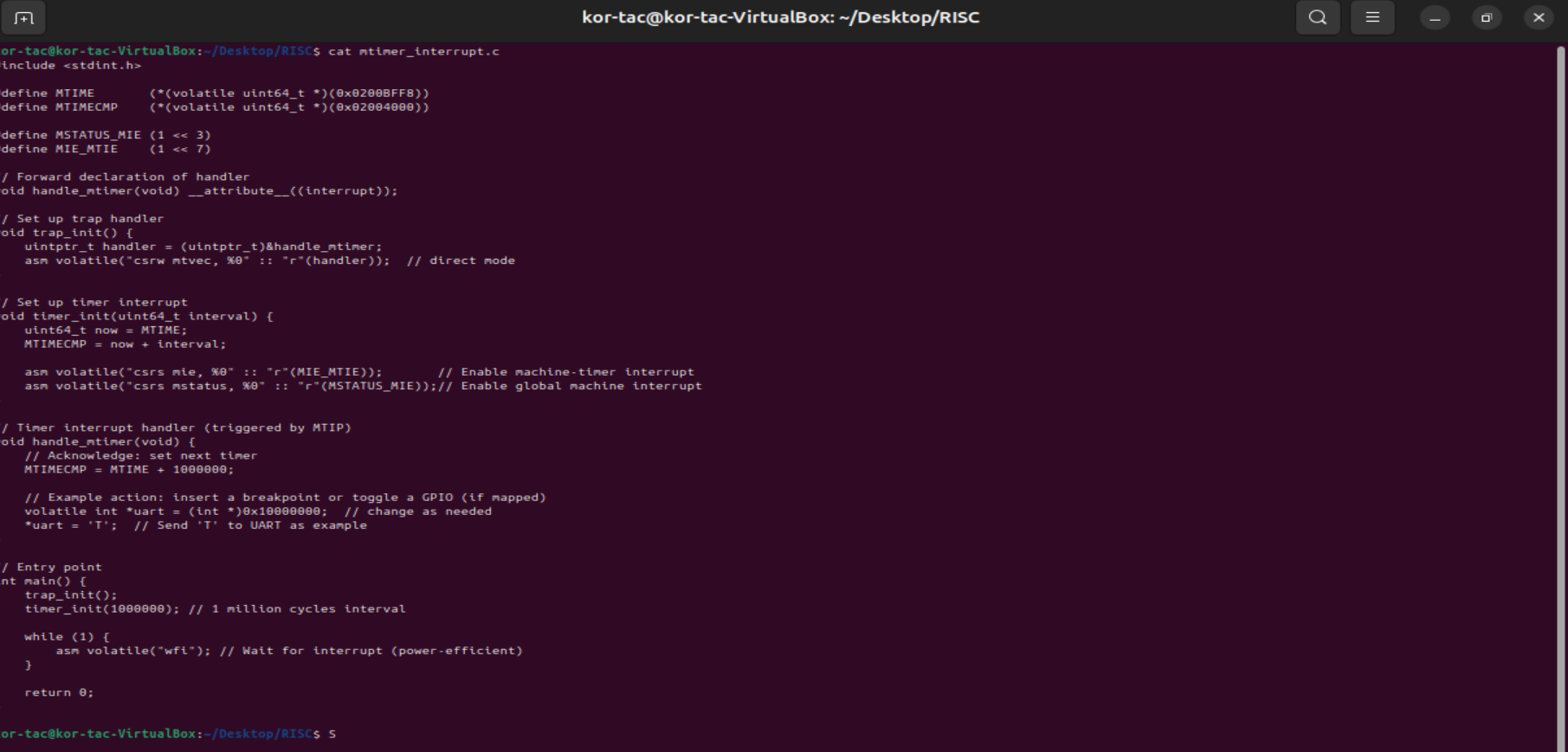
Interrupt Primer

Write the code



#include <stdint.h>

#define MTIME (\*(volatile uint64\_t \*)(0x0200BFF8))

#define MTIMECMP (\*(volatile uint64\_t \*)(0x02004000))

#define MSTATUS\_MIE (1 << 3)

#define MIE\_MTIE (1 << 7)

// Forward declaration of handler

void handle\_mtimer(void) \_\_attribute\_\_((interrupt));

// Set up trap handler

void trap\_init() {

uintptr\_t handler = (uintptr\_t)&handle\_mtimer;

asm volatile("csrw mtvec, %0" :: "r"(handler)); // direct mode

}

// Set up timer interrupt

void timer\_init(uint64\_t interval) {

uint64\_t now = MTIME;

MTIMECMP = now + interval;

asm volatile("csrs mie, %0" :: "r"(MIE\_MTIE)); // Enable machine-timer interrupt

asm volatile("csrs mstatus, %0" :: "r"(MSTATUS\_MIE));// Enable global machine interrupt

}

// Timer interrupt handler (triggered by MTIP)

void handle\_mtimer(void) {

// Acknowledge: set next timer

MTIMECMP = MTIME + 1000000;

// Example action: insert a breakpoint or toggle a GPIO (if mapped)

volatile int \*uart = (int \*)0x10000000; // change as needed

\*uart = 'T'; // Send 'T' to UART as example

}

// Entry point

int main() {

trap\_init();

timer\_init(1000000); // 1 million cycles interval

while (1) {

asm volatile("wfi"); // Wait for interrupt (power-efficient)

}

return 0;

}

