:    nop
```

特別要注意的是，第四號中斷在程式碼中到底是 irq_entries_start 中的第Ｘ號組合語言並非一對一的對應，例如在這個例子中是「第八號組合語言」。Linux 對這個中斷的「軟體編號」是「0x58」。
由 common_interrupt 的程式碼發現，所有中斷服務的函數都會跳到下面這段組合語言，呼叫 interrupt_entry 將所有暫存器放入堆疊，這部分的重點是「製造堆疊」，堆疊內的資料型態為「pt_regs」，並且將專斷的軟體編號(0x58)放到 pr_regs 的 orig_ax。

```
common_interrupt:
addq  $-0x80, (%rsp)            /* Adjust vector to [-256, -1] range */
call    interrupt_entry
UNWIND_HINT_REGS indirect=1
call    do_IRQ      /* rdi points to pt_regs */
```

問題一的截圖六，如果裝置發出中斷，do_IRQ 會把中斷向量的編號記錄下來，再根據編號往下找函數指標，接著從 do_IRQ 一層一層呼叫到 serial8250_interrupt，由截圖七可看到中斷向量所指向的程式碼。其中 do_IRQ 為 C 語言，do_IRQ 的程式如下，這部分的重點是：由於 orig_ax 放的是「中斷的軟體編號」，因此將這個編號作為「中斷向量物件」的索引，即__this_cpu_read(vector_irq[vector])，並在 handle_irq 中呼叫該函數。在這個例子中的函數即 serial8250_interrupt。約略等同於下列程式碼
desc = __this_cpu_read(vector_irq[vector]);desc->action->handler(...);